ON THE IMPLEMENTATION OF ALGOL 68

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

by

Sidney Marshall

Thayer School of Engineering
Dartmouth College
Hanover, New Hampshire

June 1972

This research was supported by the
Advanced Research Projects Agency
of the Department of Defense and was
monitored by the Air Force Office of
Scientific Research under Contract
No. F44620-68-C-0015.

Examining Committee:

Chairman

Director of Graduate Study

(SCHOOL) THAYER SCHOOL OF ENGINEERING

DARTMOUTH COLLEGE

(TITLE) On the Implementation of ALGOL 68

by

(NAME) Sidney Marshall

(DEGREE) Doctor of Philosophy

(DATE & YEAR) JUNE 1972

ABSTRACT

This thesis is concerned with implementing a compiler for the computer language ALGOL 68. The compiler contains two passes that are syntax directed followed by a third code generating pass.

Imbedded in the syntax for pass 1 and pass 2 are "actions" that are subroutine calls that perform the actual compilation. All declarations are analyzed in pass 1 and stored in tables for use by pass 2. Pass 2 rereads the source program and generates a modified Polish postfix intermediate code. Pass 2 also determines the proper sequence of "coercions" to apply. These coercions transform one data type into another and an extensive set of coercions is provided by ALGOL 68. Determining the proper sequence of coercions to apply in a particular case is not trivial and an algorithm that determines this sequence is presented.

A garbage collector is described for use in ALGOL 68 programs. This garbage collector collects all ALGOL 68 data types and requires no push-down stack.

The purpose of this thesis was to see if practical compilers for ALGOL 68 could be written. It was found that such a compiler could be written but that there were some language features that could be modified to simplify the compiler writing task.

PREFACE

I wish to express my appreciation to Kiewit computation center and especially Professors Thomas Kurtz and Robert Hargraves for the facilities provided me to complete this thesis.

I would also like to extend special thanks to Professor Robert Hargraves for the many discussions regarding my thesis.

I would like to thank Professor Miles Hayes, the chairman of my dissertation committee, for his interest and encouragement.

Of course, without my wife, Halina, who tolerated my bad moods and encouraged my good modes, this thesis would never have been completed. She deserves more thanks than I could possibly give in this preface.

TABLE OF CONTENTS

INTRODUCTION

Summary

The computer language ALGOL 68 is defined by [Wijngaarden, A. van (Ed.), Mailloux, B.J., Peck, J.E.L., Koster, C.H.A., Numerische Mathematik, 14, 79-218 (1969)]. The purpose of this thesis is to develop a practical compiler for the language.

The structure of the language inherently requires that the compiler be two-pass, since operators and identifiers may be used before their declaration. This compiler is essentially two-pass.

Both passes are syntax-directed. The syntax was derived from the report and is an LR(1) production scheme which is original in this thesis. The semantics for both passes are embedded in the syntax in the form of actions. In pass 1 the semantics (actions) are mainly concerned with constructing the symbol table. In pass 2, the actions generate an intermediate output code, which is a modified Polish postfix form that is easily converted to the desired machine code. The particular intermediate language is original and was devised for this thesis. (In this implementation, pass 3 performs the conversion to HIS-G635 machine code.)

ALGOL-68 provides an elaborate set of coercion rules for converting values from one mode to another. Determining in a specific instance the correct coercions to apply can be

quite complicated. In this thesis, an original and elaborate three part algorithm performs this task, and appears in pass 2 (a posteriori mode routine, coercion setup routine, compile coercion routine).

Also original in this thesis is the garbage collector, a part of both the compile time and run time storage allocation functions. The garbage collector is interesting in that only one bit per word is required for supplementary storage in order to carry out the "collection" process (no stack is required).

Brief description of ALGOL 68

ALGOL 68 is a new language developed from the experience gained from ALGOL 60 and has many similar features [Wigngaarden, A. van (Ed.), Mailloux, B.J., Peck, J.E.L., Koster, C.H.A., "Report on the Algorithmic Language ALGOL 68", Numerische Mathematik, 14, 79-218 (1969)]. The block structure of ALGOL 60 is retained as well as most of the other features. One difference is that statements in ALGOL 60 are generalized into clauses in ALGOL 68 and are defined to have a value. It is therefore possible to enter a new block or range in an arithmetic formula in ALGOL 68 while this can be done in ALGOL 60 only with a procedure.

The 'for' loop in ALGOL 68 is a little more restrictive but allows a more efficient implementation. The value of the running variable or any of the loop parameters cannot be modified, so the loop can be set up once and none of the loop parameters will change.

ALGOL 68 defines all of the input/output conventions including formatted and formatless and binary input/output. Even file opening and closing routines are defined.

One of the major additions in ALGOL 68 is its concept of mode. Instead of a finite list of types for values that can be manipulated by the program, there are an infinite number of types called modes of data that a program could manipulate. These data types or modes are defined in a recursive fashion and include all of the data types found in ALGOL 60. There are also modes specifying higher precision than normal and modes specifying structures, procedures, and unions.

A value whose mode is structured consists of an ordered collection of values of various modes arranged in fields. This collection of values can be manipulated as a unit, or the individual constituent values can be manipulated by specifying the tag associated with the field. By using structured values complex numbers and general list structures are handled by ALGOL 68 in a natural way.

Procedures are values that are actually algorithms or routines. Procedure values may be used as elements of arrays or structures or may invoke the routine which is their value. This invocation may be recursive. Since there is no label mode in ALGOL 68, the procedure mode is used for this purpose. Such a procedure is a routine that causes a jump to the specified label. In this way the environment problems associated with labels are handled by the procedure environment handling methods.

The procedure linkage has been improved in ALGOL 68. Since the mode of all values is known at compile time and the mode of all formal parameters is also known, all necessary mode converting code can be compiled and no checks at run time need to be performed. The concept of call by name in ALGOL 60 is not used in ALGOL 68, but any formal parameter can be declared to be a procedure. This causes the actual parameter to be turned into a procedure (if it is not already one) which is executed each time the procedure calls on it. This carrys out the effect of the ALGOL 60 call by name.

One important feature of ALGOL 68 is that the modes of all run time values are known at compile time. However, it is sometimes desirable to determine modes at run time. A united mode is a mode that represents a value that can be one of a list of modes. At run time tests can be made to determine which mode a united value really is. This test must be performed before using the value so the mode of all values at run time is still known at compile time.

New operators can be declared in ALGOL 68 so vector calculations can be written as compactly as numerical calculations. Operators are similar to procedures with one or two formal parameters.

ALGOL 68 allows the programmer to generate very general list structures, but the programmer is given no way to free a particular structure. Instead, a garbage collector is used to free all memory that the program can no longer use. This makes it impossible for the programmer to inadvertantly free a structure that will be used again.

Arrays in ALGOL 68 are similar to those in ALGOL 60 and allow arrays with elements of any mode. Arrays of structures or procedures are allowed. By slicing an array either an element of the array or a subarray can be obtained.

ALGOL 68 contains several constructions designed to make the language more convenient to use. There is a 'case' clause that is similar to a conditional clause except that an integer selects one of several clauses to be executed. The value of a case clause is the value of the clause executed.

ALGOL 68 is careful to distinguish between the concepts of declaring an identifier and allocating memory. In ALGOL 60 declarations accomplish both, and these two functions are inseparable. In ALGOL 68 declarations assign meaning to an identifier but do not allocate space for a value. Generators allocate memory but are not associated with any identifier. Of course a declaration can associate a generator with an identifier but this is not necessary.

Generators can be used to generate new elements of a list structure whose access is through other elements of the list structure.

BRIEF DESCRIPTION OF THE COMPILER

General overview

The compiler is divided into several sections. There is the input preprocessor that reads the source program character by character and combines groups of characters into useful syntactic units. This is the only routine that manipulates source program characters. Both pass 1 and pass 2 use the input preprocessor to read the source program.

Pass 1 reads the entire source program using the input preprocessor and constructs tables containing all declarations in the program. Pass 1 also constructs a table with entries for every left parenthesis, vertical bar, and left bracket in the program. This table will be used by pass 2 to enable it to choose the proper syntax for a closed clause.

At the end of pass 1 several of the tables constructed during pass 1 are cleaned up. Mode indications are replaced by the modes to which they refer in the mode table and duplicate modes are combined. Some definitions in the DEF table are spurious because of pass 1's limited context sensitivity and these are deleted. The length of values of all modes is calculated and stored in the mode table. Patterns for all modes are also generated.

Pass 2 then rereads the entire source program via the input preprocessor and generates intermediate compiled code in table CODE. This intermediate code resembles Polish code and contains pointers to the definition tables so that the following pass is not concerned with any analysis. Pass 2 contains all of the coercion routines that determine all coercions to be applied to coercends and their order.

Both pass 1 and pass 2 are syntax directed by a syntax written by the author. The syntax used for pass 1 and pass 2 are in the appendix together with a description of all the actions contained in the syntax.

Pass 3 reads the intermediate code generated in pass 2 and compiles machine code. At the end of the compiled code pass 3 adds loader information which contains patterns for all values used by the program and all external symbol definitions and references. Pass 3 also contains a mode cannonizer that generates cannonical forms for modes so that identical modes in different programs can be recognized by the loader with a simple compare operation.

The loader then loads all necessary library programs, binds them together, and executes the resulting code. When the program exits the job is terminated.

Source program scanner

Since it would be cumbersome to use a syntax analyzer to analyze each character of a program, a small input preprocessor is used to buffer the analyzer from the source program. It combines one to several characters of the source program into a single syntactic unit which it presents to the syntax analyzer. Thus, the analyzer can deal with units such as identifiers or numbers without having to construct these units out of characters. The preprocessor operates as a finite state machine with its next state determined from the current state and the current source character. On the basis of its state the preprocessor can cause the source character to be added to an internal string, skip a source character, back up the source string one or two characters, or look up its internal string in a symbol table and give a pointer to the entry to the syntax analyzer to analyze.

Every time the preprocessor is called it returns a pointer to an entry in the symbol table (STAB). This entry contains a pointer to the string that was matched and a pointer to a chain of definitions for the symbol. Routines that call the preprocessor never have to deal with source program characters as the pointer to the symbol table entry contains all the information about the symbol.

The syntax analyzer

The syntax analyzer is a small routine that analyzes the source program and calls the various routines necessary to compile it. It does this by attempting to match the source program to a production of 'PROG'. This is done by matching the source program against the successive alternatives of 'PROG'. As each alternative is composed of names of other productions, the algorithm is recursive. The syntax consists of a set of production rules. Each production rule has a name and a list of alternatives. Each alternative is a list of elements. An element can be the name of a production rule or an action. All actions are subroutines that return either 'OK' or 'FAIL'. The action of the syntax analyzer can be described as a program with the following steps:

1. Initialize S to be a pointer to the first element of the first alternative of the production rule 'PROG'.

2. If S points to an action then transfer to it. If the action returns 'OK' then go to step 4. If the action returns 'FAIL' then go to step 6.

3. If S points to the name of a production rule then push the pointer S onto the control stack, make S point to the first element of the first alternative of the production rule pointed to by S, and go to step 2.

4. OK: If there are no more elements in the alternative of the production pointed to by S then pop the top of the control stack into S and repeat this step.

5. Step pointer S so that it points to the next element in the current alternative and go to step 2.

6. FAIL: If the pointer S does not point to the first element of an alternative then the syntax analyzer

fails and a terminal error message is printed.

7. If there is another alternative after the alternative pointed to by S then make S point to the first element of the next alternative and go to step 2.

8. Pop the top of the control stack into S and go to step 6.

There are two types of actions used in the syntax. One type of action is a subroutine that performs some function of the compiler and always returns with 'OK'. The other type of action is a match action and examines the next syntactic unit from the input preprocessor to see if it matches a syntactic unit specified by the match action. If it does not match, the match action returns with 'FAIL' otherwise the input preprocessor is advanced one syntactic unit and the action returns with 'OK'.

Since the analyzer fails if a 'FAIL' return occurs anywhere except at the first element of an alternative the syntax is arranged with match type actions as the first element of each alternative except possibly the last. The other elements of an alternative are then made up of actions that usually return 'OK' or names of production rules.

The syntax can be considered as a program with the first element of an alternative being a conditional statement and the rest of the elements of the alternative being a sequence of statements to be executed. Actions correspond to normal statements, and names of production rules correspond to recursive subroutine calls. Since the control stack is stacked and unstacked as production rules are entered and exited, actions can use the control stack for temporary storage provided that no such storage is left on the control stack at the end of an alternative. Such storage corresponds to local storage of a recursive procedure. Another stack, called the working stack, is provided for temporary storage that does not appear and disappear as production rules are entered and exited. The allocation in this stack is under the sole control of the actions. It is mainly used to construct lists whose length is not known initially. This is accomplished by storing a pointer to the current top of the working stack in the control stack. The elements of the list are then successively pushed on the working stack. The pointer to the old top of the working stack is retrieved and all of the words added after this point are elements of the list. Structured and united declarers are constructed in this manner.

Tables used by the compiler

There are several tables that are constructed and used by the compiler. Since the size of these tables is unknown at the start of compilation, these tables must be allocated dynamically. There is a table allocation routine that will allocate any specified number of words at the end of any table. To do this it may have to move adjacent tables to make room. Consequently, all program references to any table must be relative to the base of the table as the table may be moved at any time.

Two of the tables are actually stacks and the allocator maintains both a table control word and a top of stack pointer for these tables. For the other tables, the allocator maintains only a table control word. A table control word consists of a pointer to the base of the table and its current length.

The tables are used to store information accumulated during each pass and pass it on to the next pass. Since the compiler can be described in terms of how it constructs, changes, and references the tables, a short description of each table is now given:

WORK (working stack)

This is a stack which is used by actions during passes 1 and 2 and by some of the cleanup routines in pass 1.5 to store temporary information. Its stacking and unstacking is controlled by the actions.

STACK (control stack)

This stack is used by the syntax analyzer to remember the state of the surrounding parse. It is also used as temporary storage by some actions during pass 1 and 2 and by some of the cleanup routines in pass 1.5.

MODE

This table contains entries that represent modes. In final form all modes are pointers to an entry in this table and all constituent modes of a mode are also represented as pointers to this table.

BOUND

This table contains entries that represent bound information of array declarers. Declarers refer to both a mode and a bound table entry.

DEF

Entries in this table represent declarations in the source program. Entries in this table are chained together and represent all possible definitions for a given identifier or indicant. Entries representing declarations in the same range are also chained together.

PROG

This table records the parenthesis structure of the source program. It is used by pass 2 to anticipate to a limited degree the structure of the source program. Every left parenthesis, left bracket, and vertical bar in the source program creates an entry in this table.

STAB

This is the symbol table. Each entry contains a pointer to the external character representation of the symbol and a pointer to its definition chain in the DEF table.

ITAB

This is the identifier table and contains the character strings of all symbols in the source program.

CODE

The intermediate code generated during pass 2 is placed in this table.

LBL

Internally generated labels are integers. During pass 3 this table is used to store the address of a label and a pointer to a chain of addresses that should have this label as a value.

GEN

The machine language object code and associated loader information are put in this table during pass 3.

TYPE

A template for every value to be used by the object program is stored in this table. These templates will form part of the loader information generated at the end of pass 3.

ZZZ

This table contains instruction sequences for standard prelude operators. It is used by pass 3 in generating code for the standard operators.

SDEF

This table contains a list of symbols defined by the program. This table is mainly used when creating a new library procedure.

An advantage of organizing all dynamic memory together is that no individual table will run out of space unless there is no space for any table. There is also no problem of guessing how much space should be allocated to a given table. A possible disadvantage is that table references are slower because all references are relative to a base word requiring extra additions and subtractions to change relative table references to absolute references and vice versa.

PASS 1

It is the job of pass 1 to build the MODE, BOUND, DEF, and PROG tables. Every time a left parenthesis, vertical bar, or a left bracket is encountered in the source program an entry is created in the PROG table. Encountering a left parenthesis or a left bracket causes the routine SRNGE (start range) to be entered. This routine stores several words in the control stack including a pointer to the entry just created. When the matching right parenthesis or bracket is encountered these saved words will be restored from the control stack by the routine ERNGE (end range). A vertical bar is treated as a combination right parenthesis and left parenthesis. ERNGE also stores in the table entry the number of commas, colons, and semicolons since the last left parenthesis, left bracket, or bar not contained in any nested set and a flag indicating whether or not the clause is a procedure denotation. The PROG table is used by pass 2 to determine whether a left parenthesis is the start of a procedure denotation, parallel clause, serial clause, etc.

Declarations can only occur after a semicolon, left parenthesis, vertical bar, or comma following another declaration. There are four types of declarations: mode, priority, operation, and identity. Mode declarations have three forms:

```
MODE X =

STRUCT X =
```

UNION X =

These forms can be recognized by the use of one of the reserved words 'MODE', 'STRUCT', or 'UNION' followed by an unreserved word. When one of the above forms is encountered, the mode following the equal sign is evaluated and an entry in the DEF table is created defining X as a mode indication whose definition is the mode and bound just evaluated.

Priority declarations have only one form:

PRIORITY X =

This form can be recognized by the presence of the reserved word 'PRIORITY'. When this form is recognized an entry is made in the DEF table defining X to have a priority equal to the digit following the equal sign.

Operation declarations have two forms:

OP X = <procedure denotation>

OP (<mode sequence>)<mode or empty> =

Both of these forms can be recognized by the presence of the reserved word 'OP'. The two forms can be distinguished by whether or not the symbol following 'OP' is a left parenthesis. If it is the second form then an entry is made in the DEF table defining X to be an operator with the procedure mode specified by the symbols between the 'OP' and the '='. If it is the first form then the entry is made in the DEF table defining X to be an operator but no mode specified. After the mode of the procedure denotation is known then this procedure mode is inserted in the table entry.

Identity declarations come in several forms:

```
<mode> X
<mode> X =
<mode> X :=
PROC X = <procedure denotation>
PROC X := <procedure denotation>
```

where <mode> is any mode declarer. Unfortunately it is impossible in Pass 1 to distinguish identifiers, mode indications, and operators from each other making it impossible to identify <mode>. This problem is circumvented by treating anything that could possibly be a <mode> as one. This can result in some spurious declarations but these will be discovered in pass 1.5 and deleted. <mode> can only start with one of the following constructions:

```
[
REF
STRUCT (
UNION (
PROC
<unreserved word>
```

If it is any construction but the last two then the mode can be evaluated with no problem. In the last case an entry is made in the MODE and BOUND tables defining the unreserved word to be a mode indication. An entry can now be made in the DEF table defining X to be an identifier whose mode and bound have just been calculated. If an equal sign does not immediately follow X then the mode has a reference inserted in front of it.

When 'PROC' is the first symbol then the situation is complicated. If the 'PROC' is followed by a left parenthesis then it is a mode and can be handled like the other modes. When there is a construction like "PROC X =", then if a procedure denotation follows the 'PROC', it should be considered to have the same mode as the procedure denotation; otherwise, its mode is procedure void. There is explicit syntax to handle all of the cases involving the symbol 'PROC'. Also, if a closed clause is encountered that is not a procedure denotation, it is arbitrarily assigned the mode procedure void as an aid to this syntax.

Labels are declared as follows: if the construction "X:" is encountered outside of a pair of brackets, X is declared to be a label. Label declarations are treated as identity declarations whose mode is the pseudo mode M$LBL.

Except for potential declarations and constructions entered in the PROG table pass 1 ignores all other symbols in the source program. There is a production rule for 'JUNK' that skips all sequences of operators, identifiers, and modes, and this production rule is used whenever a declaration would be impossible.

It is possible in a sequence of declarations to omit the reserved word or mode in the second and following declarations if it is the same as in the first declaration. For example:

```
PRIORITY + = 6, - = 6;
```

is the same as:

```
PRIORITY + = 6, PRIORITY - = 6;
```

After processing a declaration a check is made to see if the declaration is followed by a sequence like ", X =". If it is then these symbols are processed as another declaration. Otherwise, the production rule for DEC is used.

At the end of pass 1 all modes, priorities, operators, and identifiers have been defined and for each an entry in the DEF table has been made with the definition and the range for which the definition is valid. Since every entry in the PROG table contains a pointer to another entry in the PROG table for the surrounding range, the proper definition of any symbol can be found by looking for a definition in the current range and working out from there.

PASS 1.5

After the declaration tables have been constructed in pass 1 it is necessary to clean them up before pass 2 uses them. This "cleanup" is accomplished by a set of routines called "pass 1.5" which is not really a pass at all.

Since pass 1 puts an entry in the DEF table for anything resembling a declaration all spurious declarations in the DEF table must be removed. Mode, priority, and operation declarations can be accurately recognized during pass 1 and are correct. The problem arises with constructions like "X Y" where 'X' may be a mode indication and "X Y" a declaration declaring 'Y' to be an identifier of mode 'X' or 'REF X'. 'X' might also be a unary operator in which case 'Y' is not declared to be anything. To discover which case is true the routine looks up 'X' in the symbol table. If 'X' is a mode then the declaration is good. If 'X' is a priority or an operator then the declaration is spurious. If 'X' is an identifier then the declaration is spurious if the declaration for 'X' is not spurious. In the last case the routine calls itself recursively. If the declaration for 'X' is not spurious then the declaration for 'Y' is spurious. Otherwise, the spurious declaration for 'X' is deleted and another definition for 'X' is sought. When all of the identifier definitions have been checked all spurious declarations have been eliminated. While this routine is checking for spurious declarations it also checks for symbols that are multiply defined in the same range. If two definitions for the same symbol are found in the same range, a message is printed and one of the definitions is deleted.

Once the DEF table is corrected, the MODE and BOUND tables can be corrected. This routine finds every entry in the MODE or BOUND tables that is a mode indication. These entries should be replaced by the definition of the mode indication. When a mode indication entry is found the indication is looked up in the DEF table and its definition discovered. The entry in the MODE or BOUND table is then replaced by an 'XFER' entry that has a pointer to the true mode or bound. When this routine is completed all mode indications have been replaced by an 'XFER' entry pointing to an equivalent entry.

At this point the MODE table is correct although a single mode may be represented by several entries in the MODE table. It is necessary in pass 2 to be able to tell if two modes are equivalent. This routine replaces all but one of the entries referring to the same mode by 'XFER' entries. An 'XFER' entry contains a pointer to another mode table entry. A pointer to a mode table entry is made 'unique' by examining the table entry to which it points. If the table entry is an 'XFER' entry then the original pointer is replaced by the pointer found in the mode table entry. This process is repeated until the entry pointed to by the mode pointer is not of type 'XFER'. Two pointers refer to the same mode if they are both made unique and then refer to the same mode table entry. The routine that changes selected entries in the mode table to type 'XFER' is based on an algorithm by C. H. A. Koster who makes the observation that two modes are equivalent if they cannot be proved to be dissimilar. The algorithm works as follows:

1. If the two modes are of different types e.g., STRUCT and UNION, then the modes are not equivalent.
2. If the two modes are of the same type and length (two STRUCTs with the same number of fields) then the two modes are postulated to be equivalent and the two modes are equivalent if and only if all of the constituent modes and tags are equivalent or postulated to be equivalent.
3. If two modes are equivalent then all postulates made to show the equivalence are true.

This equivalence algorithm is applied to every pair of modes in the MODE table. If the two modes are discovered to be equivalent then one of the modes is replaced by an 'XFER' entry pointing to the other mode. When this routine is completed there is a unique entry associated with every mode of the source program. This algorithm must terminate because each step postulates two modes to be the same.

Since this can be done only a finite number of times before all given modes are postulated equivalent, the algorithm must terminate.

The next routine scans the DEF table for all label definitions. It assigns a unique integer to each label.

The next routine scans the mode table and enters in every valid entry the length of a value of the mode and a pointer to a pattern (stored in the TYPE table) for the value. The routine operates according to the following principles:

1. All primitive modes have their lengths and types predefined.

2. All procedure modes have a length of 4 and a pattern pointer 'PROCT'.

3. All structure modes have a length that is the sum of the lengths of its fields and a pattern that is the concatenation of the patterns for its fields.

4. All united modes have a length one more than the length of the longest constituent mode and a pattern of two words the first of which is 'UNT1' and the second is minus the length of the longest constituent mode.

5. Reference modes that do not refer to row modes have a length of 1 and a pattern of 'PTR'. Reference modes that do refer to row modes have the same length and pattern as the row mode to which it refers.

6. Row modes have a length of 4n + 1 where n is the

number of dimensions of the row mode and a pattern of n + 1 words. The first word of the pattern is 'PTR' and all the rest are 'QUAD'.

The next routine rewinds the source file and prepares the input preprocessor to reread the source program.

This is the end of pass 1.5.

PASS 2

Pass 2 rereads the source program and, with the help of tables constructed in pass 1, generates a Polish-like output code that completely describes the computation indicated by the source program. There are several constructions that pass 2 must recognize in the source program and each will be discussed separately. The constructions are:

- Identifiers
- Labels
- Primitive denotations
- Slices
- Calls
- Selections
- Generators
- Monadic formulas
- Dyadic formulas
- Assignations
- Conformity relations
- Identity relations
- Casts
- Serial clauses
- Conditional clauses
- Case clauses
- Conformity case clauses
- Parallel clauses
- Procedure denotations

Declarers

Declarations

Each of the above constructions produces appropriate Polish code so pass 3 does not have to make any further reference to the source program. The coercion process and the operator identification routine are major parts of pass 2 and are described separately in their own section.

During pass 2 the control stack is used for temporary storage during the interpretation of a syntax alternative. The working stack is used during the compilation of declarers to store partially constructed declarers in the same manner as pass 1. The main use of the working stack is to store five word blocks that describe a value in the run time stack. The format of a five word value control block is:

[type flag, number of parallel values]

[mode of value]

[location and length of code calculating value]

[lexicographical level bit word]

[-5, 0]

The value control blocks stored in the working stack during pass 2 mirror the values that will appear in the local run time stack when the object program is executed.

Identifiers

Identifiers cause a 'O$IDENT' code word with a pointer to the DEF table for the identifier to be added to the intermediate code. A value control block is pushed onto the working stack containing the mode of the identifier.

Labels

A label is compiled by adding a 'O$LBL' code to the intermediate output with a parameter that is unique. References to the label will use the same unique number.

Primitive denotations

Primitive denotations are all denotations except procedure denotations. This includes integral, real, and string denotations. The character string representing the denotation is converted to an appropriate internal representation (as converting a number to floating point) and an entry in the DEF table is created for the denotation containing the internal representation. Then a 'O$DENOT' code is added to the intermediate code with a pointer to the newly created DEF table entry. A value control block with the mode of the denotation is pushed onto the control stack.

Slices

A slice is a primary followed by a left bracket followed by a list of indexers separated by commas followed by a right bracket as "A[B, I : J AT K]". A slice is compiled by first compiling its primary and weakly coercing it to a row or reference to row mode. Then a 'O$SUB' code is added to the intermediate code followed by the code generated by compiling the indexers in sequence followed by a 'O$BUS' code with the mode of the resulting slice.

The code generated by each of the indexers depends on the type of the indexer. If the indexer is empty (i.e., contains no bounds) a 'O$VEPTY' code with the position number of the indexer is added to the intermediate code. If the indexer is a subscript then the subscript is compiled and strongly coerced to integral and a 'O$VSBCT' code with the index position added afterwards to the intermediate code. The value control block for the coerced subscript is then deleted from the working stack. If the index position is a trimmer then each bound in the trimmer is compiled and strongly coerced to integral. Then the intermediate code for each bound is followed by 'O$VLWB' for a lower bound, 'O$VUPB' for an upper bound, and 'O$VNLWB' for a new lower bound. The value control block for the bound is then deleted from the working stack.

For example, the slice:

```
          A[B,,:C,D:E AT F]
```

compiles as:

```
          [primary A]
          SUB              Beginning of subscript expression
```

```
[tertiary B]
VSBCT 1        First indexer is subscript
VEPTY 2        Second indexer is empty
[tertiary C]
VUPB  3        Third indexer contains upper bound
[tertiary D]
VLWB  4        Fourth index contains lower bound
[tertiary E]
VUPB  4        Fourth index contains upper bound
[tertiary F]
VNLWB 4        Fourth index has new lower bound
BUS   mode     Mode of slice
```

Pass 3 actually calculates all of the bounds before starting any indexing operations.

Calls

Calls consist of a primary followed by a list of units surrounded by parentheses. The primary is supposed to be a procedure and the units are its actual parameters. A call is compiled by compiling its primary and firmly coercing it to a procedure mode. A 'O$MSCW' code is then added to the intermediate code. The list of units surrounded by parentheses is compiled as a parallel clause which is a structure display. This display is then strongly coerced to a structured mode having the modes of its fields the same as the modes of the parameters of the procedure. Both the value control block for the display and the procedure are deleted from the working stack. A value control block having the mode of the result of the procedure is then pushed onto the working stack and a 'O$ENTER' code with the result mode is added to the intermediate code.

Selections

A selection consists of a tag followed by the symbol 'OF' followed by a secondary. A pointer to the tag symbol in the STAB table is stored in the control stack. The secondary is then compiled and weakly coerced. The tag is recovered from the control stack and the mode of the coerced secondary is examined. The mode must be either struct(...) or ref struct(...). If it is not there is a source program error. The structured mode is searched for a field whose tag matches the given tag. If no such field is found it is a source program error. If the mode of the coerced secondary is struct(...) then a 'O$SELCT' code is added to the intermediate code. Otherwise a 'O$RSLCT' code is added. The field number of the desired field is also added to the code. The top value control block in the working stack which corresponds to the coerced secondary has its mode changed to the mode of the selected field. If the original mode was a referenced mode then the new mode is changed to a reference to the mode of the field. The code 'O$ETC' with the mode of the value control block is added to the intermediate output.

Generators

Generators appear in the source program as declarers. When a declarer in the source program is recognized by the syntax as a generator, a value control block is pushed into the working stack with a mode of the declarer. A code word 'O$LGEN' for a local generator or 'O$HGEN' for a heap generator containing the mode of the generator is added to the intermediate code. Then a 'O$BOUND' code with a pointer to the BOUND table entry of the declarer is added to the intermediate code.

Monadic formulas

A monadic formula is an operator followed by a secondary. A pointer to the STAB table entry for the operator is stored in the control stack. The secondary is then compiled. The pointer to the STAB table entry for the operator is retrieved from the control stack and the operator identification routines are called.

Dyadic formulas

In the case of dyadic formulas the syntax does not parse the source program into proper components. To parse a program correctly would require the operator syntax to be repeated ten times - one for every priority level possible for an operator. It would also require matching "priority five operator" which is not convenient. Instead the syntax parses dyadic formulas from left to right and lets the actions compile the proper code to group operands in the proper sequence. Whenever the syntax recognizes a dyadic operator it obtains the priority of the operator and calls the operator identification routine to compile any previous higher priority dyadic operators. Then a value control block of type 'W$OP' is pushed onto the working stack. This value control block refers to the previous 'W$OP' value control block if any and contains the priority of the operator and a pointer to the operator's STAB table entry.

Whenever the end of a dyadic formula is encountered in the source program all dyadic operators in the working stack are compiled by assuming a following priority zero operator and calling the operator identification routine. This assumed operator is never stored in the working stack.

Assignations

An assignation is a tertiary followed by a 'becomes' symbol (':=') followed by a unit. The tertiary is compiled and softly coerced. The mode of the coerced tertiary is remembered in the control stack. The unit is then compiled and strongly coerced to a mode that is the mode remembered in the control stack dereferenced once. The top two value control blocks in the working stack are combined into one and a 'ASGN' code is inserted in front of all code for the assignation and a 'ASGNE' is added after all code. Each code has the mode of the assignation associated with it. The mode of the assignation remembered in the control stack is then deleted.

Conformity relations

Conformity relations allow the programmer to discover the current mode of the value of a united value and make this contained value available to the programmer. In the conformity relation:

```
          A ::= B
```

The tertiary B is to be evaluated first. If the current mode of the value of B can be assigned to A (if the mode is corrcct) then it is assigned to A and the value of the conformity relation is true. Otherwise no assignation takes place and the value is false. Notice that the tertiary B is evaluated before the tertiary A and that A might not be evaluated. This backwards elaboration requires some branching back and forth to make the elaboration order correct. If the '::=' symbol is replaced by a '::' symbol the meaning is the same except that the assignation never takes place and the left tertiary is never evaluated. The intermediate code generated for the above code is:

```
          TRA    #1        Generated label
```

```
MREF            Allocate space
LBL    #2       Generated label
CONF  mode      Mode of tertiary
TF     #3       Generated label
MREF            Allocate space
[tertiary A]
CASGN ref mode  Do assignation
TRUE            Get true value
TRA    #4       Generated label
LBL    #1       Generated label
MAX             Protect temporary memory
[tertiary B]
CONE            Save pointer to B
TRA    #2       Generated label
LBL    #3       Generated label
FALSE           Get false value
LBL    #4       Generated label
```

Conformity relations are unique in that they require a run time execution different from the order in the source program.

Identity relations

An identity relation consists of a tertiary followed by either a ':=:' or a ':/=:' followed by another tertiary. The two tertiaries are compiled. A value control block of type 'W$BAL' with a count of two is then pushed onto the working stack. The coercion routine is called to calculate the a posteriori mode of a soft coercion applied to the top value control block in the working stack. This is the mode to which both tertiaries are to be coerced. Each tertiary is then coerced separately to this mode. The working stack now contains three value control blocks from the identity relation: two value control blocks for the two tertiaries and the 'W$BAL' value control block. These three value control blocks are combined into one block and the mode is set to boolean. For example, the identity relation

```
          A :=: B
```

compiles as:

```
          [tertiary A]
          [tertiary B]
          IS
```

Casts

Casts consist of a virtual mode declarer followed by a colon followed by a unitary clause and surrounded by parentheses. The unitary clause of the cast is compiled and then strongly coerced to the mode specified by the declarer.

Serial clauses

Serial clauses are made up of declarations, labels, and clause trains. All of the declarations must occur before the first label or end of a clause train. A clause train is made up of a sequence of units. A serial clause is compiled by compiling its constituents in sequence. Whenever a unit that is not the last unit of a clause train is compiled the value of the unit is strongly coerced to void and all value control blocks associated with it are deleted from the working stack. When a serial clause is completely compiled, the working stack contains value control blocks arising from the final units of the clause trains. A value control block is now pushed onto the working stack of type 'W$BAL' that contains a count of the number of clause trains in the serial clause. When the serial clause is coerced the collection of value control blocks will be replaced by one value control block for the serial clause.

Conditional clauses

A conditional clause has the form:

(A \ B \ C)

where A, B, and C are serial clauses. The serial clause A is compiled and strongly coerced to boolean. A transfer false code is generated that transfers to a generated label. All value control blocks associated with the serial clause A are deleted from the working stack. The serial clause B is then compiled. It is followed by a code to unconditionally transfer to a second generated label and a code to define the first generated label. The serial clause C is then compiled followed by the defining of the second generated label. A value control block of type 'W$BAL' with a count of two is then pushed onto the working stack so that the coercion routines will coerce B and C simultaneously. The resulting intermediate code is as follows:

```
[serial clause A]
TF    #1       Generated label
[serial clause B]
LBL   #1       Generated label
[serial clause C]
```

If a conditional clause has its last clause missing then an else clause consisting of 'SKIP' is assumed and the conditional clause is compiled in the normal manner.

Conditional clauses can be extended by using 'thenf' or 'elsf' symbols ('\:'). These extensions are handled in an identical manner as the basic conditional clause. In the case of an 'elsf' symbol as in

(A \ B \: C \ D \ E)

the serial clauses C, D, and E are a conditional clause that is the else clause of another conditional clause. Both conditional clauses are compiled in the same way. In the case of the 'thenf' symbol as in

(A \: B \ C \ D)

the serial clauses B, C, and D form a conditional clause that is the then clause of another conditional clause. The

B, C, and D clauses are compiled into a conditional clause in the normal manner but the outer clause is missing an else part. A 'SKIP' is compiled for the else part then the outer conditional clause is compiled in the normal manner.

Case clauses

A case clause is similar to a conditional clause except that the clause to be executed is selected by an integral value rather than a boolean value. An example of a case clause is:

(A \ B , C , D \ E)

A and E are serial clauses and B, C, and D are unitary clauses. The serial clause A is compiled and strongly coerced to integral. A 'CASE' code with an argument of n followed by n + 1 'TRA' codes are then generated where n is the number of unitary clauses separated by commas in the case clause. All value control blocks associated with the A clause are then deleted from the working stack. The unitary clauses B, C, and D are then compiled with labels inserted in front of each unitary clause. A value control block of type 'W$BAL' with a count of n is then pushed onto the working stack. Then the serial clause E is compiled and a label inserted in front of this compiled code. A value control block of type 'W$BAL' with a count of two is then pushed onto the working stack.

The resulting intermediate code has the following form:

```
[serial clause A]
CASE   3          Number of unitary clauses
TRA    #4         Generated label
TRA    #1         Generated label
TRA    #2         Generated label
TRA    #3         Generated label
LBL    #1         Generated label
[unitary clause B]
LBL    #2         Generated label
[unitary clause C]
LBL    #3         Generated label
[unitary clause D]
LBL    #4         Generated label
[serial clause E]
```

If the last serial clause is missing then a SKIP is assumed and compilation proceeds normally.

Conformity case clauses

A conformity case clause is a case clause except the selection of the clause to be executed is determined by a set of conformity relations. The format of a conditional case clause is:

```
( A , B , C ::= D \ E , F , G \ H )
```

This is equivalent to the following clause:

```
( A ::= D \ E
\: B ::= D \ F
\: C ::= D \ G
\ H )
```

This example should execute by first evaluating the tertiary D and then checking whether or not it conforms to A or B or C and doing appropriate assignments and transfering to the appropriate clause E, F, G, or H. The intermediate code generated for the above example is:

```
TRA   #1       Generated label
MREF           Allocate space
LBL   #2       Generated label
CONF  mode     Mode of tertiary
TF    #3       Generated label
MREF           Allocate space
[tertiary A]
CASGN ref mode Do conformity
TRA   #6       Generated label
MREF           Allocate space
LBL   #3       Generated label
CONF  mode     Mode of tertiary
TF    #4       Generated label
MREF           Allocate space
[tertiary B]
CASGN ref mode Do conformity
TRA   #7       Generated label
MREF           Allocate space
LBL   #4       Generated label
```

```
CONF  mode      Mode of tertiary
TF    #5        Generated label
MREF            Allocate space
[tertiary C]
CASGN ref mode  Do conformity
TRA   #8        Generated label
MREF            Allocate space
LBL   #5        Generated label
TRA   #9        Generated label
LBL   #1        Generated label
MAX             Protect temporary memory
[tertiary D]
CONE            Save pointer to D
TRA   #2        Generated label
LBL   #6        Generated label
[tertiary E]
TRA   #10       Generated label
LBL   #7        Generated label
[tertiary F]
TRA   #10       Generated label
LBL   #8        Generated label
[tertiary G]
LBL   #10       Generated label
TRA   #11       Generated label
LBL   #9        Generated label
[tertiary H]
```

LBL #11 Generated label

If the last clause is missing a SKIP is assumed and compilation proceeds normally.

Parallel clauses

Parallel clauses consist of a list of unitary clauses separated by commas and surrounded by parentheses. Parallel clauses can be used as either row displays or structure displays. In either case code is generated for each of the constituent unitary clauses and a set of value control blocks is pushed onto the working stack for each of the unitary clauses. When all of the unitary clauses have been compiled a 'W$PAR' type of value control block is pushed onto the working stack that contains a count of the number of constituent unitary clauses in the parallel clause. This collection of value control blocks will be combined into a single value control block by the coercion routines.

The compiler considers the actual parameters pack in a procedure call to be a parallel clause. This allows the coercion required for the actual parameters to be done by the same coercion routines as do the structure displays.

Procedure denotations

Procedure denotations consist of a list of formal parameters surrounded by parentheses followed by a possible mode and a colon and followed by its body. Each formal parameter causes a 'O$FORMP' code to be generated with a pointer to the DEF table entry of the formal parameter as its argument. The modes of the formal parameters are successively stored in the working stack. After the formal parameters are scanned the mode of the result is scanned and also stored in the working stack. The list of modes of the formal parameters in the working stack and the mode of the result are considered to represent a procedure mode whose parameter modes are the same as the modes stored in the working stack and having the same result mode. This mode is found in the mode table and a pointer to the table entry in the mode table representing this mode is stored on the control stack. The list of modes representing the formal parameters are then deleted from the working stack. The body of the procedure is compiled causing value control blocks to be stored in the working stack representing its value. The mode of the body is recovered from the mode of the procedure denotation stored in the control stack and the body is strongly coerced to this mode. Then the mode of the procedure is recovered from the control stack and the routine PCDR is called to make a procedure value out of the body. For example, the procedure denotation:

```
(REAL X) REAL : X
```

generates the following Polish code:

```
TRA   #1      Generated label
```

```
EPDN   #2          Generated label
LL     L           LL of scope of procedure
SRNGE  R           Range of procedure denotation
FORMP  X           Formal parameter
IDENT  X           Body of procedure
ERNGE  R           Range of procedure denotation
RETN   REAL        Mode of result
LLE    L1          LL of surrounding range
LBL    #1          Generated label
EPDV   PROC(REAL)REAL Mode of procedure
EPDE   #2          Generated label
```

The scope of a procedure is defined as the smallest scope of any of the constituents of the procedure that are not local. In order to calculate the scope of a procedure every value in the working stack contains a bit word indicating which nonlocal identifiers, operators, labels, or declarers were used in calculating the value. Each bit position corresponds to the level difference between the current level and the level of the identifier etc. Whenever a range is exited, all of the bit words in the value control blocks belonging to the exited range are shifted left one position. This makes the bit words correct for the external range. The scope of a procedure is calculated as the range corresponding to the first bit that is on in the bit word of the value control block for the procedure body.

Declarers

A declarer is generally used to specify a mode. The ALGOL 68 report defines three types of declarers: virtual, actual, and formal. Virtual declarers are used only to specify a mode. This mode is used only at compile time and generates no object code directly. Actual declarers appear in mode declarations and in generators. Mode declarations associate a source program symbol with a declarer for use in other declarers. Generators cause the allocation of storage at run time to contain a value of the specified mode. The bounds of all constituent arrays must also be specified and this is the major difference between virtual and actual declarers. The run time form of an actual declarer is a set of procedures that define the bounds and states of all constituent arrays in the declarer. (Formal declarers in the current implementation are treated as virtual declarers. If they were not they would compile as bound and state checking procedures. These procedures would have no effect except the production of terminal error messages when a bound or state was wrong.)

Since the definition of mode is recursive the evaluation of a declarer is also recursive. During the analysis of a declarer the working stack is used to hold the partial development of the declarer. In the case of an actual declarer intermediate code is generated for each set of bound tertiaries enclosed in brackets as follows. When the left bracket is encountered a transfer around the following code is generated followed by three codes defining the start of a procedure denotation. Each tertiary that is a bound is compiled and strongly coerced to an integral mode. The value control block for the coerced tertiary is deleted from the working stack and the intermediate code for the tertiary is followed by a 'O$LWB' or 'O$UPB' with dimension number for lower bound or upper bound respectively and a 'O$FIX' or 'O$FLEX' code with dimension number for a fixed or flexible bound respectively. When all bounds have been compiled and the right bracket is encountered a 'O$DBUS' code with the mode of an element is added to the intermediate code. Then the codes 'O$RETN', 'O$LLE', 'O$DLEN', and 'O$LBL' are added to the intermediate code. Since a procedure value for the procedure denotation compiled for arrays in declarers does not exist, calling such a procedure is slightly different. Two labels defined by the procedure denotation are stored in the PROG table for the pseudo-range between brackets and these labels are the starting address of the procedure and a location containing the amount of static temporary storage the procedure needs. With this information the caller can construct a procedure value and use it in the normal way. For example, the declarer:

```
          [A : B FLEX, C FLEX : D] REAL
```

causes the following intermediate code for the array part of the declarer to be generated:

```
TRA    #3       Generated label
EPDN   #1       Generated label
LL     range    Range defined by brackets
DSUB            Start procedure denotation
[tertiary A]
LWB    1        First lower bound
FIX    1        Bound is fixed
[tertiary B]
UPB    1        First upper bound
FLEX   1        Bound is flexible
[tertiary C]
LWB    2        Second lower bound
FLEX   2        Bound is flexible
[tertiary D]
UPB    2        Second upper bound
FIX    2        Bound is fixed
DBUS   mode     Mode of element
RETN   void     Value of result is void
LLE    range    Range of procedure environment
DLEN   #2       Generated label
LBL    #3       Generated label
```

Pass 3 will define the label of the 'O$DLEN' code to be a storage location containing the required length of static temporary storage needed by the procedure. If a declarer is a structure containing two arrays then two procedures are compiled.

Declarations

There are four kinds of declarations: mode declarations, priority declarations, operation declarations, and identity declarations.

Mode declarations are compiled by compiling the actual declarer of the declaration and ignoring the rest of the declaration. All table entries for the declaration were created during pass 1 so pass 2 can skip entering the mode in tables.

Priority declarations define the priority of an operator. They result in no compiled code and as pass 1 has created the appropriate table entry for the declaration, pass 2 can ignore the entire declaration.

Operation declarations associate a procedure with an operator. Operation declarations are compiled exactly as are identity declarations by assuming that the operator is an identifier with the mode of the procedure associated with it.

Identity declarations associate a value of a particular mode with an identifier. There is an extension to the language that permits constructions like

```
REF REAL X = LOC REAL
```

to be shortened to

```
REAL X
```

but the syntax treats both forms in an identical manner and both forms result in exactly the same intermediate code.

An identity declaration consists of a formal declarer (but this implementation requires a virtual declarer) followed by an identifier followed by an equals symbol followed by unit. The mode of the declarer is evaluated and saved. A pointer to the DEF table entry for the identifier is saved. The unit is compiled and strongly coerced to the mode specified by the declarer. A 'O$IDNTY' code with a pointer to the identifier's DEF table entry is inserted in the intermediate code before the code for the coerced unit and a 'O$IDNTE' code is added after the code for the coerced unit.

The operator identification process

In ALGOL 68 several declarations for the same operator may be valid at the same time. It is up to the compiler to determine which of the definitions is to be used at each occurence of an operator. Conceptually, the process is simple. If the operands of an operator can be firmly coerced to the modes required by a declaration then that declaration is the one to be used. Otherwise, other definitions must be tried. Operators declared in the innermost range are tried first and operators declared in successively larger enclosing ranges are tried if the preceeding operators are inappropriate. Of course no coercions are actually applied to the operands until it is known which definition for the operator will be used.

In the present compiler, operators declared in the standard prelude are applied even though strong coercions are necessary for its operands. This is done to reduce the number of declarations needed in the standard prelude for operators such as '+' which is defined for all combinations of the modes integer, real, and complex. This allows using the definition for a (REAL, REAL) REAL '+' when one of the operands is real and the other is integral. The integral operand can be strongly coerced to the mode real before applying the operator. The ability to use strong coercions on the operands greatly reduces the number of operators that need to be defined.

If the operator is identified as a user defined operator then a standard procedure call to the operator procedure with its operands as formal parameters is compiled. Otherwise, if the operator is identified as a standard prelude operator, code for the operands is compiled followed by a 'O$OPE' command with a pointer to the STAB table entry for the operator as a parameter. Pass 3 will generate inline code instead of a procedure call in this case.

The coercion process

One of the distinguishing characteristics of ALGOL 68 is its coercion process. Most computer languages have a set of informal rules for automatically changing one type of value to another. For example, a procedure without parameters is called if it appears in an arithmetic formula and the result of the procedure call is used. In ALGOL 68 there is an infinite number of possible automatic mode changing operations and these have been formalized in the coercion process. There are eight coercions: deproceduring, dereferencing, proceduring, uniting, widening, rowing, hipping, and voiding. Each coercion is capable of converting a value of one class of modes into a value of a related class of modes. Not all coercions are allowed everywhere in the program. In some places all coercions are allowed while in others only some of the coercions are allowed. Every syntactic position in which a value can be specified is given a "strength" indicating which coercions are allowed. Strong positions (actual parameters in procedure calls, the right hand side of an assignation, the first serial clause in a conditional or case clause, subscripts) allow all of the coercions to be used. Firm positions (operands in formulas, the procedure in a procedure call) allow only deproceduring, dereferencing, proceduring, and uniting coercions. Weak positions (the array value in a slice, the secondary in a selection) allow only deproceduring and dereferencing. Soft positions (the left hand side of an assignation, one of the sides of an identity relation) allow only deproceduring.

The coercion process is further complicated by "balancing". If a conditional clause can return one of two values then both values must have the same mode. However coercion may be used to make the modes the same. Furthermore, while only coercions allowed by the strength of the position of the conditional clause may be applied to one of the values, the other value is in a strong position and any coercion can be used. It is not obvious which value is strong so it is not obvious which value might determine the mode of the conditional clause. A similar problem arises in row displays where one of the elements in the display may be firm and determine the mode of the row value and all of the other elements may be strong.

The coercion routines are divided into several parts: routines to determine the a posteriori mode in the coercion process, routines to construct the actual coercion sequence, and routines to use the coercion sequence list and actually perform the coercion. By separating the coercion routines in this way the individual routines can be simplified. For strong coercions the a posteriori mode is known so the coercion list can be set up and coercions performed. If the coercion process fails then the source program is in error. There are two types of firm positions: operands and primaries in procedure calls. In the case of operands the a posteriori mode is discovered by the operator identification routine. In the case of a primary in a procedure call the a posteriori mode is found by the a posteriori mode routine. Weak positions are values that are sliced or selected and the a posteriori mode is discovered by the a posteriori mode routine. Soft positions also use the a posteriori mode routine to discover the a posteriori mode but in the case of an identity relation the two sides are balanced before the coercion routines are called. A short description of the coercion routines follows.

<u>The a posteriori mode routine</u>

The a posteriori mode routine requires a pointer to where in the working stack the value control block for the value to be coerced is stored and the strength of the coercion (firm, weak, or soft). It returns the a posteriori mode for the coercion process. Basically the discovery of the a posteriori mode is simple. It is complicated by the possibility of balanced expressions where any of the constituent values may determine the a posteriori mode. The algorithm used to determine the a posteriori mode is as follows:

1. Set the depth counter to zero and set a pointer to the value control block to be coerced and call it the current value control block. Also clear the target mode and zero out the saved mode count.

2. If the current value control block is of type 'W$PAR' then there is a source program error.

3. If the type of the current value control block is

of type 'W$SKIP', 'W$NIL', or 'W$VAC' or if the mode of the current value control block is 'M$LBL' then go to step 30.

4. If the current value control block is not of type 'W$BAL' then go to step 7.

5. Increment the depth counter by the count in the value control block. This is a count of the number of balanced values.

6. Go to step 30.

7. Clear target mode, zero mode count, clear temp 1 and temp 2.

8. Get the mode of the current value control block and call it the current mode. Deprocedure this mode as many times as possible.

9. If the resulting current mode is not reference to something then go to step 14.

10. Store current mode (which is a reference mode) in temp 1.

11. Store current mode in temp 2.

12. Dereference or deprocedure the current mode (whichever is appropriate) and increment the mode count by 1.

13. If the current mode can be dereferenced or deprocedured then go to step 11.

14. If it is not a soft coercion then go to step 21.

15. If temp 1 is clear then it is an error.

16. If the target is clear then go to step 18.

17. If the mode count is larger than the saved mode count then go to step 30.

18. Store the mode saved in temp 1 in target mode.

19. Store the mode count in the saved mode count.

20. Go to step 30.

21. If the target mode is not clear then go to step 28.

22. If this is a firm coercion then go to step 26.

23. If temp 2 is clear then go to step 26.

24. If temp 2 does not contain a reference mode then go to step 26.

25. Store the contents of temp 2 in target mode and go to step 30.

26. Store current mode in target mode.

27. Go to step 30.

28. If target mode is not a reference mode then go to step 30.

29. Store current mode in target mode.

30. Step the pointer to the current value control block back one block deeper in the working stack.

31. Decrement the depth counter by one.

32. If the depth counter is still positive then go to step 2.

33. If the target mode is clear then there is a source program error otherwise return the target mode as

the a posteriori mode.

The coercion setup routine

The coercion setup routine constructs a detailed list of coercion instructions in the control stack for coercing the value specified by a given value control block in the working stack (the a priori mode) to a given a posteriori mode. The algorithm is recursive and handles balanced expressions and displayed values. As the algorithm is rather complicated a simplified explanation is given first.

Of the eight coercions only dereferencing and deproceduring make a mode "simpler" while all of the others make a mode "more complicated". Therefore, a coercion sequence will start with deproceduring and dereferencing and end with other coercions. The basic scheme is to deprocedure and dereference the a priori and a posteriori modes as much as possible and check if the results are the same. If so, a coercion sequence can be constructed by coercing the a priori mode to its reduced mode and then reversing the coercion sequence from the a posteriori mode to its reduced mode. Of course referencing is not allowed and deproceduring followed by proceduring and dereferencing followed by referencing must be deleted from the sequence. If the two reduced modes are not the same then the reduced mode of the a priori mode must be united, widened, or rowed. What is actually done is to "unwiden", "unrow", or "ununite" the reduced mode of the a posteriori mode and again find the reduced mode of the resulting mode. (In the case of ununiting all of the alternative modes must be tried in succession until successful.) If the a priori and a posteriori reduced modes ever match then a coercion sequence can be constructed. In the case of a skip, nil, vacuum, or label the coercion can be deduced immediately from the a posteriori mode.

Balancing is trivial if the a posteriori mode is known. The component values (which may also be balanced) are individually coerced to the a posteriori mode and balance instruction is added to the coercion sequence. If any component value cannot be coerced to the a posteriori mode then the entire balancing fails. This failure may cause the coercion setup routine to try another alternative in a united mode or the entire coercion process may fail.

Row or structure displays are coerced by finding the base mode of the a posteriori mode. If it is a row mode then the component values of the parallel expression are coerced to the mode of an element of the row mode. If the base mode is a structure then the component modes are coerced to the modes of the fields of the structured mode. If the base mode is a united mode then each alternative of the united mode is tried until the parallel expression can be coerced.

One complicating factor is that the strength of the coercion can change when a united mode is encountered. This is handled by a flag indicating whether or not strong coercions are allowed.

The complete coercion setup algorithm is:

1. Save return address; assume VALP points to value control block to be coerced, BMODE is a posteriori mode.

2. Set CF and FF to -1; these flags mean no strong coercions performed yet and strong coercions are allowed respectively.

3. If the current value control block has the type 'W$BAL', 'W$PAR', 'W$SKIP', 'W$NIL', 'W$VAC', or the mode of the value control block is 'M$LBL' then go to the BAL, PAR, SKIP, NIL, VAC, or LBL routine respectively.

4. Push a zero in the working stack.

5. Push into the working stack the coercion sequence required to reduce the mode of the current value control block to its reduced mode. Set AMODE equal to this reduced mode.

6. Push a pointer to the current top of the control stack in the working stack. Set a count in this pushed word to minus one.

7. Push into the control stack the coercion sequence required to reduce the mode in BMODE to its reduced mode. Then set BMODE to this reduced mode.

8. If AMODE does not contain the same mode as BMODE

then go to step 16.

9. Delete words from the working stack until a word with a count of minus one is deleted. This word was pushed in the working stack in step 6.

10. If the top word of the working stack equals the top word of the control stack and the words are not zero then delete both words from their respective stacks and repeat this step (10).

11. If the top of the working stack is a REF coercion and the top coercions in the control stack are a series of ROW or ROWE coercions preceeded by a ref coercion then delete all of these words and push a REFRW (reference row) coercion onto the control stack.

12. Remove the top word from the working stack. If this word is a zero then continue on to step 13. Otherwise, the word is either a REF or PROC coercion. Change it to a DEREF or DEPR (deprocedure) coercion and push it onto the control stack and repeat this step (12).

13. Push onto the control stack a VALP command with a pointer to the current value control block pointed to by VALP.

14. Decrement the value control block pointer so that it points to the previous value control block in the working stack.

15. Return successfully. The control stack contains the complete coercion sequence for coercing the

given value control block to the a posteriori mode.

16. If BMODE contains the void mode and strong coercions are allowed (FF $\neq$ 0) then go to the VOID routine.

17. If BMODE contains either a primitive mode or a row mode and strong coercions are allowed (FF $\neq$ 0) then go to step 23.

18. If BMODE does not contain a united mode then go to step 33.

19. If AMODE does not contain a united mode then go to step 24.

20. If there is any mode from which the mode in AMODE is united that is not among the modes from which the mode in BMODE is united then go to step 33.

21. Push into the control stack the coercion UNION with the mode contained in BMODE.

22. Go to step 9.

23. Set CF to zero to indicate strong coercions are being used.

24. Push into the control stack a coercion command that is the type of the mode contained in BMODE with the mode contained in BMODE.

25. Push into the working stack a pointer to the current top of the control stack with a count of zero.

26. Remove top word in working stack which is a pointer to the control stack and a count. If the count

is minus one then go to step 35.

27. Using the pointer to the control stack popped from the working stack in step 26 get the mode that was stored on the control stack when the pointer was stored in the working stack.

28. Increment the count in the word popped from the working stack. If the count now exceeds the number of modes contained in the mode recovered from the control stack then go to step 35.

29. Push back into the working stack the pointer to the control stack with an incremented count.

30. Store the mode selected by the incremented count and the control stack mode in BMODE. This is either a component mode of a united mode, element mode of a rowed mode, or a mode that can be widened.

31. If the mode in BMODE was selected from a united mode then set FF equal to zero (allow only firm coercions). Otherwise set FF not equal to zero.

32. Go to step 7.

33. Remove the top word in the working stack which is a pointer to the control stack and successively delete words from the control stack until it is the same length as when it was marked. If any words were deleted from the control stack set BMODE to the mode contained in the last word deleted.

34. Go to step 26.

35. Delete words from the working stack until a zero word is deleted. This will delete all words stored in the working stack by the coercion routine.

36. Return with failure. There is no coercion that can be applied to the given value to give a value with the given a posteriori mode.

VOID ROUTINE

V1. Delete words from the working stack until a word with a count of minus one is deleted.

V2. Delete words from the working stack that indicate a reference coercion.

V3. Get the mode contained in the top word in the working stack and store this mode together with a VOID coercion command in the control stack.

V4. Go to step 10 in the main coercion setup routine.

SKIP ROUTINE

S1. Set CF equal to zero to indicate strong coercion.

S2. Push into the control stack a SKIP coercion command with the mode saved in BMODE.

S3. Go to step 13 in the main coercion setup routine.

NIL ROUTINE

N1. Set CF equal to zero to indicate strong coercion.

N2. If the mode contained in BMODE is not a reference mode then go to step 36 in the main coercion setup routine.

N3. Push into the control stack a NIL coercion command with the mode saved in BMODE.

N4. Go to step 13 in the main coercion setup routine.

LBL ROUTINE

L1. Set CF equal to zero to indicate strong coercion.

L2. Push into the control stack the coercion sequence required to deprocedure the mode contained in BMODE as far as possible.

L3. Push into the control stack a HIP command with the resulting deprocedured mode.

L4. Go to step 13 in the main coercion setup routine.

VAC ROUTINE

E1. Set CF equal to zero to indicate strong coercion.

E2. Remember the length of the control stack in VACT in the case of failure.

E3. Push into the control stack the coercion sequence required to deprocedure the mode contained in BMODE as far as possible.

E4. Push into the control stack a ROW command with the resulting deprocedured mode.

E5. If the resulting mode was a rowed mode then go to step 13 in the main coercion setup routine.

E6. Using the contents of VACT delete all words that were added to the control stack by this routine.

E7. Go to step 36 in the main coercion setup routine.

BAL ROUTINE

B1. Push into the working stack the return address from the main coercion setup routine; a zero that will turn into the strength of the balanced coercion; the mode stored in BMODE; the current length of the control stack; the current value control block pointer; and the count stored in the current value control block (which is a

'W$BAL' value control block).

B2. Push into the control stack a BAL command with a count equal to the count in the current value control block.

B3. Push into the control stack a MODE command with the mode stored in BMODE.

B4. Push into the control stack a VALP command with a pointer to the current value control block which is stored in VALP.

B5. Decrement the contents of VALP so that it points to the previous value control block stored in the working stack.

B6. Recursively call the main coercion setup routine with the value control block pointer in VALP and the a posteriori mode in BMODE as parameters.

B7. If the coercion attempt fails then go to step B14.

B8. Get the coercion strength required from CF and or this into the coercion strength saved in the working stack.

B9. Restore the mode in BMODE to its previous value from the saved mode stored in the working stack.

B10. Decrement the count stored in the working stack. If the count is still nonzero go to step B6. Otherwise continue in sequence.

B11. Store in CF the strength saved in the working stack.

B12. Recover the saved return address in the working stack. Then delete all of the words pushed into the working stack by the BAL routine.

B13. Give a successful return to the entire coercion process.

B14. Restore VALP from the copy saved in the working stack.

B15. Restore BMODE from the copy saved in the working stack.

B16. Using the saved length of the control stack in the working stack delete all words added to the control stack by the BAL routine.

B17. Recover the saved return address in the working stack. Then delete all of the words pushed into the working stack by the BAL routine.

B18. Give a failure return to the entire coercion process.

PAR ROUTINE

P1. Push into the working stack the return address from the main coercion setup routine; a zero that will turn into the strength of the parallel expression coercion; the mode stored in BMODE; the current length of the control stack; the current value control block pointer; and the count stored in the current value control block.

P2. Push into the control stack the coercion sequence required to deprocedure the mode contained in BMODE as far as possible.

P3. Store the mode resulting from the deproceduring in BMODE and also store as target mode in the saved state in the working stack.

P4. If the mode contained in BMODE is not a united mode then go to step P16.

P5. Store the number of modes contained in the united mode in the saved state in the working stack.

P6. Push into the control stack a UNION command with the mode contained in BMODE.

P7. Using the index and the target mode saved in the control stack get the mode in the united mode referred to and store it in BMODE.

P8. Recursively call the main coercion setup routine with the value control block pointer in VALP and the a posteriori mode in BMODE as parameters.

P9. If the coercion attempt fails then go to step P13.

P10. If the coercion strength required was strong (if CF is zero) then go to step P13.

P11. Store the strength required (always firm) in the saved state in the working stack.

P12. Go to step B11 in the BAL routine.

P13. Decrement the index saved in the working stack so that it points to the next mode in the union to try.

P14. If there are more modes to try go to step P7.

P15. Go to step B14 in the BAL routine.

P16. Store in the control stack a PAR command with a count equal to the count in the current value control block (whose type is 'W$PAR').

P17. Store in the control stack a MODE command with the mode in BMODE.

P18. Store in the control stack a VALP command with the current contents of VALP.

P19. Decrement the pointer in VALP so that it points to the previous value control block stored in the working stack.

P20. If the mode contained in BMODE is a rowed mode then go to step P24.

P21. If the mode contained in BMODE is neither a procedure with parameters mode or a structured mode then go to step B14 in the BAL routine.

P22. If the number of parameters in the procedure mode or the number of fields in the structured mode does not equal the count in the original 'W$PAR' value control block then go to step B14 in the BAL routine.

P23. Save the number of parameters in the procedure mode or the number of fields in the structured mode in the saved state in the working stack as an index.

P24. Get the target mode saved in the working stack and if it is not a rowed mode go to step P27.

P25. Get mode of element of rowed mode and store it in BMODE for coercion setup routine.

P26. Go to step P28.

P27. Using the index and the target mode saved in the working stack get the current target mode and store it in BMODE.

P28. Recursively call the main coercion setup routine with the value control block pointer in VALP and the a posteriori mode in BMODE as parameters.

P29. If the coercion attempt fails then go to step B14 in the BAL routine.

P30. Decrement the index saved in the working stack by one.

P31. If the index is still not zero then go to step P24 to process the remaining fields or parameters.

P32. Recover the saved target mode from the working stack.

P33. If this target mode is not a rowed mode then set the coercion strength required that is saved in the working stack to strong.

P34. Go to step B11 in the BAL routine.

<u>The compile coercion routine</u>

The coercion setup routine makes a list of all the coercions to apply to coerce the given value to the a posteriori mode. These coercions are not applied until the compile coercion routine is called. Every value control block in the working stack contains a pointer to the start of code generated for the value and the length of the code. This makes it possible to place code before or after the code for the value, depending on the coercion to be applied. Balanced and displayed values have their constituent values coerced, and then all of the constituent value control blocks are combined into a single value control block for the entire balanced or displayed value. At the end of the coercion all of the value control blocks of the coerced valuc will be combined into a single value control block of type 'W$VALUE'. Except for balanced and displayed values the compile coercion routine is straight forward.

If the coercion command is DEPR, DEREF, REFRW, UNION, PRIM, ROW, or VOID the command with its mode is inserted in the output code after the code compiled for the current value control block. This added word is to be considered as part of the code for the current value control block.

If the coercion command is SKIP, NIL, or VAC then the current value control block has a single word of code associated with it. The type of the value control block is changed to 'W$VALUE' and the mode in the coercion command is stored in the associated code word.

If the coercion command is PROC then the PCDR routine is called to make the current value control block refer to a procedure mode.

If the coercion command is HIP then the current value control block refers to a IDENT code word. This is changed to a O$GOTO code word and a O$HIP code word is inserted in the compiled code after the O$HIP code word. The current value control block is then made to refer to both of these code words.

If the coercion command is VALP then the associated value control block pointer is stored in the current value control block pointer.

If the coercion command is MODE then the mode associated with this command is stored in the current value control block.

If the coercion command is BAL then the following steps are performed:

1. Remember the position of the current value control block whose type is 'W$BAL' and generate a unique label for use by this routine.

2. Make the current value control block pointer point to the preceeding value control block.

3. For N - 1 times where N is the count stored in the 'W$BAL' value control block remembered in step 1

repeat step 4.

4. Make the current value control block pointer point to the preceeding value control block. After the code for the now current value control block add the commands 'O$MA', 'O$DELV', and 'O$JUMP' with the generated label as parameter.

5. Combine the 'W$BAL' value control block and the N value control blocks preceeding it into one value control block that refers to all of the code that was referred to in the individual value control blocks. This step may require moving value control blocks that were stacked on top of the 'W$BAL' value control block to keep the working stack compact.

6. Add to the code for the combined value control block the commands 'O$MA' and 'O$LBL' with the generated label as parameter. These commands are to be considered as part of the code for the value control block.

7. If there was range information in the 'W$BAL' value control block in the form of a range number then insert a 'O$SRNGE' with this range number as parameter in front of the code for the combined value control block and a 'O$ERNGE' with the range number after the code.

An important point to remember is that all of the components of a balanced expression are coerced before they are balanced. This means that all of the component value control blocks will always be of type 'W$VALUE'.

If the coercion command is PAR then the following steps are performed:

1. Remember the position of the current value control block whose type is 'W$PAR'.

2. Make the current value control block pointer point to the preceeding value control block.

3. For N times where N is the count stored in the 'W$PAR' value control block remembered in step 1 repeat step 4.

4. After the code for the current value control block add a 'O$FS' command. Then make the current value control block pointer point to the preceeding value control block.

5. Combine the 'W$PAR' value control block and the N value control blocks preceeding it into a single value control block that refers to all of the code that was referred to in the individual value control blocks. This step may require moving value control blocks that were stacked on top of the 'W$PAR' value control block to keep the working stack compact.

6. Insert a 'O$DISP' command in front of the code for the new value control block and add a 'O$EDISP' command after the code.

Notice that the component value control blocks are also coerced before the PAR command is encountered so all of the component value control blocks will be of type 'O$VALUE'.

The PCDR routine

The PCDR routine inserts and adds the appropriate code so pass 3 can compile procedures. It is called by the coercion routines for a proceduring coercion and by the procedure denotation routine. (It is interesting to note that ALGOL 68 provides no denotation for a procedure without parameters. All such procedures arise from the proceduring coercion.) This routine takes one argument which is the mode of the resulting procedure. It operates on the current value control block.

In ALGOL 68 the scope of a procedure is defined as the largest scope that does not exceed the scope of any mode indication, operator, or identifier in the procedure body. This does not include any mode indication, operator, or identifier that is defined within the procedure itself. Therefore, to determine the scope of a procedure, it is necessary to know the scopes of all mode indications, operators, and identifiers in the procedure. This is accomplished by storing in every value control block a bit word whose bits are associated with the various scopes an identifier, etc. could have at that point in the program. The first bit is associated with the current range, the next bit with the next outer range, and so on. Whenever value control blocks are combined into a single value control block the bit word for the new value control block is calculated by oring together all of the bit words from the original value control blocks. Whenever a range is exited, the bit words associated with value control blocks from that range are all shifted one place to the left. In this manner the scope of a value can be determined by examining the bit word in its associated value control block.

The PCDR routine inserts in front of the code for the current value control block

```
JUMP  #1      Generated label
EPDN  #2      Generated label
LL    range   Procedure scope
```

and adds after the code for the current value control block

```
RETN  mode    Result mode of procedure
LLE   range   Current range
LBL   #1      Generated label
EPDV  mode    Mode of procedure
EPDE  #2      Generated label
```

The value control block is then made to refer to all of these added code words.

PASS 3

In order to understand the operation of pass 3, it is necessary to understand the run time environment in which the compiled code will execute. At all times symbolic index register D contains a pointer to the base of the stack for the current environment. A new level of environment is created whenever a procedure is entered. At this time the contents of index register D are changed to point to the new environment stack. The contents of index register D are restored when the procedure is exited normally. At the base of the stack for a given environment are stored pointers to the surrounding environments stack and to the last environment before the last procedure was entered. These two pointers are distinct. The surrounding environment pointer is a pointer to the environment stack when the procedure was declared and the last environment is the environment that will become the current environment when the procedure exits normally.

Pass 3 reads the Polish code generated during pass 2 and converts it to machine code. During pass 3 the working stack contains 'blocks' that will mirror the run time value stack. Each block represents a value stored in the run time value stack. When a range is entered, a block for each operator or identifier declared in that range is pushed onto the working stack. Every appearance of an identifier in the source program also gives rise to a block on the working stack. The assignation 'X := Y' would cause blocks for X and Y to be pushed onto the working stack. Then, when the assignation itself is compiled, these two blocks would be deleted from the working stack and a block for the value of the assignation would be pushed onto the working stack. In this way the compiler always has a record of the state of the run time value stack.

Whenever a block is created on the working stack, space in the run time value stack is allocated and the address of this space is stored in the block. This means that all values that exist at run time have a place to be stored in the stack even though this value is never actually stored.

A block contains the information as to how the corresponding value can be accessed. Values may be in a register, in the stack, relative to a display register, or relative to an index register. There is a subroutine in pass 3 called MVA (make value available) that will make the value referred to by a given block available for processing. It compiles the necessary code to either bring the value to a register or bring a pointer to the value to an index register. In the latter case the index register may point to a location a known distance away from the value.

The language ALGOL 68 implies a fairly specific organization of memory at run time. This organization is implied by the types of values required at run time and the manipulations allowed on these values. First, a run time representation of values of the various modes must be determined. A void value requires no memory space and its representation is immaterial. A real value requires one word of memory and its representation is the representation required by the floating point hardware of the Honeywell 635. An integral value requires one word of memory and its representation is the 2's complement representation required by the fixed point hardware. A boolean value requires one word of memory and is a word of all zeros for a boolean 'false' and a word of the least significant bit one and all the rest zero for a boolean value of 'true'. A character value requires one word of memory and is a word with the value of the character in the least significant nine bits.

A structured value is made up of one to several fields. The memory representation of a structured value is the memory representations of its component fields in consecutive memory locations in the same order as the order of the fields in the structure.

A united value is a value can contain a value of one of several modes. At run time it must be possible to determine the mode of the current value contained in the united value. The memory representation for a united value consists of a word that contains a pointer indicating the mode of the contained value followed by as many consecutive words as necessary to contain the longest possible value. The leftmost of these words contains the memory representation of the contained value.

A reference value is a value that refers to another value. Except in the case of references to row values, the representation of a reference value requires one word. It contains a pointer to the referred value in the upper half and a pointer indicating the mode of the value in the lower half.

A row value or a reference to a row value presents several problems in its representation. This is because there are two types of row values: flexible and fixed. Also, a row value consists of a descriptor and a set of values accessed through the descriptor. Since only the mode of a row value is known at compile time and not its dimension, the amount of space required by a row value cannot be known at compile time. However the length of the descriptor which is a function of the dimensionality of the row value is known at compile time. A row value can therefore be assumed to consist of two parts: a descriptor which is considered to be the value of the array and its elements which are allocated in memory somewhere else. A row value can be manipulated by only manipulating its descriptor thereby making the length of row values known at compile time. Of course, the allocation of memory for the elements of a row value has not been considered but this problem can be considered separately.

A separate problem arises with references to row values.

ALGOL 68 allows the programmer to slice a reference to a row value to obtain a reference to a part of the row value. This operation requires the construction of a new descriptor to reference the slice. However, memory space must be found for the new descriptor. Since the scope of the slice is the same as the scope of the original reference to row value, a logical place for the new descriptor is with the reference to row value. This means that the descriptor for a reference to row value is associated with the name part of the value rather than with the element part of the value.

Another problem arises with references to flexible row values. These values can dynamically change their size during the execution of the program and in particular their size can increase. If the elements of a flexible array are to be stored in consecutive memory locations then all of the elements of the array must be moved when the number of elements increases. However, moving the elements of the array cannot result in any effect discernable by the program so there must be a unique descriptor associated with a flexible array so that when this descriptor is changed to reflect the changed location of the elements all references to the array will be through this descriptor. In other words, the descriptor must be associated with the element part of the value and not the name part. This is just the opposite to the fixed array situation. Since the bounds are not in general known at compile time, a general reference to row value must be able to handle both the fixed and flexible case.

The memory representation of a reference to row value consists of a flag word followed by a descriptor. If the row value is flexible then the flag word points to the descriptor for the row value and the descriptor in the reference to row value is not used. If the row value is fixed then the flag word is zero and the descriptor for the array is stored in the following memory locations. This representation will require a run time check of the flag word whenever a reference to row value is used. The memory representation for a descriptor is a one word pointer that points to the first element of the array followed by as many quadruples of words as there are dimensions in the row value. Each quadruple contains in order an integral lower bound, an integral upper bound, a stride, and a word in which the states are stored.

Procedure values are unusual values in that they are, strictly speaking, executable code. The memory representation for a procedure contains a pointer to this code so that actual code is never manipulated. Procedure values also contain an environment pointer that specifies the environment of the procedure when its denotation was encountered in the program. This environment may be different from the environment where the procedure is called. Also, a procedure value contains the amount of temporary storage the procedure will require when executed so that a new stack frame can be allocated when the procedure is called. The memory representation for a procedure value contains four words. The first word is always zero and is included to make the procedure call easier. The second word contains in its upper half the amount of temporary storage needed by the procedure in the new stack frame. The third word contains a pointer to the environment of the procedure. This is actually a pointer to the base of the stack frame allocated for the innermost range that the procedure can reference nonlocally. The fourth word contains a pointer to the code for the procedure.

In ALGOL 68 as in ALGOL 60 there is the concept of a program environment. At any point in the source program an identifier, mode indication, or operator has a meaning. This identifier, mode indication, or operator identifies a declaration for the same identifier elsewhere in the program. This declaration is found by first looking for an appropriate declaration somewhere in the current range. If this search fails then the range containing this range is searched excluding contained ranges. This process is repeated using bigger and bigger ranges until an appropriate declaration is found. If no declaration is found then there is an error in the source program. In this way all identifiers etc. in a program are associated with a specific declaration.

ALGOL 68 allows recursion or the ability of a procedure to call itself. This means that a procedure can call itself without exiting and therefore cause two copies of itself to exist simultaneously. Since there are two copies of the procedure there are two copies of all declarations contained in the procedure. A question arises as to which instance of a declaration is the one that a given identifier identifies. This question is answered by defining the proper declaration as the declaration that is in the same copy as the identifier if the declaration was copied otherwise the unique declaration that was not copied.

The identification problem can be solved at run time by associating a stack frame with each activation of a range. A stack frame is an area of memory that is allocated in the run time value stack for use by a specific range. Memory for all identifiers in a given range is allocated in the corresponding stack frame. Every stack frame contains an environment pointer which is a pointer to the stack frame associated with the surrounding range. The memory associated with an identifier can be accessed at run time by following the chain of environment pointers through stack frames associated with surrounding ranges until the proper stack frame is found. The memory associated with the identifier is then at a known distance from the base of this stack frame. Since it can be determined at compile time how many environment pointers need to be followed and the offset of the memory associated with the identifier, accessing values at run time is reasonably straight forward.

A new stack frame is needed only when a procedure is called because only procedures can be recursive. When a procedure is entered, a stack frame is allocated on the stack and the environment pointer for the procedure is stored at the base of the new stack frame. This will allow the procedure to access values nonlocal to the procedure stack frame. Also stored in the new stack frame is a pointer to the stack frame in use at the point of the procedure call. This is needed to restore the environment to this stack frame when the procedure exits normally. It can be seen that an environment pointer is a pointer to an active stack frame.

Since the running program must always be able to access the local stack frame, an index register is reserved to always point to the base of the local stack frame. The contents of this index register is changed only when a procedure is entered or exited or when a goto is executed that jumps out of the current range. It is also necessary for the running program to know the current extent of the run time value stack. This is so new stack frames can be allocated as well as memory for local generators. A second index register is reserved for this purpose and always points to the next free location in the run time value stack.

Here follows a list of the codes generated by pass 2 and the action pass 3 takes with each.

OP (DEF)

The DEF table entry is for a user defined operator. The procedure value of this operator is to be pushed into the run time value stack and the stack marked. This is equivalent to the sequence:

```
IDENT (DEF)
MSCW
```

OPE (DEF)

The DEF table entry is for a predefined standard operator. The operands for this operator have already been pushed into the run time value stack. The macro prototype referred to by the DEF table entry is elaborated causing the operand values to be removed from the run time value stack and the value of the formula to be pushed into the run time value stack. The specific code generated comes from the macro prototype. A typical macro for the '+' operator is as follows:

```
INB  *+2   JUMP IF SECOND OPERAND NOT IN REGISTER
ADQ. A     A STANDS FOR FIRST OPERAND
IFA  *+2   JUMP IF FIRST OPERAND IS IN REGISTER
LDQ  A     A STANDS FOR FIRST OPERAND
```

```
          ADQ. B      B STANDS FOR SECOND OPERAND
```

The IFA and INB pseudo operations are used to test whether the operands are in a register. A period ('.') after an operation indicates the end of code generation. If neither operand is in a register the above macro prototype will generate the following code:

```
          LDQ  A
          ADQ  B
```

LBL (label number)

This code is used to define a label in the object code. The value of the label is the current place in the generated output. No object code is generated.

JUMP (label number)

This code is used to jump to a label defined by the LBL code. A jump instruction is to be compiled to the given label number at the current place in the object code. The instruction

```
          TRA  [label]-*,IC
```

is generated.

DISP (MODE)

This code is used to signal the beginning of either a row or structure display. It is immediately followed by codes for the fields or elements of the display. This code is currently ignored by the compiler.

EDISP (MODE)

This code is used to signal the end of either a row or structure display. MODE is the mode of the display. The run time value stack contains the values of the fields or elements of the display. These values are to be removed and replaced by the single value of the display. No object code is generated but the control blocks representing the fields of the structure are replaced by a single control block representing the structured value. Row displays have not yet been implemented.

ENTER (MODE)

This code is used to enter a procedure or call a user defined operator. The run time value stack contains an active procedure value followed by values of the actual parameters. Code required to enter the procedure is compiled and the procedure value and all actual parameters are deleted from the run time value stack. At run time a procedure will return a value in the run time stack so the compiler can assume that the a value whose mode is MODE is pushed into the run time value stack. The object code generated is (where M is the address of the MSCW):

```
          EAX1  M,D       Get address of MSCW in XR - 1
          LDX2  M+2,D     Get address of new stack frame
          TSX0  R$ENTER  Call standard entry subroutine
```

GOTO (DEF)

This code is used to jump to a user label. The DEF table entry corresponds to the user label. This code is different from the JUMP code because the location of the label and the location of the goto may be in different environments. The object code generated depends on the relative environments of the current environment and the environment of the label. The instruction

```
          LDX  D,2,D   Exit one procedure environment
```

is repeated as many times as the number of procedures exited. If no procedures are exited then this instruction is omitted. If there is any change of range then the instruction

```
          LDX  S,[saved S],D  Restore stack pointer
```

is generated. Finally, the instruction

```
          TRA  [label]-*,IC    Jump to label
```

is generated. If the goto does not cross a range boundary then the last instruction is the only instruction generated.

HIP (MODE)

A GOTO code is used in a place where a value is normally expected. A HIP code follows a GOTO code and causes a value of the specified MODE to be pushed into the run time value stack. It is immaterial what value is pushed as the preceeding jump will always take place and the value will never be used. No object code is generated.

VOID (MODE)

The top value in the run time value stack is deleted and a void value takes its place. No object code is generated.

LGEN (MODE)

This code is used to invoke a local generator of the specified mode at run time. The value of the generator is to be pushed into the run time value stack. First, the stack is marked with a MSCW. This is so the garbage collector, if called, will be able to mark the current stack frame. Then the following instructions are added to the object code (where M is the address of the MSCW word):

```
EAX1  M,D     Get pointer to MSCW word in XR - 1
EAX2  [len]   Get length of value in XR - 2
TSX0  R$LGEN  Call local generator routine
ADA   type,DL Add type field to pointer to value
```

The A register then contains a pointer to the generated value.

HGEN (MODE)

This code is used to invoke a heap generator of the specified mode at run time. The value of the generator is to be pushed into the run time value stack. First, the stack is marked with a MSCW. This is so the garbage collector, if called, will be able to mark the current stack frame. Then the following instructions are added to the object code (where M is the address of the MSCW word):

```
EAX1  M,D      Get pointer to MSCW word in XR - 1
EAX2  [len]    Get length of value in XR - 2
TSX0  R$HGEN   Call heap generator routine
ADA   type,DL  Add type field to pointer to value
```

The A register then contains a pointer to the generated value.

CONF (MODE)

This code is used in conformity relations to check the mode of the right hand side of a conformity. The mode of the value of the right hand side of a conformity (which is stored in a special way) is to be checked against MODE. If the modes match a boolean true value otherwise a boolean false value is to be pushed into the run time value stack. The type of the right hand side is in XR - 0. The following code is generated:

```
CMPX0 type,DU See if type matches
TNZ   3,IC    Jump if type does not match
LDQ   1,DL    Get a true in Q register
TRA   2,IC    And jump to end of sequence
LDQ   0,DL    Get a false in Q register
```

This code leaves the result in the Q register.

TF (LBL)

The top value in the run time value stack is of mode boolean. This value is deleted and code is compiled to jump to the given label only if the boolean value was false. Otherwise the compiled code does nothing. The object code generated is:

```
          SZN  [bool]          Sense boolean value
          TZE  [label]-*,IC    Conditionally jump to label
```

Some optimization is done if the boolean value was just generated.

CASE (N)

The CASE code is followed by N+1 TRA codes. The top value in the run time value stack is of mode integer. This value is deleted from the run time value stack. Code is compiled so that if the integer value was less than zero or greater than N the immediately following TRA code will be executed. Otherwise, the I+1'th TRA code will be executed where I was the value of the integer. The code generated is:

```
          CMPQ  MAX+1,DL  MAX is the maximum allowed value
          TRC   3,IC      Too big so jump to zeroth TRA
          STC2  1,IC      Store address of zeroth TRA
          TRA   [],QL     Jump to proper TRA
```

Notice that the third instruction modifies the fourth instruction.

CASGN (MODE)

The top value in the run time value stack is the left hand side of a conformity relation and the second from the top value in the run time value stack is a pointer to the value of the right hand side of the conformity relation. MODE is the mode of the left hand side value. The value of the right hand side without the union prefix is to be assigned to the left hand side name and both value deleted from the run time value stack. First the instruction

```
          LDX0  ptr,D  Get pointer to RHS in XR - 0
```

is added to the object code. XR - 0 then points to the united value and the value (ununited) starts in the following location. The move routine is then called to move the ununited value at [1,0] to the location specified by the left hand side control block.

SELCT (N)

The top value in the run time value stack is a structured value. It is to be replaced by the Nth field of the structured value. The offset for the selected field is calculated and the move routine is called to compile object code to move the selected field to the run time stack. The control block for the structured value is replaced by a control block for the selected value.

RSLCT (N)

The top value of the run time value stack is a reference to structure value. It is to be replaced by a reference to the Nth field of the structured value. No object code is compiled but the control block for the reference to structured value is replaced by a control block for the selected field. This is accomplished by adding the field offset to the saved address of the structure.

ETC (MODE)

This code always follows either a SELCT or RSLCT code. The MODE is the mode of the selected field and is the mode of the result of the selection. This code generates no object code but is used to set the mode of the selection in the control block.

SKIP (MODE)

This code causes code to be generated that pushes a value of the given mode into the run time value stack. Any such value is acceptable for a SKIP. The object code generated for a skip (where A is the address of the skip and N is the length of the value) is:

```
STZ  A,D      Zero out value
STZ  A+1,D    Zero out value
STZ  A+2,D    Zero out value
...  .....    .... ... .....
```

```
          STZ  A+N-1,D  ZERO out value
```

If the mode of the value is void then no instructions are generated because the length of a void is zero.

NIL (MODE)

This code causes code to be generated that pushes the name 'nil' into the run time value stack. The given mode is the mode of the nil. The object code generated for a nil (where A is the address of the nil and N is the length of the value) is:

```
          STZ  A,D      Zero out value
          STZ  A+1,D    Zero out value
          STZ  A+2,D    Zero out value
          ...  .....    .... ... .....
          STZ  A+N-1,D  Zero out value
```

Generally, only a single instruction will be generated.

IS (MODE)

The top two values in the run time value stack are operands in an identity relation. These two values are deleted from the run time value stack and replaced by the result of the identity relation. The object code generated (where N1 is the first name and N2 is the second name) is:

```
          EAX0  N1      Get first name in XR - 0
          STX0  2,IC    Store in fourth instruction
          EAX0  N2      Get second name in XR - 0
          CMPX0 []        See if names are equal
          TNZ   3,IC      Transfer if not equal
          LDQ   1,DL      Get a true in Q register
          TRA   2,IC      Transfer to end of sequence
          LDQ   0,DL      Get a false in Q register
```

N1 and N2 in the above code are actually more complicated addresses and generally involve index modification.

ISNT (MODE)

The top two values in the run time value stack are operands in an identity relation. These two values are deleted from the run time value stack and replaced by the result of the identity relation. The object code generated (where N1 is the first name and N2 is the second name) is:

```
          EAX0  N1        Get first name in XR - 0
          STX0  2,IC      Store in fourth instruction
          EAX0  N2        Get second name in XR - 0
          CMPX0 []        See if names are equal
          TZE   3,IC      Transfer if equal
          LDQ   1,DL      Get a true in Q register
          TRA   2,IC      Transfer to end of sequence
          LDQ   0,DL      Get a false in Q register
```

N1 and N2 in the above code are actually more complicated addresses and generally involve index modification.

TRUE (MODE)

This code causes the generation of code that pushes the boolean value 'true' into the run time value stack. The instruction

```
          LDQ   1,DL    Get a true in Q register
```

is added to the object code.

FALSE (MODE)

This code causes the generation of code that pushes the boolean value 'false' into the run time value stack. The instruction

```
          LDQ   0,DL    Get a false in Q register
```

is added to the object code.

MSCW

The top value in the run time value stack is a procedure. Code is generated to change this procedure value into an active procedure value by allocating space needed by the procedure and linking this value to the last active procedure value in the run time value stack. First the stack is marked in the first word of the procedure value. Then the following instructions are added to the object code (where M is the address of the procedure value):

```
          STX   S,LSTMK,D Save the current stack pointer
          EAX1  M,D     Get pointer to MSCW in XR - 1
          LDX2  M+1,D   Get length needed in XR - 2
```

```
TSX0  R$PLGEN Allocate space for procedure
EAX0  0,AU    Get address of space in XR - 0
EAA   M,D     Get address of MSCW in A
ADA   T$MSCWT,DL Get type for pointer also in A
STA   0,0     Store at base of new space
STZ   1,0     Zero stack save word in new space
LDA   M+2,D   Get environment pointer for proc
STA   2,0     And store in new space
EAA   0,D     Get base of current stack
STA   3,0     And store in new space
STX0  M+2,D   Store pointer to new space in MSCW
LXL0  M+1,D   Get type of new space
SXL0  M+2,D   And move it
STZ   M+1,D   Zero out old len/type word
```

When these instructions are executed a new stack frame will be allocated and a pointer to this new stack frame is kept in the old procedure value. When the procedure is entered the new stack frame will point to the current stack frame.

IDNTY (DEF)

The given DEF table entry is for an identifier or operator. Code is compiled to bring a reference to this value to the run time value stack in preparation for an identity declaration or operator declaration. Actually, no code is generated but a control block that references the location the identifier is assigned is created.

IDNTE

The second from the top value in the run time value stack is a reference to an identifier or an operator stored there by a IDNTY code. The top value in the run time value stack is the value the identifier or operator should be identical to. Code is generated to assign the top value to the location specified by the reference that is second from the top. Then both values are deleted from the run time value stack. The object code generated is identical to the object code generated for the code 'ASGNE'.

FORMP (DEF)

The given DEF table entry refers to a formal parameter in a procedure denotation. Space for the value of the formal parameter is allocated in the run time value stack and object code is generated to move the actual parameter to the location assigned to the formal parameter. The object code generated has the form:

```
          LDQ   4,1     Get actual parameter
          STQ   6,2     And store in formal parameter
          LDQ   5,1     Get actual parameter
          STQ   7,2     And store in formal parameter
          LDQ   6,1     Get actual parameter
          STQ   8,2     And store in formal parameter
          ...   ...     .....................
```

The number of pairs of instructions generated equals the length of the value of the formal parameter. Successive parameters continue at the address where the preceeding parameter left off.

IDENT (DEF)

The given DEF table entry is for an identifier or operator. A control block for this value is created in the run time value stack but no object code is generated.

DENOT (DEF)

The given DEF table entry is for a denotation value. Code is compiled to bring the value of the denotation to the run time value stack. A typical object code sequence for an integral denotation is:

```
          TRA   2,IC    Transfer around value
          [actual integer here]
          LDQ   -1,IC   Load integer in Q
```

The only denotations currently implemented are real, integral, boolean, and character.

ASGN (MODE)

The given mode is the mode of an assignation that immediately follows this code. No code needs to be generated for this code.

ASGNE (MODE)

The second from the top value in the run time value stack is the left hand side of an assignation. The top value is the right hand side of the assignation. Code is compiled to assign the right hand side value to the place specified by the left hand side value. Then the right hand side value which is the top value is deleted from the run time value stack leaving the left hand side value on the stack as the value of the assignation. The code generated for an assignation merely moves each word of the source to the location specified by the destination. If an array value is moved, code is generated to move each element of the array by means of a loop.

LL (PROG)

The given pointer to the PROG table indicates a new environment in the object program. The old environment is pushed into a compiler stack (the control stack) and the compiler is set for the new environment. No object code is generated.

LLE (PROG)

The environment that was saved by the LL code is restored and the compiler is set for the old environment. No object code is generated.

DELV (MODE)

The top value in the run time value stack whose mode is the given mode is deleted from the stack. No object code is generated.

SRNGE (PROG)

The old environment is pushed into a compiler stack (the control stack) and the compiler is set for the new environment. Code is generated to store a mark in the run time value stack and link it to the previous mark or active procedure. Then the instruction

```
          STX   S,LSTMK,D  Save stack pointer in last MSCW+1
```

is generated to save the current value of the local stack pointer.

ERNGE (PROG)

The environment that was saved by the SRNGE code is restored and the compiler is set for the old environment. The value of the range just exited is made available and pushed into the run time value stack. Then the instruction

```
          LDX   S,LSTMK,D  Restore stack pointer
```

is generated to restore the local stack pointer to the value it had when the range was entered.

EPDN (LBL)

This code is used to define the entry point to a procedure. The LBL is defined at the current place in the generated code. Then the instruction

```
          EAX   D,0,2  Set new environment for procedure
```

is generated to set XR - D to point to the new stack frame.

RETN (MODE)

This code is used to return the result of a procedure to the caller and return control. The two instructions

```
          LDX0  0,D     Save pointer to current stack frame
          LDX   S,3,0   Save pointer to old stack frame
```

are added to the object code. Then the value of the procedure is moved to address [0,0]. This is the address of the calling MSCW. Then the instruction

```
          TSX0  R$RET   Return to caller
```

is added to the object code. The routine R$RET restores the caller's environment and returns control.

EPDV (MODE)

The given mode is the mode of a procedure denotation. Code is generated to push a value of this mode in the run time value stack. Other information required for the creation of the procedure value has been accumulated during the compilation of the procedure denotation such as the environment of the procedure and the storage required by it. The following instructions are added to the object code:

```
          STZ   P,D     Zero out first word of value
          EAA   MAXST   Get size of stack frame needed
          ADA   type,DL Add type of stack frame
          STA   P+1,D   Store in second word of value
          get environment pointer in A register
          STA   P+2,D   Store in third word of value
```

The environment pointer is obtained by compiling as many of the following instructions as necessary to reach the proper environment:

```
          LDA   2,D     Get surrounding environment
          LDA   2,AU    Get surrounding environment
```

If the environments are the same then the instruction

```
          EAA   0,D     Get current environment in A
```

is compiled.

EPDE (LBL)

The given entry is the entry point of a procedure denotation. This entry point is inserted in the procedure value created by the code EPDV. The following instructions are added to the object code:

```
          EAA   ENTRY-*,IC get entry to proc in A
          STA   P+3,D   Store in fourth word of value
```

This stores the entry point to the procedure in the procedure value.

DLEN (LBL)

This code is used for bound procedures. The maximum length of the run time value stack in the preceeding procedure is added as code to the generated output. Then the given label is defined to be the address of the word just added to the generated output.

VSBCT (N)

This code indicates that the top value in the run time value stack is an integral subscript for the Nth subscript position of an array. This fact is noted in a word pushed onto the control stack. No object code is generated.

VLWB (N)

This code indicates that the top value in the run time value stack is a lower bound trimmer for the Nth subscript position of an array. This fact is noted in a word pushed onto the control stack. No object code is generated.

VUPB (N)

This code indicates that the top value in the run time value stack is an upper bound trimmer for the Nth subscript position of an array. This fact is noted in a word pushed onto the control stack. No object code is generated.

VNLWB (N)

This code indicates that the top value in the run time value stack is a new lower bound trimmer for the Nth subscript position of an array. This fact is noted in a word pushed onto the control stack. No object code is generated.

VEPTY (N)

This code indicates that the Nth subscript position in a trimmer is empty. This fact is noted in a word pushed onto the control stack. No object code is generated.

LWB (N)

This code indicates that the top value in the run time value stack is the Nth lower bound in a declarer. Code is generated to store this value in the appropriate location of the declarer being constructed.

UPB (N)

This code indicates that the top value in the run time value stack is the Nth upper bound in a declarer. Code is generated to store this value in the appropriate location of the declarer being constructed.

FIX (N)

This code indicates that the last bound calculated is fixed. Code is generated to store this state in the declarer being constructed.

FLEX (N)

This code indicates that the last bound calculated is flexible. Code is generated to store this state in the declarer being constructed.

SUB

This code indicates that a slice is about to be compiled. The compiler is set to evaluate the following trimscript.

BUS (MODE)

This code indicates that the top values in the run time value stack are subscripts and trimmers to an array value below them in the run time value stack. Code is compiled to do the indicated trimming and subscripting. The given mode is the resulting mode of the slice. All trimmer and subscript values are deleted from the run time value stack and the array value now on top of the stack is replaced with the slice value.

BOUND (BOUND)

This code always follows either a LGEN or a HGEN code and specifies bounds for generated values that require them. Code is compiled that will insert the required bounds by calling appropriate bound procedures.

RBUS (MODE)

This code indicates that the top values in the run time value stack are subscripts and trimmers to an array value below them in the run time value stack. Code is compiled to do the indicated trimming and subscripting. The given mode is the resulting mode of the slice. All trimmer and subscript values are deleted from the run time value stack and the array value now on top of the stack is replaced with the value of the slice.

DSUB

Code is generated for the header of a bound procedure. This involves the fetching of the actual parameter which is a pointer to the descriptor.

DBUS (BOUND)

Code is generated for the trailer of a bound procedure. This involves generating code to calculate the length of the array and return to the caller and calculate the strides in the descriptor.

PRIM (MODE)

The top value in the run time value stack is an integer or a long integer. Code is compiled to change this value to a real or long real value.

DEREF (MODE)

The top value in the run time value stack is a reference mode. Code is compiled to dereference this value whose mode is the given mode.

ROW (MODE)

Code is generated to row the value on top of the run time value stack to the given mode. This is not currently implemented.

DEPR (MODE)

The top value in the run time value stack is a procedure without parameters. Code is generated to deprocedure this value by calling the procedure and the resulting value replaces the procedure value in the run time value stack. This code is equivalent to the code 'MSCW' followed by the code 'ENTER'.

UNION (MODE)

Code is generated to unite the value on top of the run time value stack to a value of the given mode. This value replaces the original value on the run time value stack. The generated code merely inserts a type word in front of the value that contains the type of the value.

MREF

Space is allocated on top of the run time value stack by the compiler for a pointer. No code needs to be generated.

CONE

The second from the top value in the run time value stack is space for a pointer and the top value is the right hand side of a conformity relation. Code is compiled to store a pointer in the space provided to the right hand side value. Then both values are deleted from the run time value stack. (The value of the pointer is not really lost because a transfer instruction will be compiled to code to pick up the value.)

MAX

The compiler's stack pointer is set to the maximum value that it has attained in the current range.

MA

Code is compiled if necessary to convert the top value in the run time value stack to standard form. This is necessary in balanced expressions where a value may be generated in several places and must have the same form from all places.

FS

Code is compiled if necessary to store the top value in the run time stack in the stack in standard form. This is necessary when constructing a display where all values of the display must be in memory.

THE LOADER

A compiled program consists of a body of code that is absolutely relocatable (i.e., code that does not have to be altered if it is moved in memory) followed by a list of modes with their associated patterns followed by a list of SYMDEF's and SYMREF's followed by one word containing the global environment length required by the program. Each pattern contains a pointer to a chain of addresses that should point to the pattern.

The main program is compiled as a declaration of a procedure whose name is all blanks with no arguments and no result that when called causes execution of the program.

If a program makes a reference to an identifier declared in an independently compiled program, the compiler assumes that the identifier is declared in a global environment and assigns space in the global environment. Conversely, if a program declares an identifier that will be used by another program, space is also assigned in the global environment. When there are several programs to be bound into a single program each program has its own global environment.

The loader maintains several tables for its use. The program table contains an entry for each program segment loaded and contains the base address of the segment and space for the location and length of the global environment for this segment. Entries in the type table contain a pattern and a link to a chain of addresses in the program segment that refer to the pattern and an optional cannonical mode. When constructing the type table the loader will combine entries having the same mode so that there is a unique pattern associated with each mode. This is so conformity relations will work properly with values from different program segments. There is a SYMDEF/SYMREF table that contains an entry for each symbol that is used by more than one program segment. This table contains for each symbol a chain of definitions and references of the symbol with the address associated with the symbol in the various program environments.

The loader first constructs appropriate table entries for the main program. It then searches for a symbol in the SYMDEF/SYMREF table that has no definition. If it finds one it loads a program segment of this name and makes further entries in the various tables. When there are no more symbols without definitions it constructs patterns for all entries in the type table and fills in all addresses that reference these patterns. The loader then allocates space for the global environments and inserts their addresses in front of the corresponding program segment. All of the global declarations in the segments are then executed causing values for all the global symbols to appear in their respective global environments. Then, using entries in the SYMDEF/SYMREF table the value for each defined symbol is moved to all locations in other global environments where it is used. Finally the value of the symbol consisting of all blanks (which is the procedure value of the main program) is used for an ordinary procedure call and the program starts execution.

Cannonical modes

One of the problems associated with the loader is the unique representation of modes in precompiled programs. During compilation, modes are represented by list structure in the MODE table. Unfortunately, different programs may result in different internal representations for the same mode because of a different order of appearance of modes in the program. This different representation does not bother the compiler but it means that the internal representation of modes is not a suitable external representation for use by the loader.

One method for generating a cannonical form for a mode is to use a list copying routine and copy the list structure that is the internal representation for the mode. This will create a list structure that represents the mode and that is unique for a given mode [Cheney, C.J., "A nonrecursive list compacting algorithm", Communications of the ACM, 13, 677-678 (1970)]. One disadvantage of this technique is that every mode specified must include all component modes even though these modes are specified elsewhere. This means that the memory required for a list of cannonical modes will consume much more space than a single list structure that specifies the same list of modes.

The above method works for all modes except united modes. United modes are considered to be equivalent even though their component modes are in a different order or some modes are repeated. The cannonization process described above will not necessarily cannonize two united modes into identical list structures. If, however, a cannonical ordering is imposed on all modes and the cannonical form for a united mode is defined to be with each component mode occuring once and in cannonical order then united modes will be cannonized properly. This solution requires a cannonical ordering on all modes to be defined. A set of rules for a cannonical ordering of modes is as follows:

1. A primitive mode (real, int, bool, char, bits, bytes, sema etc.) comes before a row mode which comes before a reference mode which comes before a structured mode which comes before a procedure mode which comes before a united mode.

2. A primitive mode comes before a long primitive mode which comes before a long long primitive mode etc. Since there are only a finite number of non-long primitive modes they are ordered in alphabetical order.

3. Two reference modes are ordered in the same order as their dereferenced modes.

4. A structured mode with fewer fields comes before a structured mode with more fields. If two structured modes have the same number of fields then the corresponding tags for the fields are examined. If there is any pair of corresponding tags that is different then the structured mode whose first different tag is alphabetically before the corresponding tag in the other structured mode comes first. Otherwise, if all corresponding tags are equal then the structured mode whose first different mode that comes before the corresponding mode in the other structured mode comes first. (Notice that all modes cannot be the same or the two structured modes would be identical and neither structured mode would come before the other.)

5. A procedure mode with fewer arguments comes before a procedure mode with more arguments. If two procedure modes have the same number of arguments then corresponding modes of the arguments are considered. If there is any pair of corresponding argument modes that is different then the procedure mode whose first different argument mode that comes before the corresponding argument mode in the other procedure mode comes first. If all pairs of modes are the same then the procedure mode with the result mode that comes before the result mode of the other procedure mode comes first. (If the result modes are also the same then the two procedure

modes are identical.)

6. The cannonical ordering between two united modes is determined by first ordering the constituent modes of each united mode in cannonical order. If two constituent modes of a united mode are the same then one of them is deleted. The united mode with the least number of constituent modes comes first. If two united modes have the same number of constituent modes then the united mode whose first different constituent mode that comes before the corresponding constituent mode in the other united mode comes first. (At least one constituent must be different or the two united modes would be equivalent.)

There is one subtle point in this method for cannonically ordering modes. The ordering of the constituent modes of a united mode depends on their cannonical order and it might be possible to construct two different but consistent orderings involving united modes. Fortunately this problem does not arise because if there are two consistent cannonical orderings for a set of modes then two modes appearing at the same rank in the different orderings are equivalent. This can be seen by examining the cannonizing rules and noting that if the constituent modes of a mode have to be examined then the modes are equivalent if the constituent modes are equivalent.

As an aside, an alternative method for determining equivalences in the mode table between pass 1 and pass 2 would be to use the cannonical ordering rules to sort the mode table. One would have to be careful in writing such a routine because the number of constituent modes in a united mode may change during the sorting process as modes are discovered to be equivalent. The execution time of such a routine would be of the order of (n log n) instead of n squared where n is the number of modes in the MODE table.

The garbage collector

In ALGOL 68 there are two ways in which memory can be freed at run time. Local values are released when the range that created them is exited. This local storage is allocated on a stack and when a range is exited, the stack pointer is restored to the value it had when the range was entered. Heap storage is allocated in the heap and storage is reclaimed from the heap only by the process of garbage collection. When all available space is used up the garbage collector determines all of the memory that can still be accessed by the program. This memory is compacted into two contiguous areas: the stack and the heap. The remainder of the memory is then free for allocation for local or heap values.

In order for a garbage collector to operate correctly it must be able to determine the exact structure of memory.

All locations that contain pointers must be found and the type of value to which these pointers point must be known. This includes all temporary memory locations that can contain a pointer. This means that run time memory organization is very rigid. Since memory is compacted by the garbage collector, all pointers that point to moved values must be relocated.

All pointers at run time consist of an address and a tag. The address part contains the address of the value pointed to and the tag part contains the address of a pattern for the value that is pointed to. A pattern is a list of data specification words followed by a pattern terminator word. A data specification word specifies the type of data in the value. There are six types of data specification words corresponding to six uses of memory. They are:

SKIP N (skip N words)

The next N words are not being used currently but they should be marked so that the compactor won't squeeze this data structure. The skipped words are part of the temporary storage allocated for a procedure.

OCT (value that is not a pointer)

The next word is a non-pointer data item and should be marked.

PTR (pointer)

The next word is a pointer with the pointer in the upper half and a pointer to the pattern for the pointer.

WPTR (working pointer)

The next word is a pointer in the supervisor that should not be marked because it is outside of garbage collectable memory. Pointers of this type never appear in collectable memory and are used by routines calling the garbage collector to relocate internal pointers.

ROW N (N dimensional row value)

The next 4*N+1 words form a row descriptor with the first word pointing to the first element of the row value and succeeding sets of four words containing the lower bound, upper bound, stride, and states. All words of the descriptor and all elements should be marked.

ULEN (union length word)

The next word is the first word of a united value. It contains the length of the value in the upper half and a pattern pointer for the current value in the lower half. This following value should be marked as well as any extra words not used by the current value.

The strategy for garbage collection is to trace the entire structure of memory and record words in use in a separate bit table. The bit table contains a bit for every word in collectable storage. Words in use are then moved to adjacent locations with words associated with the stack moving down and word assiciated with the heap moving up. Then all pointers are adjusted so that they point to the new location of their target. The garbage collector is divided into a list structure tracing part and a memory compacting part.

The list structure tracing routine is used first to mark all words usable by the program and second to relocate all pointers after the memory compacting routine has shuffled memory around. The algorithm for the tracing routine is as follows:

1. Clear the mark table and the array chain pointer. Set A, N, and F equal to zero and B equal to the pointer to the list structure to mark. This pointer has an address pointer and a type pointer.
2. If F is zero then go to step 4.
3. Using the relocation table relocate the address pointer in B without changing the type pointer in B. (If the pointer to be relocated does not point to a word that is marked then it is relocated to point to the first following marked word. This situation arises when a pointer to the end of the stack is saved when a new range

is entered.)

4. If either the address pointer or the type pointer in B is zero then go to step 20.

5. If the type pointer in B points to a 'WPTR' pattern word then go to step 15.

6. If the type pointer in B does not point to a 'SKIP' pattern word then go to step 8.

7. Set N consecutive bits in the mark table starting with the bit corresponding to the word referenced by the address pointer in B where N is the number contained in the 'SKIP' pattern word. Increment the address pointer in B by N and the type pointer in B by 1 and go to step 17.

8. If the mark table bit corresponding to the word referenced by the address pointer in B is set then go to step 16.

9. Set the mark table bit corresponding to the word referenced by the address pointer in B.

10. If the type pointer in B points to a 'OCT' pattern word then go to step 16.

11. If the type pointer in B does not point to a 'ULEN' pattern word then go to step 13.

12. Let C stand for the word referenced by the address pointer in B. Then simultaneously move the type pointer from the C word to the word in B; move the type pointer from the word in B to the word in A; and move the

type pointer from the word in A to the C word. Increment the address pointer in B by 1 and go to step 4.

13. If the type pointer in B does not point to a 'ROW' pattern word or if F is not zero go to step 15.

14. The address pointer in B points to a descriptor of N dimensions where N is the number in the 'ROW' pattern word. Pack the state information somewhere in the descriptor to make room for a 'place' and initialize all N places to zero.

15. Let C stand for the word referenced by the address pointer in B. Then simultaneously move the C word to B; move B to A; and move A to the C word. Then go to step 2.

16. Increment the address pointer and the type pointer in B both by 1.

17. If the type pointer in B points to a positive word (not a pattern terminator word) then go to step 4. (note that 'OCT', 'PTR', 'ROW' etc. pattern words are all positive.)

18. If A is zero then go to step 27.

19. The type pointer in B points to a pattern terminator word. Add this word to B to restore B to the original address pointer and type pointer.

20. If the type pointer in A does not point to a 'ULEN' pattern word then go to step 22.

21. Decrement the address pointer in B by 1. Let C

stand for the word referenced by the address pointer in B. Then simultaneously move the type pointer from the word in A to the word in B; move the type pointer from the word in B to the C word; and move the type pointer from the C word to the word in A. Then go to step 16.

22. Let C stand for the word referenced by the address pointer in A. Then simultaneously move A to B; move B to the C word; and move the C word to A.

23. If the type pointer in B does not point to a 'ROW' pattern word then go to step 16.

24. The address pointer in B points to a descriptor of N dimensions where N is the number in the 'ROW' pattern word. Let L be the lower bound, U be the upper bound, and D be the stride. Increment the place in the first quadruple by one. Also increment the address pointer in the descriptor by the quadruple's stride. If the place is less than or equal to U-L then go to step 15. If the place is now greater than U-L then set the place to zero, decrement the address pointer in the descriptor by D(U-L+1) and repeat this step with the next quadruple if it exists.

25. If F is zero then link this array descriptor to the chain pointed to by the array chain pointer and go to step 16.

26. Unpack the state information in the descriptor and go to step 16.

27. If F is zero set to one all bits in the mark table corresponding to words in the interior of arrays chained to the array chain pointer.

28. Marking is complete. If F is zero set it equal to one and go to the compacting routine. If F is one then garbage collection is complete.

The compacting routine is based on the fact that there must be at least one free word between consecutive blocks of marked words. This free word can contain relocation information that is used to relocate pointers in the list structure [Haddon, B.K. & Waite, W.M., "A Compaction Procedure for Variable-length Storage Elements", Computer Journal 10, 2, 162-165 (August 1967)]. The algorithm for the compacting routine is as follows:

1. Assume that bits before the start of the bit table are ones and bits after the bit table are zero. Set pointer A to point to the word following the first 1-0 transition in the bit table.

2. Set pointer B to point to the word following the next 0-1 transition in the bit table. If there is no such rext transition then go to step 12.

3. Set pointer C to point to the word following the next 1-0 transition in the bit table.

4. Store in a word N words beyond the word pointed

to by A a relocation word containing the current contents of pointer B (old address) and pointer A (new address). Increment the value of N by one.

5. Set the pointer A1 equal to the pointer A and the pointer B1 equal to the pointer B.

6. Swap the two words referred to by pointers A1 and B1 then increment the pointers A1 and B1 so they refer to the next sequential word.

7. If the pointer in B1 does not equal the pointer in C then go to step 6.

8. If the stack/heap boundary was moved then store the new location of the stack/heap boundary in H. If pointer C is below the stack/heap boundary then copy pointer A1 into H.

9. Divide C-A by B-A and let the quotient be Q and the remainder be R. If N is less than or equal to R then move N words starting with location Q(B-A)+A to location B. If N is greater than R then move R words starting with Q(B-A)+A to location B+N-R. After the swap performed in step 6 the relocation table may be split into two parts and be anywhere in the free area. This step combines the two parts and puts the relocation table in a contiguous area at the bottom of the free area.

10. Set pointer A to the current value of pointer A1.

11. Go to step 2.

12. Call the monitor to allocate or release memory as appropriate. All memory that is in use is below pointer A. Enough extra memory must be included to satisfy the current request for memory and contain the bit table.

13. Allocate space at the top of memory for the new bit table. The base of the bit table is the new top of the heap.

14. Set the pointer A1 equal to the pointer H and the pointer B1 equal to the base of the bit table minus A plus H. This prepares swapping the heap up to the base of the bit table.

15. Swap the two words referred to by pointers A1 and B1 then increment the pointers A1 and B1 so they refer to the next sequential word.

16. If the pointer in B1 does not equal the base of the bit table then go to step 15.

17. Set the base of the heap pointer to the base of the bit table minus A plus H.

18. Join up the relocation table in a similar way as in step 9.

19. Sort the relocation table by old addresses then adjust heap references to reference the new location of the heap.

20. Compacting is complete. Go to step 1 in the

marking routine.

Although it is not done in the present implementation, the garbage collector is capable of printing out a complete symbolic dump of memory. This is possible because every pointer contains a type pointer that completely describes the data the pointer points to. Such a symbolic dump would be very useful for program debugging.

USE OF THE COMPILER

In the current implementation of ALGOL 68 no bold faced words are used. Instead, there is a list of reserved words that cannot be used as identifiers or indicants. Also, spaces are significant and separate successive indicants, reserved words, and identifiers. The following is a list of symbols used in the current implementation for the symbols defined in the ALGOL 68 report:

(	begin symbol
)	end symbol
E	times ten to the power symbol
%:=	over and becomes symbol
%::=	modulo and becomes symbol
OR	or symbol
AND	and symbol
/=	differs from symbol
<	is less than symbol
<=	is at most symbol
>=	is at least symbol
>	is greater than symbol
/	divided by symbol
%	over symbol
%:	modulo symbol
NOT	not symbol
ABS	absolute value of symbol
ODD	odd symbol

:/=:	is not symbol
AT	at symbol
(	if symbol
\	then symbol
\	else symbol
)	fi symbol
OF	of symbol
	go to symbol
SKIP	skip symbol
NIL	nil symbol
\:	then if symbol
\:	else if symbol
"	quote symbol
@	comment symbol

Since no bold face characters are used, certain ambiguities appear in the language that require additional conventions not found in the report. In particular, square brackets must be used for arrays. All casts must be inclosed in parentheses to distinguish them from labelled statements. Also the following list of reserved words cannot be used for identifiers or indicants:

LONG

STRUCT

REF

FLEX

EITHER

PROC

UNION

MODE

PRIORITY

LOC

OP

AT

OF

HEAP

LIBRARY

FOR

FROM

BY

TO

WHILE

DO

Oher predefined words may be used generally but if they are their predefined meaning will be lost in the range in which this is done.

Programs to be run in ALGOL 68 are first typed into the Dartmouth Timesharing System in the normal way. Every line starts with a line number but the compiler ignores it. The command "SYSTEM ALGOL68" is then typed followed by the command "RUN". This will cause the program to be compiled and run. An example of a valid program is:

```
1 (REAL X, Y;
2  X := 1; Y := 2;
3  X := X + Y)
```

If it is desired to precompile a procedure for inclusion in another program then the procedure declaration is written in the normal way and is followed by the characters ";SKIP". For example, the program:

```
1 PROC ADD = (REAL X, Y)REAL : X + Y; SKIP
```

will precompile a procedure whose result is the sum of its arguments. Since this program does not begin with a left parenthesis the compiler will not execute the program but instead write out the object machine code in a file called ".OBJECT.". This file should be renamed to the name of the procedure being declared, "ADD" in the example.

To include a precompiled procedure in another program there must be a declaration for it. The declaration

```
LIBRARY PROC(REAL, REAL)REAL ADD;
```

will add the file named ADD into the object code. It will also declare ADD to be a procedure that can be used in the program. A complete program using ADD might be:

```
1  (REAL A, B, C;
2  LIBRARY PROC(REAL, REAL)REAL ADD;
3  A:=1; B:=2;
4  C:=ADD(A, B))
```

CONCLUSIONS

The major innovations of ALGOL 68 are the mode concept with the idea of complete compile time knowlege of run time modes. The united modes are introduced to allow limited run time determination of modes. ALGOL 68 also has a well defined set of input and output procedures thus filling a major lack in ALGOL 60. Unfortunately, the implied garbage collector in ALGOL 68 costs a good deal of overhead even against users who do not use this facility. If only one construction in a program requires garbage collection then all operations in the program will be slowed down. The garbage collector precludes the use of ALGOL 68 as an implementation language because of its insistance on a known run time set of modes at compile time. This makes it impossible to work with data whose structure will be determined at run time. Also, there is no way to get at the individual bits of a machine word. This makes it impossible to define the structure of hardware defined word formats.

The major innovation of PL/1 is its concept of a based pointer. In most languages identifiers are associated with memory locations but in PL/1 identifiers are associated with a particular offset relative to an unspecified pointer. An identifier is used by associating it with a particular base pointer. Therefore, identifiers in PL/1 correspond roughly to structure tags in ALGOL 68. However, in PL/1, the same pointer or base may be associated with different structures permitting the same data to be accessed in different ways. This allows the programmer flexibility (and pitfalls) in referencing data. Unfortunately, the syntax of PL/1 is extremely messy and programs written in PL/1 are not mathematically "neat" like programs written in ALGOL.

SIMULA 67 combines the concepts of a procedure and a data structure. A procedure call such as f(a, b, c) can also be looked upon as initializing a three component data structure with the values a, b, and c. This is accomplished by allowing references to the parameters of a procedure by an independent procedure and not releasing space allocated for temporary storage of a procedure after it is exited. The first parameter of the above procedure could be referenced by writing f.a . If a procedure has no computation in it then it degenerates into a pure data structure. Coroutines are also easy to implement in SIMULA 67. It is unfortunate that ALGOL 68 does not allow such convenient accessing of structures or the ability of using coroutines.

APPENDIX

This appendix contains the actual syntax used for pass 1 and pass 2 together with a brief description of the imbedded actions. the name of each syntax rule appears to the left and is followed by a colon. The alternatives of a syntax rule are separated by semicolons and the last alternative is followed by a period. The elements in an alternative are separated by commas. Elements that are names of other production rules consist of this name. Actions that attempt to match the input program against a literal string consist of this string surrounded by single quotes. Other actions are underlined. If an action has an argument then this argument follows the underlined name of the action and is enclosed in brackets.

Following the syntax for each pass is an explanation of the actions. Names followed by (SYNTAX) are actions imbedded in the syntax. Names followed by (SUBROUTINE) are subroutines used by the actions.

PASS 1 SYNTAX

```
PROG:   PRO,PASS1.5,PASS2,DONE.

PRO:    PDEN,DEL;

        SRNGE,[CLOR],CLEAR,[IDNT],MRNGE,SER,DECI,ERNGE,

        [CLOR].

COND:   '\',DECI,ERNGE,[BARR],SRNGE,[BARR],MRNGE,SER,

        ELSE;

        '\:',DECI,ERNGE,[BARR],SRNGE,[BARR],MRNGE,SER,

        THEN;

        EMPTY.

THEN:   '\',DECI,ERNGE,[BARR],SRNGE,[BARR],MRNGE,SER,

        ELSE;

        '\:',DECI,ERNGE,[BARR],SRNGE,[BARR],MRNGE,SER,

        THEN;

        ERROR8.

ELSE:   '\',DECI,ERNGE,[BARR],SRNGE,[BARR],MRNGE,SER;

        '\:',DECI,ERNGE,[BARR],SRNGE,[BARR],MRNGE,SER,

        THEN;

        EMPTY.

SER:    UNIT,UTAIL.

UTAIL:  ';',ADO,[SEMIC],UNIT,UTAIL;

        ':',ADO,[CLNC],DECLC,UNIT,UTAIL;

        '.',ADO,[SEMIC],IDENT,':',STID,DECL,UNIT,UTAIL;

        JTAIL.

UNIT:   STZ,[IDNT],DEC.

JUNKE:  JUNK;
```

```
         EMPTY.
JUNK:    IDENT,STIDB,JUNKE;
         '=',SMO,[IDNT],JUNK;
         ':=',SMO,[IDNT],JUNKE;
         PDEN,DEL,SMO,[IDNT],JUNKE;
         BOX,DEL,SMO,[IDNT],JUNKE;
         'STRUCT',SUNIT,DEL,SMO,[IDNT],JUNKE;
         'UNION',UUNIT,DEL,SMO,[IDNT],JUNKE;
         'PROC',ARGE,SMO,[IDNT],JUNKE;
         'REF',SMO,[IDNT],JUNK;
         'LOC',SMO,[IDNT],JUNK;
         'HEAP',SMO,[IDNT],JUNK;
         'OF',SMO,[IDNT],JUNK.
FJ:      FORS;
         JUNKE.
JTAIL:   ',',ADO,[COMAC],SMO,[IDNT],JTL1;
         EMPTY.
JTL1:    FORS,JTAIL;
         JUNKE,JTAIL.
SUNIT:   '(',PUSH,[IDNT],MARK,FSEQ,')',CST,POP,[IDNT].
UUNIT:   '(',PUSH,[IDNT],MARK,MSEQ,')',CUN,POP,[IDNT].
FORS:    'FOR',IDENT,FROMS;
         FROMS.
FROMS:   'FROM',JUNK,BYS;
         BYS.
BYS:     'BY',JUNK,TOS;
```

```
        TOS.

TOS:    'TO',JUNK,WHLES;

        WHLES.

WHLES:  'WHILE',SER,DOS;

        DOS.

DOS:    'DO',SMO,[IDNT],FJ.

BOX:    '[',SRNGE,[BOXR],BELEM,BTAIL,']',CBOX,ERNGE,

        [BOXR],REST,[CBOXT].

BTAIL:  ',',ADO,[COMAC],BELEM,BTAIL;

        EMPTY.

BELEM:  STZ,[IDNT],JUNKE,BEL2.

BEL2:   ':',ADO,[CLNC],STZ,[IDNT],JUNKE,BEL3;

        'AT',ADO,[CLNC],STZ,[IDNT],JUNK;

        'FLEX',':',ADO,[CLNC],STZ,[IDNT],JUNKE,BEL4;

        'EITHER',':',ADO,[CLNC],STZ,[IDNT],JUNKE,BEL4;

        STZ,[IDNT].

BEL3:   'AT',JUNK;

        BEL4.

BEL4:   'FLEX';

        'EITHER';

        EMPTY.

DEC:    'PRIORITY',IDENT,STID,'=',PRT3;

        'OP',OPDEC;

        'MODE',IDENT,STID,'=',MT3;

        'STRUCT',SDEC;

        'UNION',UDEC;
```

```
         'PROC',PDEC;
         'REF',RESLT,EREF,IDEC;
         BOX,RESLT,EBOX,IDEC;
         'LOC',RESLT,IDEC;
         'HEAP',RESLT,IDEC;
         'LIBRARY',RESLT,SAVE,[FMODE],DECLB,LIBL;
         IDENT,STID,ID;
         JTL1.
PRT:     ',',ADO,[COMAC],PRT1;
         EMPTY.
PRT1:    IDENT,STID,PRT2;
         SMO,[IDNT],DEC.
PRT2:    '=',PRT3;
         ID.
PRT3:    PUSH,[IDNT],IDENT,CNVRT,POP,[IDNT],DECP,PRT.
OPDEC:   '(',MARK,MSEQ,')',FIELD,EPR,DECO,DELS,'=',FJ,
         OPTL;
         IDENT,STID,EVOID,DECO,'=',PDEN,DECOP,OPTL;
         '=',STID,EVOID,DECO,'=',PDEN,DECOP,OPTL;
         ERROR3.
OPTL:    ',',ADO,[COMAC],OPT1;
         EMPTY.
OPT1:    IDENT,STID,OPT2;
         OPDEC;
         SMO,[IDNT],DEC.
OPT2:    '=',JUNK,OPTL;
```

```
        ID.

MTAIL:  ',',ADO,[COMAC],MT1;

        EMPTY.

MT1:    IDENT,STID,MT2;

        SMO,[IDNT],DEC.

MT2:    '=',MT3;

        ID.

MT3:    PUSH,[IDNT],MOD,POP,[IDNT],DECM,MTAIL.

SDEC:   SUNIT,IDM;

        IDENT,STID,'=',SUNIT,DECM,STAIL;

        ERROR5.

STAIL:  ',',ADO,[COMAC],ST1;

        EMPTY.

ST1:    IDENT,STID,ST2;

        SMO,[IDNT],DEC.

ST2:    '=',SUNIT,DECM,STAIL;

        ID.

UDEC:   UUNIT,IDM;

        IDENT,STID,'=',UUNIT,DECM,UNT;

        ERROR6.

UNT:    ',',ADO,[COMAC],UT1;

        EMPTY.

UT1:    IDENT,STID,UT2;

        SMO,[IDNT],DEC.

UT2:    '=',UUNIT,DECM,UNT;

        ID.
```

```
PDEC:   IDENT,STID,PDEC1;
        MARK,ARGE,RESLT,EPR,IDEC.
PDEC1:  '=',PUSH,[IDNT],PDEC2,POP,[IDNT],TAG,PTE;
        ':=',PUSH,[IDNT],PDEC2,POP,[IDNT],EREF,TAG,PTA;
        IDENT,MARK,MMI,EPR,STID,IDEC;
        MARK,EVOID,EPR,IDEC.
PDEC2:  PDEN;
        JUNKE,MARK,EVOID,EPR.
RESLT:  'PROC',MARK,ARGE,RESLT,EPR;
        BOX,RESLT,EBOX;
        'REF',RESLT,EREF;
        'STRUCT',SUNIT,IDE;
        'UNION',UUNIT,IDE;
        IDENT,STID,RES1.
RES1:   IDENT,MMI,STID;
        EVOID.
IDE:    IDENT,STID;
        SMO,[IDNT].
PTE:    ',',ADO,[COMAC],PTE1;
        EMPTY.
PTE1:   IDENT,STID,PTE2;
        SMO,[IDNT],DEC.
PTE2:   '=',PUSH,[IDNT],PDEN,POP,[IDNT],TAG,PTE;
        ID.
PTA:    ',',ADO,[COMAC],PTA1;
        EMPTY.
```

```
PTA1:   IDENT,STID,PTA2;
        SMO,[IDNT],DEC.
PTA2:   ':=',PUSH,[IDNT],PDEN,POP,[IDNT],EREF,TAG,PTA;
        ',',ADO,[COMAC],MARK,EVOID,EPR,TAG;
        ID.
MOD:    'REF',MOD,EREF;
        BOX,MOD1,EBOX;
        'STRUCT',SUNIT;
        'UNION',UUNIT;
        IDENT,STID,MMI;
        'PROC',MARK,ARGE,MOD1,EPR.
ARGE:   '(',MSEQ,')';
        EMPTY.
MOD1:   MOD;
        EVOID.
FIELD:  'REF',SMO,[FMODE],FIELD,EREF,SAVE,[FMODE];
        BOX,SMO,[FMODE],FIELD,EBOX,SAVE,[FMODE];
        'STRUCT',SUNIT,IDENT,STID,SAVE,[FMODE];
        'UNION',UUNIT,IDENT,STID,SAVE,[FMODE];
        'PROC',MARK,ARGE,FLD2,EPR,SAVE,[FMODE];
        IDENT,STID,FLD4,SAVE,[FMODE];
        ERROR7.
FLD2:   IDENT,STID,FLD3;
        FIELD.
FLD3:   IDENT,MMI,STID;
        EVOID.
```

```
FLD4:   IDENT,MMI,STID;
        REST,[FMODE].
MSEQ:   MOD,MSEQ1.
MSEQ1:  ',',MOD,MSEQ1;
        EMPTY.
FSEQ:   FIELD,TAG,FSEQ1.
FSEQ1:  ',',FIELD,TAG,FSEQ1;
        EMPTY.
IDM:    IDENT,STID,IDEC;
        JUNKE,DEL,JTAIL.
ID:     IDENT,MMI,STID,IDEC;
        JUNKE,JTAIL.
CTAIL:  ',',ADO,[COMAC],DEC,CTAIL;
        EMPTY.
IDEC:   '=',SAVE,[FMODE],TAG,JUNKE,ETL;
        ':=',EREF,SAVE,[FMODE],TAG,JUNKE,ATL;
        JUNK,DEL,JTAIL;
        EREF,SAVE,[FMODE],TAG,ATL.
ETL:    ',',ADO,[COMAC],ET1;
        SMO,[IDNT].
ET1:    IDENT,STID,ET2;
        SMO,[IDNT],DEC.
ET2:    '=',REST,[FMODE],TAG,JUNKE,ETL;
        ID.
ATL:    ',',ADO,[COMAC],AT1;
        SMO,[IDNT].
```

```
AT1:    IDENT,STID,AT2;
        SMO,[IDNT],DEC.
AT2:    ':=',REST,[FMODE],TAG,JUNKE,ATL;
        ',',ADO,[COMAC],REST,[FMODE],TAG,AT1;
        IDENT,MMI,STID,IDEC;
        JUNK,JTAIL;
        REST,[FMODE],TAG.
LIBL:   ',',ADO,[COMAC],LIB1;
        SMO,[IDNT].
LIB1:   IDENT,STID,LIB2;
        SMO,[IDNT],DEC.
LIB2:   ',',ADO,[COMAC],REST,[FMODE],DECLB,LIB1;
        IDENT,MMI,STID,IDEC;
        JUNK,JTAIL;
        REST,[FMODE],DECLB.
PDEN:   '(',SRNGE,[CLOR],PUSH,[MK],MRNGE,SER,COND,')',
        MOD1,PBOD,EPDEN.
PBOD:   ':',JUNKE,STZ,[PDPOS];
        SMO,[PDPOS].
```

PASS 1 ACTIONS

EMPTY (SYNTAX)

No action. This action is used when no action is to be performed and there are no other actions in the alternative.

PASS1.5 (SYNTAX)

This action is pass 1.5. When pass 1.5 finishes, control is returned to the syntax analyzer.

PASS2 (SYNTAX)

This is the top production rule for pass 2. The name of the rule is 'PRO' in pass 2 syntax.

DONE (SYNTAX)

This action is called when pass 2 is finished. The syntax analyzer is exited and pass 3 is executed.

STZ (SYNTAX)

This routine stores a zero in the upper half of the word referred to by the argument.

SMO (SYNTAX)

This routine stores an octal 400000 in the upper half of the word referred to by the argument.

CLEAR (SYNTAX)

This routine stores a zero in the word referred to by the argument.

STIDB (SYNTAX)

This routine examines location 1$IDNT. If this location is zero, routine STID is transferred to. Otherwise, an octal 400000 is stored in the upper half of location 1$IDNT.

STID (SYNTAX)

This routine stores the contents of XR - 7 in the upper half of location 1$IDNT.

DECLC (SYNTAX)

This routine checks the contents of location 1$IDNT. If this location is negative, an immediate exit is made. Otherwise, routine DECL is transferred to.

DEL (SYNTAX)

The top word in the working stack is deleted. The word that was second from the top is now the top word.

PUSH (SYNTAX)

The word referred to by the argument is pushed onto the control stack. The old top of stack is now second from the top.

POP (SYNTAX)

The word on the top of the control stack is stored in the location referred to by the argument. The top stack word is deleted so that the old second from the top word becomes the top of stack word.

DELS (SYNTAX)

The top word in the control stack is deleted. The word that was second from the top is now the top word.

MARK (SYNTAX)

The current length of the working stack is pushed onto the control stack.

MRNGE (SYNTAX)

The current length of the working stack is stored in location 1$MK.

EVOID (SYNTAX)

A void mode declarer is pushed onto the working stack. A void mode declarer consists of a pointer to the void entry in the mode table in the upper half and zero in the lower half.

SAVE (SYNTAX)

The top word of the working stack is copied to the location referred to by the argument. The working stack is not altered.

REST (SYNTAX)

The word referred to by the argument is pushed onto the working stack.

TAG (SYNTAX)

The contents of 1$IDNT is pushed onto the working stack.

IDENT (SYNTAX)

The next symbol in the input stream is checked. If it is a reserved symbol this routine gives a nomatch return. Otherwise, the next symbol is accepted and a pointer to the STAB table entry of the symbol is placed in XR - 7.

CST (SYNTAX)

The top word of the control stack is assumed to be the length of the working stack stored there by the MARK routine. This word is deleted from the control stack and the words added to the working stack since the marking are considered. The considered words refer to the fields of a structure when taken in pairs. The first word of the pair is a declarer word for the field and the second word contains in the upper half a pointer to the STAB table where the tag for the field is stored. All of the considered words are deleted from the working stack and are replaced by a single declarer word for the stuctured mode having the fields specified by the considered words.

CNVRT (SYNTAX)

The last symbol accepted is checked. If it is not a single digit between "1" and "9" a fatal error occurs. Otherwise, the value of the digit is pushed onto the working stack in the upper half of the word.

EREF (SYNTAX)

The top word in the working stack is assumed to be a declarer. It is replaced by a declarer whose mode is reference to the mode of the original declarer.

EPDEN (SYNTAX)

The word 1$PDPOS is examined. If the word is positive then the next paragraph explains the rest of the routine. If the word is negative then the top word of the working stack is deleted. Subroutine DECIA is called. Location 1$MK contains the old length of the working stack and words are deleted from the working stack until its length is the same as the stored in 1$MK. The top word of the control stack is popped and stored in 1$MK. Subroutine ERNGA is called with a parameter of CLOR. A void declarer is pushed onto the working stack and the routine exits.

If 1$PDPOS is positive then SEMIC and CLNC must be zero or a fatal error occurs. Consider the words added to the working stack since the last mark. They are declarers for the formal parameters followed by a declarer for the mode of the result. The top word of the working stack is deleted and remembered. All of the other considered words are deferenced (if not reference mode a fatal error occurs) then subroutine DECIA is called. The contents of 1$MK is pushed onto the control stack. The remembered word is restored on top of the working stack. The considered words with the result word are replaced with a single word declarer of the mode procedure with parameters by a call to the subroutine EPROC. The top of the control stack is popped and stored in 1$MK. The declarer on the top of the working stack is remembered and deleted. Subroutine ERNGA is called with a parameter of the mode of the remembered declarer with the sign bit set. The remembered declarer is restored on the working stack and a normal exit occurs.

MMI (SYNTAX)

A declarer is pushed onto the working stack referring to the symbol referred to by the contents of location 1$IDNT. This is done by creating entries in the mode and bound tables that refer to the symbol and range. A later routine will replace these entries by the actual mode that the symbol refers to.

EPROC (SUBROUTINE)

Words added to the working stack since the mark which is stored on top of the control stack are considered to be declarers for the arguments and the result of the procedure mode. All of these words on the working stack are deleted and replaced with a single word which is a declarer for the indicated procedure mode. The mark word in the control stack is deleted. A return is then made to the calling routine.

EPR (SYNTAX)

A call is made on EPROC.

CUN (SYNTAX)

Words added to the working stack since the mark which is stored on top of the control stack are considered to be declarers for the component modes of a united mode. All of these words on the working stack are delted and replaced with a single word which is a declarer for the indicated united mode. The mark word in the control stack is deleted.

DECM (SYNTAX)

An entry is made in the DEF table defining the symbol in 1$IDNT to be a mode indication for the mode specified by the declarer stored in the top word of the working stack. The

DEF table entry is linked to the other DEF table entries for the same symbol and a note as to the lexicographical level of the symbol is made. The declarer on top of the working stack is deleted.

DECP (SYNTAX)

An entry is made in the DEF table defining the symbol in 1$IDNT to be a priority indication of the priority specified by the number stored in the top word of the working stack. The DEF table entry is created by a call to the subroutine SETD. The pointer on top of the working stack is deleted.

DECO (SYNTAX)

An entry is made in the DEF table defining the symbol in 1$IDNT to be an operator with the mode specified by the declarer stored in the top word of the working stack. The DEF table entry is created by a call to the subroutine SETD. The declarer on top of the working stack is deleted. A pointer to the DEF table entry is pushed onto the control stack.

DECOP (SYNTAX)

The top word on the control stack is a pointer to a DEF table operator definition entry. This word is popped from the control stack. The top word of the working stack is a declarer. It is popped from the working stack and stored in the DEF table entry pointed to by the word popped from the control stack as the mode of the operator.

DECI (SYNTAX)

A call is made on subroutine DECIA. Words are then deleted from the working stack until the length of the working stack equals the length stored in 1$MK.

DECIA (SUBROUTINE)

All words added to the working stack beyond the length stored in 1$MK are considered. These words are taken in pairs: the first word of a pair is a declarer and the second is a pointer to a symbol in the STAB table. For each pair an entry is made in the DEF table defining the symbol to be an identifier having the corresponding declarer. The DEF table is then linked to the other DEF table entries for the same symbol and a note as to the lexicographical level of the symbol is made.

DECL (SYNTAX)

A label declarer is pushed onto the working stack. Then the contents of 1$IDNT is pushed onto the working stack.

SETD (SUBROUTINE)

This subroutine assumes that XR - 1 points to a newly allocated entry in the DEF table and that location 1$IDNT contains in the upper half of a pointer to the STAB table to the symbol currently under consideration and in the lower half its lexicographical level. The current DEF table entry is linked to other DEF table entries for the same symbol. The chain of DEF table entries for a given symbol are ordered in decreasing order of lexicographical level and in the order MODE, OP, PRIOR, IDENT within the same lexicographical level. A note of the current lexicographical level is made in the currnt DEF table entry.

CBOX (SYNTAX)

The top word of the control stack contains the length of the working stack when it is marked. This top word of the control stack is deleted and the words added to the working stack are considered. These words contain bounds information contained in a row declarer. This bounds information is deleted from the working stack and stored in an entry in the BOUND table. Subroutine ERNGA is called with a parameter of BOXR. A pointer to the newly created BOUND table entry is then pushed onto the working stack.

EBOX (SYNTAX)

The top word of the working stack is a declarer and the second from the top word is a pointer to an entry in the BOUND table. These two words are deleted from the working stack and a declarer consisting of a pointer to the BOUND table entry and a pointer to a newly created entry in the mode table is pushed onto the working stack. The newly created entry in the mode table is the mode row of (n times as indicated by the BOUND table entry) followed by the mode referred to in the declarer that was on top of the working stack.

ADO (SYNTAX)

One is added into the upper half of the word specified by the argument.

MBND (SYNTAX)

A word made up of the upper half of the location 1$IDNT and the argument is pushed onto the working stack.

SRNGE (SYNTAX)

An entry is made in the PROG table containing the argument. Then the contents of the locations 1$COMAC, 1$CLNC, 1$CURR, 1$SEMIC, and 1$IDNT are successively pushed onto the working stack and set to zero. A pointer to the PROG table entry is then stored in the upper half of 1$CURR and the lower halves of 1$LEVEL and 1$IDNT.

ERNGE (SYNTAX)

The subroutine ERNGA is called with the argument as the argument.

ERNGA (SUBROUTINE)

The pointer in 1$CURR is used to locate an entry in the PROG table. The argument is stored in the PROG table entry. So is the contents of 1$IDNT, 1$SEMIC, 1$CLNC, and 1$COMAC. Then the locations 1$IDNT, 1$SEMIC, 1$CURR, 1$CLNC, and 1$COMAC are restored from successive words popped from the working stack. Then the lower half of 1$IDNT is stored in the PROG table entry.

PASS 2 SYNTAX

```
PRO:    START,CLO,VOID,REST,[FMODE],SVAL,STRNG,DECX;
        SRNGE,STZ,[RNGE],SER,VOID,EVOID,DELV,ERNGE.
CLO:    LPAR,[PROC],SRNGE,INLL,PUSH,[T$CODE],MARK,FORMP,
        ')',PMOD,PUSEA,[DECLR],EPR,PUSH,[DECLR],':',QUAT,
        EPDEN,ORLL,ERNGE,EPDNE;
        LPAR,[PARALLEL],SRNGE,INLL,PUSH,[T$CODE],QS,')',
        PARN,ORLL,ERNGE;
        LPAR,[CAST],SRNGE,INLL,PMOD,':',QUAT,')',SVAL,
        STRNG,ORLL,ERNGE;
        LPAR,[SERIAL],SRNGE,INLL,SER,')',ORLL,ERNGE;
        LPAR,[ANY],SRNGE,INLL,IF,')',ORLL,ERNGE.
RLTR:   '::',CSCT;
        '::=',CSCTB.
SER:    PUSH,[T$CODE],TRAIN,TRTL,BALN.
TRTL:   '.',ADO,[CNT],SER;
        EMPTY.
TRAIN:  UNIT,UTAIL.
UTAIL:  ';',VOID,DELV,ODELV,TRAIN;
        EMPTY.
IF:     CPAR,PUSH,[T$CODE],PUSH,[T$CODE],CONF,TSEQ,RLTR,
        TERT,ENTL,[B1],CONE,CLEAR,[CNT],BALZR,SVAL,FIRM,DELV,
        ODELV,THEN,ELSE,BAL2;
        PUSH,[T$CODE],SER,DIF,THEN,ELSE,BAL2.
THEN:   BAR,[CONFORMITY],XRNGE,PUSH,[T$CODE],QSEQ,
        BALZR;
        BAR,[CASE],XRNGE,SER;

        BARF,[ANY],XRNGE,IF,ESKIP.

ELSE:   BAR,[CASE],XRNGE,SER,ENTL,[B2];

        BARF,[CASE],XRNGE,PUSH,[B2X],IF,POP,[B2X],ENTL,

        [B2];

        ESKIP,ENTL,[B2].

QSEQ:   QUAT,ENTL,[B2],QSEQT.

QSEQT:  ',',ADO,[CNT],QSEQ;

        EMPTY.

QS:     QUAT,QST.

QST:    ',',ADO,[CNT],QS;

        EMPTY.

TSEQ:   TERT,ENTL,[B1],ENTC,TSEQT.

TSEQT:  ',',ADO,[CNT],TSEQ;

        EMPTY.

UNIT:   DEC;

        LSEQ,QUAT.

LSEQ:   LABEL,LSEQ;

        EMPTY.

QUAT:   'FOR';

        TERT,QTAIL.

QTAIL:  ':=',SASGN,QUAT,DASGN;

        '::',SCT,TERT,DCT;

        '::=',SCTAB,TERT,DCTAB;

        ':=:',SIS,TERT,IDNTY;

        ':/=:',SISNT,TERT,IDNTY;
        EMPTY.

TERT:   MFOR,STAIL.

STAIL:  OPER,FOP,TERT,STAIL;

        FOPS.

MFOR:   SEC;

        OPER,WMOP,MFOR,MOP.

SEC:    TAGOF,SEC,SELCT;

        AMOD,SHEAP,EREF,WGEN;

        'HEAP',AMOD,SHEAP,EREF,WGEN;

        'LOC',AMOD,SLOC,EREF,WGEN;

        PRIM;

        CLO,ACTP.

PRIM:   'SKIP',ESKIP;

        'NIL',ENIL;

        'TRUE',ETRUE;

        'FALSE',EFALS;

        DENOT,WDEN,ITAIL;

        IDENT,WIDEN,ITAIL.

ITAIL:  '[',SVAL,WEAK,OSUB,SRNGE,INLL,INDEX,XTAIL,']',

        OBUS,ORLL,ERNGE;

        ACTP.

ACTP:   LPAR,[PARAMETER],SVAL,FIRM,SRNGE,PUSH,[TMODE],

        INLL,PUSH,[T$CODE],QS,')',PARN,ORLL,CALL,ERNGE,ACTP;

        EMPTY.

XTAIL:  ',',ADO,[CNT],INDEX,XTAIL;

        EMPTY.
```

```
INDEX: BOUND,IX1;

       ':',IX2;

       IX5;

       EPTY.

IX1:   ':',LWB,IX3;

       IX5;

       SBCT,ADO,[SBCNT].

IX2:   BOUND,UPB,IX4;

       IX5;

       EPTY.

IX3:   BOUND,UPB,IX4;

       IX4.

IX4:   IX5;

       EMPTY.

IX5:   'AT',BOUND,NLWB.

BOUND: TERT,SINT,DELV.

VMOD:  MOD,SVIRT,CLEAR,[DFLG].

AMOD:  MOD,SACT,CLEAR,[DFLG].

MOD:   BOX,MOD,EBOX;

       'REF',PUSH,[DFLG],VMOD,EREF,POP,[DFLG];

       'STRUCT',SUNIT;

       'UNION',UUNIT;

       'PROC',PUSH,[DFLG],PROCT,POP,[DFLG];

       MIND,SDCLR.

PROCT: PROCM;

       PMOD,MARK,PUSEA,[DECLR],EPR.
```

```
PROCM: LPAR,[ANY],PUSH,[DFLG],CLEAR,[DFLG],MARK,MSEQ,
       ')',PMOD,PUSEA,[DECLR],EPR,POP,[DFLG].

PMOD:  VMOD;

       EVOID.

BOX:   '[',SRNGE,DSUB,BOXER,BTAIL,CBOX,']',DBUS,ERNGE.

BOXER: BOUND,OLWB,FOPT,':',BOUND,OUPB,FOPT,SACT;

       ':',SVIRT;

       SVIRT.

FOPT:  'FLEX',OFLEX;

       OFIX.

BTAIL: ',',ADO,[CNT],BOXER,BTAIL;

       EMPTY.

SUNIT: LPAR,[ANY],EVOID,MARK,FSEQ,')',CST.

UUNIT: LPAR,[ANY],PUSH,[DFLG],CLEAR,[DFLG],MARK,MSEQ,
       ')',CUN,POP,[DFLG].

MSEQ:  VMOD,PUSEA,[DECLR],MODT.

MODT:  ',',MSEQ;

       EMPTY.

FSEQ:  FIELD,PUSHW,[DECLR],PUSHW,[TG],FSEQ1.

FSEQ1: ',',FSEQ;

       EMPTY.

FIELD: MIND,FLD1;

       MOD,TAG;

       TAG.

FLD1:  TAG,SDCLR;

       EMPTY.
```

```
PARAM: QUAT.

PTAIL: ',',ADO,[CNT],PARAM,PTAIL;

       EMPTY.

FORMP: VMOD,IDENT,PUSEA,[DECLR],OFORM,FTL.

FTL:   ',',FT1;

       EMPTY.

FT1:   IDENT,PUSEA,[DECLR],OFORM,FTL;

       FORMP.

DEC:   'STRUCT',CLEAR,[DFLG],ST1;

       'UNION',UN1;

       'MODE',MD1;

       'PRIORITY',PR1;

       'OP',OPDEC;

       'PROC',PDT;

       MOD,SBLNK,IDEC;

       'HEAP',MOD,SHEAP,IDEC;

       'LOC',MOD,SLOC,IDEC;

       'LIBRARY',VMOD,IDENT,LIBL.

PDT:   IDENT,STID,GMOD,IDEC1;

       PROCT,SBLNK,IDEC.

ST1:   SUNIT,SBLNK,IDEC;

       MIND,'=',SUNIT,SACT,CLEAR,[DFLG],STL.

STL:   ',',ST2;

       ';',TRAIN.

ST2:   MIND,'=',SUNIT,SACT,CLEAR,[DFLG],STL;

       DEC.
```

```
UN1:    UUNIT,SBLNK,IDEC;

        MIND,'=',UUNIT,UTL.

UTL:    ',',UT2;

        ';',TRAIN.

UT2:    MIND,'=',UUNIT,UTL;

        DEC.

MD1:    MIND,'=',AMOD,MTAIL.

MTAIL:  ',',MD2;

        ';',TRAIN.

MD2:    MD1;

        DEC.

PR1:    TAG,'=',DENOT,PRT.

PRT:    ',',PR2;

        ';',TRAIN.

PR2:    PR1;

        DEC.

OPDEC:  PROCM,OPER,STID,OIDNY,OIDN,'=',QUAT,REST,

        [FMODE],SVAL,STRNG,EVOID,OENTR,DELV,OPTL;

        OPER,STID,OIDNY,OIDN,'=',QUAT,REST,[FMODE],SVAL,

        STRNG,EVOID,OENTR,DELV,OPTL.

OPTL:   ',',OPT1;

        ';',TRAIN.

OPT1:   OPDEC;

        DEC.

IDEC:   IDENT,STID,SAVE,[FMODE],IDEC1;

        SHPBK,SACT,CLEAR,[DFLG],EREF,WGEN,STAIL,QTAIL.
```

```
IDEC1: '=',SVIRT,CLEAR,[DFLG],OIDN,QUAT,REST,[FMODE],
       SVAL,STRNG,DECX,ETL;
       SACT,CLEAR,[DFLG],EREF,SAVE,[FMODE],AT2.

ETL:   ',',ET1;
       ';',TRAIN.

ET1:   IDENT,STID,'=',OIDN,QUAT,REST,[FMODE],SVAL,
       STRNG,DECX,ETL;
       DEC.

ATL:   ',',AT1;
       ';',TRAIN.

AT1:   IDENT,STID,AT2;
       DEC.

AT2:   ':=',OIDN,REST,[FMODE],OASGN,SLCBK,WGEN,QUAT,
       DEREF,SVAL,STRNG,ASGNE,DELV,DECX,ATL;
       OIDN,REST,[FMODE],SLCBK,WGEN,DECX,ATL.

LIBL:  ',',LIB1;
       ';',TRAIN.

LIB1:  IDENT,LIBL;
       DEC.

DECX:  EVOID,OENTR,DELV.
```

PASS 2 ACTIONS

EMPTY (SYNTAX)

No action. This action is used when no action is to be performed and there are no other actions in the alternative.

START (SYNTAX)

This action peeks at the next (first) input symbol. If it is not a left parenthesis then the action returns 'FAIL'. Otherwise, DECLR and FMODE are set to 'PROCEDURE VOID', the last identifier encountered is claimed to be all blanks, the flag LIBF is set to indicate this is a program and not a subroutine, and the action OIDN is executed.

SRNGE (SYNTAX)

The contents of 2$RNGE is pushed onto the control stack. Then the contents of the locations 2$CNT, 2$DECLR, 2$FMODE, 2$B1X, 2$B1S, 2$B2X, 2$B2S, 2$L1, and 2$OPT are successively pushed onto the control stack and set to zero. The lower half of 2$RNGE contains the highest range number encountered. This number is incremented to the next range number and stored in both halves of 2$RNGE as the new range number and the maximum range number.

XRNGE (SYNTAX)

Location 2$CNT is zeroed. The lower half of 2$RNGE contains the highest range number encountered. This number is incremented to the next range number and stored in both halves of 2$RNGE as the new range number and the maximum range number.

ERNGE (SYNTAX)

The locations 2$OPT, 2$L1, 2$B2S, 2$B2X, 2$B1S, 2$B1X, 2$FMODE, 2$DECLR, and 2$CNT are restored from successive words popped from the control stack. Then the next word is popped from the control stack and the upper half of it is stored in the upper half of 2$RNGE as the new range number.

TAG (SYNTAX)

The next symbol in the input stream is checked. If it is a reserved word this routine gives a nomatch return. Otherwise, the next symbol is accepted and a pointer to the STAB table entry of the symbol is stored in 2$TG.

OPER (SYNTAX)

TLUG is called with D$OP as an argument.

MIND (SYNTAX)

TLUG is called with D$MODE as an argument.

IDENT (SYNTAX)

TLUG is called with D$IDENT as an argument.

DENOT (SYNTAX)

The next symbol in the input stream is checked. If it is a reserved word this routine gives a nomatch return. If it is not a real, integer, format, character, or string denotation a nomatch return is given. A two word representation of the denotation is calculated. An entry is created in the DEF table containing the mode of the denotation and the two word representation of its value. The denotation in the input stream is then accepted.

TLUG (SUBROUTINE)

The next two symbols in the input stream are checked. If the second symbol is "OF" then a nomatch return is given. Otherwise, the first symbol is looked up in the symbol table entry for the first definition visible from the current range. If the type of symbol table entry is not the same as the argument to this routine a nomatch return is given. Otherwise, a pointer to the DEF table entry is saved in 2$LASTS, a pointer to the symbol is saved in 2$TG, and the symbol is accepted.

LPAR (SYNTAX)

The next symbol in the input stream is checked. If it is not a left parenthesis or if it does not have the attributes specified by the argument a nomatch return is given. Otherwise, the symbol is accepted.

BAR (SYNTAX)

The next symbol in the input stream is checked. If it is not a vertical bar or if it does not have the attributes specified by the argument a nomatch return is given. Otherwise, the symbol is accepted.

BARF (SYNTAX)

The next symbol in the input stream is checked. If it is not a vertical bar colon or if it does not have the attributes specified by the argument a nomatch return is given. Otherwise, the symbol is accepted.

CPAR (SYNTAX)

If the current range has no commas, semicolons, or colons then a nomatch return is given. Otherwise, a normal return is given.

TAGOF (SYNTAX)

The next two symbols in the input stream are checked. If the second symbol is not "OF" then a nomatch return is given. Otherwise, the first symbol is pushed into the control stack and both input symbols are accepted.

LABEL (SYNTAX)

The next two symbols in the input stream are checked. If the second symbol is not ":" then a nomatch return is given. Otherwise, the first symbol is looked up in the symbol table for the first definition visible from the current range.If the entry is not an identifier then a nomatch return is given. Otherwise, CAD is called with the argument [O$LBL, value of label found in the table entry]. Then both input symbols are accepted.

SASGN (SYNTAX)

SOFT is called with a pointer to the top control block in the working stack. The mode of the result is stored in the control stack. Then INS is called with the argument [O$ASGN, mode] and [-1, 1] is added to the LOC/LEN word in the value control block on top of the working stack.

DASGN (SYNTAX)

The mode is popped from the control stack. This mode is dereferenced and stored as a target mode. STRNG is called with a pointer to the top value control block in the working stack and saved target mode as arguments. The top two control blocks in the working stack are combined and the mode of the resulting control block is set equal to the saved mode. CAD is called with the argument [O$ASGNE, saved mode]. Then [0, 1] is added to the LOC/LEN word in the value control block on top of the working stack.

ASGNE (SYNTAX)

CAD is called with the argument [O$ASGNE, contents of 2$DECLR].

SCTAB (SYNTAX)

SOFT is called with a pointer to the top control block in the working stack as an argument. Then INS is called with the following arguments (mode is the result of SOFT and *n are generated labels):

```
[O$JUMP, *1]
[O$MREF, 0]
[O$LBL, *2]
[O$CONF, mode]
[O$TF, *3]
[O$MREF, 0]
```

and CAD is called with the following arguments:

```
[O$CASGN, mode]
[O$TRUE, 0]
[O$JUMP, *4]
[O$LBL, *1]
[O$MAX, 0]
```

The label *2 is stored in the control stack.

DCTAB (SYNTAX)

CAD is called with the following arguments:

```
[O$CONE, 0]
```

```
[O$JUMP, *2]
[O$LBL, *3]
[O$DELV, mode]
[O$FALSE, 0]
[O$LBL, *4]
```

The top two control blocks are combined into one and its mode is set to boolean. The symbols *2, *3, and *4 have the same value they did when the corresponding SCTAB code was processed.

SISNT (SYNTAX)

O$ISNT is pushed onto the control stack.

SIS (SYNTAX)

O$IS is pushed onto the control stack.

IDNTY (SYNTAX)

A new control block of type W$BAL keying 2 values is created and it is made to refer to the code word popped from the control stack. BB is called with a pointer to the newly created control block. This will calculate the target mode for both keyed values. Then C and DC are called with this mode as target mode for both values. The top three control blocks are combined into a single control block and its mode is set equal to boolean.

ADD (SUBROUTINE)

The argument is inserted into the output code after the code for the control block pointed to by VALP (or the current control block). The following code is moved one word to make room for it.

INS (SUBROUTINE)

The argument is inserted into the output code in front of the code for the control block pointed to by VALP (or the current control block). The following code is moved one word to make room for it.

MATCH (SUBROUTINE)

The mode table is searched for an entry that specifies a mode equal to the mode specified by the argument. This subroutine is used to insure that only one entry of the mode corresponds to any given mode.

DELWW (SUBROUTINE)

Several control blocks starting with the one pointed to by XR-1 are combined into a single control block. This single block has a starting location specified in the old top control block and a length necessary to include everything included in any of the old control blocks.

DELW (SUBROUTINE)

Several control blocks starting with the one pointed to by XR-1 are combined into a single control block. This single block is the old top block.

WAD (SUBROUTINE)

A control block is created and made to refer to a word added with CAD which is immediately called.

CAD (SUBROUTINE)

The argument word is added to the end of the code that is being generated.

DEREF (SYNTAX)

The mode saved in FMODE is dereferenced and stored in DECLR.

EVOID (SYNTAX)

A void mode is stored in DECLR.

EREF (SYNTAX)

The mode in DECLR is changed to a mode that is a reference to that mode.

EREF1 (SUBROUTINE)

A mode that is a reference to the argument mode is returned.

CST (SYNTAX)

The top word in the control stack is where the working stack was marked. The working stack contains pairs of words after the mark consisting of:

[mode, bound]

[tag, 0]

An entry is created in the bound table for the bounds and a mode is constructed from the modes and tags. The subroutine MATCH is called to find the unique M$STRCT entry in the mode table for the structured mode. All words added to the working stack after the mark are deleted and a word referring to the structured mode and bound is pushed onto the working stack. The mark in the control stack is popped.

EPR (SYNTAX)

The top word in the control stack is where the working stack is marked. Words added to the working stack since the mark contain modes of the arguments of the procedure and the mode of the result. All words added after the stack was marked are deleted and a procedure mode with the proper modes as arguments and result found by MATCH is pushed onto the working stack. The mark word is deleted from the control stack.

CUN (SYNTAX)

The top word of the control stack is where the working stack was marked. Words added to the working stack since the mark contain modes of a united mode. All words added after the stack was marked are deleted and a united mode united from these modes found by match is pushed onto the working stack. The mark word is deleted from the control stack.

DSUB (SYNTAX)

In the PROG table entry for the row declarer just encountered there is a label for the bounds procedure and a label for jumping around the bounds procedure. The following code is added to the generated output:

```
[O$JUMP, end of procedure label]
[O$EPDN, procedure label]
[O$LL, current level]
[O$DSUB, 0]
```

This is the header of the bounds procedure.

DBUS (SYNTAX)

The following code is added to the generated output:

```
[O$DBUS, row PROG table entry]
[O$RETN, void]
[O$LLE, current range]
[O$DLEN, length label]
```

[O$LBL, end of procedure label]

This is the trailer of the bounds procedure.

CBOX (SYNTAX)

Two words are allocated in the BOUND table. These two words are set to the following contents:

[B$ROW, dimension]

[0, current range]

Then a pointer to this table entry is pushed into the working stack.

EBOX (SYNTAX)

The top word in the working stack points to an entry in the BOUND table. The bound part of DECLR is stored in the upper half of the second word of this table entry. The mode part of DECLR is changed to a row mode with the number of dimensions indicated in the BOUND table entry and whose element mode is the original mode in DECLR. The top word in the working stack containing the pointer to the BOUND table entry is deleted.

EPDEN (SYNTAX)

The top word in the control stack is the mode of the procedure denotation. This mode is unstacked from the control stack and the routine STRNG is called with a pointer to the top control block in the working stack and a target mode of the result of the procedure mode of the procedure denotation. The new top word in the control stack is the starting code location of the procedure denotation. It is unstacked and the LOC/LEN word of the top control block is changed to include starting location of the procedure denotation. The mode of the procedure denotation is stored in the last control block. The routine INS is called with the argument [O$SRNGE, current range]. The routine ADD is called with the argument [O$ERNGE, current range]. Then [-1, +1] is added to LOC/LEN word of the top control block.

EPDNE (SYNTAX)

The top control block LOC/LEN word is made to include the last generated code word. Then the routine PCDR is called.

VOID (SYNTAX)

The routine STRNG is called with the top control block in the working stack as an argument and VOID as a target mode.

ODELV (SYNTAX)

The routine CAD is called with the argument [O$DELV, void].

GMOD (SYNTAX)

The mode of the last identifier accepted from the input stream is stored in FMODE.

SVAL (SYNTAX)

The top control block in the working stack is made the current control block.

PUSH (SYNTAX)

The word indicated by the argument is fetched and pushed into the control stack.

POP (SYNTAX)

The top word of the control stack is popped and stored in the location indicated by the argument.

PUSHW (SYNTAX)

The word indicated by the argument is fetchcd and pushed into the working stack.

PUSEA (SYNTAX)

The word indicated by the argument is fetched and pushed into the working stack with the lower half zero.

SAVE (SYNTAX)

The contents of DECLR are fetched and stored in the word indicated by the argument.

REST (SYNTAX)

The word indicated by the argument is fetched and stored in DECLR.

ADO (SYNTAX)

One is added to the location specified by the argument.

STZ (SYNTAX)

A zero is stored in the upper half of the word specified by the argument.

CLEAR (SYNTAX)

A zero is stored in the word specified by the argument.

SHEAP (SYNTAX)

O$HGEN is stored in the location GTYPE.

SLOC (SYNTAX)

O$LGEN is stored in the location GTYPE.

SBLNK (SYNTAX)

Zero is stored in the location GTYPE.

SHPBK (SYNTAX)

If location GTYPE is nonzero a normal return is given. Otherwise, O$HGEN is stored in the location GTYPE.

SLCBK (SYNTAX)

If location GTYPE is nonzero a normal return is given. Otherwise, O$LGEN is stored in the location GTYPE.

SBCT (SYNTAX)

The word

[O$VSBCT, CNT+1]

is added to the end of the generated output where CNT+1 is the current subscript position.

LWB (SYNTAX)

The word

[O$VLWB, CNT+1]

is added to the end of the generated output where CNT+1 is the current subscript position.

UPB (SYNTAX)

The word

[O$VUPB, CNT+1]

is added to the end of the generated output where CNT+1 is the current subscript position.

NLWB (SYNTAX)

The word

[O$VNLWB, CNT+1]

is added to the end of the generated output where CNT+1 is the current subscript position.

EPTY (SYNTAX)

The word

[O$VEPTY, CNT+1]

is added to the end of the generated output where CNT+1 is the current subscript position.

OLWB (SYNTAX)

The word

[O$LWB, CNT+1]

is added to the end of the generated output where CNT+1 is the current subscript position.

OUPB (SYNTAX)

The word

[O$UPB, CNT+1]

is added to the end of the generated output where CNT+1 is the current subscript position.

OFIX (SYNTAX)

The word

[O$FIX, CNT+1]

is added to the end of the generated output where CNT+1 is the current subscript position.

OFLEX (SYNTAX)

The word

[O$FLEX, CNT+1]

is added to the end of the generated output where CNT+1 is the current subscript position.

OSUB (SYNTAX)

The word

[O$SUB, 0]

is added to the end of the generated output.

OBUS (SYNTAX)

A number of O$VEPTY commands are added to the generated output code until every position of the row value is subscripted. The new mode of the subscripted base is calculated and stored in the top value control block in the working stack. Then either the word

[O$BUS, mode]

or the word

[O$RBUS, mode]

is added to the generated output code if the mode of the base is not a reference mode or is a reference mode respectively. The length in the top value control block in the working stack is updated to include this added word.

SINT (SYNTAX)

The top value control block is strongly coerced to the mode integral.

SACT (SYNTAX)

A one is stored in location DFLG to indicate an actual declarer.

SVIRT (SYNTAX)

A minus one is stored in location DFLG to indicate a virtual declarer.

STID (SYNTAX)

A pointer to the last symbol accepted from the input stream is stored in IDNT.

PARN (SYNTAX)

A new control block of type W$PAR is created on the working stack. The count is set equal to one plus CNT and zero is stored for the range information. Also, the top word from the control stack is popped and stored as a pointer to the start of the code for this value.

BAL2 (SYNTAX)

A new value control block of type W$BAL is created on the working stack. The count is set equal to two and zero is stored for the range information. Also, the top word from the control stack is popped and stored as a pointer to the start of the code for this value.

BALZR (SYNTAX)

A new value control block of type W$BAL is created on the working stack. The count is set equal to one plus CNT and zero is stored for the range information. Also, the top word from the control stack is popped and stored as a pointer to the start of the code for this value.

BALN (SYNTAX)

A new value control block of type W$BAL is created on the working stack. The count is set equal to one plus CNT and the current range number is stored for the range information. Also, the top word from the control stack is popped and stored as a pointer to the start of the code for this value.

INLL (SYNTAX)

A pointer to the top of the working stack is pushed into the control stack. Then RLL (the current range LL word) is pushed into the control stack. RLL is set to zero.

ORLL (SYNTAX)

The contents of RLL (the current range LL word) is ored into the LL word of the top value control block in the working stack. Location RLL is shifted left one place and the contents of the word popped from the control stack is ored into the shifted contents of location RLL. The next word from the control stack is popped to get the length of the working stack when this range was entered. The contents of all LL words in value control blocks that were created after this range was entered are shifted left one place to change their representation to that of the outer range.

WGEN (SYNTAX)

A new value control block of type W$VALUE is created on the working stack. The mode of the value is set to the contents of DECLR. The value control block is made to refer to a word of output code containing [contents of GTYPE, contents of DECLR] and this word is added to the output code.

ENTL (SYNTAX)

A code word containing [O$LBL, label referred to by the argument] is inserted in front of the code for the value referred to by the value control block on top of the working stack. Then the label referred to by the argument is incremented.

ENTC (SYNTAX)

The following is inserted in front of the code referred to by the top value control block in the working stack:

[O$CONF, mode]

[O$TF, B1]

[O$MREF, 0]

and the following is added after the code referred to by the value control block:

[O$CASGN, mode]

[O$JUMP, (B2)]

B1 and B2 are the labels stored in locations B1 and B2. The value of the label in B2 is incremented after this sequence is generated. Then the top value control block is deleted from the working stack.

CSCT (SYNTAX)

RELF is set to zero to indicate "conforms to".

CSCTB (SYNTAX)

RELF is set to one to indicate "conforms to and becomes".

CONF (SYNTAX)

A unique label is obtained and stored in LL. The B1 and B2 label generators are initialized to the start of two label blocks containing two plus the number of commas in the current range number of labels. The following code words are added to the output:

```
          [O$JUMP, L1]
          [O$MREF, 0]
```

CONE (SYNTAX)

The following is inserted in front of the code for the value referred to by the top value control block in the working stack:

```
          [O$JUMP, (B2)]
          [O$LBL, L1]
          [O$MAX, 0]
```

and the following is added after the code referred to by the value control block:

```
          [O$CONE, 0]
          [O$JUMP, B1S]
```

(B1S is the initial value of the label B1 before incrementing) The label generator B2 is reset back to the beginning and the last value control block in the working stack is deleted.

DIF (SYNTAX)

If the range following the current range has semicolons or no commas then the value on top of the working stack is coerced to a boolean mode. The label generator B2 is set to a unique label. [O$TF, (B2)] is added to the output code where (B2) is the label in B2. The top value control block in the working stack is deleted.

Otherwise, the top value on top of the working stack is coerced to integral mode. Label generator B2 is set to the start of a block of N+2 labels where N is the number of commas in the following range. [O$CASE, N+1] is added to the output code where N is the number of commas in the following range. [O$JUMP, last] is added to the output code where last is the last label in the B2 block. Then [O$JUMP, B2+] is successively added to the output code N+1 times with labels from the first N+1 labels in the block B2. The B2 label generator is reset to the beginning of the label block. Then the top value control block in the working stack is deleted.

DELV (SYNTAX)

The last value control block in the working stack is deleted.

OIDNY (SYNTAX)

Location OPNT points to the last operator declared. This pointer is stepped to the next operator definition in the DEF table and a copy is stored in LASTS as the last operator read.

FOP (SYNTAX)

The last symbol accepted from the input stream is looked up in the symbol table for the first priority definition. This priority is stored as the current priority and the routine COP is called. A new value control block of type W$OP is created on top of the working stack. The operator symbol and its priority and a pointer to the last W$OP value control block found in OPT are stored in it. A pointer to the current W$OP value control block is stored in OPT.

FOPS (SYNTAX)

The current priority is set to zero and the routine COP is called.

WMOP (SYNTAX)

The pointer to the last symbol accepted from the input stream (which is an operator) is stored on top of the control stack.

MOP (SYNTAX)

The pointer to the operator is popped from the control stack, the unary operator flag is set, and the routine COPM is called.

SELCT (SYNTAX)

The top value control block in the working stack is weakly coerced. The mode of the top value control block must be either a structured mode or a reference to a structured mode or it is an error. The tag is popped from the control stack and the structured mode is searched for a field with this tag. If not found, a terminal error message is typed out. If the original mode was a structured mode then the code [O$SELCT, num] is added after the current value. If the mode was a reference to a structured mode then the code [O$RSLCT,num] is added after the current value. (num is the number of the field with the matching tag.) Then the code [O$ETC, mode] is added where mode is the mode of the field if the original mode was a structured mode and reference to the mode of the field if the original mode was a reference to a structured mode. The top value control block in the working stack is modified to include the two added code words and its mode is changed to the mode in the O$ETC code word.

CALL (SYNTAX)

The top word from the control stack is popped and stored in location DECLR as the mode of the procedure.The actual parameter value control blocks on top of the working stack are strongly coerced to the mode of the formal parameters of the procedure. The code [O$MSCW, 0] is inserted in front of the code for the top value control block (arguments). The code [O$ENTER, mode] is added after the code for the top value control block where mode is the mode of the result of the procedure. The top two value control blocks are combined into a single value and made to include all of the code added here, and its mode is set equal to the mode of the result of the procedure.

MARK (SYNTAX)

A pointer to the top of the working stack is pushed in the control stack. Then a zero is pushed onto the working stack (to save room for a header word).

ESKIP (SYNTAX)

A new value control block of type W$SKIP is created on the working stack. The code [O$SKIP, 0] is added to the generated output code and the value control block is made to refer to this code word. The LL word is set to zero.

ENIL (SYNTAX)

A new value control block of type W$NIL is created on the working stack. The code [O$NIL, 0] is added to the generated output code and the value control block is made to refer to this code word. The LL word is set to zero.

EFALS (SYNTAX)

A new value control block of type W$VALUE is created on the working stack. The code [O$FALSE, 0] is added to the generated output code and the value control block is made to refer to this code word. The LL word is set to zero.

ETRUE (SYNTAX)

A new value control block of type W$VALUE is created on the working stack. The code [O$TRUE, 0] is added to the generated output code and the value control block is made to refer to this code word. The LL word is set to zero.

SDCLR (SYNTAX)

The mode of the last symbol accepted from the input stream which is a mode indication is stored in location DECLR.

OIDN (SYNTAX)

The code [O$IDNTY, def] is added to the output code where def is a pointer to the definition of the last symbol accepted from the input stream which is an identifier.

OENTR (SYNTAX)

The code [O$IDNTE, DECLR] is added to the output code where DECLR is the mode in DECLR.

OASGN (SYNTAX)

The code [O$ASGN, DECLR] is added to the output code where DECLR is the mode in DECLR.

OFORM (SYNTAX)

The code [O$FORMP, formal] is added to the output code where formal is a pointer to the definition of the last symbol accepted from the input stream which is an identifier.

WIDEN (SYNTAX)

A new value control block of type W$VALUE is created on the working stack. It is made to refer to a newly added code word [O$IDENT, def] where def is a pointer to the definition of the identifier. The declarer for the identifier is stored in the value control block. The LL of the last identifier is set in the value control block and ored into RLL.

WDEN (SYNTAX)

A new value control block of type W$VALUE is created on the working stack. It is made to refer to a newly added code word [O$DENOT, def] where def is a pointer to the definition of the denotation. A declarer for the denotation is stored in the value control block. Zero is stored in the LL location of the value control block.

PCDR (SUBROUTINE)

The following is inserted in front of current value in VALP:

[O$JUMP, *1] (*1 is a unique label)

[O$EPDN, *2] (*2 is another unique label)

[O$LL, range] (range is scope of proc)

and added after the current value:

[O$RETN, mode of result of procedure]

[O$LLE, 0]

[O$LBL, *1]

[O$EPDV, mode of procedure]

[O$EPDE, *2]

Current value control block is made to include all of these added code words.

BIBLIOGRAPHY

- "Compact garbage", Science and Technology, 90, 39-40 (1969).

Baecker, H.D., "The use of ALGOL 68 for trees", The Computer Journal, 13, 25-27 (1970).

Berry, D.M., Introduction to Oregano, Technical Report 70-29, Brown University, 1970.

Berry, D.M., Some Aspects of the Structure of ALGOL 68, Division of Applied Mathematics, Brown University, Providence, R.I., 1970.

Berry, D.M., The Importance of Implementation Models in ALGOL 68, Report No. 70-C-287, General Electric Research and Development Center, Schenectady, N.Y., 1970.

Bowlden, H.J., A Symbol Table with Scope Recognition for the B-6500, Scientific Paper 70-1K4-COMPS-P1. Westinghouse Research Laboratories, Pittsburgh, Pa., 1970.

Bowlden, H.J., ALGOL 68 Structural Flowchart, Research Report 69-1C4- COMPS-R2, Westinghouse Research Lab. Pittsburgh, Pa., 1969.

Bowlden, H.J., Environmental Factors in Computer Design and Implementation, Paper presented at The ALGOL 68 Implementation Seminar, Vancouver, 1969.

Branquart, P., & Lewi, J., A Scheme of Storage Allocation and Garbage Collection for ALGOL 68, Report R 133, MBLE Reseach Lab. Brussels, July 1970.

Branquart, P., & Lewi, J., Analysis of the Paranthesis Structure of ALGOL 68, Report R130, MBLE Research Lab. Brussels, April 1970.

Branquart, P., & Lewi, J., General Principles of an ALGOL 68 Garbage Collector, Technical Note N60, MBLE Research Lab. Brussels, January 1970.

Branquart, P., & Lewi, J., On Object Language and Storage Allocation in ALGOL 68 Compilers, Report R117, MBLE Research Lab. Brussels, September 1969.

Branquart, P., & Lewi, J., On the Implementation of Local Names in ALGOL 68, Report R121, MBLE Research Lab. Brussels, November 1969.

Branquart, P., & Lewi, J., On the Implementation of Coercions in ALGOL 68, Report R123, MBLE Research Lab.

Brussels, January 1970.

Branquart, P., & Lewi, J., On the Implementation of Local Names in ALGOL 68 (Revised version), Report R121, MBLE Research Lab. Brussels, September 1970.

Branquart, P., Cardinael, J.P., Descaille, J.P., & Van Begin, M., Output of the Syntactic Analyzer of the ALGOL 68-X8.1 Compiler (PART II), Technical Note N73, MBLE Research Lab. Brussels, December 1971.

Branquart, P., Lewi, J., & Cardinael, J.P., A Context-Free Syntax of ALGOL 68 (Revised Version), Technical Note N66, MBLE Research Lab. Brussels, August 1970.

Branquart, P., Lewi, J., & Cardinael, J.P., Local Generators and the ALGOL 68 Working Stack, Technical Note N62, MBLE Reseach Lab. Brussels, September 1970.

Branquart, P., Lewi, J., Sintzoff, M., & Wodon, P.L., Structural Composition of Semantics in ALGOL 68, Report R125, MBLE Research Lab. Brussels, April 1970.

Branquart, P., Lewi,J., & Cardinael, J.P., Decision Table for the Analysis of the Parenthesis Structure of ALGOL 68 (updated), Technical Note N68, MBLE Research Lab.

Brussels, October 1970.

Carr, C.S., Luther, D.A. & Erdmann, S., _The Tree - Meta Compiler - Compiler System: A Meta Compiler System for the UNIVAC 1108 and the General Electric 645_, Rome Air Development Center, Air Force Systems Command, Griffiss Air Force Base, New York, Technical Report 69-83, 1969.

Cheney, C.J., "A nonrecursive list compacting algorithm", _Communications of the ACM_, 13, 677-678 (1970).

Currie, I.F., _Working Description of ALGOL 68-R_, RRE Memorandum No. 2660, Ministry of Aviation Aupply, RRE Malvern Worcs, England, 1970.

Dahl, O.J., Myhrhaug, B., and Nygaard, K., _Common Base Language_, Publication No. S-22, Norwegian Computing Center, Oslo, Norway, 1970.

Duncan, F.G. (Ed.), _ALGOL Bulletin No. 30_, February 1969.

Duncan, F.G. (Ed.), _ALGOL Bulletin No. 31_, March 1971.

Duncan, F.G. (Ed.), _ALGOL Bulletin No. 32_, May 1971.

Fenichel, R.R. & Yochelson, J.C., "A LISP garbage-collector for virtual-memory computer systems", Communications of the ACM, 12, 611-612 (1969).

Finch, P.M., Defining and Applied Occurences of Identifiers, Paper presented at The ALGOL 68 Implementation Seminar, Vancouver, 1969.

Freiburghouse, R.A., The Multics PL/1 Compiler, Paper presented at the Fall Joint Computer Conference, 1969.

Goos, G., Eine Implementierung von ALGOL 68, Technische Hochschule, Munich, 1969.

Goos, G., Some Problems in Compiling ALGOL 68, Paper presented at ACM SIGPLAN ALGOL 68 Symposium, June 1970.

Goyer, P., A Garbage Collector to be Implemented on a CDC 3100, Publication No. 34. Departement d'Informatique, Universite de Montreal, Montreal, 1970.

Haddon, B.K. & Waite, W.M., "A Compaction Procedure for Variable-length Storage Elements", Computer Journal 10, 2, 162-165 (August 1967).

Hansen, W.J., "Compact list representation: Definition, garbage collection, and system implementation", Communications of the ACM, 12, 499-507 (1969).

Haynes, H.R. & Schutte, L.J., Compilation of Optimized Syntactic Recognizers from Floyd-Evans Productions, 1970.

Hodgson, G.S., ALGOL 68 Extended Syntax, University of Manchester, Department of Computer Science, 1970.

Koster, C.H.A., A Compiler Compiler, Mathematisch Centrum, Amsterdam, 1971.

Lauer, P., Formal Definition of ALGOL 60, Technical Report 25.088, IBM Laboratory Vienna, Austria, 1968.

Lindsey, C.H. & Van der Muelen, S.G., Informal Introduction to ALGOL 68, North-Holland Publishing Company, Amsterdam, London, 1971.

Lindsey, C.H., An ISO-code Representation for ALGOL 68, Paper presented to the Seminar on ALGOL 68 Implementation, University of British Columbia, 1969.

Lucas, P., Lauer, P., & Stigleitner, H., Method and Notation for the Formal Definition of Programming Languages, Technical Report 25.087, IBM Laboratory Vienna, Austria, 1968.

Mailloux, B.J., _On the Implementation of ALGOL 68_, Ph.D. thesis, Mathematisch Centrum, Amsterdam, 1968.

Marshall, S., _An ALGOL 68 Garbage Collector_, Dartmouth College, Kiewit Computation Center, Technical Memorandum TM011, 1969.

Naur, P.(Ed.), "Revised report on the algorithmic language ALGOL 60", _Communications of the ACM_, 6, 1-23 (1963).

Peck, J.E.L. (Ed.), _ALGOL 68 Implementation_, North-Holland Publishing Company, Amsterdam, London, 1971.

Peck, J.E.L. (Ed.), _Proceedings of an Informal Conference on ALGOL 68 Implementation_, University of British Columbia, 1969.

Peck, J.E.L., _An ALGOL 68 Companion_, University of British Columbia, Preliminary Edition, 1971.

Peck, J.E.L., _On Storage of Modes and Some Context Conditions_, Paper presented at The ALGOL 68 Implementation Seminar, Vancouver, 1969.

Pierce, R.H., _An ALGOL 68 Run-Time Organisation_, M.S. thesis, Victoria University of Manchester, 1971.

Reynolds, J.C., "GEDANKEN - A simple typeless language based on the principle of completeness and the reference concept", *Communications of the ACM*, 13, 308-319 (1970).

Schneider, V.B., *A One-Pass Algorithm for Compiling ALGOL 68 Declarations*, Purdue Research Foundation, Purdue University, 1970.

Schneider, V.B., *A Translation Grammar for ALGOL 68*, Paper presented at The Spring Joint Computer Conference, Bonn, 1970.

Schorr, H. & Waite, W.M., "An efficient machine-independent procedure for garbage collection in various list structures", *Communications of the ACM*, 10, 501-506 (1967).

Sintzoff, M. (Ed.), Branquart, P., Lewi, J., & Wodon, P.L., *Remarks on the Draft Reports on ALGOL 68*, Report R96, MBLE Research Lab. Brussels, January 1969.

Wegner, P., *Data Structure Models for Programming Languages*, Technical Report 70-30, Center for Computer and Information Sciences, Brown University, Providence, R.I., 1970.

Westland, J., *An ALGOL 68 Syntax and Parser*, M.S. thesis, University of Calgary, Calgary, Alberta, 1969.

Wijngaarden, A. van (Ed.), Mailloux, B.J., Peck, J.E.L., Koster, C.H.A., "Report on the Algorithmic Language ALGOL 68", Numerische Mathematik, 14, 79-218 (1969).

Wirth, N., "The design of a PASCAL compiler", Software - Practice and Experience, 1, 309-333 (1971).

Wirth, N., "The programming language Pascal", Acta Informatica, 1, 35-63 (1971).

Woodward, P.M., "Practical experience with ALGOL 68", Software - Practice and Experience, 2, 7-19 (1972).

Woodward, P.M., & Bond, S.G., Users' Guide to ALGOL 68-R, Ministry of Defence, RRE Malvern Worcs, England, 1971.

Woodward, P.M., Primer of ALGOL 68-R, RRE Memorandum No.2660, Ministry of Technology, RRE Malvern Worcs, England, 1970.

VOLUME 2

LISTING OF THE COMPILER

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

@   @  @@@@    @@@    @@@   @@@@    @@@   @@@@    @@@            @@@@@  @       @@@   @@@@@  @         @      @@@
@   @  @   @  @   @  @   @      @  @   @      @  @   @           @   @  @      @   @  @   @  @        @      @   @
@   @  @   @  @         @@    @@@     @@    @@@     @@           @@@@@  @      @      @   @  @        @       @@@
@@@@@  @   @  @        @@         @  @@         @   @@           @   @  @      @ @@@  @   @  @        @@@@   @   @
@   @  @   @  @   @  @@           @  @@         @  @@            @   @  @      @   @  @   @  @        @   @  @   @
@   @  @@@@    @@@   @@@@@  @@@@   @@@@@  @@@@   @@@@@           @   @  @@@@@   @@@   @@@@@  @@@@@    @@@    @@@

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

      05/26/72          00:40:10
```

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

@   @  @@@@     @@@   @@@   @@@@    @@@   @@@@    @@@                 @@@@@   @       @@@   @@@@@  @       @      @@@
@   @  @   @   @   @ @   @      @  @   @      @  @   @                @   @   @      @   @  @   @  @      @      @   @
@   @  @   @   @        @@   @@@      @@   @@@      @@                @@@@@   @      @      @   @  @      @       @@@
@@@@@  @   @   @      @@        @   @@        @   @@                  @   @   @      @ @@@  @   @  @      @@@@   @   @
@   @  @   @   @   @ @@         @  @@         @  @@                   @   @   @      @   @  @   @  @      @   @  @   @
@   @  @@@@     @@@  @@@@@  @@@@   @@@@@  @@@@   @@@@@                @   @   @@@@@   @@@   @@@@@  @@@@@   @@@    @@@

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

      05/26/72          00:42:26
```

```
$       UPDATE  LIST
$       ALTER   1646,1646
INIT    ZERO                        PAUSE FOR PATCHES
$       ALTER   2506
        TZE     PL3                 TRANSFER IF GLOBAL
$       ALTER   4780
        TZE     PCDR2               TRANSFER IF GLOBAL
$       ALTER   5762,5762
DONE    NOP
$       ALTER   11194,11197
        DEF     M,M94,(18/PROC,18/BOOL,18/BOOL)
        DEF     M,M95,(18/PROC,18/CHAR,18/BOOL)
        DEF     M,M96,(18/PROC,18/INT,18/BOOL)
        DEF     M,M97,(18/PROC,18/REAL,18/BOOL)
```

```
01557 01  05-26-72   17,863      CONSTANTS                                                                    PAGE      1

                                       FAULT VECTOR

                                        1          TTL      CONSTANTS
                                        2          TTLS     FAULT VECTOR
                                        3          ABS
                                        4          INHIB    ON
                                        5          HEAD     M
                          500005        6 PAUSE    BOOL     500005
                          500101        7 OPEN     BOOL     500101
                          500105        8 CLOSE    BOOL     500105
                          500133        9 READ     BOOL     500133
                          500134       10 WRITE    BOOL     500134
                          500113       11 SETP     BOOL     500113
                          500000       12 TERM     BOOL     500000
                          500006       13 MREQ     BOOL     500006
                          500100       14 OPENS    BOOL     500100
                          500103       15 CAT      BOOL     500103
                          500107       16 TRUNC    BOOL     500107
                          500012       17 JTIME    BOOL     500012
                                       18          HEAD
                          000000       19 RELZER   EQU      *
      000000  002407 7102 00           20          TRA      $START          START UP SYSTEM
      000001  000246 0012 02           21          TALLYD   SIB,10,2        SPECIAL INTERRUPT TALLY WORD
                          000002       22          DUP      2,15
      000002  000000 000000            23          ZERO
      000003  002164 7172 00           24          XED      ,$TRAP
      000004  000000 000000                        ZERO
      000005  002164 7172 00                       XED      ,$TRAP
      000006  000000 000000                        ZERO
      000007  002164 7172 00                       XED      ,$TRAP
      000010  000000 000000                        ZERO
      000011  002164 7172 00                       XED      ,$TRAP
      000012  000000 000000                        ZERO
      000013  002164 7172 00                       XED      ,$TRAP
      000014  000000 000000                        ZERO
      000015  002164 7172 00                       XED      ,$TRAP
      000016  000000 000000                        ZERO
      000017  002164 7172 00                       XED      ,$TRAP
      000020  000000 000000                        ZERO
      000021  002164 7172 00                       XED      ,$TRAP
      000022  000000 000000                        ZERO
      000023  002164 7172 00                       XED      ,$TRAP
      000024  000000 000000                        ZERO
      000025  002164 7172 00                       XED      ,$TRAP
   END OF BINARY CARD 00000001
      000026  000000 000000                        ZERO
      000027  002164 7172 00                       XED      ,$TRAP
      000030  000000 000000                        ZERO
      000031  002164 7172 00                       XED      ,$TRAP
      000032  000000 000000                        ZERO
      000033  002164 7172 00                       XED      ,$TRAP
      000034  000000 000000                        ZERO
```

```
                                        FAULT VECTOR

     000035  002164 7172 00                         XED      ,$TRAP
     000036  000000 000000                          ZERO
     000037  002164 7172 00                         XED      ,$TRAP
```

TASK CONTROL BLOCKS

```
                                     TTLS    TASK CONTROL BLOCKS
                                  *
                                  * THE FORMAT OF A TASK CONTROL BLOCK IS
                                  *
                                  * -6   LINK      0
                                  * -5   PRIORITY  1
                                  * -4   STAT1     2
                                  * -3   STAT2     3
                                  * -2   IC        4
                                  * -1   XED TRAP  5
                                  *  0   SAVED IC  6
                                  *  1   IC        7
                                  *  2   X0,X1     8
                                  *  3   X2,X3     9
                                  *  4   X4,X5    10
                                  *  5   X6,X7    11
                                  *  6   A-REG    12
                                  *  7   Q-REG    13
                                  *  8   E-REG    14
                                  *  9   TIMER    15
                                  *
                                  TCB    MACRO    HEAD,NAME,PRIORITY,ADDRESS
                                         HEAD     #1
                                         EIGHT
                                  #2     ZERO                       LINK
                                         ZERO     #3                PRIORITY
                                         ZERO                       STATUS WORD 1
                                         ZERO                       STATUS WORD 2
                                         ZERO                       RETURN IC
                                         XED      ,$TRAP            TRAP ROUTINE
                                         ZERO                       SAVED IC
                                         ZERO     #4                IC
                                         OCT      0,0,0,0,0,0,0,0   SAVED REGISTERS
                                         ENDM     TCB
                    000040               TCB      ,NULL,0,-1        NULL BLOCK
END OF BINARY CARD 00000002
                    000060               TCB      ,MTCB,1,MTASK     MASTER FAULT AND INTERRUPT ROUTINE
                    000100               TCB      ,STCB,2,SPEC      SPECIAL INTERRUPT ROUTINE
END OF BINARY CARD 00000003
                    000120               TCB      Z,TCB,10,FIN      DEBUGGER
END OF BINARY CARD 00000004
                    000140               TCB      A,TCB,20,INIT
END OF BINARY CARD 00000005
                    000160               TCB      B,TCB,15,$BUF1
                    000200               TCB      ,ITCB,99,IDLE     IDLE PROGRAM
END OF BINARY CARD 00000006
```

EXECUTIVE

```
                                     66        TTLS     EXECUTIVE
                                     67        HEAD     ,
                      777777         68 ERROR  EQU      -1
    000220   000000 000000           69 REG    ZERO
    000221   000000 000000           70 IC     ZERO
    000222   000000 000000           71 STAT   ZERO
    000223   000000 000000           72 LINK   ZERO
    000224   000000 000000           73 TASK   ZERO
                      000047         74 NIC    EQU      $NULL+7
                      000050         75 NREG   EQU      $NULL+8
    000225   002164 7172 00          76 XEDT   XED      TRAP
    000226   000000 000000           77 FFLAG  ZERO
    000227   000502060702            78        DATE
```

```
                                         EXECUTIVE TASKS

                                         79        TTLS    EXECUTIVE TASKS
                                         80        HEAD
                           777777        81 ERROR  EQU     -1
       000230  000000 000000             82 TTY    ZERO
       000231  000000 000000             83 TTYF   ZERO
  END OF BINARY CARD 00000007
       000232  000000 777777             84 STTYF  ZERO    ,-1
       000233  000000011207
                           000234        85        EVEN
                           000234        86 TTYB   BSS     8
       000244  000246 0012 02            87 SIF    TALLYD  SIB,10,2
       000245  000000011207
                           000246        88        EVEN
                           000246        89 SIB    BSS     10*2
       000272  000001 000001             90 TTYSI  ZERO    1,1
       000273  000005710204
                           000300        91        EIGHT
       000300  000000 000310             92 LREG   ZERO    0,WLOC
       000301  000001 000000             93        ZERO    1,0
       000302  000000 000000             94        ZERO
       000303  000000 000311             95        ZERO    ,WN
       000304  000000000000              96        OCT     0,0,0,0
       000305  000000000000
       000306  000000000000
       000307  000000000000
       000310  000000 000234             97 WLOC   ZERO    ,TTYB
       000311  000000 000006             98 WN     ZERO    ,6
       000312  000006710204
                           000320        99        EIGHT
                           000320       100 RREG   EQU     *
       000320  000001 000410            101 WREG   ZERO    1,LOC
  END OF BINARY CARD 00000008
       000321  000001 000410            102        ZERO    1,LOC
       000322  000000 000000            103        ZERO
       000323  000000 000411            104        ZERO    ,LEN
       000324  000000000000             105        OCT     0,0,0,0
       000325  000000000000
       000326  000000000000
       000327  000000000000
                           000330       106        EIGHT
       000330  000000 000410            107 FREG   ZERO    ,LOC
       000331  000000 000410            108        ZERO    ,LOC
       000332  000000 000000            109        ZERO
       000333  000000 000411            110        ZERO    ,LEN
       000334  000000000000             111        OCT     0,0,0,0
       000335  000000000000
       000336  000000000000
       000337  000000000000
                           000340       112        EIGHT
       000340  000000 000344            113 OREG   ZERO    0,OREG+4
```

EXECUTIVE TASKS

```
      000341  000000 000000          114          ZERO
      000342  000000 000000          115          ZERO
      000343  000000 000000          116          ZERO
      000344  000000000000           117          OCT      0,0,0,0
      000345  000000000000
      000346  000000000000
  END OF BINARY CARD 00000009
      000347  000000000000
                          000350     118          EIGHT
      000350  000000 000000          119 OSREG    ZERO
      000351  000000 000000          120          ZERO
      000352  000000 000005          121          ZERO     0,5
      000353  000000 000000          122          ZERO
      000354  000000000000           123          OCT      0,0,0,0
      000355  000000000000
      000356  000000000000
      000357  000000000000
                          000360     124          EIGHT
      000360  000000 000364          125 CREG     ZERO     0,CREG+4
      000361  000000 000364          126          ZERO     0,CREG+4
      000362  000367 000000          127          ZERO     CREG+7
      000363  000000 000000          128          ZERO
      000364  000000000000           129          OCT      0,0,0
      000365  000000000000
      000366  000000000000
      000367  007660007660           130          OCT      007660007660
                          000370     131          EIGHT
      000370  000000000000           132 TRREG    OCT      0,0,0,0,0,0,0,0
      000371  000000000000
      000372  000000000000
      000373  000000000000
      000374  000000000000
  END OF BINARY CARD 00000010
      000375  000000000000
      000376  000000000000
      000377  000000000000
                          000400     133          EIGHT
      000400  000000 000000          134 SREG     ZERO
      000401  000000 000000          135          ZERO
      000402  000000 000000          136          ZERO
      000403  000000 000000          137          ZERO
      000404  000000 000000          138          ZERO
      000405  000000 000000          139          ZERO
      000406  000000 000000          140          ZERO
      000407  000000 000000          141          ZERO
      000410  000000 000000          142 LOC      ZERO
      000411  000000 000000          143 LEN      ZERO
                          000412     144          EVEN
      000412  015012123104           145 FTAB     OCT      015012123104,117127116040
      000413  117127116040
```

```
                                    EXECUTIVE TASKS
     000414  015012115105         146          OCT        015012115105,115040040040
     000415  115040040040
     000416  015012115115         147          OCT        015012115115,105040040040
     000417  105040040040
     000420  015012106124         148          OCT        015012106124,101107040040
     000421  101107040040
     000422  015012124111         149          OCT        015012124111,115105122040
 END OF BINARY CARD 00000011
     000423  115105122040
     000424  015012103115         150          OCT        015012103115,104040040040
     000425  104040040040
     000426  015012104122         151          OCT        015012104122,114040040040
     000427  114040040040
     000430  015012114103         152          OCT        015012114103,113125120040
     000431  113125120040
     000432  015012123120         153          OCT        015012123120,105103040040
     000433  105103040040
     000434  015012120101         154          OCT        015012120101,122040040040
     000435  122040040040
     000436  015012132105         155          OCT        015012132105,122117120040
     000437  122117120040
     000440  015012117116         156          OCT        015012117116,103040040040
     000441  103040040040
     000442  015012123124         157          OCT        015012123124,101122124040
     000443  101122124040
     000444  015012117126         158          OCT        015012117126,106114040040
     000445  106114040040
     000446  015012104111         159          OCT        015012104111,126040040040
     000447  126040040040
     000450  015012105130         160          OCT        015012105130,106040040040
 END OF BINARY CARD 00000012
     000451  106040040040
```

```
                                          DEBUGGER

                                        161          TTLS     DEBUGGER
                                        162          HEAD     Z
    000452  000000 000000               163 TEMP     ZERO
    000453  000000 000000               164 TEMP1    ZERO
    000454  000000 000000               165 WTEMP    ZERO
    000455  000140 000000               166 DTCB     ZERO     A$TCB
    000456  000000 000000               167 RETRN    ZERO
    000457  000461 0000 40              168 OUT      TALLYB   OUTB
    000460  000461 0000 40              169 OUTP     TALLYB   OUTB
                      000461            170 OUTB     BSS      300
    001135  001137 0000 40              171 IN       TALLYB   INB
    001136  001137 0000 40              172 INP      TALLYB   INB
                      001137            173 INB      BSS      100
                                        174          DETAIL   SAVE,OFF
                      001303            175          DUP      1,100
    001303  777777 000000               176 BTAB     ZERO     -1
 END OF BINARY CARD 00000017
                                        177          DETAIL   RESTORE
    001447  004514 7172 00              178 BXED     XED      BKST
    001450  001451 0003 40              179          TALLYB   CRLF,3
    001451  015012000000                180 CRLF     OCT      015012000000
    001452  001453 0027 40              181          TALLYB   BPIT,23
    001453  015012102122                182 BPIT     OCT      015012102122
    001454  105101113120                183          UASCI    5,EAKPOINT IN TABLE
    001455  117111116124
    001456  040111116040
    001457  124101102114
    001460  105040040040
    001461  001462 0011 40              184          TALLYB   BREAK,9
    001462  015012102122                185 BREAK    OCT      015012102122
    001463  105101113040                186          UASCI    2,EAK
    001464  040040040040
    001465  001466 0030 40              187          TALLYB   BNF,24
    001466  015012102122                188 BNF      OCT      015012102122
    001467  105101113120                189          UASCI    5,EAKPOINT NOT FOUND
    001470  117111116124
    001471  040116117124
    001472  040106117125
 END OF BINARY CARD 00000018
    001473  116104040040
    001474  001475 0010 40              190          TALLYB   ERRM,8
    001475  015012105122                191 ERRM     OCT      015012105122
    001476  122117122040                192          UASCI    1,ROR
    001477  001500 0016 40              193          TALLYB   BPMS,14
    001500  015012102122                194 BPMS     OCT      015012102122
    001501  105101113120                195          UASCI    3,EAKPOINT
    001502  117111116124
    001503  040040040040
                      001504            196          EVEN
    001504  015012102122                197 BMESS    OCT      015012102122,105101113040
```

```
              Z                       DEBUGGER

     001505  105101113040
     001506  001137 000144           198 RPRAM  ZERO     INB,100
     001507  120101004132            199 CTAB   VFD      018/120101,18/PAT
     001510  104125004142            200        VFD      018/104125,18/DUM
     001511  122105004210            201        VFD      018/122105,18/RDUM
     001512  124122004335            202        VFD      018/124122,18/TRA
     001513  103117004345            203        VFD      018/103117,18/CON
     001514  102122004351            204        VFD      018/102122,18/BRE
     001515  125116004400            205        VFD      018/125116,18/UNB
     001516  124131004220            206        VFD      018/124131,18/TYPE
     001517  105130004430            207        VFD      018/105130,18/EXIT
     001520  124111004267            208        VFD      018/124111,18/TIME
  END OF BINARY CARD 00000019
     001521  777777004432            209        VFD      018/777777,18/ERR
     001522  777777004432            210        VFD      018/777777,18/ERR
     001523  777777004432            211 CTABE  VFD      018/777777,18/ERR
```

```
                                  212          TTLS     ALGOL68
                                  213          HEAD     A
                 777777           214 ERROR    EQU      -1
                 000144           215 BLEN     EQU      100
                 001524           216          EVEN
  001524  001530000144            217 CPRAM    VFD      18/INBUF,18/BLEN,18/3
  001525  000003000000
  001526  000000015012            218 CRLF     OCT      015012
  001527  000000 000000           219          ZERO
                 001530           220 INBUF    BSS      BLEN
  001674  001526 0003 42          221 INI      TALLYB   CRLF,3,2
  001675  001526 0003 42          222 IN       TALLYB   CRLF,3,2
  001676  001530 0621 40          223 INP      TALLYB   INBUF,4*BLEN+1
  001677  000000 000000           224 EOF      ZERO
                 001700           225 IBUF     BSS      40
  001750  001700 7776 40          226 NITAL    TALLYB   IBUF,-2
  001751  000000 000000           227 IDENT    ZERO
  001752  000000 000000           228 NICHR    ZERO
  001753  001752 0000 41          229 NICH     TALLYB   NICHR,0,1
  001754  777777000400            230 BACK4    OCT      777777000400
                                  231 TE       MACRO    LOWER LIMIT;UPPER LIMIT;ROUTINE ADDRESS
                                  232          VFD      018/#1,036/#2,18/#3
                                  233          ENDM     TE
  001755  000000011207
                 001756           234          EVEN
                 001756           235 NIT8     TE       060,071,NILN     DIGIT
                 001760           236          TE       012,012,NISK     LINE-FEED
END OF BINARY CARD 00000020
                 001762           237 NIT7     TE       012,012,NILN     LINE-FEED
                 001764           238 NIT1     TE       101,132,NIDEN    LETTER
                 001766           239          TE       060,071,NIDIG    DIGIT
                 001770           240          TE       015,015,NICR     CARRIAGE RETURN
                 001772           241          TE       000,040,NISK     CONTROL CHARACTER OR SPACE
                 001774           242          TE       056,056,NIDOT    PERIOD
                 001776           243          TE       100,100,NICOM    AT SIGN
                 002000           244          TE       042,042,NIQU     QUOTE
                 002002           245          TE       044,044,NIFOR    DOLLAR SIGN
                 002004           246          TE       050,051,NIXIT    LEFT OR RIGHT PARENTHESIS
                 002006           247          TE       054,054,NIXIT    COMMA
END OF BINARY CARD 00000021
                 002010           248          TE       073,073,NIXIT    SEMICOLON
                 002012           249          TE       133,133,NIXIT    LEFT BRACKET
                 002014           250          TE       135,135,NIXIT    RIGHT BRACKET
                 002016           251          TE       041,137,NIOPR    ALL OTHER SPECIAL CHARACTERS
                 002020           252          TE       1000,1000,NIEOF  END OF FILE
                 002022           253 NIT2     TE       101,132,NIDEN    LETTER
                 002024           254          TE       060,071,NIDEN    DIGIT
                 002026           255          TE       000,137,NIBEX    ALL OTHERS
                 002030           256          TE       1000,1000,NIBEX  END OF FILE
                 002032           257 NIT3     TE       060,071,NIDIG    DIGIT
```

```
           A                           ALGOL68

                    002034       258          TE       056,056,NIDEC   PERIOD
 END OF BINARY CARD 00000022
                    002036       259          TE       105,105,NIEXP   LETTER E
                    002040       260          TE       000,137,NIBEX   ALL OTHERS
                    002042       261          TE       1000,1000,NIBEX END OF FILE
                    002044       262 NIT4     TE       060,071,NIDC1   DIGIT
                    002046       263          TE       000,137,NIB2X   ALL OTHERS
                    002050       264          TE       1000,1000,NIB2X END OF FILE
                    002052       265 NIT5     TE       053,053,NIEX1   PLUS SIGN
                    002054       266          TE       055,055,NIEX1   MINUS SIGN
                    002056       267          TE       060,071,NIDIG   DIGIT
                    002060       268          TE       000,137,NIBEX   ALL OTHERS
                    002062       269          TE       1000,1000,NIBEX END OF FILE
 END OF BINARY CARD 00000023
                    002064       270 NIT6     TE       060,071,NIDG1   DIGIT
                    002066       271          TE       000,137,NIB2X   ALL OTHERS
                    002070       272          TE       1000,1000,NIB2X END OF FILE
                    002072       273 NIT9     TE       060,071,NIDIG   DIGIT
                    002074       274          TE       000,137,NIBEX   ALL OTHERS
                    002076       275          TE       1000,1000,NIBEX END OF FILE
                    002100       276 NIT10    TE       100,100,NISK    AT SIGN
                    002102       277          TE       000,137,NICOM   ALL OTHERS
                    002104       278          TE       1000,1000,NIEOF END OF FILE
                    002106       279 NIT11    TE       057,057,NIOPR   SLASH
                    002110       280          TE       072,072,NIOPR   COLON
 END OF BINARY CARD 00000024
                    002112       281          TE       075,075,NIOPR   EQUAL
                    002114       282          TE       000,137,NIBEX   ALL OTHERS
                    002116       283          TE       1000,1000,NIBEX END OF FILE
                    002120       284 NIT12    TE       015,015,NIBEX   CARRIAGE RETURN
                    002122       285          TE       042,042,NIQU1   QUOTE
                    002124       286          TE       000,137,NIQU    ALL OTHERS
                    002126       287          TE       1000,1000,NIBEX END OF FILE
                    002130       288 NIT13    TE       042,042,NIQU    QUOTE
                    002132       289          TE       000,137,NIBEX   ALL OTHERS
                    002134       290          TE       1000,1000,NIBEX END OF FILE
                    002136       291 NIT14    TE       044,044,NIXIT   DOLLAR SIGN
 END OF BINARY CARD 00000025
                    002140       292          TE       015,015,NIBEX   CARRIAGE RETURN
                    002142       293          TE       000,137,NIFOR   ALL OTHERS
                    002144       294          TE       1000,1000,NIBEX END OF FILE
                    002146       295          TE       0000,1000,ERROR CATCH ALL WILD TABLE SEARCHES
     002150  777777000077        296 NTMSK    OCT      777777000077
                    000000       297 SC       EQU      0
                    000001       298 DF       EQU      1
     002151  000114 0004 43      299 SYMSC    TALLYB   ITABX,4,ITACX
     002152  000000 000000       300 PEEKF    ZERO
     002153  000000 000000       301 QEEKF    ZERO
     002154  000000 000000       302 TEMP1    ZERO
     002155  000000 000000       303 TEMP2    ZERO
```

```
          A                                ALGOL68

     002156  000000 000000             304 TEMP3  ZERO
     002157  002160 000640             305 LONGT  TALLYB  **1,6
     002160  114117116107              306        UASCI   2,LONG
     002161  040040040040
     002162  000000 000000             307 LNGCT  ZERO
                                       308        TTL     EXECUTIVE
```

```
                TTLS     GENERAL MACROS
        QUEUE   MACRO    CURRENT TASK,QUEUE
                INHIB    SAVE,ON
                EAX0     #1                 GET POINTER TO TASK CONTROL BLOCK
                EAX1     #2                 ADDRESS OF QUEUE WHERE TASK IS TO BE INSERTED
                LDA      1,0                GET PRIORITY OF CURRENT TASK
        #3      EAX2     0,1                SAVE POINTER TO POINTER TO TASK BLOCK BEING CHECK
                LDX1     0,2                GET POINTER TO FOLLOWING TASK
                TZE      #4                 TRANSFER IF NO FOLLOWING TASK
                CMPA     1,1                CHECK PRIORITY OF QUEUED TASK
                TRC      #3                 TRANSFER TO CONTINUE SEARCH
        #4      STX1     0,0                STORE POINTER TO FOLLOWING TASK IN CURRENT TASK
                STX0     0,2                STORE POINTER TO CURRENT TASK IN PREVIOUS TASK
                INHIB    RESTORE
                ENDM     QUEUE
        *
        RESET   MACRO    RESTART LOCATION
                INHIB    SAVE,ON
                LDX0     #1,DU              GET ADDRESS OF RESTART IN XR - 0
                STX0     ,$IC,I             SAVE IN SAVED INSTRUCTION COUNTER
                TRA      ,$EXIT             AND EXIT ROUTINE WITHOUT PRESERVING REGISTERS
                INHIB    RESTORE
                ENDM     RESET
        *
        DOWN    MACRO    SEMAPHORE
                INHIB    SAVE,ON
                SREG     ,$REG,I            SAVE REGISTERS
                SZN      #1                 IS SEMAPHORE IN USE
                TZE      #2                 TRANSFER IF NOT IN USE
                STI      ,$IC,I             SAVE INDICATORS
                LDX0     #3,DU              GET RESTART ADDRESS
                STX0     ,$IC,I             SAVE AS INSTRUCTION COUNTER
                QUEUE    (,$LINK,I),(#1)    ADD TASK TO SEMAPHORE QUEUE
                TRA      ,$EXIT             AND EXIT
        #2      AOS      #1                 MARK SEMAPHORE BUSY
        #3      EQU      *                  END LOCATION
                INHIB    RESTORE
                ENDM     DOWN
        *
        UP      MACRO    SEMAPHORE
                INHIB    SAVE,ON
                SREG     ,$REG,I            SAVE REGISTERS
                STI      ,$IC,I             SAVE INDICATORS
                LDX0     #3,DU              GET RESTART ADDRESS
                STX0     ,$IC,I             SAVE AS INSTRUCTION COUNTER
                QUEUE    (,$LINK,I),,$TASK  PUT CURRENT TASK ON ACTIVE TASK QUEUE
                SZN      #1                 IS THIS SEMAPHORE BUSY
                TZE      ERROR              NO - ERROR
                LDX0     #1                 GET ADDRESS OF FIRST TASK QUEUED ON SEMAPHORE
                TNZ      #2                 TRANSFER IF THERE IS ONE
```

```
           STZ       #1                 MAKE SEMAPHORE IDLE
           TRA       ,$EXIT             AND CONTINUE
#2         LDX1      0,0                GET ADDRESS OF FOLLOWING TASK IN QUEUE
           STX1      #1                 MAKE TOP TASK IN QUEUE
           QUEUE     (0,0),,$TASK       PUT PREVIOUS TOP TASK ON QUEUE
           TRA       ,$EXIT             AND CONTINUE
#3         EQU       *
           INHIB     RESTORE
           ENDM      UP
*
UPS        MACRO     SEMAPHORE
           INHIB     SAVE,ON
           SZN       #1                 IS THIS SEMAPHORE BUSY
           TZE       ERROR              NO - ERROR
           LDX0      #1                 GET ADDRESS OF FIRST TASK QUEUED ON SEMAPHORE
           TNZ       #2                 TRANSFER IF THERE IS ONE
           STZ       #1                 MAKE SEMAPHORE IDLE
           TRA       #3                 AND CONTINUE
#2         LDX1      0,0                GET ADDRESS OF FOLLOWING TASK IN QUEUE
           STX1      #1                 MAKE TOP TASK IN QUEUE
           QUEUE     (0,0),,$TASK       PUT PREVIOUS TOP TASK ON QUEUE
#3         EQU       *
           INHIB     RESTORE
           ENDM      UPS
*
EXEC       MACRO     REG,MME ADDRESS
           INHIB     SAVE,ON
           LDX0      ,$NIC,DU           GET POINTER TO NULL ROUTINE IC
           STX0      ,$IC               STORE IN IC POINTER
           LDX0      ,$NREG,DU          GET POINTER TO NULL ROUTINE REGISTERS
           STX0      ,$REG              STORE IN REGISTER POINTER
           LREG      #1                 GET REGISTERS FOR EXEC CALL
           LDX6      ,$LINK             GET ADDRESS OF TASK CONTROL BLOCK IN XR - 6
           ADX6      2,DU               PUT TRAP ADDRESS IN XR - 6
           STZ       ,$LINK             SET NULL ROUTINE FLAG FOR TRAP ROUTINE
           MME       M$#2               ISSUE EXEC CALL
           TRA       ,$EXIT             AND EXIT
           INHIB     RESTORE
           ENDM      EXEC
```

```
                                     TTLS    FAULTS, INTERRUPTS, AND EXITS
                                     INHIB   ON
                                     HEAD    ,
     002163  000000011207
                        002164       EVEN
     002164  000221 5542 51    TRAP  STC1    IC,I               SAVE IC IN TASK CONTROL BLOCK
 END OF BINARY CARD 00000026
     002165  002166 7102 00          TRA     *+1                BREAK XED
     002166  000220 7532 51          SREG    REG,I              SAVE REGISTERS IN TASK CONTROL BLOCK
     002167  000226 4502 00          STZ     FFLAG              INITIALIZE FAULT FLAG
     002170  000221 2202 51          LDX0    IC,I               GET SAVED IC OR POINTER TO TASK CONTROL BLOCK
     002171  000001 1602 03          SBX0    1,DU               GET POINTER TO XED THAT CALLED THIS ROUTINE
     002172  777777 2352 10    TRAP1 LDA     -1,0               GET SAVED IC STORED ON TRAP OR FAULT
     002173  000221 7552 51          STA     IC,I               STORE AS IC FOR CURRENT TASK
     002174  000043 1002 03          CMPX0   35,DU              SEE IF THIS IS A FAULT OR TRAP
     002175  002232 6032 00          TRC     TRAP2              TRANSFER IF TRAP
     002176  000021 1002 03          CMPX0   16+1,DU            IS THIS A SPECIAL INTERRUPT
     002177  002213 6012 00          TNZ     TRAP4              TRANSFER IF OTHER FAULT
                        002200       QUEUE   $STCB,,$TASK       START UP SPECIAL INTERRUPT ROUTINE
     002212  002244 7102 00          TRA     TRAP3              AND GO CONTINUE PROCESSING FAULTS AND TRAPS
 END OF BINARY CARD 00000027
     002213  000070 4402 00    TRAP4 SXL0    $MTCB+8            STORE FAULT NUMBER IN MASTER TASK XR - 1
     002214  000223 2202 00          LDX0    LINK               GET ADDRESS OF CURRENT TASK CONTROL BLOCK
     002215  000070 7402 00          STX0    $MTCB+8            STORE IN MASTER TASK XR - 0
     002216  000226 7502 00          STC2    FFLAG              SET FAULT FLAG
                        002217       QUEUE   $MTCB,TASK         START MASTER TASK RUNNING
     002231  002244 7102 00          TRA     TRAP3              CONTINUE PROCESSING TRAPS AND FAULTS
                        002232 TRAP2 QUEUE   (-5,0),TASK        ADD TRAPPED TASK TO TASK QUEUE
 END OF BINARY CARD 00000028
     002244  000221 2202 51    TRAP3 LDX0    IC,I               GET POINTER TO NEXT TRAP OR FAULT IF ANY
     002245  000043 1002 03          CMPX0   35,DU              IS IT A FAULT
     002246  002172 6022 00          TNC     TRAP1              TRANSFER IF FAULT
     002247  000000 2352 10          LDA     0,0                LOAD A WITH A CALL TO THIS ROUTINE IF A TRAP
     002250  000225 1152 00          CMPA    XEDT               IS IT A CALL TO THIS ROUTINE
     002251  002172 6002 00          TZE     TRAP1              TRANSFER IF TRAP
     002252  000226 2342 00          SZN     FFLAG              DID A FAULT OCCUR
     002253  002270 6012 00          TNZ     EXIT               YES - DO NOT RETURN TO CURRENT TASK
     002254  000223 2342 00          SZN     LINK               IS THERE A CURRENT TASK
     002255  002270 6002 00          TZE     EXIT               NO - NOTHING TO QUEUE
                        002256       QUEUE   (LINK,I),TASK      PUT CURRENT TASK IN TASK QUEUE
 END OF BINARY CARD 00000029
     002270  000224 2202 00    EXIT  LDX0    TASK               GET POINTER TO TASK QUEUE
     002271  777777 6002 00          TZE     ERROR              THERE SHOULD BE AT LEAST THE DUMMY TASK
     002272  000223 7402 00          STX0    LINK               STORE POINTER TO CURRENT TASK CONTROL BLOCK
     002273  000002 6212 10          EAX1    2,0                GET ADDRESS OF STATUS WORDS
     002274  000222 7412 00          STX1    STAT               AND STORE
     002275  000007 6212 10          EAX1    7,0                GET POINTER TO CURRENT SAVED IC
     002276  000221 7412 00          STX1    IC                 AND STORE
     002277  000010 6212 10          EAX1    8,0                GET POINTER TO CURRENT SAVED REGISTERS
     002300  000220 7412 00          STX1    REG                AND STORE
```

```
002301  000000 2212 10        443        LDX1     0,0          GET POINTER TO NEXT TASK IN QUEUE
002302  000224 7412 00        444        STX1     TASK         MAKE IT TOP TASK IN QUEUE
002303  000220 0732 51        445        LREG     REG,I        RESTORE REGISTERS OF CURRENT TASK
002304  000221 6302 51        446        RET      IC,I         AND RETURN
```

```
                                       FAULT AND INTERRUPT TASKS

                                  447          TTLS    FAULT AND INTERRUPT TASKS
                                  448          HEAD
                                  449          INHIB   ON
     002305  777777 4502 11       450 MTASK    STZ     -1,1               CLEAR OUT FAULT IC WORD
     002306  000007 2352 10       451 FT       LDA     7,0                GET CURRENT IC
     002307  000006 7552 10       452          STA     6,0                SAVE IT
     002310  000002 7412 10       453          STX1    2,0                SAVE TYPE OF FAULT
     002311  002465 2352 03       454          LDA     $FAULT,DU          GET ADDRESS OF FAULT ROUTINE
     002312  000007 7552 10       455          STA     7,0                AND CAUSE TASK TO START UP THERE
                        002313    456          QUEUE   (0,0),,$TASK       RESTART FAULTED TASK
  END OF BINARY CARD 00000030
                        002325    457          RESET   MTASK              SET UP NEXT ENTRANCE TO THIS ROUTINE
                                  458 *
                                  459          INHIB   ON
     002330  000001 2352 00       460 SPEC     LDA     1                  LOAD SPECIAL INTERRUPT TALLY WORD
     002331  000244 1152 00       461          CMPA    SIF                COMPARE WITH INITIAL VALUE OF TALLY WORD
     002332  002337 6012 00       462          TNZ     SPEC1              TRANSFER IF THERE IS SOMETHING IN SPECIAL QUEUE
     002333  000020 4502 00       463          STZ     16                 ALLOW SPECIAL INTERRUPTS
                        002334    464          RESET   SPEC               SET UP NEXT ENTRANCE TO THIS ROUTINE
A    002337  000001 2372 44       465 SPEC1    LDAQ    1,SD               FETCH SPECIAL INTERRUPT WORD
     002340  000272 1152 00       466          CMPA    TTYSI              IS IT A TTY SPECIAL
     002341  002330 6012 00       467          TNZ     SPEC               IGNORE ALL OTHER SPECIALS
     002342  000232 2342 00       468          SZN     STTYF              IS THERE A TASK WAITING FOR A SPECIAL
  END OF BINARY CARD 00000031
     002343  002330 6002 00       469          TZE     SPEC               IF NOT - IGNORE IT
                        002344    470          UP      STTYF              START UP TASK WAITING FOR SPECIAL
  END OF BINARY CARD 00000032
     002405  002330 7102 00       471          TRA     SPEC               AND LOOP
                                  472          INHIB   ON
     002406  002406 7102 00       473 STRT1    TRA     *                  A TRANSFER TO ZERO HAS OCCURRED - ERROR
     002407  002406 2352 00       474 START    LDA     STRT1              GET NEW INSTRUCTION FOR LOCATION ZERO
     002410  000000 7552 00       475          STA     0                  AND STORE IT THERE
                        002411    476          QUEUE   Z$TCB,,$TASK       START UP DEBUGGER
  END OF BINARY CARD 00000033
                        002423    477          QUEUE   B$TCB,,$TASK       START UP PRINT BUFFER ROUTINE
                        002435    478          QUEUE   $ITCB,,$TASK       START UP IDLE PROGRAM
  END OF BINARY CARD 00000034
                        002447    479          QUEUE   A$TCB,,$TASK       START UP COMPILER
     002461  002270 7102 00       480          TRA     ,$EXIT             AND EXIT
     002462  000000 2252 03       481 IDLE     LDX5    0,DU               LOAD PARAMETER FOR PAUSE
     002463  500005 0012 00       482          MME     M$PAUSE            PAUSE FOR ANYTHING
     002464  002462 7102 00       483          TRA     IDLE               AND LOOP
```

```
                                     484        TTLS     FAULT ROUTINE
                                     485        INHIB    ON
                          002465     486 FAULT  DOWN     TTYF                SEIZE TTY DEVICE
END OF BINARY CARD 00000035
      002507   000223 2202 00        487        LDX0     ,$LINK              GET POINTER TO TASK CONTROL BLOCK IN XR - 0
      002510   000222 2212 51        488        LDX1     ,$STAT,I            GET FAULT NUMBER IN XR - 1
      002511   000001 0612 03        489        ADX1     1,DU                FUDGE XR - 1 FOR REST OF ROUTINE
      002512   000016 1012 03        490        CMPX1    12+2,DU             IS IT A DERAIL FAULT
      002513   002527 6012 00        491        TNZ      F2                  NO - TRANSFER
      002514   000006 2272 10        492        LDX7     6,0                 GET ADDRESS OF FAULT
      002515   000001 1672 03        493        SBX7     1,DU                GET ADDRESS OF DERAIL INSTRUCTION
      002516   004626 7002 00        494        TSX0     Z$BTLU              SEE IF ADDRESS IS IN BREAKPOINT TABLE
      002517   002525 7102 00        495        TRA      F1                  NO - TRANSFER
      002520   000223 2202 00        496        LDX0     ,$LINK              RESTORE XR - 0
END OF BINARY CARD 00000036
      002521   000006 7472 10        497        STX7     6,0                 DECREMENT SAVED IC TO POINT TO DERAIL
      002522   000222 2212 51        498        LDX1     ,$STAT,I            RESTORE XR - 1
      002523   001504 2372 00        499        LDAQ     Z$BMESS             GET BREAKPOINT MESSAGE
      002524   002530 7102 00        500        TRA      F3                  AND PRINT IT OUT
      002525   000223 2202 00        501 F1     LDX0     ,$LINK              RESTORE XR - 0
      002526   000222 2212 51        502        LDX1     ,$STAT,I            RESTORE XR - 1
      002527   000410 2372 11        503 F2     LDAQ     FTAB-2,1            GET FAULT MESSAGE
      002530   000234 7572 00        504 F3     STAQ     TTYB                AND STORE IN OUTPUT BUFFER
      002531   000236 2352 03        505        LDA      TTYB+2,DU           GET SC TALLY WORD FOR REST OF BUFFER
      002532   000040 0752 07        506        ADA      =040,DL             SET 9-BIT CHARACTER BIT IN TALLY WORD
      002533   000230 7552 00        507        STA      TTY                 STORE TALLY WORD
      002534   000006 2362 10        508        LDQ      6,0                 LOAD Q WITH IC AT TIME OF FAULT
      002535   002614 7072 00        509        TSX7     ADDR                TYPE OUT ADDRESS
      002536   000040 2362 07        510        LDQ      =0040,DL            GET A SPACE
      002537   000230 7562 52        511        STQ      TTY,SC              AND ADD TO OUTPUT
      002540   000000 6362 10        512        EAQ      0,0                 GET ADDRESS OF TASK CONTROL BLOCK OF FAULT
      002541   002614 7072 00        513        TSX7     ADDR                AND TYPE OUT ADDRESS
      002542   000015 2362 07        514        LDQ      =0015,DL            CARRIAGE RETURN
      002543   000230 7562 52        515        STQ      TTY,SC              STORE IN OUTPUT
      002544   000012 2362 07        516        LDQ      =0012,DL            LINE FEED
      002545   000230 7562 52        517        STQ      TTY,SC              STORE IN OUTPUT
      002546   000177 2362 07        518        LDQ      =0177,DL            FILL CHARACTER - FILL OUT LAST WORD OF BUFFER
END OF BINARY CARD 00000037
      002547   000230 7562 52        519        STQ      TTY,SC              STORE IN OUTPUT
      002550   002564 2202 03        520        LDX0     F4,DU               GET ADDRESS OF RESTART ROUTINE
      002551   000221 7402 51        521        STX0     ,$IC,I              STORE AS SAVED INSTRUCTION COUNTER
                          002552     522        EXEC     LREG,WRITE          ISSUE WRITE MME
      002564   000223 2202 00        523 F4     LDX0     ,$LINK              GET POINTER TO TASK CONTROL BLOCK
      002565   000006 2352 10        524        LDA      6,0                 GET SAVED IC
      002566   000007 7552 10        525        STA      7,0                 STORE IN INSTRUCTION COUNTER
      002567   000006 4502 10        526        STZ      6,0                 CLEAR OUT SAVED IC
                          002570     527        UPS      TTYF                FREE UP TTY
END OF BINARY CARD 00000038
      002612   000456 4502 00        528        STZ      Z$RETRN             RESET RETURN FLAG
      002613   002270 7102 00        529        TRA      ,$EXIT              AND EXIT
```

FAULT ROUTINE

```
002614  000006 2262 03         530 ADDR   LDX6    6,DU              INITIALIZE DIGIT COUNTER TO 6
002615  000006 2352 07         531 ADDR1  LDA     =006,DL           GET LEFT HALF OF ASCII DIGIT IN A (06X)
002616  000003 7372 00         532        LLS     3                 GET ASCII OCTAL DIGIT IN A REGISTER
002617  000230 7552 52         533        STA     TTY,SC            STORE IN OUTPUT
002620  000001 1662 03         534        SBX6    1,DU              DECREMENT DIGIT COUNTER
002621  002615 6012 00         535        TNZ     ADDR1             TRANSFER IF MORE DIGITS TO CONVERT
002622  000000 7102 17         536        TRA     0,7               RETURN
```

```
                                            537          TTLS     TELETYPE TRANSPUT ROUTINES
                                            538          INHIB    ON
   END OF BINARY CARD 00000039
        002623  000000 2352 10              539 TTB      LDA      0,0                    GET TALLY WORD FOR DATA
        002624  000001 0602 03              540          ADX0     1,DU                   GET RETURN ADDRESS IN XR - 0
        002625  002674 7402 00              541          STX0     TTBX                   SAVE RETURN
        002626  002766 7552 00              542          STA      IN                     STORE TALLY WORD AS INPUT TALLY WORD
        002627  002766 2352 52              543 TTB1     LDA      IN,SC                  GET NEXT CHARACTER TO OUTPUT
        002630  002675 6072 00              544          TTF      TTB2                   TRANSFER IF THERE IS ONE
        002631  003260 2342 00              545          SZN      IOF                    IS THE BUFFER OUTPUT TASK CURRENTLY BUSY
        002632  002674 6012 00              546          TNZ      TTBX                   YES - NO NEED TO START IT
                            002633          547          UP       BUFS                   START BUFFER OUTPUT ROUTINE
   END OF BINARY CARD 00000040
        002674  000000 7102 00              548 TTBX     TRA      **                     AND RETURN TO CALLER
        002675  003256 7552 52              549 TTB2     STA      OUT,SC                 STORE CHARACTER IN OUTPUT BUFFER
        002676  002627 6072 00              550          TTF      TTB1                   TRANSFER IF MORE ROOM IN BUFFER
   END OF BINARY CARD 00000041
        002677  003260 2342 00              551          SZN      IOF                    IS THE BUFFER OUTPUT TASK CURRENTLY BUSY
        002700  002743 6012 00              552          TNZ      TTB3                   TRANSFER IF MUST WAIT FOR A FREE BUFFER
                            002701          553          UP       BUFS                   START BUFFER OUTPUT ROUTINE
   END OF BINARY CARD 00000042
        002742  002627 7102 00              554          TRA      TTB1                   AND CONTINUE IN NEW BUFFER
                            002743          555 TTB3     DOWN     RELS                   WAIT FOR A FREE BUFFER
   END OF BINARY CARD 00000043
        002765  002627 7102 00              556          TRA      TTB1                   AND CONTINUE
        002766  000000 000000               557 IN       ZERO
                            002767          558 TTYR     DOWN     TTYF                   SEIZE TTY
   END OF BINARY CARD 00000044
        003011  000000 6202 01              559          EAX0     0,AU                   GET ADDRESS OF READ IN XR - 0
        003012  000410 4402 00              560          SXL0     LOC                    STORE IN MEMORY
        003013  000000 6202 05              561          EAX0     0,AL                   GET LENGTH OF READ IN XR - 0
        003014  000411 4402 00              562          SXL0     LEN                    STORE IN MEMORY
        003015  003073 6202 00              563          EAX0     TTYTR                  GET ADDRESS OF TTY WRAPUP ROUTINE IN XR - 0
        003016  000221 7402 51              564          STX0     ,$IC,I                 SAVE AS IC
                            003017          565          EXEC     RREG,READ              ISSUE READ MME
   END OF BINARY CARD 00000045
                            003031          566 TTYW     DOWN     TTYF                   SEIZE TTY
        003053  000000 6202 01              567          EAX0     0,AU                   GET ADDRESS OF WRITE IN XR - 0
        003054  000410 4402 00              568          SXL0     LOC                    STORE IN MEMORY
   END OF BINARY CARD 00000046
        003055  000000 6202 05              569          EAX0     0,AL                   GET LENGTH OF WRITE IN XR - 0
        003056  000411 4402 00              570          SXL0     LEN                    STORE IN MEMORY
        003057  003073 6202 00              571          EAX0     TTYTR                  GET ADDRESS OF TTY WRAPUP ROUTINE IN XR - 0
        003060  000221 7402 51              572          STX0     ,$IC,I                 SAVE AS IC
                            003061          573          EXEC     WREG,WRITE             ISSUE WRITE MME
                            003073          574 TTYTR    UP       TTYF                   RELEASE TTY
   END OF BINARY CARD 00000048
        003134  000000 7102 10              575          TRA      0,0                    AND RETURN
                            003135          576 BUF1     DOWN     BUFS                   GET A BUFFER WITH SOMETHING TO OUTPUT
   END OF BINARY CARD 00000049
```

```
      003157  003257 2352 00       BUF2   LDA     OUT1            GET INITIAL TALLY WORD IN BUFFER
      003160  003256 1152 00              CMPA    OUT             COMPARE WITH CURRENT TALLY WORD
      003161  003135 6002 00              TZE     BUF1            IF EQUAL - NOTHING TO DO - TRANSFER
      003162  000000 2352 07              LDA     0,DL            GET A NULL CHARACTER IN AL
      003163  000003 2362 07              LDQ     3,DL            GET A CHARACTER POSITION MASK IN QL
      003164  003256 3162 00       BUF3   CANQ    OUT             SEE IF CURRENT TALLY WORD HAS FILLED OUT A WORD
      003165  003170 6002 00              TZE     BUF4            TRANSFER IF WORD IS FILLED OUT
      003166  003256 7552 52              STA     OUT,SC          STORE A NULL IN OUTPUT BUFFER
      003167  003164 7102 00              TRA     BUF3            AND GO CHECK AGAIN
      003170  003257 2352 00       BUF4   LDA     OUT1            GET POINTER TO BUFFER IN AU
      003171  003256 2202 00              LDX0    OUT             GET POINTER TO END OF BUFFER IN XR - 0
      003172  003257 1602 00              SBX0    OUT1            SUBTRACT LOCATION TO GET LENGTH OF FILLED BUFFER
      003173  000000 6362 10              EAQ     0,0             GET LENGTH OF IO OPERATION IN QU
      003174  000022 7712 00              ARL     18              MOVE LOCATION NEXT TO LENGTH
      003175  000022 7372 00              LLS     18              GET LOC/LEN IN A REGISTER
      003176  003261 2362 00              LDQ     SWAPC           GET BUFFER SWAPPING CONSTANT IN Q
      003177  003257 1762 00              SBQ     OUT1            SUBTRACT OLD INITIAL TALLY WORD
      003200  003257 7562 00              STQ     OUT1            STORE NEW INITIAL TALLY WORD
      003201  003256 7562 00              STQ     OUT             STORE NEW CURRENT TALLY WORD
      003202  003260 7502 00              STC2    IOF             SET I/O FLAG
      003203  003031 7002 00              TSX0    $TTYW           AND DO TTY OUTPUT
      003204  000222 2372 51              LDAQ    ,$STAT,I        GET STATUS RETURN WORDS IN AQ REGISTER
  END OF BINARY CARD 00000050
      003205  000001 1152 03              CMPA    1,DU            SEE IF ANY ERROR
      003206  777777 6032 00              TRC     $ERROR          FAIL ON ANY ERROR
      003207  003260 4502 00              STZ     IOF             RESET I/O FLAG
      003210  003255 2202 00              LDX0    RELS            CHECK IF OUTPUT WAITING FOR THIS ROUTINE
      003211  003253 6002 00              TZE     BUF5            TRANSFER IF NOT
                         003212           UP      RELS            RELEASE INPUT BUFFER
  END OF BINARY CARD 00000051
      003253  003157 7102 00       BUF5   TRA     BUF2            AND LOOP
      003254  000000 000001        BUFS   ZERO    0,1
      003255  000000 000001        RELS   ZERO    0,1
      003256  003262 1000 40       OUT    TALLYB  BUF,2*BLEN
      003257  003262 1000 40       OUT1   TALLYB  BUF,2*BLEN
      003260  000000 000000        IOF    ZERO
  END OF BINARY CARD 00000052
      003261  006744 2001 00       SWAPC  TALLY   2*BUF+BLEN/2,4*BLEN+1
                         000400    BLEN   BOOL    400
                         003262    BUF    EQU     *-RELZER
                         003262           BSS     BLEN
```

```
                                    TTLS    FILE TRANSPUT ROUTINES
  003662  000220 7532 51      OPEN  SREG    ,$REG,I           SAVE REGISTERS
  003663  000342 7412 00            STX1    OREG+2            STORE ACCESS IN XR - 4
  003664  000340 7422 00            STX2    OREG              STORE FILE REFERENCE NUMBER OF CATALOG IN XR - 0
  003665  000344 7572 00            STAQ    OREG+4            STORE NAME IN AQ REGISTER
  003666  004063 6202 00            EAX0    FTRAP             GET POINTER TO TRAP ROUTINE IN XR - 0
  003667  000221 7402 51            STX0    ,$IC,I            AND STORE AS INSTRUCTION COUNTER
                 003670             EXEC    OREG,OPEN         ISSUE OPEN MME
  003702  000220 7532 51      CLOSE SREG    ,$REG,I           SAVE REGISTERS
  003703  000000 6202 02            EAX0    0,QU              GET FILE REFERENCE NUMBER IN XR - 0
  003704  000340 7402 00            STX0    OREG              AND STORE IN XR - 0
  003705  004063 6202 00            EAX0    FTRAP             GET POINTER TO TRAP ROUTINE IN XR - 0
END OF BINARY CARD 00000053
  003706  000221 7402 51            STX0    ,$IC,I            AND STORE AS INSTRUCTION COUNTER
                 003707             EXEC    OREG,CLOSE        ISSUE CLOSE MME
  003721  000220 7532 51      READ  SREG    ,$REG,I           SAVE REGISTERS
  003722  000000 6202 01            EAX0    0,AU              GET CORE ADDRESS OF DATA
  003723  000410 4402 00            SXL0    LOC               STORE AS CORE POINTER IN MEMORY
  003724  000000 6202 05            EAX0    0,AL              GET NUMBER OF WORDS TO TRANSFER IN XR - 0
  003725  000411 4402 00            SXL0    LEN               STORE NUMBER OF WORDS TO TRANSFER IN LOW HALF
  003726  000000 6202 02            EAX0    0,QU              GET FILE REFERENCE NUMBER IN XR - 0
  003727  000330 7402 00            STX0    FREG              STORE AS FILE REFERENCE NUMBER OF SOURCE FILE
  003730  004063 6202 00            EAX0    FTRAP             GET ADDRESS OF CLEANUP ROUTINE IN XR - 0
  003731  000221 7402 51            STX0    ,$IC,I            RESTART THERE AFTER TRAP
                 003732             EXEC    FREG,READ         ISSUE READ MME
END OF BINARY CARD 00000054
  003744  000220 7532 51      WRITE SREG    ,$REG,I           SAVE REGISTERS
  003745  000000 6202 01            EAX0    0,AU              GET CORE ADDRESS OF DATA
  003746  000410 4402 00            SXL0    LOC               STORE AS CORE POINTER IN MEMORY
  003747  000000 6202 05            EAX0    0,AL              GET NUMBER OF WORDS TO TRANSFER IN XR - 0
  003750  000411 4402 00            SXL0    LEN               STORE NUMBER OF WORDS TO TRANSFER IN LOW HALF
  003751  000000 6202 02            EAX0    0,QU              GET FILE REFERENCE NUMBER IN XR - 0
  003752  000331 7402 00            STX0    FREG+1            STORE FILE REFERENCE NUMBER OF DESTINATION
  003753  004063 6202 00            EAX0    FTRAP             GET ADDRESS OF CLEANUP ROUTINE IN XR - 0
  003754  000221 7402 51            STX0    ,$IC,I            RESTART THERE AFTER TRAP
                 003755             EXEC    FREG,WRITE        ISSUE WRITE MME
END OF BINARY CARD 00000055
  003767  000220 7532 51      SETP  SREG    ,$REG,I           SAVE REGISTERS
  003770  000000 6202 02            EAX0    0,QU              GET FILE REFERENCE NUMBER IN XR - 0
  003771  000400 7402 00            STX0    SREG              STORE IN REGISTERS
  003772  000404 7552 00            STA     SREG+4            STORE DESIRED POINTER IN REGISTERS
  003773  004063 6202 00            EAX0    FTRAP             GET ADDRESS OF CLEANUP ROUTINE
  003774  000221 7402 51            STX0    ,$IC,I            STORE AS RESTART ADDRESS
                 003775             EXEC    SREG,SETP         ISSUE MME
  004007  000220 7532 51      OPENS SREG    ,$REG,I           SAVE REGISTERS
END OF BINARY CARD 00000056
  004010  004063 6202 00            EAX0    FTRAP             GET ADDRESS OF CLEANUP ROUTINE IN XR - 0
  004011  000221 7402 51            STX0    ,$IC,I            RESTART THERE AFTER TRAP
                 004012             EXEC    OSREG,OPENS       ISSUE OPEN SCRATCH MME
  004024  000220 7532 51      CAT   SREG    ,$REG,I           SAVE REGISTERS
```

FILE TRANSPUT ROUTINES

```
     004025  000360 7422 00        661        STX2    CREG             STORE FILE REFERENCE NUMBER OF CATALOG
     004026  000361 7432 00        662        STX3    CREG+1           STORE FILE REFERENCE NUMBER OF FILE
     004027  000364 7572 00        663        STAQ    CREG+4           STORE NAME OF FILE TO BE CATALOGED
     004030  004063 6202 00        664        EAX0    FTRAP            GET ADDRESS OF CLEANUP ROUTINE IN XR - 0
     004031  000221 7402 51        665        STX0    ,$IC,I           RESTART THERE AFTER TRAP
                         004032    666        EXEC    CREG,CAT
  END OF BINARY CARD 00000057
     004044  000220 7532 51        667 TRUNC  SREG    ,$REG,I          SAVE REGISTERS
     004045  000370 7562 00        668        STQ     TRREG            STORE FILE REFERENCE NUMBER OF FILE TO TRUNCATE
     004046  000374 7552 00        669        STA     TRREG+4          STORE NEW LENGTH IN A REGISTER
     004047  004063 6202 00        670        EAX0    FTRAP            GET ADDRESS OF CLEANUP ROUTINE IN XR - 0
     004050  000221 7402 51        671        STX0    ,$IC,I           RESTART THERE AFTER TRAP
                         004051    672        EXEC    TRREG,TRUNC      ISSUE TRUNCATE MME
     004063  000000 7102 10        673 FTRAP  TRA     0,0              RETURN TO CALLER
                                   674        TTL     DEBUGGER
```

```
                                              675          TTLS     COMMAND INTERPRETER
                                              676          HEAD     Z
                                              677          INHIB    OFF
 END OF BINARY CARD 00000058
     004064  000244 2350 00                   678 DDT      LDA      $SIF                GET INITIAL SPECIAL INTERRUPT TALLY WORD
     004065  000001 7550 00                   679          STA      1                   AND STORE IN TALLY WORD
                          004066              680          DOWN     $STTYF              GET CONTROL ONLY ON TTY SPECIAL
     004110  000001 4500 00                   681          STZ      1                   DISABLE SPECIAL INTERRUPTS
     004111  001506 2350 00                   682          LDA      RPRAM               GET READ TTY CONTROL WORD
 END OF BINARY CARD 00000059
     004112  002767 7000 00                   683          TSX0     $TTYR               READ TTY ON SPECIAL
     004113  000222 2350 51                   684          LDA      ,$STAT,I            GET STATUS WORD 1 IN A REGISTER
     004114  000002 1150 03                   685          CMPA     2,DU                CHECK STATUS RETURN WORD 1
     004115  004064 6030 00                   686          TRC      DDT                 IGNORE INPUT ON ANY ERROR
     004116  004617 7000 00                   687          TSX0     BREM                REMOVE BREAKPOINTS
     004117  001136 2350 00                   688          LDA      INP                 GET POINTER TO START OF INPUT BUFFER
     004120  001135 7550 00                   689          STA      IN                  AND INITIALIZE INPUT TALLY WORD
     004121  000460 2350 00                   690          LDA      OUTP                GET POINTER TO START OF OUTPUT BUFFER
     004122  000457 7550 00                   691          STA      OUT                 AND INITIALIZE OUTPUT TALLY WORD
     004123  001137 2210 00                   692          LDX1     INB                 GET FIRST TWO CHARACTERS INPUT IN XR - 1
     004124  001523 7410 00                   693          STX1     CTABE               PAD END OF TABLE TO INSURE MATCH
     004125  001507 2220 03                   694          LDX2     CTAB,DU             GET ADDRESS OF START OF TABLE IN XR - 2
     004126  000300 5202 01                   695          RPT      0,1,TZE             SEARCH TABLE FOR COMMAND
     004127  000000 1010 12                   696          CMPX1    0,2                 COMPARE AGAINST FIRST TWO CHARACTERS IN XR - 1
     004130  777777 2350 12                   697          LDA      -1,2                GET WORD THAT MATCHED IN A REGISTER
     004131  000000 7100 05                   698          TRA      0,AL                AND TRANSFER TO APPROPRIATE ROUTINE
```

```
                                          699          TTL      DEBUG COMMANDS
      004132  004522 7000 00              700 PAT      TSX0     GARG                GET PATCH ADDRESS
      004133  004432 7100 00              701          TRA      ERR                 NO PATCH ADDRESS - ERROR
      004134  000022 7350 00              702          ALS      18                  MOVE TO ADDRESS PORTION OF WORD
      004135  000452 7550 00              703          STA      TEMP                STORE AS TALLY WORD
      004136  004522 7000 00              704 PAT1     TSX0     GARG                GET NEXT NUMBER TO PATCH
      004137  004435 7100 00              705          TRA      FIN                 NO MORE - EXIT
   END OF BINARY CARD 00000060
      004140  000452 7550 56              706          STA      TEMP,ID             STORE PATCH AND INCREMENT PATCH ADDRESS
      004141  004136 7100 00              707          TRA      PAT1                AND SEE IF THERE IS ANY MORE TO DO
      004142  004522 7000 00              708 DUM      TSX0     GARG                GET FIRST LOCATION TO DUMP
      004143  035236 2350 07              709          LDA      T$TABLE,DL          IF NO ARGUMENT - DUMP TABLE DESCRIPTOR
      004144  000022 7350 00              710          ALS      18                  MOVE TO ADDRESS POSITION OF WORD
      004145  000452 7550 00              711          STA      TEMP                STORE AS TALLY WORD
      004146  004522 7000 00              712          TSX0     GARG                GET NUMBER OF WORDS TO DUMP
      004147  000001 2350 07              713          LDA      1,DL                IF NO ARGUMENT GIVEN, ASSUME ONE
      004150  000006 7350 00              714          ALS      6                   POSITION COUNT TO TALLY FIELD
      004151  000452 2550 00              715          ORSA     TEMP                STORE IN TALLY WORD
      004152  004522 7000 00              716          TSX0     GARG                GET OFFSET OR BASE OF TABLE TO DUMP
      004153  777777 2350 07              717          LDA      -1,DL               IF NO ARGUMENT, OFFSET IS MINUS ONE
      004154  035214 2350 05              718          LDA      T$START+1,AL        GET BASE OF TABLE IN AU
      004155  000000 6350 01              719          EAA      0,AU                ZERO OUT AL
      004156  000453 7550 00              720          STA      TEMP1               AND SAVE
      004157  000452 0550 00              721          ASA      TEMP                AND ADD OFFSET TO ADDRESS TO DUMP
      004160  000452 2360 00              722 DUM1     LDQ      TEMP                GET ADDRESS OF NEXT WORD TO BE DUMPED
      004161  777700 3160 07              723          CANQ     =0777700,DL         SEE IF THERE IS ANYTHING ELSE TO DUMP
      004162  004435 6000 00              724          TZE      FIN                 NO - EXIT
      004163  004577 7000 00              725          TSX0     WRITE               WRITE CR/LF
      004164  001451 0003 40              726          TALLYB   CRLF,3              CR/LF
      004165  000453 1760 00              727          SBQ      TEMP1               MAKE ADDRESS RELATIVE TO BASE OR OFFSET
   END OF BINARY CARD 00000061
      004166  004567 7000 00              728          TSX0     ADDR                TYPE ADDRESS
      004167  000004 2270 03              729          LDX7     4,DU                INITIALIZE NUMBER OF WORDS PER LINE
      004170  000040 2360 07              730 DUM2     LDQ      =0040,DL            GET A SPACE
      004171  000457 7560 52              731          STQ      OUT,SC              STORE IN OUTPUT
      004172  000452 2360 56              732          LDQ      TEMP,ID             GET NEXT WORD TO DUMP
      004173  004565 7000 00              733          TSX0     WORD                TYPE IT OUT
      004174  000001 1670 03              734          SBX7     1,DU                DECREMENT NUMBER OF WORDS LEFT ON THIS LINE
      004175  004202 6000 00              735          TZE      DUM3                TRANSFER IF TIME FOR A NEW LINE
      004176  000452 2350 00              736          LDA      TEMP                GET TALLY WORD IN A REGISTER
      004177  777700 3150 07              737          CANA     =0777700,DL         HAS TALLY RUN OUT
      004200  004170 6010 00              738          TNZ      DUM2                TRANSFER IF MORE TO DO
      004201  004435 7100 00              739          TRA      FIN                 NO MORE - EXIT
      004202  000461 2350 03              740 DUM3     LDA      OUTB,DU             GET ADDRESS OF START OF BUFFER IN AU
      004203  000017 0750 07              741          ADA      15,DL               GET LENGTH OF DATA IN AL
      004204  003031 7000 00              742          TSX0     STTYW               WRITE LINE TO TELETYPE
      004205  000460 2350 00              743          LDA      OUTP                GET INITIAL OUTPUT POINTER
      004206  000457 7550 00              744          STA      OUT                 AND INITIALIZE OUTPUT POINTER
      004207  004160 7100 00              745          TRA      DUM1                TRANSFER TO PRINT NEXT LINE
      004210  000455 2350 00              746 RDUM     LDA      DTCB                GET ADDRESS OF TASK CONTROL BLOCK
```

```
              Z                                         COMMAND INTERPRETER

     004211  000007 2360 01          747          LDQ      7,AU              GET SAVED IC IN Q
     004212  000017 7560 01          748          STQ      15,AU             STORE IN TIMER REGISTER PART OF SAVED REGISTERS
     004213  000010 0750 03          749          ADA      8,DU              MAKE A REGISTER POINT TO SAVED REGISTERS
 END OF BINARY CARD 00000062
     004214  001000 0750 07          750          ADA      8*64,DL           SET TALLY FOR EIGHT WORD DUMP
     004215  000452 7550 00          751          STA      TEMP              STORE FOR DUMP ROUTINE
     004216  000453 4500 00          752          STZ      TEMP1             MAKE OFFSET ZERO
     004217  004160 7100 00          753          TRA      DUM1              AND DUMP REGISTERS
     004220  004522 7000 00          754 TYPE     TSX0     GARG              GET FIRST LOCATION TO TYPE
     004221  004432 7100 00          755          TRA      ERR               IF NO STARTING ADDRESS THEN ERROR
     004222  000022 7350 00          756          ALS      18                MOVE TO ADDRESS POSITION OF WORD
     004223  000452 7550 00          757          STA      TEMP              STORE AS TALLY WORD
     004224  004522 7000 00          758          TSX0     GARG              GET NUMBER OF WORDS TO TYPE
     004225  000001 2350 07          759          LDA      1,DL              IF NO ARGUMENT IS GIVEN, ASSUME ONE
     004226  000010 7350 00          760          ALS      8                 POSITION COUNT TO TALLY FIELD TIMES FOUR
     004227  000040 0750 07          761          ADA      =040,DL           SET 9-BIT CHARACTER FLAG IN TALLY WORD
     004230  000452 2550 00          762          ORSA     TEMP              STORE IN TALLY WORD
     004231  004522 7000 00          763          TSX0     GARG              GET OFFSET OR BASE OF TABLE TO TYPE
     004232  777777 2350 07          764          LDA      -1,DL             IF NO ARGUMENT, OFFSET IS MINUS ONE
     004233  035214 2350 05          765          LDA      T$START+1,AL      GET BASE OF TABLE IN AU
     004234  000000 6350 01          766          EAA      0,AU              ZERO OUT AL
     004235  000453 7550 00          767          STA      TEMP1             AND SAVE
     004236  000452 0550 00          768          ASA      TEMP              AND ADD OFFSET TO ADDRESS TO TYPE
     004237  000452 2360 00          769 TYPE1    LDQ      TEMP              GET TALLY OF NEXT WORD TO BE TYPED
     004240  777700 3160 07          770          CANQ     =0777700,DL       SEE IF THERE IS ANYTHING ELSE TO DUMP
     004241  004435 6000 00          771          TZE      FIN               NO - EXIT
 END OF BINARY CARD 00000063
     004242  004577 7000 00          772          TSX0     WRITE             WRITE CR/LF
     004243  001451 0003 40          773          TALLYB   CRLF,3            CR/LF
     004244  000453 1760 00          774          SBQ      TEMP1             MAKE ADDRESS RELATIVE TO BASE OR OFFSET
     004245  004567 7000 00          775          TSX0     ADDR              TYPE ADDRESS
     004246  000040 2270 03          776          LDX7     32,DU             INITIALIZE NUMBER OF CHARACTERS PER LINE
     004247  000040 2360 07          777          LDQ      =0040,DL          GET A SPACE AND THREE NULLS
     004250  000457 7560 56          778          STQ      OUT,ID            STORE IN OUTPUT
     004251  000452 2360 52          779 TYPE2    LDQ      TEMP,SC           GET NEXT CHARACTER TO TYPE
     004252  000457 7560 52          780          STQ      OUT,SC            AND STORE IN OUTPUT BUFFER
     004253  000001 1670 03          781          SBX7     1,DU              DECREMENT NUMBER OF CHARACTERS LEFT ON THIS LINE
     004254  004261 6000 00          782          TZE      TYPE3             TRANSFER IF TIME FOR A NEW LINE
     004255  000452 2350 00          783          LDA      TEMP              GET TALLY WORD IN A REGISTER
     004256  777700 3150 07          784          CANA     =0777700,DL       HAS TALLY RUN OUT
     004257  004251 6010 00          785          TNZ      TYPE2             TRANSFER IF MORE TO DO
     004260  004435 7100 00          786          TRA      FIN               NO MORE - EXIT
     004261  000461 2350 03          787 TYPE3    LDA      OUTB,DU           GET ADDRESS OF START OF BUFFER IN AU
     004262  000013 0750 07          788          ADA      11,DL             GET LENGTH OF DATA IN AL
     004263  003031 7000 00          789          TSX0     $TTYW             WRITE LINE TO TELETYPE
     004264  000460 2350 00          790          LDA      OUTP              GET INITIAL OUTPUT POINTER
     004265  000457 7550 00          791          STA      OUT               AND INITIALIZE OUTPUT POINTER
     004266  004237 7100 00          792          TRA      TYPE1
     004267  004577 7000 00          793 TIME     TSX0     WRITE             WRITE CRLF
 END OF BINARY CARD 00000064
```

```
            Z                               COMMAND INTERPRETER

      004270  001451 0003 40          794          TALLYB  CRLF,3
      004271  500012 0010 00          795          MME     M$JTIME             GET RUNNING TIME IN A REGISTER
      004272  004304 7550 00          796          STA     TIMET               AND SAVE
      004273  004306 7000 00          797          TSX0    BCD                 TYPE OUT RUNNING TIME
      004274  000040 2350 07          798          LDA     =0040,DL            GET A SPACE
      004275  000457 7550 52          799          STA     OUT,SC              AND OUTPUT IT
      004276  004304 2350 00          800          LDA     TIMET               GET CURRENT RUNNING TIME
      004277  004305 1750 00          801          SBA     LTIME               SUBTRACT LAST TIME TIME WAS PRINTED
      004300  004306 7000 00          802          TSX0    BCD                 AND PRINT TIME INCREASE
      004301  004304 2350 00          803          LDA     TIMET               GET CURRENT TIME
      004302  004305 7550 00          804          STA     LTIME               AND MAKE IT LAST TIME
      004303  004435 7100 00          805          TRA     FIN                 AND EXIT
      004304  000000 000000           806 TIMET    ZERO
      004305  000000 000000           807 LTIME    ZERO
      004306  000000 2360 03          808 BCD      LDQ     0,DU                ZERO OUT Q REGISTER
      004307  004334 5070 00          809          DVF     BCDC                DIVIDE BY 1000 SECONDS
      004310  004322 7010 00          810          TSX1    BCD1                PRINT A DIGIT
      004311  004322 7010 00          811          TSX1    BCD1                PRINT A DIGIT
      004312  004322 7010 00          812          TSX1    BCD1                PRINT A DIGIT
      004313  000056 2360 07          813          LDQ     =0056,DL            GET A DECIMAL POINT
      004314  000457 7560 52          814          STQ     OUT,SC              AND OUTPUT IT
      004315  004322 7010 00          815          TSX1    BCD1
   END OF BINARY CARD 00000065
      004316  004322 7010 00          816          TSX1    BCD1                PRINT A DIGIT
      004317  004322 7010 00          817          TSX1    BCD1                PRINT A DIGIT
      004320  004322 7010 00          818          TSX1    BCD1                PRINT A DIGIT
      004321  000000 7100 10          819          TRA     0,0                 AND RETURN
      004322  240000 4010 03          820 BCD1     MPF     =0240000,DU         MULTIPLY BY 10/16
      004323  004333 7550 00          821          STA     BCDT                SAVE INTEGER PART
      004324  000005 7370 00          822          LLS     5                   DELETE INTEGER PART
      004325  000001 7710 00          823          ARL     1                   AND RESTORE FRACTION IN A
      004326  004333 2360 00          824          LDQ     BCDT                GET INTEGER PART IN Q
      004327  000037 7720 00          825          QRL     36-5                GET INTEGER DIGIT IN QL
      004330  000060 0760 07          826          ADQ     =0060,DL            ADD AN ASCII ZERO
      004331  000457 7560 52          827          STQ     OUT,SC              AND OUTPUT DIGIT
      004332  000000 7100 11          828          TRA     0,1                 RETURN
      004333  000000 000000           829 BCDT     ZERO
      004334  000364110000            830 BCDC     DEC     64000000
      004335  000456 2340 00          831 TRA      SZN     RETRN               IS THE PROGRAM CURRENTLY RUNNING
      004336  004432 6010 00          832          TNZ     ERR                 YES - CANNOT TRANSFER IN A RUNNING PROGRAM
      004337  004522 7000 00          833          TSX0    GARG                GET ADDRESS OF TRANSFER
      004340  004432 7100 00          834          TRA     ERR                 NO ADDRESS GIVEN - ERROR
      004341  000000 6200 05          835          EAX0    0,AL                PUT ADDRESS IN XR - 0
      004342  000455 2210 00          836          LDX1    DTCB                GET POINTER TO TASK CONTROL BLOCK IN XR - 1
      004343  000007 7400 11          837          STX0    7,1                 STORE ADDRESS IN SAVED INSTRUCTION COUNTER
   END OF BINARY CARD 00000066
      004344  004345 7100 00          838          TRA     CON                 AND RESTART JOB
      004345  000456 2340 00          839 CON      SZN     RETRN               IS THE PROGRAM RUNNING
      004346  004432 6010 00          840          TNZ     ERR                 YES - CANNOT CONTINUE A RUNNING PROGRAM
      004347  000456 7500 00          841          STC2    RETRN               SET RETURN FLAG
      004350  004435 7100 00        842          TRA      FIN           AND EXIT
      004351  004522 7000 00        843 BRE      TSX0     GARG          GET BREAKPOINT ADDRESS
      004352  004372 7100 00        844          TRA      BRE4          UNCONDITIONAL BREAK IF NO ARGUMENT
      004353  000000 6270 05        845 BRE1     EAX7     0,AL          PUT ADDRESS IN XR - 7 FOR TABLE LOOK UP
      004354  004626 7000 00        846          TSX0     BTLU          SEARCH BREAKPOINT TABLE
      004355  004363 7100 00        847          TRA      BRE2          TRANSFER IF NOT FOUND
      004356  004577 7000 00        848          TSX0     WRITE         WRITE MESSAGE ABOUT DUPLICATE BREAKPOINT FOUND
      004357  001453 0027 40        849          TALLYB   BPIT,23       CR/LF/BREAKPOINT IN TABLE
      004360  000000 6360 17        850          EAQ      0,7           PUT ADDRESS OF BREAKPOINT IN Q
      004361  004567 7000 00        851          TSX0     ADDR          ALSO TYPE ADDRESS
      004362  004367 7100 00        852          TRA      BRE3          AND GO GET NEXT BREAK ADDRESS IF ANY
      004363  001303 7470 11        853 BRE2     STX7     BTAB,1        STORE ADDRESS OF NEW BREAKPOINT
      004364  000001 0610 03        854          ADX1     1,DU          STEP TO NEXT TABLE ENTRY
      004365  000001 3350 07        855          LCA      1,DL          GET A NEGATIVE NUMBER
      004366  001303 7550 11        856          STA      BTAB,1        STORE END OF TABLE MARKER
      004367  004522 7000 00        857 BRE3     TSX0     GARG          GET NEXT BREAK ADDRESS
      004370  004435 7100 00        858          TRA      FIN           NO MORE SO EXIT
      004371  004353 7100 00        859          TRA      BRE1          GO SERVICE NEW BREAK ADDRESS
  END OF BINARY CARD 00000067
      004372  000456 2340 00        860 BRE4     SZN      RETRN         IS TASK HALTED
      004373  004432 6000 00        861          TZE      ERR           YES - CANNOT HALT IT AGAIN
      004374  001135 7560 50        862          STQ      IN,CI         SAVE END OF ARGUMENT FOR NEXT CALL
      004375  000455 2200 00        863          LDX0     DTCB          GET ADDRESS OF TASK CONTROL BLOCK
      004376  000007 2270 10        864          LDX7     7,0           GET ADDRESS OF NEXT INSTRUCTION TO EXECUTE
      004377  004353 7100 00        865          TRA      BRE1          AND INSERT BREAKPOINT THERE
      004400  004522 7000 00        866 UNB      TSX0     GARG          GET ADDRESS OF BREAKPOINT TO REMOVE
      004401  004425 7100 00        867          TRA      UNB4          NO ARGUMENT SO REMOVE ALL OF THEM
      004402  000000 6270 05        868 UNB1     EAX7     0,AL          GET BREAKPOINT ADDRESS IN XR - 7
      004403  004626 7000 00        869          TSX0     BTLU          FIND BREAKPOINT IN TABLE
      004404  004420 7100 00        870          TRA      UNB3          TRANSFER IF NOT IN BREAKPOINT TABLE
      004405  000000 6220 11        871          EAX2     0,1           SAVE LOCATION OF BREAKPOINT IN TABLE
      004406  000001 0610 03        872          ADX1     1,DU          STEP OVER ENTRY
      004407  004627 7000 00        873          TSX0     BTLU1         AND SEARCH FOR END OF TABLE
      004410  000001 1610 03        874          SBX1     1,DU          GET POINTER TO LAST ENTRY IN TABLE
      004411  001303 2350 11        875          LDA      BTAB,1        FETCH LAST ENTRY IN BREAKPOINT TABLE
      004412  001303 7550 12        876          STA      BTAB,2        STORE ON TOP OF ELEMENT TO BE REMOVED
      004413  000001 3350 07        877          LCA      1,DL          GET END OF TABLE FLAG
      004414  001303 7550 11        878          STA      BTAB,1        AND STORE AT NEW END OF BREAKPOINT TABLE
      004415  004522 7000 00        879 UNB2     TSX0     GARG          GET NEXT ARGUMENT
      004416  004435 7100 00        880          TRA      FIN           NO MORE - EXIT
      004417  004402 7100 00        881          TRA      UNB1          MORE TO DO SO LOOP
  END OF BINARY CARD 00000068
      004420  004577 7000 00        882 UNB3     TSX0     WRITE         TYPE OUT ERROR MESSAGE
      004421  001466 0030 40        883          TALLYB   BNF,24        CR/LF/BREAKPOINT NOT FOUND
      004422  000000 6360 17        884          EAQ      0,7           GET ATTEMPTED ADDRESS IN Q REGISTER
      004423  004567 7000 00        885          TSX0     ADDR          AND TYPE IT
      004424  004415 7100 00        886          TRA      UNB2          AND SEE IF THERE IS ANY MORE TO DO
      004425  000001 3350 07        887 UNB4     LCA      1,DL          GET AN END OF TABLE FLAG
      004426  001303 7550 00        888          STA      BTAB          AND STORE IT AT THE BEGINNING OF THE TABLE
      004427  004435 7100 00        889          TRA      FIN           AND EXIT
```

```
              Z                                  COMMAND INTERPRETER

     004430  000000 2200 03          890 EXIT    LDX0     0,DU              GET SAFE TERMINATE STATUS
     004431  500000 0010 00          891         MME      M$TERM            TERMINATE EXECUTION
     004432  004577 7000 00          892 ERR     TSX0     WRITE             TYPE OUT ERROR
     004433  001475 0010 40          893         TALLYB   ERRM,8            CR/LF/ERROR
     004434  004435 7100 00          894         TRA      FIN               AND EXIT
     004435  004606 7000 00          895 FIN     TSX0     BSET              INSERT BREAKPOINTS IN PROGRAM
     004436  004577 7000 00          896         TSX0     WRITE             WRITE TO TTY
     004437  001451 0003 40          897         TALLYB   CRLF,3            ADD CARRIAGE RETURN/LINE FEED TO OUTPUT BUFFER
     004440  000003 2360 07          898         LDQ      3,DL              GET A CHARACTER POSITION MASK IN Q
     004441  000000 2350 07          899         LDA      0,DL              GET A NULL CHARACTER IN AL
     004442  000457 3160 00          900 FIN1    CANQ     OUT               SEE IF OUTPUT POINTER IS AT CHARACTER ONE OF WORD
     004443  004446 6000 00          901         TZE      FIN2              TRANSFER IF YES
     004444  000457 7550 52          902         STA      OUT,SC            ADD A NULL CHARACTER TO THE OUTPUT BUFFER
     004445  004442 7100 00          903         TRA      FIN1              AND TRY AGAIN
 END OF BINARY CARD 00000069
     004446  000457 2350 00          904 FIN2    LDA      OUT               GET OUTPUT POINTER IN A REGISTER
     004447  000461 1750 03          905         SBA      OUTB,DU           SUBTRACT LOCATION OF OUTPUT BUFFER
     004450  000022 7710 00          906         ARL      18                RIGHT JUSTIFY NUMBER OF WORDS IN OUTPUT BUFFER
     004451  000461 0750 03          907         ADA      OUTB,DU           ADD BASE OF OUTPUT BUFFER - GET I/O CONTROL WORD
     004452  003031 7000 00          908         TSX0     $TTYW             DO TTY OUTPUT
     004453  000456 2340 00          909         SZN      RETRN             IS THE RETURN FLAG SET
     004454  004064 6000 00          910         TZE      DDT               NO - WAIT FOR ANOTHER SPECIAL
     004455  000456 4500 00          911         STZ      RETRN             RESET RETURN FLAG
     004456  000455 2200 00          912         LDX0     DTCB              GET POINTER TO TASK CONTROL BLOCK IN XR - 0
     004457  000007 2270 10          913         LDX7     7,0               GET DESTINATION ADDRESS IN XR - 7
     004460  004626 7000 00          914         TSX0     BTLU              AND SEE IF IT IS A BREAKPOINT
     004461  004500 7100 00          915         TRA      FIN3              NO - CAN RETURN NORMALLY
     004462  001303 7230 11          916         LXL3     BTAB,1            GET LOWER HALF OF INSTRUCTION AT BREAKPOINT
     004463  004521 4430 00          917         SXL3     BINST             STORE IN SPECIAL INSTRUCTION WORD
     004464  000000 2230 17          918         LDX3     0,7               GET UPPER HALF OF INSTRUCTION AT BREAKPOINT
     004465  004521 7430 00          919         STX3     BINST             STORE IN SPECIAL INSTRUCTION WORD
     004466  000000 2360 17          920         LDQ      0,7               GET ACTUAL BREAKPOINT IN Q REGISTER
     004467  000452 7560 00          921         STQ      TEMP              AND SAVE IN MEMORY
     004470  001447 2360 00          922         LDQ      BXED              GET AN XED BKST TO RESTART PROPERLY
     004471  000000 7560 17          923         STQ      0,7               STORE AT BREAKPOINT LOCATION
     004472  000455 2200 00          924         LDX0     DTCB              GET POINTER TO TASK CONTROL BLOCK IN XR - 0
     004473  000007 7470 10          925         STX7     7,0               STORE STARTING ADDRESS IN SAVED IC
 END OF BINARY CARD 00000070
     004474  000014 2350 10          926         LDA      12,0              FETCH SAVED A REGISTER
     004475  000453 7550 00          927         STA      TEMP1             SAVE FOR LATER RESTORATION
     004476  000452 2350 00          928         LDA      TEMP              GET IMAGE OF BREAKPOINT
     004477  000014 7550 10          929         STA      12,0              STORE IN SAVED A REGISTER
                        004500       930 FIN3    QUEUE    (DTCB,I),,$TASK   RESTART PROGRAM
     004512  004064 7100 00          931         TRA      DDT               AND LET DEBUGGER WAIT FOR ANOTHER SPECIAL
     004513  000000011007
                        004514       932         EVEN
     004514  000000 7550 04          933 BKST    STA      0,IC              RESTORE BREAKPOINT
     004515  004516 7170 00          934         XED      *+1               PRESERVE IC
     004516  000453 2350 00          935         LDA      TEMP1             RESTORE A REGISTER
     004517  004520 7170 00          936         XED      *+1               PRESERVE IC
```

```
   004520  000221 6340 51       937          LDI      ,$IC,I           RESTORE INDICATORS
   004521  000000 0000 00       938 BINST    ARG      **               EXECUTE INSTRUCTION PREVIOUSLY AT BREAKPOINT
```

```
                                   939          TTLS    SUBROUTINES
  END OF BINARY CARD 00000071
       004522  004564 4500 00      940 GARG     STZ     FG                  RESET NEGATIVE FLAG
       004523  001135 2360 52      941          LDQ     IN,SC               GET NEXT INPUT CHARACTER
       004524  000015 1160 07      942          CMPQ    =0015,DL            IS IT A CARRIAGE RETURN
       004525  004562 6000 00      943          TZE     GARG3               YES - GIVE NO ARGUMENT RETURN
       004526  000055 1160 07      944          CMPQ    =0055,DL            IS IT A MINUS SIGN
       004527  004533 6010 00      945          TNZ     GARG4               NO - SKIP SETTING NEGATIVE FLAG
       004530  004564 0540 00      946          AOS     FG                  SET NEGATIVE FLAG
       004531  000000 2360 07      947          LDQ     0,DL                MAKE FIRST DIGIT LOOK LIKE A ZERO
       004532  004541 7100 00      948          TRA     GARG5               AND GO TO CONVERT FOLLOWING NUMBER
       004533  000060 1760 07      949 GARG4    SBQ     =0060,DL            SUBTRACT DIGIT ZERO
       004534  004522 6040 00      950          TMI     GARG                NOT A DIGIT SO GET NEXT CHARACTER
       004535  000012 1160 07      951          CMPQ    10,DL               IS IT A NON-DIGIT
       004536  004522 6050 00      952          TPL     GARG                YES - SKIP
       004537  000010 1160 07      953          CMPQ    8,DL                IS IT AN OCTAL DIGIT
       004540  004432 6050 00      954          TPL     ERR                 NO - ERROR
       004541  000000 2350 07      955 GARG5    LDA     0,DL                CLEAR A REGISTER
       004542  000041 7360 00      956 GARG1    QLS     33                  MOVE OCTAL DIGIT TO LEFT OF Q REGISTER
       004543  000003 7370 00      957          LLS     3                   ADD DIGIT TO NUMBER IN A REGISTER
       004544  001135 2360 52      958          LDQ     IN,SC               GET NEXT INPUT CHARACTER
       004545  000060 1760 07      959          SBQ     =0060,DL            SUBTRACT DIGIT ZERO
       004546  004552 6040 00      960          TMI     GARG2               TRANSFER IF SEPARATOR
       004547  000010 1160 07      961          CMPQ    8,DL                IS IT AN OCTAL DIGIT
  END OF BINARY CARD 00000072
       004550  004542 6040 00      962          TMI     GARG1               YES - ACCUMULATE DIGIT
       004551  004432 7100 00      963          TRA     ERR                 NO - ERROR
       004552  000060 0760 07      964 GARG2    ADQ     =0060,DL            RESTORE CHARACTER
       004553  000015 1160 07      965          CMPQ    =0015,DL            IS IT A CARRIAGE RETURN
       004554  004556 6010 00      966          TNZ     GARG6               NO - GO CHECK FOR NEGATIVE NUMBER
       004555  001135 7560 50      967          STQ     IN,CI               STORE POSSIBLE CARRIAGE RETURN FOR NEXT TIME
       004556  004564 2340 00      968 GARG6    SZN     FG                  SEE IF NUMBER HAD A MINUS SIGN IN FRONT OF IT
       004557  000001 6000 10      969          TZE     1,0                 RETURN IF NOT
       004560  000000 5310 00      970          NEG                         NEGATE NUMBER
       004561  000001 7100 10      971          TRA     1,0                 AND RETURN
       004562  001135 7560 50      972 GARG3    STQ     IN,CI               STORE CR IN INPUT LINE FOR NEXT CALL TO ROUTINE
       004563  000000 7100 10      973          TRA     0,0                 NO ARGUMENT EXIT
       004564  000000 000000       974 FG       ZERO
       004565  777764 2210 03      975 WORD     LDX1    -12,DU              GET NUMBER OF DIGITS IN A WORD IN XR - 1
       004566  004570 7100 00      976          TRA     LOOP                AND PRINT WORD
       004567  777772 2210 03      977 ADDR     LDX1    -6,DU               GET NUMBER OF DIGITS IN AN ADDRESS
       004570  000000 2350 07      978 LOOP     LDA     0,DL                CLEAR A REGISTER
       004571  000003 7370 00      979          LLS     3                   SHIFT IN FIRST OCTAL DIGIT FROM Q TO A REGISTER
       004572  000060 0750 07      980          ADA     =0060,DL            GET IN A ASCII OCTAL DIGIT
       004573  000457 7550 52      981          STA     OUT,SC              STORE IN OUTPUT BUFFER
       004574  000001 0610 03      982          ADX1    1,DU                DECREMENT CHARACTER COUNTER
       004575  004570 6040 00      983          TMI     LOOP                TRANSFER IF MORE TO DO
  END OF BINARY CARD 00000073
       004576  000000 7100 10      984          TRA     0,0                 NO MORE SO RETURN
       004577  000000 2350 10      985 WRITE    LDA     0,0                 GET TALLY WORD SUPPLIED
```

```
          2                             SUBROUTINES

    004600   000454 7550 00         986          STA      WTEMP           STORE IN MEMORY
    004601   000454 2350 52         987 WRIT1    LDA      WTEMP,SC        GET NEXT CHARACTER TO OUTPUT
    004602   004604 6070 00         988          TTF      *+2             TRANSFER IF THERE IS ONE
    004603   000001 7100 10         989          TRA      1,0             NO MORE - RETURN
    004604   000457 7550 52         990          STA      OUT,SC          STORE IN OUTPUT BUFFER
    004605   004601 7100 00         991          TRA      WRIT1           AND LOOP
    004606   000000 2210 03         992 BSET     LDX1     0,DU            GET RELATIVE POINTER TO START OF BREAKPOINT TABLE
    004607   002000 2230 03         993          LDX3     2*512,DU        GET A DRL INSTRUCTION IN XR - 3
    004610   001303 2350 11         994 BSET1    LDA      BTAB,1          GET NEXT ENTRY IN BREAKPOINT TABLE
    004611   000000 6040 10         995          TMI      0,0             TRANSFER IF ALL DONE
    004612   000000 7220 01         996          LXL2     0,AU            GET LOW HALF OF WORD WHERE BREAKPOINT IS GOING
    004613   001303 4420 11         997          SXL2     BTAB,1          SAVE IN BREAKPOINT TABLE ENTRY
    004614   000000 4430 01         998          SXL3     0,AU            STORE BREAKPOINT IN PROGRAM
    004615   000001 0610 03         999          ADX1     1,DU            STEP TO NEXT ENTRY IN BREAKPOINT TABLE
    004616   004610 7100 00        1000          TRA      BSET1           AND LOOP
    004617   000000 2210 03        1001 BREM     LDX1     0,DU            GET RELATIVE POINTER TO START OF BREAKPOINT TABLE
    004620   001303 2350 11        1002 BREM1    LDA      BTAB,1          GET NEXT ENTRY IN BREAKPOINT TABLE
    004621   000000 6040 10        1003          TMI      0,0             ALL DONE SO EXIT
    004622   001303 7220 11        1004          LXL2     BTAB,1          GET SAVED LOWER HALF OF BREAKPOINT LOCATION
    004623   000000 4420 01        1005          SXL2     0,AU            RESTORE LOW HALF IN PROGRAM
 END OF BINARY CARD 00000074
    004624   000001 0610 03        1006          ADX1     1,DU            STEP TO NEXT ENTRY IN BREAKPOINT TABLE
    004625   004620 7100 00        1007          TRA      BREM1           AND LOOP
    004626   000000 2210 03        1008 BTLU     LDX1     0,DU            GET POINTER TO START OF TABLE
    004627   001303 2340 11        1009 BTLU1    SZN      BTAB,1          IS THIS THE END OF THE TABLE
    004630   000000 6040 10        1010          TMI      0,0             YES - NOT FOUND
    004631   001303 1070 11        1011          CMPX7    BTAB,1          IS THIS THE DESIRED ENTRY
    004632   000001 6000 10        1012          TZE      1,0             YES - FOUND
    004633   000001 0610 03        1013          ADX1     1,DU            INCREMENT POINTER
    004634   004627 7100 00        1014          TRA      BTLU1           AND LOOP
                                   1015          TTL      ALGOL68
```

```
                                     1016          TTLS     PASS0
                                     1017          HEAD     A
                                     1018          INHIB    OFF
     004635   001675 2360 52         1019 NCHAR    LDQ      IN,SC             GET NEXT CHARACTER IN Q IF ANY
     004636   004662 6070 00         1020          TTF      NCH1              TRANSFER IF THERE WAS ONE
     004637   004641 7400 00         1021          STX0     NCHX              SAVE RETURN ADDRESS
     004640   001677 2340 00         1022          SZN      EOF               IS IT THE END OF THE FILE
     004641   000000 6010 00         1023 NCHX     TNZ      **                TRANSFER IF END OF FILE
     004642   001673 2350 00         1024          LDA      INBUF+BLEN-1      GET LAST WORD IN BUFFER
     004643   001527 7550 00         1025          STA      INBUF-1           STORE IN FRONT OF BUFFER IN CASE OF BACKUP
     004644   001524 2370 00         1026          LDAQ     CPRAM             GET PARAMETERS FOR READ INTO BUFFER FROM SOURCE
     004645   003721 7000 00         1027          TSX0     $READ             READ INTO BUFFER
     004646   004641 2200 00         1028          LDX0     NCHX              RESTORE RETURN LOCATION
     004647   001676 2350 00         1029          LDA      INP               GET TALLY WORD FOR NCHAR ROUTINE
     004650   001675 7550 00         1030          STA      IN                INITIALIZE INPUT TALLY WORD TO START OF BUFFER
     004651   000222 2370 51         1031          LDAQ     ,$STAT,I          GET STATUS OF OPERATION
 END OF BINARY CARD 00000075
     004652   000002 1150 03         1032          CMPA     2,DU              IS STATUS OK
     004653   777777 6030 00         1033          TRC      ERROR             NO - ERROR
     004654   000001 1150 03         1034          CMPA     1,DU              SEE IF STATUS IS SHORT READ
     004655   004635 6020 00         1035          TNC      NCHAR             NO - CONTINUE
     004656   001677 7500 00         1036          STC2     EOF               YES - SET END OF FILE FLAG
     004657   000010 7360 00         1037          QLS      6+2               MULTIPLY RESIDUE BY 4 AND POSITION TO TALLY FIELD
     004660   001675 0560 00         1038          ASQ      IN                MAKE TALLY RUN OUT ON LAST READ CHARACTER
     004661   004635 7100 00         1039          TRA      NCHAR             AND CONTINUE
     004662   004635 6000 00         1040 NCH1     TZE      NCHAR             TRANSFER IF NULL AND GET NEXT CHARACTER
     004663   000177 1160 07         1041          CMPQ     =0177,DL          IS IT A FILL CHARACTER
     004664   000001 6010 10         1042          TNZ      1,0               NO - SO RETURN
     004665   004635 7100 00         1043          TRA      NCHAR             YES - GET NEXT CHARACTER
     004666   004773 7400 00         1044 NID      STX0     NIEOF             SAVE RETURN ADDRESS
     004667   001750 2350 00         1045          LDA      NITAL             GET INITIAL IDENTIFIER TALLY WORD
     004670   001751 7550 00         1046          STA      IDENT             STORE IN IDENT TALLY WORD
     004671   001752 2210 03         1047          LDX1     NICHR,DU          GET POINTER TO SAVED CHARACTER
     004672   001764 2220 03         1048 NISK     LDX2     NIT1,DU           GET POINTER TO TABLE 1
     004673   004676 7100 00         1049          TRA      NILP1             AND ENTER LOOP
     004674   001753 2360 50         1050 NILP     LDQ      NICH,CI           GET LAST CHARACTER SUPPLIED BY READ ROUTINE
     004675   001751 7560 52         1051          STQ      IDENT,SC          AND STORE IN IDENTIFIER
     004676   004635 7000 00         1052 NILP1    TSX0     NCHAR             GET NEXT CHARACTER FROM INPUT FILE IN QL
     004677   001000 2360 07         1053          LDQ      =01000,DL         IF END OF FILE - GET PSEUDO CHARACTER 1000
 END OF BINARY CARD 00000076
     004700   000022 7360 00         1054          QLS      18                SHIFT CHARACTER TO QU
     004701   001752 7560 00         1055          STQ      NICHR             AND STORE FOR COMPARISON ROUTINE
     004702   000000011007
     004703   001300 5602 01         1056          RPDA     ,1,TZE            SEARCH CURRENT TABLE FOR THIS TYPE OF CHARACTER
     004704   000000 2370 12         1057          LDAQ     0,2               GET RANGE TO CHECK FROM TABLE
     004705   000000 1110 11         1058          CWL      0,1               SEE IF CHARACTER IS WITHIN RANGE
     004706   000000 6000 06         1059          TZE      0,QL              TRANSFER TO APPROPRIATE ROUTINE
     004707   777777 7100 00         1060          TRA      ERROR             ILLEGAL CHARACTER
     004710   002022 2220 03         1061 NIDEN    LDX2     NIT2,DU           GET POINTER TO BUILD IDENTIFIER TABLE
     004711   004674 7100 00         1062          TRA      NILP              STORE CURRENT CHARACTER AND CONTINUE
```

```
              A                                    PASS0

       004712  001751 0110 52        1063 NIDG1  NOP     IDENT,SC          SKIP OVER PLUS OR MINUS ALREADY STORED
       004713  002032 2220 03        1064 NIDIG  LDX2    NIT3,DU           GET POINTER TO BUILD NUMBER TABLE
       004714  004674 7100 00        1065        TRA     NILP              STORE CURRENT CHARACTER AND CONTINUE
       004715  002044 2220 03        1066 NIDEC  LDX2    NIT4,DU           GET POINTER TO BUILD FRACTION TABLE
       004716  004676 7100 00        1067        TRA     NILP1             DO NOT STORE DECIMAL POINT AND CONTINUE
       004717  000056 2360 07        1068 NIDC1  LDQ     =0056,DL          GET A PERIOD
       004720  001751 7560 52        1069        STQ     IDENT,SC          STORE IN IDENT BEING BUILT
       004721  002032 2220 03        1070        LDX2    NIT3,DU           GET POINTER TO BUILD NUMBER TABLE
       004722  004674 7100 00        1071        TRA     NILP              STORE CURRENT CHARACTER AND CONTINUE
       004723  002052 2220 03        1072 NIEXP  LDX2    NIT5,DU           GET POINTER TO BUILD EXPONENT TABLE
       004724  004674 7100 00        1073        TRA     NILP              STORE CURRENT CHARACTER AND CONTINUE
       004725  001753 2360 50        1074 NIEX1  LDQ     NICH,CI           GET CURRENT CHARACTER (+ OR -)
   END OF BINARY CARD 00000077
       004726  001751 7560 50        1075        STQ     IDENT,CI          STORE IN NUMBER BEING BUILT IF NEEDED
       004727  002064 2220 03        1076        LDX2    NIT6,DU           GET POINTER TO EXPONENT CHECK TABLE
       004730  004676 7100 00        1077        TRA     NILP1             CONTINUE
       004731  001762 2220 03        1078 NICR   LDX2    NIT7,DU           GET POINTER TO LF AFTER CR CHECK TABLE
       004732  004676 7100 00        1079        TRA     NILP1             AND CONTINUE
       004733  001756 2220 03        1080 NILN   LDX2    NIT8,DU           GET POINTER TO LINE NUMBER SKIP TABLE
       004734  004676 7100 00        1081        TRA     NILP1             AND CONTINUE
       004735  002072 2220 03        1082 NIDOT  LDX2    NIT9,DU           GET POINTER TO PERIOD CHECK TABLE

       004736  004674 7100 00        1083        TRA     NILP              STORE CURRENT CHARACTER AND CONTINUE
       004737  002100 2220 03        1084 NICOM  LDX2    NIT10,DU          GET POINTER TO COMMENT SKIP TABLE
       004740  004676 7100 00        1085        TRA     NILP1             AND CONTINUE
       004741  002106 2220 03        1086 NIOPR  LDX2    NIT11,DU          GET POINTER TO OPERATOR TABLE
       004742  004674 7100 00        1087        TRA     NILP              STORE CURRENT CHARACTER AND CONTINUE
       004743  002120 2220 03        1088 NIQU   LDX2    NIT12,DU          GET POINTER TO STRING TABLE
       004744  004674 7100 00        1089        TRA     NILP              STORE OPENING QUOTE AND CONTINUE
       004745  002130 2220 03        1090 NIQU1  LDX2    NIT13,DU          GET POINTER TO STRING END CHECK TABLE
       004746  004676 7100 00        1091        TRA     NILP1             AND CONTINUE
       004747  002136 2220 03        1092 NIFOR  LDX2    NIT14,DU          GET POINTER TO FORMAT TABLE
       004750  004674 7100 00        1093        TRA     NILP              STORE CURRENT CHARACTER AND CONTINUE
       004751  001754 2360 00        1094 NIB2X  LDQ     BACK4             PREPARE TO BACK UP FOUR CHARACTER POSITIONS
       004752  001675 0560 00        1095        ASQ     IN                BACK UP INPUT ROUTINE TALLY WORD 4 CHARACTERS
       004753  004757 7100 00        1096        TRA     NIS2              AND SKIP FORWARD TWO CHARACTERS
   END OF BINARY CARD 00000078
       004754  001754 2360 00        1097 NIBEX  LDQ     BACK4             PREPARE TO BACK UP FOUR CHARACTER POSITIONS
       004755  001675 0560 00        1098        ASQ     IN                BACKUP INPUT ROUTINE TALLY WORD 4 CHARACTERS
       004756  001675 0110 52        1099        NOP     IN,SC             SKIP FOREWARD ONE CHARACTER POSITION
       004757  001675 0110 52        1100 NIS2   NOP     IN,SC             SKIP FOREWARD ONE CHARACTER POSITION
       004760  001675 0110 52        1101        NOP     IN,SC             SKIP FOREWARD ONE CHARACTER POSITION
       004761  004764 7100 00        1102        TRA     NIX               AND EXIT
       004762  001753 2360 50        1103 NIKIT  LDQ     NICH,CI           GET LAST CHARACTER CONSIDERED
       004763  001751 7560 52        1104        STQ     IDENT,SC          AND STORE IN IDENT
       004764  001750 2360 00        1105 NIX    LDQ     NITAL             GET INITIAL TALLY IN Q
       004765  001751 2350 00        1106        LDA     IDENT             GET FINAL IDENT TALLY WORD
       004766  000000 5310 00        1107        NEG                       NEGATE TALLY COUNT FOR CORRECT OUTPUT TALLY COUNT
       004767  001751 7560 00        1108        STQ     IDENT             STORE TALLY POINTER TO FIRST CHARACTER OF IDENT
       004770  001751 7510 06        1109        STCA    IDENT,06          STORE IN CORRECT TALLY COUNT
       004771  004773 2200 00        1110        LDX0    NIEOF             GET RETURN ADDRESS IN XR - 0
```

```
      004772  000001 7100 $0               TRA     1,0              AND GIVE SUCCESSFUL RETURN
      004773  000000 7100 00          NIEOF TRA     **               EOF RETURN AND SAVED RETURN ADDRESS
      004774  004777 7400 00          NS    STX0    NSX              SAVE RETURN ADDRESS
      004775  002162 4500 00                STZ     LNGCT            INITIALIZE LONG COUNT
      004776  004666 7000 00          NSR   TSX0    NID              GET NEXT IDENTIFIER OR SYMBOL
      004777  000000 7100 00          NSX   TRA     **               GIVE END OF FILE RETURN
      005000  035222 2270 00                LDX7    T$STAB           GET ADDRESS OF BASE OF SYMBOL TABLE IN XR - 7
      005001  000000 2350 $7          NSL   LDA     SC,7             GET SC POINTER TO SYMBOL
   END OF BINARY CARD 00000079
      005002  005034 6010 00                TNZ     NS3              TRANSFER IF NOT AT THE END OF THE TABLE
      005003  000001 2350 03                LDA     ELEN-1,DU        GET ELEMENT LENGTH MINUS ONE
      005004  035222 2210 03                LDX1    T$STAB,DU        GET POINTER TO SYMBOL TABLE CONTROL WORD
      005005  005663 7000 00                TSX0    T$ALOC           ALLOCATE SPACE IN SYMBOL TABLE
      005006  777777 6270 $1                EAX7    -1,1             GET IN XR - 7 POINTER TO NEXT ENTRY IN TABLE
      005007  001751 2350 00                LDA     IDENT            GET SC POINTER TO CURRENT SYMBOL
      005010  777700 3750 07                ANA     =0777700,DL      EXTRACT CHARACTER COUNT
      005011  000000 7550 $7                STA     SC,7             STORE IN NEW SC POINTER IN SYMBOL TABLE
      005012  002151 2350 00                LDA     SYMSC            GET CURRENT END OF SYMBOL STRING POINTER
      005013  002150 3750 00                ANA     NTMSK            ZERO OUT CHARACTER COUNT FIELD
      005014  000000 2550 $7                ORSA    SC,7             OR TO MEMORY TO FORM NEW SYMBOL TABLE ENTRY
      005015  000001 4500 $7                STZ     DF,7             CREAR OUT POINTER WORD OF TABLE ENTRY
      005016  400000 6200 $7                EAX0    131072,7         GET POINTER TO THIS ENTRY IN XR - 0
      005017  035222 1600 00                SBX0    T$STAB           MAKE POINTER RELATIVE
      005020  000001 7400 $7                STX0    DF,7             AND STORE AS FINAL LINK FOR THIS ENTRY
      005021  035223 2200 00                LDX0    T$ITAB           GET LOCATION OF IDENTIFIER TABLE
      005022  002151 0400 00                ASX0    SYMSC            MAKE IDENTIFIER TABLE SC POINTER ABSOLUTE
      005023  001751 2350 52          NS1   LDA     IDENT,SC         GET CHARACTER OF SYMBOL
      005024  005030 6070 00                TTF     NS2              TRANSFER IF NOT ALL DONE
      005025  035223 3200 00                LCX0    T$ITAB           GET MINUS LOCATION OF IDENTIFIER TABLE
      005026  002151 0400 00                ASX0    SYMSC            MAKE IDENTIFIER TABLE SC POINTER RELATIVE
      005027  005053 7100 00                TRA     NS8              GO TO CHECK LONG COUNT
   END OF BINARY CARD 00000080
      005030  002151 7550 52          NS2   STA     SYMSC,SC         STORE IN SYMBOL STRING
      005031  005023 6070 00                TTF     NS1              TRANSFER IF NO STRING OVERFLOW
      005032  005151 7000 00                TSX0    NSI              GET MORE MEMORY FOR ITAB STRING
      005033  005023 7100 00                TRA     NS1              AND CONTINUE
      005034  002154 7550 00          NS3   STA     TEMP1            STORE FOR COMPARE
      005035  035223 2200 00                LDX0    T$ITAB           GET LOCATION OF IDENTIFIER TABLE
      005036  002154 0400 00                ASX0    TEMP1            MAKE SC POINTER ABSOLUTE
      005037  001751 2350 00                LDA     IDENT            GET POINTER TO CURRENT INPUT SYMBOL
      005040  002155 7550 00                STA     TEMP2            STORE FOR COMPARE
      005041  002154 2350 52          NS4   LDA     TEMP1,SC         FETCH STRING1 CHARACTER
      005042  005143 6070 00                TTF     NS5              TRANSFER IF NOT AT END OF STRING1
      005043  002155 0110 52                NOP     TEMP2,SC         CHECK STRING2
      005044  005147 6070 00                TTF     NS7              TRANSFER IF NOT AT THE END OF STRING2
      005045  035222 1670 00                SBX7    T$STAB           MAKE SYMBOL TABLE POINTER RELATIVE
      005046  000020 1070 03                CMPX7   S$LONG,DU        WAS LAST SYMBOL READ 'LONG'
      005047  005052 6010 00                TNZ     NS8,1            TRANSFER IF NOT LONG
      005050  002162 0540 00                AOS     LNGCT            INCREMENT NUMBER OF LONGS ENCOUNTERED
      005051  004776 7100 00                TRA     NSR              AND GET NEXT SYMBOL
```

```
    005052  035222 0670 00        NS8,1  ADX7    T$STAB          MAKE SYMBOL TABLE POINTER ABSOLUTE
    005053  000000 6260 17        NS8    EAX6    0,7             SAVE POINTER TO IDENTIFIER WITHOUT LONGS
    005054  035222 1660 00               SBX6    T$STAB          MAKE IT RELATIVE
    005055  000000 2350 07               LDA     0,DL            GET A ZERO IN THE A REGISTER
 END OF BINARY CARD 00000081
    005056  002162 1550 00               SSA     LNGCT           NEGATE COUNT OF LONGS IN MEMORY
    005057  005130 6000 00        NS9    TZE     NS11            TRANSFER IF NO MORE LONGS TO CONSIDER
    005060  000001 7200 17               LXL0    DF,7            GET LINK TO ENTRY WITH ONE MORE LONG
    005061  005124 6010 00               TNZ     NS10            TRANSFER IF THERE IS ONE
    005062  035222 7200 00               LXL0    T$STAB          GET CURRENT END OF SYMBOL TABLE IN XR - 0
    005063  000001 4400 17               SXL0    DF,7            STORE LINK IN CURRENT TABLE ENTRY
    005064  000001 2350 03               LDA     ELEN-1,DU       GET ELEMENT LENGTH MINUS ONE
    005065  035222 2210 03               LDX1    T$STAB,DU       GET POINTER TO SYMBOL TABLE CONTROL WORD
    005066  005663 7000 00               TSX0    T$ALOC          ALLOCATE A NEW ELEMENT
    005067  777777 6200 11               EAX0    -1,1            PUT ADDRESS OF FIRST WORD IN XR - 0
    005070  000001 4500 10               STZ     DF,0            ZERO OUT LINK WORD IN NEW ENTRY
    005071  400000 6210 10               EAX1    131072,0        GET POINTER TO THIS ENTRY IN XR - 1
    005072  035222 1610 00               SBX1    T$STAB          MAKE POINTER RELATIVE
    005073  000001 7410 10               STX1    DF,0            AND STORE AS FINAL LINK FOR THIS ENTRY
    005074  002151 2350 00               LDA     SYMSC           GET RELATIVE TALLY WORD TO ITAB STRING
    005075  002150 3750 00               ANA     NTMSK           SET TALLY FIELD TO ZERO
    005076  000000 7550 10               STA     SC,0            AND STORE AS TALLY IN NEW ENTRY
    005077  000000 2350 17               LDA     SC,7            GET TALLY FROM OLD ENTRY
    005100  002154 7550 00               STA     TEMP1           AND SAVE
    005101  777700 3750 07               ANA     =0777700,DL     GET TALLY FIELD
    005102  000500 0750 07               ADA     =0500,DL        ADD 5 CHARACTERS FOR (LONG )
    005103  000000 2550 10               ORSA    SC,0            AND INSERT TALLY IN NEW TABLE ENTRY
 END OF BINARY CARD 00000082
    005104  002157 2350 00               LDA     LONGT           GET TALLY WORD FOR (LONG )
    005105  002155 7550 00               STA     TEMP2           AND STORE
    005106  035223 2270 00               LDX7    T$ITAB          GET ADDRESS OF BASE OF ITAB
    005107  002151 0470 00               ASX7    SYMSC           MAKE SYMSC TALLY WORD ABSOLUTE
    005110  002154 0470 00               ASX7    TEMP1           MAKE OLD ENTRY TALLY WORD ABSOLUTE
    005111  035222 1600 00               SBX0    T$STAB          MAKE POINTER TO NEW TABLE ENTRY RELATIVE
    005112  002156 7400 00               STX0    TEMP3           AND SAVE
    005113  005114 6270 00               EAX7    *+1             GET LOOP ADDRESS IN XR - 7
    005114  002155 2350 52               LDA     TEMP2,SC        GET NEXT CHARACTER FROM (LONG ) STRING
    005115  005133 6070 00               TTF     NS12            TRANSFER TO STORE CHARACTER
    005116  005117 6270 00               EAX7    *+1             GET LOOP ADDRESS IN XR - 7
    005117  002154 2350 52               LDA     TEMP1,SC        GET NEXT CHARACTER FROM OLD TABLE ENTRY
    005120  005133 6070 00               TTF     NS12            TRANSFER TO STORE CHARACTER
    005121  035223 3200 00               LCX0    T$ITAB          GET MINUS BASE OF ITAB TABLE
    005122  002151 0400 00               ASX0    SYMSC           AND MAKE SYMSC TALLY WORD RELATIVE
    005123  002156 2200 00               LDX0    TEMP3           RECOVER RELATIVE POINTER TO NEW ENTRY
    005124  000000 6270 10        NS10   EAX7    0,0             GET IN XR - 7 POINTER TO NEW TABLE ENTRY
    005125  035222 0670 00               ADX7    T$STAB          MAKE IT ABSOLUTE
    005126  002162 0540 00               AOS     LNGCT           DECREMENT NUMBER OF LONGS TO CONSIDER
    005127  005057 7100 00               TRA     NS9             AND LOOP
    005130  035222 1670 00        NS11   SBX7    T$STAB          MAKE SYMBOL TABLE POINTER RELATIVE
    005131  004777 2200 00               LDX0    NSX             GET RETURN ADDRESS
```

```
  END OF BINARY CARD 00000083
      005132  000001 7100 10        1207          TRA       1,0                 AND RETURN
      005133  002151 7550 52        1208 NS12     STA       SYMSC,SC            STORE CHARACTER IN STRING FOR NEW ENTRY
      005134  000000 6070 17        1209          TTF       0,7                 RETURN IF NO STRING OVERFLOW
      005135  035223 3200 00        1210          LCX0      T$ITAB              GET MINUS BASE OF ITAB TABLE
      005136  002154 0400 00        1211          ASX0      TEMP1               MAKE OLD ENTRY TALLY WORD RELATIVE
      005137  005151 7000 00        1212          TSX0      NSI                 GET MORE MEMORY FOR ITAB STRING
      005140  035223 2200 00        1213          LDX0      T$ITAB              GET BASE OF ITAB TABLE
      005141  002154 0400 00        1214          ASX0      TEMP1               MAKE OLD ENTRY TALLY WORD ABSOLUTE
      005142  000000 7100 17        1215          TRA       0,7                 AND RETURN
      005143  002155 1150 52        1216 NS5      CMPA      TEMP2,SC            COMPARE WITH CHARACTER FROM STRING2
      005144  005146 6070 00        1217          TTF       NS6                 TRANSFER IF NOT AT THE END OF STRING2
      005145  005147 7100 00        1218          TRA       NS7                 AT END OF STRING2 SO NOT EQUAL
      005146  005041 6000 00        1219 NS6      TZE       NS4                 TRANSFER IF CORRESPONDING CHARACTERS ARE EQUAL
      005147  000002 0670 03        1220 NS7      ADX7      ELEN,DU             STEP TO NEXT SYMBOL TABLE ENTRY
      005150  005001 7100 00        1221          TRA       NSL                 AND LOOP
      005151  005163 7400 00        1222 NSI      STX0      NSIX                SAVE RETURN
      005152  035223 3200 00        1223          LCX0      T$ITAB              GET MINUS BASE OF ITAB TABLE
      005153  002151 0400 00        1224          ASX0      SYMSC               MAKE ITAB TABLE TALLY WORD RELATIVE
      005154  000000 2350 03        1225          LDA       0,DU                ASK FOR ONE ADDITIONAL WORD FOR ITAB
      005155  035223 2210 03        1226          LDX1      T$ITAB,DU           GET POINTER TO ITAB CONTROL WORD
      005156  005663 7000 00        1227          TSX0      T$ALOC              AND ALLOCATE SOME MEMORY
      005157  000400 2350 07        1228          LDA       =0400,DL            GET TALLY OF 4 CHARACTERS
  END OF BINARY CARD 00000084
      005160  002151 0550 00        1229          ASA       SYMSC               AND ADD TO NUMBER OF CHARACTERS IN SYMSC
      005161  035223 2200 00        1230          LDX0      T$ITAB              GET BASE OF ITAB TABLE
      005162  002151 0400 00        1231          ASX0      SYMSC               MAKE ITAB TABLE TALLY WORD ABSOLUTE
      005163  000000 7100 00        1232 NSIX     TRA       **                  AND RETURN
      005164  005206 7400 00        1233 PEEK     STX0      PEEKX               SAVE RETURN
      005165  002153 7260 00        1234 PEEK1    LXL6      QEEKF               GET LAST SYMBOL WITHOUT LONGS READ
      005166  002153 2270 00        1235          LDX7      QEEKF               GET LAST SYMBOL READ
      005167  005174 6010 00        1236          TNZ       PEEK2               TRANSFER IF THERE IS ONE
      005170  004774 7000 00        1237          TSX0      NS                  READ A SYMBOL
      005171  400000 2270 03        1238          LDX7      =0400000,DU         GET END OF FILE FLAG
      005172  002153 4460 00        1239          SXL6      QEEKF               STORE AS LAST SYMBOL WITHOUT LONGS READ
      005173  002153 7470 00        1240          STX7      QEEKF               STORE AS LAST SYMBOL READ
      005174  002152 7260 00        1241 PEEK2    LXL6      PEEKF               GET NEXT TO LAST SYMBOL WITHOUT LONGS READ
      005175  002152 2270 00        1242          LDX7      PEEKF               GET NEXT TO LAST SYMBOL READ
      005176  005206 6010 00        1243          TNZ       PEEKX               TRANSFER IF IT EXISTS
      005177  002153 7260 00        1244          LXL6      QEEKF               MOVE QEEKF
      005200  002152 4460 00        1245          SXL6      PEEKF               TO PEEKF
      005201  002153 2270 00        1246          LDX7      QEEKF               MOVE QEEKF
      005202  002152 7470 00        1247          STX7      PEEKF               TO PEEKF
      005203  005206 6040 00        1248          TMI       PEEKX               EXIT IF END OF FILE
      005204  002153 4500 00        1249          STZ       QEEKF               ERASE SYMBOL IN QEEKF
      005205  005165 7100 00        1250          TRA       PEEK1               AND TRY AGAIN
  END OF BINARY CARD 00000085
      005206  000000 6040 00        1251 PEEKX    TMI       **                  GIVE END OF FILE RETURN IF END OF FILE FLAG
      005207  005206 2200 00        1252          LDX0      PEEKX               GET RETURN ADDRESS
      005210  000001 7100 10        1253          TRA       1,0                 AND GIVE NORMAL RETURN
```

```
           A                                PASS0

                                       1254          HEAD     P
     005211  035235 7400 56            1255 MODE     STX0     A$STACK,ID          SAVE RETURN
     005212  005742 7170 00            1256          XED      T$SOVF              CHECK FOR STACK OVERFLOW
     005213  010630 7000 00            1257 MODE1    TSX0     A$XFER              MAKE MODE UNIQUE AND ABSOLUTE
     005214  035216 1670 00            1258          SBX7     T$MODE              MAKE MODE POINTER RELATIVE
     005215  005255 7470 00            1259          STX7     MOD                 AND STORE
     005216  035222 2220 00            1260          LDX2     T$STAB              GET POINTER TO START OF SYMBOL TABLE
     005217  000001 2230 12            1261 MODE2    LDX3     A$DF,2              GET POINTER TO DEFINITION CHAIN FOR THIS SYMBOL
     005220  005225 6050 00            1262 MODE3    TPL      MODE4               TRANSFER IF ANY DEFINITIONS
     005221  000002 0620 03            1263          ADX2     A$ELEN,DU           STEP POINTER TO NEXT SYMBOL
     005222  000000 2340 12            1264          SZN      0,2                 SEE IF ANY MORE ENTRIES IN TABLE
     005223  005217 6010 00            1265          TNZ      MODE2               LOOP IF MORE ENTRIES IN SYMBOL TABLE
     005224  005245 7100 00            1266          TRA      MODE7               TRANSFER IF MODE HAS NO SYMBOL
     005225  035220 0630 00            1267 MODE4    ADX3     T$DEF               MAKE DEFINITION POINTER ABSOLUTE
     005226  000001 2240 13            1268          LDX4     1,3                 GET TYPE OF DEFINITION IN XR - 4
     005227  006412 1040 03            1269          CMPX4    D$MODE,DU           SEE IF THIS IS A MODE DEFINITION
     005230  005234 6010 00            1270          TNZ      MODE4,              TRANSFER IF NOT A MODE DEFINITION
     005231  000002 2240 13            1271          LDX4     2,3                 GET MODE DEFINED IN XR - 4
     005232  005255 1040 00            1272          CMPX4    MOD                 SEE IF THIS IS THE MODE WE ARE LOOKING FOR
     005233  005236 6000 00            1273          TZE      MODE5               TRANSFER IF YES - MODE DEFINITION IS FOUND
 END OF BINARY CARD 00000086
     005234  000000 2230 13            1274 MODE4,   LDX3     0,3                 GET POINTER TO NEXT DEFINITION IN CHAIN
     005235  005220 7100 00            1275          TRA      MODE3               AND TRANSFER TO KEEP SEARCHING
     005236  000000 2350 12            1276 MODE5    LDA      A$SC,2              GET TALLY WORD FOR SYMBOL
     005237  005243 7550 00            1277          STA      MODE6               AND STORE FOR PRINT ROUTINE
     005240  035223 2200 00            1278          LDX0     T$ITAB              GET ADDRESS OF BASE OF IDENTIFIER TABLE
     005241  005243 0400 00            1279          ASX0     MODE6               AND MAKE TALLY WORD ABSOLUTE
     005242  005464 7000 00            1280          TSX0     PRINT               PRINT NAME OF MODE
     005243  000000 000000             1281 MODE6    ZERO
     005244  035235 7100 55            1282          TRA      A$STACK,DIC         AND RETURN TO CALLER
     005245  005255 2270 00            1283 MODE7    LDX7     MOD                 GET POINTER TO MODE IN XR - 7
     005246  035216 0670 00            1284          ADX7     T$MODE              MAKE MODE POINTER ABSOLUTE
     005247  000000 2210 17            1285          LDX1     0,7                 GET TYPE OF MODE IN XR - 1
     005250  005256 2220 03            1286          LDX2     MT,DU               GET POINTER TO TRANSFER TABLE IN XR - 2
     005251  020300 5202 01            1287          RPT      MTE-MT,1,TZE        SEARCH TABLE
     005252  000000 1010 12            1288          CMPX1    0,2                 FOR MODE
     005253  777777 2350 12            1289          LDA      -1,2                GET MATCHING ENTRY IN A
     005254  000000 7100 05            1290          TRA      0,AL                AND TRANSFER TO APPROPRIATE ROUTINE
     005255  000000 000000             1291 MOD      ZERO
     005256  016762 005305             1292 MT       ZERO     M$REF,REF
     005257  017001 005337             1293          ZERO     M$PROC,PROC
     005260  016757 005364             1294          ZERO     M$STRCT,STRCT
     005261  016754 005272             1295          ZERO     M$PRIM,PRIM
 END OF BINARY CARD 00000087
     005262  016770 005312             1296          ZERO     M$ROW,ROW
     005263  016776 005317             1297          ZERO     M$ROWE,ROWE
     005264  017007 005426             1298          ZERO     M$UNION,UNION
     005265  000000 005266             1299          ZERO     ,MERR
                    005266             1300 MTE      EQU      *
     005266  005464 7000 00            1301 MERR     TSX0     PRINT               PRINT AN ASTERISK FOR AN INVALID MODE
```

```
            P                              PASS0

      005267  005271 0002 40       1302          TALLYB   MERRT,2
      005270  035235 7100 55       1303          TRA      A$STACK,DIC       AND RETURN TO CALLER
      005271  052040040040         1304 MERRT    UASCI    1,*
      005272  000000 2360 17       1305 PRIM     LDQ      0,7               GET NUMBER OF PRIMITIVE MODE IN Q
      005273  000077 3760 07       1306          ANQ      =077,DL           MASK OUT ALL BUT PRIMITIVE NUMBER
      005274  000012 5060 07       1307          DIV      10,DL             CONVERT TO TWO DECIMAL DIGITS
      005275  000033 7350 00       1308          ALS      36-9              MOVE UNITS DIGIT NEXT TO TENS DIGIT
      005276  000077 7770 00       1309          LLR      72-9              MOVE TWO DIGIT NUMBER TO LEFT OF A REGISTER
      005277  060060 0750 03       1310          ADA      =0060060,DU       ADD ASCII ZEROS
      005300  005304 7550 00       1311          STA      PRIMT             AND STORE NUMBER
      005301  005464 7000 00       1312          TSX0     PRINT             PRINT PRIMITIVE TYPE
      005302  005304 0003 40       1313          TALLYB   PRIMT,3
      005303  035235 7100 55       1314          TRA      A$STACK,DIC       AND RETURN TO CALLER
      005304  000000 000000        1315 PRIMT    ZERO
      005305  005464 7000 00       1316 REF      TSX0     PRINT             PRINT 'REF '
      005306  005311 0005 40       1317          TALLYB   REFT,5
      005307  000001 2270 17       1318          LDX7     1,7               GET REFERENCED MODE IN XR - 7
  END OF BINARY CARD 00000088
      005310  005213 7100 00       1319          TRA      MODE1             AND PRINT IT
      005311  122105106040         1320 REFT     UASCI    1,REF
      005312  005325 7000 00       1321 ROW      TSX0     ROWI              PRINT '[' IF FIRST ROW
      005313  005464 7000 00       1322          TSX0     PRINT             PRINT ','
      005314  005461 0002 40       1323          TALLYB   COMMA,2
      005315  000001 2270 17       1324          LDX7     1,7               GET DEROWED MODE IN XR - 7
      005316  005213 7100 00       1325          TRA      MODE1             AND PRINT IT
      005317  005325 7000 00       1326 ROWE     TSX0     ROWI              PRINT '[' IF FIRST ROW
      005320  005464 7000 00       1327          TSX0     PRINT             PRINT ']'
      005321  005336 0002 40       1328          TALLYB   BUS,2
      005322  005334 4500 00       1329          STZ      ROWF              RESET ROW FLAG
      005323  000001 2270 17       1330          LDX7     1,7               GET DEROWED MODE IN XR - 7
      005324  005213 7100 00       1331          TRA      MODE1             AND PRINT IT
      005325  005334 2340 00       1332 ROWI     SZN      ROWF              CHECK IF ALREADY IN ROW MODE
      005326  000000 6010 10       1333          TNZ      0,0               RETURN IF SO
      005327  005333 7400 00       1334          STX0     ROWIX             SAVE RETURN
      005330  005464 7000 00       1335          TSX0     PRINT             PRINT '['
      005331  005335 0002 40       1336          TALLYB   SUB,2
      005332  005334 7500 00       1337          STC2     ROWF              SET ROW FLAG
      005333  000000 7100 00       1338 ROWIX    TRA      **                AND RETURN
      005334  000000 000000        1339 ROWF     ZERO
      005335  133040040040         1340 SUB      UASCI    1,[
  END OF BINARY CARD 00000089
      005336  135040040040         1341 BUS      UASCI    1,]
      005337  777777 2260 17       1342 PROC     LDX6     -1,7              GET NUMBER OF PARAMETERS PLUS TWO IN XR - 6
      005340  000001 6250 17       1343          EAX5     1,7               GET POINTER TO FIRST ARGUMENT IN XR - 5
      005341  000002 1660 03       1344          SBX6     2,DU              GET NUMBER OF PARAMETERS IN XR - 6
      005342  777777 6040 00       1345          TMI      $ERROR            MUST BE AT LEAST ZERO OR ERROR
      005343  005352 6010 00       1346          TNZ      PROC1             TRANSFER IF ANY ARGUMENTS
      005344  005464 7000 00       1347          TSX0     PRINT             PRINT 'PROC'
      005345  005362 0005 40       1348          TALLYB   PROCT,5
      005346  005464 7000 00       1349          TSX0     PRINT             PRINT A SPACE
```

```
              P                          PASS0

      005347  005462 0002 40       1350          TALLYB  SPACE,2
      005350  000000 2270 15       1351          LDX7    0,5                GET POINTER TO RESULT MODE
      005351  005213 7100 00       1352          TRA     MODE1              AND PRINT IT
      005352  005464 7000 00       1353 PROC1    TSX0    PRINT              PRINT 'PROC('
      005353  005362 0006 40       1354          TALLYB  PROCT,6
      005354  035235 6200 56       1355          EAX0    A$STACK,ID         GET POINTER TO NEXT WORD IN CONTROL STACK
      005355  005742 7170 00       1356          XED     T$SOVF             CHECK FOR STACK OVERFLOW
      005356  000000 7500 10       1357          STC2    0,0                SAVE RETURN
      005357  005433 7100 00       1358          TRA     UN1                AND GO PRINT OUT PARAMETER MODES
      005360  000001 2270 15       1359          LDX7    1,5                GET MODE OF RESULT
      005361  005213 7100 00       1360          TRA     MODE1              AND PRINT IT
      005362  120122117103         1361 PROCT    UASCI   2,PROC(
      005363  050040040040
  END OF BINARY CARD 00000090
      005364  005464 7000 00       1362 STRCT    TSX0    PRINT              PRINT 'STRUCT('
      005365  005424 0010 40       1363          TALLYB  STRT,8
      005366  777777 2260 17       1364          LDX6    -1,7               GET NUMBER OF FIELDS PLUS ONE IN XR - 6
      005367  000001 6250 17       1365          EAX5    1,7                GET POINTER TO FIRST FIELD IN XR - 5
      005370  000001 1660 03       1366          SBX6    1,DU               GET NUMBER OF FIELDS IN XR - 6
      005371  777777 6000 00       1367          TZE     $ERROR             ERROR - MUST BE AT LEAST ONE FIELD
      005372  000000 2270 15       1368 STR1     LDX7    0,5                GET MODE OF FIRST FIELD IN XR - 7
      005373  035216 1650 00       1369          SBX5    T$MODE             MAKE POINTER RELATIVE
      005374  035235 7450 56       1370          STX5    A$STACK,ID         SAVE FIELD POINTER
      005375  005742 7170 00       1371          XED     T$SOVF             CHECK FOR STACK OVERFLOW
      005376  035235 7460 56       1372          STX6    A$STACK,ID         SAVE NUMBER OF REMAINING FIELDS
      005377  005742 7170 00       1373          XED     T$SOVF             CHECK FOR STACK OVERFLOW
      005400  005211 7000 00       1374          TSX0    MODE               PRINT MODE OF FIELD
      005401  035235 2260 54       1375          LDX6    A$STACK,DI         RESTORE NUMBER OF REMAINING FIELDS
      005402  035235 2250 54       1376          LDX5    A$STACK,DI         RESTORE FIELD POINTER
      005403  035216 0650 00       1377          ADX5    T$MODE             MAKE POINTER ABSOLUTE
      005404  005464 7000 00       1378          TSX0    PRINT              PRINT A SPACE
      005405  005462 0002 40       1379          TALLYB  SPACE,2
      005406  000000 7200 15       1380          LXL0    0,5                GET TAG IN XR - 0
      005407  035222 0600 00       1381          ADX0    T$STAB             MAKE TAB POINTER ABSOLUTE
      005410  000000 2350 10       1382          LDA     0,0                GET TALLY WORD FOR TAG IN A
      005411  005415 7550 00       1383          STA     STR2               AND STORE FOR PRINT ROUTINE
  END OF BINARY CARD 00000091
      005412  035223 2200 00       1384          LDX0    T$ITAB             GET BASE OF IDENTIFIER TABLE IN XR - 0
      005413  005415 0400 00       1385          ASX0    STR2               AND MAKE TALLY WORD ABSOLUTE
      005414  005464 7000 00       1386          TSX0    PRINT              PRINT TAG
      005415  000000 000000        1387 STR2     ZERO
      005416  000001 1660 03       1388          SBX6    1,DU               DECREMENT NUMBER OF REMAINING FIELDS
      005417  005454 6000 00       1389          TZE     UN3                TRANSFER IF NONE LEFT - GO PRINT ')'
      005420  000001 0650 03       1390          ADX5    1,DU               MAKE XR - 5 POINT TO NEXT FIELD
      005421  005464 7000 00       1391          TSX0    PRINT              PRINT ', '
      005422  005461 0003 40       1392          TALLYB  COMMA,3
      005423  005372 7100 00       1393          TRA     STR1               AND LOOP
      005424  123124122125         1394 STRT     UASCI   2,STRUCT(
      005425  103124050040
      005426  005464 7000 00       1395 UNION    TSX0    PRINT              PRINT 'UNION('
    005427  005457 0007 40             TALLYB  UNT,7
    005430  777777 2260 17             LDX6    -1,7                 GET NUMBER OF MODES IN UNION PLUS ONE
    005431  000001 6250 17             EAX5    1,7                  GET POINTER TO FIRST MODE IN XR = 5
    005432  000001 1660 03             SBX6    1,DU                 GET NUMBER OF MODES IN XR = 6
    005433  777777 6000 00       UN1   TZE     $ERROR               ERROR - MUST BE AT LEAST ONE MODE
    005434  000000 2270 15       UN2   LDX7    0,5                  GET NEXT MODE IN XR = 7
    005435  035216 1650 00             SBX5    T$MODE               MAKE POINTER RELATIVE
    005436  035235 7450 56             STX5    A$STACK,ID           SAVE MODE POINTER
    005437  005742 7170 00             XED     T$SOVF               CHECK FOR STACK OVERFLOW
  END OF BINARY CARD 00000092
    005440  035235 7460 56             STX6    A$STACK,ID           SAVE NUMBER OF REMAINING MODES
    005441  005742 7170 00             XED     T$SOVF               CHECK FOR STACK OVERFLOW
    005442  005211 7000 00             TSX0    MODE                 PRINT OUT MODE
    005443  035235 2260 54             LDX6    A$STACK,DI           RESTORE NUMBER OF REMAINING MODES
    005444  035235 2250 54             LDX5    A$STACK,DI           RESTORE MODE POINTER
    005445  035216 0650 00             ADX5    T$MODE               MAKE POINTER ABSOLUTE
    005446  000001 1660 03             SBX6    1,DU                 DECREMENT NUMBER OF REMAINING MODES
    005447  005454 6000 00             TZE     UN3                  TRANSFER IF NO MORE MODES TO PRINT
    005450  000001 0650 03             ADX5    1,DU                 STEP MODE POINTER TO NEXT MODE
    005451  005464 7000 00             TSX0    PRINT                PRINT ', '
    005452  005461 0003 40             TALLYB  COMMA,3
    005453  005434 7100 00             TRA     UN2                  AND GO PRINT NEXT MODE
    005454  005464 7000 00       UN3   TSX0    PRINT                PRINT ')'
    005455  005463 0002 40             TALLYB  RPAR,2
    005456  035235 7100 55             TRA     A$STACK,DIC          AND RETURN TO CALLER
    005457  125116111117         UNT   UASCI   2,UNION(
    005460  116050040040
    005461  054040040040         COMMA UASCI   1,,
    005462  040040040040         SPACE UASCI   1, 
    005463  051040040040         RPAR  UASCI   1,)
    005464  000000 2350 10       PRINT LDA     0,0                  GET TALLY WORD OF STRING
    005465  005473 7550 00             STA     PRIT                 AND STORE IN PRINT TALLY WORD
  END OF BINARY CARD 00000093
    005466  005473 2350 52       PRI1  LDA     PRIT,SC              GET NEXT CHARACTER OF STRING
    005467  005471 6070 00             TTF     PRI2                 TRANSFER IF THERE IS ANOTHER CHARACTER
    005470  000001 7100 10             TRA     1,0                  RETURN TO CALLER
    005471  006607 7550 52       PRI2  STA     1$XX2,SC             STORE CHARACTER IN OUTPUT BUFFER
    005472  005466 7100 00             TRA     PRI1                 AND LOOP
    005473  000000 000000        PRIT  ZERO
    005474  006603 7400 00       PMODE STX0    1$XRET               SAVE RETURN
    005475  006610 2350 00             LDA     1$XXT                GET INITIAL PRINT ROUTINE TALLY WORD
    005476  006607 7550 00             STA     1$XX2                AND INITIALIZE OUTPUT TALLY WORD
    005477  005211 7000 00             TSX0    MODE                 ASSEMBLE MODE IN OUTPUT BUFFER
    005500  006571 7100 00             TRA     1$XXL3               AND PRINT IT
    005501  000001 2210 03       PMTBL LDX1    1,DU                 GET RELATIVE POINTER TO FIRST ENTRY IN MODE TABLE
    005502  005532 7410 00       P1    STX1    T                    STORE POINTER TO NEXT MODE TO BE PRINTED
    005503  006610 2350 00             LDA     1$XXT                GET INITIAL OUTPUT TALLY WORD IN A
    005504  006607 7550 00             STA     1$XX2                AND INITIALIZE WORKING TALLY WORD
    005505  005532 2350 00             LDA     T                    GET NUMBER OF MODE IN AU
    005506  000006 2200 03             LDX0    6,DU                 GET NUMBER OF OCTAL DIGITS TO CONVERT
```

```
          P                                   PASS0

      005507  000006 2360 07         1443 P2      LDQ      6,DL              GET ASCII DIGIT ZERO SHIFTED RIGHT
      005510  000003 7770 00         1444         LLR      3                 GET NEXT ASCII DIGIT IN QL
      005511  006607 7560 52         1445         STQ      1$XX2,SC          STORE DIGIT IN OUTPUT BUFFER
      005512  000001 1600 03         1446         SBX0     1,DU              DECREMENT NUMBER OF DIGITS LEFT TO CONVERT
      005513  005507 6010 00         1447         TNZ      P2                TRANSFER IF THERE ARE MORE DIGITS TO CONVERT
   END OF BINARY CARD 00000094
      005514  000040 2350 07         1448         LDA      =0040,DL          GET AN ASCII SPACE
      005515  006607 7550 52         1449         STA      1$XX2,SC          STORE A SPACE IN THE OUTPUT BUFFER
      005516  005532 2270 00         1450         LDX7     T                 GET POINTER TO MODE TABLE ENTRY TO BE PRINTED
      005517  005211 7000 00         1451         TSX0     MODE              GO PRINT IT
      005520  006603 7500 00         1452         STC2     1$XRET            SAVE RETURN
      005521  006571 7100 00         1453         TRA      1$XXL3            PRINT LINE
      005522  005532 2210 00         1454         LDX1     T                 GET POINTER TO MODE JUST PRINTED
      005523  035216 0610 00         1455         ADX1     T$MODE            MAKE POINTER ABSOLUTE
      005524  777777 0610 11         1456         ADX1     -1,1              STEP TO NEXT ENTRY IN MODE TABLE
      005525  000000 2340 11         1457         SZN      0,1               SEE IF THERE ARE ANY MORE ENTRIES IN TABLE
      005526  005653 6000 00         1458         TZE      STOP              NO MORE - HALT PROGRAM
      005527  000001 0610 03         1459         ADX1     1,DU              STEP OVER LINK WORD IN TABLE
      005530  035216 1610 00         1460         SBX1     T$MODE            MAKE MODE TABLE POINTER RELATIVE
      005531  005502 7100 00         1461         TRA      P1                AND GO PRINT NEXT MODE
      005532  000000 000000          1462 T       ZERO
      005533  005551 7400 00         1463 DEF     STX0     DEFX              SAVE RETURN
      005534  005562 7410 00         1464         STX1     DEFT              SAVE POINTER TO DEF TABLE ENTRY
      005535  035222 2210 00         1465         LDX1     T$STAB            GET POINTER TO START OF SYMBOL TABLE
      005536  000001 6220 11         1466 DEF1    EAX2     A$DF,1            GET POINTER TO DEFINITION CHAIN IN XR - 2
      005537  000000 2220 12         1467 DEF2    LDX2     0,2               GET POINTER TO NEXT DEFINITION IN CHAIN
      005540  005554 6040 00         1468         TMI      DEF5              TRANSFER IF THERE ARE NO MORE
      005541  005562 1020 00         1469         CMPX2    DEFT              SEE IF IT IS THE ENTRY WE ARE LOOKING FOR
   END OF BINARY CARD 00000095
      005542  005552 6010 00         1470         TNZ      DEF4              TRANSFER IF NOT - KEEP LOOKING
      005543  000000 2350 11         1471         LDA      A$SC,1            GET TALLY WORD TO SYMBOL
      005544  005550 7550 00         1472         STA      DEF3              AND STORE FOR PRINTING
      005545  035223 2200 00         1473         LDX0     T$ITAB            GET POINTER TO BASE OF IDENTIFIER TABLE
      005546  005550 0400 00         1474         ASX0     DEF3              AND MAKE TALLY WORD ABSOLUTE
      005547  005464 7000 00         1475         TSX0     PRINT             PRINT OUT SYMBOL
      005550  000000 0000 40         1476 DEF3    TALLYB   **
      005551  000000 7100 00         1477 DEFX    TRA      **                AND RETURN
      005552  035220 0620 00         1478 DEF4    ADX2     T$DEF             MAKE DEF TABLE ENTRY POINTER ABSOLUTE
      005553  005537 7100 00         1479         TRA      DEF2              AND GO CONTINUE DOWN DEFINITION CHAIN
      005554  000002 0610 03         1480 DEF5    ADX1     A$ELEN,DU         STEP TO NEXT ENTRY IN SYMBOL TABLE
      005555  000000 2340 11         1481         SZN      0,1               SEE IF AT THE END OF THE TABLE
      005556  005536 6010 00         1482         TNZ      DEF1              TRANSFER IF NOT AT END AND CONTINUE LOOKING
      005557  005464 7000 00         1483         TSX0     PRINT             PRINT OUT ERROR SYMBOL
      005560  005563 0003 40         1484         TALLYB   DEFE,3
      005561  005551 7100 00         1485         TRA      DEFX              AND EXIT
      005562  000000 000000          1486 DEFT    ZERO
      005563  056056040040           1487 DEFE    UASCI    1,..
      005564  035224 2350 00         1488 PCTBL   LDA      T$CODE            GET DESCRIPTOR TO CODE TABLE
      005565  000001 0750 07         1489         ADA      1,DL              ADJUST TALLY RUN OUT
      005566  777777 3750 07         1490         ANA      -1,DL             GET LENGTH OF CODE TABLE IN AL
```

```
          P                         PASS0

    005567  000006 7350 00        1491          ALS     6                   GET IN TALLY FIELD
 END OF BINARY CARD 00000096
    005570  005652 7550 00        1492          STA     CODEP               AND STORE
    005571  005652 6210 56        1493 PCTL     EAX1    CODEP,ID            GET RELATIVE POINTER TO NEXT CODE WORD
    005572  005574 6070 00        1494          TTF     PCT1                TRANSFER IF NOT AT END OF TABLE
    005573  005653 7100 00        1495          TRA     STOP                ALL DONE - HALT
    005574  035224 0610 00        1496 PCT1     ADX1    T$CODE              MAKE CODE TABLE POINTER ABSOLUTE
    005575  000000 2350 11        1497          LDA     0,1                 GET CODE WORD IN A REGISTER
    005576  016476 1150 03        1498          CMPA    0$OTBL,DU           SEE IF CODE IS BEFORE TABLE
    005577  777777 6040 00        1499          TMI     $ERROR              COMPILER ERROR - CODE IS ILLEGAL
    005600  017037 1150 03        1500          CMPA    0$OTBLE,DU          SEE IF BEYOND END OF TABLE
    005601  777777 6050 00        1501          TPL     $ERROR              COMPILER ERROR - CODE IS ILLEGAL
    005602  006610 2360 00        1502          LDQ     1$XXT               GET INITIAL TALLY WORD FOR PRINT ROUTINE
    005603  006607 7560 00        1503          STQ     1$XX2               AND INITIALIZE PRINT ROUTINE
    005604  000000 2360 07        1504          LDQ     0,DL                GET A NULL CHARACTER IN QL
    005605  006607 7560 52        1505          STQ     1$XX2,SC            ADD NULLS TO FILL OUT WORD
    005606  006607 7560 52        1506          STQ     1$XX2,SC            ADD NULLS TO FILL OUT WORD
    005607  000001 2360 01        1507          LDQ     1,AU                GET FIRST WORD OF ASCII
    005610  006607 7560 56        1508          STQ     1$XX2,ID            AND STORE IN PRINT BUFFER
    005611  000002 2360 01        1509          LDQ     2,AU                GET SECOND WORD OF ASCII
    005612  006607 7560 56        1510          STQ     1$XX2,ID            AND STORE IN PRINT BUFFER
    005613  000000 7200 01        1511          LXL0    0,AU                GET TYPE OF ARGUMENT FOR CODE IN XR - 0
    005614  000004 1000 03        1512          CMPX0   4,DU                SEE IF TYPE IS IN RANGE
    005615  777777 6030 00        1513          TRC     $ERROR              NO - COMPILER ERROR
 END OF BINARY CARD 00000097
    005616  005616 7100 10        1514          TRA     *,0                 GO TO APPROPRIATE PRINT ROUTINE
    005617  005625 7100 00        1515          TRA     OCT                 OCTAL ARGUMENT
    005620  005635 7100 00        1516          TRA     CMOD                MODE ARGUMENT
    005621  005640 7100 00        1517          TRA     CDEF                SYMBOL ARGUMENT
    005622  006603 7500 00        1518 PCTE     STC2    1$XRET              SAVE RETURN
    005623  006571 7100 00        1519          TRA     1$XXL3              AND PRINT BUFFER
    005624  005571 7100 00        1520          TRA     PCTL                AND LOOP
    005625  000000 6360 05        1521 OCT      EAQ     0,AL                GET NUMBER TO CONVERT IN QU
    005626  000006 2220 03        1522          LDX2    6,DU                CONVERT 6 DIGITS
    005627  000006 2350 07        1523 OCT1     LDA     6,DL                GET ASCII ZERO SHIFTED RIGHT 3 PLACES
    005630  000003 7370 00        1524          LLS     3                   GET DIGIT IN AL
    005631  006607 7550 52        1525          STA     1$XX2,SC            STORE DIGIT IN OUTPUT BUFFER
    005632  000001 1620 03        1526          SBX2    1,DU                DECREMENT NUMBER OF DIGITS LEFT TO CONVERT
    005633  005627 6010 00        1527          TNZ     OCT1                TRANSFER IF MORE DIGITS TO CONVERT
    005634  005622 7100 00        1528          TRA     PCTE                AND EXIT
    005635  000000 6270 05        1529 CMOD     EAX7    0,AL                GET MODE IN XR - 7
    005636  005211 7000 00        1530          TSX0    MODE                AND PRINT IT
    005637  005622 7100 00        1531          TRA     PCTE                AND EXIT
    005640  000000 6210 05        1532 CDEF     EAX1    0,AL                GET DEF TABLE POINTER IN XR - 1
    005641  005645 7410 00        1533          STX1    CDEF1               AND SAVE
    005642  005533 7000 00        1534          TSX0    DEF                 PRINT SYMBOL
    005643  005464 7000 00        1535          TSX0    PRINT               PRINT A SPACE
 END OF BINARY CARD 00000098
    005644  005462 0004 40        1536          TALLYB  SPACE,4
    005645  000000 2210 03        1537 CDEF1    LDX1    **,DU               GET DEF TABLE POINTER IN XR - 1
```

```
           P                                 PASS0

      005646  035220 0610 00        1538             ADX1    T$DFF              MAKE POINTER ABSOLUTE
      005647  000002 2270 11        1539             LDX7    2,1                GET MODE OF DEFINITION IN XR - 7
      005650  005211 7000 00        1540             TSX0    MODE               AND PRINT IT
      005651  005622 7100 00        1541             TRA     PCTE               AND EXIT
      005652  000000 000000         1542 CODEP       ZERO
      005653  000221 5540 51        1543 STOP        STC1    ,$IC,I             SAVE IC
      005654  000220 7530 51        1544             SREG    ,$REG,I            SAVE REGISTERS
      005655  002623 7000 00        1545             TSX0    $TTB               BUFFER OUTPUT MESSAGE
      005656  005660 0014 40        1546             TALLYB  STOPM,11+1
      005657  002270 7100 00        1547             TRA     ,$EXIT             AND DISCONTINUE PROCESS
      005660  015012012122          1548 STOPM       OCT     015012012122,105101104131,015012012012
      005661  105101104131
      005662  015012012012
                                    1549 *
                                    1550 *           THE TABLE ALLOCATION ROUTINE ALLOCATES DYNAMIC SPACE
                                    1551 *           FOR THE SEVERAL TABLES REQUIRED DURING COMPILATION,
                                    1552 *           THE CALLING SEQUENCE IS A TSX0 WITH THE A REGISTER
                                    1553 *           CONTAINING THE LENGTH,DU OF THE NEW TABLE ENTRY
                                    1554 *           REQUESTED AND XR - 1 CONTAINING A POINTER TO THE
                                    1555 *           TABLE CONTROL WORD OF THE APPROPRIATE TABLE,   THE
                                    1556 *           SOUBROUTINE RETURNS IN XR - 1 A POINTER TO THE
                                    1557 *           NEWLY CREATED TABLE ENTRY,   THE FORMAT OF A TABLE
                                    1558 *           CONTROL WORD IS UPPER HALF IS ABSOLUTE LOCATION
                                    1559 *           AND LOWER HALF IS THE CURRENT LENGTH OF THE TABLE.
                                    1560 *
                                    1561             HEAD    T
      005663  006010 7530 00        1562 ALOC        SREG    REG                SAVE REGISTERS
      005664  006031 7550 00        1563             STA     TEMP               SAVE LENGTH REQUESTED
      005665  000000 7220 11        1564             LXL2    0,1                GET CURRENT LENGTH OF TABLE
      005666  006031 0620 00        1565             ADX2    TEMP               ADD REQUESTED LENGTH TO GET NEW LENGTH
      005667  000001 0620 03        1566             ADX2    1,DU               ADD ONE FOR LINK/LENGTH WORD
      005670  000000 4420 11        1567             SXL2    0,1                SAVE NEW LENGTH OF TABLE
      005671  000000 0620 11        1568             ADX2    0,1                ADD CURRENT LOCATION OF TABLE
  END OF BINARY CARD 00000099
      005672  000001 1620 11        1569             SBX2    1,1                SUBTRACT LOCATION OF ADJACIENT TABLE
      005673  005731 6040 00        1570             TMI     ALOC4              TRANSFER IF NO TABLE MOVING HAS TO BE DONE
      005674  000145 0620 03        1571             ADX2    1+100,DU           ADD ONE PLUS EXTRA SPACE TO BE ALLOCATED
      005675  005677 7420 00        1572             STX2    AINC               SAVE ADDITIONAL LENGTH NEEDED
      005676  035233 2230 00        1573             LDX3    T$END              GET ADDRESS OF END OF TABLES
      005677  000000 6240 13        1574 AINC        EAX4    **,3               GET ADDRESS OF NEW END OF TABLE IN XR - 4
      005700  005701 5500 00        1575             SBAR    *+1                STORE CURRENT LENGTH OF CORE
      005701  000000 2360 03        1576             LDQ     **,DU              AND PUT IT IN Q
      005702  000011 7360 00        1577             QLS     9                  GET LENGTH IN QU
      005703  000000 6250 02        1578             EAX5    0,QU               PUT IN XR - 5 FOR HALFWORD OPERATION
      005704  005705 7450 00        1579             STX5    *+1                AND STORE FOR COMPARE
      005705  000000 1040 03        1580             CMPX4   **,DU              IS THERE ENOUGH CORE
      005706  005713 6020 00        1581             TNC     ALOC1              TRANSFER IF OK
      005707  000001 6250 14        1582             EAX5    1,4                GET LENGTH OF CORE NEEDED IN XR - 5
      005710  500006 0010 00        1583             MME     M$MREQ             AND TRY TO GET MORE CORE
      005711  000000 6250 15        1584             EAX5    0,5                DID WE GET IT
```

```
     005712  777777 6010 00          1585          TNZ      $ERROR             NO - ERROR
     005713  000001 1040 11          1586 ALOC1    CMPX4    1,1                IS THERE ANY MORE TO MOVE
     005714  005722 6000 00          1587          TZE      ALOC2              TRANSFER IF DONE MOVING
     005715  000000 2360 13          1588          LDQ      0,3                FETCH FROM OLD TABLE LOCATION
     005716  000000 7560 14          1589          STQ      0,4                STORE IN NEW TABLE LOCATION
     005717  000001 1630 03          1590          SBX3     1,DU               DECREMENT SOURCE POINTER
 END OF BINARY CARD 00000100
     005720  000001 1640 03          1591          SBX4     1,DU               DECREMENT DESTINATION POINTER
     005721  005713 7100 00          1592          TRA      ALOC1              AND TRY TO MOVE SOME MORE
     005722  005677 2230 00          1593 ALOC2    LDX3     AINC               GET DISPLACEMENT IN XR - 3
     005723  000001 6240 11          1594          EAX4     1,1                GET POINTER TO CONTROL WORD OF FIRST MOVED TABLE
     005724  000000 0430 14          1595 ALOC3    ASX3     0,4                MODIFY DESCRIPTOR TO POINT TO NEW TABLE LOCATION
     005725  000001 0640 03          1596          ADX4     1,DU               STEP TO NEXT DESCRIPTOR
     005726  035233 1040 03          1597          CMPX4    T$END,DU           ARE WE DONE INCREMENTING DESCRIPTORS
     005727  005724 6040 00          1598          TMI      ALOC3              TRANSFER IF MORE TO DO
     005730  005724 6000 00          1599          TZE      ALOC3              TRANSFER IF MORE TO DO
     005731  000000 7220 11          1600 ALOC4    LXL2     0,1                GET LENGTH OF TABLE
     005732  000000 0620 11          1601          ADX2     0,1                ADD CURRENT LOCATION
     005733  000000 4500 12          1602          STZ      0,2                ZERO OUT LINK WORD OF FOLLOWING ENTRY
     005734  006031 1620 00          1603          SBX2     TEMP               SUBTRACT LENGTH OF NEW ELEMENT FOR ITS LOCATION
     005735  777777 7550 12          1604          STA      -1,2               STORE LENGTH OF NEW ENTRY IN PRECEEDING WORD
     005736  006010 4420 00          1605          SXL2     REG                PUT LOCATION OF NEW ELEMENT IN XR - 1
     005737  006010 0730 00          1606          LREG     REG                RESTORE REGISTERS
     005740  000000 7100 10          1607          TRA      0,0                AND RETURN
     005741  000000011007
                       005742        1608          EVEN
     005742  000001 6070 04          1609 SOVF     TTF      1,IC               CONTINUE IF THERE IS NO STACK OVERFLOW
     005743  005744 7170 00          1610          XED      *+1                CONTINUE XED CHAIN
     005744  006030 5540 00          1611          STC1     OVFIC              SAVE RETURN ADDRESS
     005745  005746 7100 00          1612          TRA      *+1                AND BREAK XED
 END OF BINARY CARD 00000101
     005746  006020 7530 00          1613          SREG     OVFR               SAVE REGISTERS
     005747  000100 2350 07          1614          LDA      =0100,DL           GET A ONE IN THE TALLY FIELD
     005750  035235 0550 00          1615          ASA      A$STACK            AND ANTICIPATE INCREASE IN STACK LENGTH
     005751  035215 2210 03          1616          LDX1     T$STACK,DU         GET POINTER TO CONTROL STACK CONTROL WORD
     005752  005765 7100 00          1617          TRA      OVF                AND INCREASE LENGTH OF CONTROL STACK
     005753  000000011007
                       005754        1618          EVEN
     005754  000001 6070 04          1619 WOVF     TTF      1,IC               CONTINUE IF THERE IS NO STACK OVERFLOW
     005755  005756 7170 00          1620          XED      *+1                CONTINUE XED CHAIN
     005756  006030 5540 00          1621          STC1     OVFIC              SAVE RETURN ADDRESS
     005757  005760 7100 00          1622          TRA      *+1                AND BREAK XED
     005760  006020 7530 00          1623          SREG     OVFR               SAVE REGISTERS
     005761  000100 2350 07          1624          LDA      =0100,DL           GET A ONE IN THE TALLY FIELD
     005762  035234 0550 00          1625          ASA      A$WORK             AND ANTICIPATE INCREASE IN STACK LENGTH
     005763  035214 2210 03          1626          LDX1     T$WORK,DU          GET POINTER TO WORKING STACK CONTROL WORD
     005764  005765 7100 00          1627          TRA      OVF                AND INCREASE LENGTH OF WORKING STACK
     005765  035214 3200 00          1628 OVF      LCX0     T$WORK             GET MINUS BASE OF WORKING STACK
     005766  035234 0400 00          1629          ASX0     A$WORK             AND MAKE WORKING STACK POINTER RELATIVE
     005767  035215 3200 00          1630          LCX0     T$STACK            GET MINUS BASE OF CONTROL STACK
```

```
           T                                    PASS0

      005770  035235 0400 00       1631          ASX0     A$STACK           AND MAKE CONTROL STACK POINTER RELATIVE
      005771  000000 6350 00       1632          EAA      0                 GET NUMBER OF WORDS TO REQUEST (-1)
      005772  005663 7000 00       1633          TSX0     T$ALOC            AND ALLOCATE ONE LINK WORD
      005773  035214 2200 00       1634          LDX0     T$WORK            GET BASE OF WORKING STACK
   END OF BINARY CARD 00000102
      005774  035234 0400 00       1635          ASX0     A$WORK            AND MAKE WORKING STACK POINTER ABSOLUTE
      005775  035215 2200 00       1636          LDX0     T$STACK           GET BASE OF CONTROL STACK
      005776  035235 0400 00       1637          ASX0     A$STACK           AND MAKE CONTROL STACK POINTER ABSOLUTE
      005777  006020 0730 00       1638          LREG     OVFR              RESTORE REGISTERS
      006000  006030 6300 00       1639          RET      OVFIC             AND RETURN
      006001  000007710004
                        006010     1640          EIGHT
      006010  000000000000         1641 REG      OCT      0,0,0,0,0,0,0,0
      006011  000000000000
      006012  000000000000
      006013  000000000000
      006014  000000000000
      006015  000000000000
      006016  000000000000
      006017  000000000000
      006020  000000000000         1642 OVFR     OCT      0,0,0,0,0,0,0,0
      006021  000000000000
      006022  000000000000
      006023  000000000000
      006024  000000000000
      006025  000000000000
      006026  000000000000
   END OF BINARY CARD 00000103
      006027  000000000000
      006030  000000 000000        1643 OVFIC    ZERO
      006031  000000 000000        1644 TEMP     ZERO
                                   1645          HEAD     A
      006032  000000 000000        1646 INIT     ZERO                       PAUSE FOR PATCHES
      006033  000003 6360 00       1647          EAQ      3                 FILE REFERENCE NUMBER
      006034  000000 2350 07       1648          LDA      0,DL              DESIRED POINTER SETTING
      006035  003767 7000 00       1649          TSX0     $SETP             AND RESET POINTER IN SOURCE FILE
      006036  001677 4500 00       1650          STZ      EOF               RESET END OF FILE FLAG
      006037  001674 2350 00       1651          LDA      INI               GET INITIAL INPUT TALLY
      006040  001675 7550 00       1652          STA      IN                AND INITIALIZE INPUT ROUTINE
      006041  030407 6350 00       1653          EAA      1$PROG+1          GET POINTER TO 'PROG' SYNTAX
      006042  006207 7100 00       1654          TRA      OK-1              AND START
                        006043     1655          BSS      100               PATCH SPACE
                                   1656 * THE FOLLOWING ROUTINE EVALUATES THE SYNTAX OF PASS I AND PASS II
                                   1657 *
                                   1658 *
                                   1659          HEAD     A
                                   1660          CRSM     OFF
      006207  006411 7550 00       1661          STA      S                 SAVE POINTER TO CURRENT ALTERNATIVE IN S
      006210  006411 2350 56       1662 OK       LDA      S,ID              GET NEXT THING TO MATCH IN ALTERNATIVE
      006211  006367 6040 00       1663          TMI      TRACE             IF MINUS - TRANSFER TO ACTION
```

```
          A                       PASS0

     006212  006411 2360 00      1664          LDQ      S                   GET CURRENT PLACE IN ALTERNATIVE
     006213  035235 7560 56      1665          STQ      STACK,ID            SAVE IN STACK
     006214  005742 7170 00      1666          XED      T$SOVF              CHECK FOR STACK OVERFLOW
     006215  006207 7100 00      1667          TRA      OK-1                AND CONTINUE
     006216  006411 2350 00      1668 NOMAT    LDA      S                   GET PLACE IN ALTERNATIVE OF FAILURE
     006217  777700 3150 07      1669          CANA     =0777700,DL         SEE IF AT BEGINNING OF ALTERNATIVE
  END OF BINARY CARD 00000104
     006220  006233 6010 00      1670          TNZ      ERRS                SYNTACTIC ERROR - GO PRINT ERROR MESSAGE
     006221  777776 2350 01      1671          LDA      -2,AU               GET POINTER TO NEXT ALTERNATIVE
     006222  006207 7100 00      1672          TRA      OK-1                AND TRY NEXT ALTERNATIVE
     006223  035235 2350 54      1673 FAIL     LDA      STACK,DI            REPORT FAILURE TO NEXT HIGHER LEVEL
     006224  006411 7550 00      1674          STA      S                   SAVE PLACE OF REPORTED FAILURE
     006225  006216 7100 00      1675          TRA      NOMAT               AND TRY AT HIGHER LEVEL
     006226  035235 2350 54      1676 END      LDA      STACK,DI            GET PLACE IN HIGHER LEVEL TO REPORT SUCCESS
     006227  006411 7550 00      1677          STA      S                   SAVE AS CURRENT LOCATION IN SYNTAX
     006230  006207 7100 00      1678          TRA      OK-1                AND CONTINUE
     006231  000000 2350 05      1679 ERR      LDA      0,AL                GET PARAMETER FOR ERROR ROUTINE
     006232  006562 7000 00      1680          TSX0     1$XXX               PRINT ERROR MESSAGE
     006233  006257 2350 00      1681 ERRS     LDA      ERRS1               GET TALLY WORD OF MESSAGE
     006234  006562 7000 00      1682          TSX0     1$XXX               AND PRINT IT
     006235  006436 2200 00      1683          LDX0     MATT                GET LAST MATCH ATTEMPTED
     006236  035222 0600 00      1684          ADX0     T$STAB              MAKE POINTER ABSOLUTE
     006237  000000 2350 10      1685          LDA      SC,0                GET STRING POINTER TO SYMBOL
     006240  006554 7000 00      1686          TSX0     1$XXS               AND PRINT IT
     006241  006267 2350 00      1687 ERRF     LDA      ERRS2               GET TALLY WORD OF SECOND PART OF MESSAGE
     006242  006562 7000 00      1688          TSX0     1$XXX               AND PRINT IT
     006243  000024 3350 07      1689          LCA      20,DL               GET A COUNT OF TWENTY
     006244  006256 7550 00      1690          STA      ERRC                AND STORE
     006245  005164 7000 00      1691 ERR1     TSX0     A$PEEK              GET NEXT INPUT SYMBOL
  END OF BINARY CARD 00000105
     006246  006255 7100 00      1692          TRA      ERR2                NO MORE SO STOP
     006247  002152 4500 00      1693          STZ      A$PEEKF             ACCEPT SYMBOL
     006250  035222 0670 00      1694          ADX7     T$STAB              MAKE SYMBOL POINTER ABSOLUTE
     006251  000000 2350 17      1695          LDA      0,7                 GET TALLY WORD OF SYMBOL
     006252  006554 7000 00      1696          TSX0     1$XXS               AND PRINT IT OUT
     006253  006256 0540 00      1697          AOS      ERRC                DECREMENT COUNT
     006254  006245 6040 00      1698          TMI      ERR1                AND LOOP TEN TIMES
     006255  005653 7100 00      1699 ERR2     TRA      P$STOP              HALT AFTER PRINTING ERROR MESSAGE
     006256  000000 000000       1700 ERRC     ZERO
     006257  006260 0035 40      1701 ERRS1    TALLYB   *+1,28+1
     006260  101116040101        1702          UASCI    7,AN ACCEPTABLE NEXT SYMBOL IS
     006261  103103105120
     006262  124101102114
     006263  105040116105
     006264  130124040123
     006265  131115102117
     006266  114040111123
     006267  006270 0021 40      1703 ERRS2    TALLYB   *+1,16+1
     006270  116105130124        1704          UASCI    4,NEXT SYMBOLS ARE
     006271  040123131115
```

```
             A                                 PASS0

     006272  102117114123
     006273  040101122105
 END OF BINARY CARD 00000106
     006274  006275 0041 40          1705 ER1     TALLYB   *+1,32+1
     006275  111114114105            1706         UASCI    7,ILLEGAL SUBSCRIPT CONSTRUCTION
     006276  107101114040
     006277  123125102123
     006300  103122111120
     006301  124040103117
     006302  116123124122
     006303  125103124111
     006304  117116015012            1707         OCT      117116015012
     006305  006306 0017 40          1708 ER2     TALLYB   *+1,14+1
     006306  111114114105            1709         UASCI    3,ILLEGAL MODE
     006307  107101114040
     006310  115117104105
     006311  015012000000            1710         OCT      015012000000
     006312  006313 0040 40          1711 ER3     TALLYB   *+1,31+1
     006313  111114114105            1712         UASCI    7,ILLEGAL OPERATION DECLARATION
     006314  107101114040
     006315  117120105122
     006316  101124111117
     006317  116040104105
     006320  103114101122
     006321  101124111117
 END OF BINARY CARD 00000107
     006322  116015012000            1713         OCT      116015012000
     006323  006324 0033 40          1714 ER4     TALLYB   *+1,26+1
     006324  111114114105            1715         UASCI    6,ILLEGAL MODE DECLARATION
     006325  107101114040
     006326  115117104105
     006327  040104105103
     006330  114101122101
     006331  124111117116
     006332  015012000000            1716         OCT      015012000000
     006333  006334 0040 40          1717 ER5     TALLYB   *+1,31+1
     006334  111114114105            1718         UASCI    7,ILLEGAL STRUCTURE DECLARATION
     006335  107101114040
     006336  123124122125
     006337  103124125122
     006340  105040104105
     006341  103114101122
     006342  101124111117
     006343  116015012000            1719         OCT      116015012000
     006344  006345 0034 40          1720 ER6     TALLYB   *+1,27+1
     006345  111114114105            1721         UASCI    6,ILLEGAL UNION DECLARATION
     006346  107101114040
     006347  125116111117
 END OF BINARY CARD 00000108
     006350  116040104105
```

```
          A                                  PASS0

     006351  103114101122
     006352  101124111117
     006353  116015012000         1722         OCT      116015012000
     006354  006355 0020 40       1723 ER7     TALLYB   *+1,15+1
     006355  111114114105         1724         UASCI    3,ILLEGAL FIELD
     006356  107101114040
     006357  106111105114
     006360  104015012000         1725         OCT      104015012000
     006361  006362 0024 40       1726 ER8     TALLYB   *+1,19+1
     006362  111114114105         1727         UASCI    4,ILLEGAL IF CLAUSE
     006363  107101114040
     006364  111106040103
     006365  114101125123
     006366  105015012000         1728         OCT      105015012000
     006367  006377 2210 03       1729 TRACE   LDX1     TRACT,DU        GET POINTER TO INHIBIT TRACE TABLE
     006370  400000 6220 01       1730         EAX2     131072,AU       GET ADDRESS OF NEXT ACTION IN XR - 2
     006371  024300 5202 01       1731         RPT      10,1,TZE        SEE IF ACTION IS IN TABLE
     006372  000000 1020 11       1732         CMPX2    0,1             COMPARE NEXT ACTION WITH TABLE
     006373  006375 6000 00       1733         TZE      TRAC1           TRANSFER IF ACTION IS IN TABLE
     006374  006540 0110 00       1734         NOP      1$XYZ           PRINT OUT TRACE
     006375  400000 7000 01       1735 TRAC1   TSX0     131072,AU       AND TRANSFER TO ACTION
   END OF BINARY CARD 00000109
     006376  006210 7100 00       1736         TRA      A$OK            CONTINUE TO ANALYZE SYNTAX
     006377  006226 000000        1737 TRACT   ZERO     END
     006400  006223 000000        1738         ZERO     FAIL
     006401  006417 000000        1739         ZERO     MATCH
     006402  006537 000000        1740         ZERO     1$EMPTY
     006403  007215 000000        1741         ZERO     1$IDENT
     006404  000000 000000        1742         ZERO
     006405  000000 000000        1743         ZERO
     006406  000000 000000        1744         ZERO
     006407  000000 000000        1745         ZERO
     006410  000000 000000        1746         ZERO
     006411  000000 000000        1747 S       ZERO
                                  1748 *
                                  1749 *
                                  1750 *
                                  1751 SYNTAX  MACRO
                                  1752 SETE    SET      SETE-1
                                  1753         IFE      SETE,0,25
                                  1754         IFE      '#1','',3
                                  1755         TALLY    *1,*1
                                  1756         ZERO     A$FAIL+131072
                                  1757 SETE    SET      1
                                  1758         INE      '#1','',49
                                  1759 SET     SET      0
                                  1760 SETC    SET      0
                                  1761 SETT    SET      1
                                  1762         S1       (#1)
                                  1763         S1       (#2)
```

```
                 S1       (#3)
                 S1       (#4)
                 IFE      SETT,0
                 INE      SET,0,4
                 'ERROR   #1
                 'ERROR   #2
                 'ERROR   #3
                 'ERROR   #4
                 TALLY    *+SETC+3,1
         SETT    SET      1
                 S2       (#1)
                 S2       (#2)
                 S2       (#3)
                 S2       (#4)
                 ZERO     A$END+131072
                 SYNTAX   (#02),
                 ETC      (#03),
                 ETC      (#04),
                 ETC      (#05),
                 ETC      (#06),
                 ETC      (#07),
                 ETC      (#08),
                 ETC      (#09),
                 ETC      (#10),
                 ETC      (#11),
                 ETC      (#12),
                 ETC      (#13),
                 ETC      (#14),
                 ETC      (#15),
                 ETC      (#16),
                 ETC      (#17),
                 ETC      (#18),
                 ETC      (#19),
                 ETC      (#20),
                 ETC      (#21),
                 ETC      (#22),
                 ETC      (#23),
                 ETC      (#24),
                 ETC      (#25),
                 ETC      (#26),
                 ETC      (#27),
                 ETC      (#28),
                 ETC      (#29),
                 ETC      (#30)
                 ENDM     SYNTAX
         S1      MACRO
                 INE      SETT,0,12
         SETT    SET      0
                 IDRP     #1
                 IFE      '#1','*'
```

```
            A                                  PASS0

                                       1814 SETT    SET       1
                                       1815         INE       '#1','*',6
                                       1816         INE       SET,0,2
                                       1817 SET     SET       0
                                       1818         IFE       1,2,3
                                       1819 SETC    SET       SETC+1
                                       1820         IFG       '#1','1$0000'
                                       1821 SET     SET       1
                                       1822         IDRP
                                       1823         ENDM      S1
                                       1824 S2      MACRO
                                       1825         INE       SETT,0,15
                                       1826 SETT    SET       0
                                       1827         IDRP      #1
                                       1828         IFE       '#1','*'
                                       1829 SETT    SET       1
                                       1830         INE       '#1','*',8
                                       1831         INE       SET,0,3
                                       1832         ZERO      131072+RELZER+SET,#1
                                       1833 SET     SET       0
                                       1834         IFE       1,2,4
                                       1835         IFG       '#1','1$0000'
                                       1836 SET     SET       #1-RELZER
                                       1837         IFL       '#1','1$0000'
                                       1838         TALLY     #1+1,1
                                       1839         IDRP
                                       1840 SETE    SET       SETE+1
                                       1841         ENDM      S2
                                       1842         HEAD      D
     006412  000000 000000             1843 MODE    ZERO
     006413  000000 000000             1844 OP      ZERO
     006414  000000 000000             1845 PRIOR   ZERO
     006415  000000 000000             1846 IDENT   ZERO
     006416  000000 000000             1847 DENOT   ZERO
                                       1848         HEAD      A
     006417  000000 6200 05            1849 MATCH   EAX0      0,AL              GET REQUIRED SYMBOL IN XR - 0
     006420  006436 7400 00            1850         STX0      MATT              SAVE FOR COMPARE
     006421  035222 0600 00            1851         ADX0      T$STAB            MAKE IT ABSOLUTE
     006422  000000 2350 10            1852         LDA       SC,0              GET TALLY FOR SYMBOL ATTEMPTING TO MATCH WITH
     006423  006554 0110 00            1853         NOP       1$XXS             PRINT
   END OF BINARY CARD 00000110
     006424  005164 7000 00            1854         TSX0      PEEK              PEEK AT NEXT INPUT SYMBOL
     006425  006216 7100 00            1855         TRA       NOMAT             NO MORE INPUT SYMBOLS SO FAILURE
     006426  006436 1070 00            1856         CMPX7     MATT              SEE IF REQUIRED INPUT SYMBOL
     006427  006216 6010 00            1857         TNZ       NOMAT             NO - GIVE BAD RETURN
     006430  002152 4500 00            1858         STZ       PEEKF             ACCEPT INPUT SYMBOL
     006431  000000 6200 17            1859         EAX0      0,7               GET POINTER IN XR - 0
     006432  035222 0600 00            1860         ADX0      T$STAB            MAKE POINTER ABSOLUTE
     006433  000000 2350 10            1861         LDA       SC,0              GET TALLY FOR SYMBOL SUCCESSFULLY MATCHED
     006434  006554 0110 00            1862         NOP       1$XXS             AND PRINT IT OUT
```

```
                A                                      PASS0

          006435  006210 7100 00          1863           TRA      A$OK              YES - GIVE OK RETURN
          006436  000000 000000           1864 MATT      ZERO
          006437  006461 4500 00          1865 TLU       STZ      DLL               INITIALIZE DELTA LL
          006440  006457 7410 00          1866 TLU0      STX1     LAST              SAVE LOCATION OF PRECEEDING POINTER
          006441  000000 2210 11          1867           LDX1     0,1               GET RELATIVE POINTER TO TABLE ENTRY
          006442  000000 6040 10          1868           TMI      0,0               NO MORE ENTRIES - FAIL RETURN
          006443  035220 0610 00          1869           ADX1     T$DEF             MAKE TABLE ENTRY POINTER ABSOLUTE
          006444  000000 7220 11          1870           LXL2     0,1               GET LEVEL OF TABLE ENTRY IN XR - 2
          006445  006460 1020 00          1871 TLU1      CMPX2    LEVEL             COMPARE WITH LEVEL CURRENTLY BEING SEARCHED
          006446  006451 6020 00          1872           TNC      TLU2              TRANSFER IF MIGHT BE CORRECT SURROUNDING LEVEL
          006447  000001 6000 10          1873           TZE      1,0               TRANSFER IF ENTRY OF PROPER LEVEL FOUND
          006450  006440 7100 00          1874           TRA      TLU0              TRANSFER IF MUST SKIP TO NEXT ENTRY
          006451  006460 2360 00          1875 TLU2      LDQ      LEVEL             GET LEVEL CURRENTLY BEING SEARCHED FOR
      END OF BINARY CARD 00000111
          006452  035221 0760 00          1876           ADQ      T$PROG            MAKE INTO ABSOLUTE POINTER TO PROG TABLE
          006453  000002 2230 02          1877           LDX3     2,QU              GET SURROUNDING LEVEL IN XR - 3
          006454  006460 7430 00          1878           STX3     LEVEL             AND MAKE IT NEW SEARCH LEVEL
          006455  006461 0540 00          1879           AOS      DLL               INCREMENT NUMBER OF RANGES EXITED
          006456  006445 7100 00          1880           TRA      TLU1              AND CHECK CURRENT ENTRY AGAINST NEW LEVEL
          006457  000000 000000           1881 LAST      ZERO
          006460  000000 000000           1882 LEVEL     ZERO
          006461  000000 000000           1883 DLL       ZERO
          006462  006467 7400 00          1884 GLL       STX0     GLLX              SAVE RETURN
          006463  006461 7200 00          1885           LXL0     DLL               GET DELTA LL IN XR - 0
          006464  400000 2350 03          1886           LDA      =0400000,DU       GET SIGN BIT SET IN A
          006465  000000 7710 10          1887           ARL      0,0               SHIFT RIGHT NUMBER OF LEVELS OUT
          006466  777777 6000 00          1888           TZE      $ERROR            ERROR BECAUSE TOO DEEPLY NESTED CONSTRUCTION
          006467  000000 7100 00          1889 GLLX      TRA      **                AND RETURN
          006470  006510 7400 00          1890 BCD       STX0     BCDX              SAVE RETURN
          006471  000000 2210 03          1891           LDX1     0,DU              INITIALIZE XR - 1 FOR REPEAT
          006472  000000 2220 03          1892           LDX2     0,DU              INITIALIZE XR - 2 FOR REPEAT
          006473  010600 5602 01          1893           RPDB     4,1               CONVERT FOUR DIGITS
          006474  000003 7360 11          1894           QLS      3,1               INSERT 3 BITS IN Q FOR 9 BIT CHARACTERS
          006475  006511 5050 12          1895           BCD      BCDT,2            CONVERT ANOTHER DIGIT
          006476  006515 7560 00          1896           STQ      BCDTP             AND SAVE ANSWER
          006477  000000 4110 03          1897           LDE      0,DU              INITIALIZE E REGISTER
      END OF BINARY CARD 00000112
          006500  000044 7370 00          1898           LLS      36                MOVE NUMBER INTO A REGISTER AND CLEAR Q
          006501  000000 5730 00          1899           FNO                        SEARCH FOR LEADING SIGNIFICANT DIGIT
          006502  400000 2350 03          1900           LDA      =0400008,DU       GET A SIGN BIT IN A
          006503  756000 4350 03          1901           UFA      =-9B25,DU         PROPAGATE SIGN ACROSS LEADING ZEROS
          006504  006516 3750 00          1902           ANA      SUPC              GET PROPER NUMBER TO SUPRESS ZEROS
          006505  000000 5310 00          1903           NEG                        MUST SUBTRACT IT FROM ZERO
          006506  006517 0750 00          1904           ADA      ASCZ              ADD 4 ASCII ZEROS
          006507  006515 0750 00          1905           ADA      BCDTP             ADD CONVERTED NUMBER
          006510  000000 7100 00          1906 BCDX      TRA      **                RETURN
          006511  000000017500            1907 BCDT      DEC      8000,6400,5120,4096
          006512  000000014400
          006513  000000012000
          006514  000000010000
          A                          PASS0

     006515  000000 000000        BCBTP   ZERO
     006516  020020020020         SUPC    OCT      020020020020
     006517  060060060060         ASCZ    OCT      060060060060
                                          HEAD     1
     006520  000000 000000        IDNT    ZERO
     006521  000000 000000        FMODE   ZERO
     006522  000000 000000        PDPOS   ZERO
     006523  000000 000000        MK      ZERO
     006524  000000 000000        ETH     ZERO
     006525  000000 000000        FLX     ZERO
  END OF BINARY CARD 00000113
     006526  000000 000000        FIX     ZERO
     006527  000000 000000        BOXR    ZERO
     006530  000000 000000        BARR    ZERO
     006531  000000 000000        CLOR    ZERO
     006532  000000 000000        COMAC   ZERO
     006533  000000 000000        CLNC    ZERO
     006534  000000 000000        SEMIC   ZERO
     006535  000000 000000        LEVEL   ZERO
     006536  000000 000000        CURR    ZERO
     006537  006210 7100 00       EMPTY   TRA      A$OK             DO NOTHING BUT GIVE OK RETURN
     006540  006603 7400 00       XYZ     STX0     XRET             SAVE RETURN
     006541  006612 7570 00               STAQ     XAQ              SAVE AQ REGISTER
     006542  006610 2350 00               LDA      XXT              GET INITIALIZED OUTPUT TALLY WORD
     006543  006607 7550 00               STA      XX2              AND STORE
     006544  006612 2350 00               LDA      XAQ              RESTORE A REGISTER
     006545  000014 2200 03               LDX0     12,DU            GET NUMBER OF DIGITS TO CONVERT IN XR - 0
     006546  000006 2360 07       XYZ1    LDQ      6,DL             GET ASCII DIGIT ZERO SHIFTED RIGHT
     006547  000003 7770 00               LLR      3                GET NEXT OCTAL DIGIT IN QL
     006550  006607 7560 52               STQ      XX2,SC           AND STORE IN OUTPUT BUFFER
     006551  000001 1600 03               SBX0     1,DU             DECREMENT NUMBER OF DIGITS LEFT TO CONVERT
     006552  006546 6010 00               TNZ      XYZ1             TRANSFER IF MORE TO DO
     006553  006571 7100 00               TRA      XXL3             ALL DONE SO PRINT NUMBER
  END OF BINARY CARD 00000114
     006554  006603 7400 00       XXS     STX0     XRET             SAVE RETURN
     006555  006612 7570 00               STAQ     XAQ              SAVE AQ REGISTER
     006556  006606 7550 00               STA      XX1              STORE SC TALLY WORD
     006557  035223 2200 00               LDX0     T$ITAB           GET RELOCATION FOR TALLY WORD
     006560  006606 0400 00               ASX0     XX1              MAKE TALLY WORD ABSOLUTE
     006561  006565 7100 00               TRA      XXL              GO TO PRINT ROUTINE
     006562  006603 7400 00       XXX     STX0     XRET             SAVE RETURN
     006563  006612 7570 00               STAQ     XAQ              SAVE AQ REGISTER
     006564  006606 7550 00               STA      XX1              STORE SC TALLY WORD
     006565  006610 2350 00       XXL     LDA      XXT              GET INITIALIZED OUTPUT TALLY WORD
     006566  006607 7550 00               STA      XX2              AND STORE
     006567  006606 2350 52       XXL2    LDA      XX1,SC           GET NEXT CHARACTER TO OUTPUT
     006570  006604 6070 00               TTF      XXL1             TRANSFER IF THERE IS ONE
     006571  006607 2350 00       XXL3    LDA      XX2              GET FINAL TALLY WORD
     006572  006614 1750 03               SBA      XBUF,DU          SUBTRACT INITIAL VALUE OF ADDRESS OF TALLY WORD
     006573  000002 7330 00               LRS      2                MOVE CHARACTER POSITION TO Q
```

```
           1                          PASS0

     006574  000020 7310 00        1956          ARS      16                  CONCATENATE WITH NUMBER OF WORDS IN MESSAGE
     006575  000010 7370 00        1957          LLS      2+6                 GET NUMBER OF CHARACTERS IN TALLY FIELD
     006576  000100 0750 07        1958          ADA      =0100,DL            ADD ONE TO TALLY COUNT
     006577  006601 7510 06        1959          STCA     XXL4,06             STORE TALLY COUNT IN TALLY WORD
     006600  002623 7000 00        1960          TSX0     $TTB                CALL BUFFERED TTY OUTPUT ROUTINE
     006601  006614 0000 40        1961 XXL4     TALLYB   XBUF,0,0
 END OF BINARY CARD 00000115
     006602  006612 2370 00        1962          LDAQ     XAQ                 RESTORE A AND Q REGISTERS
     006603  000000 7100 00        1963 XRET     TRA      **                  AND RETURN
     006604  006607 7550 52        1964 XXL1     STA      XX2,SC              STORE IN OUTPUT STRING
     006605  006567 7100 00        1965          TRA      XXL2                AND LOOP
     006606  000000 000000         1966 XX1      ZERO
     006607  000000 000000         1967 XX2      ZERO
     006610  006614 0000 42        1968 XXT      TALLYB   XBUF,0,2            START AFTER CR/LF
     006611  000000011007
                         006612    1969          EVEN
     006612  000000000000          1970 XAQ      OCT      0,0
     006613  000000000000
     006614  015012000000          1971 XBUF     OCT      015012000000
                         006615    1972          BSS      100
                         006761    1973          BSS      100                 PATCH SPACE
```

```
          1                                               PASS 1

                                  1974        TTLS                    PASS 1
     007125  000000 2200 03       1975 STZ    LDX0    0,DU            GET A HALF WORD ZERO
     007126  000000 7400 05       1976        STX0    0,AL            ZERO OUT DESIGNATED LOCATION
     007127  006210 7100 00       1977        TRA     A$OK            AND EXIT
     007130  400000 2200 03       1978 SMO    LDX0    =0400000,DU     GET A MINUS ONE
     007131  000000 7400 05       1979        STX0    0,AL            STORE IN DESIGNATED LOCATION
     007132  006210 7100 00       1980        TRA     A$OK            AND EXIT
     007133  000000 4500 05       1981 CLEAR  STZ     0,AL            ZERO OUT WORD SPECIFIED BY ARGUMENT
     007134  006210 7100 00       1982        TRA     A$OK            AND EXIT
     007135  006520 2200 00       1983 STIDB  LDX0    IDNT            IS THERE AN IDENTIFIER ALREADY STORED
     007136  007142 6000 00       1984        TZE     STID            NO - CAN STORE IDENTIFIER
  END OF BINARY CARD 00000116
     007137  400000 2200 03       1985        LDX0    =0400000,DU     GET A BAD IDENTIFIER
     007140  006520 7400 00       1986        STX0    IDNT            AND STORE AS IDENTIFIER
     007141  006210 7100 00       1987        TRA     A$OK            AND EXIT
     007142  006520 7470 00       1988 STID   STX7    IDNT            STORE LAST READ IDENTIFIER IN IDNT
     007143  006210 7100 00       1989        TRA     A$OK            AND EXIT
     007144  006520 2200 00       1990 DECLC  LDX0    IDNT            IS THERE AN IDENTIFIER STORED
     007145  006210 6040 00       1991        TMI     A$OK            NO - BAD IDENTIFIER - NO LABEL
     007146  006210 6000 00       1992        TZE     A$OK            NO IDENTIFIER SO NO LABEL
     007147  007676 7100 00       1993        TRA     DECL            GO DECLARE IDENTIFIER AS LABEL
     007150  035234 0110 54       1994 DEL    NOP     A$WORK,DI       DELETE TOP OF WORKING STACK
     007151  006210 7100 00       1995        TRA     A$OK            AND EXIT
     007152  000000 2360 05       1996 PUSH   LDQ     0,AL            FETCH INDICATED DATA
     007153  035235 7560 56       1997        STQ     A$STACK,ID      AND SAVE IN CONTROL STACK
     007154  005742 7170 00       1998        XED     T$SOVF          CHECK FOR STACK OVERFLOW
     007155  006210 7100 00       1999        TRA     A$OK            AND EXIT
     007156  035235 2360 54       2000 POP    LDQ     A$STACK,DI      FETCH TOP OF CONTROL STACK
     007157  000000 7560 05       2001        STQ     0,AL            AND STORE IN INDICATED LOCATION
     007160  006210 7100 00       2002        TRA     A$OK            AND EXIT
     007161  035235 0110 54       2003 DELS   NOP     A$STACK,DI      DELETE ONE WORD FROM THE CONTROL STACK
     007162  006210 7100 00       2004        TRA     A$OK            AND EXIT
     007163  035234 2200 00       2005 MARK   LDX0    A$WORK          GET CURRENT POSITION OF WORKING STACK POINTER
     007164  035214 1600 00       2006        SBX0    T$WORK          MAKE IT RELATIVE
  END OF BINARY CARD 00000117
     007165  035235 7400 56       2007        STX0    A$STACK,ID      AND STORE IN THE CONTROL STACK
     007166  005742 7170 00       2008        XED     T$SOVF          CHECK FOR STACK OVERFLOW
     007167  006210 7100 00       2009        TRA     A$OK            AND EXIT
     007170  035234 2200 00       2010 MRNGE  LDX0    A$WORK          GET CURRENT WORKING STACK POINTER
     007171  035214 1600 00       2011        SBX0    T$WORK          MAKE IT RELATIVE
     007172  006523 7400 00       2012        STX0    MK              AND SAVE AS MARK IN STACK
     007173  006210 7100 00       2013        TRA     A$OK            AND EXIT
     007174  000001 6350 00       2014 EVOID  EAA     M$VOID          GET A VOID MODE IN A
     007175  035234 7550 56       2015        STA     A$WORK,ID       STORE IT IN THE WORKING STACK
     007176  005754 7170 00       2016        XED     T$WOVF          CHECK FOR STACK OVERFLOW
     007177  006210 7100 00       2017        TRA     A$OK            AND EXIT
     007200  035234 2200 00       2018 SAVE   LDX0    A$WORK          FETCH WORKING STACK POINTER
     007201  777777 2360 10       2019        LDQ     -1,0            FETCH TOP OF WORKING STACK
     007202  000000 7560 05       2020        STQ     0,AL            AND STORE IN DESIGNATED LOCATION
     007203  006210 7100 00       2021        TRA     A$OK            AND EXIT
```

```
                 1                                        PASS 1

          007204   035234 2200 00         2022 REST    LDX0      A$WORK          FETCH WORKING STACK POINTER
          007205   000000 2360 05         2023         LDQ       0,AL            FETCH DESIGNATED LOCATION
          007206   035234 7560 56         2024         STQ       A$WORK,ID       AND STORE ON WORKING STACK
          007207   005754 7170 00         2025         XED       T$WOVF          CHECK FOR STACK OVERFLOW
          007210   006210 7100 00         2026         TRA       A$OK            AND EXIT
          007211   006520 2350 00         2027 TAG     LDA       IDNT            FETCH SAVED IDENTIFIER
          007212   035234 7550 56         2028         STA       A$WORK,ID       STORE ON WORKING STACK
     END OF BINARY CARD 00000118
          007213   005754 7170 00         2029         XED       T$WOVF          CHECK FOR STACK OVERFLOW
          007214   006210 7100 00         2030         TRA       A$OK            AND EXIT
          007215   005164 7000 00         2031 IDENT   TSX0      A$PEEK          PEEK AT NEXT INPUT SYMBOL
          007216   006216 7100 00         2032         TRA       A$NOMAT         NO MORE INPUT - FAILURE
          007217   000112 1070 03         2033         CMPX7     A$TABLE,DU      CAN SYMBOL BE AN IDENTIFIER
          007220   006216 6020 00         2034         TNC       A$NOMAT         TRANSFER IF IN PERMANENT PART OF TABLE
          007221   002152 4500 00         2035         STZ       A$PEEKF         ACCEPT SYMBOL
          007222   000000 6200 17         2036         EAX0      0,7             GET POINTER IN XR - 0
          007223   035222 0600 00         2037         ADX0      T$STAB          MAKE POINTER ABSOLUTE
          007224   000000 2350 10         2038         LDA       A$SC,0          GET TALLY FOR IDENTIFIER SUCCESSFULLY MATCHED
          007225   006554 0110 00         2039         NOP       1$XXS           PRINT
          007226   006210 7100 00         2040         TRA       A$OK            AND GIVE SUCCESSFUL RETURN
          007227   035234 2200 00         2041 CST     LDX0      A$WORK          GET CURRENT WORKING STACK POINTER
          007230   035214 1600 00         2042         SBX0      T$WORK          SUBTRACT BASE OF WORK TO GET LENGTH
          007231   035235 1600 54         2043         SBX0      A$STACK,DI      SUBTRACT LENGTH WHEN STACK WAS MARKED
          007232   000002 6350 10         2044         EAA       2,0             GET LENGTH + 2 TO A REGISTER
          007233   000001 7310 00         2045         ARS       1               GET NUMBER OF FIELDS + 1 IN A REGISTER
          007234   007303 7550 00         2046         STA       CSTA            AND SAVE LENGTH OF TABLE ENTRIES
          007235   035216 2210 03         2047         LDX1      T$MODE,DU       GET POINTER TO MODE TABLE CONTROL WORD
          007236   005663 7000 00         2048         TSX0      T$ALOC          AND ALLOCATE SPACE IN THE MODE TABLE
          007237   035216 1610 00         2049         SBX1      T$MODE          MAKE POINTER TO TABLE ENTRY RELATIVE
          007240   000000 6240 11         2050         EAX4      0,1             SAVE RELATIVE POINTER TO MODE TABLE ENTRY
     END OF BINARY CARD 00000119
          007241   007304 7410 00         2051         STX1      CSTM            AND SAVE
          007242   007303 2350 00         2052         LDA       CSTA            GET LENGTH OF BOUND TABLE ENTRY
          007243   035217 2210 03         2053         LDX1      T$BOUND,DU      GET POINTER TO BOUND TABLE CONTROL WORD
          007244   005663 7000 00         2054         TSX0      T$ALOC          AND ALLOCATE SPACE IN THE BOUND TABLE
          007245   000000 6230 11         2055         EAX3      0,1             SAVE POINTER TO BOUND TABLE ENTRY IN XR - 3
          007246   035217 1630 00         2056         SBX3      T$BOUND         AND MAKE IT RELATIVE
          007247   007305 7410 00         2057         STX1      CSTB            AND SAVE POINTER TO BOUND TABLE ENTRY
          007250   035216 2220 00         2058         LDX2      T$MODE          GET ADDRESS OF BASE OF MODE TABLE
          007251   007304 0420 00         2059         ASX2      CSTM            AND MAKE MODE TABLE ENTRY POINTER ABSOLUTE
          007252   007303 2350 00         2060         LDA       CSTA            GET LENGTH OF TABLE ENTRIES
          007253   000014 7310 00         2061         ARS       12              POSITION FOR TALLY IN ID WORD
          007254   000000 6200 05         2062         EAX0      0,AL            PUT IN HALF WORD REGISTER
          007255   007305 4400 00         2063         SXL0      CSTB            AND STORE TALLY IN BOUND ENTRY POINTER
          007256   016757 6350 00         2064         EAA       M$STRCT         GET HEADER FOR MODE TABLE ENTRY
          007257   007304 7550 56         2065         STA       CSTM,ID         AND STORE IN MODE TABLE ENTRY
          007260   016757 6350 00         2066         EAA       B$STRCT         GET HEADER FOR BOUND TABLE ENTRY
          007261   007305 7550 56         2067         STA       CSTB,ID         AND STORE IN BOUND TABLE ENTRY
          007262   035235 2260 51         2068         LDX6      A$STACK,I       GET RELATIVE POINTER TO WORK WHEN MARKED
          007263   035214 0660 00         2069         ADX6      T$WORK          AND MAKE IT ABSOLUTE
```

```
           1                                         PASS 1

      007264  007306 7460 00       2070        STX6     CSTT             AND SAVE AS TALLY WORD
      007265  007306 2350 56       2071 CST1   LDA      CSTT,ID          GET NEXT DECLARER FROM WORK
      007266  007306 2200 56       2072        LDX0     CSTT,ID          GET CORESPONDING TAG FROM WORK
   END OF BINARY CARD 00000120
      007267  007304 7550 51       2073        STA      CSTM,I           STORE MODE IN MODE TABLE
      007270  007304 4400 56       2074        SXL0     CSTM,ID          AND STORE CORESPONDING TAG IN MODE TABLE
      007271  000000 6350 05       2075        EAA      0,AL             GET BOUND INFORMATION IN AU
      007272  007305 7550 56       2076        STA      CSTB,ID          AND STORE IN BOUND TABLE
      007273  007265 6070 00       2077        TTF      CST1             TRANSFER IF MORE FIELDS TO PROCESS
      007274  035234 0110 54       2078        NOP      A$WORK,DI        DELETE A WORD FROM THE WORKING STACK
      007275  035234 1060 00       2079        CMPX6    A$WORK           IS STACK DELETED BACK TO THE MARK
      007276  007274 6010 00       2080        TNZ      *-2              TRANSFER IF MORE TO REMOVE
      007277  035234 7440 51       2081        STX4     A$WORK,I         AND STORE STRUCTURE MODE
      007300  035234 4430 56       2082        SXL3     A$WORK,ID        AND STORE STRUCTURE BOUND
      007301  005754 7170 00       2083        XED      T$WOVF           CHECK FOR STACK OVERFLOW
      007302  006210 7100 00       2084        TRA      A$OK             AND EXIT
      007303  000000 000000        2085 CSTA   ZERO
      007304  000000 000000        2086 CSTM   ZERO
      007305  000000 000000        2087 CSTB   ZERO
      007306  000000 000000        2088 CSTT   ZERO
      007307  000000 6200 17       2089 CNVRT  EAX0     0,7              GET POINTER IN XR - 0
      007310  035222 0600 00       2090        ADX0     T$STAB           MAKE POINTER ABSOLUTE
      007311  000000 2350 10       2091        LDA      A$SC,0           GET STRING TALLY WORD OF NUMBER
      007312  007331 7550 00       2092        STA      CNVA             AND SAVE A COPY
      007313  035223 2200 00       2093        LDX0     T$ITAB           GET BASE OF IDENTIFIER TABLE
      007314  007331 0400 00       2094        ASX0     CNVA             AND MAKE STRING TALLY WORD ABSOLUTE
   END OF BINARY CARD 00000121
      007315  007331 2350 52       2095        LDA      CNVA,SC          GET FIRST CHARACTER OF STRING
      007316  000060 1750 07       2096        SBA      =0060,DL         SUBTRACT AN ASCII ZERO
      007317  777777 6040 00       2097        TMI      $ERROR           ERROR - NOT A DIGIT
      007320  777777 6000 00       2098        TZE      $ERROR           ERROR - ZERO IS ILLEGAL
      007321  000012 1150 07       2099        CMPA     10,DL            SEE IF FIRST CHARACTER IS A DIGIT
      007322  777777 6050 00       2100        TPL      $ERROR           ERROR - NOT A DIGIT
      007323  000000 6200 05       2101        EAX0     0,AL             PUT VALUE OF DIGIT IN XR - 0
      007324  035234 7400 56       2102        STX0     A$WORK,ID        AND STORE RESULT
      007325  005754 7170 00       2103        XED      T$WOVF           CHECK FOR STACK OVERFLOW
      007326  007331 0110 52       2104        NOP      CNVA,SC          SEE IF A SINGLE DIGIT IN STRING
      007327  777777 6070 00       2105        TTF      $ERROR           ERROR - TOO MANY CHARACTERS IN STRING
      007330  006210 7100 00       2106        TRA      A$OK             AND EXIT
      007331  000000 000000        2107 CNVA   ZERO
      007332  000002 2350 03       2108 EREF   LDA      2,DU             GET LENGTH OF ENTRY IN MODE TABLE
      007333  035216 2210 03       2109        LDX1     T$MODE,DU        GET POINTER TO MODE TABLE CONTROL WORD
      007334  005663 7000 00       2110        TSX0     T$ALOC           AND ALLOCATE MEMORY IN MODE TABLE
      007335  016762 6350 00       2111        EAA      M$REF            GET HEADER WORD FOR MODE TABLE ENTRY
      007336  000000 7550 11       2112        STA      0,1              AND STORE IN MODE TABLE ENTRY
      007337  035234 2350 54       2113        LDA      A$WORK,DI        GET MODE THAT IS TO BE REFERENCED
      007340  000000 6350 01       2114        EAA      0,AU             CLEAN OFF BOUND INFORMATION
      007341  000001 7550 11       2115        STA      1,1              AND STORE IN MODE TABLE ENTRY
      007342  035216 1610 00       2116        SBX1     T$MODE           MAKE MODE TABLE ENTRY RELATIVE
   END OF BINARY CARD 00000122
```

```
        1
    007343  035234 7410 56        2117           STX1     A$WORK,ID          AND STORE IN WORK WITH OLD BOUND INFO
    007344  005754 7170 00        2118           XED      T$WOVF             CHECK FOR STACK OVERFLOW
    007345  006210 7100 00        2119           TRA      A$OK               AND EXIT
    007346  006522 2200 00        2120 EPDEN     LDX0     PDPOS              GET PROCEDURE DENOTATION FLAG
    007347  007434 6010 00        2121           TNZ      EPDF               TRANSFER IF NOT A PROCEDURE DENOTATION
    007350  006534 2200 00        2122           LDX0     SEMIC              CHECK SEMICOLON COUNT
    007351  777777 6010 00        2123           TNZ      $ERROR             SHOULD BE ZERO
    007352  006533 2200 00        2124           LDX0     CLNC               CHECK COLON COUNT
    007353  777777 6010 00        2125           TNZ      $ERROR             SHOULD BE ZERO
    007354  006523 2200 00        2126           LDX0     MK                 GET PLACE WHERE WORKING STACK WAS MARKED
    007355  035214 0600 00        2127           ADX0     T$WORK             MAKE IT ABSOLUTE
    007356  007456 7400 00        2128           STX0     EPDD               AND SAVE
    007357  035234 2210 00        2129           LDX1     A$WORK             GET CURRENT WORKING STACK POINTER
    007360  007456 1610 00        2130           SBX1     EPDD               GET ADDED LENGTH OF WORK SINCE MARK
    007361  000001 6350 11        2131           EAA      1,1                ADD ONE AND PUT IN A REGISTER
    007362  000023 7310 00        2132           ARS      1+18               DIVIDE BY 2 AND PUT IN AL
    007363  000006 7350 00        2133           ALS      6                  PUT IN TALLY FIELD OF WORD
    007364  000002 0750 07        2134           ADA      2,DL               INCREMENT OF 2 IN AD MODIFICATION
    007365  007456 7550 00        2135           STA      EPDD               STORE AS TALLY WORD
    007366  035234 0110 54        2136           NOP      A$WORK,DI          DELETE MODE OF RESULT OF PROCEDURE
    007367  000000 2210 10        2137 EPD1      LDX1     0,0                GET DECLARER IN XR - 1
    007370  035216 0610 00        2138           ADX1     T$MODE             GET ABSOLUTE POINTER TO MODE TABLE
 END OF BINARY CARD 00000123
    007371  000000 2230 11        2139           LDX3     0,1                GET MODE TYPE IN XR - 3
    007372  016762 1030 03        2140           CMPX3    M$REF,DU           IS IT A REFERENCE MODE
    007373  777777 6010 00        2141           TNZ      $ERROR             NO - MUST BE IN A PROCEDURE
    007374  000001 2220 11        2142           LDX2     1,1                DEREFERENCE MODE
    007375  000000 7420 10        2143           STX2     0,0                AND RESTORE IN DECLARER
    007376  000002 0600 03        2144           ADX0     2,DU               STEP TO NEXT DECLARER
    007377  035234 1000 00        2145           CMPX0    A$WORK             ARE THERE ANY MORE DECLARERS
    007400  007367 6040 00        2146           TMI      EPD1               TRANSFER IF MORE
    007401  007634 7000 00        2147           TSX0     DECIA              DECLARE ALL FORMAL PARAMETERS
    007402  006523 2200 00        2148           LDX0     MK                 GET PLACE WHERE WORKING STACK WAS MARKED
    007403  035235 7400 56        2149           STX0     A$STACK,ID         STORE IN CONTROL STACK FOR EPROC
    007404  035234 0110 56        2150           NOP      A$WORK,ID          RESTORE MODE OF RESULT OF PROCEDURE
    007405  035214 0600 00        2151           ADX0     T$WORK             MAKE IT ABSOLUTE
    007406  007455 7400 00        2152           STX0     EPDS               STORE AS SOURCE ADDRESS
    007407  007456 7400 00        2153           STX0     EPDD               STORE AS DESTINATION ADDRESS
    007410  007455 2350 53        2154 EPD2      LDA      EPDS,AD            GET NEXT DECLARER
    007411  000000 6350 01        2155           EAA      0,AU               DELETE BOUNDS INFORMATION
    007412  007456 7550 56        2156           STA      EPDD,ID            AND STORE BACK IN WORK
    007413  007410 6070 00        2157           TTF      EPD2               AND TRANSFER IF MORE TO DO
    007414  007456 2200 00        2158           LDX0     EPDD               GET CURRENT WORKING STACK END
    007415  007417 7100 00        2159           TRA      EPD4               TRANSFER TO SHORTEN WORKING STACK
    007416  035234 0110 54        2160 EPD3      NOP      A$WORK,DI          DELETE A WORD FROM WORK
 END OF BINARY CARD 00000124
    007417  035234 1000 00        2161 EPD4      CMPX0    A$WORK             DONE ENOUGH
    007420  007416 6010 00        2162           TNZ      EPD3               NO - LOOP
    007421  007503 7000 00        2163           TSX0     EPROC              MAKE INTO PROCEDURE MODE
    007422  035235 2350 54        2164           LDA      A$STACK,DI         GET VALUE OF PUSHED MK
```

```
          1                                              PASS 1

     007423  006523 7550 00        2165         STA     MK               AND RESTORE MK
     007424  035234 2350 54        2166         LDA     A$WORK,DI        GET MODE OF PROCEDURE DENOTATION IN A
     007425  007456 7550 00        2167         STA     EPDD             AND SAVE
     007426  400000 0750 03        2168         ADA     =0400000,DU      SET THE SIGN BIT AS FLAG FOR RANGE TYPE
     007427  000022 7710 00        2169         ARL     18               MOVE TO LOWER HALF OF WORD
     007430  010066 7000 00        2170         TSX0    ERNGA            EXIT PROCEDURE DENOTATION RANGE
     007431  007456 2350 00        2171         LDA     EPDD             RECOVER MODE OF PROCEDURE DENOTATION
     007432  035234 7550 56        2172         STA     A$WORK,ID        AND PUSH ON TOP OF WORKING STACK
     007433  006210 7100 00        2173         TRA     A$OK             AND EXIT
     007434  035234 0110 54        2174 EPDF    NOP     A$WORK,DI        DELETE PSEUDO MODE OF RESULT
     007435  006534 2340 00        2175         SZN     SEMIC            ARE THERE ANY SEMICOLONS IN THIS RANGE
     007436  007440 6000 00        2176         TZE     EPDF0            NO - DO NOT DECLARE ANY IDENTIFIERS
     007437  007634 7000 00        2177         TSX0    DECIA            DECLARE ALL DECLARATIONS FOR THIS RANGE
     007440  006523 2200 00        2178 EPDF0   LDX0    MK               GET PLACE WHERE WORKING STACK WAS MARKED
     007441  035214 0600 00        2179         ADX0    T$WORK           MAKE IT ABSOLUTE
     007442  035234 1000 00        2180 EPDF1   CMPX0   A$WORK           THROUGH DELETING
     007443  007446 6000 00        2181         TZE     EPDF2            YES - TRANSFER
     007444  035234 0110 54        2182         NOP     A$WORK,DI        DELETE A WORD FROM THE WORKING STACK
   END OF BINARY CARD 00000125
     007445  007442 7100 00        2183         TRA     EPDF1            AND LOOP
     007446  035235 2350 54        2184 EPDF2   LDA     A$STACK,DI       GET VALUE OF PUSHED MK
     007447  006523 7550 00        2185         STA     MK               AND RESTORE MK
     007450  006531 2350 07        2186         LDA     CLOR,DL          GET TYPE OF RANGE IN AL
     007451  010066 7000 00        2187         TSX0    ERNGA            EXIT CLOSED EXPRESSION RANGE
     007452  000001 6350 00        2188         EAA     M$VOID           GET A VOID MODE IN A
     007453  035234 7550 56        2189         STA     A$WORK,ID        AND PUSH ON TOP OF WORKING STACK
     007454  006210 7100 00        2190         TRA     A$OK             AND EXIT
     007455  000000 0000 02        2191 EPDS    TALLYD  0,0,2
     007456  000000 000000         2192 EPDD    ZERO
     007457  000002 2350 03        2193 MMI     LDA     2,DU             GET LENGTH OF ENTRY IN MODE TABLE
     007460  035216 2210 03        2194         LDX1    T$MODE,DU        GET POINTER TO MODE TABLE CONTROL WORD
     007461  005663 7000 00        2195         TSX0    T$ALOC           AND ALLOCATE MEMORY IN MODE TABLE
     007462  016773 6350 00        2196         EAA     M$MMI            GET HEADER WORD FOR MODE TABLE ENTRY
     007463  000000 7550 11        2197         STA     0,1              AND STORE IN MODE TABLE ENTRY
     007464  006520 2350 00        2198         LDA     IDNT             GET IDENTIFIER DESCRIPTION
     007465  000001 7550 11        2199         STA     1,1              AND STORE IN MODE TABLE ENTRY
     007466  035216 1610 00        2200         SBX1    T$MODE           GET RELATIVE MODE TABLE ENTRY POINTER
     007467  035234 7410 51        2201         STX1    A$WORK,I         AND STORE ON WORKING STACK
     007470  000002 2350 03        2202         LDA     2,DU             GET LENGTH OF ENTRY IN BOUND TABLE
     007471  035217 2210 03        2203         LDX1    T$BOUND,DU       GET POINTER TO BOUND TABLE CONTROL WORD
     007472  005663 7000 00        2204         TSX0    T$ALOC           AND ALLOCATE MEMORY IN BOUND TABLE
   END OF BINARY CARD 00000126
     007473  016773 6350 00        2205         EAA     B$MMI            GET HEADER WORD FOR BOUND TABLE ENTRY
     007474  000000 7550 11        2206         STA     0,1              AND STORE IN BOUND TABLE ENTRY
     007475  006520 2350 00        2207         LDA     IDNT             GET IDENTIFIER DESCRIPTION
     007476  000001 7550 11        2208         STA     1,1              AND STORE IN BOUND TABLE ENTRY
     007477  035217 1610 00        2209         SBX1    T$BOUND          GET RELATIVE BOUND TABLE ENTRY POINTER
     007500  035234 4410 56        2210         SXL1    A$WORK,ID        AND STORE ON WORKING STACK WITH MODE INFO
     007501  005754 7170 00        2211         XED     T$WOVF           CHECK FOR STACK OVERFLOW
     007502  006210 7100 00        2212         TRA     A$OK             AND EXIT
```

```
            1                                                       PASS 1

       007503  017001 6350 00          2213 EPROC   EAA      M$PROC           GET HEADER WORD FOR MODE TABLE ENTRY
       007504  007513 7100 00          2214         TRA      EPR1             AND ASSEMBLE PROCEDURE
       007505  017001 6350 00          2215 EPR     EAA      M$PROC           GET HEADER WORD FOR MODE TABLE ENTRY
       007506  007513 7000 00          2216         TSX0     EPR1             AND ASSEMBLE LIKE UNION
       007507  006210 7100 00          2217         TRA      A$OK             AND EXIT
       007510  017007 6350 00          2218 CUN     EAA      M$UNION          GET HEADER WORD FOR MODE TABLE ENTRY
       007511  007513 7000 00          2219         TSX0     EPR1             ASSEMBLE UNION
       007512  006210 7100 00          2220         TRA      A$OK             AND EXIT
       007513  007556 7550 00          2221 EPR1    STA      CUNH             AND SAVE HEADER WORD
       007514  007553 7400 00          2222         STX0     CUNX             SAVE RETURN
       007515  035234 2200 00          2223         LDX0     A$WORK           GET CURRENT WORKING STACK POINTER
       007516  035214 1600 00          2224         SBX0     T$WORK           SUBTRACT BASE TO GET RELATIVE POINTER
       007517  035235 1600 54          2225         SBX0     A$STACK,DI       SUBTRACT RELATIVE POINTER WHEN MARKED
       007520  000001 6350 10          2226         EAA      1,0              GET NUMBER OF MODES + 1 IN A REGISTER
   END OF BINARY CARD 00000127
       007521  007554 7550 00          2227         STA      CUNA             AND SAVE
       007522  035216 2210 03          2228         LDX1     T$MODE,DU        GET POINTER TO MODE TABLE CONTROL WORD
       007523  005663 7000 00          2229         TSX0     T$ALOC           AND ALLOCATE SPACE IN MODE TABLE
       007524  000000 6230 11          2230         EAX3     0,1              SAVE ENTRY LOCATION IN XR - 3
       007525  007554 2350 00          2231         LDA      CUNA             GET LENGTH OF MODE TABLE ENTRY IN A
       007526  000014 7310 00          2232         ARS      12               MOVE LENGTH TO TALLY FIELD
       007527  007554 7550 00          2233         STA      CUNA             STORE AS PART OF ID WORD
       007530  007554 7410 00          2234         STX1     CUNA             STORE ADDRESS FOR TALLY WORD
       007531  035235 2200 51          2235         LDX0     A$STACK,I        GET RELATIVE POINTER TO LIST OF MODES
       007532  035214 0600 00          2236         ADX0     T$WORK           MAKE ABSOLUTE
       007533  007555 7400 00          2237         STX0     CUNB             AND SAVE AS TALLY WORD
       007534  007556 2350 00          2238         LDA      CUNH             GET HEADER WORD FOR MODE TABLE ENTRY
       007535  007554 7550 56          2239         STA      CUNA,ID          AND STORE IN MODE TABLE ENTRY
       007536  007555 2350 56          2240 CUN1    LDA      CUNB,ID          GET NEXT MODE FROM WORKING STACK
       007537  000000 6350 01          2241         EAA      0,AU             CLEAN OFF BOUND INFORMATION
       007540  007554 7550 56          2242         STA      CUNA,ID          AND STORE IN MODE TABLE ENTRY
       007541  007536 6070 00          2243         TTF      CUN1             TRANSFER IF MORE MODES TO MOVE
       007542  035235 2260 51          2244         LDX6     A$STACK,I        GET RELATIVE POINTER TO WORK WHEN MARKED
       007543  035214 0660 00          2245         ADX6     T$WORK           MAKE ABSOLUTE
       007544  035234 0110 54          2246         NOP      A$WORK,DI        DELETE A WORD FROM THE WORKING STACK
       007545  035234 1060 00          2247         CMPX6    A$WORK           IS STACK DELETED BACK TO MARK
       007546  007544 6010 00          2248         TNZ      *-2              TRANSFER IF MORE TO REMOVE
   END OF BINARY CARD 00000128
       007547  035216 1630 00          2249         SBX3     T$MODE           GET RELATIVE POINTER TO MODE TABLE ENTRY
       007550  000000 6350 13          2250         EAA      0,3              PUT IN A REGISTER WITH AL CLEAR
       007551  035234 7550 56          2251         STA      A$WORK,ID        STORE POINTER TO MODE TABLE ENTRY IN WORKING STAC
       007552  005754 7170 00          2252         XED      T$WOVF           CHECK FOR STACK OVERFLOW
       007553  000000 7100 00          2253 CUNX    TRA      **               AND RETURN
       007554  000000 000000           2254 CUNA    ZERO
       007555  000000 000000           2255 CUNB    ZERO
       007556  000000 000000           2256 CUNH    ZERO
       007557  000004 2350 03          2257 DECM    LDA      4,DU             GET LENGTH OF DEF TABLE ENTRY IN A
       007560  035220 2210 03          2258         LDX1     T$DEF,DU         GET POINTER TO DEF TABLE CONTROL WORD
       007561  005663 7000 00          2259         TSX0     T$ALOC           ALLOCATE SPACE IN DEF TABLE
       007562  006412 6350 00          2260         EAA      D$MODE           GET TYPE OF DEFINITION IN A
```

```
              1                                          PASS 1

      007563  000001 7550 11       2261        STA     1,1               STORE IN DEF TABLE ENTRY
      007564  007721 7000 00       2262        TSX0    SETD              LINK THIS ENTRY TO OTHERS FOR THIS IDENTIFIER
      007565  035234 2350 54       2263        LDA     A$WORK,DI         GET DECLARER IN A
      007566  000002 7550 11       2264        STA     2,1               STORE IN DEF TABLE ENTRY
      007567  000003 4500 11       2265        STZ     3,1               ZERO OUT LAST WORD FOR NEATNESS
      007570  006210 7100 00       2266        TRA     A$OK              AND EXIT
      007571  000003 2350 03       2267 DECP   LDA     3,DU              GET LENGTH OF DEF TABLE ENTRY IN A
      007572  035220 2210 03       2268        LDX1    T$DEF,DU          GET POINTER TO DEF TABLE CONTROL WORD
      007573  005663 7000 00       2269        TSX0    T$ALOC            ALLOCATE SPACE IN DEF TABLE
      007574  006414 6350 00       2270        EAA     D$PRIOR           GET TYPE OF DEFINITION IN A
    END OF BINARY CARD 00000129
      007575  000001 7550 11       2271        STA     1,1               STORE IN DEF TABLE ENTRY
      007576  007721 7000 00       2272        TSX0    SETD              LINK THIS ENTRY TO OTHERS FOR THIS IDENTIFIER
      007577  035234 2350 54       2273        LDA     A$WORK,DI         GET PRIORITY NUMBER
      007600  000000 6350 01       2274        EAA     0,AU              CLEAN IT UP
      007601  000002 7550 11       2275        STA     2,1               STORE IN DEF TABLE ENTRY
      007602  006210 7100 00       2276        TRA     A$OK              AND EXIT
      007603  000004 2350 03       2277 DECO   LDA     4,DU              GET LENGTH OF DEF TABLE ENTRY IN A
      007604  035220 2210 03       2278        LDX1    T$DEF,DU          GET POINTER TO DEF TABLE CONTROL WORD
      007605  005663 7000 00       2279        TSX0    T$ALOC            ALLOCATE SPACE IN DEF TABLE
      007606  006413 6350 00       2280        EAA     D$OP              GET TYPE OF DEFINITION IN A
      007607  000001 7550 11       2281        STA     1,1               STORE IN DEF TABLE ENTRY
      007610  007721 7000 00       2282        TSX0    SETD              LINK THIS ENTRY TO OTHERS FOR THIS IDENTIFIER
      007611  035234 2350 54       2283        LDA     A$WORK,DI         GET MODE OF OPERATOR IN A
      007612  000002 7550 11       2284        STA     2,1               STORE IN DEF TABLE ENTRY
      007613  000003 4500 11       2285        STZ     3,1               ZERO OUT LAST WORD FOR NEATNESS
      007614  035220 1610 00       2286        SBX1    T$DEF             MAKE POINTER TO TABLE ENTRY RELATIVE
      007615  035235 7410 56       2287        STX1    A$STACK,ID        AND STORE IN CONTROL STACK
      007616  005742 7170 00       2288        XED     T$SOVF            CHECK FOR STACK OVERFLOW
      007617  006210 7100 00       2289        TRA     A$OK              AND EXIT
      007620  035235 2210 54       2290 DECOP  LDX1    A$STACK,DI        GET POINTER TO OPERATOR DEFINITION
      007621  035220 0610 00       2291        ADX1    T$DEF             MAKE IT ABSOLUTE
      007622  035234 2350 54       2292        LDA     A$WORK,DI         GET MODE OF OPERATOR IN A
    END OF BINARY CARD 00000130
      007623  000002 7550 11       2293        STA     2,1               AND STORE IN DEFINITION OF OPERATOR
      007624  006210 7100 00       2294        TRA     A$OK              AND EXIT
      007625  007634 7000 00       2295 DECI   TSX0    DECIA             DECLARE ALL IDENTIFIERS IN CURRENT RANGE
      007626  006523 2200 00       2296        LDX0    MK                GET PLACE WHERE WORKING STACK WAS MARKED
      007627  035214 0600 00       2297        ADX0    T$WORK            MAKE IT ABSOLUTE
      007630  035234 1000 00       2298 DECI1  CMPX0   A$WORK            HAS STACK BEEN DELETED BACK TO THE MARK
      007631  006210 6000 00       2299        TZE     A$OK              YES - EXIT
      007632  035234 0110 54       2300        NOP     A$WORK,DI         DELETE A WORD FROM THE WORKING STACK
      007633  007630 7100 00       2301        TRA     DECI1             AND TRY AGAIN
      007634  007674 7400 00       2302 DECIA  STX0    DECIX             SAVE RETURN ADDRESS
      007635  006523 2200 00       2303        LDX0    MK                GET PLACE WHERE WORKING STACK WAS MARKED
      007636  007675 7400 00       2304        STX0    DECIT             AND SAVE
      007637  007675 2200 00       2305 DECIL  LDX0    DECIT             GET POINTER TO PLACE IN WORKING STACK
      007640  035214 0600 00       2306        ADX0    T$WORK            MAKE IT ABSOLUTE
      007641  035234 1000 00       2307        CMPX0   A$WORK            SEE IF BEYOND END OF WORKING STACK
      007642  007674 6050 00       2308        TPL     DECIX             TRANSFER IF ALL DONE
```

PASS 1

```
     007643  000000 2270 10        2309          LDX7     0,0              GET MODE OF SYMBOL IN XR - 7
     007644  000001 1070 03        2310          CMPX7    M$VOID,DU        SEE IF BAD DECLARATION
     007645  007671 6000 00        2311          TZE      DECI3            TRANSFER TO SKIP BAD DECLARATION
     007646  000000 2200 17        2312          LDX0     0,7              GET TYPE OF MODE IN XR - 0
     007647  016762 1000 03        2313          CMPX0    M$REF,DU         IS IT A REFERENCE MODE
     007650  007654 6010 00        2314          TNZ      DECI2            NO - GO PROCESS GOOD DECLARATION
END OF BINARY CARD 00000131
     007651  000001 2270 17        2315          LDX7     1,7              GET DEREFERENCED MODE IN XR - 7
     007652  000001 1070 03        2316          CMPX7    M$VOID,DU        SEE IF BAD DECLARATION
     007653  007671 6000 00        2317          TZE      DECI3            TRANSFER TO SKIP BAD DECLARATION
     007654  000004 2350 03        2318 DECI2    LDA      4,DU             GET LENGTH OF ENTRY IN DEF TABLE
     007655  035220 2210 03        2319          LDX1     T$DEF,DU         GET POINTER TO TABLE CONTROL WORD
     007656  005663 7000 00        2320          TSX0     T$ALOC           AND ALLOCATE SPACE IN THE DEF TABLE
     007657  007675 2200 00        2321          LDX0     DECIT            GET RELATIVE POINTER TO WORKING STACK
     007660  035214 0600 00        2322          ADX0     T$WORK           MAKE IT ABSOLUTE
     007661  000000 2350 10        2323          LDA      0,0              GET DECLARER
     007662  000002 7550 11        2324          STA      2,1              AND PUT IN THE DEF TABLE ENTRY
     007663  000003 4500 11        2325          STZ      3,1              ZERO OUT LAST WORD FOR NEATNESS
     007664  006415 6350 00        2326          EAA      D$IDENT          GET TYPE OF DEFINITION IN A
     007665  000001 7550 11        2327          STA      1,1              STORE IN DEF TABLE ENTRY
     007666  000001 2200 10        2328          LDX0     1,0              GET POINTER TO TAG
     007667  006520 7400 00        2329          STX0     IDNT             AND STORE IN IDNT
     007670  007721 7000 00        2330          TSX0     SETD             LINK THIS ENTRY TO OTHERS FOR THIS IDENTIFIER
     007671  000002 2200 03        2331 DECI3    LDX0     2,DU             GET A 2
     007672  007675 0400 00        2332          ASX0     DECIT            AND STEP TO THE NEXT DECLARATION IF ANY
     007673  007637 7100 00        2333          TRA      DECIL            AND LOOP
     007674  000000 7100 00        2334 DECIX    TRA      **               AND RETURN
     007675  000000 000000         2335 DECIT    ZERO
     007676  000031 2350 03        2336 DECL     LDA      M$LBL,DU         GET PSEUDO LABEL MODE IN AU
END OF BINARY CARD 00000132
     007677  035234 7550 56        2337          STA      A$WORK,ID        AND STORE IN WORKING STACK
     007700  005754 7170 00        2338          XED      T$WOVF           CHECK FOR STACK OVERFLOW
     007701  006520 2350 00        2339          LDA      IDNT             GET LABEL IDENTIFIER IN A
     007702  035234 7550 56        2340          STA      A$WORK,ID        AND STORE IN WORKING STACK
     007703  005754 7170 00        2341          XED      T$WOVF           CHECK FOR STACK OVERFLOW
     007704  006210 7100 00        2342          TRA      A$OK             AND EXIT
     007705  000004 2350 03        2343 DECLB    LDA      4,DU             GET LENGTH OF DEF TABLE ENTRY IN A
     007706  035220 2210 03        2344          LDX1     T$DEF,DU         GET POINTER TO DEF TABLE CONTROL WORD
     007707  005663 7000 00        2345          TSX0     T$ALOC           ALLOCATE SPACE IN DEF TABLE
     007710  006415 6350 00        2346          EAA      D$IDENT          GET TYPE OF DEFINITION IN A
     007711  000001 7550 11        2347          STA      1,1              STORE IN DEF TABLE ENTRY
     007712  000000 6200 00        2348          EAX0     0                GET LL OF IDENTIFIER IN XR - 0
     007713  007753 7500 00        2349          STC2     SETDX            SAVE RETURN
     007714  007723 7100 00        2350          TRA      SETDE            LINK THIS ENTRY TO OTHERS FOR THIS IDENTIFIER
     007715  035234 2350 54        2351          LDA      A$WORK,DI        GET MODE OF IDENTIFIER IN A
     007716  000002 7550 11        2352          STA      2,1              AND STORE IN DEFINITION OF IDENTIFIER
     007717  000003 4500 11        2353          STZ      3,1              ZERO OUT LAST WORD FOR NEATNESS
     007720  006210 7100 00        2354          TRA      A$OK             AND EXIT
     007721  007753 7400 00        2355 SETD     STX0     SETDX            SAVE RETURN
     007722  006520 7200 00        2356          LXL0     IDNT             GET CURRENT RANGE NUMBER IN XR - 0
```

```
          1                                         PASS 1

     007723  000000 4400 11        2357 SETDE   SXL0     0,1               STORE IN LINK WORD OF DEF TABLE ENTRY
     007724  006520 2250 00        2358         LDX5     IDNT              GET POINTER TO IDENTIFIER DEFINITION
 END OF BINARY CARD 00000133
     007725  000001 6220 15        2359         EAX2     A$DF,5            GET POINTER TO DEF CHAIN IN XR - 2
     007726  035222 0620 00        2360         ADX2     T$STAB            MAKE IT ABSOLUTE
     007727  000000 2230 12        2361 SETD1   LDX3     0,2               GET RELATIVE POINTER TO NEXT DEFINITION
     007730  007747 6040 00        2362         TMI      SETD4             TRANSFER IF THERE ARE NO MORE
     007731  035220 0630 00        2363         ADX3     T$DEF             MAKE XR - 3 ABSOLUTE
     007732  000000 7240 13        2364         LXL4     0,3               GET RANGE NUMBER IN XR - 4
     007733  007754 7440 00        2365         STX4     SETDT             AND SAVE FOR COMPARE
     007734  007754 1000 00        2366         CMPX0    SETDT             IS THIS PROPER PLACE IN CHAIN FOR NEW ENTRY
     007735  007740 6050 00        2367         TPL      SETD2             TRANSFER IF YES
     007736  000000 6220 13        2368         EAX2     0,3               MAKE NEXT ENTRY CURRENT ENTRY
     007737  007727 7100 00        2369         TRA      SETD1             AND TRY AGAIN
     007740  007746 6010 00        2370 SETD2   TNZ      SETD3             TRANSFER IF NO MORE CHECKING NEEDED
     007741  000001 2240 11        2371         LDX4     1,1               GET TYPE OF NEW DEFINITION IN XR - 4
     007742  000001 1040 13        2372         CMPX4    1,3               SEE IF IN RIGHT PLACE IN TABLE
     007743  007746 6040 00        2373         TMI      SETD3             YES - TRANSFER
     007744  000000 6220 13        2374         EAX2     0,3               MAKE NEW ENTRY CURRENT ENTRY
     007745  007727 7100 00        2375         TRA      SETD1             AND TRY AGAIN
     007746  035220 1630 00        2376 SETD3   SBX3     T$DEF             MAKE POINTER TO FOLLOWING ENTRY RELATIVE
     007747  000000 7430 11        2377 SETD4   STX3     0,1               AND MAKE NEW ENTRY POINT TO FOLLOWING ENTRY
     007750  000000 6240 11        2378         EAX4     0,1               GET ADDRESS OF NEW ENTRY IN XR - 4
     007751  035220 1640 00        2379         SBX4     T$DEF             MAKE IT RELATIVE
     007752  000000 7440 12        2380         STX4     0,2               AND MAKE PRECEEDING ENTRY POINT TO NEW ENTRY
 END OF BINARY CARD 00000134
     007753  000000 7100 00        2381 SETDX   TRA      **                AND RETURN TO CALLER
     007754  000000 000000         2382 SETDT   ZERO
     007755  035217 2210 03        2383 CBOX    LDX1     T$BOUND,DU        GET POINTER TO BOUND TABLE CONTROL WORD
     007756  000002 6350 00        2384         EAA      2                 GET NUMBER OF WORDS TO ALLOCATE IN AU
     007757  005663 7000 00        2385         TSX0     T$ALOC            ALLOCATE 2 WORDS IN BOUND TABLE
     007760  016770 2200 03        2386         LDX0     B$ROW,DU          GET HEADER FOR BOUND TABLE ENTRY
     007761  000000 7400 11        2387         STX0     0,1               AND STORE IN TABLE ENTRY
     007762  006532 2200 00        2388         LDX0     COMAC             GET DIMENSION OF ARRAY IN XR - 0
     007763  000001 0600 03        2389         ADX0     1,DU              ADD ONE TO MAKE EXACT
     007764  000000 4400 11        2390         SXL0     0,1               AND STORE IN TABLE ENTRY
     007765  000001 4500 11        2391         STZ      1,1               CLEAR OUT SECOND WORD IN TABLE ENTRY
     007766  006536 2200 00        2392         LDX0     CURR              GET RELATIVE POINTER TO CURRENT RANGE
     007767  000001 4400 11        2393         SXL0     1,1               AND STORE IN TABLE ENTRY
     007770  035217 1610 00        2394         SBX1     T$BOUND           MAKE POINTER TO TABLE ENTRY RELATIVE
     007771  007773 4410 00        2395         SXL1     CBOXT             AND SAVE
     007772  006210 7100 00        2396         TRA      A$OK              AND EXIT
     007773  000000 000000         2397 CBOXT   ZERO
     007774  035234 2350 54        2398 EBOX    LDA      A$WORK,DI         GET DECLARER OF ELEMENT OF ROW MODE
     007775  000000 6210 05        2399         EAX1     0,AL              GET BOUND OF DECLARER IN XR - 1
     007776  000000 6350 01        2400         EAA      0,AU              GET CLEAN MODE OF DECLARER IN AU
     007777  010026 7550 00        2401         STA      EBOXM             AND SAVE
     010000  035234 7220 54        2402         LXL2     A$WORK,DI         GET POINTER TO BOUND IN XR - 2
 END OF BINARY CARD 00000135
     010001  035217 0620 00        2403         ADX2     T$BOUND           MAKE POINTER ABSOLUTE
```

```
               1                                              PASS 1

      010002  000001 7410 12        2404        STX1     1,2                 LINK BOUND OF ELEMENT TO BOUND OF ARRAY
      010003  000000 7260 12        2405        LXL6     0,2                 GET DIMENSION OF ARRAY IN XR - 6
      010004  016776 6350 00        2406        EAA      M$ROWF              GET FINAL MODE TABLE ENTRY HEADER
      010005  010027 7550 00        2407        STA      EBOXH               AND STORE IN MEMORY
      010006  000002 6350 00        2408 EBOX1  EAA      2                   GET LENGTH OF ROW MODE ENTRY IN AU
      010007  035216 2210 03        2409        LDX1     T$MODE,DU           GET POINTER TO MODE TABLE CONTROL WORD IN XR - 1
      010010  005663 7000 00        2410        TSX0     T$ALOC              ALLOCATE SPACE FOR ROW MODE ENTRY IN MODE TABLE
      010011  010027 2350 00        2411        LDA      EBOXH               GET HEADER FOR TABLE ENTRY
      010012  000000 7550 11        2412        STA      0,1                 AND STORE IN TABLE ENTRY
      010013  010026 2350 00        2413        LDA      EBOXM               GET MODE OF ELEMENT OF ARRAY
      010014  000001 7550 11        2414        STA      1,1                 AND STORE IN TABLE ENTRY
      010015  016770 6350 00        2415        EAA      M$ROW               GET HEADER FOR SUBSEQUENT ENTRIES
      010016  010027 7550 00        2416        STA      EBOXH               AND STORE IN MEMORY
      010017  035216 1610 00        2417        SBX1     T$MODE              MAKE NEW MODE POINTER RELATIVE
      010020  010026 7410 00        2418        STX1     EBOXM               AND STORE AS NEW MODE
      010021  000001 1660 03        2419        SBX6     1,DU                DECREMENT NUMBER OF ENTRIES YET TO CREATE
      010022  010006 6010 00        2420        TNZ      EBOX1               TRANSFER IF MORE TO CREATE
      010023  035234 7410 56        2421        STX1     A$WORK,ID           STORE POINTER TO MODE TABLE ENTRY IN WORK
      010024  005754 7170 00        2422        XED      T$WOVF              CHECK FOR STACK OVERFLOW
      010025  006210 7100 00        2423        TRA      A$OK                AND EXIT
      010026  000000 000000         2424 EBOXM  ZERO
   END OF BINARY CARD 00000136
      010027  000000 000000         2425 EBOXH  ZERO
      010030  000001 2200 03        2426 ADD    LDX0     1,DU                GET A ONE FOR INCREMENTING
      010031  000000 0400 05        2427        ASX0     0,AL                INCREMENT SPECIFIED LOCATION
      010032  006210 7100 00        2428        TRA      A$OK                AND EXIT
      010033  777777 3750 07        2429 MBND   ANA      =O777777,DL         GET TYPE OF BOUND IN AL
      010034  006520 2750 00        2430        ORA      IDNT                OR IN IDENTIFIER IF ANY FOR OPTIMIZING
      010035  035234 7550 56        2431        STA      A$WORK,ID           STORE IN THE WORKING STACK
      010036  005754 7170 00        2432        XED      T$WOVF              CHECK FOR STACK OVERFLOW
      010037  006210 7100 00        2433        TRA      A$OK                AND EXIT
      010040  000000 6270 05        2434 SRNGE  EAX7     0,AL                SAVE TYPE OF RANGE IN XR - 7
      010041  000006 6350 00        2435        EAA      6                   GET LENGTH OF PROG LIST ELEMENT IN AU
      010042  035221 2210 03        2436        LDX1     T$PROG,DU           GET POINTER TO PROG TABLE CONTROL WORD IN XR - 1
      010043  005663 7000 00        2437        TSX0     T$ALOC              AND ALLOCATE SPACE IN THE PROG TABLE
      010044  000000 7470 11        2438        STX7     0,1                 STORE TYPE OF RANGE AS PROG TABLE ENTRY HEADER
      010045  035221 1610 00        2439        SBX1     T$PROG              GET RELATIVE POINTER TO PROG TABLE ENTRY
      010046  010062 2350 00        2440        LDA      SRTAL               GET TALLY WORD TO TABLE OF THINGS TO SAVE
      010047  010063 7550 00        2441        STA      SRT                 AND STORE IN TALLY LOCATION
      010050  010063 2350 20        2442 SR1    LDA      SRT,*               GET NEXT ITEM TO SAVE
      010051  035234 7550 56        2443        STA      A$WORK,ID           AND SAVE IT IN THE WORKING STACK
      010052  005754 7170 00        2444        XED      T$WOVF              CHECK FOR STACK OVERFLOW
      010053  010063 4500 57        2445        STZ      SRT,IDC             ZERO OUT ITEM AND STEP TO NEXT ITEM
      010054  010050 6070 00        2446        TTF      SR1                 TRANSFER IF THERE ARE MORE ITEMS TO CONSIDER
   END OF BINARY CARD 00000137
      010055  006536 7410 00        2447        STX1     CURR                AND SAVE RELATIVE POINTER TO CURRENT RANGE
      010056  006535 4410 00        2448        SXL1     LEVEL               STORE NEW RANGE POINTER
      010057  006535 2350 00        2449        LDA      LEVEL               GET LEVEL IN A REGISTER
      010060  006520 7550 00        2450        STA      IDNT                AND STORE IN IDENT
      010061  006210 7100 00        2451        TRA      A$OK                AND EXIT
```

```
         1                                                 PASS 1

     010062  010116 0005 51        2452 SRTAL   TALLYC   RTAB,RTABE-RTAB,I
     010063  000000 000000         2453 SRT     ZERO
     010064  010066 7000 00        2454 ERNGE   TSX0     ERNGA               EXIT RANGE
     010065  006210 7100 00        2455         TRA      A$OK                AND EXIT
     010066  006536 2210 00        2456 ERNGA   LDX1     CURR                GET RELATIVE POINTER TO PROG TABLE ENTRY
     010067  035221 0610 00        2457         ADX1     T$PROG              MAKE IT ABSOLUTE
     010070  000000 6270 05        2458         EAX7     0,AL                GET TYPE OF END OF RANGE IN XR - 7
     010071  000000 4470 11        2459         SXL7     0,1                 AND STORE IN PROG TABLE ENTRY
     010072  006520 2220 00        2460         LDX2     IDNT                GET CURRENT LEVEL
     010073  000002 7420 11        2461         STX2     2,1                 AND STORE IN PROG TABLE
     010074  777777 2220 03        2462         LDX2     -1,DU               GET A MINUS FLAG
     010075  000002 4420 11        2463         SXL2     2,1                 AND INITIALIZE DEFINITION CHAIN FOR THIS RANGE
     010076  006534 2350 00        2464         LDA      SEMIC               GET SEMI COUNT
     010077  000003 7550 11        2465         STA      3,1                 AND STORE IN PROG TABLE
     010100  006533 2350 00        2466         LDA      CLNC                GET COLON COUNT
     010101  000004 7550 11        2467         STA      4,1                 AND STORE IN PROG TABLE
     010102  006532 2350 00        2468         LDA      COMAC               GET COMMA COUNT
   END OF BINARY CARD 00000138
     010103  000005 7550 11        2469         STA      5,1                 AND STORE IN PROG TABLE
     010104  010114 2350 00        2470         LDA      ERTAL               GET TALLY WORD TO TABLE OF THINGS TO RESTORE
     010105  010115 7550 00        2471         STA      ERT                 AND STORE IN TALLY LOCATION
     010106  035234 2350 54        2472 ER1     LDA      A$WORK,DI           GET NEXT ITEM FROM WORKING STACK
     010107  010115 7550 55        2473         STA      ERT,DIC             AND RESTORE IT
     010110  010106 6070 00        2474         TTF      ER1                 TRANSFER IF THERE ARE MORE ITEMS TO CONSIDER
     010111  006520 7220 00        2475         LXL2     IDNT                GET NUMBER OF SURROUNDING RANGE
     010112  000002 7420 11        2476         STX2     2,1                 AND STORE IN PROG TABLE
     010113  000000 7100 10        2477         TRA      0,0                 AND RETURN
     010114  010123 7773 51        2478 ERTAL   TALLYC   RTABE,RTAB-RTABE,I
     010115  000000 000000         2479 ERT     ZERO
     010116  006532 000000         2480 RTAB    ZERO     COMAC
     010117  006533 000000         2481         ZERO     CLNC
     010120  006536 000000         2482         ZERO     CURR
     010121  006534 000000         2483         ZERO     SEMIC
     010122  006520 000000         2484         ZERO     IDNT
                         010123    2485 RTABE   EQU      *
                         010123    2486         BSS      10
```

```
                                     2487         TTLS                PASS 1.5
                                     2488         HEAD     A
     010135  006610 2350 00          2489 PASS2   LDA      1$XXT      GET INITIAL TALLY WORD FOR PRINT ROUTINE
     010136  006607 7550 00          2490         STA      1$XX2      AND INITIALIZE TALLY WORD
     010137  000040 2350 07          2491         LDA      =O040,DL   GET A SPACE
     010140  006607 7550 52          2492         STA      1$XX2,SC   AND STORE IN OUTPUT BUFFER
     010141  000061 2350 07          2493         LDA      =O61,DL    GET THE DIGIT ONE
 END OF BINARY CARD 00000139
     010142  006607 7550 52          2494         STA      1$XX2,SC   AND STORE IN OUTPUT BUFFER
     010143  035221 2270 00          2495         LDX7     T$PROG     GET ADDRESS OF START OF PROGRAM TABLE
     010144  000001 0670 03          2496         ADX7     1,DU       GET POINTER TO BEGINNING OF FIRST ENTRY IN TABLE
     010145  010305 7470 00          2497 PL1     STX7     TEMP       SAVE LOCATION OF CURRENT TABLE ENTRY
     010146  777777 0670 17          2498         ADX7     -1,7       STEP TO NEXT ENTRY IN TABLE
     010147  000001 0670 03          2499         ADX7     1,DU       AND STEP OVER LINK WORD
     010150  035221 7200 00          2500         LXL0     T$PROG     GET LENGTH OF PROGRAM TABLE
     010151  035221 0600 00          2501         ADX0     T$PROG     ADD BASE LOCATION OF PROGRAM TABLE
     010152  010153 7400 00          2502         STX0     *+1        AND STORE FOR COMPARE
     010153  000000 1070 03          2503         CMPX7    **,DU      ARE WE BEYOND THE END OF THE TABLE
     010154  010205 6050 00          2504         TPL      PL5        YES - GO TO CLEAN UP
     010155  000002 2260 17          2505         LDX6     2,7        GET RELATIVE POINTER TO SURROUNDING RANGE
     010156  010171 6000 00          2506         TZE      PL3        TRANSFER IF GLOBAL
     010157  035221 0660 00          2507         ADX6     T$PROG     MAKE IT ABSOLUTE
     010160  010305 1060 00          2508         CMPX6    TEMP       IS THIS THE LAST RANGE
     010161  010171 6010 00          2509         TNZ      PL3        NO - SKIP TO NEW LINE
     010162  035221 1670 00          2510 PL2     SBX7     T$PROG     GET RELATIVE POINTER TO CURRENT RANGE
     010163  000000 6350 17          2511         EAA      0,7        AND PUT IT IN AU
     010164  000022 7710 00          2512         ARL      18         AND MOVE IT TO AL
     010165  006470 7000 00          2513         TSX0     BCD        CONVERT RANGE TO DECIMAL
     010166  006607 7550 56          2514         STA      1$XX2,ID   AND STORE IN OUTPUT BUFFER
     010167  035221 0670 00          2515         ADX7     T$PROG     MAKE XR - 7 ABSOLUTE
 END OF BINARY CARD 00000140
     010170  010145 7100 00          2516         TRA      PL1        AND LOOP
     010171  006603 7500 00          2517 PL3     STC2     1$XRET     SAVE RETURN
     010172  006571 0110 00          2518         NOP      1$XXL3     AND PRINT OUT OUTPUT BUFFER
     010173  006610 2350 00          2519         LDA      1$XXT      GET INITIAL TALLY WORD FOR PRINT ROUTINE
     010174  006607 7550 00          2520         STA      1$XX2      AND INITIALIZE TALLY WORD
     010175  010304 2350 00          2521         LDA      BLNKS      GET A WORD OF SPACES
     010176  006607 7550 52          2522         STA      1$XX2,SC   STORE SPACE IN OUTPUT BUFFER
     010177  006607 7550 52          2523         STA      1$XX2,SC   STORE SPACE IN OUTPUT BUFFER
     010200  000002 2260 16          2524 PL4     LDX6     2,6        GET POINTER TO SURROUNDING RANGE
     010201  010162 6000 00          2525         TZE      PL2        NO MORE SO RESUME PRINTING RANGES
     010202  006607 7550 56          2526         STA      1$XX2,ID   STORE FOUR SPACES IN OUTPUT BUFFER
     010203  035221 0660 00          2527         ADX6     T$PROG     MAKE RANGE POINTER ABSOLUTE
     010204  010200 7100 00          2528         TRA      PL4        AND LOOP
     010205  006603 7500 00          2529 PL5     STC2     1$XRET     SAVE RETURN
     010206  006571 0110 00          2530         NOP      1$XXL3     AND PRINT LAST LINE FROM OUTPUT BUFFER
     010207  035222 2240 00          2531         LDX4     T$STAB     GET POINTER TO START OF SYMBOL TABLE
     010210  000001 6270 14          2532 CL1     EAX7     DF,4       GET POINTER TO NEXT SYMBOL IN XR - 7
     010211  777777 2200 03          2533         LDX0     -1,DU      GET AN UNUSED LEVEL NUMBER
     010212  006460 7400 00          2534         STX0     LEVEL      AND INITIALIZE LEVEL
```

```
           A                                                    PASS 1,5

     010213  000000 2260 17        2535 CL2    LDX6    0,7                 MAKE XR - 6 POINT TO NEXT DEFINITION FOR SYMBOL
     010214  006457 7470 00        2536        STX7    LAST                SAVE PREVIOUS POINTER IN LAST
     010215  010405 6040 00        2537 CL3    TMI     CL4                 NO MORE SO GO TO NEXT SYMBOL
 END OF BINARY CARD 00000141
     010216  035220 0660 00        2538        ADX6    T$DEF               GET ABSOLUTE POINTER TO NEXT TABLE ENTRY
     010217  000000 7200 16        2539        LXL0    0,6                 GET LEVEL OF CURRENT DEFINITION
     010220  006460 1000 00        2540        CMPX0   LEVEL               IS IT THE SAME AS THE LAST DEFINITION
     010221  010224 6000 00        2541        TZE     CL5                 TRANSFER IF THE SAME
     010222  000000 2250 03        2542        LDX5    0,DU                NOT THE SAME SO SET STATE TO ZERO
     010223  006460 7400 00        2543        STX0    LEVEL               AND SET LEVEL TO THAT OF CURRENT ENTRY
     010224  000001 2200 16        2544 CL5    LDX0    1,6                 GET TYPE OF DEFINITION IN XR - 0
     010225  001631 7160 10        2545        XEC     -D$MODE+CLTAB,0 JUMP TO NEXT STATE USING XR - 0 AND XR - 5
     010226  000000 6270 16        2546 CLOK   EAX7    0,6                 STEP TO NEXT DEFINITION
     010227  010213 7100 00        2547        TRA     CL2                 AND LOOP
     010230  000000 2350 16        2548 CLER   LDA     0,6                 GET LEVEL OF DEFINITION
     010231  777777 3750 07        2549        ANA     -1,DL               MASK OUT ALL BUT LEVEL
     010232  006470 7000 00        2550        TSX0    BCD                 CONVERT LEVEL TO DECIMAL
     010233  010303 7550 00        2551        STA     EM1+2               AND STORE IN ERROR MESSAGE
     010234  010300 2350 00        2552        LDA     EM1SC               GET TALLY WORD TO ERROR MESSAGE
     010235  006562 7000 00        2553        TSX0    1$XXX               AND PRINT IT
     010236  000000 2350 14        2554        LDA     SC,4                GET TALLY WORD TO IDENTIFIER
     010237  006554 7000 00        2555        TSX0    1$XXS               AND PRINT IT
     010240  000000 2260 16        2556 CLDEL  LDX6    0,6                 GET POINTER TO FOLLOWING TABLE ENTRY
     010241  006457 7460 51        2557        STX6    LAST,I              AND STORE IT IN PREVIOUS TABLE ENTRY
     010242  010215 7100 00        2558        TRA     CL3                 AND CONTINUE PROCESSING DEFINITIONS
     010243  010247 7160 15        2559 CLTAB  XEC     MTAB,5
 END OF BINARY CARD 00000142
     010244  010251 7160 15        2560        XEC     OTAB,5
     010245  010254 7160 15        2561        XEC     PTAB,5
     010246  010306 7100 00        2562        TRA     IDN
     010247  000001 2250 03        2563 MTAB   LDX5    1,DU
     010250  010230 7100 00        2564        TRA     CLER
     010251  000002 2250 03        2565 OTAB   LDX5    2,DU
     010252  010230 7100 00        2566        TRA     CLER
     010253  010226 7100 00        2567        TRA     CLOK
     010254  000003 2250 03        2568 PTAB   LDX5    3,DU
     010255  010230 7100 00        2569        TRA     CLER
     010256  010226 7100 00        2570        TRA     CLOK
     010257  010230 7100 00        2571        TRA     CLER
     010260  000003 2250 03        2572 IDTAB  LDX5    3,DU
     010261  010230 7100 00        2573        TRA     CLER
     010262  010230 7100 00        2574        TRA     CLER
     010263  010230 7100 00        2575        TRA     CLER
     010264  000004710004
                        010270     2576        EIGHT
     010270  000000000000          2577 IDREG  OCT     0,0,0,0,0,0,0,0
     010271  000000000000
     010272  000000000000
     010273  000000000000
 END OF BINARY CARD 00000143
```

```
     010274  000000000000
     010275  000000000000
     010276  000000000000
     010277  000000000000
     010300  010301 0015 40        2578 EM1SC   TALLYB  EM1,13
     010301  015012114105          2579 EM1     OCT     015012114105,126105114040,0
     010302  126105114040
     010303  000000000000
     010304  040040040040          2580 BLNKS   OCT     040040040040
     010305  000000 000000         2581 TEMP    ZERO
     010306  006460 2200 00        2582 IDN     LDX0    LEVEL          GET CURRENT LEVEL
     010307  010270 7530 00        2583         SREG    IDREG          SAVE REGISTERS
     010310  000000 6210 16        2584         EAX1    0,6            GET POINTER TO CURRENT IDENTIFIER IN XR - 1
     010311  006457 7470 00        2585         STX7    LAST           STORE LEVEL FOR TLU
     010312  035234 4500 56        2586         STZ     A$WORK,ID      PAD BOTTOM OF STACK WITH A ZERO
     010313  000002 2270 11        2587 IDN1    LDX7    2,1            GET MODE POINTER OF IDENTIFIER
     010314  035216 0670 00        2588         ADX7    T$MODE         MAKE IT ABSOLUTE
     010315  000000 2260 17        2589         LDX6    0,7            GET ACTUAL MODE TYPE IN XR - 6
     010316  017015 1060 03        2590         CMPX6   M$EMPTY,DU     IS IT A BAD MODE
     010317  010365 6000 00        2591         TZE     IDN5           YES - DELETE IT AND CONTINUE
     010320  016773 1060 03        2592         CMPX6   M$MMI,DU       IS IT A MODE MODE INDICATION
     010321  010331 6000 00        2593         TZE     IDNG           YES - JUMP TO LOOK UP MMI
END OF BINARY CARD 00000144
     010322  016762 1060 03        2594         CMPX6   M$REF,DU       IS IT A REFERENCE TYPE MODE
     010323  010355 6010 00        2595         TNZ     IDN4           NO - MUST BE A GOOD MODE SO POP
     010324  000001 2270 17        2596         LDX7    1,7            GET MODE REFERED TO BY REFERENCE
     010325  035216 0670 00        2597         ADX7    T$MODE         MAKE POINTER ABSOLUTE
     010326  000000 2260 17        2598         LDX6    0,7            GET ACTUAL MODE TYPE IN XR - 6
     010327  016773 1060 03        2599         CMPX6   M$MMI,DU       IS IT A MODE MODE INDICATION
     010330  010355 6010 00        2600         TNZ     IDN4           NO - MUST BE A GOOD IDENTIFIER
     010331  035234 6200 56        2601 IDNG    EAX0    A$WORK,ID      GET A WORD IN THE WORKING STACK
     010332  006457 2210 00        2602         LDX1    LAST           GET POINTER TO POINTER TO CURRENT IDENTIFIER
     010333  000000 7410 10        2603         STX1    0,0            AND STORE IN STACK
     010334  006460 2210 00        2604         LDX1    LEVEL          GET CURRENT LEVEL OF IDENTIFIER
     010335  000000 4410 10        2605         SXL1    0,0            AND STORE IN STACK
     010336  000001 7210 17        2606         LXL1    1,7            GET LEVEL OF MODE MODE INDICATION
     010337  006460 7410 00        2607         STX1    LEVEL          AND STORE FOR TABLE LOOK UP
     010340  000001 2210 17        2608         LDX1    1,7            GET POINTER TO DEFINITION FOR MODE INDICATION
     010341  035222 0610 00        2609         ADX1    T$STAB         MAKE IT ABSOLUTE
     010342  000001 0610 03        2610         ADX1    DF,DU          MAKE XR - 1 POINT TO DEFINITION CHAIN
     010343  006437 7000 00        2611 IDN2    TSX0    TLU            LOOK UP MODE OF MODE INDICATION
     010344  010355 7100 00        2612         TRA     IDN4           NOT FOUND SO BAD IDENTIFIER
     010345  000001 2270 11        2613         LDX7    1,1            GET TYPE OF DEFINITION IN XR - 7
     010346  006412 1070 03        2614         CMPX7   D$MODE,DU      IS IT A MODE DEFINITION
     010347  010353 6000 00        2615         TZE     IDN3           YES - GOOD IDENTIFIER SO PREVIOUS IS BAD
END OF BINARY CARD 00000145
     010350  006415 1070 03        2616         CMPX7   D$IDENT,DU     IS IT AN IDENTIFIER DEFINITION
     010351  010355 6010 00        2617         TNZ     IDN4           NO - SO BAD IDENTIFIER
     010352  010313 7100 00        2618         TRA     IDN1           IDENTIFIER SO RECURSE TO SEE IF VALID
     010353  035234 2350 54        2619 IDN3    LDA     A$WORK,DI      DELETE REFERENCE TO LAST IDENTIFIER
```

```
          A                                                 PASS 1,5

     010354  777777 6000 00          2620          TZE      $ERROR           NO MORE SO LOGICAL BUG IN PROGRAM
     010355  035234 2350 54          2621 IDN4     LDA      A$WORK,DI        GET REFERENCE TO LAST IDENTIFIER
     010356  010374 6000 00          2622          TZE      IDN6             NO MORE SO EXIT AS GOOD IDENTIFIER
     010357  000000 6200 05          2623          EAX0     0,AL             GET LEVEL IN XR - 0
     010360  006460 7400 00          2624          STX0     LEVEL            AND STORE FOR TABLE LOOK UP
     010361  000000 6210 01          2625          EAX1     0,AU             GET POINTER TO POINTER TO IDENTIFIER DEFINITION
     010362  006457 7410 00          2626          STX1     LAST             AND STORE IN TABLE LOOK UP ROUTINE
     010363  000000 2210 11          2627          LDX1     0,1              GET POINTER TO IDENTIFIER DEFINITION
     010364  035220 0610 00          2628          ADX1     T$DEF            AND MAKE IT ABSOLUTE
     010365  035234 2340 54          2629 IDN5     SZN      A$WORK,DI        TEST TOP WORD IN STACK
     010366  010401 6000 00          2630          TZE      IDN7             NO MORE SO EXIT AS BAD IDENTIFIER
     010367  035234 0110 56          2631          NOP      A$WORK,ID        RESTORE TALLY WORD
     010370  000000 2210 11          2632          LDX1     0,1              GET POINTER TO FOLLOWING DEFINITION
     010371  006457 7410 51          2633          STX1     LAST,I           AND STORE IN PREVIOUS DEFINITION
     010372  006457 2210 00          2634          LDX1     LAST             GET POINTER TO CONTINUE
     010373  010343 7100 00          2635          TRA      IDN2             AND GO TO TABLE LOOK UP ROUTINE
     010374  010270 0730 00          2636 IDN6     LREG     IDREG            RESTORE REGISTERS
     010375  006460 7400 00          2637          STX0     LEVEL            RESTORE LEVEL
 END OF BINARY CARD 00000146
     010376  006457 7470 00          2638          STX7     LAST             RESTORE LAST
     010377  010260 7160 15          2639          XEC      IDTAB,5          CONTINUE FINDING NEXT STATE
     010400  010226 7100 00          2640          TRA      CLOK             AND LOOP
     010401  010270 0730 00          2641 IDN7     LREG     IDREG            RESTORE REGISTERS
     010402  006460 7400 00          2642          STX0     LEVEL            RESTORE LEVEL
     010403  006457 7470 00          2643          STX7     LAST             RESTORE LAST
     010404  010240 7100 00          2644          TRA      CLDEL            AND DELETE IDENTIFIER DEFINITION FROM TABLE
     010405  000002 0640 03          2645 CL4      ADX4     2,DU             STEP TO NEXT IDENTIFIER
     010406  000000 2340 14          2646          SZN      SC,4             IS THERE ANOTHER IDENTIFIER
     010407  010210 6010 00          2647          TNZ      CL1              TRANSFER TO LOOP IF YES
     010410  035216 2270 00          2648          LDX7     T$MODE           GET ABSOLUTE POINTER TO MODE TABLE
     010411  010637 7000 00          2649          TSX0     XFR              ELIMINATE ALL MMI ENTRIES IN MODE TABLE
     010412  000002 2200 11          2650          LDX0     2,1              CAUSE XFR TO LOAD MODE DEFINITIONS FROM DEF TABLE
     010413  035217 2270 00          2651          LDX7     T$BOUND          GET ABSOLUTE POINTER TO BOUND TABLE
     010414  010637 7000 00          2652          TSX0     XFR              ELIMINATE ALL MMI ENTRIES IN BOUND TABLE
     010415  000002 7200 11          2653          LXL0     2,1              MAKE XFR TO LOAD BOUND DEFINITIONS FROM DEF TABLE
                                     2654
     010416  000001 2200 03          2655          LDX0     1,DU             GET RELATIVE POINTER TO START OF MODE TABLE
     010417  010706 7100 00          2656          TRA      SQ2              AND JUMP TO SQUEEZE MODE TABLE
     010420  035234 4500 56          2657 START    STZ      A$WORK,ID        PUT ZERO ON BOTTOM OF WORKING STACK
     010421  035235 4500 56          2658          STZ      A$STACK,ID       PUT ZERO ON BOTTOM OF CONTROL STACK
     010422  011305 2200 00          2659          LDX0     M1               GET FIRST MODE FOR COMPARE
     010423  035234 7400 56          2660          STX0     A$WORK,ID        AND STORE IN WORKING STACK
 END OF BINARY CARD 00000147
     010424  005754 7170 00          2661          XED      T$WOVF           CHECK FOR STACK OVERFLOW
     010425  011306 2200 00          2662          LDX0     M2               GET SECOND MODE FOR COMPARE
     010426  035234 7400 56          2663          STX0     A$WORK,ID        AND STORE IN WORKING STACK
     010427  005754 7170 00          2664          XED      T$WOVF           CHECK FOR STACK OVERFLOW
     010430  035234 2270 54          2665 POP      LDX7     A$WORK,DI        GET FIRST MODE TO CHECK
     010431  010544 6000 00          2666          TZE      EXIT             NO MORE SO EQUIVELENT MODES
     010432  010630 7000 00          2667          TSX0     XFER             IF MODE REFERS TO ANOTHER - GET ORIGINAL MODE
```

```
          A                                   PASS 1.5

     010433  000000 6210 17               EAX1    0,7            AND PUT IT IN XR - 1
     010434  035234 2270 54               LDX7    A$WORK,DI      GET SECOND MODE TO CHECK
     010435  010630 7000 00               TSX0    XFER           IF MODE REFERS TO ANOTHER - GET ORIGINAL MODE
     010436  000000 6220 17               EAX2    0,7            AND PUT IT IN XR - 2
     010437  010440 7410 00               STX1    *+1            SAVE POINTER TO FIRST MODE
     010440  000000 1020 03               CMPX2   **,DU          ARE THE TWO MODES THE SAME
     010441  010430 6000 00               TZE     POP            YES - CONTINUE TO PROVE EQUIVELENCE
     010442  777777 2350 11               LDA     -1,1           GET LINK WORD OF FIRST MODE
     010443  777777 1150 12               CMPA    -1,2           SEE IF IT IS THE SAME AS THE SECOND MODE LINK
     010444  010550 6010 00               TNZ     NEQ            IF NOT THE SAME THERE IS NO HOPE
     010445  000000 2350 11               LDA     0,1            GET TYPE OF FIRST MODE
     010446  000000 1150 12               CMPA    0,2            IS IT THE SAME TYPE AS THE SECOND MODE
     010447  010550 6010 00               TNZ     NEQ            NO - MODES ARE DIFFERENT
     010450  010451 7410 00               STX1    *+1            STORE POINTER TO FIRST MODE FOR COMPARE
     010451  000000 1020 03               CMPX2   **,DU          IS IT SMALLER THAN SECOND MODE POINTER
 END OF BINARY CARD 00000148
     010452  010456 6050 00               TPL     NSWAP          NO - DO NOT SWAP MODES
     010453  000000 6230 11               EAX3    0,1            SAVE FIRST MODE IN XR - 3
     010454  000000 6210 12               EAX1    0,2            MOVE SECOND MODE TO XR - 1
     010455  000000 6220 13               EAX2    0,3            MOVE FIRST MODE TO XR - 2
     010456  000000 2360 12        NSWAP  LDQ     0,2            GET FIRST WORD OF SECOND MODE
     010457  035216 1610 00               SBX1    T$MODE         MAKE FIRST MODE POINTER RELATIVE
     010460  035216 1620 00               SBX2    T$MODE         MAKE SECOND MODE RELATIVE
     010461  035235 7560 56               STQ     A$STACK,ID     SAVE FOR POSSIBLE RESTORATION
     010462  005742 7170 00               XED     T$SOVF         CHECK FOR STACK OVERFLOW
     010463  035235 7420 56               STX2    A$STACK,ID     SAVE LOCATION OF WORD FOR POSSIBLE RESTORATION
     010464  005742 7170 00               XED     T$SOVF         CHECK FOR STACK OVERFLOW
     010465  035216 0620 00               ADX2    T$MODE         MAKE POINTER TO SECOND MODE ABSOLUTE
     010466  017012 2200 03               LDX0    M$XFER,DU      GET MODE REFERENCE INDICATION
     010467  000000 7400 12               STX0    0,2            AND STORE IN SECOND MODE
     010470  000000 4410 12               SXL1    0,2            STORE POINTER TO TARGET MODE
     010471  035216 0610 00               ADX1    T$MODE         MAKE FIRST MODE POINTER ABSOLUTE
     010472  000000 6200 01               EAX0    0,AU           GET TYPE OF MODE IN XR - 0
     010473  016762 1000 03               CMPX0   M$REF,DU       IS IT REF
     010474  010510 6000 00               TZE     SNGL           YES - CHECK SINGLE POINTER
     010475  016770 1000 03               CMPX0   M$ROW,DU       IS IT ROW
     010476  010510 6000 00               TZE     SNGL           YES - CHECK SINGLE POINTER
     010477  016776 1000 03               CMPX0   M$ROWE,DU      IS IT END ROW
 END OF BINARY CARD 00000149
     010500  010510 6000 00               TZE     SNGL           YES - CHECK SINGLE POINTER
     010501  017007 1000 03               CMPX0   M$UNION,DU     IS IT UNION
     010502  010515 6000 00               TZE     MULT           YES - CHECK LIST OF MODES
     010503  016757 1000 03               CMPX0   M$STRCT,DU     IS IT STRUCTURE
     010504  010515 6000 00               TZE     MULT           YES - CHECK LIST OF MODES
     010505  017001 1000 03               CMPX0   M$PROC,DU      IS IT PROCEDURE
     010506  010515 6000 00               TZE     MULT           YES - CHECK LIST OF MODES
     010507  010550 7100 00               TRA     NEQ            NOT ANYTHING ABOVE SO DIFFERENT MODES
     010510  000001 2270 11        SNGL   LDX7    1,1            GET FIRST MODE TO CHECK
     010511  010534 7000 00               TSX0    PUSH           AND PUSH IT ON THE STACK
     010512  000001 2270 12               LDX7    1,2            GET SECOND MODE TO CHECK
```

```
             A                                            PASS 1,5

      010513  010534 7000 00        2716          TSX0      PUSH             AND PUSH IT ON THE STACK
      010514  010430 7100 00        2717          TRA       POP              CONTINUE PROVING EQUIVELENCE
      010515  777777 2230 11        2718 MULT     LDX3      -1,1             GET NUMBER OF WORDS IN TABLE ENTRY
      010516  000001 1630 03        2719 MULT1    SBX3      1,DU             DECREMENT NUMBER OF MODES LEFT TO CHECK
      010517  010430 6000 00        2720          TZE       POP              ALL DONE - CONTINUE PROVING EQUIVELENCE
      010520  000001 0610 03        2721          ADX1      1,DU             STEP TO NEXT MODE IN MODE 1
      010521  000001 0620 03        2722          ADX2      1,DU             STEP TO NEXT MODE IN MODE 2
      010522  000000 7240 11        2723          LXL4      0,1              GET TAG OF FIRST MODE
      010523  010305 7440 00        2724          STX4      TEMP             AND STORE FOR COMPARE
      010524  000000 7240 12        2725          LXL4      0,2              GET TAG OF SECOND MODE
      010525  010305 1040 00        2726          CMPX4     TEMP             SEE IF TAGS ARE THE SAME
    END OF BINARY CARD 00000150
      010526  010550 6010 00        2727          TNZ       NEQ              NO - FAIL TO PROVE EQUIVELENCE
      010527  000000 2270 11        2728          LDX7      0,1              GET FIRST MODE
      010530  010534 7000 00        2729          TSX0      PUSH             AND PUSH IT ON THE STACK
      010531  000000 2270 12        2730          LDX7      0,2              GET SECOND MODE
      010532  010534 7000 00        2731          TSX0      PUSH             AND PUSH IT ON THE STACK
      010533  010516 7100 00        2732          TRA       MULT1            GO BACK FOR MORE MODES
      010534  010543 7400 00        2733 PUSH     STX0      PUSHX            SAVE RETURN
      010535  035216 1610 00        2734          SBX1      T$MODE           MAKE FIRST MODE POINTER RELATIVE
      010536  035216 1620 00        2735          SBX2      T$MODE           MAKE SECOND MODE POINTER RELATIVE
      010537  035234 7470 56        2736          STX7      A$WORK,ID        PUSH MODE ON THE STACK
      010540  005754 7170 00        2737          XED       T$WOVF           CHECK FOR STACK OVERFLOW
      010541  035216 0610 00        2738          ADX1      T$MODE           MAKE FIRST MODE POINTER RELATIVE
      010542  035216 0620 00        2739          ADX2      T$MODE           MAKE SECOND MODE POINTER RELATIVE
      010543  000000 7100 00        2740 PUSHX    TRA       **               AND RETURN
      010544  035235 2350 54        2741 EXIT     LDA       A$STACK,DI       DELETE A WORD FROM THE STACK
      010545  010674 6000 00        2742          TZE       DONE             TRANSFER IF ALL DONE DELETING
      010546  035235 2350 54        2743          LDA       A$STACK,DI       DELETE A WORD FROM THE STACK
      010547  010544 7100 00        2744          TRA       EXIT             AND LOOP
      010550  035235 2220 54        2745 NEQ      LDX2      A$STACK,DI       GET ADDRESS OF WORD TO RESTORE
      010551  010556 6000 00        2746          TZE       FAIL1            TRANSFER IF NO MORE TO RESTORE
      010552  035235 2360 54        2747          LDQ       A$STACK,DI       GET RESTORED WORD
      010553  035216 0620 00        2748          ADX2      T$MODE           MAKE POINTER ABSOLUTE
    END OF BINARY CARD 00000151
      010554  000000 7560 12        2749          STQ       0,2              RESTORE WORD
      010555  010550 7100 00        2750          TRA       NEQ              AND LOOP
      010556  035234 2350 54        2751 FAIL1    LDA       A$WORK,DI        DELETE A WORD FROM WORK
      010557  010674 6000 00        2752          TZE       DONE             TRANSFER IF ALL DONE
      010560  035234 2350 54        2753          LDA       A$WORK,DI        DELETE A WORD FROM WORK
      010561  010556 7100 00        2754          TRA       FAIL1            AND LOOP
      010562  010627 7400 00        2755 RCHK     STX0      RCHKX            SAVE RETURN
      010563  035214 1660 00        2756          SBX6      T$WORK           MAKE XR - 6 RELATIVE
      010564  035235 4500 56        2757          STZ       A$STACK,ID       MARK CURRENT EXTENT OF CONTROL STACK
      010565  005742 7170 00        2758          XED       T$SOVF           CHECK FOR STACK OVERFLOW
      010566  010630 7000 00        2759 RCHK1    TSX0      A$XFER           MAKE MODE POINTER ABSOLUTE
      010567  000000 2200 17        2760          LDX0      0,7              GET TYPE OF MODE IN XR - 0
      010570  016770 1000 03        2761          CMPX0     M$ROW,DU         IS IT A ROW MODE
      010571  010574 6000 00        2762          TZE       RCHK2            TRANSFER IF ROW MODE
      010572  016776 1000 03        2763          CMPX0     M$ROWE,DU        IS IT AN END ROW MODE
    010573  010601 6010 00           TNZ     RCHK3           TRANSFER IF NOT A ROW MODE
    010574  035235 2340 54   RCHK2   SZN     A$STACK,DI      IS CONTROL STACK MARK DELETED YET
    010575  010574 6010 00           TNZ     RCHK2           TRANSFER IF MORE TO DELETE
    010576  010627 2200 00           LDX0    RCHKX           GET RETURN ADDRESS IN XR - 0
    010577  035214 0660 00           ADX6    T$WORK          MAKE XR - 6 ABSOLUTE
    010600  000001 7100 10           TRA     1,0             AND GIVE ROW RETURN
    010601  016757 1000 03   RCHK3   CMPX0   M$STRCT,DU      IS IT A STRUCTURED MODE
END OF BINARY CARD 00000152
    010602  010622 6010 00           TNZ     RCHK6           NO - TRANSFER TO UNSTACK
    010603  035216 1670 00           SBX7    T$MODE          GET RELATIVE MODE IN XR - 7
    010604  000000 6350 17           EAA     0,7             GET MODE IN AU WITH ZERO IN AL
    010605  035235 7550 56           STA     A$STACK,ID      AND STORE IN CONTROL STACK
    010606  005742 7170 00           XED     T$SOVF          CHECK FOR STACK OVERFLOW
    010607  035235 0540 54   RCHK4   AOS     A$STACK,DI      INCREMENT FIELD IN TOP MODE
    010610  035235 7200 51           LXL0    A$STACK,I       GET POSITION IN XR - 0
    010611  035235 2270 51           LDX7    A$STACK,I       GET MODE IN XR - 7
    010612  035216 0670 00           ADX7    T$MODE          MAKE MODE POINTER ABSOLUTE
    010613  777777 1000 17           CMPX0   -1,7            IS CURRENT POSITION BEYOND END OF STRUCTURE
    010614  010622 6000 00           TZE     RCHK6           TRANSFER IF YES
    010615  010617 7400 00           STX0    RCHK5           STORE POSITION FOR X REGISTER ADDING
    010616  035235 0110 56           NOP     A$STACK,ID      RESTORE TOP WORD IN CONTROL STACK
    010617  000000 0670 03   RCHK5   ADX7    **,DU           GET POINTER TO CURRENT FIELD IN XR - 7
    010620  000000 2270 17           LDX7    0,7             GET MODE OF CURRENT FIELD IN XR - 7
    010621  010566 7100 00           TRA     RCHK1           AND SEE IF THIS MODE IS A ROW TYPE MODE
    010622  035235 2340 54   RCHK6   SZN     A$STACK,DI      CHECK CONTENTS OF TOP OF CONTROL STACK
    010623  010626 6000 00           TZE     RCHK7           TRANSFER IF ALL UNSTACKED
    010624  035235 0110 56           NOP     A$STACK,ID      RESTORE TOP WORD IN CONTROL STACK
    010625  010607 7100 00           TRA     RCHK4           GO CONTINUE CHECKING
    010626  035214 0660 00   RCHK7   ADX6    T$WORK          MAKE XR - 6 ABSOLUTE
    010627  000000 7100 00   RCHKX   TRA     **              AND RETURN AS NO ROW MODE
END OF BINARY CARD 00000153
    010630  010634 7400 00   XFER    STX0    XFERX           SAVE RETURN
    010631  035216 0670 00   XFER1   ADX7    T$MODE          GET ABSOLUTE POINTER TO MODE
    010632  000000 2200 17           LDX0    0,7             GET TYPE OF MODE
    010633  017012 1000 03           CMPX0   M$XFER,DU       DOES IT REFER TO ANOTHER MODE
    010634  000000 6010 00   XFERX   TNZ     **              NO - EXIT
    010635  000000 7270 17           LXL7    0,7             GET MODE THAT IS REFERED TO
    010636  010631 7100 00           TRA     XFER1           AND LOOP
    010637  010666 7400 00   XFR     STX0    XFRX            SAVE RETURN
    010640  000001 0670 03   XFR1    ADX7    1,DU            SET XR - 7 TO POINT TO THE NEXT TABLE ENTRY
    010641  777777 2340 17           SZN     -1,7            SEE IF THERE IS A NEXT ENTRY
    010642  010672 6000 00           TZE     XFR6            TRANSFER IF ALL DONE
    010643  000000 2200 17           LDX0    0,7             GET TYPE OF TABLE ENTRY IN XR - 0
    010644  016773 1000 03           CMPX0   M$MMI,DU        IS IT A MODE MODE INDICATION
    010645  010670 6010 00           TNZ     XFR5            TRANSFER IF NOT A MMI
    010646  000001 2210 17           LDX1    1,7             GET IDENTIFIER OF MMI IN XR - 1
    010647  035222 0610 00           ADX1    T$STAB          MAKE IT ABSOLUTE
    010650  000001 0610 03           ADX1    DF,DU           MAKE IT POINT ABSOLUTELY TO DEFINITION CHAIN
    010651  000001 7200 17           LXL0    1,7             GET LEVEL OF IDENTIFIER IN XR - 0
    010652  006460 7400 00           STX0    LEVEL           AND STORE AS PARAMETER TO TLU ROUTINE
```

```
               A                                               PASS 1,5

      010653  006437 7000 00       2812 XFR2    TSX0     TLU               SEARCH FOR IDENTIFIER IN SYMBOL TABLE
      010654  010660 7100 00       2813         TRA      XFR3              NOT FOUND - BAD ENTRY IN TABLE
      010655  000001 2200 11       2814         LDX0     1,1               GET TYPE OF SYMBOL TABLE ENTRY IN XR - 0
   END OF BINARY CARD 00000154
      010656  006412 1000 03       2815         CMPX0    D$MODE,DU         IS IT A MODE DEFINITION
      010657  010664 6000 00       2816         TZE      XFR4              YES - JUMP TO INSERT XFER
      010660  017015 2200 03       2817 XFR3    LDX0     M$EMPTY,DU        GET A BAD ENTRY FLAG IN XR - 0
      010661  000000 7400 17       2818         STX0     0,7               AND STORE IT IN TABLE ENTRY TO REPLACE MMI
      010662  000000 4470 17       2819         SXL7     0,7               STORE UNIQUE NUMBER IN LOWER HALF AS BAD MODE
      010663  010670 7100 00       2820         TRA      XFR5              AND GO CHECK NEXT ENTRY IN TABLE
      010664  017012 2200 03       2821 XFR4    LDX0     M$XFER,DU         GET A TRANSFER IN XR - 0
      010665  000000 7400 17       2822         STX0     0,7               STORE IN TABLE ENTRY TO REPLACE MMI
      010666  000000 7160 00       2823 XFRX    XEC      **                GET POINTER TO EQUIVELENT ENTRY
      010667  000000 4400 17       2824         SXL0     0,7               STORE IN TABLE ENTRY
      010670  777777 0670 17       2825 XFR5    ADX7     -1,7              STEP XR - 7 OVER CURRENT ENTRY
      010671  010640 7100 00       2826         TRA      XFR1              AND LOOP
      010672  010666 2200 00       2827 XFR6    LDX0     XFRX              GET RETURN ADDRESS IN XR - 0
      010673  000001 7100 10       2828         TRA      1,0               AND RETURN AFTER EXECUTED INSTRUCTION

      010674  011306 2200 00       2829 DONE    LDX0     M2                GET RELATIVE POINTER TO MODE 2
      010675  035216 0600 00       2830         ADX0     T$MODE            MAKE IT ABSOLUTE
      010676  777777 0600 10       2831 SQ1     ADX0     -1,0              STEP TO NEXT TABLE ENTRY
      010677  000001 0600 03       2832         ADX0     1,DU              STEP ACROSS LINK WORD
      010700  035216 1600 00       2833         SBX0     T$MODE            MAKE POINTER RELATIVE
      010701  011306 7400 00       2834         STX0     M2                AND STORE AS MODE 2
      010702  035216 0600 00       2835         ADX0     T$MODE            GET ABSOLUTE POINTER TO MODE 2
      010703  777777 2340 10       2836         SZN      -1,0              SEE IF AT END OF MODE TABLE
   END OF BINARY CARD 00000155
      010704  010420 6010 00       2837         TNZ      START             ATTEMPT TO PROVE EQUIVELENCE
      010705  011305 2200 00       2838         LDX0     M1                GET RELATIVE POINTER TO MODE 1
      010706  035216 0600 00       2839 SQ2     ADX0     T$MODE            MAKE IT ABSOLUTE
      010707  777777 0600 10       2840         ADX0     -1,0              STEP TO NEXT TABLE ENTRY
      010710  000001 0600 03       2841         ADX0     1,DU              STEP ACROSS LINK WORD
      010711  035216 1600 00       2842         SBX0     T$MODE            MAKE POINTER RELATIVE
      010712  011305 7400 00       2843         STX0     M1                AND STORE AS MODE 1
      010713  035216 0600 00       2844         ADX0     T$MODE            GET ABSOLUTE POINTER TO MODE 1
      010714  777777 2340 10       2845         SZN      -1,0              SEE IF AT END OF MODE TABLE
      010715  010676 6010 00       2846         TNZ      SQ1               JUMP TO SET MODE 2
      010716  000003 6360 00       2847         EAQ      3                 FILE REFERENCE NUMBER
      010717  000000 2350 07       2848         LDA      0,DL              DESIRED POINTER SETTING
      010720  003767 7000 00       2849         TSX0     $SETP             AND RESET POINTER IN SOURCE FILE
      010721  001677 4500 00       2850         STZ      A$EOF             RESET EOF FLAG
      010722  002152 4500 00       2851         STZ      A$PEEKF           INITIALIZE INPUT ROUTINE
      010723  002153 4500 00       2852         STZ      A$QEEKF           INITIALIZE INPUT ROUTINE
      010724  001674 2350 00       2853         LDA      A$INI             GET INITIAL INPUT ROUTINE TALLY
      010725  001675 7550 00       2854         STA      A$IN              AND INITIALIZE INPUT ROUTINE
      010726  035220 2210 00       2855         LDX1     T$DEF             GET POINTER TO START OF DEFINITION TABLE
      010727  000001 0610 03       2856 LBL1    ADX1     1,DU              STEP OVER ENTRY HEADER WORD
      010730  000001 2220 11       2857         LDX2     1,1               GET TYPE OF DEFINITION IN XR - 2
      010731  006415 1020 03       2858         CMPX2    D$IDENT,DU        IS IT AN IDENTIFIER DEFINITION
   END OF BINARY CARD 00000156
```

```
         A                                        PASS 1,5

    010732  010741 6010 00      2859          TNZ     LBL3          TRANSFER IF NOT
    010733  000002 2220 11      2860          LDX2    2,1           GET MODE OF IDENTIFIER IN XR - 2
    010734  000031 1020 03      2861          CMPX2   M$LBL,DU      IS IT A LABEL DEFINITION
    010735  010741 6010 00      2862          TNZ     LBL3          TRANSFER IF NOT
    010736  016464 7200 00      2863 LBL2     LXL0    2$GLBL        GET NEXT LABEL IN XR - 0
    010737  016464 0540 00      2864          AOS     2$GLBL        INCREMENT LABEL GENERATOR
    010740  000003 7400 11      2865          STX0    3,1           STORE UNIQUE LABEL IN DEFINITION
    010741  777777 0610 11      2866 LBL3     ADX1    -1,1          STEP TO NEXT DEFINITION
    010742  000000 2340 11      2867          SZN     0,1           SEE IF THERE ARE ANY MORE DEFINITIONS
    010743  010727 6010 00      2868          TNZ     LBL1          TRANSFER IF THERE ARE MORE
    010744  035216 2210 00      2869          LDX1    T$MODE        GET POINTER TO BASE OF MODE TABLE
    010745  000001 0610 03      2870 UP1      ADX1    1,DU          STEP OVER HEADER WORD
    010746  777777 2220 11      2871          LDX2    -1,1          GET NUMBER OF WORDS IN THIS MODE TABLE ENTRY
    010747  010760 6000 00      2872          TZE     SETMS         EXIT IF END OF MODE TABLE IS REACHED
    010750  000001 0610 03      2873 UP2      ADX1    1,DU          STEP OVER MODE TYPE WORD
    010751  000001 1620 03      2874          SBX2    1,DU          DECREMENT NUMBER OF FIELDS LEFT TO UPDATE IN MODE
    010752  010745 6000 00      2875          TZE     UP1           TRANSFER IF FINISHED WITH CURRENT MODE
    010753  000000 2270 11      2876          LDX7    0,1           GET MODE OF FIELD IN XR - 7
    010754  010630 7000 00      2877          TSX0    A$XFER        MAKE MODE POINTER UNIQUE AND ABSOLUTE
    010755  035216 1670 00      2878          SBX7    T$MODE        MAKE MODE POINTER RELATIVE
    010756  000000 7470 11      2879          STX7    0,1           AND RESTORE UNIQUE MODE OF FIELD
    010757  010750 7100 00      2880          TRA     UP2           AND LOOP
END OF BINARY CARD 00000157
    010760  000037 2270 03      2881 SETMS    LDX7    M$PTR,DU      GET POINTER TO FIRST NONPRIMITIVE MODE
    010761  011305 7470 00      2882 SETML    STX7    M1            STORE CURRENT MODE IN M1
    010762  010773 7000 00      2883          TSX0    SETM          SET LENGTH AND TYPE FIELDS IN CURRENT MODE
    010763  011305 2270 00      2884          LDX7    M1            GET POINTER TO CURRENT MODE
    010764  035216 0670 00      2885          ADX7    T$MODE        MAKE MODE POINTER ABSOLUTE
    010765  777777 0670 17      2886          ADX7    -1,7          STEP CURRENT MODE POINTER TO NEXT MODE
    010766  000001 0670 03      2887          ADX7    1,DU          STEP OVER LINK WORD
    010767  777777 2200 17      2888          LDX0    -1,7          GET LENGTH OF NEXT ENTRY IN MODE TABLE
    010770  006210 6000 00      2889          TZE     A$OK          EXIT IF NO MORE ENTRIES IN MODE TABLE
    010771  035216 1670 00      2890          SBX7    T$MODE        MAKE MODE POINTER ABSOLUTE
    010772  010761 7100 00      2891          TRA     SETML         AND LOOP
    010773  000000 6230 10      2892 SETM     EAX3    0,0           SAVE RETURN IN ANOTHER REGISTER
    010774  010630 7000 00      2893          TSX0    A$XFER        MAKE CURRENT MODE UNIQUE AND ABSOLUTE
    010775  000000 7210 17      2894          LXL1    0,7           GET TYPE POINTER FOR MODE
    010776  777777 7220 17      2895          LXL2    -1,7          GET LENGTH OF VALUE OF CURRENT MODE
    010777  000000 6010 13      2896          TNZ     0,3           RETURN IF ALREADY SET
    011000  000000 2200 17      2897          LDX0    0,7           GET TYPE OF MODE IN XR - 0
    011001  035216 1670 00      2898          SBX7    T$MODE        MAKE MODE POINTER RELATIVE
    011002  016762 1000 03      2899          CMPX0   M$REF,DU      SEE IF IT IS A REFERENCE TYPE MODE
    011003  011021 6000 00      2900          TZE     REF           YES - DO IT
    011004  017001 1000 03      2901          CMPX0   M$PROC,DU     IS IT A PROCEDURE MODE
    011005  011077 6000 00      2902          TZE     PROC          YES - DO IT
END OF BINARY CARD 00000158
    011006  016757 1000 03      2903          CMPX0   M$STRCT,DU    IS IT A STRUCTURED MODE
    011007  011122 6000 00      2904          TZE     STR           YES - DO IT
    011010  017007 1000 03      2905          CMPX0   M$UNION,DU    IS IT A UNITED MODE
    011011  011225 6000 00      2906          TZE     UNION         YES - DO IT
```

```
      011012  016770 1000 03      2907        CMPX0    M$ROW,DU        IS IT A ROW MODE
      011013  011074 6000 00      2908        TZE      ROW             YES - DO IT
      011014  016776 1000 03      2909        CMPX0    M$ROWE,DU       IS IT THE END OF A ROW MODE
      011015  011074 6000 00      2910        TZE      ROW             YES - DO IT
      011016  017015 1000 03      2911        CMPX0    M$EMPTY,DU      IS IT A BAD MODE
      011017  000000 6000 13      2912        TZE      0,3             YES - RETURN
      011020  777777 7100 00      2913        TRA      $ERROR          NO - BAD MODE ####(ROW)####
      011021  011072 4500 00      2914 REF    STZ      REFT            INITIALIZE DIMENSION COUNT
      011022  011073 7470 00      2915        STX7     REFM            SAVE MODE POINTER
      011023  035216 0670 00      2916        ADX7     T$MODE          MAKE MODE POINTER ABSOLUTE
      011024  000001 2270 17      2917 REF1   LDX7     1,7             GET DEREFERENCED MODE IN XR - 7
      011025  010630 7000 00      2918 ROW1   TSX0     A$XFER          MAKE MODE POINTER ABSOLUTE
      011026  000000 2200 17      2919        LDX0     0,7             GET TYPE OF MODE IN XR - 0
      011027  016770 1000 03      2920        CMPX0    M$ROW,DU        IS IT A ROW MODE
      011030  011033 6010 00      2921        TNZ      REF2            TRANSFER IF NOT MIDDLE OF ROW MODE
      011031  011072 0540 00      2922        AOS      REFT            INCREMENT DIMENSION COUNT
      011032  011024 7100 00      2923        TRA      REF1            AND LOOP
      011033  016776 1000 03      2924 REF2   CMPX0    M$ROWE,DU       IS IT THE LAST ROW IN A ROW MODE
 END OF BINARY CARD 00000159
      011034  011036 6010 00      2925        TNZ      REF3            TRANSFER IF NOT
      011035  011072 0540 00      2926        AOS      REFT            INCREMENT DIMENSION COUNT
      011036  035227 2210 03      2927 REF3   LDX1     T$TYPE,DU       GET POINTER TO TYPE TABLE CONTROL WORD
      011037  000001 6350 00      2928        EAA      1               ALLOCATE ONE WORD
      011040  011072 2340 00      2929        SZN      REFT            SEE IF REFERENCE TO ROW MODE
      011041  011043 6000 00      2930        TZE      REF4            TRANSFER IF NOT
      011042  000002 0750 03      2931        ADA      2,DU            ALLOCATE THREE WORDS
      011043  005663 7000 00      2932 REF4   TSX0     T$ALOC          ALLOCATE MEMORY IN TYPE TABLE
      011044  011072 2340 00      2933        SZN      REFT            SEE IF REFERENCE TO ROW MODE
      011045  011061 6000 00      2934        TZE      REF5            TRANSFER IF NOT
      011046  027221 2350 03      2935        LDA      T$ROW,DU        GET ROW TYPE IN A
      011047  011072 0750 00      2936        ADA      REFT            ADD DIMENSION IN AL
      011050  000001 7550 11      2937        STA      1,1             AND STORE AS SECOND TYPE WORD
      011051  027215 2350 03      2938        LDA      T$PTR,DU        GET POINTER TYPE IN A
      011052  000000 7550 11      2939        STA      0,1             AND STORE AS FIRST TYPE WORD
      011053  011072 2350 00      2940        LDA      REFT            GET NUMBER OF DIMENSIONS IN AL
      011054  000002 7350 00      2941        ALS      2               GET DESCRIPTOR LENGTH IN AL
      011055  027222 0750 03      2942        ADA      T$SKIP,DU       ADD SKIP TYPE
      011056  000002 7550 11      2943        STA      2,1             AND STORE IN TYPE TABLE
      011057  000002 6220 05      2944        EAX2     2,AL            GET LENGTH OF VALUE IN XR - 2
      011060  011064 7100 00      2945        TRA      REF6            AND CONTINUE
      011061  000001 2220 03      2946 REF5   LDX2     1,DU            GET LENGTH OF VALUE IN XR - 2
 END OF BINARY CARD 00000160
      011062  027215 2350 03      2947        LDA      T$PTR,DU        GET TYPE IN AU
      011063  000000 7550 11      2948        STA      0,1             AND STORE IN TYPE TABLE
      011064  035227 1610 00      2949 REF6   SBX1     T$TYPE          MAKE TYPE TABLE POINTER RELATIVE
      011065  011073 2270 00      2950        LDX7     REFM            GET POINTER TO REFERENCE MODE IN XR - 7
      011066  035216 0670 00      2951        ADX7     T$MODE          MAKE POINTER ABSOLUTE
      011067  000000 4410 17      2952        SXL1     0,7             STORE TYPE IN MODE TABLE
      011070  777777 4420 17      2953        SXL2     -1,7            STORE VALUE LENGTH IN MODE TABLE
      011071  000000 7100 13      2954        TRA      0,3             AND RETURN
```

```
    011072  000000 000000        2955 REFT    ZERO
    011073  000000 000000        2956 REFM    ZERO
    011074  011072 4500 00       2957 ROW     STZ      REFT             CLEAR DIMENSION COUNTER
    011075  011073 7470 00       2958         STX7     REFM             SAVE CURRENT MODE
    011076  011025 7100 00       2959         TRA      ROW1             GO GENERATE TYPE
    011077  035227 2210 03       2960 PROC    LDX1     T$TYPE,DU        GET POINTER TO TYPE TABLE CONTROL WORD
    011100  000004 6350 00       2961         EAA      4                GET NUMBER OF WORDS NEEDED IN AU
    011101  005663 7000 00       2962         TSX0     T$ALOC           AND ALLOCATE WORDS IN TYPE TABLE
    011102  000000 6220 11       2963         EAX2     0,1              GET POINTER TO NEW TYPE TABLE ENTRY IN XR - 2
    011103  011116 6240 00       2964         EAX4     PROCT            GET POINTER TO PROCEDURE TYPE IN XR - 4
    011104  000000011007
    011105  011600 5602 01       2965         RPD      4,1              MOVE
    011106  000000 2350 14       2966         LDA      0,4              FROM PROTOTYPE TYPE
    011107  000000 7550 12       2967         STA      0,2              TO NEW TYPE ENTRY
END OF BINARY CARD 00000161
    011110  035227 1610 00       2968         SBX1     T$TYPE           MAKE POINTER TO NEW TYPE ENTRY RELATIVE
    011111  000004 2220 03       2969         LDX2     4,DU             GET LENGTH OF VALUE IN XR - 2
    011112  035216 0670 00       2970         ADX7     T$MODE           MAKE MODE TABLE POINTER ABSOLUTE
    011113  000000 4410 17       2971         SXL1     0,7              STORE TYPE IN MODE TABLE
    011114  777777 4420 17       2972         SXL2     -1,7             STORE VALUE LENGTH IN MODE TABLE
    011115  000000 7100 13       2973         TRA      0,3              AND RETURN
    011116  027217 000000        2974 PROCT   ZERO     T$OCT
    011117  027217 000000        2975         ZERO     T$OCT
    011120  027215 000000        2976         ZERO     T$PTR
    011121  027215 000000        2977         ZERO     T$PTR
    011122  035235 7430 51       2978 STR     STX3     A$STACK,I        SAVE RETURN IN CONTROL STACK
    011123  035234 2200 00       2979         LDX0     A$WORK           GET POINTER TO END OF WORKING STACK
    011124  035214 1600 00       2980         SBX0     T$WORK           SUBTRACT BASE TO GET LENGTH OF WORKING STACK
    011125  035235 4400 56       2981         SXL0     A$STACK,ID       AND STORE (RETURN, MARK) = -3
    011126  005742 7170 00       2982         XED      T$SOVF           CHECK FOR STACK OVERFLOW
    011127  000000 6350 17       2983         EAA      0,7              GET MODE IN AU
    011130  000022 7710 00       2984         ARL      18               MOVE MODE TO AL
    011131  000001 0750 03       2985         ADA      1,DU             ADD A ONE IN AU
    011132  035235 7550 56       2986         STA      A$STACK,ID       AND STORE (COUNT, MODE) = -2
    011133  005742 7170 00       2987         XED      T$SOVF           CHECK FOR STACK OVERFLOW
    011134  035235 4500 56       2988         STZ      A$STACK,ID       STORE (VALUE LEN, TYPE LEN) = -1
    011135  005742 7170 00       2989         XED      T$SOVF           CHECK FOR STACK OVERFLOW
END OF BINARY CARD 00000162
    011136  035235 2200 00       2990         LDX0     A$STACK          GET POINTER TO END OF CONTROL STACK IN XR - 0
    011137  777776 7270 10       2991         LXL7     -2,0             GET MODE IN XR - 7
    011140  035216 0670 00       2992         ADX7     T$MODE           MAKE MODE POINTER ABSOLUTE
    011141  777776 0670 10       2993 STR1    ADX7     -2,0             ADD FIELD NUMBER TO GET POINTER TO FIELD
    011142  000000 2270 17       2994         LDX7     0,7              GET MODE OF FIELD IN XR - 7
    011143  010773 7000 00       2995         TSX0     SETM             GET TYPE AND LENGTH OF FIELD MODE
    011144  000000 6350 11       2996         EAA      0,1              GET TYPE IN AU
    011145  035234 7550 56       2997         STA      A$WORK,ID        AND STORE FIELD TYPE IN WORKING STACK
    011146  005754 7170 00       2998         XED      T$WOVF           CHECK FOR STACK OVERFLOW
    011147  035235 2200 00       2999         LDX0     A$STACK          GET POINTER TO END OF CONTROL STACK IN XR - 0
    011150  777777 0420 10       3000         ASX2     -1,0             INCREMENT STRUCTURE VALUE LENGTH BY FIELD LENGTH
    011151  777777 7220 10       3001         LXL2     -1,0             GET TYPE LENGTH FOR STRUCTURE IN XR - 2
```

```
              A                                                    PASS 1.5

      011152  035227 0610 00         3002          ADX1     T$TYPE            GET ABSOLUTE POINTER TO FIELD TYPE IN XR - 1
      011153  777777 0620 11         3003          ADX2     -1,1              INCREMENT STRUCTURE TYPE LENGTH BY FIELD TYPE
      011154  777777 4420 10         3004          SXL2     -1,0              RESTORE STRUCTURE TYPE LENGTH
      011155  777776 2210 10         3005          LDX1     -2,0              GET FIELD NUMBER IN XR - 1
      011156  000001 0610 03         3006          ADX1     1,DU              STEP TO NEXT FIELD
      011157  777776 7410 10         3007          STX1     -2,0              AND RESTORE FIELD NUMBER
      011160  777776 7270 10         3008          LXL7     -2,0              GET MODE OF STRUCTURE IN XR - 7
      011161  035216 0670 00         3009          ADX7     T$MODE            MAKE MODE POINTER ABSOLUTE
      011162  777777 1010 17         3010          CMPX1    -1,7              COMPARE TO SEE IF ALL FIELDS HAVE BEEN CONSIDERED
      011163  011141 6010 00         3011          TNZ      STR1              TRANSFER IF MORE FIELDS TO CONSIDER
   END OF BINARY CARD 00000163
      011164  777777 2350 10         3012          LDA      -1,0              GET LENGTH OF STRUCTURE TYPE IN AL
      011165  000000 6350 05         3013          EAA      0,AL              GET TYPE LENGTH IN AU
      011166  035227 2210 03         3014          LDX1     T$TYPE,DU         GET POINTER TO TYPE TABLE CONTROL WORD
      011167  005663 7000 00         3015          TSX0     T$ALOC            ALLOCATE SPACE FOR TYPE IN TYPE TABLE
      011170  035235 2200 00         3016          LDX0     A$STACK           GET POINTER TO END OF CONTROL STACK
      011171  777776 7270 10         3017          LXL7     -2,0              GET MODE OF STRUCTURE IN XR - 7
      011172  035216 0670 00         3018          ADX7     T$MODE            MAKE MODE POINTER ABSOLUTE
      011173  035227 1610 00         3019          SBX1     T$TYPE            GET RELATIVE STRUCTURE TYPE IN XR - 1
      011174  000000 4410 17         3020          SXL1     0,7               AND STORE TYPE IN MODE
      011175  777777 2220 10         3021          LDX2     -1,0              GET LENGTH OF STRUCTURED VALUE IN XR - 2
      011176  777777 4420 17         3022          SXL2     -1,7              AND STORE IN MODE
      011177  000000 6250 11         3023          EAX5     0,1               GET POINTER TO TYPE IN XR - 5
      011200  035227 0650 00         3024          ADX5     T$TYPE            MAKE TYPE POINTER ABSOLUTE
      011201  777775 7230 10         3025          LXL3     -3,0              GET LENGTH OF WORKING STACK BEFORE STRUCTURE
      011202  035214 0630 00         3026          ADX3     T$WORK            GET POINTER TO FIRST WORD PUSHED IN WORK
      011203  000000 6260 13         3027          EAX6     0,3               SAVE POINTER TO FIRST WORD PUSHED IN WORK
      011204  000000 2240 13         3028 STR2     LDX4     0,3               GET TYPE OF NEXT FIELD IN XR - 4
      011205  035227 0640 00         3029          ADX4     T$TYPE            MAKE TYPE POINTER ABSOLUTE
      011206  777777 2350 14         3030          LDA      -1,4              GET LENGTH OF FIELD TYPE IN AU
      011207  000010 7710 00         3031          ARL      8                 POSITION COUNT FOR REPEAT
      011210  001400 6200 05         3032          EAX0     768,AL            PUT COUNT WITH A AND B BITS IN XR - 0
      011211  000000 5602 01         3033          RPDX     ,1                MOVE
   END OF BINARY CARD 00000164
      011212  000000 2360 14         3034          LDQ      0,4               FROM FIELD TYPE
      011213  000000 7560 15         3035          STQ      0,5               TO STRUCTURE TYPE
      011214  000001 0630 03         3036          ADX3     1,DU              STEP TO NEXT FIELD
      011215  035234 1030 00         3037          CMPX3    A$WORK            CHECK IF ANY FIELDS ARE LEFT
      011216  011204 6010 00         3038          TNZ      STR2              TRANSFER IF THERE ARE ANY MORE FIELDS
      011217  035234 0110 54         3039 STR3     NOP      A$WORK,DI         DELETE A WORD FROM THE WORKING STACK
      011220  035234 1060 00         3040          CMPX6    A$WORK            SEE IF DELETED BACK TO THE MARK
      011221  011217 6010 00         3041          TNZ      STR3              TRANSFER IF MORE WORDS TO DELETE
      011222  035235 0110 54         3042          NOP      A$STACK,DI        DELETE A WORD FROM THE CONTROL STACK
      011223  035235 0110 54         3043          NOP      A$STACK,DI        DELETE A WORD FROM THE CONTROL STACK
      011224  035235 7100 55         3044          TRA      A$STACK,DIC       AND RETURN
      011225  035227 2210 03         3045 UNION    LDX1     T$TYPE,DU         GET POINTER TO TYPE TABLE CONTROL WORD
      011226  000002 6350 00         3046          EAA      2                 GET NUMBER OF WORDS NEEDED IN TYPE TABLE
      011227  005663 7000 00         3047          TSX0     T$ALOC            ALLOCATE TWO WORDS IN TYPE TABLE
      011230  011304 2350 00         3048          LDA      UNT1              GET FIRST WORD OF UNION TYPE
      011231  000000 7550 11         3049          STA      0,1               AND STORE IN TYPE TABLE
```

```
             A                                          PASS 1.5

      011232  035227 1610 00     3050        SBX1    T$TYPE          MAKE TYPE TABLE ENTRY POINTER RELATIVE
      011233  035216 0670 00     3051        ADX7    T$MODE          MAKE MODE POINTER ABSOLUTE
      011234  000000 4410 17     3052        SXL1    0,7             AND STORE TYPE IN MODE TABLE ENTRY
      011235  035216 1670 00     3053        SBX7    T$MODE          MAKE MODE POINTER RELATIVE
      011236  000000 6350 13     3054        EAA     0,3             GET RETURN ADDRESS IN AU
      011237  000022 7710 00     3055        ARL     18              MOVE TO AL
   END OF BINARY CARD 00000165
      011240  035235 7550 56     3056        STA     A$STACK,ID      AND STORE (VALUE LEN, RETURN) = -2
      011241  005742 7170 00     3057        XED     T$SOVF          CHECK FOR STACK OVERFLOW
      011242  000000 6350 17     3058        EAA     0,7             GET MODE IN AU
      011243  000022 7710 00     3059        ARL     18              MOVE MODE TO AL
      011244  000001 0750 03     3060        ADA     1,DU            SET CURRENT FIELD TO ONE
      011245  035235 7550 56     3061        STA     A$STACK,ID      AND STORE (COUNT, MODE) = -1
      011246  005742 7170 00     3062        XED     T$SOVF          CHECK FOR STACK OVERFLOW
      011247  035235 2200 00     3063        LDX0    A$STACK         GET POINTER TO END OF CONTROL STACK
      011250  777777 7270 10     3064        LXL7    -1,0            GET MODE POINTER IN XR - 7
      011251  035216 0670 00     3065        ADX7    T$MODE          MAKE POINTER ABSOLUTE
      011252  777777 0670 10     3066 UN1    ADX7    -1,0            ADD CURRENT FIELD NUMBER
      011253  000000 2270 17     3067        LDX7    0,7             GET MODE OF CURRENT FIELD IN XR - 7
      011254  010773 7000 00     3068        TSX0    SETM            GET LENGTH OF FIELD MODE
      011255  035235 2200 00     3069        LDX0    A$STACK         GET POINTER TO END OF CONTROL STACK
      011256  777776 1020 10     3070        CMPX2   -2,0            SEE IF THIS FIELD HAS CURRENT MAXIMUM LENGTH
      011257  011261 6020 00     3071        TNC     UN2             TRANSFER IF NOT BIGGEST YET
      011260  777776 7420 10     3072        STX2    -2,0            SET NEW MAXIMUM
      011261  777777 2210 10     3073 UN2    LDX1    -1,0            GET CURRENT FIELD NUMBER
      011262  000001 0610 03     3074        ADX1    1,DU            STEP TO NEXT FIELD
      011263  777777 7410 10     3075        STX1    -1,0            AND RESTORE AS NEW CURRENT FIELD
      011264  777777 7270 10     3076        LXL7    -1,0            GET MODE OF UNION IN XR - 7
      011265  035216 0670 00     3077        ADX7    T$MODE          MAKE MODE POINTER ABSOLUTE
   END OF BINARY CARD 00000166
      011266  777777 1010 17     3078        CMPX1   -1,7            COMPARE FIELD NUMBER AGAINST NUMBER IN MODE
      011267  011252 6010 00     3079        TNZ     UN1             TRANSFER IF MORE FIELDS TO CONSIDER
      011270  000000 7210 17     3080        LXL1    0,7             GET TYPE POINTER FOR UNION IN XR - 1
      011271  035227 0610 00     3081        ADX1    T$TYPE          MAKE TYPE TABLE POINTER ABSOLUTE
      011272  777776 2220 10     3082        LDX2    -2,0            GET MAXIMUM LENGTH OF UNION MODE
      011273  000001 4420 11     3083        SXL2    1,1             AND STORE IN TYPE TABLE ENTRY
      011274  027222 2200 03     3084        LDX0    T$SKIP,DU       GET A SKIP TYPE IN XR - 0
      011275  000001 7400 11     3085        STX0    1,1             AND STORE IN TYPE TABLE ENTRY
      011276  035227 1610 00     3086        SBX1    T$TYPE          MAKE TYPE TABLE POINTER RELATIVE
      011277  000001 0620 03     3087        ADX2    1,DU            ADD ONE FOR LENGTH/TYPE WORD IN UNION
      011300  777777 4420 17     3088        SXL2    -1,7            AND STORE LENGTH OF VALUE IN MODE TABLE
      011301  035235 0110 54     3089        NOP     A$STACK,DI      DELETE A WORD FROM THE CONTROL STACK
      011302  035235 2350 54     3090        LDA     A$STACK,DI      GET RETURN ADDRESS IN AL
      011303  000000 7100 05     3091        TRA     0,AL            AND RETURN
      011304  027220 000000      3092 UNT1   ZERO    T$ULEN
      011305  000000 000000      3093 M1     ZERO
      011306  000000 000000      3094 M2     ZERO
```

```
                                  3095        TTLS      PASS 2              PASS 2
                                  3096        HEAD      2
    011307  005164 7000 00        3097 START  TSX0      A$PEEK              PEEK AT NEXT INPUT SYMPOL
    011310  006216 7100 00        3098        TRA       A$NOMAT             NO MATCH IF END OF FILE
    011311  000046 1070 03        3099        CMPX7     S$LPAR,DU           IS NEXT SYMBOL A LEFT PARENTHESIS
    011312  006216 6010 00        3100        TNZ       A$NOMAT             NO - GO TO NO MATCH
    011313  000050 6350 00        3101        EAA       M$PROCV             GET A PROCEDURE VOID MODE IN AU
 END OF BINARY CARD 00000167
    011314  016460 7550 00        3102        STA       DECLR               AND STORE AS CURRENT MODE
    011315  013257 7550 00        3103        STA       FMODE               STORE AS CURRENT DECLARATION MODE
    011316  000157 2210 03        3104        LDX1      D$BLANK,DU          GET POINTER TO BLANK DEFINITION
    011317  011731 7410 00        3105        STX1      LASTS               AND STORE AS LAST IDENTIFIER ENCOUNTERED
    011320  011322 7500 00        3106        STC2      LIBF                THIS IS A PROGRAM AND NOT A SUBROUTINE
    011321  014424 7100 00        3107        TRA       OIDN                GO SET UP DECLARATION HEADER
    011322  000000 000000         3108 LIBF   ZERO
    011323  011357 2200 00        3109 SRNGE  LDX0      RNGE                GET CURRENT RANGE
    011324  035235 7400 56        3110        STX0      A$STACK,ID          AND SAVE IN CONTROL STACK
    011325  005742 7170 00        3111        XED       T$SOVF              CHECK FOR STACK OVERFLOW
    011326  011360 2350 00        3112        LDA       SRTAL               GET TALLY WORD FOR SAVING RANGE INFORMATION
    011327  011361 7550 00        3113        STA       SRT                 AND STORE
    011330  011361 2350 20        3114 SR1    LDA       SRT,*               GET NEXT WORD TO BE SAVED
    011331  035235 7550 56        3115        STA       A$STACK,ID          AND SAVE IN THE CONTROL STACK
    011332  005742 7170 00        3116        XED       T$SOVF              CHECK FOR STACK OVERFLOW
    011333  011361 4500 57        3117        STZ       SRT,IDC             ZERO OUT SAVED WORD AND STEP TO NEXT ONE
    011334  011330 6070 00        3118        TTF       SR1                 TRANRFER IF THERE ARE MORE WORDS TO BE SAVED
    011335  016457 4500 00        3119 XRNG1  STZ       CNT                 INITIALIZE COUNT FOR NEW RANGE
    011336  011357 7200 00        3120        LXL0      RNGE                GET MAXIMUM NUMBER OF RANGE ENTERED
    011337  035221 0600 00        3121        ADX0      T$PROG              MAKE IT ABSOLUTE
    011340  777777 0600 10        3122        ADX0      -1,0                STEP TO NEXT RANGE NUMBER
    011341  000001 0600 03        3123        ADX0      1,DU                STEP OVER LINK WORD
 END OF BINARY CARD 00000168
    011342  035221 1600 00        3124        SBX0      T$PROG              MAKE IT RELATIVE
    011343  011357 7400 00        3125        STX0      RNGE                STORE AS CURRENT RANGE
    011344  011357 4400 00        3126        SXL0      RNGE                STORE AS MAXIMUM RANGE
    011345  006210 7100 00        3127        TRA       A$OK                AND EXIT
    011346  011335 7100 00        3128 XRNGE  TRA       XRNG1               GO TO EXCHANGE RANGE ROUTINE
    011347  011362 2350 00        3129 ERNGE  LDA       ERTAL               GET TALLY WORD FOR RESTORING RANGE INFORMATION
    011350  011363 7550 00        3130        STA       ERT                 AND STORE
    011351  035235 2350 54        3131 ER1    LDA       A$STACK,DI          RESTORE WORD FROM CONTROL STACK
    011352  011363 7550 55        3132        STA       ERT,DIC             AND STORE IN MEMORY
    011353  011351 6070 00        3133        TTF       ER1                 TRANSFER IF MORE WORDS ARE TO BE RESTORED
    011354  035235 2200 54        3134        LDX0      A$STACK,DI          GET PREVIOUS RANGE NUMBER
    011355  011357 7400 00        3135        STX0      RNGE                STORE AS CURRENT RANGE NUMBER
    011356  006210 7100 00        3136        TRA       A$OK                AND EXIT
    011357  000000 000000         3137 RNGE   ZERO
    011360  011364 0012 51        3138 SRTAL  TALLYC    RTAB,RTABE-RTAB,I
    011361  000000 000000         3139 SRT    ZERO
    011362  011376 7766 51        3140 ERTAL  TALLYC    RTABE,RTAB-RTABE,I
    011363  000000 000000         3141 ERT    ZERO
    011364  016457 000000         3142 RTAB   ZERO      CNT
```

```
          2                                                PASS 2

     011365  016460 000000         3143            ZERO     DFCLR
     011366  013257 000000         3144            ZERO     FMODE
     011367  013653 000000         3145            ZERO     B1X
 END OF BINARY CARD 00000169
     011370  013654 000000         3146            ZERO     B1S
     011371  013656 000000         3147            ZERO     B2X
     011372  013657 000000         3148            ZERO     B2S
     011373  013660 000000         3149            ZERO     L1
     011374  016475 000000         3150            ZERO     OPT
     011375  013450 000000         3151            ZERO     SBCNT
                           011376  3152 RTABE      EQU      *
                           011376  3153            BSS      10
     011410  005164 7000 00        3154 TAG        TSX0     A$PEEK           PEEK AT NEXT INPUT SYMBOL
     011411  006216 7100 00        3155            TRA      A$NOMAT          NO MORE - NO MATCH
     011412  000112 1070 03        3156            CMPX7    A$TABLE,DU       CAN IT BE A TAG
     011413  006216 6020 00        3157            TNC      A$NOMAT          NO - TOO EARLY IN SYMBOL TABLE
     011414  002152 4500 00        3158            STZ      A$PEEKF          ACCEPT SYMBOL
     011415  011423 7470 00        3159            STX7     TG               STORE ACCEPTED TAG
     011416  000000 6200 17        3160            EAX0     0,7              GET TAG IN XR - 0
     011417  035222 0600 00        3161            ADX0     T$STAB           MAKE SYMBOL POINTER ABSOLUTE
     011420  000000 2350 10        3162            LDA      A$SC,0           GET STRING TALLY WORD FOR SYMBOL
     011421  006554 0110 00        3163            NOP      1$XXS            AND PRINT IT
     011422  006210 7100 00        3164            TRA      A$OK             AND EXIT
     011423  000000 000000         3165 TG         ZERO
     011424  006413 2200 03        3166 OPER       LDX0     D$OP,DU          GET OPERATOR TYPE OF ENTRY
     011425  011730 7400 00        3167            STX0     TLUT             AND SAVE IN TEMP
     011426  000000 2200 03        3168            LDX0     S$ERM1,DU        GET A POINTER TO OPERATOR MESSAGE
 END OF BINARY CARD 00000170
     011427  006436 7400 00        3169            STX0     A$MATT           AND STORE FOR POSSIBLE ERROR MESSAGE
     011430  011674 7100 00        3170            TRA      TLUG             AND SEE IF IT IS AN OPERATOR
     011431  006412 2200 03        3171 MIND       LDX0     D$MODE,DU        GET MODE TYPE OF ENTRY
     011432  011730 7400 00        3172            STX0     TLUT             AND SAVE IN TEMP
     011433  000002 2200 03        3173            LDX0     S$ERM2,DU        GET A POINTER TO INDICATION MESSAGE
     011434  006436 7400 00        3174            STX0     A$MATT           AND STORE FOR POSSIBLE ERROR MESSAGE
     011435  011674 7100 00        3175            TRA      TLUG             AND SEE IF IT IS A MODE
     011436  006415 2200 03        3176 IDENT      LDX0     D$IDENT,DU       GET IDENTIFIER TYPE OF ENTRY
     011437  011730 7400 00        3177            STX0     TLUT             AND SAVE IN TEMP
     011440  000004 2200 03        3178            LDX0     S$ERM3,DU        GET A POINTER TO IDENTIFIER MESSAGE
     011441  006436 7400 00        3179            STX0     A$MATT           AND STORE FOR POSSIBLE ERROR MESSAGE
     011442  011674 7100 00        3180            TRA      TLUG             AND SEE IF IT IS AN IDENTIFIER
     011443  000006 2200 03        3181 DENOT      LDX0     S$ERM4,DU        GET A POINTER TO DENOTATION MESSAGE
     011444  006436 7400 00        3182            STX0     A$MATT           AND STORE FOR POSSIBLE ERROR MESSAGE
     011445  005164 7000 00        3183            TSX0     A$PEEK           PEEK AT NEXT INPUT SYMBOL
     011446  006216 7100 00        3184            TRA      A$NOMAT          NO MORE SO FAIL
     011447  000000 6200 16        3185            EAX0     0,6              GET POINTER TO IDENTIFIER WITHOUT LONGS
     011450  000112 1000 00        3186            CMPX0    A$TABLE          IS IT IN PERMANENT PART OF TABLE
     011451  006216 6020 00        3187            TNC      A$NOMAT          TRANSFER IF YES - CANNOT BE DENOTATION
     011452  035222 0600 00        3188            ADX0     T$STAB           MAKE POINTER ABSOLUTE
     011453  000000 2350 10        3189            LDA      A$SC,0           GET STRING POINTER FOR SYMBOL
     011454  011650 7550 00        3190            STA      DENT             AND STORE IN MEMORY
```

```
END OF BINARY CARD 00000171
     011455  035223 2200 00       3191         LDX0    T$ITAB          GET RELOCATION OF IDENTIFIER TABLE
     011456  011650 0400 00       3192         ASX0    DENT            MAKE TALLY WORD ABSOLUTE
     011457  011650 2350 50       3193         LDA     DENT,CI         GET FIRST CHARACTER OF STRING
     011460  000042 1150 07       3194         CMPA    =0042,DL        IS IT A QUOTE SYMBOL
     011461  011563 6000 00       3195         TZE     DST             YES - GO EVALUATE STRING
     011462  000044 1150 07       3196         CMPA    =0044,DL        IS IT A DOLLAR SIGN
     011463  011563 6000 00       3197         TZE     DST             YES - GO EVALUATE STRING
     011464  000056 1150 07       3198         CMPA    =0056,DL        IS IT A PERIOD
     011465  011472 6000 00       3199         TZE     DNUM            YES - GO EVALUATE NUMBER
     011466  000060 1350 07       3200         SBLA    =0060,DL        SUBTRACT AN ASCII ZERO
     011467  000012 1150 07       3201         CMPA    10,DL           IS IT A DIGIT
     011470  011472 6020 00       3202         TNC     DNUM            YES - GO EVALUATE NUMBER
     011471  006216 7100 00       3203         TRA     A$NOMAT         NO - NOT A DENOTATION
     011472  400000 4310 03       3204 DNUM    FLD     =0400000,DU     GET A FLOATING ZERO IN EAQ REGISTER
     011473  011640 7570 00       3205         STAQ    DN              ZERO OUT NUMBER ACCUMULATION LOCATION
     011474  011621 7000 00       3206         TSX0    DCNV            CONVERT NUMBER TO NEXT NONDIGIT
     011475  011553 7100 00       3207         TRA     DINT            INTEGER IS CONVERTED IN (E, N)
     011476  011644 4500 00       3208         STZ     DCNT            ZERO OUT FRACTION DIGIT COUNTER
     011477  000060 0750 07       3209         ADA     =0060,DL        RESTORE LAST CHARACTER READ IN A REGISTER
     011500  000056 1150 07       3210         CMPA    =0056,DL        IS IT A PERIOD
     011501  011505 6010 00       3211         TNZ     DNUM1           TRANSFER IF NOT A DECIMAL POINT
     011502  011621 7000 00       3212         TSX0    DCNV            CONVERT NUMBER TO NEXT NONDIGIT
END OF BINARY CARD 00000172
     011503  011537 7100 00       3213         TRA     DNUM5           TRANSFER IF NO FOLLOWING EXPONENT
     011504  000060 0750 07       3214         ADA     =0060,DL        RESTORE LAST CHARACTER READ IN A REGISTER
     011505  000105 1150 07       3215 DNUM1   CMPA    =0105,DL        IS CHARACTER THE LETTER E
     011506  777777 6010 00       3216         TNZ     $ERROR          NO - ILLEGAL NUMBER
     011507  011643 4500 00       3217         STZ     ESGN            ZERO OUT EXPONENT SIGN
     011510  000000 2360 07       3218         LDQ     0,DL            INITIALIZE Q FOR ACCUMULATION OF EXPONENT
     011511  011650 2350 52       3219         LDA     DENT,SC         GET NEXT CHARACTER AFTER LETTER E
     011512  011514 6070 00       3220         TTF     DNUM2           TRANSFER IF THERE IS A CHARACTER
     011513  777777 7100 00       3221         TRA     $ERROR          NO MORE - ILLEGAL NUMBER
     011514  000053 1150 07       3222 DNUM2   CMPA    =0053,DL        IS IT A PLUS SIGN
     011515  011530 6000 00       3223         TZE     DNUM4           YES - GO CONVERT EXPONENT
     011516  000055 1150 07       3224         CMPA    =0055,DL        IS IT A MINUS SIGN
     011517  011522 6010 00       3225         TNZ     DNUM3           TRANSFER IF NOT - MUST BE A DIGIT
     011520  011643 0540 00       3226         AOS     ESGN            SET MINUS FLAG IN FLAG
     011521  011530 7100 00       3227         TRA     DNUM4           AND GO CONVERT EXPONENT
     011522  000060 1750 07       3228 DNUM3   SBA     =0060,DL        GET VALUE OF ASCII DIGIT IN A
     011523  000012 1150 07       3229         CMPA    10,DL           CHECK TO SEE IF IT IS A DIGIT
     011524  777777 6030 00       3230         TRC     $ERROR          TRANSFER IF NOT - ILLEGAL NUMBER
     011525  011646 7550 00       3231         STA     DT              SAVE DIGIT IN MEMORY
     011526  000012 4020 07       3232         MPY     10,DL           MULTIPLY ACCUMULATED EXPONENT BY TEN
     011527  011646 0760 00       3233         ADQ     DT              AND ADD NEW DIGIT
     011530  011650 2350 52       3234 DNUM4   LDA     DENT,SC         GET NEXT DIGIT OF EXPONENT
END OF BINARY CARD 00000173
     011531  011522 6070 00       3235         TTF     DNUM3           TRANSFER IF THERE IS ONE
     011532  000144 1160 07       3236         CMPQ    100,DL          IS EXPONENT IN REASONABLE RANGE
     011533  777777 6030 00       3237         TRC     $ERROR          NO - ILLEGAL NUMBER
```

PASS 2

```
   011534  011643 2340 00       3238          SZN     ESGN           IS EXPONENT NEGATIVE
   011535  011537 6000 00       3239          TZE     DNUM5          TRANSFER IF POSITIVE
   011536  000000 5330 00       3240          NEGL                   NEGATE EXPONENT
   011537  011644 1760 00       3241 DNUM5    SBQ     DCNT           SUBTRACT NUMBER OF DIGITS AFTER DECIMAL POINT
   011540  000000 6200 06       3242          EAX0    0,QL           GET EFFECTIVE EXPONENT IN XR - 0
   011541  011640 2370 00       3243          LDAQ    DN             MAKE EAQ REGISTER CONTAIN INTEGER OF NUMBER
   011542  000000 6200 10       3244          EAX0    0,0            CHECK SIGN OF EXPONENT
   011543  011560 6000 00       3245 DNUM6    TZE     DREAL          REAL IS CONVERTED IN EAQ
   011544  011550 6040 00       3246          TMI     DNUM7          TRANSFER IF NEGATIVE EXPONENT
   011545  010500 4610 03       3247          FMP     =10.0,DU       MULTIPLY NUMBER BY TEN
   011546  777777 6200 10       3248          EAX0    -1,0           DECREMENT EXPONENT BY ONE
   011547  011543 7100 00       3249          TRA     DNUM6          AND LOOP
   011550  011642 5670 00       3250 DNUM7    DFDV    TEN            DIVIDE NUMBER BY TEN
   011551  000001 6200 10       3251          EAX0    1,0            INCREMENT EXPONENT BY ONE
   011552  011543 7100 00       3252          TRA     DNUM6          AND LOOP
   011553  011640 2370 00       3253 DINT     LDAQ    DN             GET MANTISSA OF CONVERTED NUMBER
   011554  216000 4350 03       3254          UFA     =71B25,DU      FIX NUMBER IN AQ REGISTER
   011555  011640 7560 00       3255          STQ     DN             AND STORE LSH AS VALUE OF DENOTATION
   011556  000007 6350 00       3256          EAA     M$INT          GET MODE OF DENOTATION
END OF BINARY CARD 00000174
   011557  011603 7100 00       3257          TRA     DEND           AND GO TO CLEANUP ROUTINE
   011560  011640 4570 00       3258 DREAL    DFST    DN             STORE CONVERTED VALUE OF DENOTATION
   011561  000011 6350 00       3259          EAA     M$REAL         GET MODE OF DENOTATION
   011562  011603 7100 00       3260          TRA     DEND           AND GO TO CLEANUP ROUTINE
   011563  011650 0110 52       3261 DST      NOP     DENT,SC        STEP STRING POINTER OVER QUOTE OR DOLLAR SIGN
   011564  011650 2350 00       3262          LDA     DENT           GET STRING TALLY WORD IN A REGISTER
   011565  011640 7550 00       3263          STA     DN             AND STORE AS FIRST PART OF VALUE
   011566  035223 3200 00       3264          LCX0    T$ITAB         GET MINUS BASE OF IDENTIFIER TABLE IN XR - 0
   011567  011640 0400 00       3265          ASX0    DN             MAKE TALLY WORD RELATIVE
   011570  000006 7710 00       3266          ARL     6              MOVE TALLY TO AL
   011571  000001 1750 07       3267          SBA     1,DL           DECREMENT TALLY BY ONE TO MAKE ACCURATE
   011572  007777 3750 07       3268          ANA     =07777,DL      GET CLEAN NUMBER OF CHARACTERS IN STRING
   011573  011641 7550 00       3269          STA     DN+1           AND STORE AS SECOND PART OF VALUE
   011574  000001 1150 07       3270          CMPA    1,DL           SEE IF THERE IS ONE CHARACTER IN THE STRING
   011575  011602 6010 00       3271          TNZ     DST1           TRANSFER IF DIFFERENT FROM ONE
   011576  000005 6350 00       3272          EAA     M$CHAR         GET MODE OF VALUE
   011577  011650 2360 50       3273          LDQ     DENT,CI        GET VALUE OF CHARACTER IN Q
   011600  011640 7560 00       3274          STQ     DN             AND STORE AS VALUE OF DENOTATION
   011601  011603 7100 00       3275          TRA     DEND           AND GO TO CLEANUP ROUTINE
   011602  000053 6350 00       3276 DST1     EAA     M$STRNG        GET MODE OF VALUE
   011603  011647 7550 00       3277 DEND     STA     DM             STORE MODE OF VALUE
   011604  000005 2350 03       3278          LDA     5,DU           GET NUMBER OF WORDS IN LIST ELEMENT
END OF BINARY CARD 00000175
   011605  035220 2210 03       3279          LDX1    T$DEF,DU       GET POINTER TO TABLE CONTROL WORD IN XR - 1
   011606  005663 7000 00       3280          TSX0    T$ALOC         ALLOCATE MEMORY IN THE DEFINITION TABLE
   011607  006416 6350 00       3281          EAA     D$DENOT        GET TYPE OF LIST ELEMENT IN A
   011610  000001 7550 11       3282          STA     1,1            AND STORE IN LIST ELEMENT
   011611  006520 7470 00       3283          STX7    1$IDNT         STORE POINTER TO IDENTIFIER FOR SETD
   011612  007721 7000 00       3284          TSX0    1$SETD         LINK THIS TABLE ENTRY IN DEFINITION CHAIN
   011613  011647 2350 00       3285          LDA     DM             GET MODE OF CURRENT DENOTATION
```

```
                2                                        PASS 2

       011614  000002 7550 11        3286        STA     2,1                AND STORE IN TABLE ENTRY
       011615  011640 2370 00        3287        LDAQ    DN                 GET VALUE OF DENOTATION
       011616  000003 7550 11        3288        STA     3,1                AND STORE IN TABLE ENTRY
       011617  000004 7560 11        3289        STQ     4,1                STORE LOW HALF IN TABLE ENTRY
       011620  011722 7100 00        3290        TRA     PAROK              GO ACCEPT SYMBOL
       011621  011650 2350 52        3291 DCNV   LDA     DENT,SC            GET NEXT CHARACTER OF NUMBER
       011622  011625 6070 00        3292        TTF     DCNV1              TRANSFER IF THERE IS A CHARACTER
       011623  000000 2360 07        3293        LDQ     0,DL               INITIALIZE Q REGISTER TO ZERO
       011624  000000 7100 10        3294        TRA     0,0                AND RETURN WITH STRING EXHAUSTED RETURN
       011625  000060 1750 07        3295 DCNV1  SBA     =0060,DL           GET VALUE OF DIGIT IN A
       011626  000012 1150 07        3296        CMPA    10,DL              IS CHARACTER A DIGIT
       011627  000001 6030 10        3297        TRC     1,0                RETURN IF NOT A DIGIT
       011630  011644 0540 00        3298        AOS     DCNT               INCREMENT DIGIT COUNTER
       011631  011645 7510 01        3299        STCA    D,01               STORE DIGIT IN LOW PART OF WORD
       011632  011640 2370 00        3300        LDAQ    DN                 RESTORE MANTISSA OF ACCUMULATED NUMBER
    END OF BINARY CARD 00000176
       011633  010500 4610 03        3301        FMP     =10.0,DU           MULTIPLY BY TEN
       011634  011645 4750 00        3302        FAD     D                  ADD NEW DIGIT
       011635  011640 7570 00        3303        STAQ    DN                 SAVE MANTISSA
       011636  011621 7100 00        3304        TRA     DCNV               AND LOOP
       011637  000000011007
                            011640   3305        EVEN
       011640  000000000000          3306 DN     OCT     0,0
       011641  000000000000

       011642  010500000000          3307 TEN    DEC     10,0
       011643  000000 000000         3308 ESGN   ZERO
       011644  000000 000000         3309 DCNT   ZERO
       011645  066000000000          3310 D      VFD     8/27
       011646  000000 000000         3311 DT     ZERO
       011647  000000 000000         3312 DM     ZERO
       011650  000000 000000         3313 DENT   ZERO
       011651  011673 7400 00        3314 LK     STX0    LKX                SAVE RETURN
       011652  000001 7200 11        3315        LXL0    1,1                SEE IF DEFINITION ALREADY CHAINED IN LEVEL CHAIN
       011653  011656 6000 00        3316        TZE     LK1                TRANSFER IF NOT ALREADY CHAINED
       011654  035220 1610 00        3317        SBX1    T$DEF              MAKE DEFINITION POINTER RELATIVE FOR RETURN
       011655  011673 7100 00        3318        TRA     LKX                AND RETURN
       011656  016470 2230 03        3319 LK1    LDX3    LINK0,DU           GET POINTER TO LL0 CHAIN POINTER
       011657  000000 7200 11        3320        LXL0    0,1                CHECK LL OF DEFINITION
       011660  011663 6000 00        3321        TZE     LK2                TRANSFER IF LL0
    END OF BINARY CARD 00000177
       011661  000002 6230 10        3322        EAX3    2,0                GET RELATIVE POINTER TO LEVEL CHAIN POINTER
       011662  035221 0630 00        3323        ADX3    T$PROG             MAKE POINTER ABSOLUTE
       011663  000000 7200 13        3324 LK2    LXL0    0,3                CHECK IF AT END OF CHAIN
       011664  011670 6040 00        3325        TMI     LK3                TRANSFER IF AT END OF CHAIN
       011665  000001 6230 10        3326        EAX3    1,0                GET POINTER TO FOLLOWING ELEMENT IN XR - 3
       011666  035220 0630 00        3327        ADX3    T$DEF              MAKE POINTER ABSOLUTE
       011667  011663 7100 00        3328        TRA     LK2                AND LOOP TO TRY AGAIN
       011670  000001 4400 11        3329 LK3    SXL0    1,1                STORE END CHAIN POINTER IN CURRENT DEFINITION
       011671  035220 1610 00        3330        SBX1    T$DEF              GET RELATIVE POINTER TO CURRENT DEFINITION
```

```
     011673  000000 7100 00        3332 LKX     TRA      **                AND RETURN
     011674  011357 2200 00        3333 TLUG    LDX0     RNGE              GET CURRENT RANGE NUMBER
     011675  006460 7400 00        3334         STX0     A$LEVEL           STORE FOR TABLE LOOK UP ROUTINE
     011676  005164 7000 00        3335         TSX0     A$PEEK            PEEK AT NEXT INPUT SYMBOL
     011677  006216 7100 00        3336         TRA      A$NOMAT           NO MORE SO FAIL
     011700  002153 2200 00        3337         LDX0     A$QEEKF           GET A LOOK AT FOLLOWING SYMBOL
     011701  000064 1000 03        3338         CMPX0    S$OF,DU           IS IT THE WORD OF
     011702  006216 6000 00        3339         TZE      A$NOMAT           YES - CURRENT SYMBOL IS A FIELD SELECTOR
     011703  000000 6210 17        3340         EAX1     0,7               GET SYMBOL IN XR - 1
     011704  035222 0610 00        3341         ADX1     T$STAR            MAKE IT ABSOLUTE
     011705  000001 0610 03        3342         ADX1     A$DF,DU           MAKE IT POINT TO CHAIN OF DEFINITIONS
     011706  006437 7000 00        3343         TSX0     A$TLU             LOOK UP SYMBOL IN SYMBOL TABLE
END OF BINARY CARD 00000178
     011707  006216 7100 00        3344         TRA      A$NOMAT           NO MORE SO FAIL
     011710  000001 2200 11        3345         LDX0     1,1               GET TYPE OF ENTRY IN XR - 0
     011711  011730 1000 00        3346         CMPX0    TLUT              SEE IF IT IS OF THE REQUIRED TYPE
     011712  006216 6010 00        3347         TNZ      A$NOMAT           NO - FAIL
     011713  006415 1000 03        3348         CMPX0    D$IDENT,DU        IS THIS AN IDENTIFIER
     011714  011717 6010 00        3349         TNZ      TLUG2             NO - TRANSFER
     011715  011651 7000 00        3350         TSX0     LK                LINK DEFINITION TO CHAIN FOR CURRENT LL
     011716  011720 7100 00        3351         TRA      TLUG3             AND CONTINUE
     011717  035220 1610 00        3352 TLUG2   SBX1     T$DEF             MAKE DEFINITION POINTER RELATIVE
     011720  011731 7410 00        3353 TLUG3   STX1     LASTS             AND STORE
     011721  011423 7470 00        3354         STX7     TG                SAVE LAST SYMBOL READ AS TAG
     011722  002152 4500 00        3355 PAROK   STZ      A$PEEKF           ACCEPT SYMBOL
     011723  000000 6200 17        3356         EAX0     0,7               GET POINTER TO SYMBOL IN XR - 0
     011724  035222 0600 00        3357         ADX0     T$STAR            MAKE POINTER ABSOLUTE
     011725  000000 2350 10        3358         LDA      A$SC,0            GET TALLY FOR IDENTIFIER SUCCESSFULLY MATCHED
     011726  006554 0110 00        3359         NOP      1$XXS             PRINT
     011727  006210 7100 00        3360         TRA      A$OK              AND GIVE SUCCESSFUL RETURN
     011730  000000 000000         3361 TLUT    ZERO
     011731  000000 000000         3362 LASTS   ZERO
     011732  000000 6200 05        3363 LPAR    EAX0     0,AL              GET PARAMETER IN XR - 0
     011733  012025 7400 00        3364         STX0     LPARP             AND STORE
     011734  000046 2200 03        3365         LDX0     S$LPAR,DU         GET A POINTER TO LEFT PARENTHESIS
END OF BINARY CARD 00000179
     011735  006436 7400 00        3366         STX0     A$MATT            AND STORE FOR POSSIBLE ERROR MESSAGE
     011736  005164 7000 00        3367 TPCHK   TSX0     A$PEEK            PEEK AT NEXT INPUT SYMBOL
     011737  006216 7100 00        3368         TRA      A$NOMAT           NO MORE SO FAIL
     011740  006436 1070 00        3369         CMPX7    A$MATT            SEE IF NEXT SYMBOL MATCHES
     011741  006216 6010 00        3370         TNZ      A$NOMAT           NO - FAIL
     011742  012025 2200 00        3371         LDX0     LPARP             GET TYPE OF PARENTHESIS NEEDED
     011743  012005 6000 00        3372         TZE      CHK10             TRANSFER IF ANY TYPE IS ACCEPTABLE
     011744  011357 7200 00        3373         LXL0     RNGE              GET POINTER TO CURRENT RANGE IN XR - 0
     011745  011747 6010 00        3374         TNZ      *+2               TRANSFER IF NOT ZERO
     011746  777772 2200 03        3375         LDX0     -6,DU             SET XR - 0 TO POINT TO RANGE MINUS ONE
     011747  035221 0600 00        3376         ADX0     T$PROG            MAKE IT ABSOLUTE
     011750  777777 2220 03        3377         LDX2     -1,DU             GET ALL BITS IN XR - 2
     011751  000007 7210 10        3378         LXL1     7,0               GET TYPE OF RANGE IN XR - 1
     011752  011755 6050 00        3379         TPL      CHK1              TRANSFER IF NOT A PROCEDURE DENOTATION
```

```
      011753  000004 1620 03        3380          SBX2     PDEN,DU              RESET PROCEDURE DENOTATION BIT IN XR - 2
      011754  011764 7100 00        3381          TRA      CHK3                 AND TRANSFER
      011755  006530 1010 03        3382 CHK1     CMPX1    1$BARR,DU            SEE IF RANGE IS TERMINATED BY A BAR OR BARF
      011756  011761 6010 00        3383          TNZ      CHK2                 TRANSFER IF NOT A BAR RANGE
      011757  000001 1620 03        3384          SBX2     FB,DU                RESET BAR BIT IN XR - 2
      011760  011764 7100 00        3385          TRA      CHK3                 AND TRANSFER
      011761  006531 1010 03        3386 CHK2     CMPX1    1$CLOR,DU            SEE IF RANGE IS TERMINATED BY )
      011762  777777 6010 00        3387          TNZ      $ERROR               TRANSFER IF NOT - COMPILER ERROR
   END OF BINARY CARD 00000180
      011763  000002 1620 03        3388          SBX2     FP,DU                RESET PARENTHESIS BIT IN XR - 2
      011764  000013 2210 10        3389 CHK3     LDX1     7+4,0                GET NUMBER OF COLONS IN RANGE
      011765  011770 6010 00        3390          TNZ      CHK4                 TRANSFER IF ANY COLONS
      011766  000040 1620 03        3391          SBX2     ZCOL,DU              RESET ZERO COLONS BIT IN XR - 2
      011767  011771 7100 00        3392          TRA      CHK5                 AND TRANSFER
      011770  000100 1620 03        3393 CHK4     SBX2     MCOL,DU              RESET MANY COLONS BIT IN XR - 2
      011771  000014 2210 10        3394 CHK5     LDX1     7+5,0                GET NUMBER OF COMMAS IN RANGE
      011772  011775 6010 00        3395          TNZ      CHK6                 TRANSFER IF ANY COMMAS
      011773  000010 1620 03        3396          SBX2     ZCOM,DU              RESET ZERO COMMAS BIT IN XR - 2
      011774  011776 7100 00        3397          TRA      CHK7                 AND TRANSFER
      011775  000020 1620 03        3398 CHK6     SBX2     MCOM,DU              RESET MANY COMMAS BIT IN XR - 2
      011776  000012 2210 10        3399 CHK7     LDX1     7+3,0                GET NUMBER OF SEMICOLONS IN RANGE
      011777  012002 6010 00        3400          TNZ      CHK8                 TRANSFER IF ANY SEMICOLONS
      012000  000200 1620 03        3401          SBX2     ZSEM,DU              RESET ZERO SEMICOLONS BIT IN XR - 2
      012001  012003 7100 00        3402          TRA      CHK9                 AND TRANSFER
      012002  000400 1620 03        3403 CHK8     SBX2     MSEM,DU              RESET MANY SEMICOLONS BIT IN XR - 2
      012003  012025 3020 00        3404 CHK9     CANX2    LPARP                SEE IF RANGE IS ACCEPTABLE
      012004  006216 6010 00        3405          TNZ      A$NOMAT              TRANSFER IF NOT ACCEPTABLE
      012005  002152 4500 00        3406 CHK10    STZ      A$PEEKF              ACCEPT SYMBOL
      012006  000000 6200 17        3407          EAX0     0,7                  GET POINTER TO SYMBOL IN XR - 0
      012007  035222 0600 00        3408          ADX0     T$STAB               MAKE POINTER ABSOLUTE
      012010  000000 2350 10        3409          LDA      A$SC,0               GET TALLY FOR IDENTIFIER SUCCESSFULLY MATCHED
   END OF BINARY CARD 00000181
      012011  006554 0110 00        3410          NOP      1$XXS                PRINT
      012012  006210 7100 00        3411          TRA      A$OK                 AND GIVE SUCCESSFUL RETURN
      012013  000000 6200 05        3412 BAR      EAX0     0,AL                 GET PARAMETER IN XR - 0
      012014  012025 7400 00        3413          STX0     LPARP                AND STORE
      012015  000062 2200 03        3414          LDX0     S$BAR,DU             GET A POINTER TO BAR
      012016  006436 7400 00        3415          STX0     A$MATT               AND STORE FOR POSSIBLE ERROR MESSAGE
      012017  011736 7100 00        3416          TRA      TPCHK                AND SEE IF NEXT SYMBOL IS OF PROPER TYPE
      012020  000000 6200 05        3417 BARF     EAX0     0,AL                 GET PARAMETER IN XR - 0
      012021  012025 7400 00        3418          STX0     LPARP                AND STORE
      012022  000110 2200 03        3419          LDX0     S$BARF,DU            GET A POINTER TO BARF
      012023  006436 7400 00        3420          STX0     A$MATT               AND STORE FOR POSSIBLE ERROR MESSAGE
      012024  011736 7100 00        3421          TRA      TPCHK                AND SEE IF NEXT SYMBOL IS OF PROPER TYPE
      012025  000000 000000         3422 LPARP    ZERO
      012026  011357 2200 00        3423 CPAR     LDX0     RNGE                 GET POINTER TO CURRENT RANGE CONTROL ENTRY
      012027  035221 0600 00        3424          ADX0     T$PROG               MAKE POINTER ABSOLUTE
      012030  000005 2210 10        3425          LDX1     5,0                  GET NUMBER OF COMMAS IN RANGE
      012031  006216 6000 00        3426          TZE      A$NOMAT              FAIL IF NO COMMAS
      012032  000004 2210 10        3427          LDX1     4,0                  GET NUMBER OF COLONS IN RANGE
```

```
     012033  006216 6010 00        3428         TNZ      A$NOMAT          FAIL IF ANY COLONS
     012034  000003 2210 10        3429         LDX1     3,0              GET NUMBER OF SEMICOLONS IN RANGE
     012035  006216 6010 00        3430         TNZ      A$NOMAT          FAIL IF ANY SEMICOLONS
     012036  006210 7100 00        3431         TRA      A$OK             SUCCEED IF PARALLEL CONSTRUCTION
 END OF BINARY CARD 00000182
     012037  000010 2200 03        3432 TAGOF   LDX0     S$ERM5,DU        GET A POINTER TO TAG OF MESSAGE
     012040  006436 7400 00        3433         STX0     A$MATT           AND STORE FOR POSSIBLE ERROR MESSAGE
     012041  005164 7000 00        3434         TSX0     A$PEEK           PEEK AT NEXT TWO INPUT SYMBOLS
     012042  006216 7100 00        3435         TRA      A$NOMAT          NO MORE SO FAIL
     012043  002153 2200 00        3436         LDX0     A$QEEKF          GET FOLLOWING INPUT SYMBOL
     012044  000064 1000 03        3437         CMPX0    S$OF,DU          IS IT THE SYMBOL OF
     012045  006216 6010 00        3438         TNZ      A$NOMAT          NO - FAIL
     012046  035235 7470 56        3439         STX7     A$STACK,ID       STORE TAG IN CONTROL STACK
     012047  005742 7170 00        3440         XED      T$SOVF           CHECK FOR STACK OVERFLOW
     012050  000000 6200 17        3441         EAX0     0,7              GET TAG IN XR - 0
     012051  035222 0600 00        3442         ADX0     T$STAB           MAKE POINTER ABSOLUTE
     012052  000000 2350 10        3443         LDA      A$SC,0           GET TALLY WORD FOR TAG IN A
     012053  006554 0110 00        3444         NOP      1$XXS            AND PRINT IT
     012054  002152 4500 00        3445         STZ      A$PEEKF          ACCEPT TAG
     012055  002153 2200 00        3446         LDX0     A$QEEKF          GET OF SYMBOL
     012056  035222 0600 00        3447         ADX0     T$STAB           MAKE IT ABSOLUTE
     012057  000000 2350 10        3448         LDA      A$SC,0           GET TALLY WORD FOR OF IN A
     012060  006554 0110 00        3449         NOP      1$XXS            AND PRINT IT
     012061  002153 4500 00        3450         STZ      A$QEEKF          ACCEPT OF
     012062  006210 7100 00        3451         TRA      A$OK             AND EXIT
     012063  005164 7000 00        3452 LABEL   TSX0     A$PEEK           PEEK AT NEXT TWO INPUT SYMBOLS
     012064  006216 7100 00        3453         TRA      A$NOMAT          NO MORE SO FAIL
 END OF BINARY CARD 00000183
     012065  002153 2200 00        3454         LDX0     A$QEEKF          GET FOLLOWING INPUT SYMBOL
     012066  000016 1000 03        3455         CMPX0    S$COLON,DU       IS IT A COLON SYMBOL
     012067  006216 6010 00        3456         TNZ      A$NOMAT          NO - FAIL
     012070  011357 2200 00        3457         LDX0     RNGE             GET CURRENT RANGE NUMBER
     012071  006460 7400 00        3458         STX0     A$LEVEL          STORE FOR TABLE LOOK UP ROUTINE
     012072  000000 6210 17        3459         EAX1     0,7              GET SYMBOL IN XR - 1
     012073  035222 0610 00        3460         ADX1     T$STAB           MAKE IT ABSOLUTE
     012074  000001 0610 03        3461         ADX1     A$DF,DU          MAKE IT POINT TO CHAIN OF DEFINITIONS
     012075  006437 7000 00        3462         TSX0     A$TLU            LOOK UP POSSIBLE LABEL IN SYMBOL TABLE
     012076  006216 7100 00        3463         TRA      A$NOMAT          NOT IN TABLE SO FAIL
     012077  000001 2220 11        3464         LDX2     1,1              GET TYPE OF ENTRY IN XR - 2
     012100  006415 1020 03        3465         CMPX2    D$IDENT,DU       SEE IF IT AN IDENTIFIER
     012101  006216 6010 00        3466         TNZ      A$NOMAT          NO - PROBABLE PROGRAM ERROR
     012102  000003 2350 11        3467         LDA      3,1              GET VALUE OF LABEL IN AU
     012103  000022 7710 00        3468         ARL      18               MOVE TO AL
     012104  016504 0750 03        3469         ADA      O$LBL,DU         ADD DEFINE LABEL CODE
     012105  012614 7000 00        3470         TSX0     CAD              AND ADD LABEL DEFINITION TO OUTPUT CODE
     012106  000000 6200 17        3471         EAX0     0,7              GET LABEL IN XR - 0
     012107  035222 0600 00        3472         ADX0     T$STAB           MAKE POINTER ABSOLUTE
     012110  000000 2350 10        3473         LDA      A$SC,0           GET TALLY WORD FOR LABEL IN A
     012111  006554 0110 00        3474         NOP      1$XXS            AND PRINT IT
     012112  002152 4500 00        3475         STZ      A$PEEKF          ACCEPT LABEL
```

```
  END OF BINARY CARD 00000184
      012113  002153 2200 00         3476         LDX0     A$QEEKF         GET COLON SYMBOL
      012114  035222 0600 00         3477         ADX0     T$STAB          MAKE POINTER ABSOLUTE
      012115  000000 2350 10         3478         LDA      A$SC,0          GET TALLY WORD FOR COLON IN A
      012116  006554 0110 00         3479         NOP      1$XXS           AND PRINT IT
      012117  002153 4500 00         3480         STZ      A$QEEKF         ACCEPT COLON
      012120  006210 7100 00         3481         TRA      A$OK            AND EXIT
      012121  035234 2210 00         3482 SASGN   LDX1     A$WORK          GET POINTER TO END OF WORKING STACK IN XR - 1
      012122  777777 1610 11         3483         SBX1     -1,1            GET POINTER TO LAST CONTROL BLOCK
      012123  035234 1610 00         3484         SBX1     A$WORK          MAKE IT RELATIVE
      012124  016446 7410 00         3485         STX1     VALP            AND STORE AS POINTER TO CURRENT VALUE
      012125  015105 7000 00         3486         TSX0     SOFT            DO SOFT COERCION
      012126  016203 2350 00         3487         LDA      TMODE           GET FINAL MODE IN AU
      012127  035235 7550 56         3488         STA      A$STACK,ID      AND SAVE IN CONTROL STACK
      012130  005742 7170 00         3489         XED      T$SOVF          CHECK FOR STACK OVERFLOW
      012131  000022 7710 00         3490         ARL      18              MOVE IT BACK TO AL
      012132  016633 0750 03         3491         ADA      C$ASGN,DU       ADD ASSIGN COMMAND
      012133  012403 7000 00         3492         TSX0     INS             AND INSERT IT IN FRONT OF VALUE
      012134  035234 2210 00         3493         LDX1     A$WORK          GET POINTER TO END OF WORKING STACK
      012135  777777 1610 11         3494         SBX1     -1,1            GET POINTER TO LAST CONTROL ELEMENT
      012136  777777 3350 07         3495         LCA      -1,DL           PREPARE TO ADD -1 TO LOC AND +1 TO LEN
      012137  000002 0550 11         3496         ASA      2,1             ADJUST LOC AND LEN IN LAST CONTROL ELEMENT
      012140  006210 7100 00         3497         TRA      A$OK            AND EXIT
  END OF BINARY CARD 00000185
      012141  035235 2210 54         3498 DASGN   LDX1     A$STACK,DI      GET MODE OF ASSIGNATION IN XR - 1
      012142  012171 7410 00         3499         STX1     DASGT           AND SAVE
      012143  035216 0610 00         3500         ADX1     T$MODE          MAKE MODE ABSOLUTE
      012144  000001 2270 11         3501         LDX7     1,1             GET DEREFERENCED MODE IN XR - 7
      012145  010630 7000 00         3502         TSX0     A$XFER          AND MAKE IT UNIQUE AND ABSOLUTE
      012146  035216 1670 00         3503         SBX7     T$MODE          MAKE MODE POINTER RELATIVE
      012147  016460 7470 00         3504         STX7     DECLR           AND STORE AS TARGET MODE FOR RHS
      012150  035234 2210 00         3505         LDX1     A$WORK          GET POINTER TO END OF WORKING STACK
      012151  777777 1610 11         3506         SBX1     -1,1            GET POINTER TO LAST CONTROL BLOCK
      012152  035234 1610 00         3507         SBX1     A$WORK          MAKE POINTER RELATIVE
      012153  016446 7410 00         3508         STX1     VALP            AND STORE POINTER TO CURRENT VALUE
      012154  015072 7000 00         3509         TSX0     STRNG           COERCE RHS TO PROPER MODE
      012155  035234 2210 00         3510         LDX1     A$WORK          GET A POINTER TO THE END OF THE WORKING STACK
      012156  777777 1610 11         3511         SBX1     -1,1            GET POINTER TO LAST CONTROL BLOCK IN XR - 1
      012157  000000 0540 11         3512         AOS      0,1             PREPARE TO COMBINE TOP TWO CONTROL BLOCKS
      012160  012171 2220 00         3513         LDX2     DASGT           GET MODE OF ASSIGNATION IN XR - 2
      012161  000001 7420 11         3514         STX2     1,1             AND STORE AS MODE OF RESULT
      012162  000002 0540 11         3515         AOS      2,1             INCREMENT LENGTH OF RHS TO INCLUDE ENTER CODE
      012163  012530 7000 00         3516         TSX0     DELW            COMBINE ASSIGNATION INTO ONE CONTROL BLOCK
      012164  012171 2350 00         3517         LDA      DASGT           GET MODE OF ASSIGNATION IN AU
      012165  000022 7710 00         3518         ARL      18              MOVE MODE TO AL
      012166  016636 0750 03         3519         ADA      C$ASGNE,DU      ADD DO ASSIGNMENT CODE
  END OF BINARY CARD 00000186
      012167  012614 7000 00         3520         TSX0     CAD             AND ADD IT TO OUTPUT CODE
      012170  006210 7100 00         3521         TRA      A$OK            AND EXIT
      012171  000000 000000          3522 DASGT   ZERO
      012172  016460 2350 00        3523 ASGNE  LDA     DECLR           GET MODE OF ASSIGNATION
      012173  000022 7710 00        3524        ARL     18              MOVE MODE TO AL
      012174  016636 0750 03        3525        ADA     O$ASGNE,DU      ADD END OF ASSIGNATION CODE
      012175  012614 7000 00        3526        TSX0    CAD             AND ADD TO OUTPUT CODE
      012176  006210 7100 00        3527        TRA     A$OK            AND EXIT
X     012177  000000 0110 00        3528 SCTAB  NOP     ####            NEEDS APPROPRIATE ROUTINE
      012200  035234 2210 00        3529 SCT    LDX1    A$WORK          GET POINTER TO END OF WORKING STACK
      012201  777777 1610 11        3530        SBX1    -1,1            GET POINTER TO LAST CONTROL BLOCK
      012202  035234 1610 00        3531        SBX1    A$WORK          MAKE POINTER RELATIVE
      012203  016446 7410 00        3532        STX1    VALP            SAVE POINTER IN CURRENT VALUE POINTER
      012204  015105 7000 00        3533        TSX0    SOFT            COERCE LHS SOFTLY
      012205  016464 2350 00        3534        LDA     GLBL            GET A UNIQUE LABEL
      012206  016464 0540 00        3535        AOS     GLBL            INCREMENT LABEL GENERATOR
      012207  016507 0750 03        3536        ADA     O$JUMP,DU       ADD JUMP CODE
      012210  012403 7000 00        3537        TSX0    INS             AND INSERT TRANSFER CODE BEFORE VALUE
      012211  017020 6350 00        3538        EAA     O$MREF          GET MAKE REFERENCE VALUE BLOCK COMMAND
      012212  012403 7000 00        3539        TSX0    INS             AND INSERT COMMAND IN FRONT OF VALUE
      012213  016464 2350 00        3540        LDA     GLBL            GET A UNIQUE LABEL
      012214  016464 0540 00        3541        AOS     GLBL            INCREMENT LABEL GENERATOR
    END OF BINARY CARD 00000187
      012215  035235 7550 56        3542        STA     A$STACK,ID      STORE PRESENT STATE OF LABEL GENERATOR IN STACK
      012216  005742 7170 00        3543        XED     T$SOVF          CHECK FOR STACK OVERFLOW
      012217  016504 0750 03        3544        ADA     O$LBL,DU        ADD DEFINE LABEL CODE
      012220  012403 7000 00        3545        TSX0    INS             AND INSERT BEFORE VALUE
      012221  016446 2210 00        3546        LDX1    VALP            GET POINTER TO CURRENT VALUE
      012222  035234 0610 00        3547        ADX1    A$WORK          MAKE CONTROL BLOCK POINTER ABSOLUTE
      012223  000001 2270 11        3548        LDX7    1,1             GET MODE OF VALUE IN XR - 7
      012224  010630 7000 00        3549        TSX0    A$XFER          MAKE MODE POINTER UNIQUE AND ABSOLUTE
      012225  000001 2350 17        3550        LDA     1,7             GET REQUIRED MODE OF RHS IN AU
      012226  000022 7710 00        3551        ARL     18              MOVE MODE TO AL
      012227  016542 0750 03        3552        ADA     O$CONF,DU       ADD TEST CONFORMITY CODE
      012230  012403 7000 00        3553        TSX0    INS             AND INSERT CODE BEFORE VALUE
      012231  016464 2350 00        3554        LDA     GLBL            GET A UNIQUE LABEL
      012232  016464 0540 00        3555        AOS     GLBL            INCREMENT LABEL GENERATOR
      012233  016545 0750 03        3556        ADA     O$TF,DU         ADD TRANSFER IF FALSE CODE
      012234  012403 7000 00        3557        TSX0    INS             AND INSERT TRANSFER CODE BEFORE VALUE
      012235  017020 6350 00        3558        EAA     O$MREF          GET MAKE REFERENCE VALUE BLOCK COMMAND
      012236  012403 7000 00        3559        TSX0    INS             AND INSERT CODE BEFORE VALUE
      012237  016446 2210 00        3560        LDX1    VALP            GET POINTER TO CURRENT VALUE CONTROL BLOCK
      012240  035234 0610 00        3561        ADX1    A$WORK          MAKE POINTER ABSOLUTE
      012241  000001 2350 11        3562        LDA     1,1             GET MODE OF VALUE IN AU
      012242  000022 7710 00        3563        ARL     18              MOVE MODE TO AL
    END OF BINARY CARD 00000188
      012243  016553 0750 03        3564        ADA     O$CASGN,DU      ADD CONFORMITY ASSIGNMENT CODE
      012244  012614 7000 00        3565        TSX0    CAD             AND ADD CODE AFTER VALUE
      012245  016603 6350 00        3566        EAA     O$TRUE          GET ENTER TRUE CODE
      012246  012614 7000 00        3567        TSX0    CAD             AND ADD CODE AFTER VALUE
      012247  016464 2350 00        3568        LDA     GLBL            GET A UNIQUE LABEL
      012250  016464 0540 00        3569        AOS     GLBL            INCREMENT LABEL GENERATOR
      012251  016507 0750 03        3570        ADA     O$JUMP,DU       ADD UNCONDITIONAL JUMP CODE
```

```
           2                                        PASS 2

       012252  012614 7000 00        3571          TSX0     CAD               AND ADD CODE AFTER VALUE
       012253  016464 2350 00        3572          LDA      GLBL              GET CURRENT VALUE OF LABEL GENERATOR
       012254  000004 1750 07        3573          SBA      4,DL              GET LABEL INITIALLY GENERATED
       012255  016504 0750 03        3574          ADA      O$LBL,DU          ADD DEFINE LABEL CODE
       012256  012614 7000 00        3575          TSX0     CAD               AND ADD CODE AFTER VALUF
       012257  017026 6350 00        3576          EAA      O$MAX             GET SET STACK POINTER TO MAXIMUM COMMAND
       012260  012614 7000 00        3577          TSX0     CAD               AND ADD CODE AFTER VALUE
       012261  006210 7100 00        3578          TRA      A$OK              AND EXIT
X      012262  000000 0110 00        3579 DCTAB    NOP      ####              NEEDS APPROPRIATE ROUTINE
       012263  017023 6350 00        3580 DCT      EAA      O$CONE            GET CONFORMITY CLEANUP COMMAND IN A
       012264  012614 7000 00        3581          TSX0     CAD               AND ADD AFTER RHS VALUE
       012265  035235 2350 54        3582          LDA      A$STACK,DI        GET SAVED VALUE OF LABEL GENERATOR
       012266  012321 7550 00        3583          STA      DCTT              AND SAVE
       012267  012321 0540 00        3584          AOS      DCTT              INCREMENT TO NEXT LABEL TO BE USED
       012270  016507 0750 03        3585          ADA      O$JUMP,DU         ADD UNCONDITIONAL JUMP CODE
   END OF BINARY CARD 00000189
       012271  012614 7000 00        3586          TSX0     CAD               AND ADD AFTER RHS VALUE
       012272  012321 2350 00        3587          LDA      DCTT              GET NEXT LABEL TO BE USED
       012273  012321 0540 00        3588          AOS      DCTT              AND INCREMENT FOR NEXT TIME
       012274  016504 0750 03        3589          ADA      O$LBL,DU          ADD DEFINE LABEL CODE
       012275  012614 7000 00        3590          TSX0     CAD               AND ADD AFTER RHS VALUE
       012276  016606 6350 00        3591          EAA      O$FALSE           GET ENTER FALSE CODE
       012277  012614 7000 00        3592          TSX0     CAD               AND ADD CODE AFTER CURRENT VALUE
       012300  012321 2350 00        3593          LDA      DCTT              GET NEXT VALUE OF LABEL TO CONSIDER
       012301  016504 0750 03        3594          ADA      O$LBL,DU          ADD DEFINE LABEL CODE
       012302  012614 7000 00        3595          TSX0     CAD               ADD CODE AFTER CURRENT VALUE
       012303  016446 2210 00        3596          LDX1     VALP              GET POINTER TO CURRENT VALUE CONTROL BLOCK
       012304  035234 0610 00        3597          ADX1     A$WORK            MAKE POINTER ABSOLUTE
       012305  000000 0540 11        3598          AOS      0,1               MAKE CONTROL BLOCK KEY 2 VALUES
       012306  000003 6350 00        3599          EAA      M$BOOL            GET A BOOLEAN MODE
       012307  000001 7550 11        3600          STA      1,1               AND STORE AS MODE OF RESULT
       012310  012530 7000 00        3601          TSX0     DELW              COMBINE INTO SINGLE VALUE
       012311  035234 2210 00        3602          LDX1     A$WORK            GET POINTER TO END OF WORKING STACK
       012312  000005 1610 03        3603          SBX1     5,DU              GET POINTER TO LAST CONTROL BLOCK IN WORK
       012313  777772 2200 03        3604          LDX0     -6,DU             GET MINUS NUMBER OF WORDS INSERTED BEFORE VALUE
       012314  000002 0400 11        3605          ASX0     2,1               UPDATE LEN/LOC WORD TO REFER TO START OF VALUE
       012315  035224 7200 00        3606          LXL0     T$CODE            GET POINTER TO END OF GENERATED CODE
       012316  000002 1600 11        3607          SBX0     2,1               SUBTRACT STARTING LOCATION OF VALUE TO GET LENGTH
   END OF BINARY CARD 00000190
       012317  000002 4400 11        3608          SXL0     2,1               AND STORE LENGTH IN LOC/LEN WORD
       012320  006210 7100 00        3609          TRA      A$OK              AND EXIT
       012321  000000 000000         3610 DCTT     ZERO
       012322  016600 2350 03        3611 SISNT    LDA      O$ISNT,DU         GET ISNT IDENTITY CODE
       012323  012325 7100 00        3612          TRA      SIS1              AND GO DO IDENTITY
       012324  016575 2350 03        3613 SIS      LDA      O$IS,DU           GET IS IDENTITY CODE
       012325  035235 7550 56        3614 SIS1     STA      A$STACK,ID        STORE RELATOR IN CONTROL STACK
       012326  005742 7170 00        3615          XED      T$SOVF            CHECK FOR STACK OVERFLOW
       012327  006210 7100 00        3616          TRA      A$OK              AND EXIT
       012330  035235 2350 54        3617 IDNTY    LDA      A$STACK,DI        GET RELATOR IN AU
       012331  020000 2360 03        3618          LDQ      W$BAL,DU          GET BALANCED HEADER FOR WORK ENTRY
```

```
    012332  000002 0760 07        3619          ADQ     2,DL            INDICATE 2 VALUES ARE BALANCED
    012333  012572 7000 00        3620          TSX0    WAD             GO ADD TO WORKING AND CONTROL STACKS
    012334  035234 2200 00        3621          LDX0    A$WORK          GET POINTER TO END OF WORKING STACK
    012335  000005 1600 03        3622          SBX0    5,DU            GET POINTER TO LAST CONTROL BLOCK IN WORK
    012336  035234 1600 00        3623          SBX0    A$WORK          MAKE POINTER RELATIVE
    012337  016446 7400 00        3624          STX0    VALP            AND STORE AS POINTER TO CURRENT CONTROL BLOCK
    012340  000001 2230 03        3625          LDX3    1,DU            INDICATE SOFT COERCION OF IDENTITY OPERANDS
    012341  015114 7000 00        3626          TSX0    BB              CALCULATE DESIRED MODES OF OPERANDS
    012342  016202 2270 00        3627          LDX7    BMODE           GET TARGET MODE FOR OPERANDS
    012343  012374 7470 00        3628          STX7    IDNTT           AND SAVE
    012344  035234 2200 00        3629          LDX0    A$WORK          GET POINTER TO END OF WORKING STACK
END OF BINARY CARD 00000191
    012345  000012 1600 03        3630          SBX0    10,DU           GET POINTER TO SECOND OPERAND CONTROL BLOCK
    012346  035234 1600 00        3631          SBX0    A$WORK          MAKE POINTER RELATIVE
    012347  016446 7400 00        3632          STX0    VALP            AND STORE AS CURRENT VALUE CONTROL BLOCK POINTER
    012350  015253 7000 00        3633          TSX0    C               SET UP COERCION FOR RIGHT OPERAND
    012351  777777 7100 00        3634          TRA     $ERROR          IMPOSSIBLE COERCION - ERROR
    012352  016204 7000 00        3635          TSX0    DC              DO INDICATED COERCIONS
    012353  012374 2270 00        3636          LDX7    IDNTT           GET TARGET MODE FOR OPERANDS
    012354  016202 7470 00        3637          STX7    BMODE           AND STORE AS TARGET MODE
    012355  035234 2200 00        3638          LDX0    A$WORK          GET POINTER TO END OF WORKING STACK
    012356  000017 1600 03        3639          SBX0    15,DU           GET POINTER TO FIRST OPERAND CONTROL BLOCK
    012357  035234 1600 00        3640          SBX0    A$WORK          MAKE POINTER RELATIVE
    012360  016446 7400 00        3641          STX0    VALP            AND STORE AS POINTER TO CURRENT VALUE
    012361  015253 7000 00        3642          TSX0    C               SET UP COERCION FOR LEFT OPERAND
    012362  777777 7100 00        3643          TRA     $ERROR          IMPOSSIBLE COERCION - ERROR
    012363  016204 7000 00        3644          TSX0    DC              DO INDICATED COERCIONS
    012364  035234 2210 00        3645          LDX1    A$WORK          GET POINTER TO END OF WORKING STACK
    012365  000005 1610 03        3646          SBX1    5,DU            GET POINTER TO LAST BLOCK - BAL2
    012366  012530 7000 00        3647          TSX0    DELW            COMBINE OPERANDS INTO A SINGLE VALUE
    012367  035234 2210 00        3648          LDX1    A$WORK          GET POINTER TO END OF WORKING STACK
    012370  000005 1610 03        3649          SBX1    5,DU            GET POINTER TO LAST VALUE CONTROL BLOCK
    012371  000003 6350 00        3650          EAA     M$BOOL          GET MODE OF IDENTITY RELATION
    012372  000001 7550 11        3651          STA     1,1             AND STORE AS MODE OF VALUE
END OF BINARY CARD 00000192
    012373  006210 7100 00        3652          TRA     A$OK            AND EXIT
    012374  000000 000000         3653 IDNTT    ZERO
    012375  012456 7400 00        3654 ADB      STX0    INSX            SAVE RETURN
    012376  016446 2210 00        3655          LDX1    VALP            GET POINTER TO CURRENT CONTROL BLOCK
    012377  035234 0610 00        3656          ADX1    A$WORK          MAKE POINTER ABSOLUTE
    012400  000002 7250 11        3657          LXL5    2,1             GET LENGTH OF VALUE IN XR - 5
    012401  000002 0650 11        3658          ADX5    2,1             AND ADD LOCATION OF VALUE
    012402  012407 7100 00        3659          TRA     ADD1            GO TO INSERTION ROUTINE
    012403  012456 7400 00        3660 INS      STX0    INSX            SAVE RETURN
    012404  016446 2210 00        3661          LDX1    VALP            GET POINTER TO CURRENT CONTROL BLOCK
    012405  035234 0610 00        3662          ADX1    A$WORK          MAKE IT ABSOLUTE
    012406  000002 2250 11        3663          LDX5    2,1             GET POINTER TO START OF VALUE IN XR - 5
    012407  012457 7550 00        3664 ADD1     STA     N               SAVE WORD TO BE INSERTED
    012410  035224 7220 00        3665          LXL2    T$CODE          SAVE OLD END ADDRESS OF CODE
    012411  000000 6350 00        3666          EAA     1-1             GET PARAMETER TO ASK FOR ONE WORD
```

```
             2                                                  PASS 2

        012412  035224 2210 03          3667         LDX1    T$CODE,DU          GET POINTER TO CODE TABLE CONTROL WORD
        012413  005663 7000 00          3668         TSX0    T$ALOC             AND ALLOCATE MEMORY
        012414  035224 0650 00          3669         ADX5    T$CODE             GET ABSOLUTE POINTER TO INSERTED WORD
        012415  012460 7450 00          3670         STX5    A                  AND STORE IN MEMORY
        012416  012460 1010 00          3671 INS1    CMPX1   A                  HAVE WE MOVED ENOUGH
        012417  012424 6000 00          3672         TZE     INS2               TRANSFER IF DONE MOVING
        012420  777777 2350 11          3673         LDA     -1,1               GET WORD TO MOVE
     END OF BINARY CARD 00000193
        012421  000000 7550 11          3674         STA     0,1                AND STORE MOVED BACK ONE LOCATION
        012422  000001 1610 03          3675         SBX1    1,DU               DECREMENT MOVE POINTER
        012423  012416 7100 00          3676         TRA     INS1               AND LOOP
        012424  012457 2350 00          3677 INS2    LDA     N                  GET WORD TO BE INSERTED
        012425  000000 7550 11          3678         STA     0,1                AND STORE IN OUTPUT CODE
        012426  035224 1650 00          3679         SBX5    T$CODE             MAKE CODE POINTER RELATIVE
        012427  000001 2240 03          3680         LDX4    1,DU               GET A ONE FOR INCREMENTING
        012430  035234 2200 00          3681         LDX0    A$WORK             GET POINTER TO END OF WORKING STACK
        012431  000005 1600 03          3682 INS3    SBX0    5,DU               MAKE XR - 0 POINT TO PRECEEDING STACK ENTRY
        012432  035214 1000 00          3683         CMPX0   T$WORK             SEE IF POINTER IS BELOW WORKING STACK
        012433  012456 6020 00          3684         TNC     INSX               TRANSFER IF ALL BLOCKS RELOCATED
        012434  000000 2210 10          3685         LDX1    0,0                GET TYPE OF ENTRY IN XR - 1
        012435  760000 3010 03          3686         CANX1   W$OBJCT,DU         DOES ENTRY CONTAIN A LOC/LEN POINTER
        012436  012431 6000 00          3687         TZE     INS3               TRANSFER IF NOT TO CONTINUE
        012437  000002 1050 10          3688         CMPX5   2,0                TEST INSERT POINTER FOR FORBIDDEN RANGE
        012440  012447 6000 00          3689         TZE     INS5               TRANSFER IF OK
        012441  012447 6040 00          3690         TMI     INS5               TRANSFER IF OK
        012442  000002 7220 10          3691         LXL2    2,0                GET LENGTH OF ELEMENT IN XR - 2
        012443  000002 0620 10          3692         ADX2    2,0                ADD LOCATION TO GET END OF ELEMENT
        012444  012445 7420 00          3693         STX2    *+1                STORE FOR COMPARE
        012445  000000 1050 03          3694         CMPX5   **,DU              TEST INSERT POINTER FOR FORBIDDEN RANGE
        012446  777777 6040 00          3695         TMI     $ERROR             COMPILER ERROR - INSERTING IN MIDDLE
     END OF BINARY CARD 00000194
        012447  000002 1050 10          3696 INS5    CMPX5   2,0                COMPARE INSERT POINTER WITH LOC OF VALUE
        012450  012452 6000 00          3697         TZE     INS6               TRANSFER IF EQUAL TO INCREMENT LOC
        012451  012454 6050 00          3698         TPL     INS4               TRANSFER IF LOC IS LESS THAN POINTER
        012452  000002 0440 10          3699 INS6    ASX4    2,0                INCREMENT LOCATION OF VALUE BY ONE
        012453  012431 7100 00          3700         TRA     INS3               TRANSFER TO LOOK FOR MORE RELOCATION
        012454  060000 3010 03          3701 INS4    CANX1   W$MULT,DU          IS THIS A MULTIPLE VALUE
        012455  012431 6010 00          3702         TNZ     INS3               YES - TRANSFER TO KEEP UPDATING
        012456  000000 7100 00          3703 INSX    TRA     **                 NO - RETURN
        012457  000000 000000           3704 N       ZERO
        012460  000000 000000           3705 A       ZERO
        012461  012512 7400 00          3706 MATCH   STX0    MATX               SAVE RETURN
        012462  777777 6360 11          3707         EAQ     -1,1               GET LENGTH - 1 OF ENTRY IN Q REGISTER
        012463  777777 6000 00          3708         TZE     $ERROR             COMPILER ERROR - ONE WORD MODE
        012464  000012 7360 00          3709         QLS     10                 POSITION LENGTH FOR REPEAT COUNT
        012465  000000 6230 02          3710         EAX3    0,QU               AND SAVE IN XR - 3
        012466  035216 2240 00          3711         LDX4    T$MODE             GET POINTER TO BASE OF MODE TABLE
        012467  000001 0640 03          3712 MAT1    ADX4    1,DU               STEP OVER LINK WORD
        012470  777777 2340 14          3713         SZN     -1,4               CHECK FOR END OF MODE TABLE
        012471  777777 6000 00          3714         TZE     $ERROR             IF NO MORE - ERROR
```

```
            2                                          PASS 2
      012472  777777 1010 14         3715          CMPX1    -1,4            SEE IF CURRENT ENTRY HAS THE RIGHT LENGTH
      012473  012507 6010 00         3716          TNZ      MAT2            TRANSFER IF WRONG LENGTH
      012474  000000 2200 14         3717          LDX0     0,4             GET TABLE ENTRY MODE TYPE IN XR - 0
   END OF BINARY CARD 00000195
      012475  000000 1000 12         3718          CMPX0    0,2             COMPARE WITH DESIRED MODE TYPE
      012476  012507 6010 00         3719          TNZ      MAT2            TRANSFER IF NOT EQUAL TO TRY ANOTHER TABLE ENTRY
      012477  001440 6200 13         3720          EAX0     768+32,3        PUT COUNT, TNZ, AND AB BITS IN XR - 0
      012500  000001 6250 14         3721          EAX5     1,4             PUT ADDRESS + 1 OF TABLE ENTRY IN XR - 5
      012501  000001 6260 12         3722          EAX6     1,2             PUT ADDRESS + 1 OF DESIRED CONTENTS IN XR - 6
      012502  00000001100/
      012503  000040 5602 01         3723          RPDX     ,1,TNZ          COMPARE
      012504  000000 2350 15         3724          LDA      0,5             TABLE ENTRY
      012505  000000 1150 16         3725          CMPA     0,6             WITH DESIRED TABLE ENTRY CONTENTS
      012506  012511 6000 00         3726          TZE      MAT3            TRANSFER IF THE SAME
      012507  777777 0640 14         3727 MAT2     ADX4     -1,4            STEP OVER CURRENT TABLE ENTRY
      012510  012467 7100 00         3728          TRA      MAT1            AND LOOP
      012511  035216 1640 00         3729 MAT3     SBX4     T$MODE          MAKE POINTER RELATIVE
      012512  000000 7100 00         3730 MATX     TRA      **              AND RETURN
      012513  000000 6220 11         3731 DELWW    EAX2     0,1             GET POINTER TO PAR OR BAL WORK ELEMENT
      012514  000000 7230 11         3732          LXL3     0,1             GET NUMBER OF CONTROLLED VALUES IN XR - 3
      012515  777777 1620 12         3733 DELWL    SBX2     -1,2            STEP XR - 2 TO PREVIOUS CONTROL ELEMENT
      012516  000001 1630 03         3734          SBX3     1,DU            DECREMENT NUMBER OF VALUES TO GO
      012517  012515 6010 00         3735          TNZ      DELWL           TRANSFER IF MORE VALUES TO SKIP
      012520  000002 2230 11         3736          LDX3     2,1             GET POINTER TO START OF CODE FOR VALUE IN XR - 3
      012521  000000 6240 11         3737          EAX4     0,1             GET POINTER TO PAR OR BAL ENEMENT IN XR - 4
      012522  777777 1640 14         3738          SBX4     -1,4            GET POINTER TO LAST VALUE CONTROL BLOCK IN GROUP
   END OF BINARY CARD 00000196
      012523  000002 7250 14         3739          LXL5     2,4             GET LENGTH OF LAST VALUE IN XR - 5
      012524  000002 0650 14         3740          ADX5     2,4             ADD LOCATION TO GET END ADDRESS
      012525  000002 7430 12         3741          STX3     2,2             STORE START ADDRESS IN FIRST CONTROL BLOCK
      012526  000002 4500 11         3742          STZ      2,1             CLEAN OUT LEN/LOC IN BAL OR PAR ELEMENT
      012527  000002 7450 11         3743          STX5     2,1             STORE END ADDRESS IN BAL OR PAR ELEMENT
      012530  012571 7400 00         3744 DELW     STX0     DELWX           SAVE RETURN
      012531  000000 7220 11         3745          LXL2     0,1             GET NUMBER OF VALUES IN PAR OR BAL
      012532  000000 6230 11         3746          EAX3     0,1             SAVE STARTING LOCATION OF MOVE IN XR - 3
      012533  000002 7250 11         3747          LXL5     2,1             GET LENGTH OF CODE IN XR - 5
      012534  000002 0650 11         3748          ADX5     2,1             ADD LOCATION TO GET POINTER TO END OF CODE
      012535  000003 2350 11         3749          LDA      3,1             GET LL WORD OF CURRENT VALUE
      012536  777777 1610 11         3750 DELW1    SBX1     -1,1            STEP TO NEXT ENTRY IN STACK
      012537  000003 2750 11         3751          ORA      3,1             OR IN LL WORD OF THIS VALUE
      012540  000001 1620 03         3752          SBX2     1,DU            DECREMENT NUMBER LEFT TO SKIP
      012541  012536 6010 00         3753          TNZ      DELW1           TRANSFER IF MORE TO SKIP
      012542  000003 7550 13         3754          STA      3,3             STORE ORED LL WORD AS LL WORD OF COMBINED VALUE
      012543  400000 6350 00         3755          EAA      W$VALUE         GET VALUE HEADER FOR NEW ENTRY
      012544  000000 7550 13         3756          STA      0,3             AND STORE IN NEW ENTRY
      012545  000002 2240 11         3757          LDX4     2,1             GET BASE OF VALUE IN XR - 4
      012546  000002 7440 13         3758          STX4     2,3             STORE BASE IN NEW CONTROL BLOCK
      012547  000002 1650 11         3759          SBX5     2,1             SUBTRACT BASE FROM END OF CODE ADDRESS
      012550  000002 4450 13         3760          SXL5     2,3             AND STORE AS LENGTH OF NEW VALUE
   END OF BINARY CARD 00000197
```

```
           2                                                   PASS 2

      012551  000000 6240 13        3761         EAX4     0,3              GET POINTER WHERE TO START MOVE IN XR - 4
      012552  035234 1640 00        3762         SBX4     A$WORK           SUBTRACT LOCATION OF END OF MOVE
      012553  000000 6350 14        3763         EAA      0,4              PUT MINUS NUMBER OF WORDS TO MOVE IN A
      012554  000000 5310 00        3764         NEG                       MAKE IT POSITIVE
      012555  000010 7710 00        3765         ARL      8                POSITION FOR REPEAT
      012556  001400 6200 05        3766         EAX0     768,AL           GET NUMBER OF WORDS TO MOVE MOD 256 IN XR - 0
      012557  000001 1750 07        3767         SBA      1,DL             MAKE 0=256 EXCEPTION NOT BOTHER LOOP COUNT
      012560  012566 6040 00        3768         TMI      DELW3            TRANSFER IF NOTHING TO MOVE
      012561  000000 5602 01        3769 DELW2   RPDX     ,1               MOVE
      012562  000000 2360 13        3770         LDQ      0,3              FROM SPECIFIED LOCATION
      012563  000000 7560 11        3771         STQ      0,1              TO NEW LOCATION
      012564  000001 1750 03        3772         SBA      1,DU             SEE IF MORE TO MOVE
      012565  012561 6050 00        3773         TPL      DELW2            TRANSFER IF MORE TO MOVE
      012566  035234 0110 54        3774 DELW3   NOP      A$WORK,DI        DELETE A WORD FROM THE WORKING STACK
      012567  035234 1010 00        3775         CMPX1    A$WORK           HAVE WE DELETED BACK TO LAST WORD MOVED
      012570  012566 6010 00        3776         TNZ      DELW3            NO - TRANSFER
      012571  000000 7100 00        3777 DELWX   TRA      **               AND EXIT
      012572  012613 7400 00        3778 WAD     STX0     WADX             SAVE RETURN
      012573  035234 7560 56        3779         STQ      A$WORK,ID        STORE HEADER FOR CODE CONTROL BLOCK
      012574  005754 7170 00        3780         XED      T$WOVF           CHECK FOR STACK OVERFLOW
      012575  035234 4500 56        3781         STZ      A$WORK,ID        STORE ZERO FOR MODE WORD
      012576  005754 7170 00        3782         XED      T$WOVF           CHECK FOR STACK OVERFLOW
  END OF BINARY CARD 00000198
      012577  035224 2360 00        3783         LDQ      T$CODE           GET POINTER TO CURRENT END OF CODE
      012600  000000 6360 06        3784         EAQ      0,QL             PUT IN QL
      012601  000001 0760 07        3785         ADQ      1,DL             SET NUMBER OF WORDS OF CODE EQUAL TO ONE
      012602  035234 7560 56        3786         STQ      A$WORK,ID        STORE AS LOC/LEN WORD
      012603  005754 7170 00        3787         XED      T$WOVF           CHECK FOR STACK OVERFLOW
      012604  016465 2360 00        3788         LDQ      LL               GET CURRENT LEXICOLOGICAL LEVEL
      012605  035234 7560 56        3789         STQ      A$WORK,ID        AND STORE IN STACK

      012606  005754 7170 00        3790         XED      T$WOVF           CHECK FOR STACK OVERFLOW
      012607  000005 6360 00        3791         EAQ      5                GET NUMBER OF WORDS IN CONTROL BLOCK
      012610  035234 7560 56        3792         STQ      A$WORK,ID        AND STORE AFTER CONTROL BLOCK
      012611  005754 7170 00        3793         XED      T$WOVF           CHECK FOR STACK OVERFLOW
      012612  012614 7000 00        3794         TSX0     CAD              ADD WORD TO CODE
      012613  000000 7100 00        3795 WADX    TRA      **               AND EXIT
      012614  012623 7400 00        3796 CAD     STX0     CADX             SAVE RETURN
      012615  012624 7550 00        3797         STA      CADT             SAVE WORD TO BE INSERTED
      012616  000000 6350 00        3798         EAA      0                REQUEST 0 + 1 WORD
      012617  035224 2210 03        3799         LDX1     T$CODE,DU        GET POINTER TO CODE TABLE CONTROL WORD
      012620  005663 7000 00        3800         TSX0     T$ALOC           AND GET ANOTHER WORD FOR CODE
      012621  012624 2350 00        3801         LDA      CADT             GET WORD TO BE ADDED
      012622  777777 7550 11        3802         STA      -1,1             AND STORE AT END OF CODE
      012623  000000 7100 00        3803 CADX    TRA      **               AND RETURN
      012624  000000 000000         3804 CADT    ZERO
  END OF BINARY CARD 00000199
      012625  013257 2270 00        3805 DEREF   LDX7     FMODE            GET SAVED MODE IN XR - 7
      012626  010630 7000 00        3806         TSX0     A$XFER           MAKE IT UNIQUE AND ABSOLUTE
      012627  000000 2220 17        3807         LDX2     0,7              GET TYPE OF MODE IN XR - 2
      012630  016762 1020 03        3808         CMPX2    M$REF,DU         IS IT A REFERENCE MODE
```

```
          2                                              PASS 2

     012631  777777 6010 00               TNZ       $ERROR           NO - COMPILER ERROR
     012632  000001 2270 17               LDX7      1,7              GET DEREFERENCED MODE
     012633  010630 7000 00               TSX0      A$XFER           MAKE IT UNIQUE
     012634  035216 1670 00               SBX7      T$MODE           MAKE IT RELATIVE
     012635  016460 7470 00               STX7      DECLR            AND MAKE IT CURRENT MODE
     012636  006210 7100 00               TRA       A$OK             AND EXIT
     012637  000001 6350 00        EVOID  EAA       M$VOID           GET A VOID DECLARER IN A
     012640  016460 7550 00               STA       DECLR            AND MAKE IT CURRENT DECLARER
     012641  006210 7100 00               TRA       A$OK             AND EXIT
     012642  016460 2270 00        EREF   LDX7      DECLR            GET CURRENT DECLARER IN XR - 7
     012643  012646 7000 00               TSX0      EREF1            AND REFERENCE MODE
     012644  016460 7470 00               STX7      DECLR            AND STORE AS NEW DECLARER
     012645  006210 7100 00               TRA       A$OK             AND EXIT
     012646  012656 7400 00        EREF1  STX0      EREFX            SAVE RETURN
     012647  010630 7000 00               TSX0      A$XFER           MAKE MODE POINTER UNIQUE AND ABSOLUTE
     012650  035216 1670 00               SBX7      T$MODE           MAKE MODE POINTER RELATIVE
     012651  012660 7470 00               STX7      EREFT+1          STORE MODE IN REFERENCE MODE PROTOTYPE
     012652  000002 2210 03               LDX1      2,DU             MATCH 2 WORDS
 END OF BINARY CARD 00000200
     012653  012657 2220 03               LDX2      EREFT,DU         AT REFERENCE MODE PROTOTYPE
     012654  012461 7000 00               TSX0      MATCH            GET POINTER TO REQUIRED MODE
     012655  000000 6270 14               EAX7      0,4              PUT REFERENCED MODE IN XR - 7
     012656  000000 7100 00        EREFX  TRA       **               AND RETURN
     012657  016762 000000         EREFT  ZERO      M$REF,0
     012660  000000 000000                ZERO
     012661  035234 2200 00        CST    LDX0      A$WORK           GET POINTER TO END OF WORKING STACK
     012662  035214 1600 00               SBX0      T$WORK           GET CURRENT LENGTH OF WORKING STACK
     012663  035235 1600 54               SBX0      A$STACK,DI       SUBTRACT MARK TO GET NO OF ADDED WORDS
     012664  000001 6350 10               EAA       1,0              ADD ONE TO GET 2N+2 FOR N FIELDS
     012665  000001 7310 00               ARS       1                DIVIDE BY 2
     012666  012725 7550 00               STA       CSTA             STORE NUMBER OF FIELDS PLUS ONE
     012667  035217 2210 03               LDX1      T$BOUND,DU       GET POINTER TO BOUND TABLE CONTROL WORD
     012670  005663 7000 00               TSX0      T$ALOC           GET ENTRY IN BOUND TABLE
     012671  012726 7410 00               STX1      CSTB             STORE POINTER TO BOUND TABLE ENTRY
     012672  012725 2350 00               LDA       CSTA             GET NUMBER OF WORDS IN BOUND TABLE ENTRY
     012673  000014 7710 00               ARL       18-6             MOVE TO TALLY PART OF WORD
     012674  000000 6200 05               EAX0      0,AL             MOVE TO XR - 0
     012675  012726 4400 00               SXL0      CSTB             AND STORE IN TALLY WORD
     012676  035214 2200 00               LDX0      T$WORK           GET POINTER TO BASE OF WORKING STACK
     012677  035235 0600 56               ADX0      A$STACK,ID       GET POINTER TO START OF MARKED DATA
     012700  000001 0600 03               ADX0      1,DU             SKIP HEADER WORD
 END OF BINARY CARD 00000201
     012701  012727 7400 00               STX0      CSTC             STORE IN FETCH TALLY WORD
     012702  012730 7400 00               STX0      CSTD             STORE IN STORE TALLY WORD
     012703  016757 6350 00               EAA       B$STRCT          GET HEADER WORD FOR BOUND TABLE ENTRY
     012704  012726 7550 56               STA       CSTB,ID          AND STORE IN BOUND TABLE ENTRY
     012705  012727 2350 56        CST1   LDA       CSTC,ID          GET MODE OF FIELD
     012706  012730 7550 51               STA       CSTD,I           AND STORE IN MODE LIST
     012707  000000 6360 05               EAQ       0,AL             GET BOUND INFORMATION IN QU
     012710  012727 2350 56               LDA       CSTC,ID          GET TAG INFORMATION
```

```
              2                                                    PASS 2

      012711  000000 6220 01        3857          EAX2    0,AU                AND PUT IT IN XR - 2
      012712  012730 4420 56        3858          SXL2    CSTD,ID             STORE TAG IN MODE LIST
      012713  012726 7560 56        3859          STQ     CSTB,ID             STORE BOUND IN BOUND TABLE ENTRY
      012714  012705 6070 00        3860          TTF     CST1                TRANSFER IF MORE FIELDS
      012715  035217 1610 00        3861          SBX1    T$BOUND             MAKE BOUND POINTER RELATIVE
      012716  016460 4410 00        3862          SXL1    DECLR               AND STORE IN CURRENT DECLARER
      012717  012730 2200 00        3863          LDX0    CSTD                GET POINTER TO END OF MODE LIST
      012720  035234 0110 54        3864 CST2     NOP     A$WORK,DI           DELETE A WORD FROM THE WORKING STACK
      012721  035234 1000 00        3865          CMPX0   A$WORK              SEE IF DELETED BACK TO MODE LIST
      012722  012720 6010 00        3866          TNZ     CST2                TRANSFER IF MORE TO DELETE
      012723  016757 6350 00        3867          EAA     M$STRCT             GET HEADER WORD FOR MODE TABLE ENTRY
      012724  012735 7100 00        3868          TRA     CUN2                AND EVALUATE MODE
      012725  000000 000000         3869 CSTA     ZERO
      012726  000000 000000         3870 CSTB     ZERO
   END OF BINARY CARD 00000202
      012727  000000 000000         3871 CSTC     ZERO
      012730  000000 000000         3872 CSTD     ZERO
      012731  017001 6350 00        3873 EPR      EAA     M$PROC              GET HEADER WORD FOR MODE TABLE ENTRY
      012732  012734 7100 00        3874          TRA     CUN1                AND EVALUATE MODE
      012733  017007 6350 00        3875 CUN      EAA     M$UNION             GET HEADER WORD FOR MODE TABLE ENTRY
      012734  016460 4500 00        3876 CUN1     STZ     DECLR               CLEAR OUT BOUND INFORMATION IN CURRENT DECLARER
      012735  035214 2220 00        3877 CUN2     LDX2    T$WORK              GET POINTER TO BASE OF WORKING STACK
      012736  035235 0620 54        3878          ADX2    A$STACK,DI          GET POINTER TO WHERE THE STACK WAS MARKED
      012737  000000 7550 12        3879          STA     0,2                 STORE HEADER WORD AT THIS PLACE
      012740  777777 6210 12        3880          EAX1    -1,2                DECREMENT PLACE AND MOVE TO XR - 1
      012741  777777 6610 03        3881          ERX1    -1,DU               GET MINUS PLACE IN XR - 1
      012742  035234 0610 00        3882          ADX1    A$WORK              ADD POINTER TO END FOR LENGTH
      012743  012461 7000 00        3883          TSX0    MATCH               FIND MODE IN MODE TABLE
      012744  016460 7440 00        3884          STX4    DECLR               AND STORE MODE POINTER IN CURRENT DECLARER
      012745  035214 2220 00        3885          LDX2    T$WORK              GET POINTER TO BASE OF WORKING STACK
      012746  035235 0620 51        3886          ADX2    A$STACK,I           ADD RELATIVE MARK LOCATION
      012747  035234 0110 54        3887 CUN3     NOP     A$WORK,DI           DELETE A WORD FROM THE WORKING STACK
      012750  035234 1020 00        3888          CMPX2   A$WORK              SEE IF DELETED BACK TO MARK
      012751  012747 6010 00        3889          TNZ     CUN3                TRANSFER IF MORE TO DELETE
      012752  006210 7100 00        3890          TRA     A$OK                AND EXIT
      012753  011357 2210 00        3891 DSUB     LDX1    RNGE                GET POINTER TO CURRENT RANGE
      012754  035221 0610 00        3892          ADX1    T$PROG              MAKE POINTER ABSOLUTE
   END OF BINARY CARD 00000203
      012755  000003 7200 11        3893          LXL0    3,1                 GET SAVED IC LABEL IF ANY
      012756  012762 6010 00        3894          TNZ     DSUB1               TRANSFER IF LABEL IS ALREADY DEFINED
      012757  016464 7200 00        3895          LXL0    GLBL                GET UNIQUE LABEL IN XR - 0
      012760  016464 0540 00        3896          AOS     GLBL                STEP LABEL GENERATOR
      012761  000003 4400 11        3897          SXL0    3,1                 AND STORE IN PROG TABLE ENTRY
      012762  000004 7200 11        3898 DSUB1    LXL0    4,1                 GET SAVED LENGTH LABEL IF ANY
      012763  012767 6010 00        3899          TNZ     DSUB2               TRANSFER IF LABEL IS ALREADY DEFINED
      012764  016464 7200 00        3900          LXL0    GLBL                GET UNIQUE LABEL IN XR - 0
      012765  016464 0540 00        3901          AOS     GLBL                STEP LABEL GENERATOR
      012766  000004 4400 11        3902          SXL0    4,1                 AND STORE IN PROG TABLE ENTRY
      012767  016464 7200 00        3903 DSUB2    LXL0    GLBL                GET A UNIQUE LABEL
      012770  016464 0540 00        3904          AOS     GLBL                STEP LABEL GENERATOR
      012771  035235 4400 56      3905        SXL0     A$STACK,ID        AND STORE IN CONTROL STACK
      012772  005742 7170 00      3906        XED      T$SOVF            CHECK FOR STACK OVERFLOW
      012773  000000 6350 10      3907        EAA      0,0               GET LABEL IN AU
      012774  000022 7710 00      3908        ARL      18                MOVE TO AL
      012775  016507 0750 03      3909        ADA      O$JUMP,DU         ADD JUMP CODE
      012776  012614 7000 00      3910        TSX0     CAD               AND ADD TO CODE
      012777  011357 2210 00      3911        LDX1     RNGE              GET POINTER TO CURRENT RANGE
      013000  035221 0610 00      3912        ADX1     T$PROG            MAKE POINTER ABSOLUTE
      013001  000003 2350 11      3913        LDA      3,1               GET SAVED IC LABEL IN AL
      013002  777777 3750 07      3914        ANA      -1,DL             ZERO OUT AU
   END OF BINARY CARD 00000204
      013003  016660 0750 03      3915        ADA      O$EPDN,DU         ADD ENTER PROCEDURE CODE
      013004  012614 7000 00      3916        TSX0     CAD               AND ADD TO CODE
      013005  011357 2210 00      3917        LDX1     RNGE              GET RANGE POINTER IN XR - 1
      013006  035221 0610 00      3918        ADX1     T$PROG            MAKE POINTER ABSOLUTE
      013007  000002 2350 11      3919        LDA      2,1               GET POINTER TO SURROUNDING RANGE IN AU
      013010  000022 7710 00      3920        ARL      18                MOVE TO AL
      013011  016641 0750 03      3921        ADA      O$LL,DU           ADD SET LL CODE
      013012  012614 7000 00      3922        TSX0     CAD               AND ADD TO CODE
      013013  016746 6350 00      3923        EAA      O$DSUB            GET START DECLARER BOX CODE
      013014  012614 7000 00      3924        TSX0     CAD               AND ADD TO OUTPUT CODE
      013015  006210 7100 00      3925        TRA      A$OK              AND EXIT
      013016  011357 2350 00      3926 DBUS   LDA      RNGE              GET POINTER TO CURRENT RANGE IN AU
      013017  000022 7710 00      3927        ARL      18                MOVE TO AL
      013020  016751 0750 03      3928        ADA      O$DBUS,DU         ADD DO BUS CODE
      013021  012614 7000 00      3929        TSX0     CAD               AND ADD TO OUTPUT CODE
      013022  016663 6350 00      3930        EAA      O$RETN            GET RETURN CODE IN AU
      013023  000001 0750 07      3931        ADA      M$VOID,DL         ADD VOID MODE IN AL
      013024  012614 7000 00      3932        TSX0     CAD               AND ADD TO CODE
      013025  011357 2210 00      3933        LDX1     RNGE              GET POINTER TO CURRENT RANGE IN XR - 1
      013026  035221 0610 00      3934        ADX1     T$PROG            MAKE RANGE POINTER ABSOLUTE
      013027  000002 2350 11      3935        LDA      2,1               GET POINTER TO EXTERNAL RANGE IN AU
      013030  000022 7710 00      3936        ARL      18                MOVE TO AL
   END OF BINARY CARD 00000205
      013031  016644 0750 03      3937        ADA      O$LLE,DU          ADD END LL CODE
      013032  012614 7000 00      3938        TSX0     CAD               AND ADD TO CODE
      013033  011357 2210 00      3939        LDX1     RNGE              GET POINTER TO CURRENT RANGE IN XR - 1
      013034  035221 0610 00      3940        ADX1     T$PROG            MAKE POINTER ABSOLUTE
      013035  000004 2350 11      3941        LDA      4,1               GET LENGTH LABEL IN AL
      013036  777777 3750 07      3942        ANA      -1,DL             ZERO OUT AU
      013037  016674 0750 03      3943        ADA      O$DLEN,DU         ADD DEFINE LENGTH COMMAND
      013040  012614 7000 00      3944        TSX0     CAD               AND ADD TO CODE
      013041  035235 2350 54      3945        LDA      A$STACK,DI        GET TRANSFER LABEL IN AL
      013042  777777 3750 07      3946        ANA      -1,DL             ZERO OUT AU
      013043  016504 0750 03      3947        ADA      O$LBL,DU          ADD DEFINE LABEL CODE
      013044  012614 7000 00      3948        TSX0     CAD               AND ADD TO CODE
      013045  006210 7100 00      3949        TRA      A$OK              AND EXIT
      013046  035217 2210 03      3950 CBOX   LDX1     T$BOUND,DU        GET POINTER TO BOUND TABLE CONTROL WORD IN XR - 1
      013047  000002 6350 00      3951        EAA      2                 GET LENGTH OF ENTRY IN AU
      013050  005663 7000 00      3952        TSX0     T$ALOC            ALLOCATE ROW ENTRY IN BOUND TABLE
```

```
          2                                                    PASS 2

     013051  016770 2200 03        3953          LDX0     B$ROW,DU         GET HEADER FOR TABLE ENTRY
     013052  000000 7400 11        3954          STX0     0,1              AND STORE IN TABLE ENTRY
     013053  016457 7200 00        3955          LXL0     CNT              GET NUMBER OF COMMAS IN BOX
     013054  000001 0600 03        3956          ADX0     1,DU             GET DIMENSION OF ARRAY
     013055  000000 4400 11        3957          SXL0     0,1              AND STORE IN TABLE ENTRY
     013056  000001 4500 11        3958          STZ      1,1              CLEAR OUT SECOND WORD OF TABLE ENTRY
   END OF BINARY CARD 00000206
     013057  011357 2200 00        3959          LDX0     RNGE             GET POINTER TO CURRENT RANGE
     013060  000001 4400 11        3960          SXL0     1,1              AND STORE IN TABLE ENTRY
     013061  035217 1610 00        3961          SBX1     T$BOUND          MAKE POINTER TO TABLE ENTRY RELATIVE
     013062  035234 4410 56        3962          SXL1     A$WORK,ID        AND STORE IN WORKING STACK
     013063  005754 7170 00        3963          XED      T$WOVF           CHECK FOR STACK OVERFLOW
     013064  006210 7100 00        3964          TRA      A$OK             AND EXIT
     013065  016460 2350 00        3965 EBOX     LDA      DECLR            GET DECLARER FOR ELEMENT OF ARRAY
     013066  000000 6210 05        3966          EAX1     0,AL             GET BOUND INFORMATION IN XR - 1
     013067  000000 6350 01        3967          EAA      0,AU             GET MODE OF ELEMENT CLEAN IN AU
     013070  013123 7550 00        3968          STA      EBOXM+1          AND STORE FOR MATCH ROUTINE
     013071  035234 7220 54        3969          LXL2     A$WORK,DI        GET POINTER TO BOUND INFORMATION FOR ARRAY
     013072  035217 0620 00        3970          ADX2     T$BOUND          MAKE POINTER ABSOLUTE
     013073  000001 7410 12        3971          STX1     1,2              STORE POINTER TO ELEMENT BOUND IN ARRAY BOUND
     013074  000001 7230 12        3972          LXL3     1,2              GET POINTER TO PROPER PROG TABLE ENTRY IN XR - 3
     013075  035221 0630 00        3973          ADX3     T$PROG           MAKE POINTER ABSOLUTE
     013076  000000 6270 01        3974          EAX7     0,AU             GET MODE OF ELEMENT IN XR - 7
     013077  000005 4470 13        3975          SXL7     5,3              AND STORE IN PROG TABLE ENTRY
     013100  000000 2350 12        3976          LDA      0,2              GET DIMENSION OF ARRAY IN AL
     013101  777777 3750 07        3977          ANA      -1,DL            ZERO OUT AU
     013102  000000 5310 00        3978          NEG                       GET MINUS NUMBER OF DIMENSIONS IN A
     013103  013124 7550 00        3979          STA      EBOXC            AND STORE FOR COUNTING
     013104  016776 6350 00        3980          EAA      M$ROWE           GET INITIAL HEADER FOR MODE IN A
   END OF BINARY CARD 00000207
     013105  013122 7550 00        3981          STA      EBOXM+0          AND STORE FOR MATCH ROUTINE
     013106  000002 2210 03        3982 EBOX1    LDX1     2,DU             GET LENGTH OF ELEMENT IN XR - 1
     013107  013122 2220 03        3983          LDX2     EBOXM,DU         GET POINTER TO PROTOTYPE ENTRY TO MATCH IN XR - 2
     013110  012461 7000 00        3984          TSX0     MATCH            MATCH PROTOTYPE TO MODE TABLE
     013111  016770 6350 00        3985          EAA      M$ROW            GET HEADER FOR SUBSEQUENT MODE TABLE ENTRIES
     013112  013122 7550 00        3986          STA      EBOXM            AND STORE FOR MATCH ROUTINE
     013113  013123 7440 00        3987          STX4     EBOXM+1          STORE NEW MODE TABLE POINTER FOR MATCHING
     013114  013124 0540 00        3988          AOS      EBOXC            COUNT NUMBER OF TIMES TO DO THIS
     013115  013106 6010 00        3989          TNZ      EBOX1            TRANSFER IF MORE TO DO
     013116  035234 7440 51        3990          STX4     A$WORK,I         STORE MODE OF ARRAY IN WORK WITH BOUND
     013117  035234 2350 51        3991          LDA      A$WORK,I         GET NEW DECLARER IN A
     013120  016460 7550 00        3992          STA      DECLR            AND STORE AS DECLARER
     013121  006210 7100 00        3993          TRA      A$OK             AND EXIT
     013122  000000000000          3994 EBOXM    OCT      0,0
     013123  000000000000
     013124  000000 000000         3995 EBOXC    ZERO
     013125  035235 2270 54        3996 EPDEN    LDX7     A$STACK,DI       GET MODE OF PROCEDURE DENOTATION
     013126  013202 7470 00        3997          STX7     EPDNM            AND SAVE IN LOCAL LOCATION
     013127  010630 7000 00        3998          TSX0     A$XFER           MAKE MODE POINTER UNIQUE AND ABSOLUTE
     013130  035235 7200 54        3999          LXL0     A$STACK,DI       GET NUMBER OF PARAMETERS + 2 IN XR - 0
```

```
              2                                                   PASS 2

      013131  013204 7400 00         4000          STX0     EPDNT               AND SAVE
      013132  777777 0670 17         4001          ADX7     -1,7                GET POINTER TO END OF MODE IN XR - 7
   END OF BINARY CARD 00000208
      013133  777777 2270 17         4002          LDX7     -1,7                GET MODE OF RESULT OF PROCEDURE
      013134  016460 7470 00         4003          STX7     DECLR               AND STORE AS TARGET MODE FOR PROCEDURE BODY
      013135  035234 2210 00         4004          LDX1     A$WORK              GET POINTER TO END OF CONTROL STACK
      013136  777777 1610 11         4005          SBX1     -1,1                GET POINTER TO LAST CONTROL BLOCK IN XR - 1
      013137  035234 1610 00         4006          SBX1     A$WORK              MAKE CONTROL BLOCK POINTER RELATIVE
      013140  016446 7410 00         4007          STX1     VALP                AND STORE AS CURRENT VALUE POINTER
      013141  015072 7000 00         4008          TSX0     STRNG               COERCE BODY OF PROCEDURE DENOTATION
      013142  016446 2210 00         4009          LDX1     VALP                GET POINTER TO CURRENT CONTROL BLOCK
      013143  035234 0610 00         4010          ADX1     A$WORK              MAKE POINTER ABSOLUTE
      013144  000002 7200 11         4011          LXL0     2,1                 GET LENGTH OF PROC DENOTATION IN XR - 0
      013145  000002 0600 11         4012          ADX0     2,1                 ADD LENGTH TO GET END ADDRESS
      013146  013204 1600 00         4013          SBX0     EPDNT               SUBTRACT ACTUAL START ADDRESS
      013147  000002 4400 11         4014          SXL0     2,1                 STORE NEW LENGTH IN CONTROL BLOCK
      013150  013204 2200 00         4015          LDX0     EPDNT               GET LOCATION OF PROC DENOTATION
      013151  000002 7400 11         4016          STX0     2,1                 STORE NEW LOCATION IN CONTROL BLOCK
      013152  013202 2350 00         4017          LDA      EPDNM               GET MODE OF PROCEDURE DENOTATION
      013153  000000 6350 01         4018          EAA      0,AU                ZERO OUT AL
      013154  000001 7550 11         4019          STA      1,1                 AND STORE AS MODE IN CONTROL BLOCK FOR VALUE
      013155  011357 2350 00         4020          LDA      RNGE                GET CURRENT RANGE NUMBER IN AU
      013156  000022 7710 00         4021          ARL      18                  MOVE TO AL
      013157  016652 0750 03         4022          ADA      O$SRNGE,DU          ADD SRNGE COMMAND
      013160  012403 7000 00         4023          TSX0     INS                 AND INSERT IN FRONT OF PROCEDURE DENOTATION CODE
   END OF BINARY CARD 00000209
      013161  011357 2350 00         4024          LDA      RNGE                GET CURRENT RANGE NUMBER IN AU
      013162  000022 7710 00         4025          ARL      18                  MOVE TO AL
      013163  016655 0750 03         4026          ADA      O$ERNGE,DU          ADD ERNGE COMMAND
      013164  012375 7000 00         4027          TSX0     ADD                 AND ADD AFTER PROCEDURE DENOTATION CODE
      013165  016446 2200 00         4028          LDX0     VALP                GET POINTER TO VALUE CONTROL BLOCK
      013166  035234 0600 00         4029          ADX0     A$WORK              MAKE POINTER ABSOLUTE
      013167  777776 3350 07         4030          LCA      -2,DL               GET [-1, +2] IN A REGISTER
      013170  000002 0550 10         4031          ASA      2,0                 UPDATE LOC/LEN WORD TO INCLUDE SRNGE AND ERNGE
      013171  006210 7100 00         4032          TRA      A$OK                AND EXIT
      013172  035234 2210 00         4033 EPDNE    LDX1     A$WORK              GET POINTER TO END OF WORKING STACK
      013173  777777 1610 11         4034          SBX1     -1,1                GET POINTER TO TOP CONTROL BLOCK IN WORKING STACK
      013174  035224 7200 00         4035          LXL0     T$CODE              GET CURRENT END OF OUTPUT CODE
      013175  000002 1600 11         4036          SBX0     2,1                 SUBTRACT START OF PROC DENOTATION
      013176  000002 4400 11         4037          SXL0     2,1                 AND STORE NEW PROC DENOTATION LENGTH
      013177  013202 2270 00         4038          LDX7     EPDNM               GET MODE OF PROCEDURE DENOTATION
      013200  014530 7000 00         4039          TSX0     PCDR                MAKE VALUE A PROCEDURE
      013201  006210 7100 00         4040          TRA      A$OK                AND EXIT
      013202  000000 000000          4041 EPDNM    ZERO
      013203  000000 000000          4042 EPDNL    ZERO
      013204  000000 000000          4043 EPDNT    ZERO
      013205  000001 6350 00         4044 VOID     EAA      M$VOID              GET VOID MODE IN AU
      013206  016460 7550 00         4045          STA      DECLR               STORE AS CURRENT MODE
   END OF BINARY CARD 00000210
      013207  777773 6350 00         4046          EAA      -5                  GET VALUE POINTER TO LAST VALUE IN WORK
```

```
              2                                                PASS 2

     013210  016446 7550 00         4047          STA    VALP              AND STORE IN VALUE CONTROL BLOCK POINTER
     013211  015072 7000 00         4048          TSX0   STRNG             COERCE VALUE TO VOID
     013212  006210 7100 00         4049          TRA    A$OK              AND EXIT
     013213  016647 6350 00         4050 ODELV    EAA    O$DELV            GET DELETE VALUE CODE IN AU
     013214  000001 0750 07         4051          ADA    M$VOID,DL         ADD VOID MODE TO AL
     013215  012614 7000 00         4052          TSX0   CAD               AND ADD TO OUTPUT CODE
     013216  006210 7100 00         4053          TRA    A$OK              AND EXIT
     013217  011731 2210 00         4054 GMOD     LDX1   LASTS             GET RELATIVE POINTER TO DEF TABLE ENTRY
     013220  035220 0610 00         4055          ADX1   T$DEF             MAKE POINTER ABSOLUTE
     013221  000002 2350 11         4056          LDA    2,1               GET MODE OF LAST IDENTIFIER
     013222  013257 7550 00         4057          STA    FMODE             AND STORE FOR DECLARATION ROUTINES
     013223  006210 7100 00         4058          TRA    A$OK              AND EXIT
     013224  035234 2210 00         4059 SVAL     LDX1   A$WORK            GET POINTER TO END OF WORKING STACK
     013225  777777 1610 11         4060          SBX1   -1,1              GET POINTER TO LAST CONTROL BLOCK IN XR - 1
     013226  035234 1610 00         4061          SBX1   A$WORK            MAKE POINTER RELATIVE
     013227  016446 7410 00         4062          STX1   VALP              AND STORE AS CURRENT VALUE POINTER
     013230  006210 7100 00         4063          TRA    A$OK              AND EXIT
     013231  000000 2360 05         4064 PUSH     LDQ    0,AL              FETCH INDICATED DATA
     013232  035235 7560 56         4065          STQ    A$STACK,ID        AND SAVE IN CONTROL STACK
     013233  005742 7170 00         4066          XED    T$SOVF            CHECK FOR STACK OVERFLOW
     013234  006210 7100 00         4067          TRA    A$OK              AND EXIT
 END OF BINARY CARD 00000211
     013235  035235 2360 54         4068 POP      LDQ    A$STACK,DI        RECOVER DATA IN Q REGISTER
     013236  000000 7560 05         4069          STQ    0,AL              AND RESTORE TO INDICATED PLACE
     013237  006210 7100 00         4070          TRA    A$OK              AND EXIT
     013240  000000 2350 05         4071 PUSHW    LDA    0,AL              GET WORD SPECIFIED BY ARGUMENT
     013241  035234 7550 56         4072          STA    A$WORK,ID         AND PUSH IT INTO WORKING STACK
     013242  005754 7170 00         4073          XED    T$WOVF            CHECK FOR STACK OVERFLOW
     013243  006210 7100 00         4074          TRA    A$OK              AND EXIT
     013244  000000 2350 05         4075 PUSEA    LDA    0,AL              GET WORD SPECIFIED BY ARGUMENT
     013245  000000 6350 01         4076          EAA    0,AU              ZERO OUT LOWER HALF
     013246  035234 7550 56         4077          STA    A$WORK,ID         AND PUSH IT INTO WORKING STACK
     013247  005754 7170 00         4078          XED    T$WOVF            CHECK FOR STACK OVERFLOW
     013250  006210 7100 00         4079          TRA    A$OK              AND EXIT
     013251  016460 2360 00         4080 SAVE     LDQ    DECLR             GET CURRENT DECLARER
     013252  000000 7560 05         4081          STQ    0,AL              AND STORE IN DESIGNATED LOCATION
     013253  006210 7100 00         4082          TRA    A$OK              AND EXIT
     013254  000000 2360 05         4083 REST     LDQ    0,AL              FETCH FROM DESIGNATED LOCATION
     013255  016460 7560 00         4084          STQ    DECLR             AND STORE AS CURRENT DECLARER
     013256  006210 7100 00         4085          TRA    A$OK              AND EXIT
     013257  000000 000000          4086 FMODE    ZERO
     013260  000000 0540 05         4087 ADO      AOS    0,AL              ADD ONE TO LOCATION SPECIFIED BY ARGUMENT
     013261  006210 7100 00         4088          TRA    A$OK              AND EXIT
     013262  000000 2200 03         4089 STZ      LDX0   0,DU              GET A HALF WORD ZERO
 END OF BINARY CARD 00000212
     013263  000000 7400 05         4090          STX0   0,AL              AND STORE IN UPPER HALF OF ARGUMENT
     013264  006210 7100 00         4091          TRA    A$OK              AND EXIT
     013265  000000 4500 05         4092 CLEAR    STZ    0,AL              ZERO OUT WORD SPECIFIED BY ARGUMENT
     013266  006210 7100 00         4093          TRA    A$OK              AND EXIT
     013267  016537 2350 03         4094 SHEAP    LDA    O$HGEN,DU         GET HEAP CODE
     013270  016466 7550 00        4095        STA      GTYPE             STORE FOR TYPE OF GENERATOR
     013271  006210 7100 00        4096        TRA      A$OK              AND EXIT
     013272  016534 2350 03        4097 SLOC   LDA      O$LGEN,DU         GET LOC CODE
     013273  016466 7550 00        4098        STA      GTYPE             STORE FOR TYPE OF GENERATOR
     013274  006210 7100 00        4099        TRA      A$OK              AND EXIT
     013275  016466 4500 00        4100 SBLNK  STZ      GTYPE             SET TYPE OF GENERATOR TO UNSPECIFIED
     013276  006210 7100 00        4101        TRA      A$OK              AND EXIT
     013277  016466 2340 00        4102 SHPBK  SZN      GTYPE             IS DECLARER TYPE UNSPECIFIED
     013300  006210 6010 00        4103        TNZ      A$OK              NO - NOTHING TO DO
     013301  016537 2350 03        4104        LDA      O$HGEN,DU         GET HEAP CODE
     013302  016466 7550 00        4105        STA      GTYPE             AND STORE FOR TYPE OF GENERATOR
     013303  006210 7100 00        4106        TRA      A$OK              AND EXIT
     013304  016466 2340 00        4107 SLCBK  SZN      GTYPE             HAS GENERATOR A TYPE YET
     013305  006210 6010 00        4108        TNZ      A$OK              YES - EXIT
     013306  016534 2350 03        4109        LDA      O$LGEN,DU         GET LOCAL GENERATOR CODE
     013307  016466 7550 00        4110        STA      GTYPE             AND STORE AS TYPE OF GENERATOR
     013310  006210 7100 00        4111        TRA      A$OK              AND EXIT
  END OF BINARY CARD 00000213
     013311  016677 6350 00        4112 SBCT   EAA      O$VSBCT           GET SUBSCRIPT CODE
     013312  013333 7100 00        4113        TRA      ACNT              GO ADD SUBSCRIPT POSITION IN AL
     013313  016702 6350 00        4114 LWB    EAA      O$VLWB            GET LOWER BOUND CODE
     013314  013333 7100 00        4115        TRA      ACNT              GO ADD SUBSCRIPT POSITION IN AL
     013315  016705 6350 00        4116 UPB    EAA      O$VUPB            GET UPPER BOUND CODE
     013316  013333 7100 00        4117        TRA      ACNT              GO ADD SUBSCRIPT POSITION IN AL
     013317  016710 6350 00        4118 NLWB   EAA      O$VNLWB           GET NEW LOWER BOUND CODE
     013320  013333 7100 00        4119        TRA      ACNT              GO ADD SUBSCRIPT POSITION IN AL
     013321  016713 6350 00        4120 EPTY   EAA      O$VEPTY           GET EMPTY POSITION CODE
     013322  013333 7100 00        4121        TRA      ACNT              GO ADD SUBSCRIPT POSITION IN AL
     013323  016716 6350 00        4122 OLWB   EAA      O$LWB             GET SET LOWER BOUND CODE
     013324  013333 7100 00        4123        TRA      ACNT              GO ADD SUBSCRIPT POSITION IN AL
     013325  016721 6350 00        4124 OUPB   EAA      O$UPB             GET SET UPPER BOUND CODE
     013326  013333 7100 00        4125        TRA      ACNT              GO ADD SUBSCRIPT POSITION IN AL
     013327  016724 6350 00        4126 OFIX   EAA      O$FIX             GET SET FIXED CODE
     013330  013333 7100 00        4127        TRA      ACNT              GO ADD SUBSCRIPT POSITION IN AL
     013331  016727 6350 00        4128 OFLEX  EAA      O$FLEX            GET SET FLEXIBLE CODE
     013332  013333 7100 00        4129        TRA      ACNT              GO ADD SUBSCRIPT POSITION IN AL
     013333  016457 0750 00        4130 ACNT   ADA      CNT               ADD SUBSCRIPT POSITION IN AL
     013334  000001 0750 07        4131        ADA      1,DL              MAKE COUNTING START WITH ONE
     013335  012614 7000 00        4132        TSX0     CAD               AND ADD TO CODE
     013336  006210 7100 00        4133        TRA      A$OK              AND EXIT
  END OF BINARY CARD 00000214
     013337  016732 6350 00        4134 OSUB   EAA      O$SUB             GET SUB CODE
     013340  012614 7000 00        4135        TSX0     CAD               AND ADD TO CODE
     013341  006210 7100 00        4136        TRA      A$OK              AND EXIT
     013342  013444 4500 00        4137 OBUS   STZ      BUSF              ZERO OUT REFERENCE FLAG
     013343  013450 3350 00        4138        LCA      SBCNT             GET MINUS NUMBER OF SUBSCRIPTS IN BOX
     013344  013445 7550 00        4139        STA      BUSC              AND STORE FOR COUNTING
     013345  000001 2200 03        4140        LDX0     1,DU              GET A ONE IN XR - 0
     013346  013446 4400 00        4141        SXL0     BUSD              AND STORE IN COMMAND
     013347  011357 2210 00        4142        LDX1     RNGE              GET POINTER TO CURRENT RANGE IN XR - 1
```

```
            2                                            PASS 2

      013350  035221 0610 00          4143          ADX1    T$PROG              MAKE POINTER ABSOLUTE
      013351  000005 2260 11          4144          LDX6    5,1                 GET NUMBER OF COMMAS IN RANGE
      013352  035234 2210 00          4145          LDX1    A$WORK              GET POINTER TO END OF WORKING STACK
      013353  777777 1610 11          4146          SBX1    -1,1                GET POINTER TO LAST BLOCK IN XR - 1
      013354  000001 2270 11          4147          LDX7    1,1                 GET MODE OF BASE IN XR - 7
      013355  013447 7470 00          4148          STX7    BUSM                SAVE MODE OF SLICE
      013356  010630 7000 00          4149          TSX0    A$XFER              MAKE MODE POINTER ABSOLUTE
      013357  000000 2200 17          4150          LDX0    0,7                 GET TYPE OF MODE IN XR - 0
      013360  016762 1000 03          4151          CMPX0   M$REF,DU            IS IT A REFERENCE MODE
      013361  013371 6010 00          4152          TNZ     OBUS2               TRANSFER IF NOT REFERENCE
      013362  000006 2350 03          4153          LDA     O$RBUS-O$BUS,DU     GET FLAG FOR REFERENCE SLICE
      013363  013444 7550 00          4154          STA     BUSF                AND STORE IN BUS FLAG
      013364  000001 2200 17          4155          LDX0    1,7                 GET DEREFERENCED MODE IN XR - 0
  END OF BINARY CARD 00000215
      013365  013447 7400 00          4156          STX0    BUSM                AND SAVE
      013366  000001 2270 17          4157 OBUS1    LDX7    1,7                 GET DEREFERENCED MODE IN XR - 7
      013367  010630 7000 00          4158          TSX0    A$XFER              MAKE IT ABSOLUTE
      013370  000000 2200 17          4159          LDX0    0,7                 GET TYPE OF MODE IN XR - 0
      013371  016770 1000 03          4160 OBUS2    CMPX0   M$ROW,DU            IS IT A ROW MODE
      013372  013375 6000 00          4161          TZE     OBUS3               YES - OK
      013373  016776 1000 03          4162          CMPX0   M$ROWE,DU           IS IT AN END ROW MODE
      013374  777777 6010 00          4163          TNZ     $ERROR              NO - CANNOT SUBSCRIPT A NONROW BASE
      013375  013446 0540 00          4164 OBUS3    AOS     BUSD                INCREMENT COMMAND TO NEXT POSITION
      013376  000001 1660 03          4165          SBX6    1,DU                DECREMENT NUMBER OF COMMAS LEFT
      013377  013366 6050 00          4166          TPL     OBUS1               TRANSFER IF MORE TO GO
      013400  000001 2270 17          4167          LDX7    1,7                 GET DEROWED MODE IN XR - 7
      013401  010630 7000 00          4168 OB1      TSX0    A$XFER              MAKE MODE POINTER ABSOLUTE
      013402  000000 2200 17          4169          LDX0    0,7                 GET TYPE OF MODE IN XR - 7
      013403  016770 1000 03          4170          CMPX0   M$ROW,DU            IS IT A ROW MODE
      013404  013407 6000 00          4171          TZE     OB2                 YES - TRANSFER
      013405  016776 1000 03          4172          CMPX0   M$ROWE,DU           IS IT AN END ROW MODE
      013406  013414 6010 00          4173          TNZ     OB3                 NO - TRANSFER
      013407  000001 2270 17          4174 OB2      LDX7    1,7                 GET DEROWED MODE IN XR - 7
      013410  013446 2350 00          4175          LDA     BUSD                GET EMPTY POSITION COMMAND
      013411  012614 7000 00          4176          TSX0    CAD                 AND ADD TO OUTPUT CODE
      013412  013446 0540 00          4177          AOS     BUSD                STEP TO NEXT POSITION
  END OF BINARY CARD 00000216
      013413  013401 7100 00          4178          TRA     OB1                 AND LOOP
      013414  013447 2270 00          4179 OB3      LDX7    BUSM                GET POSSIBLY DEREFERENCED MODE OF BASE
      013415  013445 2340 00          4180 OB4      SZN     BUSC                ARE THERE ANY MORE SUBSCRIPTS
      013416  013423 6000 00          4181          TZE     OBUS6               NO - TRANSFER
      013417  013445 0540 00          4182          AOS     BUSC                DECREMENT NUMBER OF REMAINING SUBSCRIPTS
      013420  010630 7000 00          4183          TSX0    A$XFER              MAKE MODE POINTER ABSOLUTE
      013421  000001 2270 17          4184          LDX7    1,7                 GET DEROWED MODE IN XR - 7
      013422  013415 7100 00          4185          TRA     OB4                 AND LOOP
      013423  013444 2340 00          4186 OBUS6    SZN     BUSF                IS IT A REFERENCE SLICE
      013424  013426 6000 00          4187          TZE     OBUS7               TRANSFER IF NOT REFERENCE
      013425  012646 7000 00          4188          TSX0    EREF1               REFERENCE MODE OF ELEMENT
      013426  035234 2210 00          4189 OBUS7    LDX1    A$WORK              GET POINTER TO END OF WORK
      013427  777777 1610 11          4190          SBX1    -1,1                GET POINTER TO BASE CONTROL BLOCK
      013430  000001 7470 11       4191         STX7     1,1               STORE NEW MODE IN CONTROL BLOCK
      013431  000000 6350 17       4192         EAA      0,7               GET MODE OF RESULT IN AU
      013432  000022 7710 00       4193         ARL      18                MOVE TO AL
      013433  016735 0750 03       4194         ADA      O$BUS,DU          ADD BUS CODE
      013434  013444 0750 00       4195         ADA      BUSF              CHANGE BUS TO RBUS IF REFERENCE SLICE
      013435  012614 7000 00       4196         TSX0     CAD               AND ADD TO CODE
      013436  035234 2210 00       4197         LDX1     A$WORK            GET POINTER TO END OF WORKING STACK
      013437  777777 1610 11       4198         SBX1     -1,1              GET POINTER TO SLICE CONTROL BLOCK
      013440  035224 7200 00       4199         LXL0     T$CODE            GET POINTER TO END OF CODE
   END OF BINARY CARD 00000217
      013441  000002 1600 11       4200         SBX0     2,1               SUBTRACT ADDRESS OF START OF SLICE CODE
      013442  000002 4400 11       4201         SXL0     2,1               AND STORE AS NEW SLICE CODE LENGTH
      013443  006210 7100 00       4202         TRA      A$OK              AND EXIT
      013444  000000 000000        4203 BUSF    ZERO
      013445  000000 000000        4204 BUSC    ZERO
      013446  016713 000000        4205 BUSD    ZERO     O$VEPTY,0
      013447  000000 000000        4206 BUSM    ZERO
      013450  000000 000000        4207 SBCNT   ZERO
      013451  777773 2210 03       4208 SINT    LDX1     -5,DU             GET RELATIVE POINTER TO LAST CONTROL BLOCK
      013452  016446 7410 00       4209         STX1     VALP              AND STORE AS POINTER TO CURRENT BLOCK
      013453  000007 6350 00       4210         EAA      M$INT             GET AN INTEGRAL MODE
      013454  016460 7550 00       4211         STA      DECLR             AND STORE AS DESIRED MODE
      013455  015072 7000 00       4212         TSX0     STRNG             MAKE BOUND AN INTEGER
      013456  006210 7100 00       4213         TRA      A$OK              AND EXIT
      013457  013472 2340 00       4214 SACT    SZN      DFLG              CHECK VIRTUAL/ACTUAL FLAG
      013460  777777 6040 00       4215         TMI      $ERROR            ERROR - ALREADY VIRTUAL
      013461  000001 2350 03       4216         LDA      1,DU              GET ACTUAL FLAG
      013462  013472 7550 00       4217         STA      DFLG              AND STORE IN VIRTUAL/ACTUAL FLAG
      013463  006210 7100 00       4218         TRA      A$OK              AND EXIT
      013464  013472 2340 00       4219 SVIRT   SZN      DFLG              CHECK VIRTUAL/ACTUAL FLAG
      013465  013467 6000 00       4220         TZE      SV1               TRANSFER IF NOT YET SET
      013466  777777 6050 00       4221         TPL      $ERROR            ERROR - ALREADY ACTUAL
   END OF BINARY CARD 00000218
      013467  000001 3350 03       4222 SV1     LCA      1,DU              GET VIRTUAL FLAG
      013470  013472 7550 00       4223         STA      DFLG              AND STORE IN VIRTUAL/ACTUAL FLAG
      013471  006210 7100 00       4224         TRA      A$OK              AND EXIT
      013472  000000 000000        4225 DFLG    ZERO
      013473  016462 7470 00       4226 STID    STX7     IDNT              STORE POINTER TO LAST SYMBOL IN IDNT
      013474  006210 7100 00       4227         TRA      A$OK              AND EXIT
      013475  016457 2350 00       4228 PARN    LDA      CNT               GET NUMBER OF UNITS
      013476  000001 0750 07       4229         ADA      1,DL              ADD ONE FOR THE LAST UNIT
      013477  040000 0750 03       4230         ADA      W$PAR,DU          ADD CONTROL WORD
      013500  000000 2360 03       4231         LDQ      0,DU              GET A ZERO FOR NO RANGE INFORMATION
      013501  013517 7100 00       4232         TRA      BALN1             AND CONTINUE
      013502  020000 2350 03       4233 BAL2    LDA      W$BAL,DU          GET CONTROL WORD
      013503  000002 0750 07       4234         ADA      2,DL              ADD NUMBER OF UNITS
      013504  000000 2360 03       4235         LDQ      0,DU              GET A ZERO FOR NO RANGE INFORMATION
      013505  013517 7100 00       4236         TRA      BALN1             AND CONTINUE
      013506  016457 2350 00       4237 BALZR   LDA      CNT               GET NUMBER OF UNITS
      013507  000001 0750 07       4238         ADA      1,DL              ADD ONE FOR THE LAST UNIT
```

```
          2                                           PASS 2

     013510  020000 0750 03       4239          ADA      W$BAL,DU          ADD CONTROL WORD
     013511  000000 2360 03       4240          LDQ      0,DU              GET A ZERO FOR NO RANGE INFORMATION
     013512  013517 7100 00       4241          TRA      BALN1             AND CONTINUE
     013513  016457 2350 00       4242 BALN     LDA      CNT               GET NUMBER OF UNITS
     013514  000001 0750 07       4243          ADA      1,DL              ADD ONE FOR THE LAST UNIT
 END OF BINARY CARD 00000219
     013515  020000 0750 03       4244          ADA      W$BAL,DU          ADD CONTROL WORD
     013516  011357 2360 00       4245          LDQ      RNGE              GET CURRENT RANGE NUMBER IN QU
     013517  035234 7550 56       4246 BALN1    STA      A$WORK,ID         AND STORE IN WORKING STACK
     013520  005754 7170 00       4247          XED      T$WOVF            CHECK FOR STACK OVERFLOW
     013521  000000 6350 02       4248          EAA      0,QU              GET RANGE NUMBER OR ZERO IN AU
     013522  000022 7710 00       4249          ARL      18                MOVE TO LOWER HALF OF MODE WORD
     013523  035234 7550 56       4250          STA      A$WORK,ID         AND STORE IN WORKING STACK
     013524  005754 7170 00       4251          XED      T$WOVF            CHECK FOR STACK OVERFLOW
     013525  035235 2350 54       4252          LDA      A$STACK,DI        GET POINTER TO START OF CODE FOR VALUE
     013526  000000 6350 05       4253          EAA      0,AL              PUT IN AU
     013527  035234 7550 56       4254          STA      A$WORK,ID         STORE LOC/LEN IN WORKING STACK
     013530  005754 7170 00       4255          XED      T$WOVF            CHECK FOR STACK OVERFLOW
     013531  035234 4500 56       4256          STZ      A$WORK,ID         STORE ZERO IN WORKING STACK
     013532  005754 7170 00       4257          XED      T$WOVF            CHECK FOR STACK OVERFLOW
     013533  000005 6350 00       4258          EAA      5                 GET LENGTH OF CONTROL ELEMENT IN WORKING STACK
     013534  035234 7550 56       4259          STA      A$WORK,ID         STORE IN WORKING STACK
     013535  005754 7170 00       4260          XED      T$WOVF            CHECK FOR STACK OVERFLOW
     013536  006210 7100 00       4261          TRA      A$OK              AND EXIT
     013537  035234 2200 00       4262 INLL     LDX0     A$WORK            GET POINTER TO END OF CURRENT WORKING STACK
     013540  035214 1600 00       4263          SBX0     T$WORK            MAKE POINTER RELATIVE
     013541  035235 7400 56       4264          STX0     A$STACK,ID        AND SAVE IN CONTROL STACK
     013542  005742 7170 00       4265          XED      T$SOVF            CHECK FOR STACK OVERFLOW
 END OF BINARY CARD 00000220
     013543  013573 2350 00       4266          LDA      RLL               GET CURRENT RANGE LL WORD
     013544  035235 7550 56       4267          STA      A$STACK,ID        AND SAVE IN CONTROL STACK
     013545  005742 7170 00       4268          XED      T$SOVF            CHECK FOR STACK OVERFLOW
     013546  013573 4500 00       4269          STZ      RLL               INITIALIZE LL WORD FOR NEW RANGE
     013547  006210 7100 00       4270          TRA      A$OK              AND EXIT
     013550  035234 2200 00       4271 ORLL     LDX0     A$WORK            GET POINTER TO END OF WORKING STACK
     013551  777777 1600 10       4272          SBX0     -1,0              GET POINTER TO LAST BLOCK IN WORKING STACK
     013552  013573 2350 00       4273          LDA      RLL               GET CURRENT RANGE LL WORD
     013553  000003 2550 10       4274          ORSA     3,0               AND OR IT INTO LAST CONTROL BLOCK
     013554  000001 7350 00       4275          ALS      1                 GET LL WORD FOR NEXT OUTER RANGE
     013555  035235 2750 54       4276          ORA      A$STACK,DI        OR IN SAVED LL FROM PREVIOUS OUTER RANGE
     013556  013573 7550 00       4277          STA      RLL               AND STORE AS CURRENT LL WORD
     013557  035235 2200 54       4278          LDX0     A$STACK,DI        GET LENGTH OF WORK ON RANGE ENTRY
     013560  035214 0600 00       4279          ADX0     T$WORK            MAKE POINTER ABSOLUTE
     013561  013570 7400 00       4280          STX0     ORLL2             AND STORE FOR COMPARES
     013562  035234 2210 00       4281          LDX1     A$WORK            GET POINTER TO END OF WORKING STACK
     013563  013570 7100 00       4282          TRA      ORLL2             GO TO END CHECK
     013564  777777 1610 11       4283 ORLL1    SBX1     -1,1              GET POINTER TO PREVIOUS BLOCK
     013565  000003 2350 11       4284          LDA      3,1               GET LL WORD IN BLOCK
     013566  000001 7350 00       4285          ALS      1                 SHIFT TO OUTER RANGE REPRESENTATION
     013567  000003 7550 11       4286          STA      3,1               AND RESTORE LL WORD
     013570  000000 1010 03         4287 ORLL2  CMPX1     **,DU          IS THIS WHERE STACK WAS MARKED
   END OF BINARY CARD 00000221
     013571  013564 6010 00         4288        TNZ       ORLL1          TRANSFER IF MORE BLOCKS TO MODIFY
     013572  006210 7100 00         4289        TRA       A$OK           AND EXIT
     013573  000000 000000          4290 RLL    ZERO
     013574  400000 6350 00         4291 WGEN   EAA       W$VALUE        GET HEADER WORD IN A
     013575  035234 7550 56         4292        STA       A$WORK,ID      STORE IN WORKING STACK
     013576  005754 7170 00         4293        XED       T$WOVF         CHECK FOR STACK OVERFLOW
     013577  016460 2350 00         4294        LDA       DECLR          GET CURRENT DECLARER IN A
     013600  000000 6350 01         4295        EAA       0,AU           CLEAN OFF BOUNDS INFORMATION
     013601  035234 7550 56         4296        STA       A$WORK,ID      STORE IN WORKING STACK
     013602  005754 7170 00         4297        XED       T$WOVF         CHECK FOR STACK OVERFLOW
     013603  035224 2350 00         4298        LDA       T$CODE         GET POINTER TO END OF CODE
     013604  000000 6350 05         4299        EAA       0,AL           PUT IN AU
     013605  000001 0750 07         4300        ADA       1,DL           GENERATE LOC/LEN WORD
     013606  035234 7550 56         4301        STA       A$WORK,ID      STORE IN WORKING STACK
     013607  005754 7170 00         4302        XED       T$WOVF         CHECK FOR STACK OVERFLOW
     013610  035234 4500 56         4303        STZ       A$WORK,ID      ZERO OUT LL WORD
     013611  005754 7170 00         4304        XED       T$WOVF         CHECK FOR STACK OVERFLOW
     013612  000005 6350 00         4305        EAA       5              GET NUMBER OF WORDS IN CONTROL BLOCK
     013613  035234 7550 56         4306        STA       A$WORK,ID      STORE IN WORKING STACK
     013614  005754 7170 00         4307        XED       T$WOVF         CHECK FOR STACK OVERFLOW
     013615  000000 6350 00         4308        EAA       0              ALLOCATE ONE WORD
     013616  035224 2210 03         4309        LDX1      T$CODE,DU      GET POINTER TO CODE TABLE CONTROL WORD
   END OF BINARY CARD 00000222
     013617  005663 7000 00         4310        TSX0      T$ALOC         GET A WORD IN CODE
     013620  016460 2350 00         4311        LDA       DECLR          GET CURRENT DECLARER
     013621  000022 7710 00         4312        ARL       18             MOVE MODE TO AL
     013622  016466 0750 00         4313        ADA       GTYPE          ADD LOCAL OR HEAP GENERATOR
     013623  777777 7550 11         4314        STA       -1,1           STORE IN CODE
     013624  016460 2270 00         4315        LDX7      DECLR          GET MODE OF DECLARER IN XR - 7
     013625  010630 7000 00         4316        TSX0      A$XFER         MAKE MODE POINTER ABSOLUTE
     013626  000001 2270 17         4317        LDX7      1,7            GET DEREFERENCED MODE IN XR - 7
     013627  010562 7000 00         4318        TSX0      A$RCHK         SEE IF THERE IS A BOUND PART IN DECLARER
     013630  013640 7100 00         4319        TRA       WGENX          TRANSFER IF NO BOUND PART IN DECLARER
     013631  016460 2350 00         4320        LDA       DECLR          GET BOUND PART OF DECLARER IN AL
     013632  777777 3750 07         4321        ANA       -1,DL          ZERO OUT AU
     013633  016740 0750 03         4322        ADA       O$BOUND,DU     ADD BOUND CODE
     013634  012614 7000 00         4323        TSX0      CAD            AND ADD TO CODE
     013635  035234 2210 00         4324        LDX1      A$WORK         GET POINTER TO END OF WORKING STACK
     013636  777777 1610 11         4325        SBX1      -1,1           GET POINTER TO LAST BLOCK IN WORK
     013637  000002 0540 11         4326        AOS       2,1            INCLUDE BOUND CODE IN VALUE
     013640  006210 7100 00         4327 WGENX  TRA       A$OK           AND EXIT
     013641  000000 6350 25         4328 ENTL   EAA       0,AL*          GET LABEL IN AU AND INCREMENT LABEL
     013642  000022 7710 00         4329        ARL       18             MOVE LABEL TO AL
     013643  016504 0750 03         4330        ADA       O$LBL,DU       ADD DEFINE LABEL CODE
     013644  035234 2210 00         4331        LDX1      A$WORK         GET POINTER TO END OF WORKING STACK
   END OF BINARY CARD 00000223
     013645  777777 1610 11         4332        SBX1      -1,1           GET POINTER TO LAST BLOCK IN WORKING STACK
     013646  035234 1610 00         4333        SBX1      A$WORK         MAKE CURRENT VALUE POINTER RELATIVE
```

```
           2                                                  PASS 2

     013647   016446 7410 00        4334         STX1    VALP            AND STORE FOR INSERTION ROUTINE
     013650   012403 7000 00        4335         TSX0    INS             INSERT ONE WORD IN FRONT OF CODE
     013651   006210 7100 00        4336         TRA     A$OK            AND EXIT
     013652   013653 0110 56        4337 B1      NOP     B1X,ID
     013653   000000 0000 00        4338 B1X     TALLY   **
     013654   000000 000000         4339 B1S     ZERO
     013655   013656 0110 56        4340 B2      NOP     B2X,ID
     013656   000000 0000 00        4341 B2X     TALLY   **
     013657   000000 000000         4342 B2S     ZERO
     013660   000000 000000         4343 L1      ZERO
     013661   035234 2210 00        4344 ENTC    LDX1    A$WORK          GET POINTER TO END OF WORKING STACK
     013662   777777 1610 11        4345         SBX1    -1,1            MAKE XR - 1 POINT TO LAST CONTROL BLOCK
     013663   000001 2270 11        4346         LDX7    1,1             GET MODE OF LHS TERTIARY
     013664   035234 1610 00        4347         SBX1    A$WORK          MAKE CURRENT VALUE POINTER RELATIVE
     013665   016446 7410 00        4348         STX1    VALP            AND STORE FOR INSERTION ROUTINE
     013666   013714 4470 00        4349         SXL7    ENTCT           AND SAVE MODE
     013667   010630 7000 00        4350         TSX0    A$XFER          MAKE MODE UNIQUE AND ABSOLUTE
     013670   000001 2350 17        4351         LDA     1,7             GET DEREFERENCED MODE IN AU
     013671   000022 7710 00        4352         ARL     18              MOVE MODE TO AL
     013672   016542 0750 03        4353         ADA     O$CONF,DU       ADD CONFORMITY CHECK CODE
 END OF BINARY CARD 00000224
     013673   012403 7000 00        4354         TSX0    INS             INSERT CODE IN FRONT OF VALUE
     013674   013652 2350 51        4355         LDA     B1,I            GET CURRENT LABEL IN BLOCK ONE
     013675   000022 7710 00        4356         ARL     18              PUT IN AL
     013676   016545 0750 03        4357         ADA     O$TF,DU         ADD TRANSFER IF FALSE CODE
     013677   012403 7000 00        4358         TSX0    INS             INSERT CODE IN FRONT OF VALUE
     013700   017020 6350 00        4359         EAA     O$MREF          GET MAKE REFERENCE VALUE BLOCK COMMAND
     013701   012403 7000 00        4360         TSX0    INS             AND INSERT IT IN FRONT OF VALUE
     013702   013714 2350 00        4361         LDA     ENTCT           GET MODE OF TERTIARY IN AL
     013703   016553 0750 03        4362         ADA     O$CASGN,DU      GET CONFORMITY ASSIGN IN AU
     013704   012614 7000 00        4363         TSX0    CAD             ADD AFTER TERTIARY
     013705   013655 6350 20        4364         EAA     B2,*            GET NEXT BLOCK TWO LABEL IN AU
     013706   000022 7710 00        4365         ARL     18              MOVE LABEL TO AL
     013707   016507 0750 03        4366         ADA     O$JUMP,DU       ADD UNCONDITIONAL JUMP CODE
     013710   012614 7000 00        4367         TSX0    CAD             AND ADD TO END OF OUTPUT CODE
     013711   017020 6350 00        4368         EAA     O$MREF          GET MAKE REFERENCE VALUE BLOCK COMMAND
     013712   012614 7000 00        4369         TSX0    CAD             AND ADD TO CODE AFTER VALUE
     013713   014100 7100 00        4370         TRA     DELV            GO TO DELETE VALUE
     013714   000000 000000         4371 ENTCT   ZERO
     013715   000000 000000         4372         ZERO
     013716   016463 4500 00        4373 CSCT    STZ     RELF            SET CONFORMITY FLAG TO CONFORMS TO
     013717   006210 7100 00        4374         TRA     A$OK            AND EXIT
     013720   000001 2350 03        4375 CSCTB   LDA     1,DU            GET A NONZERO NUMBER
 END OF BINARY CARD 00000225
     013721   016463 7550 00        4376         STA     RELF            SET FLAG TO CONFORMS TO AND BECOMES
     013722   006210 7100 00        4377         TRA     A$OK            AND EXIT
     013723   016464 7200 00        4378 CONF    LXL0    GLBL            GET A UNIQUE LABEL
     013724   016464 0540 00        4379         AOS     GLBL            STEP LABEL GENERATOR
     013725   013660 7400 00        4380         STX0    L1              AND STORE IN L1
     013726   011357 2210 00        4381         LDX1    RNGE            GET POINTER TO CURRENT RANGE
```

```
              2                                          PASS 2

     013727  035221 0610 00         4382        ADX1     T$PROG            MAKE IT ABSOLUTE
     013730  000005 2350 11         4383        LDA      5,1               GET NUMBER OF COMMAS IN AU
     013731  000022 7710 00         4384        ARL      18                MOVE TO AL
     013732  000002 0750 07         4385        ADA      2,DL              GET NUMBER OF UNITS PLUS ONE IN AL
     013733  016464 7200 00         4386        LXL0     GLBL              GET STARTING LABEL OF BLOCK
     013734  016464 0550 00         4387        ASA      GLBL              INCREMENT LABEL GENERATOR TO END OF BLOCK
     013735  013653 7400 00         4388        STX0     B1X               AND STORE STARTING LABEL IN BLOCK
     013736  013654 7400 00         4389        STX0     B1S               AND SAVE START OF BLOCK
     013737  016464 7200 00         4390        LXL0     GLBL              GET STARTING LABEL OF BLOCK
     013740  016464 0550 00         4391        ASA      GLBL              INCREMENT LABEL GENERATOR TO END OF BLOCK
     013741  013656 7400 00         4392        STX0     B2X               AND STORE STARTING LABEL IN BLOCK
     013742  013657 7400 00         4393        STX0     B2S               AND SAVE START OF BLOCK
     013743  013660 2350 00         4394        LDA      L1                GET A LABEL
     013744  000022 7710 00         4395        ARL      18                PUT IT IN AL
     013745  016507 0750 03         4396        ADA      O$JUMP,DU         ADD UNCONDITIONAL JUMP CODE
     013746  012614 7000 00         4397        TSX0     CAD               AND ADD TO OUTPUT CODE
   END OF BINARY CARD 00000226
     013747  017020 6350 00         4398        EAA      O$MREF            GET MAKE REFERENCE VALUE BLOCK COMMAND
     013750  012614 7000 00         4399        TSX0     CAD               AND ADD TO OUTPUT CODE
     013751  006210 7100 00         4400        TRA      A$OK              AND EXIT
     013752  035234 2210 00         4401 CONE   LDX1     A$WORK            GET POINTER TO END OF WORKING STACK
     013753  777777 1610 11         4402        SBX1     -1,1              GET POINTER TO LAST CONTROL BLOCK IN XR - 1
     013754  035234 1610 00         4403        SBX1     A$WORK            MAKE CURRENT VALUE POINTER RELATIVE
     013755  016446 7410 00         4404        STX1     VALP              AND STORE FOR INSERTION ROUTINE
     013756  015101 7000 00         4405        TSX0     FIRM              MAKE RIGHT TERTIARY FIRM
     013757  777773 6210 00         4406        EAX1     -5                GET POINTER TO LAST CONTROL BLOCK IN WORK
     013760  016446 7410 00         4407        STX1     VALP              AND STORE FOR INSERTION ROUTINE
     013761  013655 6350 20         4408        EAA      B2,*              GET NEXT ADDRESS IN BLOCK TWO
     013762  000022 7710 00         4409        ARL      18                MOVE LABEL TO AL
     013763  016507 0750 03         4410        ADA      O$JUMP,DU         ADD UNCONDITIONAL JUMP CODE
     013764  012403 7000 00         4411        TSX0     INS               INSERT CODE IN FRONT OF VALUE
     013765  013660 2350 00         4412        LDA      L1                GET LABEL SAVED IN L1
     013766  000022 7710 00         4413        ARL      18                MOVE LABEL TO AL
     013767  016504 0750 03         4414        ADA      O$LBL,DU          ADD DEFINE LABEL CODE
     013770  012403 7000 00         4415        TSX0     INS               INSERT CODE IN FRONT OF VALUE
     013771  017026 6350 00         4416        EAA      O$MAX             GET SET STACK POINTER TO MAXIMUM COMMAND
     013772  012403 7000 00         4417        TSX0     INS               AND INSERT IN FRONT OF VALUE
     013773  017023 6350 00         4418        EAA      O$CONE            GET CONFORMITY CLEANUP COMMAND IN A
     013774  012614 7000 00         4419        TSX0     CAD               AND ADD AFTER VALUE
   END OF BINARY CARD 00000227
     013775  013654 2350 00         4420        LDA      B1S               GET START OF BLOCK ONE IN A
     013776  000022 7710 00         4421        ARL      18                MOVE LABEL TO AL
     013777  016507 0750 03         4422        ADA      O$JUMP,DU         ADD UNCONDITIONAL JUMP CODE
     014000  012614 7000 00         4423        TSX0     CAD               AND ADD TO OUTPUT CODE
     014001  013657 2200 00         4424        LDX0     B2S               GET START OF BLOCK TWO
     014002  013656 7400 00         4425        STX0     B2X               AND RESTART BLOCK TWO
     014003  035234 2210 00         4426        LDX1     A$WORK            GET POINTER TO END OF WORKING STACK
     014004  000005 1610 03         4427        SBX1     5,DU              GET POINTER TO LAST VALUE CONTROL BLOCK IN WORK
     014005  000002 2350 07         4428        LDA      2,DL              GET NUMBER OF WORDS ADDED TO CODE AFTER VALUE
     014006  000002 0550 11         4429        ASA      2,1               AND ADD TO WORD COUNT IN VALUE CONTROL BLOCK
```

```
         2                                            PASS 2

    014007  006210 7100 00       4430         TRA    A$OK           AND EXIT
    014010  035234 2210 00       4431 DIF     LDX1   A$WORK         GET POINTER TO END OF WORKING STACK
    014011  777777 1610 11       4432         SBX1   -1,1           GET POINTER TO LAST CONTROL BLOCK IN XR - 1
    014012  035234 1610 00       4433         SBX1   A$WORK         MAKE POINTER RELATIVE
    014013  016446 7410 00       4434         STX1   VALP           AND STORE AS POINTER TO CURRENT VALUE
    014014  011357 7200 00       4435         LXL0   RNGE           GET POINTER TO HIGHEST RANGE NUMBER
    014015  035221 0600 00       4436         ADX0   T$PROG         MAKE POINTER ABSOLUTE
    014016  777777 0600 10       4437         ADX0   -1,0           STEP POINTER TO NEXT RANGE
    014017  000001 0600 03       4438         ADX0   1,DU           STEP OVER LINK WORD
    014020  000005 2210 10       4439         LDX1   5,0            GET NUMBER OF COMMAS IN FOLLOWING RANGE
    014021  014064 6000 00       4440         TZE    DIF1           TRANSFER IF NO COMMAS - MUST BE IF STATEMENT
    014022  000003 2210 10       4441         LDX1   3,0            GET NUMBER OF SEMICOLONS IN FOLLOWING RANGE
  END OF BINARY CARD 00000228
    014023  014064 6010 00       4442         TNZ    DIF1           TRANSFER IF SEMICOLONS - MUST BE IF STATEMENT
    014024  035221 1600 00       4443         SBX0   T$PROG         MAKE XR - 0 RELATIVE
    014025  014031 7400 00       4444         STX0   DIFX0          SAVE XR - 0
    014026  000007 6350 00       4445         EAA    M$INT          GET AN INTEGRAL MODE IN AU
    014027  016460 7550 00       4446         STA    DECLR          AND STORE AS TARGET MODE
    014030  015072 7000 00       4447         TSX0   STRNG          COERCE CONDITION OF CASE TO INTEGRAL MODE
    014031  000000 2200 03       4448 DIFX0   LDX0   **,DU          RESTORE XR - 0
    014032  035221 0600 00       4449         ADX0   T$PROG         AND MAKE IT ABSOLUTE
    014033  000005 2350 10       4450         LDA    5,0            GET NUMBER OF COMMAS IN FOLLOWING RANGE
    014034  000022 7710 00       4451         ARL    18             PUT NUMBER OF COMMAS IN AL
    014035  000002 0750 07       4452         ADA    2,DL           ADD 2 TO GET NUMBER OF TRANSFERS IN CASE
    014036  016464 7200 00       4453         LXL0   GLBL           GET STARTING LABEL OF A UNIQUE BLOCK
    014037  016464 0550 00       4454         ASA    GLBL           STEP LABEL GENERATOR TO END OF BLOCK
    014040  013656 7400 00       4455         STX0   B2X            INITIALIZE LOCAL LABEL GENERATOR
    014041  013657 7400 00       4456         STX0   B2S            SAVE STARTING LOCATION OF LOCAL LABEL GENERATOR
    014042  000001 1750 07       4457         SBA    1,DL           GET MAXIMUM VALID INDEX IN CASE STATEMENT IN AL
    014043  000000 6270 05       4458         EAX7   0,AL           AND SAVE IN XR - 7
    014044  016550 0750 03       4459         ADA    O$CASE,DU      ADD CASE BRANCH COMMAND
    014045  012614 7000 00       4460         TSX0   CAD            AND ADD TO END OF OUTPUT CODE
    014046  016464 2350 00       4461         LDA    GLBL           GET LABEL JUST BEYOND END OF LABEL BLOCK
    014047  777777 3750 07       4462         ANA    -1,DL          CLEAN UP LABEL
    014050  000001 1750 07       4463         SBA    1,DL           GET LAST LABEL IN BLOCK IN AL
  END OF BINARY CARD 00000229
    014051  016507 0750 03       4464         ADA    O$JUMP,DU      ADD UNCONDITIONAL JUMP CODE
    014052  012614 7000 00       4465         TSX0   CAD            AND ADD TO END OF OUTPUT CODE
    014053  013655 6350 20       4466 DIFL    EAA    B2,*           GET NEXT LABEL IN LOCAL BLOCK IN AU
    014054  000022 7710 00       4467         ARL    18             MOVE LABEL TO AL
    014055  016507 0750 03       4468         ADA    O$JUMP,DU      ADD UNCONDITIONAL JUMP CODE
    014056  012614 7000 00       4469         TSX0   CAD            AND ADD TO END OF OUTPUT CODE
    014057  000001 1670 03       4470         SBX7   1,DU           DECREMENT NUMBER OF LABELS LEFT TO ADD
    014060  014053 6010 00       4471         TNZ    DIFL           TRANSFER IF MORE LABELS TO ADD
    014061  013657 2200 00       4472         LDX0   B2S            GET STARTING LABEL IN BLOCK
    014062  013656 7400 00       4473         STX0   B2X            AND REINITIALIZE LOCAL LABEL GENERATOR
    014063  014100 7100 00       4474         TRA    DELV           GO DELETE CONDITION CONTROL BLOCK FROM WORK
    014064  000003 6350 00       4475 DIF1    EAA    M$BOOL         GET A BOOLEAN MODE IN AU
    014065  016460 7550 00       4476   -     STA    DECLR          AND STORE AS TARGET MODE
    014066  015072 7000 00       4477         TSX0   STRNG          COERCE CONDITION TO BOOLEAN MODE
```

```
     014067  016464 7200 00        4478           LXL0    GLBL             GET A UNIQUE LABEL
     014070  016464 0540 00        4479           AOS     GLBL             STEP LABEL GENERATOR TO ANOTHER LABEL
     014071  013656 7400 00        4480           STX0    B2X              INITIALIZE LOCAL LABEL GENERATOR TO UNIQUE LABEL
     014072  013657 7400 00        4481           STX0    B2S              AND STORE LABEL AS START OF BLOCK
     014073  000000 6350 10        4482           EAA     0,0              GET LABEL IN AU
     014074  000022 7710 00        4483           ARL     18               MOVE LABEL TO AL
     014075  016545 0750 03        4484           ADA     O$TF,DU          ADD TRANSFER IF FALSE CODE
     014076  012614 7000 00        4485           TSX0    CAD              AND ADD TO END OF OUTPUT CODE
   END OF BINARY CARD 00000230
     014077  014100 7100 00        4486           TRA     DELV             AND DELETE CONTROL BLOCK FOR CONDITION VALUE
     014100  000000 2220 03        4487 DELV      LDX2    0,DU             INITIALIZE COUNT TO ZERO
     014101  035234 2210 00        4488           LDX1    A$WORK           GET POINTER TO END OF WORKING STACK
     014102  000000 2350 07        4489           LDA     0,DL             INITIALIZE LL WORD
     014103  777777 1610 11        4490 DELV1     SBX1    -1,1             STEP BACK ONE CONTROL BLOCK
     014104  000003 2750 11        4491           ORA     3,1              OR IN LL OF THIS VALUE
     014105  000000 7200 11        4492           LXL0    0,1              GET NUMBER OF ADDITIONAL BLOCKS IN THIS VALUE
     014106  014116 7400 00        4493           STX0    DELVT            AND STORE
     014107  014116 0620 00        4494           ADX2    DELVT            INCREMENT NUMBER OF BLOCKS TO STEP
     014110  000001 1620 03        4495           SBX2    1,DU             DECREMENT NUMBER JUST STEPPED
     014111  014103 6050 00        4496           TPL     DELV1            TRANSFER IF MORE BLOCKS TO STEP
     014112  035234 0110 54        4497 DELV2     NOP     A$WORK,DI        DELETE A WORD FROM THE WORKING STACK
     014113  035234 1010 00        4498           CMPX1   A$WORK           ARE ALL BLOCKS DELETED
     014114  014112 6010 00        4499           TNZ     DELV2            TRANSFER IF MORE TO DELETE
     014115  006210 7100 00        4500           TRA     A$OK             AND EXIT
     014116  000000 000000         4501 DELVT     ZERO
     014117  014136 2210 00        4502 OIDNY     LDX1    OPNT             GET POINTER TO LAST OPERATOR DEFINITION
     014120  035220 0610 00        4503           ADX1    T$DEF            MAKE POINTER ABSOLUTE
     014121  777777 0610 11        4504 OIDY1     ADX1    -1,1             STEP TO NEXT DEFINITION TABLE ENTRY
     014122  000001 0610 03        4505           ADX1    1,DU             STEP OVER LINK WORD
     014123  000000 7200 11        4506           LXL0    0,1              GET LL OF NEW DEFINITION
     014124  014121 6000 00        4507           TZE     OIDY1            SKIP ENTRY IF LL0 BY TRANSFERING
   END OF BINARY CARD 00000231
     014125  000001 2200 11        4508           LDX0    1,1              GET TYPE OF DEFINITION IN XR - 0
     014126  006413 1000 03        4509           CMPX0   D$OP,DU          IS IT AN OPERATOR DEFINITION ENTRY
     014127  014121 6010 00        4510           TNZ     OIDY1            TRANSFER IF NOT
     014130  000002 2350 11        4511           LDA     2,1              GET MODE OF OPERATOR
     014131  013257 7550 00        4512           STA     FMODE            AND STORE IN FMODE FOR DECLARATION ROUTINES
     014132  035220 1610 00        4513           SBX1    T$DEF            MAKE POINTER RELATIVE
     014133  014136 7410 00        4514           STX1    OPNT             AND STORE AS CURRENT OPERATOR DEFINITION
     014134  011731 7410 00        4515           STX1    LASTS            AND STORE AS DEFINITION OF LAST SYMBOL READ
     014135  006210 7100 00        4516           TRA     A$OK             AND EXIT
     014136  000001 000000         4517 OPNT      ZERO    1
     014137  014202 7470 00        4518 FOP       STX7    COPER            SAVE POINTER TO OPERATOR SYMBOL
     014140  015067 7470 00        4519           STX7    COPET            SAVE OPERATOR POINTER FOR POSSIBLE ERROR MESSAGE
     014141  011357 2200 00        4520           LDX0    RNGE             GET CURRENT RANGE NUMBER
     014142  006460 7400 00        4521           STX0    A$LEVEL          AND STORE FOR TABLE LOOK UP ROUTINE
     014143  000001 6210 17        4522           EAX1    A$DF,7           GET SYMBOL POINTER IN XR - 1
     014144  035222 0610 00        4523           ADX1    T$STAB           MAKE IT ABSOLUTELY POINT TO DEFINITION CHAIN
     014145  006437 7000 00        4524 FOP1      TSX0    A$TLU            LOOK UP OPERATOR IN SYMBOL TABLE
     014146  015036 7100 00        4525           TRA     COPEP            TRANSFER IF CANNOT IDENTIFY OPERATOR PRIORITY
```

```
            2                                                PASS 2

      014147  000001 2220 11         4526          LDX2     1,1               GET TYPE OF DEFINITION IN XR - 2
      014150  006414 1020 03         4527          CMPX2    D$PRIOR,DU        IS IT A PRIORITY DEFINITION
      014151  014145 6010 00         4528          TNZ      FOP1              TRANSFER IF NO - KEEP SEARCHING
      014152  000002 2220 11         4529          LDX2     2,1               GET PRIORITY OF OPERATOR IN XR - 2
   END OF BINARY CARD 00000232
      014153  016474 7420 00         4530          STX2     CPR               AND STORE PRIORITY AS CURRENT PRIORITY
      014154  014621 7500 00         4531          STC2     COPX              SAVE RETURN
      014155  014620 7100 00         4532          TRA      COP               FORCE HIGHER PRIORITY OPERATORS
      014156  035234 6200 56         4533          EAX0     A$WORK,ID         GET A WORD IN THE WORKING STACK
      014157  005754 7170 00         4534          XED      T$WOVF            CHECK FOR STACK OVERFLOW
      014160  004000 6350 00         4535          EAA      W$OP              GET HEADER WORD FOR OPERATOR BLOCK
      014161  000000 7550 10         4536          STA      0,0               AND STORE IN WORKING STACK
      014162  016475 2220 00         4537          LDX2     OPT               GET POINTER TO LAST OPERATOR BLOCK
      014163  000000 4420 10         4538          SXL2     0,0               AND STORE IN LOWER HALF OF HEADER
      014164  035214 1600 00         4539          SBX0     T$WORK            GET RELATIVE POINTER TO THIS BLOCK
      014165  016475 7400 00         4540          STX0     OPT               AND STORE NEW LINK TO OPERATOR BLOCK
      014166  014202 2350 00         4541          LDA      COPER             GET POINTER TO OPERATOR SYMBOL
      014167  035234 7550 56         4542          STA      A$WORK,ID         AND STORE IN WORKING STACK
      014170  005754 7170 00         4543          XED      T$WOVF            CHECK FOR STACK OVERFLOW
      014171  016474 2350 00         4544          LDA      CPR               GET PRIORITY OF CURRENT OPERATOR
      014172  035234 7550 56         4545          STA      A$WORK,ID         STORE IN WORKING STACK
      014173  005754 7170 00         4546          XED      T$WOVF            CHECK FOR STACK OVERFLOW
      014174  035234 4500 56         4547          STZ      A$WORK,ID         STORE A ZERO IN THE WORKING STACK
      014175  005754 7170 00         4548          XED      T$WOVF            CHECK FOR STACK OVERFLOW
      014176  000005 6350 00         4549          EAA      5                 GET NUMBER OF WORDS IN THIS CONTROL BLOCK
      014177  035234 7550 56         4550          STA      A$WORK,ID         AND STORE IN WORKING STACK
      014200  005754 7170 00         4551          XED      T$WOVF            CHECK FOR STACK OVERFLOW
   END OF BINARY CARD 00000233
      014201  006210 7100 00         4552          TRA      A$OK              AND EXIT
      014202  000000 000000          4553 COPER    ZERO
      014203  016474 4500 00         4554 FOPS     STZ      CPR               SET CURRENT PRIORITY TO ZERO
      014204  014621 7500 00         4555          STC2     COPX              SAVE RETURN
      014205  014620 7100 00         4556          TRA      COP               FORCE OUT ALL OPERATORS
      014206  006210 7100 00         4557          TRA      A$OK              AND EXIT
      014207  035235 7470 56         4558 WMOP     STX7     A$STACK,ID        STORE OPERATOR SYMBOL IN CONTROL STACK
      014210  005742 7170 00         4559          XED      T$SOVF            CHECK FOR STACK OVERFLOW
      014211  006210 7100 00         4560          TRA      A$OK              AND EXIT
      014212  035235 2210 54         4561 MOP      LDX1     A$STACK,DI        GET POINTER TO OPERATOR SYMBOL IN XR - 1
      014213  014220 2350 00         4562          LDA      MOPC              GET UNARY OPERATOR FLAG
      014214  015071 7550 00         4563          STA      OPF               AND STORE FLAG
      014215  014621 7500 00         4564          STC2     COPX              SAVE RETURN
      014216  014633 7100 00         4565          TRA      COPM              DO UNARY OPERATOR
      014217  006210 7100 00         4566          TRA      A$OK              AND EXIT
      014220  000003 777777          4567 MOPC     ZERO     3,-1
      014221  014317 4500 00         4568 SELCT    STZ      SELF              RESET REFERENCE FLAG
      014222  035234 2210 00         4569          LDX1     A$WORK            GET POINTER TO END OF WORKING STACK
      014223  777777 1610 11         4570          SBX1     -1,1              GET POINTER TO LAST CONTROL BLOCK
      014224  035234 1610 00         4571          SBX1     A$WORK            MAKE POINTER RELATIVE
      014225  016446 7410 00         4572          STX1     VALP              AND STORE AS POINTER TO CURRENT VALUE
      014226  015103 7000 00         4573          TSX0     WEAK              DO WEAK COERCION ON SELECTEE
```

```
  END OF BINARY CARD 00000234
      014227  016203 2270 00         4574           LDX7     TMODE            GET MODE OF SELECTAND
      014230  010630 7000 00         4575           TSX0     A$XFER           MAKE IT UNIQUE AND ABSOLUTE
      014231  000000 2220 17         4576           LDX2     0,7              GET TYPE OF MODE IN XR - 2
      014232  016757 1020 03         4577           CMPX2    M$STRCT,DU       IS IT A STRUCTURED MODE
      014233  014245 6000 00         4578           TZE      SEL0             TRANSFER IF STRUCTURED MODE
      014234  016762 1020 03         4579           CMPX2    M$REF,DU         IS IT A REFERENCE MODE
      014235  014305 6010 00         4580           TNZ      SELE             TRANSFER IF NOT - ERROR
      014236  000001 2270 17         4581           LDX7     1,7              GET DEREFERENCED MODE IN XR - 7
      014237  010630 7000 00         4582           TSX0     A$XFER           MAKE IT UNIQUE AND ABSOLUTE
      014240  000000 2220 17         4583           LDX2     0,7              GET TYPE OF DEREFERENCED MODE IN XR - 2
      014241  016757 1020 03         4584           CMPX2    M$STRCT,DU       IS IT A STRUCTURED MODE
      014242  014305 6010 00         4585           TNZ      SELE             NO - SELECTION ERROR
      014243  000003 2350 03         4586           LDA      O$RSLCT-O$SELCT,DU GET CONSTANT TO CONVERT CODE TO REFERENCE
      014244  014317 7550 00         4587           STA      SELF             AND STORE IN REFERENCE FLAG
      014245  035235 2200 54         4588 SEL0      LDX0     A$STACK,DI       GET POINTER TO TAG SYMBOL IN XR - 0
      014246  777777 2220 17         4589           LDX2     -1,7             GET NUMBER OF FIELDS PLUS ONE IN XR - 2
      014247  000000 6230 17         4590           EAX3     0,7              GET POINTER TO FIELDS IN XR - 3
      014250  016556 6350 00         4591           EAA      O$SELCT          GET CODE FOR SELECTION IN AU
      014251  000001 1620 03         4592 SEL1      SBX2     1,DU             DECREMENT NUMBER OF REMAINING FIELDS
      014252  014305 6000 00         4593           TZE      SELE             TRANSFER IF NO MORE - ERROR
      014253  000001 0630 03         4594           ADX3     1,DU             STEP FIELD POINTER TO NEXT FIELD
      014254  000001 0750 07         4595           ADA      1,DL             STEP FIELD NUMBER IN CODE
  END OF BINARY CARD 00000235
      014255  000000 7240 13         4596           LXL4     0,3              GET TAG FOR FIELD IN XR - 4
      014256  014316 7440 00         4597           STX4     SELT             AND STORE FORE COMPARE
      014257  014316 1000 00         4598           CMPX0    SELT             SEE IF PROPER FIELD HAS BEEN FOUND
      014260  014251 6010 00         4599           TNZ      SEL1             TRANSFER IF NOT - KEEP LOOKING
      014261  035216 1630 00         4600           SBX3     T$MODE           MAKE FIELD POINTER RELATIVE
      014262  014317 0750 00         4601           ADA      SELF             CHANGE O$SELCT TO O$RSLCT IF REFERENCE SELECTION
      014263  012614 7000 00         4602           TSX0     CAD              ADD SELECT CODE TO OUTPUT CODE
      014264  035216 0630 00         4603           ADX3     T$MODE           MAKE FIELD POINTER ABSOLUTE
      014265  000000 2270 13         4604           LDX7     0,3              GET MODE OF FIELD IN XR - 7
      014266  010630 7000 00         4605           TSX0     A$XFER           MAKE MODE UNIQUE AND ABSOLUTE
      014267  035216 1670 00         4606           SBX7     T$MODE           MAKE MODE RELATIVE
      014270  014317 2340 00         4607           SZN      SELF             IS THIS A REFERENCE TYPE SELECTION
      014271  014273 6000 00         4608           TZE      SEL2             TRANSFER IF NOT
      014272  012646 7000 00         4609           TSX0     EREF1            REFERENCE MODE OF FIELD
      014273  000000 6350 17         4610 SEL2      EAA      0,7              GET MODE OF SELECTION IN AU
      014274  000022 7710 00         4611           ARL      18               PUT MODE OF FIELD IN AL
      014275  016564 0750 03         4612           ADA      O$ETC,DU         ADD ETC CODE
      014276  012614 7000 00         4613           TSX0     CAD              AND ADD TO OUTPUT CODE
      014277  035234 2210 00         4614           LDX1     A$WORK           GET POINTER TO END OF WORKING STACK
      014300  777777 1610 11         4615           SBX1     -1,1             GET POINTER TO LAST CONTROL BLOCK
      014301  000002 2350 07         4616           LDA      2,DL             GET NUMBER OF WORDS ADDED TO VALUE
      014302  000002 0550 11         4617           ASA      2,1              AND INCREMENT LENGTH OF VALUE
  END OF BINARY CARD 00000236
      014303  000001 7470 11         4618           STX7     1,1              ALSO STORE MODE OF SELECTION IN VALUE
      014304  006210 7100 00         4619           TRA      A$OK             AND EXIT
      014305  014310 2350 00         4620 SELE      LDA      SELEM            GET TALLY WORD FOR ERROR MESSAGE
```

```
     014306  006562 7000 00        4621        TSX0    1$XXX             AND TYPE ERROR MESSAGE
     014307  006241 7100 00        4622        TRA     A$ERRF            AND PRINT FOLLOWING SYMBOLS
     014310  014311 0022 40        4623 SELEM  TALLYB  **1,17*1
     014311  111114114105          4624        UASCI   5,ILLEGAL SELECTION
     014312  107101114040
     014313  123105114105
     014314  103124111117
     014315  116040040040
     014316  000000 000000         4625 SELT   ZERO
     014317  000000 000000         4626 SELF   ZERO
     014320  035235 2210 54        4627 CALL   LDX1    A$STACK,DI        GET MODE OF PROCEDURE
     014321  016460 7410 00        4628        STX1    DECLR             STORE AS TARGET MODE FOR ACTUAL PARAMETERS
     014322  035234 2210 00        4629        LDX1    A$WORK            GET POINTER TO END OF WORKING STACK
     014323  777777 1610 11        4630        SBX1    -1,1              GET POINTER TO LAST CONTROL BLOCK IN XR - 1
     014324  035234 1610 00        4631        SBX1    A$WORK            MAKE POINTER RELATIVE
     014325  016446 7410 00        4632        STX1    VALP              AND STORE POINTER AS CURRENT VALUE
     014326  015072 7000 00        4633        TSX0    STRNG             COERCE ACTUAL PARAMETERS TO FORMAL PARAMETERS
     014327  035234 2210 00        4634        LDX1    A$WORK            GET POINTER TO END OF WORKING STACK
     014330  777777 1610 11        4635        SBX1    -1,1              GET POINTEE TO LAST CONTROL BLOCK
  END OF BINARY CARD 00000237
     014331  035234 1610 00        4636        SBX1    A$WORK            MAKE POINTER RELATIVE
     014332  016446 7410 00        4637        STX1    VALP              AND STORE IN CURRENT VALUE CONTROL BLOCK POINTER
     014333  016611 6350 00        4638        EAA     O$MSCW            GET MARK STACK COMMAND
     014334  012403 7000 00        4639        TSX0    INS               AND INSERT IN FRONT OF ARGUMENT CODE
     014335  016446 2210 00        4640        LDX1    VALP              GET POINTER TO ARGUMENT CODE CONTROL BLOCK
     014336  035234 0610 00        4641        ADX1    A$WORK            MAKE POINTER ABSOLUTE
     014337  000000 6240 11        4642        EAX4    0,1               SAVE POINTER TO VALUE CONTROL BLOCK
     014340  777777 1610 11        4643        SBX1    -1,1              GET POINTER TO SECOND TO LAST CONTROL BLOCK
     014341  000001 2270 11        4644        LDX7    1,1               GET MODE OF PROCEDURE IN XR - 7
     014342  010630 7000 00        4645        TSX0    A$XFER            MAKE MODE UNIQUE AND ABSOLUTE
     014343  000000 2230 17        4646        LDX3    0,7               GET TYPE OF MODE IN XR - 3
     014344  017001 1030 03        4647        CMPX3   M$PROC,DU         SEE IF IT IS A PROCEDURE MODE
     014345  777777 6010 00        4648        TNZ     $ERROR            ERROR - PROCEDURE CALL MUST BE ON A PROCEDURE
     014346  777777 0670 17        4649        ADX7    -1,7              GET POINTER TO END OF PROCEDURE MODE PLUS ONE
     014347  777777 2350 17        4650        LDA     -1,7              GET MODE OF RESULT OF PROCEDURE IN A
     014350  000001 7550 14        4651        STA     1,4               STORE MODE OF RESULT IN VALUE CONTROL BLOCK
     014351  000022 7710 00        4652        ARL     18                PUT RESULT MODE IN AL
     014352  016520 0750 03        4653        ADA     O$ENTER,DU        GET AN ENTER PROCEDURE CODE
     014353  012614 7000 00        4654        TSX0    CAD               AND ADD TO OUTPUT CODE AFTER ARGUMENT CODE
     014354  016446 2210 00        4655        LDX1    VALP              GET POINTER TO ARGUMENT CODE CONTROL BLOCK
     014355  035234 0610 00        4656        ADX1    A$WORK            MAKE POINTER ABSOLUTE
     014356  000001 2350 07        4657        LDA     1,DL              GET A 1 IN AL
  END OF BINARY CARD 00000238
     014357  000002 0550 11        4658        ASA     2,1               INCREMENT LENGTH FIELD IN LAST CONTROL BLOCK
     014360  000000 0550 11        4659        ASA     0,1               PUT A ONE IN NUMBER OF KEYED CONTROL BLOCKS
     014361  012530 7000 00        4660        TSX0    DELW              COMBINE LAST TWO CONTROL BLOCKS
     014362  006210 7100 00        4661        TRA     A$OK              AND EXIT
     014363  035234 2200 00        4662 MARK   LDX0    A$WORK            GET POINTER TO CURRENT END OF WORKING STACK
     014364  035214 1600 00        4663        SBX0    T$WORK            MAKE IT RELATIVE
     014365  035235 7400 56        4664        STX0    A$STACK,ID        AND STORE MARK IN CONTROL STACK
```

```
      014366  005742 7170 00       4665         XED     T$SOVF          CHECK FOR STACK OVERFLOW
      014367  035234 4500 56       4666         STZ     A$WORK,ID       INSERT SPACE FOR HEADER WORD IN WORKING STACK
      014370  005754 7170 00       4667         XED     T$WOVF          CHECK FOR STACK OVERFLOW
      014371  006210 7100 00       4668         TRA     A$OK            AND EXIT
      014372  200000 6360 00       4669 ESKIP   EAQ     W$SKIP          GET HEADER FOR SKIP CONTROL BLOCK
      014373  016567 6350 00       4670         EAA     O$SKIP          GET CODE FOR SKIP
      014374  016465 4500 00       4671         STZ     LL              ZERO OUT LL FOR SKIP
      014375  012572 7000 00       4672         TSX0    WAD             ENTER CODE AND CONTROL BLOCK FOR SKIP
      014376  006210 7100 00       4673         TRA     A$OK            AND EXIT
      014377  100000 6360 00       4674 ENIL    EAQ     W$NIL           GET HEADER FOR NIL CONTROL BLOCK
      014400  016572 6350 00       4675         EAA     O$NIL           GET CODE FOR NIL
      014401  016465 4500 00       4676         STZ     LL              ZERO OUT LL FOR NIL
      014402  012572 7000 00       4677         TSX0    WAD             ENTER CODE AND CONTROL BLOCK FOR NIL
      014403  006210 7100 00       4678         TRA     A$OK            AND EXIT
      014404  016606 6350 00       4679 EFALS   EAA     O$FALSE         GET FALSE CODE WORD
   END OF BINARY CARD 00000239
      014405  014407 7100 00       4680         TRA     ETR1            AND CONTINUE
      014406  016603 6350 00       4681 ETRUE   EAA     O$TRUE          GET TRUE CODE WORD
      014407  400000 6360 00       4682 ETR1    EAQ     W$VALUE         GET HEADER FOR CONTROL BLOCK IN Q
      014410  016465 4500 00       4683         STZ     LL              ZERO OUT LL FOR BOOLEAN DENOTATION
      014411  012572 7000 00       4684         TSX0    WAD             ENTER CODE AND CONTROL BLOCK FOR BOOLEAN VALUE
      014412  035234 2210 00       4685         LDX1    A$WORK          GET POINTER TO END OF WORKING STACK
      014413  000005 1610 03       4686         SBX1    5,DU            GET POINTER TO LAST CONTROL BLOCK
      014414  000003 6350 00       4687         EAA     M$BOOL          GET MODE OF BOOLEAN DENOTATION
      014415  000001 7550 11       4688         STA     1,1             AND STORE IN CONTROL BLOCK
      014416  006210 7100 00       4689         TRA     A$OK            AND EXIT
      014417  011731 2210 00       4690 SDCLR   LDX1    LASTS           GET POINTER TO DEFINITION OF LAST DECLARER
      014420  035220 0610 00       4691         ADX1    T$DEF           MAKE IT ABSOLUTE
      014421  000002 2350 11       4692         LDA     2,1             GET DECLARER IN A REGISTER
      014422  016460 7550 00       4693         STA     DECLR           AND STORE AS CURRENT DECLARER
      014423  006210 7100 00       4694         TRA     A$OK            AND EXIT
      014424  011731 2350 00       4695 OIDN    LDA     LASTS           GET POINTER TO LAST IDENTIFIER DEFINITION
      014425  000022 7710 00       4696         ARL     18              MOVE TO AL
      014426  016614 0750 03       4697         ADA     O$IDNTY,DU      ADD IDENTITY DECLARATION CODE
      014427  012614 7000 00       4698         TSX0    CAD             AND ADD TO OUTPUT CODE
      014430  011731 2210 00       4699         LDX1    LASTS           GET POINTER TO LAST IDENTIFIER
      014431  035220 0610 00       4700         ADX1    T$DEF           MAKE DEFINITION POINTER ABSOLUTE
      014432  011651 7000 00       4701         TSX0    LK              LINK DEFINITION TO CHAIN FOR CURRENT LL
   END OF BINARY CARD 00000240
      014433  011731 2210 00       4702 OIDN2   LDX1    LASTS           GET POINTER TO LAST IDENTIFIER
      014434  035220 0610 00       4703         ADX1    T$DEF           MAKE DEFINITION POINTER ABSOLUTE
      014435  000000 7200 11       4704         LXL0    0,1             GET LEVEL OF IDENTIFIER IN XR = 0
      014436  006210 6010 00       4705         TNZ     A$OK            TRANSFER IF NOT ZERO
      014437  035232 2210 03       4706         LDX1    T$SDEF,DU       GET POINTER TO SYMDEF TABLE CONTROL WORD
      014440  000000 6350 00       4707         EAA     1-1             PREPARE TO ALLOCATE ONE WORD
      014441  005663 7000 00       4708         TSX0    T$ALOC          ALLOCATE ONE WORD IN SYMDEF TABLE
      014442  011731 2350 00       4709         LDA     LASTS           GET POINTER TO DEFINITION IN AU
      014443  777777 7550 11       4710         STA     -1,1            AND STORE IN SYMDEF TABLE
      014444  006210 7100 00       4711         TRA     A$OK            AND EXIT
      014445  016460 2350 00       4712 OENTR   LDA     DECLR           GET MODE OF CURRENT DECLARER
```

```
          2                                                 PASS 2

     014446  000022 7710 00        4713         ARL     18              MOVE TO AL
     014447  016617 0750 03        4714         ADA     O$IDNTE,DU      ADD DO IDENTITY DECLARATION CODE
     014450  012614 7000 00        4715         TSX0    CAD             AND ADD TO OUTPUT CODE
     014451  006210 7100 00        4716         TRA     A$OK            AND EXIT
     014452  016460 2350 00        4717 OASGN   LDA     DECLR           GET MODE OF CURRENT DECLARER
     014453  000022 7710 00        4718         ARL     18              MOVE TO AL
     014454  016633 0750 03        4719         ADA     O$ASGN,DU       ADD ASSIGN OPERATOR CODE
     014455  012614 7000 00        4720         TSX0    CAD             AND ADD TO OUTPUT CODE
     014456  006210 7100 00        4721         TRA     A$OK            AND EXIT
     014457  011731 2350 00        4722 OFORM   LDA     LASTS           GET LAST SYMBOL OR FORMAL PARAMETER
     014460  000022 7710 00        4723         ARL     18              GET DEF TABLE POINTER IN AL
   END OF BINARY CARD 00000241
     014461  016622 0750 03        4724         ADA     O$FORMP,DU      GET FORMAL PARAMETER CODE
     014462  012614 7000 00        4725         TSX0    CAD             ADD TO OUTPUT CODE
     014463  006210 7100 00        4726         TRA     A$OK            AND EXIT
     014464  006462 7000 00        4727 WIDEN   TSX0    A$GLL           GET LL OF IDENTIFIER
     014465  016465 7550 00        4728         STA     LL              SAVE FOR VALUE CONTROL BLOCK INSERTION ROUTINE
     014466  013573 2550 00        4729         ORSA    RLL             OR INTO RANGE LL WORD
     014467  011731 2350 00        4730         LDA     LASTS           GET POINTER TO DEFINITION OF IDENTIFIER
     014470  000022 7710 00        4731         ARL     18              PUT IT IN AL
     014471  016625 0750 03        4732         ADA     O$IDENT,DU      ADD IDENTIFIER CODE
     014472  400000 6360 00        4733         EAQ     W$VALUE         GET HEADER FOR CONTROL BLOCK IN Q
     014473  012572 7000 00        4734         TSX0    WAD             ADD CONTROL BLOCK AND OUTPUT CODE
     014474  035234 2210 00        4735         LDX1    A$WORK          GET POINTER TO END OF WORKING STACK
     014475  777777 1610 11        4736         SBX1    -1,1            GET POINTER TO LAST CONTROL BLOCK
     014476  011731 2220 00        4737         LDX2    LASTS           GET POINTER TO IDENTIFIER DEFINITION
     014477  035220 0620 00        4738         ADX2    T$DEF           MAKE IT ABSOLUTE
     014500  000002 2350 12        4739         LDA     2,2             GET DECLARER FOR IDENTIFIER
     014501  000001 7550 11        4740         STA     1,1             AND STORE IN CONTROL BLOCK FOR IDENTIFIER
     014502  006210 7100 00        4741         TRA     A$OK            AND EXIT
     014503  000000 6210 17        4742 WDEN    EAX1    0,7             GET POINTER TO STAB IN XR - 1
     014504  035222 0610 00        4743         ADX1    T$STAB          MAKE POINTER ABSOLUTE
     014505  000001 2350 11        4744         LDA     A$DF,1          GET POINTER TO DENOTATION DEFINITION IN AU
     014506  000022 7710 00        4745         ARL     18              PUT IT IN AL
   END OF BINARY CARD 00000242
     014507  016630 0750 03        4746         ADA     O$DENOT,DU      GET DENOTATION CODE
     014510  400000 6360 00        4747         EAQ     W$VALUE         GET HEADER FOR CONTROL BLOCK
     014511  016465 4500 00        4748         STZ     LL              ZERO OUT LL FOR DENOTATION
     014512  012572 7000 00        4749         TSX0    WAD             AND ADD CONTROL BLOCK AND OUTPUT CODE
     014513  000001 6210 17        4750         EAX1    A$DF,7          GET POINTER TO DEFINITION CHAIN IN XR - 1
     014514  035222 0610 00        4751         ADX1    T$STAB          MAKE IT ABSOLUTE
     014515  000000 2220 11        4752         LDX2    0,1             GET POINTER TO DEFINITION IN XR - 2
     014516  777777 6040 00        4753         TMI     $ERROR          NO DEFINITION - COMPILER ERROR
     014517  035220 0620 00        4754         ADX2    T$DEF           MAKE IT ABSOLUTE
     014520  000001 2230 12        4755         LDX3    1,2             GET TYPE OF DEFINITION IN XR - 3
     014521  006416 1030 03        4756         CMPX3   D$DENOT,DU      IS IT A DENOTATION
     014522  777777 6010 00        4757         TNZ     $ERROR          NO - COMPILER ERROR
     014523  035234 2210 00        4758         LDX1    A$WORK          GET POINTER TO END OF WORKING STACK IN XR - 1
     014524  777777 1610 11        4759         SBX1    -1,1            GET POINTER TO LAST CONTROL BLOCK
     014525  000002 2350 12        4760         LDA     2,2             GET DECLARER FOR DENOTATION
```

```
      014526  000001 7550 11      4761         STA     1,1            STORE IN DENOTATION CONTROL BLOCK
      014527  006210 7100 00      4762         TRA     A$OK           AND EXIT
      014530  014612 7400 00      4763 PCDR    STX0    PCDRX          SAVE RETURN
      014531  014616 4470 00      4764         SXL7    PCDL4          STORE PROCEDURE MODE IN CODE SEQUENCE
      014532  010630 7000 00      4765         TSX0    A$XFER         MAKE MODE POINTER ABSOLUTE
      014533  777777 0670 17      4766         ADX7    -1,7           GET POINTER TO END OF PROCEDURE MODE
      014534  777777 2270 17      4767         LDX7    -1,7           GET MODE OF RESULT OF PROCEDURE
   END OF BINARY CARD 00000243
      014535  014613 4470 00      4768         SXL7    PCDL1          AND STORE IN CODE SEQUENCE
      014536  016464 2350 00      4769         LDA     GLBL           GET A NEW LABEL IN AL
      014537  016464 0540 00      4770         AOS     GLBL           STEP LABEL GENERATOR TO NEXT LABEL
      014540  016507 0750 03      4771         ADA     O$JUMP,DU      ADD UNCONDITIONAL JUMP CODE
      014541  012403 7000 00      4772         TSX0    INS            AND INSERT IN FRONT OF PROCEDURED VALUE
      014542  016464 2350 00      4773         LDA     GLBL           GET A NEW LABEL IN AL
      014543  016464 0540 00      4774         AOS     GLBL           STEP LABEL GENERATOR TO NEXT LABEL
      014544  016660 0750 03      4775         ADA     O$EPDN,DU      ADD ENTER PROCEDURE CODE
      014545  012403 7000 00      4776         TSX0    INS            AND INSERT IN FRONT OF PROCEDURED VALUE
      014546  016446 2210 00      4777         LDX1    VALP           GET POINTER TO CURRENT CONTROL BLOCK
      014547  035234 0610 00      4778         ADX1    A$WORK         MAKE POINTER ABSOLUTE
      014550  011357 2220 00      4779         LDX2    RNGE           GET POINTER TO CURRENT RANGE
      014551  014614 4420 00      4780         SXL2    PCDL2          STORE RANGE NUMBER IN CODE SEQUENCE
      014552  014562 6000 00      4781         TZE     PCDR2          TRANSFER IF GLOBAL
      014553  000003 2350 11      4782         LDA     3,1            GET LL WORD FOR PROCEDURED VALUE
      014554  014562 6040 00      4783 PCDR1   TMI     PCDR2          TRANSFER IF SCOPE OF VALUE IS CURRENT RANGE
      014555  035221 0620 00      4784         ADX2    T$PROG         MAKE RANGE POINTER ABSOLUTE
      014556  000002 2220 12      4785         LDX2    2,2            GET POINTER FOR SURROUNDING RANGE
      014557  014562 6000 00      4786         TZE     PCDR2          TRANSFER IF NO SURROUNDING RANGE
      014560  000001 7350 00      4787         ALS     1              CHECK IF SCOPE IS NEXT OUTER RANGE
      014561  014554 7100 00      4788         TRA     PCDR1          TRANSFER TO SCOPE CHECK
      014562  000000 6350 12      4789 PCDR2   EAA     0,2            GET RANGE POINTER IN AU
   END OF BINARY CARD 00000244
      014563  000022 7710 00      4790         ARL     18             MOVE TO AL
      014564  016641 0750 03      4791         ADA     O$LL,DU        ADD LL CODE
      014565  012403 7000 00      4792         TSX0    INS            AND INSERT IN FRONT OF PROCEDURED VALUE
      014566  016464 7200 00      4793         LXL0    GLBL           GET CURRENT VALUE OF LABEL GENERATOR
      014567  000002 1600 03      4794         SBX0    2,DU           GET VALUE OF FIRST GENERATED LABEL
      014570  014615 4400 00      4795         SXL0    PCDL3          AND STORE IN CODE SEQUENCE
      014571  000001 0600 03      4796         ADX0    1,DU           GET VALUE OF SECOND GENERATED LABEL
      014572  014617 4400 00      4797         SXL0    PCDL5          AND STORE IN CODE SEQUENCE
      014573  014617 2350 00      4798         LDA     PCDL5          GET NEXT CODE WORD
      014574  012375 7000 00      4799         TSX0    ADD            AND ADD TO CODE SEQUENCE
      014575  014616 2350 00      4800         LDA     PCDL4          GET NEXT CODE WORD
      014576  012375 7000 00      4801         TSX0    ADD            AND ADD TO CODE SEQUENCE
      014577  014615 2350 00      4802         LDA     PCDL3          GET NEXT CODE WORD
      014600  012375 7000 00      4803         TSX0    ADD            AND ADD TO CODE SEQUENCE
      014601  014614 2350 00      4804         LDA     PCDL2          GET NEXT CODE WORD
      014602  012375 7000 00      4805         TSX0    ADD            AND ADD TO CODE SEQUENCE
      014603  014613 2350 00      4806         LDA     PCDL1          GET NEXT CODE WORD
      014604  012375 7000 00      4807         TSX0    ADD            AND ADD TO CODE SEQUENCE
      014605  016446 2210 00      4808         LDX1    VALP           GET POINTER TO CURRENT CONTROL BLOCK
```

```
         2                                                 PASS 2

     014606  035234 0610 00        4809        ADX1     A$WORK            MAKE POINTER ABSOLUTE
     014607  000003 3350 03        4810        LCA      3,DU              GET A -3 IN AU
     014610  000010 0750 07        4811        ADA      8,DL              AND ADD 8 TO LENGTH
  END OF BINARY CARD 00000245
     014611  000002 0550 11        4812        ASA      2,1               UPDATE LOC/LEN WORD
     014612  000000 7100 00        4813 PCDRX  TRA      **                AND RETURN
     014613  016663 000000         4814 PCDL1  ZERO     O$RETN,**
     014614  016644 000000         4815 PCDL2  ZERO     O$LLE,0
     014615  016504 000000         4816 PCDL3  ZERO     O$LBL,**
     014616  016666 000000         4817 PCDL4  ZERO     O$EPDV,**
     014617  016671 000000         4818 RCDL5  ZERO     O$EPDE,**
     014620  016475 2210 00        4819 COP    LDX1     OPT               GET RELATIVE LINK TO OPERATOR
     014621  000000 6000 00        4820 COPX   TZE      **                NO OPERATOR SO DONE
     014622  035214 0610 00        4821        ADX1     T$WORK            MAKE OPERATOR POINTER ABSOLUTE
     014623  000002 2220 11        4822        LDX2     2,1               GET PRIORITY OF OPERATOR IN XR - 2
     014624  016474 1020 00        4823        CMPX2    CPR               COMPARE WITH CURRENT OPERATOR PRIORITY
     014625  014621 6040 51        4824        TMI      COPX,I            IF LESS THEN EXIT
     014626  000004 2350 03        4825        LDA      4,DU              GET UNARY/BINARY FLAG IN A
     014627  015071 7550 00        4826        STA      OPF               AND STORE IN FLAG
     014630  000000 7220 11        4827        LXL2     0,1               GET LINK TO NEXT OPERATOR
     014631  016475 7420 00        4828        STX2     OPT               AND STORE LINK TO NEXT OPERATOR
     014632  000001 2210 11        4829        LDX1     1,1               GET POINTER TO OPERATOR DEFINITION CHAIN
     014633  015067 7410 00        4830 COPM   STX1     COPET             SAVE POINTER FOR POSSIBLE ERROR MESSAGE
     014634  035222 0610 00        4831        ADX1     T$STAB            MAKE POINTER ABSOLUTE
     014635  000001 0610 03        4832        ADX1     A$DF,DU           AND MAKE IT POINT TO DEFINITION CHAIN
     014636  000003 2260 03        4833        LDX6     3,DU              SET COERCION STRENGTH TO FIRM
  END OF BINARY CARD 00000246
     014637  011357 2200 00        4834        LDX0     RNGE              GET CURRENT RANGE NUMBER
     014640  006460 7400 00        4835        STX0     A$LEVEL           STORE FOR TABLE LOOK UP ROUTINE
     014641  006437 7000 00        4836 COP1   TSX0     A$TLU             LOOK UP OPERATOR IN SYMBOL TABLE
     014642  015040 7100 00        4837        TRA      COPE              TRANSFER TO ERROR - CANNOT IDENTIFY OPERATOR
     014643  000001 2220 11        4838        LDX2     1,1               GET TYPE OF SYMBOL DEFINITION
     014644  006413 1020 03        4839        CMPX2    D$OP,DU           IS IT AN OPERATOR DEFINITION
     014645  015040 6010 00        4840        TNZ      COPE              NO - CANNOT IDENTIFY OPERATOR
     014646  000002 2270 11        4841        LDX7     2,1               GET MODE OF OPERATOR
     014647  010630 7000 00        4842        TSX0     A$XFER            MAKE MODE POINTER UNIQUE AND ABSOLUTE
     014650  777777 2230 17        4843        LDX3     -1,7              GET NUMBER OF PARAMETERS PLUS TWO IN XR - 3
     014651  015071 1030 00        4844        CMPX3    OPF               ARE THERE THE RIGHT NUMBER OF PARAMETERS
     014652  014641 6010 00        4845        TNZ      COP1              NO - KEEP LOOKING
     014653  000000 2230 17        4846        LDX3     0,7               GET TYPE OF MODE
     014654  017001 1030 03        4847        CMPX3    M$PROC,DU         IS IT A PROCEDURE
     014655  777777 6010 00        4848        TNZ      $ERROR            ERROR - COMPILER ERROR
     014656  015071 7200 00        4849        LXL0     OPF               GET UNARY/BINARY FLAG
     014657  014662 6050 00        4850        TPL      COP5              TRANSFER IF BINARY OPERATOR
     014660  000001 1670 03        4851        SBX7     1,DU              ADJUST MODE TABLE POINTER
     014661  014664 7100 00        4852        TRA      COP6              TRANSFER FOR UNARY OPERATOR
     014662  000001 2230 17        4853 COP5   LDX3     1,7               GET MODE OF LEFT OPERAND
     014663  016453 7430 00        4854        STX3     MODE1             AND SAVE
     014664  000002 2230 17        4855 COP6   LDX3     2,7               GET MODE OF RIGHT OPERAND
  END OF BINARY CARD 00000247
```

```
          2                                       PASS 2

     014665  016454 7430 00      4856         STX3    MODE2         AND SAVE
     014666  000003 2230 17      4857         LDX3    3,7           GET MODE OF RESULT
     014667  016455 4430 00      4858         SXL3    MODE3         AND SAVE
     014670  000002 7200 11      4859         LXL0    2,1           GET MACRO FLAG IN XR = 0
     014671  015070 7400 00      4860         STX0    COPT          AND STORE
     014672  035220 1610 00      4861         SBX1    T$DEF         MAKE DEFINITION POINTER RELATIVE
     014673  016456 7410 00      4862         STX1    OPDEF         AND SAVE
     014674  035234 2210 00      4863         LDX1    A$WORK        GET POINTER TO END OF WORKING STACK
     014675  777777 1610 11      4864         SBX1    -1,1          GET POINTER TO LAST ENTRY IN WORKING STACK
     014676  035234 1610 00      4865         SBX1    A$WORK        MAKE POINTER RELATIVE
     014677  016446 7410 00      4866         STX1    VALP          AND INITIALIZE POINTER TO CURRENT VALUE
     014700  016454 2210 00      4867         LDX1    MODE2         GET MODE OF RIGHT OPERAND
     014701  016202 7410 00      4868         STX1    BMODE         MAKE IT TARGET MODE
     014702  015253 7000 00      4869         TSX0    C             ATTEMPT COERCION OF RHS
     014703  015020 7100 00      4870         TRA     COP4          TRANSFER IF UNSUCCESSFUL
     014704  015070 2340 00      4871         SZN     COPT          CHECK MACRO FLAG
     014705  014710 6010 00      4872         TNZ     COP61         TRANSFER IF OPERATOR IS DEFINED BY MACRO
     014706  016176 2340 00      4873         SZN     CF            SEE IF STRONG COERCIONS WERE USED
     014707  015020 6000 00      4874         TZE     COP4          TRANSFER IF STRONG - ILLEGAL FOR OPERATORS
     014710  015071 7200 00      4875 COP61   LXL0    OPF           GET UNARY/BINARY FLAG
     014711  014721 6050 00      4876         TPL     COP7          TRANSFER IF BINARY OPERATOR
     014712  016204 7000 00      4877         TSX0    DC            CONTINUE COERCION OF RHS TO COMPLETION
   END OF BINARY CARD 00000248
     014713  015025 7000 00      4878         TSX0    CFS           FORCE OPERAND TO STACK IF USER OPERATOR
     014714  035234 2210 00      4879         LDX1    A$WORK        GET POINTER TO END OF WORKING STACK
     014715  777777 1610 11      4880         SBX1    -1,1          GET POINTER TO LAST CONTROL BLOCK
     014716  016455 7220 00      4881         LXL2    MODE3         GET MODE OF RESULT OF OPERATOR
     014717  000001 7420 11      4882         STX2    1,1           AND STORE AS MODE OF VALUE
     014720  014756 7100 00      4883         TRA     COP8          AND CONTINUE
     014721  016446 2210 00      4884 COP7    LDX1    VALP          GET CURRENT VALUE POINTER
     014722  035214 0610 00      4885         ADX1    T$WORK        MAKE POINTER ABSOLUTE
     014723  000000 2220 11      4886         LDX2    0,1           GET TYPE OF CONTROL BLOCK IN XR = 2
     014724  004000 1020 03      4887         CMPX2   W$OP,DU       SEE IF IT IS AN OPERATOR CONTROL BLOCK
     014725  777777 6010 00      4888         TNZ     $ERROR        IF NOT - COMPILER ERROR
     014726  777777 1610 11      4889         SBX1    -1,1          STEP TO PRECEEDING CONTROL BLOCK
     014727  035234 1610 00      4890         SBX1    A$WORK        MAKE CONTROL BLOCK POINTER RELATIVE
     014730  016446 7410 00      4891         STX1    VALP          AND STORE AS POINTER TO CURRENT VALUE
     014731  016453 2210 00      4892         LDX1    MODE1         GET MODE OF LHS
     014732  016202 7410 00      4893         STX1    BMODE         MAKE IT TARGET MODE
     014733  015253 7000 00      4894         TSX0    C             ATTEMPT COERCION OF LHS
     014734  015016 7100 00      4895         TRA     COP3          TRANSFER IF UNSUCCESSFUL
     014735  015070 2340 00      4896         SZN     COPT          CHECK MACRO FLAG
     014736  014741 6010 00      4897         TNZ     COP71         TRANSFER IF OPERATOR IS DEFINED BY MACRO
     014737  016176 2340 00      4898         SZN     CF            SEE IF STRONG COERCIONS WERE USED
     014740  015016 6000 00      4899         TZE     COP3          TRANSFER IF STRONG - ILLEGAL FOR OPERATORS
   END OF BINARY CARD 00000249
     014741  016204 7000 00      4900 COP71   TSX0    DC            DO COERCION OF LHS
     014742  015025 7000 00      4901         TSX0    CFS           FORCE OPERAND TO STACK IF USER OPERATOR
     014743  016204 7000 00      4902         TSX0    DC            DO COERCION OF RHS
     014744  015025 7000 00      4903         TSX0    CFS           FORCE OPERAND TO STACK IF USER OPERATOR
```

```
              2                                             PASS 2

    014745  035234 2210 00          4904          LDX1     A$WORK            GET POINTER TO END OF WORKING STACK
    014746  777777 1610 11          4905          SBX1     -1,1              MAKE XR - 1 POINT TO LAST ELEMENT IN STACK
    014747  000002 2220 03          4906          LDX2     2,DU              GET A 2 AS NUMBER OF ELEMENTS BELOW THIS ONE
    014750  000000 4420 11          4907          SXL2     0,1               STORE IN COUNT FIELT
    014751  016455 7220 00          4908          LXL2     MODE3             GET MODE OF VALUE
    014752  000001 7420 11          4909          STX2     1,1               STORE MODE OF FORMULA IN VALUE CONTROL BLOCK
    014753  012530 7000 00          4910          TSX0     DELW              COMBINE OPERATOR AND OPERANDS TO ONE VALUE
    014754  035234 2210 00          4911          LDX1     A$WORK            GET POINTER TO END OF WORKING STACK
    014755  777777 1610 11          4912          SBX1     -1,1              GET POINTER TO NEW COMBINED VALUE CONTROL ELEMENT
    014756  006462 7000 00          4913 COP8     TSX0     A$GLL             GET LL WORD FOR IDENTIFIED OPERATOR
    014757  000003 2550 11          4914          ORSA     3,1               AND OR IT INTO COMBINED VALUE
    014760  013573 2550 00          4915          ORSA     RLL               OR INTO RANGE LL WORD
    014761  035234 1610 00          4916          SBX1     A$WORK            MAKE VALUE CONTROL BLOCK POINTER RELATIVE
    014762  016446 7410 00          4917          STX1     VALP              AND STORE POINTER TO CURRENT VALUE
    014763  015070 2340 00          4918          SZN      COPT              CHECK MACRO FLAG
    014764  014775 6000 00          4919          TZE      COP9              TRANSFER IF USER DEFINED OPERATOR
    014765  016456 2350 00          4920          LDA      OPDEF             GET POINTER TO OPERATOR DEFINITION
    014766  000022 7710 00          4921          ARL      18                MOVE IT TO AL
  END OF BINARY CARD 00000250
    014767  016501 0750 03          4922          ADA      O$OPE,DU          ADD MACRO OPERATOR CODE
    014770  012614 7000 00          4923          TSX0     CAD               AND ADD AT END OF OUTPUT CODE
    014771  035234 2210 00          4924          LDX1     A$WORK            GET POINTER TO END OF WORKING STACK
    014772  777777 1610 11          4925          SBX1     -1,1              GET POINTER TO LAST CONTROL BLOCK IN WORK
    014773  000002 0540 11          4926          AOS      2,1               MAKE LOC/LEN WORD INCLUDE ADDED CODE
    014774  015013 7100 00          4927          TRA      COP10             AND CONTINUE
    014775  016456 2350 00          4928 COP9     LDA      OPDEF             GET POINTER TO OPERATOR DEFINITION
    014776  000022 7710 00          4929          ARL      18                MOVE TO AL
    014777  016476 0750 03          4930          ADA      O$OP,DU           AD OPERATOR OP CODE
    015000  012403 7000 00          4931          TSX0     INS               AND INSERT CODE IN FRONT OF VALUE
    015001  035234 2210 00          4932          LDX1     A$WORK            GET POINTER TO END OF WORKING STACK
    015002  777777 1610 11          4933          SBX1     -1,1              GET POINTER TO FINAL VALUE
    015003  777776 3350 07          4934          LCA      -2,DL             GET -1 IN AU AND 2 IN AL
    015004  000002 0550 11          4935          ASA      2,1               MAKE LOC AND LEN REFER TO NEW LONGER VALUE
    015005  016455 2350 00          4936          LDA      MODE3             GET POINTER TO RESULT MODE
    015006  016520 0750 03          4937          ADA      O$ENTER,DU        ADD ENTER OP CODE
    015007  012614 7000 00          4938          TSX0     CAD               AND ADD AT END OF OUTPUT CODE
    015010  016456 2210 00          4939          LDX1     OPDEF             GET POINTER TO IDENTIFIED OPERATOR
    015011  035220 0610 00          4940          ADX1     T$DEF             MAKE POINTER ABSOLUTE
    015012  011651 7000 00          4941          TSX0     LK                LINK DEFINITION TO CHAIN FOR CURRENT LL
    015013  015071 7200 00          4942 COP10    LXL0     OPF               GET UNARY/BINARY FLAG
    015014  014620 6050 00          4943          TPL      COP               LOOP IF BINARY OPERATOR
  END OF BINARY CARD 00000251
    015015  014621 7100 51          4944          TRA      COPX,I            EXIT IF UNARY OPERATOR
    015016  035235 2340 54          4945 COP3     SZN      A$STACK,DI        DELETE A WORD FROM THE CONTROL STACK
    015017  015016 6010 00          4946          TNZ      COP3              TRANSFER IF NOT FINISHED
    015020  035235 2340 54          4947 COP4     SZN      A$STACK,DI        DELETE A WORD FROM THE CONTROL STACK
    015021  015020 6010 00          4948          TNZ      COP4              TRANSFER IF NOT FINISHED
    015022  016456 2210 00          4949 COP41    LDX1     OPDEF             GET POINTER TO CURRENT OPERATOR DEFINITION
    015023  035220 0610 00          4950          ADX1     T$DEF             MAKE IT ABSOLUTE
    015024  014641 7100 00          4951          TRA      COP1              AND TRY TO FIND ANOTHER DEFINITION
```

```
      015025  015070 2340 00        4952 CFS    SZN     COPT            SEE IF OPERATOR IS DEFINED BY A MACRO
      015026  000000 6010 10        4953        TNZ     0,0             RETURN IF SO
      015027  015035 7400 00        4954        STX0    CFSX            SAVE RETURN
      015030  017034 6350 00        4955        EAA     O$FS            GET O$FS COMMAND
      015031  012375 7000 00        4956        TSX0    ADD             AND ADD FORCE TO STACK COMMAND AFTER OPERAND
      015032  016446 2210 00        4957        LDX1    VALP            GET POINTER TO VALUE CONTROL BLOCK
      015033  035234 0610 00        4958        ADX1    A$WORK          MAKE POINTER ABSOLUTE
      015034  000002 0540 11        4959        AOS     2,1             MAKE VALUE INCLUDE FORCE TO STACK COMMAND
      015035  000000 7100 00        4960 CFSX   TRA     **              AND RETURN
      015036  015056 2350 00        4961 COPEP  LDA     COPPM           GET TALLY WORD FOR ERROR MESSAGE
      015037  015041 7100 00        4962        TRA     COPE1           AND PRINT MESSAGE
      015040  015047 2350 00        4963 COPE   LDA     COPEM           GET TALLY WORD FOR ERROR MESSAGE
      015041  006562 7000 00        4964 COPE1  TSX0    1$XXX           PRINT ERROR MESSAGE
      015042  015067 2210 00        4965        LDX1    COPET           GET POINTER TO SYMBOL TABLE ENTRY
 END OF BINARY CARD 00000252
      015043  035222 0610 00        4966        ADX1    T$STAB          MAKE IT ABSOLUTE
      015044  000000 2350 11        4967        LDA     A$SC,1          GET TALLY WORD IN A REGISTER
      015045  006554 7000 00        4968        TSX0    1$XXS           PRINT BAD OPERATOR
      015046  006241 7100 00        4969        TRA     A$ERRF          AND PRINT OUT FOLLOWING SYMBOLS
      015047  015050 0031 40        4970 COPEM  TALLYB  *+1,24+1
      015050  103101116116          4971        UASCI   6,CANNOT IDENTIFY OPERATOR
      015051  117124040111
      015052  104105116124
      015053  111106131040
      015054  117120105122
      015055  101124117122
      015056  015057 0041 40        4972 COPPM  TALLYB  *+1,32+1
      015057  103101116116          4973        UASCI   8,CANNOT FIND PRIORITY OF OPERATOR
      015060  117124040106
      015061  111116104040
      015062  120122111117
      015063  122111124131
      015064  040117106040
      015065  117120105122
      015066  101124117122
      015067  000000 000000         4974 COPET  ZERO
      015070  000000 000000         4975 COPT   ZERO
 END OF BINARY CARD 00000253
      015071  000000 000000         4976 OPF    ZERO
      015072  015100 7400 00        4977 STRNG  STX0    STRNX           SAVE RETURN
      015073  016460 2270 00        4978        LDX7    DECLR           GET TARGET MODE
      015074  016202 7470 00        4979        STX7    BMODE           AND STORE FOR COERCION ROUTINES
      015075  015253 7000 00        4980        TSX0    C               COERCE VALUE TO TARGET MODE
      015076  777777 7100 00        4981        TRA     $ERROR          COERCION FAILED - ERROR
      015077  016204 7000 00        4982        TSX0    DC              DO INDICATED COERCIONS
      015100  000000 7100 00        4983 STRNX  TRA     **              AND RETURN
      015101  000003 2230 03        4984 FIRM   LDX3    3,DU            INDICATE FIRM COERCION
      015102  015106 7100 00        4985        TRA     BLC             GO DO COERCION
      015103  000002 2230 03        4986 WEAK   LDX3    2,DU            INDICATE WEAK COERCION
      015104  015106 7100 00        4987        TRA     BLC             GO DO COERCION
```

```
               2                                                   PASS 2

      015105  000001 2230 03          4988 SOFT   LDX3    1,DU              INDICATE SOFT COERCION
      015106  015113 7400 00          4989 BLC    STX0    BLCX              SAVE RETURN
      015107  015114 7000 00          4990        TSX0    BB                GET TARGET MODE FOR COERCION
      015110  015253 7000 00          4991        TSX0    C                 COERCE VALUE TO TARGET MODE
      015111  777777 7100 00          4992        TRA     $ERROR            COERCION FAILED - ERROR
      015112  016204 7000 00          4993        TSX0    DC                DO INDICATED COERCIONS
      015113  000000 7100 00          4994 BLCX   TRA     **                AND RETURN
      015114  015247 7400 00          4995 BB     STX0    BX                SAVE RETURN
      015115  016446 2210 00          4996        LDX1    VALP              GET POINTER TO VALUE CONTROL BLOCK IN XR - 1
      015116  035234 0610 00          4997        ADX1    A$WORK            MAKE VALUE POINTER ABSOLUTE
    END OF BINARY CARD 00000254
      015117  016202 4500 00          4998        STZ     BMODE             ZERO OUT TARGET MODE
      015120  015250 4500 00          4999        STZ     BC                INITIALIZE LOCAL COUNT
      015121  000000 2240 03          5000        LDX4    0,DU              INITIALIZE COUNT
      015122  000000 2200 11          5001 BB1    LDX0    0,1               GET TYPE OF VALUE IN XR - 0
      015123  040000 3000 03          5002        CANX0   W$PAR,DU          SEE IF IT IS A DISPLAYED VALUE
      015124  015244 6010 00          5003        TNZ     BB13              FAIL IF IT IS A DISPLAYED VALUE
      015125  020000 3000 03          5004        CANX0   W$BAL,DU          IS IT A BALANCED VALUE
      015126  015133 6000 00          5005        TZE     BB3               TRANSFER IF NOT A BALANCED VALUE
      015127  000000 7200 11          5006        LXL0    0,1               GET COUNT IN XR - 0
      015130  015131 7400 00          5007        STX0    BB2               STORE TO ADD X REGISTERS TOGETHER
      015131  000000 0640 03          5008 BB2    ADX4    **,DU             ADD NUMBER OF BALANCED VALUES TO DO
      015132  015241 7100 00          5009        TRA     BB12              AND CONTINUE
      015133  310000 3000 03          5010 BB3    CANX0   W$SKIP+W$NIL+W$VAC,DU SEE IF VALUE IS A SKIP, NIL, OR VACUUM
      015134  015241 6010 00          5011        TNZ     BB12              TRANSFER IF SO TO CONTINUE
      015135  000001 2270 11          5012        LDX7    1,1               GET MODE OF VALUE IN XR - 7
      015136  000031 1070 03          5013        CMPX7   M$LBL,DU          SEE IF VALUE IS A LABEL
      015137  015241 6000 00          5014        TZE     BB12              TRANSFER IF SO
      015140  015251 4500 00          5015        STZ     BN                INITIALIZE FLAG/MODE COUNT
      015141  000000 2250 03          5016        LDX5    0,DU              INITIALIZE XR - 5
      015142  000000 2260 03          5017        LDX6    0,DU              INITIALIZE XR - 6
      015143  010630 7000 00          5018 BB4    TSX0    A$XFER            GET ABSOLUTE POINTER TO MODE IN XR - 7
      015144  777777 2200 17          5019        LDX0    -1,7              GET LENGTH OF MODE IN XR - 0
    END OF BINARY CARD 00000255
      015145  000002 1000 03          5020        CMPX0   2,DU              SEE IF IT IS A COERCIBLE MODE LENGTH
      015146  015173 6010 00          5021        TNZ     BB7               TRANSFER IF NOT COERCIBLE
      015147  000000 2200 17          5022        LDX0    0,7               GET TYPE OF MODE IN XR - 0
      015150  017001 1000 03          5023        CMPX0   M$PROC,DU         SEE IF IT IS A PROCEDURE MODE
      015151  015154 6010 00          5024        TNZ     BB5               TRANSFER IF NOT TO CONTINUE CHECKING
      015152  000001 2270 17          5025        LDX7    1,7               GET DEPROCEDURED MODE IN XR - 7
      015153  015143 7100 00          5026        TRA     BB4               AND CONTINUE COERCING
      015154  016762 1000 03          5027 BB5    CMPX0   M$REF,DU          SEE IF TYPE IS A REFERENCE
      015155  015173 6010 00          5028        TNZ     BB7               TRANSFER IF NOT
      015156  000000 6250 17          5029        EAX5    0,7               SAVE FIRST REFERENCE MODE IN XR - 5
      015157  000000 6260 17          5030 BB6    EAX6    0,7               SAVE COERCIBLE MODE IN XR - 6
      015160  000001 2270 17          5031        LDX7    1,7               DEREFERENCE OR DEPROCEDURE MODE
      015161  015251 0540 00          5032        AOS     BN                COUNT MODES FROM FIRST REFERENCE
      015162  010630 7000 00          5033        TSX0    A$XFER            MAKE MODE ABSOLUTE
      015163  777777 2200 17          5034        LDX0    -1,7              GET LENGTH OF MODE IN XR - 0
      015164  000002 1000 03          5035        CMPX0   2,DU              IS IT A COERCIBLE MODE LENGTH
```

```
                 2                                                PASS 2

     015165  015173 6010 00        5036         TNZ      BB7            TRANSFER IF NOT COERCIBLE
     015166  000000 2200 17        5037         LDX0     0,7            GET TYPE OF MODE IN XR - 0
     015167  017001 1000 03        5038         CMPX0    M$PROC,DU      IS IT A PROCEDURE MODE
     015170  015157 6000 00        5039         TZE      BB6            TRANSFER IF YES TO CONTINUE COERCION
     015171  016762 1000 03        5040         CMPX0    M$REF,DU       IS IT A REFERENCE MODE
     015172  015157 6000 00        5041         TZE      BB6            TRANSFER IF YES TO CONTINUE COERCION
 END OF BINARY CARD 00000256
     015173  035216 1670 00        5042 BB7     SBX7     T$MODE         GET RELATIVE MODE IN XR - 7
     015174  000002 1030 03        5043         CMPX3    2,DU           WHAT IS THE STRENGTH OF THE COERCION
     015175  015211 6050 00        5044         TPL      BB9            TRANSFER IF STRENGTH IS FIRM OR WEAK
     015176  035216 1650 00        5045         SBX5     T$MODE         MAKE XR - 5 POINTER RELATIVE
     015177  777777 6040 00        5046         TMI      $ERROR         - IMPOSSIBLE COERCION
     015200  016202 2340 00        5047         SZN      BMODE          IS THERE ALREADY A BMODE
     015201  015205 6000 00        5048         TZE      BB8            TRANSFER IF NO BMODE CALCULATED YET
     015202  015251 2350 00        5049         LDA      BN             GET NUMBER OF MODES TO BASE MODE
     015203  015250 1150 00        5050         CMPA     BC             COMPARE WITH PREVIOUS NUMBER
     015204  015241 6050 00        5051         TPL      BB12           TRANSFER IF OLD MODE IS STILL GOOD
     015205  016202 7450 00        5052 BB8     STX5     BMODE          STORE TARGET MODE IN BMODE
     015206  015251 2350 00        5053         LDA      BN             GET CURRENT MODE COUNT
     015207  015250 7550 00        5054         STA      BC             AND ALSO STORE
     015210  015241 7100 00        5055         TRA      BB12           AND CONTINUE
     015211  016202 2340 00        5056 BB9     SZN      BMODE          IS THERE A TARGET MODE CALCULATED YET
     015212  015231 6010 00        5057         TNZ      BB11           TRANSFER IF THERE IS A TARGET MODE
     015213  000003 1030 03        5058         CMPX3    3,DU           IS THIS A FIRM COERCION
     015214  015226 6050 00        5059         TPL      BB10           TRANSFER IF FIRM COERCION
     015215  000000 6260 16        5060         EAX6     0,6            IS THERE POSSIBLY A REFERENCE TO BASE MODE
     015216  015226 6000 00        5061         TZE      BB10           TRANSFER IF NOT
     015217  000000 2200 16        5062         LDX0     0,6            GET TYPE OF MODE IN XR - 0
     015220  016762 1000 03        5063         CMPX0    M$REF,DU       IS IT A REFERENCE MODE
 END OF BINARY CARD 00000257
     015221  015226 6010 00        5064         TNZ      BB10           TRANSFER IF NOT REFERENCE
     015222  035216 1660 00        5065         SBX6     T$MODE         MAKE MODE POINTER RELATIVE
     015223  016202 7460 00        5066         STX6     BMODE          AND STORE AS TARGET MODE
     015224  015252 7500 00        5067         STC2     BF             SET FLAG INDICATING BMODE IS REFERENCE
     015225  015241 7100 00        5068         TRA      BB12           AND CONTINUE
     015226  016202 7470 00        5069 BB10    STX7     BMODE          STORE MODE AS TARGET MODE
     015227  015252 4500 00        5070         STZ      BF             RESET REFERENCE FLAG
     015230  015241 7100 00        5071         TRA      BB12           AND CONTINUE
     015231  016202 1070 00        5072 BB11    CMPX7    BMODE          ARE BOTH TARGET MODES THE SAME
     015232  015241 6000 00        5073         TZE      BB12           TRANSFER IF THE SAME TO CONTINUE
     015233  000003 1030 03        5074         CMPX3    3,DU           IS IT A FIRM COERCION
     015234  015244 6050 00        5075         TPL      BB13           TRANSFER IF FIRM COERCION - FAILURE
     015235  015252 2340 00        5076         SZN      BF             IS REFERENCE FLAG SET
     015236  015241 6000 00        5077         TZE      BB12           TRANSFER IF NOT - FAILURE
     015237  016202 7470 00        5078         STX7     BMODE          STORE NEW DEREFERENCED TARGET MODE
     015240  015252 4500 00        5079         STZ      BF             RESET REFERENCE FLAG
     015241  000005 1610 03        5080 BB12    SBX1     5,DU           STEP TO PREVIOUS VALUE
     015242  000001 1640 03        5081         SBX4     1,DU           DECREMENT NUMBER OF VALUES LEFT TO PROCESS
     015243  015122 6050 00        5082         TPL      BB1            TRANSFER IF ANY MORE VALUES TO PROCESS
     015244  016202 2270 00        5083 BB13    LDX7     BMODE          GET CALCULATED TARGET MODE
     015245  016203 7470 00        5084        STX7    TMODE               AND SAVE FOR REFERENCE
     015246  777777 6000 00        5085        TZE     $ERROR              TRANSFER IF NO MODE WAS CALCULATED - ERROR
 END OF BINARY CARD 00000258
     015247  000000 7100 00        5086 BX     TRA     **                  AND EXIT
     015250  000000 000000         5087 BC     ZERO
     015251  000000 000000         5088 BN     ZERO
     015252  000000 000000         5089 BF     ZERO
     015253  035234 2210 00        5090 C      LDX1    A$WORK              GET POINTER TO END OF WORKING STACK
     015254  035214 1610 00        5091        SBX1    T$WORK              MAKE POINTER RELATIVE
     015255  016200 7410 00        5092        STX1    CW                  AND STORE AS VALP CORRECTION CONSTANT
     015256  016446 0410 00        5093        ASX1    VALP                MAKE VALP RELATIVE TO T$WORK
     015257  035235 4500 56        5094        STZ     A$STACK,ID          STORE A ZERO IN BOTTOM OF CONTROL STACK
     015260  005742 7170 00        5095        XED     T$SOVF              CHECK FOR STACK OVERFLOW
     015261  000001 3350 03        5096 C0     LCA     1,DU                GET FIRM COERCION FLAG
     015262  016176 7550 00        5097        STA     CF                  AND STORE FLAG INDICATING FIRM COERCION
     015263  016177 7550 00        5098        STA     FF                  STORE IN STRONG COERCIONS ARE ALLOWED FLAG
     015264  015530 7400 00        5099        STX0    CX                  SAVE RETURN
     015265  016446 2200 00        5100        LDX0    VALP                GET POINTER TO VALUE CONTROL BLOCK TO COERCE
     015266  035214 0600 00        5101        ADX0    T$WORK              MAKE POINTER ABSOLUTE
     015267  000000 2210 10        5102        LDX1    0,0                 GET TYPE OF VALUE IN XR - 1
     015270  020000 3010 03        5103        CANX1   W$BAL,DU            IS IT A BALANCED VALUE
     015271  015641 6010 00        5104        TNZ     BAL                 TRANSFER IF VALUE IS BALANCED
     015272  040000 3010 03        5105        CANX1   W$PAR,DU            IS IT A DISPLAY VALUE
     015273  015725 6010 00        5106        TNZ     PAR                 TRANSFER IF VALUE IS A DISPLAY
     015274  200000 3010 03        5107        CANX1   W$SKIP,DU           SEE IF VALUE IS A SKIP
 END OF BINARY CARD 00000259
     015275  015550 6010 00        5108        TNZ     CSK                 TRANSFER IF VALUE IS A SKIP
     015276  100000 3010 03        5109        CANX1   W$NIL,DU            SEE IF VALUE IS A NIL
     015277  015557 6010 00        5110        TNZ     CNIL                TRANSFER IF VALUE IS A NIL
     015300  010000 3010 03        5111        CANX1   W$VAC,DU            SEE IF VALUE IS A VACUUM
     015301  015606 6010 00        5112        TNZ     VAC                 TRANSFER IF VALUE IS A VACUUM
     015302  400000 3010 03        5113        CANX1   W$VALUE,DU          IS IT A VALUE CONTROL BLOCK
     015303  777777 6000 00        5114        TZE     $ERROR              NO - COMPILER ERROR MUST BE SOMETHING
     015304  000001 2270 10        5115        LDX7    1,0                 GET MODE OF VALUE TO COERCE IN XR - 7
     015305  000031 1070 03        5116        CMPX7   M$LBL,DU            SEE IF MODE IS A LABEL
     015306  015573 6000 00        5117        TZE     LC                  TRANSFER IF VALUE IS A LABEL
     015307  035234 4500 56        5118        STZ     A$WORK,ID           FLAG BOTTOM OF WORKING STACK WITH A ZERO
     015310  005754 7170 00        5119        XED     T$WOVF              CHECK FOR STACK OVERFLOW
     015311  016124 7000 00        5120        TSX0    CRED                REDUCE MODE TO NONPROC OR NONREF MODE
     015312  035234 7550 56        5121        STA     A$WORK,ID           STORING IN WORKING STACK
     015313  005754 7170 00        5122        XED     T$WOVF              CHECK FOR STACK OVERFLOW
     015314  016201 7470 00        5123        STX7    AMODE               STORE REDUCED MODE IN AMODE
     015315  035235 2200 00        5124        LDX0    A$STACK             GET POINTER TO TOP OF CONTROL STACK
     015316  035215 1600 00        5125        SBX0    T$STACK             MAKE POINTER RELATIVE
     015317  000000 6350 10        5126        EAA     0,0                 GET RELATIVE POINTER IN AU
     015320  777777 0750 07        5127        ADA     -1,DL               GET FLAG IN AL
     015321  035234 7550 56        5128        STA     A$WORK,ID           AND STORE IN WORKING STACK
     015322  005754 7170 00        5129        XED     T$WOVF              CHECK FOR STACK OVERFLOW
 END OF BINARY CARD 00000260
     015323  016202 2270 00        5130 C1     LDX7    BMODE               GET DESIRED MODE IN XR - 7
```

```
          2                                                 PASS 2

    015324  016124 7000 00         5131        TSX0     CRED           REDUCE MODE TO NONPROC OR NONREF MODE
    015325  035235 7550 56         5132        STA      A$STACK,ID     STORING IN CONTROL STACK
    015326  005742 7170 00         5133        XED      T$SOVF         CHECK FOR STACK OVERFLOW
    015327  016202 7470 00         5134        STX7     BMODE          STORE REDUCED MODE AS NEW TARGET MODE
    015330  016201 1070 00         5135        CMPX7    AMODE          ARE THE TWO REDUCED MODES THE SAME
    015331  015402 6010 00         5136        TNZ      C11            NO - MUST WORK ON BMODE FURTHER SO TRANSFER
    015332  035234 7200 54         5137 C2     LXL0     A$WORK,DI      DELETE A WORD FROM THE WORKING STACK
    015333  015332 6050 00         5138        TPL      C2             TRANSFER IF MORE TO DELETE
    015334  035234 2350 54         5139 C3     LDA      A$WORK,DI      GET NEXT COERCION FROM WORKING STACK
    015335  015370 6000 00         5140        TZE      C10            TRANSFER IF NO MORE
    015336  035235 1150 54         5141        CMPA     A$STACK,DI     SEE IF THE SAME AS COERCION IN CONTROL STACK
    015337  015334 6000 00         5142        TZE      C3             IF SO - DELETE BOTH COERCIONS AND LOOP
    015340  035235 2200 00         5143        LDX0     A$STACK        GET POINTER TO END OF CONTROL STACK IN XR - 0
    015341  000000 2210 10         5144        LDX1     0,0            GET LAST COERCION IN CONTROL STACK
    015342  016770 1010 03         5145        CMPX1    M$ROW,DU       IS IT A ROWING COERCION
    015343  015346 6000 00         5146        TZE      C4             TRANSFER IF SO
    015344  016776 1010 03         5147        CMPX1    M$ROWE,DU      IS IT AN END ROWING COERCION
    015345  015362 6010 00         5148        TNZ      C6             TRANSFER IF NOT
    015346  035235 2200 54         5149 C4     LDX0     A$STACK,DI     GET PREVIOUS COERCION FROM CONTROL STACK
    015347  016770 1000 03         5150        CMPX0    M$ROW,DU       IS IT A ROW COERCION
    015350  015346 6000 00         5151        TZE      C4             TRANSFER IF YES AND DELETE IT
  END OF BINARY CARD 00000261
    015351  016776 1000 03         5152        CMPX0    M$ROWE,DU      IS IT AN END ROW COERCION
    015352  015346 6000 00         5153        TZE      C4             TRANSFER IF YES AND DELETE IT
    015353  016762 1000 03         5154        CMPX0    M$REF,DU       IS COERCION A REFERENCE COERCION
    015354  015361 6010 00         5155        TNZ      C5             TRANSFER IF NOT A REFERENCE COERCION
U   015355  000000 2200 03         5156        LDX0     M$REFRW,DU     GET A REFERENCE ROW COERCION
    015356  035235 7400 56         5157        STX0     A$STACK,ID     AND CHANGE LAST REF COERCION TO REF ROW
    015357  035234 2350 54         5158        LDA      A$WORK,DI      DELETE WORK REFERENCE COERCION
    015360  015362 7100 00         5159        TRA      C6             AND CONTINUE
    015361  035235 0110 56         5160 C5     NOP      A$STACK,ID     RESTORE NONREFERENCE COERCION
    015362  035235 0110 56         5161 C6     NOP      A$STACK,ID     RESTORE NON-MATCHING COERCION IN STACK
    015363  000003 0750 03         5162 C7     ADA      3,DU           CHANGE REF TO DEREF AND PROC TO DEPROC
    015364  035235 7550 56         5163 C8     STA      A$STACK,ID     AND STORE COERCION IN CONTROL STACK
    015365  005742 7170 00         5164        XED      T$SOVF         CHECK FOR STACK OVERFLOW
    015366  035234 2350 54         5165 C9     LDA      A$WORK,DI      GET NEXT COERCION FROM WORKING STACK
    015367  015363 6010 00         5166        TNZ      C7             TRANSFER IF THERE ARE MORE COERCIONS
    015370  016446 2350 00         5167 C10    LDA      VALP           GET POINTER TO CURRENT CONTROL BLOCK
    015371  016200 1750 00         5168        SBA      CW             MAKE POINTER RELATIVE TO A$WORK
    015372  000022 7710 00         5169        ARL      18             PUT IN AL
    015373  016446 0750 03         5170        ADA      VALP,DU        ADD VALP COERCION COMMAND
    015374  035235 7550 56         5171        STA      A$STACK,ID     AND ADD TO CONTROL STACK
    015375  005742 7170 00         5172        XED      T$SOVF         CHECK FOR STACK OVERFLOW
    015376  777773 2200 03         5173        LDX0     -5,DU          GET LENGTH OF CONTROL BLOCK IN XR - 0
  END OF BINARY CARD 00000262
    015377  016446 0400 00         5174        ASX0     VALP           STEP TO NEXT CONTROL BLOCK
    015400  015530 2200 00         5175        LDX0     CX             GET RETURN ADDRESS IN XR - 0
    015401  000001 7100 10         5176        TRA      1,0            AND GIVE SUCCESSFUL RETURN
    015402  016177 2340 00         5177 C11    SZN      FF             ARE STRONG COERCIONS ALLOWED
    015403  015406 6000 00         5178        TZE      C11,3          TRANSFER IF STRONG COERCIONS ARE NOT ALLOWED
```

```
     015404  000001 1070 03        5179         CMPX7   M$VOID,DU        IS THE TARGET MODE VOID
     015405  015531 6000 00        5180         TZE     VC               TRANSFER IF YES TO VOID ROUTINE
     015406  010630 7000 00        5181 C11,3   TSX0    A$XFER           GET ABSOLUTE POINTER TO MODE IN XR - 7
     015407  000000 2200 17        5182         LDX0    0,7              GET TYPE OF MODE IN XR - 0
     015410  016177 2340 00        5183         SZN     FF               ARE STRONG COERCIONS ALLOWED
     015411  015420 6000 00        5184         TZE     C11,7            TRANSFER IF STRONG COERCIONS ARE NOT ALLOWED
     015412  016754 1000 03        5185         CMPX0   M$PRIM,DU        IS IT A PRIMITIVE MODE
     015413  015460 6000 00        5186         TZE     C14              TRANSFER IF YES
     015414  016770 1000 03        5187         CMPX0   M$ROW,DU         IS IT A ROW MODE
     015415  015460 6000 00        5188         TZE     C14              TRANSFER IF YES
     015416  016776 1000 03        5189         CMPX0   M$ROWE,DU        IS IT AN END ROW MODE
     015417  015460 6000 00        5190         TZE     C14              TRANSFER IF YES
     015420  017007 1000 03        5191 C11,7   CMPX0   M$UNION,DU       IS IT A UNION MODE
     015421  015514 6010 00        5192         TNZ     C17              TRANSFER IF NOT
     015422  000000 6260 17        5193         EAX6    0,7              SAVE UNITED MODE IN XR - 6
     015423  016201 2270 00        5194         LDX7    AMODE            GET MODE OF VALUE TO BE COERCED
     015424  010630 7000 00        5195         TSX0    A$XFER           MAKE MODE UNIQUE AND ABSOLUTE
  END OF BINARY CARD 00000263
     015425  000000 2200 17        5196         LDX0    0,7              GET TYPE OF MODE IN XR - 0
     015426  017007 1000 03        5197         CMPX0   M$UNION,DU       IS IT A UNITED MODE
     015427  015455 6010 00        5198         TNZ     C13              TRANSFER IF NOT
     015430  777777 2210 17        5199         LDX1    -1,7             GET LENGTH OF SOURCE UNION
     015431  777777 2350 16        5200         LDA     -1,6             GET LENGTH OF DESIRED MODE UNION
     015432  777777 6350 01        5201         EAA     -1,AU            DECREMENT LENGTH TO GET NUMBER OF MODES IN UNION
     015433  000010 7710 00        5202         ARL     8                POSITION COUNT FOR REPEAT
     015434  000000 6200 05        5203 C12     EAX0    0,AL             GET COUNT IN XR - 0 FOR REPEAT
     015435  000001 6240 16        5204         EAX4    1,6              GET POINTER TO NEXT TARGET MODE IN XR - 4
     015436  000001 2230 17        5205         LDX3    1,7              GET FIRST MODE OF VALUE UNION IN XR - 3
     015437  000100 5202 01        5206         RPTX    ,1,TZE           SEARCH FOR VALUE MODE IN DESIRED MODE
     015440  000000 1030 14        5207         CMPX3   0,4              COMPARE GIVEN MODE WITH DESIRED MODE
     015441  015455 6010 00        5208         TNZ     C13              TRANSFER IF NOT A POSSIBLE COERCION
     015442  000001 0670 03        5209         ADX7    1,DU             STEP VALUE MODE POINTER
     015443  000001 1610 03        5210         SBX1    1,DU             DECREMENT NUMBER OF COMPONENTS LEFT TO CHECK
     015444  015434 6010 00        5211         TNZ     C12              TRANSFER IF MORE TO CHECK
     015445  000000 6270 16        5212         EAX7    0,6              RESTORE DESIRED MODE IN XR - 7
     015446  035216 1670 00        5213         SBX7    T$MODE           MAKE MODE POINTER RELATIVE
     015447  000000 6350 17        5214         EAA     0,7              GET MODE IN A REGISTER
     015450  000022 7710 00        5215         ARL     18               MOVE MODE TO AL
     015451  017007 0750 03        5216         ADA     M$UNION,DU       ADD UNITE COMMAND
     015452  035235 7550 56        5217         STA     A$STACK,ID       AND STORE IN CONTROL STACK
  END OF BINARY CARD 00000264
     015453  005742 7170 00        5218         XED     T$SOVF           CHECK FOR STACK OVERFLOW
     015454  015332 7100 00        5219         TRA     C2               GO TO END CLEANUP
     015455  017007 2200 03        5220 C13     LDX0    M$UNION,DU       RESTORE TYPE OF MODE IN XR - 0
     015456  000000 6270 16        5221         EAX7    0,6              RESTORE DESIRED MODE IN XR - 7
     015457  015461 7100 00        5222         TRA     C14,1            TRANSFER BECAUSE NOT STRONG COERCION
     015460  016176 4500 00        5223 C14     STZ     CF               INDICATE STRONG COERCION
     015461  035235 7400 51        5224 C14,1   STX0    A$STACK,I        STORE XR - 0 AS COERCION COMMAND IN CONTROL STACK
     015462  035216 1670 00        5225         SBX7    T$MODE           MAKE MODE POINTER RELATIVE
     015463  035235 4470 56        5226         SXL7    A$STACK,ID       AND STORE MODE IN CONTROL STACK
     015464  005742 7170 00          5227         XED      T$SOVF          CHECK FOR STACK OVERFLOW
     015465  035235 2200 00          5228         LDX0     A$STACK         GET POINTER TO END OF CONTROL STACK
     015466  035215 1600 00          5229         SBX0     T$STACK         MAKE POINTER RELATIVE
     015467  000000 6350 10          5230         EAA      0,0             GET CONTROL STACK MARK IN AU
     015470  035234 7550 56          5231         STA      A$WORK,ID       AND STORE IN WORKING STACK
     015471  005754 7170 00          5232         XED      T$WOVF          CHECK FOR STACK OVERFLOW
     015472  035234 2350 54          5233 C15     LDA      A$WORK,DI       GET LAST CONTROL STACK MARK
     015473  000001 6210 05          5234         EAX1     1,AL            CHECK IF LAST WORD IN WORKING STACK
     015474  015525 6000 00          5235         TZE      C20             TRANSFER IF COERCION ATTEMPT FAILED
     015475  000000 6270 01          5236         EAX7     0,AU            GET STACK MARK IN XR - 7
     015476  035215 0670 00          5237         ADX7     T$STACK         MAKE STACK POINTER ABSOLUTE
     015477  777777 7270 17          5238         LXL7     -1,7            GET OLD TARGET MODE IN XR - 7
     015500  010630 7000 00          5239         TSX0     A$XFER          MAKE MODE POINTER UNIQUE AND ABSOLUTE
  END OF BINARY CARD 00000265
     015501  777777 1010 17          5240         CMPX1    -1,7            SEE IF ALL POSSIBILITIES HAVE BEEN TRIED
     015502  015514 6000 00          5241         TZE      C17             TRANSFER IF ALL POSSIBILITIES HAVE BEEN TRIED
     015503  015505 7410 00          5242         STX1     C16             STORE INDEX REGISTER TO ADD TO ANOTHER X REGISTER
     015504  000000 2260 17          5243         LDX6     0,7             GET TYPE OF MODE IN XR - 6
     015505  000000 0670 03          5244 C16     ADX7     **,DU           GET POINTER TO NEXT MODE TO BE TARGET MODE
     015506  000000 2270 17          5245         LDX7     0,7             GET NEXT TARGET MODE IN XR - 7
     015507  016202 7470 00          5246         STX7     BMODE           AND STORE AS TARGET MODE
     015510  035234 4410 56          5247         SXL1     A$WORK,ID       STORE INCREMENTED INDEX IN WORKING STACK
     015511  017007 1660 03          5248         SBX6     M$UNION,DU      SET FIRM FLAG IF UNION
     015512  016177 7460 00          5249         STX6     FF              AND STORE IN MEMORY
     015513  015323 7100 00          5250         TRA      C1              AND CONTINUE
     015514  035234 2200 54          5251 C17     LDX0     A$WORK,DI       GET CONTROL STACK MARK IN XR - 0
     015515  035234 0110 56          5252         NOP      A$WORK,ID       RESTORE WORKING STACK POINTER
     015516  035215 0600 00          5253         ADX0     T$STACK         MAKE POINTER ABSOLUTE
     015517  016202 2270 00          5254         LDX7     BMODE           GET CURRENT TARGET MODE IN XR - 7
     015520  015522 7100 00          5255         TRA      C19             GO ENTER LOOP
     015521  035235 7270 54          5256 C18     LXL7     A$STACK,DI      GET PREVIOUS TARGET MODE IN XR - 7
     015522  035235 1000 00          5257 C19     CMPX0    A$STACK         HAS STACK BEEN DELETED BACK TO THE MARK
     015523  015521 6010 00          5258         TNZ      C18             TRANSFER IF MORE TO DELETE
     015524  015472 7100 00          5259         TRA      C15             TRANSFER TO CONTINUE COERCION
     015525  016202 7470 00          5260 C20     STX7     BMODE           RESTORE BMODE FOR FAILURE RETURN
     015526  035234 2340 54          5261 C21     SZN      A$WORK,DI       DELETE A WORD FROM THE WORKING STACK
  END OF BINARY CARD 00000266
     015527  015526 6010 00          5262         TNZ      C21             LOOP IF MORE TO DELETE
     015530  000000 7100 00          5263 CX      TRA      **              AND GO TO ERROR RETURN
     015531  016176 4500 00          5264 VC      STZ      CF              INDICATE STRONG COERCION
     015532  035234 7200 54          5265 VC0     LXL0     A$WORK,DI       DELETE CONTROL WORDS FROM WORKING STACK
     015533  015532 6050 00          5266         TPL      VC0             TRANSFER IF MORE TO DELETE
     015534  035234 2200 54          5267 VC.5    LDX0     A$WORK,DI       GET LAST COERCION USED ON GIVEN VALUE
     015535  016762 1000 03          5268         CMPX0    M$REF,DU        IS IT A REFERENCE
     015536  015541 6010 00          5269         TNZ      VC1             TRANSFER IF NOT
     015537  035234 7270 51          5270         LXL7     A$WORK,I        GET REFERENCE MODE IN XR - 7
     015540  015534 7100 00          5271         TRA      VC.5            AND LOOP
     015541  035234 0110 56          5272 VC1     NOP      A$WORK,ID       RESTORE NONREF COERCION
     015542  000000 6350 17          5273         EAA      0,7             GET FINAL MODE IN AU
     015543  000022 7710 00          5274         ARL      18              MOVE TO AL
```

```
           2                                           PASS 2

     015544  016531 0750 03        5275          ADA     O$VOID,DU         ADD VOID COERCION
     015545  035235 7550 56        5276          STA     A$STACK,ID        AND ADD TO CONTROL STACK
     015546  005742 7170 00        5277          XED     T$SOVF            CHECK FOR STACK OVERFLOW
     015547  015334 7100 00        5278          TRA     C3                AND MOVE WORK TO STACK AND EXIT
     015550  016176 4500 00        5279 CSK      STZ     CF                INDICATE STRONG COERCION
     015551  016202 2350 00        5280          LDA     BMODE             GET TARGET MODE IN AU
     015552  000022 7710 00        5281          ARL     18                MOVE MODE TO AL
     015553  016567 0750 03        5282          ADA     O$SKIP,DU         ADD SKIP COERCION COMMAND
     015554  035235 7550 56        5283          STA     A$STACK,ID        AND STORE IN CONTROL STACK
   END OF BINARY CARD 00000267
     015555  005742 7170 00        5284          XED     T$SOVF            CHECK FOR STACK OVERFLOW
     015556  015370 7100 00        5285          TRA     C10               GO TO END CLEANUP
     015557  016176 4500 00        5286 CNIL     STZ     CF                INDICATE STRONG COERCION
     015560  016202 2270 00        5287          LDX7    BMODE             GET TARGET MODE IN XR - 7
     015561  010630 7000 00        5288          TSX0    A$XFER            MAKE MODE POINTER UNIQUE AND ABSOLUTE
     015562  000000 2200 17        5289          LDX0    0,7               GET TYPE OF MODE IN XR - 0
     015563  016762 1000 03        5290          CMPX0   M$REF,DU          IS IT A REFERENCE MODE
     015564  015530 6010 00        5291          TNZ     CX                GO GIVE FAILURE RETURN

     015565  016202 2350 00        5292          LDA     BMODE             GET TARGET MODE IN AU
     015566  000022 7710 00        5293          ARL     18                MOVE MODE POINTER TO AL
     015567  016572 0750 03        5294          ADA     O$NIL,DU          ADD NIL COERCION COMMAND
     015570  035235 7550 56        5295          STA     A$STACK,ID        AND STORE IN CONTROL STACK
     015571  005742 7170 00        5296          XED     T$SOVF            CHECK FOR STACK OVERFLOW
     015572  015370 7100 00        5297          TRA     C10               GO TO END CLEANUP
     015573  016176 4500 00        5298 LC       STZ     CF                INDICATE STRONG COERCION
     015574  016202 2270 00        5299          LDX7    BMODE             GET TARGET MODE IN XR - 7
     015575  016152 7000 00        5300          TSX0    CDEP              DEPROCEDURE IT
     015576  035235 7550 56        5301          STA     A$STACK,ID        STORING IN CONTROL STACK
     015577  005742 7170 00        5302          XED     T$SOVF            CHECK FOR STACK OVERFLOW
     015600  000000 6350 17        5303          EAA     0,7               GET DEPROCEDURED MODE IN AU
     015601  000022 7710 00        5304          ARL     18                MOVE MODE TO AL
     015602  016526 0750 03        5305          ADA     O$HIP,DU          ADD HIP COERCION
   END OF BINARY CARD 00000268
     015603  035235 7550 56        5306          STA     A$STACK,ID        AND ADD TO CONTROL STACK
     015604  005742 7170 00        5307          XED     T$SOVF            CHECK FOR STACK OVERFLOW
     015605  015370 7100 00        5308          TRA     C10               GO TO CLEAN UP
     015606  016176 4500 00        5309 VAC      STZ     CF                INDICATE STRONG COERCION
     015607  035235 2200 00        5310          LDX0    A$STACK           GET POINTER TO END OF CONTROL STACK
     015610  035215 1600 00        5311          SBX0    T$STACK           MAKE STACK POINTER RELATIVE
     015611  015632 7400 00        5312          STX0    VACT              AND SAVE STACK MARK IF FAILURE
     015612  016202 2270 00        5313          LDX7    BMODE             GET TARGET MODE
     015613  016152 7000 00        5314          TSX0    CDEP              AND DEPROCEDURE IT
     015614  035235 7550 56        5315          STA     A$STACK,ID        STORING IN CONTROL STACK
     015615  005742 7170 00        5316          XED     T$SOVF            CHECK FOR STACK OVERFLOW
     015616  016202 7470 00        5317          STX7    BMODE             STORE NEW TARGET MODE
     015617  000000 6350 17        5318          EAA     0,7               GET MODE IN AU
     015620  000022 7710 00        5319          ARL     18                MOVE TO AL
U    015621  000000 0750 03        5320          ADA     O$VAC,DU          ADD COERCION COMMAND
     015622  035235 7550 56        5321          STA     A$STACK,ID        AND STORE IN CONTROL STACK
     015623  005742 7170 00        5322          XED     T$SOVF            CHECK FOR STACK OVERFLOW
     015624  010630 7000 00         5323          TSX0     A$XFER           MAKE MODE ABSOLUTE
     015625  000000 2200 17         5324          LDX0     0,7              GET TYPE OF MODE IN XR - 0
     015626  016770 1000 03         5325          CMPX0    M$ROW,DU         IS IT A ROW MODE
     015627  015370 6000 00         5326          TZE      C10              TRANSFER IF YES
     015630  016776 1000 03         5327          CMPX0    M$ROWE,DU        IS IT AN END ROW MODE
   END OF BINARY CARD 00000269
     015631  015370 6000 00         5328          TZE      C10              TRANSFER IF YES
     015632  000000 2200 03         5329 VACT     LDX0     **,DU            GET PLACE WHERE STACK WAS MARKED
     015633  035215 0600 00         5330          ADX0     T$STACK          MAKE STACK POINTER ABSOLUTE
     015634  035235 7270 54         5331 VAC1     LXL7     A$STACK,DI       DELETE A WORD FROM THE CONTROL STACK
     015635  035235 1000 00         5332          CMPX0    A$STACK          SEE IF ENOUGH HAS BEEN DELETED
     015636  015634 6010 00         5333          TNZ      VAC1             TRANSFER IF MORE TO DELETE
     015637  016202 7470 00         5334          STX7     BMODE            RESTORE BMODE
     015640  015530 7100 00         5335          TRA      CX               AND GIVE ERROR RETURN
                                    5336 *
                                    5337 *        -6 RETURN
                                    5338 *        -5 STRENGTH OF COERCION ALREADY USED
                                    5339 *        -4 BMODE
                                    5340 *        -3 A$STACK-T$STACK
                                    5341 *        -2 VALP
                                    5342 *        -1 INDEX
                                    5343 *
     015641  016074 7000 00         5344 BAL      TSX0     CPUSH            PUSH CURRENT STATE IN WORKING STACK
     015642  015641 0750 03         5345          ADA      BAL,DU           ADD BALANCE COMMAND TO COUNT
     015643  035235 7550 56         5346          STA      A$STACK,ID       AND STORE IN CONTROL STACK
     015644  005742 7170 00         5347          XED      T$SOVF           CHECK FOR STACK OVERFLOW
     015645  016202 2350 00         5348          LDA      BMODE            GET TARGET MODE IN AU
     015646  000022 7710 00         5349          ARL      18               MOVE TARGET MODE TO AL
     015647  016452 0750 03         5350          ADA      MODE,DU          ADD MODE COMMAND
     015650  035235 7550 56         5351          STA      A$STACK,ID       AND STORE IN CONTROL STACK
     015651  005742 7170 00         5352          XED      T$SOVF           CHECK FOR STACK OVERFLOW
     015652  016446 2350 00         5353          LDA      VALP             GET POINTER TO BALANCE CONTROL BLOCK
     015653  016200 1750 00         5354          SBA      CW               MAKE POINTER RELATIVE TO A$WORK
     015654  000022 7710 00         5355          ARL      18               MOVE VALUE POINTER TO AL
     015655  016446 0750 03         5356          ADA      VALP,DU          ADD VALP CONTROL WORD
     015656  035235 7550 56         5357          STA      A$STACK,ID       AND STORE IN CONTROL STACK
   END OF BINARY CARD 00000270
     015657  005742 7170 00         5358          XED      T$SOVF           CHECK FOR STACK OVERFLOW
     015660  777773 2200 03         5359          LDX0     -5,DU            PREPARE TO DECREMENT VALP
     015661  016446 0400 00         5360          ASX0     VALP             DECREMENT TO CONSIDER NEXT VALUE
     015662  015261 7000 00         5361 BAL1     TSX0     C0               COERCE VALUE TO TARGET MODE OR FAIL
     015663  015704 7100 00         5362          TRA      BAL3             TRANSFER IF COERCION FAILED
     015664  035234 2210 00         5363          LDX1     A$WORK           GET POINTER TO TOP OF WORKING STACK
     015665  016176 2350 00         5364          LDA      CF               GET STRENGTH REQUIRED IN LAST COERCION
     015666  777773 2550 11         5365          ORSA     -5,1             UPDATE STRENGTH REQUIRED FOR BALANCE
     015667  777774 2270 11         5366          LDX7     -4,1             GET ORIGINAL TARGET MODE
     015670  016202 7470 00         5367          STX7     BMODE            AND MAKE IT CURRENT TARGET MODE
     015671  777777 0540 11         5368          AOS      -1,1             STEP INDEX BY ONE
     015672  015662 6010 00         5369          TNZ      BAL1             TRANSFER IF ANOTHER BALANCED VALUE TO COERCE
     015673  777773 2350 11         5370 BAL2,    LDA      -5,1             GET STRENGTH REQUIRED TO DO COERCION
```

```
          2                                          PASS 2

     015674  016176 7550 00        5371          STA     CF              AND STORE AS RESULTING STRENGTH
     015675  035234 0110 54        5372          NOP     A$WORK,DI       DELETE A WORD FROM THE WORKING STACK
     015676  035234 0110 54        5373          NOP     A$WORK,DI       DELETE A WORD FROM THE WORKING STACK
     015677  035234 0110 54        5374          NOP     A$WORK,DI       DELETE A WORD FROM THE WORKING STACK
     015700  035234 0110 54        5375          NOP     A$WORK,DI       DELETE A WORD FROM THE WORKING STACK
     015701  035234 0110 54        5376          NOP     A$WORK,DI       DELETE A WORD FROM THE WORKING STACK
     015702  035234 2200 54        5377          LDX0    A$WORK,DI       GET RETURN ADDRESS IN XR - 0
     015703  000001 7100 10        5378          TRA     1,0             AND GIVE SUCCESSFUL RETURN
     015704  035234 2200 00        5379 BAL3     LDX0    A$WORK          GET POINTER TO END OF WORKING STACK
   END OF BINARY CARD 00000271
     015705  777776 2210 10        5380          LDX1    -2,0            GET ORIGINAL VALUE CONTROL BLOCK POINTER
     015706  016446 7410 00        5381          STX1    VALP            AND MAKE IT CURRENT VALUE POINTER
     015707  777774 2210 10        5382          LDX1    -4,0            GET ORIGINAL TARGET MODE
     015710  016202 7410 00        5383          STX1    BMODE           AND MAKE IT CURRENT TARGET MODE
     015711  777775 2210 10        5384          LDX1    -3,0            GET PLACE WHERE STACK WAS MARKED
     015712  035215 0610 00        5385          ADX1    T$STACK         MAKE POINTER ABSOLUTE
     015713  035235 0110 54        5386 BAL4     NOP     A$STACK,DI      DELETE A WORD FROM THE CONTROL STACK
     015714  035235 1010 00        5387          CMPX1   A$STACK         IS STACK DELETED BACK TO MARK
     015715  015713 6010 00        5388          TNZ     BAL4            TRANSFER IF MORE TO DELETE
     015716  035234 0110 54        5389          NOP     A$WORK,DI       DELETE A WORD FROM THE WORKING STACK
     015717  035234 0110 54        5390          NOP     A$WORK,DI       DELETE A WORD FROM THE WORKING STACK
     015720  035234 0110 54        5391          NOP     A$WORK,DI       DELETE A WORD FROM THE WORKING STACK
     015721  035234 0110 54        5392          NOP     A$WORK,DI       DELETE A WORD FROM THE WORKING STACK
     015722  035234 0110 54        5393          NOP     A$WORK,DI       DELETE A WORD FROM THE WORKING STACK
     015723  035234 2200 54        5394          LDX0    A$WORK,DI       GET RETURN ADDRESS IN XR - 0
     015724  000000 7100 10        5395          TRA     0,0             AND GIVE FAILURE RETURN
     015725  016074 7000 00        5396 PAR      TSX0    CPUSH           PUSH CURRENT STATE IN WORKING STACK
     015726  016202 2270 00        5397          LDX7    BMODE           GET TARGET MODE FOR DISPLAY
     015727  016152 7000 00        5398          TSX0    CDEP            DEPROCEDURE TARGET MODE
     015730  035235 7550 56        5399          STA     A$STACK,ID      STORING IN CONTROL STACK
     015731  005742 7170 00        5400          XED     T$SOVF          CHECK FOR STACK OVERFLOW
     015732  016202 7470 00        5401          STX7    BMODE
   END OF BINARY CARD 00000272
     015733  035234 2210 00        5402          LDX1    A$WORK          GET POINTER TO END OF WORKING STACK
     015734  777774 4470 11        5403          SXL7    -4,1            STORE REDUCED MODE IN SAVED STATE
     015735  010630 7000 00        5404          TSX0    A$XFER          MAKE MODE POINTER ABSOLUTE
     015736  000000 2200 17        5405          LDX0    0,7             GET TYPE OF MODE IN XR - 0
     015737  017007 1000 03        5406          CMPX0   M$UNION,DU      IS IT A UNION MODE
     015740  015773 6010 00        5407          TNZ     PAR3            NO - DO DISPLAY COERCION
     015741  777777 2200 17        5408          LDX0    -1,7            GET LENGTH OF UNION MODE IN XR - 1
     015742  000001 1600 03        5409          SBX0    1,DU            DECREASE IT BY ONE FOR HEADER
     015743  777777 7400 11        5410          STX0    -1,1            AND STORE IN INDEX LOCATION IN SAVED STATE
     015744  035216 1670 00        5411          SBX7    T$MODE          MAKE MODE POINTER RELATIVE
     015745  000000 6350 17        5412          EAA     0,7             GET TARGET MODE IN AU
     015746  000022 7710 00        5413          ARL     18              MOVE IT TO AL
     015747  017007 0750 03        5414          ADA     M$UNION,DU      ADD UNION COERCION COMMAND
     015750  035235 7550 56        5415          STA     A$STACK,ID      AND STORE IN CONTROL STACK
     015751  005742 7170 00        5416          XED     T$SOVF          CHECK FOR STACK OVERFLOW
     015752  777774 7270 11        5417 PAR1     LXL7    -4,1            GET TARGET MODE IN XR - 7
     015753  010630 7000 00        5418          TSX0    A$XFER          MAKE IT UNIQUE AND ABSOLUTE
```

```
      015754  777777 0670 11        5419        ADX7    -1,1            ADD CURRENT INDEX TO GET MODE POINTER
      015755  000000 2270 17        5420        LDX7    0,7             GET CURRENT TARGET IN XR - 7
      015756  016202 7470 00        5421        STX7    BMODE           AND STORE AS CURRENT TARGET MODE
      015757  015261 7000 00        5422        TSX0    C0              COERCE VALUE TO TARGET MODE
      015760  015766 7100 00        5423        TRA     PAR2            TRANSFER IF FAILURE
   END OF BINARY CARD 00000273
      015761  035234 2210 00        5424        LDX1    A$WORK          GET POINTER TO END OF WORKING STACK
      015762  016176 2350 00        5425        LDA     CF              GET COERCION STRENGTH REQUIRED
      015763  015766 6000 00        5426        TZE     PAR2            TRANSFER IF IT WAS STRONG
      015764  777773 7550 11        5427        STA     -5,1            AND STORE IN SAVED STATE
      015765  015673 7100 00        5428        TRA     BAL2            TRANSFER TO SUCCESS CLEANUP
      015766  035234 2210 00        5429 PAR2   LDX1    A$WORK          GET POINTER TO END OF WORKING STACK
      015767  777777 2200 03        5430        LDX0    -1,DU           GET MINUS ONE FOR DECREMENTING
      015770  777777 0400 11        5431        ASX0    -1,1            DECREMENT INDEX IN SAVED STATE
      015771  015752 6010 00        5432        TNZ     PAR1            TRANSFER IF MORE MODES TO CONSIDER
      015772  015704 7100 00        5433        TRA     BAL3            GO TO FAILURE EXIT CLEANUP
      015773  035216 1670 00        5434 PAR3   SBX7    T$MODE          MAKE MODE POINTER RELATIVE
      015774  777777 3350 11        5435        LCA     -1,1            GET NUMBER OF ELEMENTS IN DISPLAY IN AL
      015775  015725 0750 03        5436        ADA     PAR,DU          ADD PARALLEL COMMAND
      015776  035235 7550 56        5437        STA     A$STACK,ID      AND STORE IN CONTROL STACK
      015777  005742 7170 00        5438        XED     T$SOVF          CHECK FOR STACK OVERFLOW
      016000  000000 6350 17        5439        EAA     0,7             GET TARGET MODE FOR DISPLAY IN AU
      016001  000022 7710 00        5440        ARL     18              MOVE TARGET MODE TO AL
      016002  016452 0750 03        5441        ADA     MODE,DU         ADD SET MODE COMMAND
      016003  035235 7550 56        5442        STA     A$STACK,ID      AND STORE IN CONTROL STACK
      016004  005742 7170 00        5443        XED     T$SOVF          CHECK FOR STACK OVERFLOW
      016005  016446 2350 00        5444        LDA     VALP            GET POINTER TO CURRENT DISPLAY VALUE BLOCK
      016006  016200 1750 00        5445        SBA     CW              MAKE POINTER RELATIVE TO A$WORK
   END OF BINARY CARD 00000274
      016007  000022 7710 00        5446        ARL     18              MOVE POINTER TO AL
      016010  016446 0750 03        5447        ADA     VALP,DU         ADD SET VALUE POINTER COMMAND
      016011  035235 7550 56        5448        STA     A$STACK,ID      AND STORE IN CONTROL STACK
      016012  005742 7170 00        5449        XED     T$SOVF          CHECK FOR STACK OVERFLOW
      016013  777773 2200 03        5450        LDX0    -5,DU           GET LENGTH OF VALUE CONTROL BLOCK IN XR - 0
      016014  016446 0400 00        5451        ASX0    VALP            STEP TO NEXT VALUE CONTROL BLOCK
      016015  010630 7000 00        5452        TSX0    A$XFER          MAKE MODE POINTER ABSOLUTE
      016016  000000 2200 17        5453        LDX0    0,7             GET TYPE OF MODE IN XR - 0
      016017  016770 1000 03        5454        CMPX0   M$ROW,DU        IS IT A ROW MODE
      016020  016050 6000 00        5455        TZE     PAR7            TRANSFER IF ROW MODE
      016021  016776 1000 03        5456        CMPX0   M$ROWE,DU       IS IT AN END ROW MODE
      016022  016050 6000 00        5457        TZE     PAR7            TRANSFER IF END ROW MODE
      016023  777777 2220 17        5458        LDX2    -1,7            GET LENGTH OF MODE IN XR - 2
      016024  016757 1000 03        5459        CMPX0   M$STRCT,DU      IS IT A STRUCTURE MODE
      016025  016030 6010 00        5460        TNZ     PAR4            TRANSFER IF NOT STRUCTURE
      016026  000001 1620 03        5461        SBX2    1,DU            GET NUMBER OF FIELDS IN STRUCTURE IN XR - 2
      016027  016033 7100 00        5462        TRA     PAR5            AND CONTINUE
      016030  017001 1000 03        5463 PAR4   CMPX0   M$PROC,DU       IS IT A PROCEDURE MODE
      016031  015704 6010 00        5464        TNZ     BAL3            GO TO FAILURE CLEANUP
      016032  000002 1620 03        5465        SBX2    2,DU            GET NUMBER OF ARGUMENTS IN PROG IN XR - 2
      016033  000000 6350 12        5466 PAR5   EAA     0,2             GET NUMBER OF VALUES TO COERCE IN AU
```

```
     016034  000022 7710 00          5467         ARL      18                  MOVE TO AL
  END OF BINARY CARD 00000275
     016035  035234 2210 00          5468         LDX1     A$WORK              GET POINTER TO END OF WORKING STACK IN XR - 1
     016036  777777 0750 11          5469         ADA      -1,1                ADD MINUS NUMBER OF VALUES IN DISPLAY
     016037  015704 6010 00          5470         TNZ      BAL3                NOT EQUAL SO GO TO FAILURE CLEANUP
     016040  777777 7420 11          5471         STX2     -1,1                STORE VALUE INDEX IN SAVED STATE
     016041  777774 7270 11          5472 PAR6    LXL7     -4,1                GET TARGET MODE OF DISPLAY IN XR - 7
     016042  010630 7000 00          5473         TSX0     A$XFER              MAKE MODE POINTER ABSOLUTE
     016043  000000 2200 17          5474         LDX0     0,7                 GET TYPE OF MODE IN XR - 0
     016044  016770 1000 03          5475         CMPX0    M$ROW,DU            IS IT A ROW MODE
     016045  016050 6000 00          5476         TZE      PAR7                TRANSFER IF SO
     016046  016776 1000 03          5477         CMPX0    M$ROWE,DU           IS IT AN END ROW MODE
     016047  016052 6010 00          5478         TNZ      PAR8                TRANSFER IF NOT
     016050  000001 2270 17          5479 PAR7    LDX7     1,7                 GET TARGET MODE FOR ROW ELEMENT
     016051  016054 7100 00          5480         TRA      PAR9                AND CONTINUE
     016052  777777 0670 11          5481 PAR8    ADX7     -1,1                GET MODE OF ARGUMENT OR FIELD FOR NEW TARGET
     016053  000000 2270 17          5482         LDX7     0,7                 GET TARGET IN XR - 7
     016054  016202 7470 00          5483 PAR9    STX7     BMODE               STORE NEW TARGET MODE
     016055  015261 7000 00          5484         TSX0     C0                  AND COERCE DISPLAY ELEMENT
     016056  015704 7100 00          5485         TRA      BAL3                TRANSFER IF FAILURE TO FAILURE CLEANUP
     016057  035234 2210 00          5486         LDX1     A$WORK              GET POINTER TO END OF WORKING STACK
     016060  777777 2200 03          5487         LDX0     -1,DU               GET DECREMENTATION CONSTANT IN XR - 0
     016061  777777 0400 11          5488         ASX0     -1,1                DECREMENT DISPLAY INDEX IN SAVED STATE
     016062  016041 6010 00          5489         TNZ      PAR6                TRANSFER IF MORE ELEMENTS TO COERCE
  END OF BINARY CARD 00000276
     016063  777774 7270 11          5490         LXL7     -4,1                GET TARGET MODE FOR DISPLAY IN XR - 7
     016064  010630 7000 00          5491         TSX0     A$XFER              MAKE MODE POINTER ABSOLUTE
     016065  000000 2200 17          5492         LDX0     0,7                 GET TYPE OF MODE IN XR - 0
     016066  016770 1000 03          5493         CMPX0    M$ROW,DU            IS IT A ROW DISPLAY
     016067  015673 6000 00          5494         TZE      BAL2,               TRANSFER IF YES TO SUCCESS CLEANUP
     016070  016776 1000 03          5495         CMPX0    M$ROWE,DU           IS IT AN END ROW DISPLAY
     016071  015673 6000 00          5496         TZE      BAL2,               TRANSFER IF YES TO SUCCESS CLEANUP
     016072  777773 4500 11          5497         STZ      -5,1                SET REQUIRED COERCION STRENGTH TO STRONG
     016073  015673 7100 00          5498         TRA      BAL2,               TRANSFER IF SUCCESSFUL TO SUCCESS CLEANUP
     016074  016123 7400 00          5499 CPUSH   STX0     CPX                 SAVE RETURN
     016075  015530 2200 00          5500         LDX0     CX                  GET RETURN ADDRESS
     016076  035234 7400 56          5501         STX0     A$WORK,ID           AND STORE IN WORKING STACK
     016077  005754 7170 00          5502         XED      T$WOVF              CHECK FOR STACK OVERFLOW
     016100  035234 4500 56          5503         STZ      A$WORK,ID           STORE ZERO FOR STRENGTH OF COERCION
     016101  005754 7170 00          5504         XED      T$WOVF              CHECK FOR STACK OVERFLOW
     016102  016202 2350 00          5505         LDA      BMODE               GET TARGET MODE IN AU
     016103  035234 7550 56          5506         STA      A$WORK,ID           AND STORE IN WORKING STACK
     016104  005754 7170 00          5507         XED      T$WOVF              CHECK FOR STACK OVERFLOW
     016105  035235 2200 00          5508         LDX0     A$STACK             GET POINTER TO END OF CONTROL STACK
     016106  035215 1600 00          5509         SBX0     T$STACK             MAKE CONTROL STACK POINTER RELATIVE
     016107  035234 7400 56          5510         STX0     A$WORK,ID           AND STORE IN WORKING STACK
     016110  005754 7170 00          5511         XED      T$WOVF              CHECK FOR STACK OVERFLOW
  END OF BINARY CARD 00000277
     016111  016446 2200 00          5512         LDX0     VALP                GET POINTER TO VALUE CONTROL ELEMENT
     016112  035234 7400 56          5513         STX0     A$WORK,ID           AND STORE IN CONTROL STACK
     016113  005754 7170 00         5514          XED       T$WOVF            CHECK FOR STACK OVERFLOW
     016114  035214 0600 00         5515          ADX0      T$WORK            MAKE POINTER ABSOLUTE
     016115  000000 2350 10         5516          LDA       0,0               GET MULTIPLE VALUE COUNT IN AL
     016116  777777 3750 07         5517          ANA       -1,DL             ZERO OUT AU
     016117  000000 5310 00         5518          NEG                         GET MINUS ELEMENT COUNT IN A
     016120  035234 7550 56         5519          STA       A$WORK,ID         AND STORE IN WORKING STACK
     016121  005754 7170 00         5520          XED       T$WOVF            CHECK FOR STACK OVERFLOW
     016122  000000 5310 00         5521          NEG                         NEGATE COUNT AGAIN AS RESULT
     016123  000000 7100 00         5522 CPX      TRA       **                AND RETURN
     016124  000000 6210 10         5523 CRED     EAX1      0,0               SAVE RETURN IN XR - 1
     016125  010630 7000 00         5524 CRED1    TSX0      A$XFER            MAKE MODE POINTER ABSOLUTE
     016126  000000 6260 17         5525          EAX6      0,7               SAVE CURRENT MODE IN XR - 6
     016127  035216 1660 00         5526          SBX6      T$MODE            MAKE SAVED MODE POINTER RELATIVE
     016130  000000 2220 17         5527          LDX2      0,7               GET TYPE OF MODE IN XR - 2
     016131  016762 1020 03         5528          CMPX2     M$REF,DU          IS IT A REFERENCE MODE
     016132  016140 6000 00         5529          TZE       CRED2             TRANSFER IF REFERENCE MODE
     016133  017001 1020 03         5530          CMPX2     M$PROC,DU         IS IT A PROCEDURE MODE
     016134  016147 6010 00         5531          TNZ       CRED3             TRANSFER IF NOT PROCEDURE MODE
     016135  777777 2230 17         5532          LDX3      -1,7              GET NUMBER OF ARGUMENTS + 2 IN XR - 3
     016136  000002 1030 03         5533          CMPX3     2,DU              SEE IF PROCEDURE WITHOUT ARGUMENTS
   END OF BINARY CARD 00000278
     016137  016147 6010 00         5534          TNZ       CRED3             TRANSFER IF ANY ARGUMENTS
     016140  016151 7420 00         5535 CRED2    STX2      REDT              STORE TYPE = COERCION COMMAND IN TEMP
     016141  016151 4460 00         5536          SXL6      REDT              STORE MODE IN TEMP,DL
     016142  016151 2350 00         5537          LDA       REDT              GET COERCION COMMAND IN A
     016143  000000 7160 11         5538          XEC       0,1               STORE COERCION IN STACK
     016144  000001 7160 11         5539          XEC       1,1               CHECK FOR STACK OVERFLOW
     016145  000001 2270 17         5540          LDX7      1,7               GET REDUCED MODE IN XR - 7
     016146  016125 7100 00         5541          TRA       CRED1             AND LOOP
     016147  000000 6270 16         5542 CRED3    EAX7      0,6               GET FINAL MODE IN XR - 7
     016150  000002 7100 11         5543          TRA       2,1               AND RETURN
     016151  000000 000000          5544 REDT     ZERO
     016152  000000 6210 10         5545 CDEP     EAX1      0,0               SAVE RETURN IN XR - 1
     016153  010630 7000 00         5546 CDEP1    TSX0      A$XFER            MAKE MODE POINTER ABSOLUTE
     016154  000000 6260 17         5547          EAX6      0,7               SAVE CURRENT MODE IN XR - 6
     016155  035216 1660 00         5548          SBX6      T$MODE            MAKE SAVED MODE POINTER RELATIVE
     016156  000000 2220 17         5549          LDX2      0,7               GET TYPE OF MODE IN XR - 2
     016157  017001 1020 03         5550          CMPX2     M$PROC,DU         IS IT A PROCEDURE MODE
     016160  016173 6010 00         5551          TNZ       CDEP2             TRANSFER IF NOT PROCEDURE MODE
     016161  777777 2230 17         5552          LDX3      -1,7              GET NUMBER OF ARGUMENTS + 2 IN XR - 3
     016162  000002 1030 03         5553          CMPX3     2,DU              SEE IF PROCEDURE WITHOUT ARGUMENTS
     016163  016173 6010 00         5554          TNZ       CDEP2             TRANSFER IF ANY ARGUMENTS
     016164  016175 7420 00         5555          STX2      DEPT              STORE TYPE = COERCION COMMAND IN TEMP
   END OF BINARY CARD 00000279
     016165  016175 4460 00         5556          SXL6      DEPT              STORE MODE IN TEMP,DL
     016166  016175 2350 00         5557          LDA       DEPT              GET COERCION COMMAND IN A
     016167  000000 7160 11         5558          XEC       0,1               STORE COERCION IN STACK
     016170  000001 7160 11         5559          XEC       1,1               CHECK FOR STACK OVERFLOW
     016171  000001 2270 17         5560          LDX7      1,7               GET REDUCED MODE IN XR - 7
     016172  016153 7100 00         5561          TRA       CDEP1             AND LOOP
```

```
          2                                                    PASS 2

     016173  000000 6270 16         5562 CDEP2  EAX7     0,6                 GET FINAL MODE IN XR - 7
     016174  000002 7100 11         5563        TRA      2,1                 AND RETURN
     016175  000000 000000          5564 DEPT   ZERO
     016176  000000 000000          5565 CF     ZERO
     016177  000000 000000          5566 FF     ZERO
     016200  000000 000000          5567 CW     ZERO
     016201  000000 000000          5568 AMODE  ZERO
     016202  000000 000000          5569 BMODE  ZERO
     016203  000000 000000          5570 TMODE  ZERO
     016204  016206 7400 00         5571 DC     STX0     DCX                 SAVE RETURN
     016205  035235 2220 54         5572 DCL    LDX2     A$STACK,DI          GET NEXT COERCION TO APPLY IN XR - 2
     016206  000000 6000 00         5573 DCX    TZE      **                  EXIT IF THERE ARE NO MORE COERCIONS
     016207  016214 2210 03         5574        LDX1     DCTB,DU             GET POINTER TO START OF COERCION TABLE IN XR - 1
     016210  042300 5202 01         5575        RPT      DCTBE-DCTB,1,TZE SEARCH COERCION TABLE
     016211  000000 1020 11         5576        CMPX2    0,1                 FOR PROPER ROUTINE TO HANDLE COERCION
     016212  777777 2350 11         5577        LDA      -1,1                GET COERCION TABLE WORD IN A
   END OF BINARY CARD 00000280
     016213  000000 7100 05         5578        TRA      0,AL                AND TRANSFER TO ROUTINE
     016214  017001 016300          5579 DCTB   ZERO     MSPROC,DC3
     016215  017004 016243          5580        ZERO     MSDEPR,DC1
     016216  016765 016243          5581        ZERO     MSDEREF,DC1
U    016217  000000 016243          5582        ZERO     MSREFRW,DC1
     016220  017007 016243          5583        ZERO     MSUNION,DC1
     016221  016754 016243          5584        ZERO     MSPRIM,DC1
     016222  016770 016243          5585        ZERO     MSROW,DC1
     016223  016531 016243          5586        ZERO     OSVOID,DC1
     016224  016567 016263          5587        ZERO     OSSKIP,DC2
     016225  016572 016263          5588        ZERO     OSNIL,DC2
     016226  016526 016235          5589        ZERO     OSHIP,DC0
     016227  010000 016263          5590        ZERO     WSVAC,DC2
     016230  016446 016303          5591        ZERO     VALP,DC4
     016231  016452 016306          5592        ZERO     MODE,DC5
     016232  015641 016313          5593        ZERO     BAL,DBAL
     016233  015725 016377          5594        ZERO     PAR,DPAR
     016234  000000 777777          5595        ZERO     0,SERROR
                         016235     5596 DCTBE  EQU      *
     016235  016446 2210 00         5597 DC0    LDX1     VALP                GET POINTER TO VALUE CONTROL BLOCK

     016236  035234 0610 00         5598        ADX1     A$WORK              MAKE POINTER ABSOLUTE
     016237  000002 2220 11         5599        LDX2     2,1                 GET POINTER TO CORRESPONDING CODE
     016240  035224 0620 00         5600        ADX2     TSCODE              MAKE POINTER ABSOLUTE
   END OF BINARY CARD 00000281
     016241  016523 2200 03         5601        LDX0     OSGOTO,DU           GET GOTO CODE
     016242  000000 7400 12         5602        STX0     0,2                 AND STORE OVER LABEL'S IDENT CODE
     016243  035235 2350 51         5603 DC1    LDA      A$STACK,I           GET COERCION IN A REGISTER
     016244  012375 7000 00         5604        TSX0     ADD                 AND ADD AFTER CODE FOR VALUE
     016245  016446 2210 00         5605        LDX1     VALP                GET POINTER TO VALUE CONTROL BLOCK
     016246  035234 0610 00         5606        ADX1     A$WORK              MAKE POINTER ABSOLUTE
     016247  000002 0540 11         5607        AOS      2,1                 INCREMENT LENGTH OF VALUE TO INCLUDE COERCION
     016250  035235 7270 51         5608        LXL7     A$STACK,I           GET MODE OF RESULT IN XR - 7
     016251  000001 7470 11         5609        STX7     1,1                 AND SET NEW MODE OF VALUE IN CONTROL BLOCK
```

```
              2                                  PASS 2

     016252  035235 2220 51        5610        LDX2    A$STACK,I         GET TYPE OF COERCION IN XR - 2
     016253  017004 1020 03        5611        CMPX2   M$DEPR,DU         SEE IF DEPROCEDURING
     016254  016257 6000 00        5612        TZE     DC1,1             TRANSFER IF DEPROCEDURING
     016255  016765 1020 03        5613        CMPX2   M$DEREF,DU        SEE IF DEREFERENCING
     016256  016205 6010 00        5614        TNZ     DCL               TRANSFER IF NEITHER TO LOOP
     016257  010630 7000 00        5615 DC1,1  TSX0    A$XFER            MAKE ORIGINAL MODE POINTER ABSOLUTE
     016260  000001 2270 17        5616        LDX7    1,7               GET DEREFERENCED OR DEPROCEDURED MODE IN XR - 7
     016261  000001 7470 11        5617        STX7    1,1               AND STORE RESULT MODE IN VALUE CONTROL BLOCK
     016262  016205 7100 00        5618        TRA     DCL               AND LOOP
     016263  016446 2210 00        5619 DC2    LDX1    VALP              GET POINTER TO VALUE CONTROL BLOCK
     016264  035234 0610 00        5620        ADX1    A$WORK            MAKE POINTER ABSOLUTE
     016265  400000 2220 03        5621        LDX2    W$VALUE,DU        GET NEW HEADER FOR CONTROL BLOCK
     016266  000000 7420 11        5622        STX2    0,1               AND STORE IN VALUE CONTROL BLOCK
 END OF BINARY CARD 00000282
     016267  000002 2220 11        5623        LDX2    2,1               GET POINTER TO CORESPONDING CODE
     016270  035224 0620 00        5624        ADX2    T$CODE            MAKE CODE POINTER ABSOLUTE
     016271  035235 2200 51        5625        LDX0    A$STACK,I         GET TYPE OF COERCION IN XR - 0
     016272  000000 1000 12        5626        CMPX0   0,2               SEE IF SAME TYPE OF WORD IN CODE
     016273  777777 6010 00        5627        TNZ     $ERROR            NO - COMPILER ERROR
     016274  035235 7270 51        5628        LXL7    A$STACK,I         GET MODE RESULTING FROM COERCION
     016275  000000 4470 12        5629        SXL7    0,2               AND STORE IN CODE WORD
     016276  000001 7470 11        5630        STX7    1,1               ALSO STORE AS MODE OF CONTROL BLOCK
     016277  016205 7100 00        5631        TRA     DCL               AND LOOP
     016300  035235 7270 51        5632 DC3    LXL7    A$STACK,I         GET MODE OF PROCEDURE IN XR - 7
     016301  014530 7000 00        5633        TSX0    PCDR              MAKE VALUE INTO A PROCEDURE
     016302  016205 7100 00        5634        TRA     DCL               AND LOOP
     016303  035235 7210 51        5635 DC4    LXL1    A$STACK,I         GET VALUE CONTROL BLOCK POINTER IN XR - 1
     016304  016446 7410 00        5636        STX1    VALP              AND STORE AS POINTER TO CURRENT CONTROL BLOCK
     016305  016205 7100 00        5637        TRA     DCL               AND LOOP
     016306  016446 2210 00        5638 DC5    LDX1    VALP              GET POINTER TO CURRENT VALUE CONTROL BLOCK
     016307  035234 0610 00        5639        ADX1    A$WORK            MAKE POINTER ABSOLUTE
     016310  035235 7270 51        5640        LXL7    A$STACK,I         GET MODE OF VALUE AFTER COERCING
     016311  000001 7470 11        5641        STX7    1,1               AND STORE IN VALUE CONTROL BLOCK
     016312  016205 7100 00        5642        TRA     DCL               AND LOOP
     016313  016446 2210 00        5643 DBAL   LDX1    VALP              GET POINTER TO CURRENT VALUE CONTROL BLOCK
     016314  016447 7410 00        5644        STX1    VALP1             AND SAVE
 END OF BINARY CARD 00000283
     016315  035234 0610 00        5645        ADX1    A$WORK            MAKE POINTER ABSOLUTE
     016316  000001 7200 11        5646        LXL0    1,1               GET RANGE INFORMATION OR ZERO IN XR - 0
     016317  016376 4400 00        5647        SXL0    DBALT             AND SAVE
     016320  000000 2350 11        5648        LDA     0,1               GET NUMBER OF VALUES IN BALANCED VALUE
     016321  777777 3750 07        5649        ANA     =1,DL             ZERO OUT AU
     016322  000000 5310 00        5650        NEG                       GET NEGATIVE NUMBER FOR COUNTER
     016323  016375 7550 00        5651        STA     DBALC             AND STORE COUNT
     016324  000005 1610 03        5652        SBX1    5,DU              STEP TO PREVIOUS VALUE CONTROL BLOCK
     016325  035234 1610 00        5653        SBX1    A$WORK            MAKE POINTER RELATIVE
     016326  016446 7410 00        5654        STX1    VALP              AND STORE NEW POINTER TO VALUE CONTROL BLOCK
     016327  016375 0540 00        5655 DBAL1  AOS     DBALC             DECREMENT COUNTER
     016330  016345 6000 00        5656        TZE     DBAL2             TRANSFER IF LAST VALUE IN BALANCE
     016331  016446 2210 00        5657        LDX1    VALP              GET POINTER TO CURRENT VALUE CONTROL BLOCK
```

```
              2                                            PASS 2

      016332  000005 1610 03       5658        SBX1     5,DU           GET POINTER TO PREVIOUS VALUE CONTROL BLOCK
      016333  016446 7410 00       5659        STX1     VALP           AND RESTORE POINTER
      016334  016464 2350 00       5660        LDA      GLBL           GET A UNIQUE LABEL
      016335  777777 3750 07       5661        ANA      -1,DL          ZERO OUT AU
      016336  016507 0750 03       5662        ADA      OSJUMP,DU      ADD JUMP COMMAND
      016337  012375 7000 00       5663        TSX0     ADD            AND ADD TRANSFER COMMAND AFTER VALUE
      016340  016647 6350 00       5664        EAA      OSDELV         GET DELETE VALUE COMMAND
      016341  012375 7000 00       5665        TSX0     ADD            AND ADD AFTER CURRENT VALUE

      016342  017031 6350 00       5666        EAA      OSMA           GET MAKE AVAILABLE VALUE COMMAND
   END OF BINARY CARD 00000284
      016343  012375 7000 00       5667        TSX0     ADD            AND ADD AFTER CURRENT VALUE
      016344  016327 7100 00       5668        TRA      DBAL1          AND LOOP
      016345  016447 2210 00       5669 DBAL2  LDX1     VALP1          GET POINTER TO CURRENT VALUE CONTROL BLOCK
      016346  016446 7410 00       5670        STX1     VALP           RESTORE VALP
      016347  035234 0610 00       5671        ADX1     A$WORK         MAKE POINTER ABSOLUTE
      016350  012513 7000 00       5672        TSX0     DELWW          COMBINE BALANCED VALUES INTO A SINGLE VALUE
      016351  016464 2350 00       5673        LDA      GLBL           GET LABEL OF GENERATED TRANSFERS
      016352  016464 0540 00       5674        AOS      GLBL           INCREMENT LABEL GENERATOR
      016353  777777 3750 07       5675        ANA      -1,DL          ZERO OUT AU
      016354  016504 0750 03       5676        ADA      OSLBL,DU       ADD LABEL DEFINITION COMMAND
      016355  012375 7000 00       5677        TSX0     ADD            AND ADD AFTER VALUE
      016356  017031 6350 00       5678        EAA      OSMA           GET MAKE AVAILABLE VALUE COMMAND
      016357  012375 7000 00       5679        TSX0     ADD            AND ADD AFTER CURRENT VALUE
      016360  016446 2210 00       5680        LDX1     VALP           GET POINTER TO CURRENT VALUE CONTROL BLOCK
      016361  035234 0610 00       5681        ADX1     A$WORK         MAKE POINTER ABSOLUTE
      016362  000002 2350 07       5682        LDA      2,DL           GET A 2 TO UPDATE VALUE LENGTH
      016363  000002 0550 11       5683        ASA      2,1            MAKE VALUE INCLUDE LABEL AND MA COMMAND
      016364  016376 7210 00       5684        LXL1     DBALT          GET POSSIBLE RANGE NUMBER IN XR - 1
      016365  016205 6000 00       5685        TZE      DCL            LOOP IF NO RANGE INFORMATION
      016366  016376 2350 00       5686        LDA      DBALT          GET RANGE NUMBER IN AL
      016367  016652 0750 03       5687        ADA      OSSRNGE,DU     ADD SRNGE COMMAND
      016370  012403 7000 00       5688        TSX0     INS            AND INSERT IN FRONT OF BALANCED VALUE
   END OF BINARY CARD 00000285
      016371  016376 2350 00       5689        LDA      DBALT          GET RANGE NUMBER IN AL
      016372  016655 0750 03       5690        ADA      OSERNGE,DU     ADD ERNGE COMMAND
      016373  012375 7000 00       5691        TSX0     ADD            AND ADD AFTER BALANCED VALUE
      016374  016440 7100 00       5692        TRA      DPAR2          GO UPDATE LOC/LEN WORD IN VALUE CONTROL BLOCK
      016375  000000 000000        5693 DBALC  ZERO
      016376  000000 000000        5694 DBALT  ZERO
      016377  016446 2210 00       5695 DPAR   LDX1     VALP           GET POINTER TO VALUE CONTROL BLOCK
      016400  016447 7410 00       5696        STX1     VALP1          AND SAVE
      016401  035234 0610 00       5697        ADX1     A$WORK         MAKE POINTER ABSOLUTE
      016402  000000 2350 11       5698        LDA      0,1            GET NUMBER OF VALUES IN PARALLEL DISPLAY IN AL
      016403  777777 3750 07       5699        ANA      -1,DL          ZERO OUT AU
      016404  000000 5310 00       5700        NEG                     MAKE ELEMENT COUNT NEGATIVE
      016405  016445 7550 00       5701        STA      DPART          AND STORE AS COUNT
      016406  000005 1610 03       5702        SBX1     5,DU           STEP TO PREVIOUS VALUE
      016407  035234 1610 00       5703        SBX1     A$WORK         MAKE POINTER RELATIVE
      016410  016446 7410 00       5704        STX1     VALP           AND STORE AS POINTER TO CURRENT ELEMENT
      016411  017034 2350 03       5705 DPAR1  LDA      OSFS,DU        GET FORCE TO STACK COMMAND IN AU
```

```
              2                                              PASS 2

      016412  012375 7000 00         5706          TSX0    ADD                 AND ADD AFTER CODE FOR CURRENT VALUE
      016413  016446 2210 00         5707          LDX1    VALP                GET POINTER TO CURRENT VALUE CONTROL BLOCK
      016414  035234 0610 00         5708          ADX1    A$WORK              MAKE POINTER ABSOLUTE
      016415  000002 0540 11         5709          AOS     2,1                 MAKE VALUE INCLUDE FORCE TO STACK COMMAND
      016416  777773 2210 03         5710          LDX1    -5,DU               GET MINUS LENGTH OF VALUE CONTROL BLOCK IN XR - 1
  END OF BINARY CARD 00000286
      016417  016446 0410 00         5711          ASX1    VALP                AND STEP TO NEXT ELEMENT
      016420  016445 0540 00         5712          AOS     DPART               DECREMENT NEGATIVE COUNT
      016421  016411 6010 00         5713          TNZ     DPAR1               TRANSFER IF MORE ELEMENTS TO DISPLAY VALUE
      016422  016447 2210 00         5714          LDX1    VALP1               GET SAVED POINTER TO DISPLAY VALUE CONTROL BLOCK
      016423  016446 7410 00         5715          STX1    VALP                AND RESTORE IT
      016424  035234 0610 00         5716          ADX1    A$WORK              MAKE POINTER ABSOLUTE
      016425  012513 7000 00         5717          TSX0    DELWW               COMBINE DISPLAY INTO A SINGLE VALUE
      016426  016446 2210 00         5718          LDX1    VALP                GET POINTER TO VALUE CONTROL BLOCK
      016427  035234 0610 00         5719          ADX1    A$WORK              MAKE POINTER ABSOLUTE
      016430  000001 2350 11         5720          LDA     1,1                 GET MODE OF DISPLAY IN AU
      016431  000022 7710 00         5721          ARL     18                  MOVE MODE TO AL
      016432  016445 7550 00         5722          STA     DPART               SAVE FOR FUTURE USE
      016433  016512 0750 03         5723          ADA     O$DISP,DU           ADD START DISPLAY CODE
      016434  012403 7000 00         5724          TSX0    INS                 AND INSERT IN FRONT OF DISPLAY
      016435  016445 2350 00         5725          LDA     DPART               GET MODE OF DISPLAY IN AL
      016436  016515 0750 03         5726          ADA     O$EDISP,DU          ADD END DISPLAY CODE
      016437  012375 7000 00         5727          TSX0    ADD                 AND ADD AFTER DISPLAY CODE
      016440  016446 2210 00         5728 DPAR2    LDX1    VALP                GET POINTER TO VALUE CONTROL BLOCK
      016441  035234 0610 00         5729          ADX1    A$WORK              MAKE POINTER ABSOLUTE
      016442  777776 3350 07         5730          LCA     -2,DL               GET [-1, 2] IN A
      016443  000002 0550 11         5731          ASA     2,1                 AND ADD TO [LOC, LEN] WORD
      016444  016205 7100 00         5732          TRA     DCL                 AND LOOP
  END OF BINARY CARD 00000287
      016445  000000 000000          5733 DPART    ZERO
      016446  000000 000000          5734 VALP     ZERO
      016447  000000 000000          5735 VALP1    ZERO
      016450  000000 000000          5736 CNDX     ZERO
                         016450      5737 CMODE    EQU     CNDX
      016451  000000 000000          5738 MNDX     ZERO
                         016451      5739 PLOC     EQU     MNDX
      016452  000000 000000          5740 MODE     ZERO
      016453  000000 000000          5741 MODE1    ZERO
      016454  000000 000000          5742 MODE2    ZERO
      016455  000000 000000          5743 MODE3    ZERO
      016456  000000 000000          5744 OPDEF    ZERO
      016457  000000 000000          5745 CNT      ZERO
      016460  000000 000000          5746 DECLR    ZERO
      016461  000000 000000          5747 DECL1    ZERO
      016462  000000 000000          5748 IDNT     ZERO
      016463  000000 000000          5749 BELF     ZERO
      016464  000000 000001          5750 GLBL     ZERO    0,1
      016465  000000 000000          5751 LL       ZERO
      016466  000000 000000          5752 BTYPE    ZERO
      016467  000000 000000          5753 MK       ZERO
```

```
          2                                                 PASS 2

     016470  000000 777777        5754 LINKO  ZERO     0,-1
                    000001        5755 FB     BOOL     1
                    000002        5756 FP     BOOL     2
                    000004        5757 PDEN   BOOL     4
                    000010        5758 ZCOM   BOOL     10
                    000020        5759 MCOM   BOOL     20
                    000040        5760 ZCOL   BOOL     40
                    000100        5761 MCOL   BOOL     100
                    000200        5762 ZSEM   BOOL     200
                    000400        5763 MSEM   BOOL     400
     016471  000000 0110 00       5764 DONE   NOP
     016472  017160 7100 00       5765        TRA      3$PASS3          GO TO PASS 3
  END OF BINARY CARD 00000288
     016473  005564 7100 00       5766        TRA      P$PCTBL          GO PRINT OUT INTERMEDIATE POLISH CODE
     016474  000000 000000        5767 CPR    ZERO
     016475  000000 000000        5768 OPT    ZERO
                                  5769 OPER   MACRO
                                  5770        IDRP     #1
                                  5771        OPER1    #1
                                  5772        IDRP
                                  5773        ENDM     OPER
                                  5774 OPER1  MACRO
                                  5775 #1     ZERO     3$#1,#2
                                  5776        UASCI    2,#1
                                  5777        ENDM     OPER1
                                  5778        HEAD     0
                    016476        5779 OTBL   EQU      *
                    016476        5780        OPER     (OP(3),OPE(3),LBL(1),JUMP(1),DISP(2),EDISP(2),ENTER(2))
  END OF BINARY CARD 00000289
                    016523        5781        OPER     (GOTO(3),HIP(2))
                    016531        5782        OPER     (VOID(2),LGEN(2),HGEN(2),CONF(2),TF(1),CASE(1),CASGN(2))
  END OF BINARY CARD 00000290
                    016556        5783        OPER     (SELCT(1),RSLCT(1),ETC(2),SKIP(2),NIL(2),IS(1),ISNT(1))
  END OF BINARY CARD 00000291
                    016603        5784        OPER     (TRUE(1),FALSE(1),MSCW(1),IDNTY(3),IDNTE(1))
                    016622        5785        OPER     (FORMP(3),IDENT(3),DENOT(3),ASGN(2),ASGNE(2))
  END OF BINARY CARD 00000292
                    016641        5786        OPER     (LL(1),LLE(1),DELV(2))
  END OF BINARY CARD 00000293
                    016652        5787        OPER     (SRNGE(1),ERNGE(1),EPDN(1),RETN(2),EPDV(2),EPDE(1))
                    016674        5788        OPER     (DLEN(1),VSBCT(1),VLWB(1),VUPB(1),VNLWB(1),VEPTY(1))
  END OF BINARY CARD 00000294
                    016716        5789        OPER     (LWB(1),UPB(1),FIX(1),FLEX(1),SUB(1),BUS(1),BOUND(1))
  END OF BINARY CARD 00000295
                    016743        5790        OPER     (RBUS(2))
                    016746        5791        OPER     (DSUB(1),DBUS(1))
  END OF BINARY CARD 00000296
                                  5792        HEAD     B,M,0
                    016754        5793        OPER     (PRIM(2),STRCT(2),REF(2),DEREF(2),ROW(2),MMI(1))
U    016770  000000 000002               ROW  ZERO     3$ROW,2
```

```
              B,M,O                                                   PASS 2

                          016776      5794             OPER     (ROWE(2))
U     016776  000000 000002                 ROWE       ZERO     3$ROWE,2
                          017001      5795             OPER     (PROC(2),DEPR(2),UNION(2),XFER(2),EMPTY(1))
  END OF BINARY CARD 00000297
                          017020      5796             OPER     (MREF(1),CONE(1),MAX(1),MA(1),FS(1))
  END OF BINARY CARD 00000298
                          017037      5797 OTBLE       EQU      *
                                      5798             HEAD     W
                          400000      5799 VALUE       BOOL     400000
                          200000      5800 SKIP        BOOL     200000
                          100000      5801 NIL         BOOL     100000
                          040000      5802 PAR         BOOL     040000
                          020000      5803 BAL         BOOL     020000
                          010000      5804 VAC         BOOL     010000
                          060000      5805 MULT        EQU      PAR+BAL
                          760000      5806 OBJCT       EQU      MULT+SKIP+NIL+VALUE
                          004000      5807 OP          BOOL     004000
```

```
          W                                                PASS 3

                                  5808         TTLS                     PASS 3
                                  5809         HEAD      B
                         400000   5810 FA      BOOL      400000         DENOTATION
                         200000   5811 FB      BOOL      200000         DEREFERENCED VALUE IN REGISTER
                         100000   5812 FC      BOOL      100000         VALUE IS STORED IN STATIC WORKING STACK
                         040000   5813 FD      BOOL      040000         VALUE IS OFFSET,LL
                         020000   5814 FE      BOOL      020000         VALUE IS POINTED TO BY OFFSET,LL
                         010000   5815 FF      BOOL      010000         OFFSET,LL IS IN LOCAL STACK
                         004000   5816 FG      BOOL      004000         VALUE IS IN REGISTER
                         002000   5817 FH      BOOL      002000         (A,B) IS VALUE
                         001000   5818 FI      BOOL      001000         VALUE IS POINTED TO BY (A,B)
                         000400   5819 FJ      BOOL      000400         TEMPORARY ARRAY VALUE ON STACK
                                  5820         HEAD      3,L,R,0
                         000016   5821 S       BOOL      16
                         000017   5822 D       BOOL      17
                                  5823         HEAD      3
    017037  000000 000000         5824 SP      ZERO
    017040  000000 000000         5825 MAXS    ZERO
    017041  000000 000000         5826 MAXST   ZERO
    017042  000000 000000         5827 LLINK   ZERO
    017043  000000 000000         5828 DLL     ZERO
    017044  000001 000000         5829 LSTMK   ZERO      1
    017045  000000 000000         5830 PARAM   ZERO
                         000004   5831 WL      EQU       4
    017046  000000000000          5832 REG     OCT       0,0,0,0,0,0,0,0,0
    017047  000000000000
    017050  000000000000
    017051  000000000000
    017052  000000000000
    017053  000000000000
    017054  000000000000
 END OF BINARY CARD 00000299
    017055  000000000000
    017056  000000000000
    017057  000000 2210 00        5833 LD      LDX1      0
    017060  000000 2220 00        5834         LDX2      0
    017061  000000 2230 00        5835         LDX3      0
    017062  000000 2240 00        5836         LDX4      0
    017063  000000 2350 00        5837         LDA       0
    017064  000000 2360 00        5838         LDQ       0
    017065  000000 2370 00        5839         LDAQ      0
    017066  000000 4310 00        5840         FLD       0
    017067  000000 4330 00        5841         DFLD      0
    017070  000000 7410 00        5842 ST      STX1      0
    017071  000000 7420 00        5843         STX2      0
    017072  000000 7430 00        5844         STX3      0
    017073  000000 7440 00        5845         STX4      0
    017074  000000 7550 00        5846         STA       0
    017075  000000 7560 00        5847         STQ       0
    017076  000000 7570 00        5848         STAQ      0
```

```
            3                                                    PASS 3

     017077  000000 4700 00        5849          FSTR      0
     017100  000000 4570 00        5850          DFST      0
     017101  000000000011          5851 MOD      OCT       11,12,13,14,77,77,77,77,77
     017102  000000000012
  END OF BINARY CARD 00000300
     017103  000000000013
     017104  000000000014
     017105  000000000077
     017106  000000000077
     017107  000000000077

     017110  000000000077
     017111  000000000077
     017112  000000001000          5852 INST     OCT       1000,2000,3000,4000,0,0,0,0,0
     017113  000000002000
     017114  000000003000
     017115  000000004000
     017116  000000000000
     017117  000000000000
     017120  000000000000
     017121  000000000000
     017122  000000000000
     017123  017134 000000         5853 CHAN     ZERO      X1
     017124  017135 000000         5854          ZERO      X2
     017125  017136 000000         5855          ZERO      X3
     017126  017137 000000         5856          ZERO      X4
     017127  017140 000000         5857          ZERO      XA
     017130  017141 000000         5858          ZERO      XQ
  END OF BINARY CARD 00000301
     017131  017142 000000         5859          ZERO      XAQ
     017132  017142 000000         5860          ZERO      XAQ
     017133  017142 000000         5861          ZERO      XAQ
     017134  000000 000000         5862 X1       ZERO      0,0
     017135  000000 000001         5863 X2       ZERO      0,1
     017136  000000 000002         5864 X3       ZERO      0,2
     017137  000000 000003         5865 X4       ZERO      0,3
     017140  017144 000004         5866 XA       ZERO      XAQ1,4
     017141  017144 000005         5867 XQ       ZERO      XAQ1,5
     017142  017143 000004         5868 XAQ      ZERO      *+1,4
     017143  017144 000005         5869          ZERO      *+1,5
     017144  017145 000006         5870 XAQ1     ZERO      *+1,6
     017145  017146 000007         5871          ZERO      *+1,7
     017146  000000 000010         5872          ZERO      0,8
     017147  741000 000000         5873 STR      ZERO      STX+1*512,0
     017150  742000 000001         5874          ZERO      STX+2*512,1
     017151  743000 000002         5875          ZERO      STX+3*512,2
     017152  744000 000003         5876          ZERO      STX+4*512,3
     017153  755000 000004         5877          ZERO      STA,4
     017154  756000 000005         5878          ZERO      STQ,5
     017155  757000 000006         5879          ZERO      STAQ,6
     017156  470000 000007         5880          ZERO      FSTR,7
  END OF BINARY CARD 00000302
     017157  457000 000010          5881           ZERO     DFST,8
                          017160    5882 STRE      EQU      *
                          011000    5883 NOP       BOOL     011000
                          220000    5884 LDX       BOOL     220000
                          740000    5885 STX       BOOL     740000
                          620000    5886 EAX       BOOL     620000
                          624000    5887 EAX4      BOOL     624000
                          440000    5888 SXL       BOOL     440000
                          060000    5889 ADX       BOOL     060000
                          100000    5890 CMPX      BOOL     100000
                          235000    5891 LDA       BOOL     235000
                          236000    5892 LDQ       BOOL     236000
                          237000    5893 LDAQ      BOOL     237000
                          450000    5894 STZ       BOOL     450000
                          755000    5895 STA       BOOL     755000
                          756000    5896 STQ       BOOL     756000
                          757000    5897 STAQ      BOOL     757000
                          635000    5898 EAA       BOOL     635000
                          075000    5899 ADA       BOOL     075000
                          116000    5900 CMPQ      BOOL     116000
                          431000    5901 FLD       BOOL     431000
                          470000    5902 FSTR      BOOL     470000
                          433000    5903 DFLD      BOOL     433000
                          457000    5904 DFST      BOOL     457000
                          710000    5905 TRA       BOOL     710000
                          234000    5906 SZN       BOOL     234000
                          736000    5907 QLS       BOOL     736000
                          600000    5908 TZE       BOOL     600000
                          601000    5909 TNZ       BOOL     601000
                          000001    5910 AU        BOOL     01
                          000003    5911 DU        BOOL     03
                          000007    5912 DL        BOOL     07
                          000004    5913 IC        BOOL     04
                          000760    5914 OTP       BOOL     760
     017160  017242 4500 00         5915 PASS3     STZ      CPNTR              INITIALIZE CODE POINTER
     017161  035234 4500 56         5916           STZ      A$WORK,ID          PAD BOTTOM OF WORKING STACK
     017162  005754 7170 00         5917           XED      T$WOVF             CHECK FOR STACK OVERFLOW
     017163  016464 2350 00         5918           LDA      2$GLBL             GET NUMBER OF LABELS GENERATED IN AL
     017164  000022 7350 00         5919           ALS      18                 MOVE TO AU
     017165  035225 2210 03         5920           LDX1     T$LBL,DU           GET POINTER TO LABEL TABLE CONTROL WORD
     017166  005663 7000 00         5921           TSX0     T$ALOC             ALLOCATE SPACE IN LABEL TABLE FOR ALL LABELS
     017167  016464 2350 00         5922           LDA      2$GLBL             GET NUMBER OF LABELS IN AL
     017170  000012 7350 00         5923           ALS      18-8               POSITION FOR COUNT IN REPEAT
     017171  000000 6200 05         5924           EAX0     0,AL               PUT COUNT IN XR - 0
     017172  000001 1750 07         5925           SBA      1,DL               MAKE END CHECK WORK RIGHT
     017173  000000 5202 01         5926 PAS3R     RPTX     ,1                 STORE ZERO
     017174  000000 4500 11         5927           STZ      0,1                THROUGHOUT LABEL TABLE
     017175  000001 1750 03         5928           SBA      1,DU               DECREMENT 256 COUNTER
     017176  017173 6050 00         5929           TPL      PAS3R              TRANSFER IF MORE TO ZERO OUT
     017177  627000 2350 07        5930          LDA      =0627000,DL      GET EAX D,** COMMAND
     017200  017474 7000 00        5931          TSX0     GAD              PAD BOTTOM OF GENERATED INSTRUCTIONS
     017201  000004 6350 00        5932          EAA      4                GET LENGTH OF MSCW IN AU
     017202  017037 7550 00        5933          STA      SP               AND STORE AS INITIAL STACK POINTER
     017203  016470 7210 00        5934          LXL1     2$LINK0          GET POINTER TO LL0 CHAIN OF DECLARATIONS
     017204  023212 7500 00        5935          STC2     SRNGX            SAVE RETURN
 END OF BINARY CARD 00000303
     017205  023175 7100 00        5936          TRA      SRNG1            GO MAKE BLOCKS FOR ALL LL0 VALUES
     017206  017242 7200 00        5937 PAS3L    LXL0     CPNTR            GET CURRENT CODE POINTER IN XR - 0
     017207  035224 0600 00        5938          ADX0     T$CODE           MAKE POINTER ABSOLUTE
     017210  000000 2350 10        5939          LDA      0,0              GET NEXT CODE WORD IN A REGISTER
     017211  017217 6000 00        5940          TZE      PAS3E            TRANSFER IF END OF PASS 3 TO CLEANUP
     017212  017045 7550 00        5941          STA      PARAM            STORE CODE WORD IN STANDARD LOCATION
     017213  000000 2200 01        5942          LDX0     0,AU             GET ADDRESS OF APPROPRIATE ROUTINE IN XR - 0
     017214  000000 7000 10        5943          TSX0     0,0              AND TRANSFER TO APPROPRIATE SUBROUTINE
     017215  017242 0540 00        5944          AOS      CPNTR            INCREMENT CODE POINTER
     017216  017206 7100 00        5945          TRA      PAS3L            AND LOOP
     017217  035225 7210 00        5946 PAS3E    LXL1     T$LBL            GET LENGTH OF LABEL TABLE IN XR - 1
     017220  035225 0610 00        5947          ADX1     T$LBL            ADD LOCATION TO GET POINTER TO END OF TABLE
     017221  000001 3350 07        5948          LCA      1,DL             GET A MINUS ONE
     017222  000000 7550 11        5949          STA      0,1              AND STORE AT END OF LABEL TABLE
     017223  035225 2210 00        5950          LDX1     T$LBL            GET POINTER TO START OF LABEL TABLE
     017224  000001 0610 03        5951 PL1      ADX1     1,DU             STEP TO NEXT ENTRY IN LABEL TABLE
     017225  000000 7220 11        5952          LXL2     0,1              GET VALUE OF LABEL IN XR - 2
     017226  017241 6040 00        5953          TMI      PL5              TRANSFER IF AT END OF LABEL TABLE
     017227  000000 2230 11        5954          LDX3     0,1              GET ADDRESS OF FIRST INSTRUCTION WITH LABEL
     017230  017224 6000 00        5955 PL3      TZE      PL1              TRANSFER IF NO MORE INSTRUCTIONS WITH THIS LABEL
     017231  017234 7430 00        5956          STX3     PL4              SAVE VALUE OF ADDRESS OF INSTRUCTION
     017232  035226 0630 00        5957          ADX3     T$GEN            GET ABSOLUTE ADDRESS OF INSTRUCTION TO MODIFY
 END OF BINARY CARD 00000304
     017233  000000 6240 12        5958          EAX4     0,2              GET VALUE OF LABEL IN XR - 4
     017234  000000 1640 03        5959 PL4      SBX4     **,DU            MAKE LABEL RELATIVE TO INSTRUCTION ADDRESS
     017235  000000 2250 13        5960          LDX5     0,3              GET ADDRESS OF NEXT INSTRUCTION USING THIS LABEL
     017236  000000 7440 13        5961          STX4     0,3              STORE LABEL IN CURRENT INSTRUCTION

     017237  000000 6230 15        5962          EAX3     0,5              MAKE NEXT INSTRUCTION ADDRESS CURRENT
     017240  017230 7100 00        5963          TRA      PL3              AND LOOP
     017241  024720 7100 00        5964 PL5      TRA      ETYP             COMPILATION COMPLETE - GO TO LOADER
     017242  000000 000000         5965 CPNTR    ZERO
     017243  777777 7100 00        5966 STRCT    TRA      $ERROR
     017244  777777 7100 00        5967 REF      TRA      $ERROR
     017245  777777 7100 00        5968 MMI      TRA      $ERROR
     017246  777777 7100 00        5969 PROC     TRA      $ERROR
     017247  777777 7100 00        5970 XFER     TRA      $ERROR
     017250  777777 7100 00        5971 EMPTY    TRA      $ERROR
     017251  017254 7400 00        5972 OP       STX0     OPX              SAVE RETURN
     017252  022241 7000 00        5973          TSX0     IDENT            GET OPERATOR AS PROCEDURE IDENTIFIER
     017253  023516 7000 00        5974          TSX0     MSCW             MARK THE STACK TO PREPARE FOR OPERATOR ENTRY
     017254  000000 7100 00        5975 OPX      TRA      **               AND RETURN
     017255  017045 7210 00        5976 OPE      LXL1     PARAM            GET POINTER TO DEFINITION OF OPERATOR
     017256  017344 7400 00        5977 OPE0     STX0     OPEX             SAVE RETURN
     017257  035220 0610 00         5978         ADX1    T$DEF                 MAKE DEFINITION POINTER ABSOLUTE
     017260  000002 2270 11         5979         LDX7    2,1                   GET MODE OF OPERATOR
 END OF BINARY CARD 00000305
     017261  017354 7470 00         5980         STX7    OPEM                  AND SAVE OPERATOR MODE FOR FUTURE USE
     017262  000003 2220 11         5981         LDX2    3,1                   GET POINTER TO MACRO FOR THIS OPERATOR
     017263  017424 7420 00         5982         STX2    NXT                   AND STORE FOR MACRO PROCESSOR
     017264  035234 2260 00         5983         LDX6    A$WORK                GET POINTER TO END OF WORKING STACK
     017265  000004 1660 03         5984         SBX6    WL,DU                 GET POINTER TO LAST BLOCK IN WORKING STACK
     017266  017345 7000 00         5985         TSX0    OPES                  MAKE VALUE AVAILABLE TO MACRO PROCESSOR
     017267  017431 7550 00         5986         STA     BADDR                 STORE ADDRESS OF VALUE
     017270  017426 7560 00         5987         STQ     BFLAG                 STORE FLAG FOR ACCUMULATOR
     017271  017354 2270 00         5988         LDX7    OPEM                  GET MODE OF OPERATOR
     017272  010630 7000 00         5989         TSX0    A$XFER                MAKE MODE POINTER UNIQUE AND ABSOLUTE
     017273  777777 2200 17         5990         LDX0    -1,7                  GET LENGTH OF MODE IN XR - 0
     017274  000003 1000 03         5991         CMPX0   3,DU                  IS IT A PROCEDURE WITH ONE PARAMETER
     017275  017303 6000 00         5992         TZE     OPE1                  TRANSFER IF UNARY OPERATOR
     017276  000004 1660 03         5993         SBX6    WL,DU                 GET POINTER TO SECOND TO LAST BLOCK
     017277  017345 7000 00         5994         TSX0    OPES                  MAKE VALUE AVAILABLE TO MACRO PROCESSOR
     017300  017430 7550 00         5995         STA     AADDR                 STORE ADDRESS OF VALUE
     017301  017425 7560 00         5996         STQ     AFLAG                 STORE FLAG FOR ACCUMULATOR
     017302  021007 7000 00         5997         TSX0    DELV                  DELETE A BLOCK FROM THE WORKING STACK
     017303  021007 7000 00         5998 OPE1    TSX0    DELV                  DELETE A BLOCK FROM THE WORKING STACK
     017304  017354 2270 00         5999         LDX7    OPEM                  GET MODE OF OPERATOR
     017305  010630 7000 00         6000         TSX0    A$XFER                MAKE MODE POINTER UNIQUE AND ABSOLUTE
     017306  777777 0670 17         6001         ADX7    -1,7                  GET POINTER TO END OF OPERATOR MODE
 END OF BINARY CARD 00000306
     017307  777777 2270 17         6002         LDX7    -1,7                  GET MODE OF RESULT OF OPERATOR
     017310  020377 7000 00         6003         TSX0    MBLK                  MAKE A BLOCK FOR THE RESULT
     017311  000001 2350 16         6004         LDA     1,6                   GET ADDRESS OF STACKED RESULT IN AU
     017312  000000 6350 01         6005         EAA     0,AU                  ZERO OUT AL
     017313  000017 0750 07         6006         ADA     D,DL                  ADD D REGISTER MODIFICATION
     017314  017432 7550 00         6007         STA     TADDR                 AND STORE AS TEMP ADDRESS
     017315  000010 2220 03         6008         LDX2    8,DU                  GET EAQ REGISTER CONTROL NUMBER
     017316  017637 7000 00         6009         TSX0    GET                   AND MAKE ALL OTHER USE OF IT TERMINATE
     017317  000010 2220 03         6010         LDX2    8,DU                  GET EAQ REGISTER CONTROL NUMBER
     017320  017721 7000 00         6011         TSX0    DR                    AND RELEASE IT
     017321  017401 7000 00         6012         TSX0    MAC                   CALL MACRO PROCESSOR
     017322  017427 2340 00         6013         SZN     TFLAG                 SEE IF RESULT IS STACKED OR IN ACCUMULATOR
     017323  017326 6000 00         6014         TZE     OPE2                  TRANSFER IF RESULT IS IN REGISTER
     017324  100000 2250 03         6015         LDX5    BSFC,DU               GET STACKED VALUE BIT
     017325  017343 7100 00         6016         TRA     OPE3                  AND GO TO STORE IT
     017326  000000 7270 16         6017 OPE2    LXL7    0,6                   GET MODE OF RESULT
     017327  010630 7000 00         6018         TSX0    A$XFER                MAKE MODE POINTER UNIQUE AND ABSOLUTE
     017330  035216 1670 00         6019         SBX7    T$MODE                MAKE IT RELATIVE
     017331  017355 2210 03         6020         LDX1    OPET,DU               GET ADDRESS OF START OF TABLE IN XR - 1
     017332  024300 5202 02         6021         RPT     OPETE/2-OPET/2,2,TZE SEARCH FOR MODE IN TABLE
     017333  000000 1070 11         6022         CMPX7   0,1                   COMPARE MODE WITH TABLE ENTRY
     017334  777777 6010 00         6023         TNZ     SERROR                MODE NOT IN TABLE - MACRO ERROR
 END OF BINARY CARD 00000307
     017335  777776 2350 11         6024         LDA     -2,1                  GET STORE INSTRUCTION IN AL
```

```
  017336  777777 3750 07        6025         ANA       -1,DL               ZERO OUT AU
  017337  000003 7550 16        6026         STA       3,6                 AND STORE IN VALUE BLOCK
  017340  777777 2220 11        6027         LDX2      -1,1                GET REGISTER CONTROL NUMBER FOR VALUE
  017341  017637 7000 00        6028         TSX0      GET                 AND ALLOCATE REGISTER FOR VALUE
  017342  004000 2250 03        6029         LDX5      B$FG,DU             GET VALUE IN ACCUMULATOR BIT
  017343  000000 7450 16        6030 OPE3    STX5      0,6                 STORE FLAGS IN BLOCK
  017344  000000 7100 00        6031 OPEX    TRA       **                  AND EXIT
  017345  017353 7400 00        6032 OPES    STX0      OPESX               SAVE RETURN
  017346  020204 7000 00        6033         TSX0      MVA                 MAKE VALUE AVAILABLE
  017347  017352 7100 00        6034         TRA       OPES1               TRANSFER IF VALUE IS IN ACCUMULATOR
  017350  000000 2360 03        6035         LDQ       0,DU                GET NOT IN ACCUMULATOR FLAG
  017351  017353 7100 00        6036         TRA       OPESX               AND RETURN
  017352  000001 2360 03        6037 OPES1   LDQ       1,DU                GET VALUE IN ACCUMULATOR FLAG
  017353  000000 7100 00        6038 OPESX   TRA       **                  AND RETURN
  017354  000000 000000         6039 OPEM    ZERO
  017355  000003 756000         6040 OPET    ZERO      M$BOOL,STQ
  017356  000005 236000         6041         ZERO      5,LDQ
  017357  000005 756000         6042         ZERO      M$CHAR,STQ
  017360  000005 236000         6043         ZERO      5,LDQ
  017361  000007 756000         6044         ZERO      M$INT,STQ
  017362  000005 236000         6045         ZERO      5,LDQ
END OF BINARY CARD 00000308
  017363  000011 470000         6046         ZERO      M$REAL,FSTR
  017364  000007 431000         6047         ZERO      7,FLD

  017365  000014 756000         6048         ZERO      M$BITS,STQ
  017366  000005 236000         6049         ZERO      5,LDQ
  017367  000016 756000         6050         ZERO      M$BYTES,STQ
  017370  000005 236000         6051         ZERO      5,LDQ
  017371  000020 757000         6052         ZERO      M$LINT,STAQ
  017372  000006 237000         6053         ZERO      6,LDAQ
  017373  000022 457000         6054         ZERO      M$LREAL,DFST
  017374  000010 433000         6055         ZERO      8,DFLD
  017375  000025 757000         6056         ZERO      M$LBITS,STAQ
  017376  000006 237000         6057         ZERO      6,LDAQ
  017377  000027 757000         6058         ZERO      M$LBYTS,STAQ
  017400  000006 237000         6059         ZERO      6,LDAQ
                     017401     6060 OPETE   EQU       *
  017401  017423 7400 00        6061 MAC     STX0      MACX                SAVE RETURN
  017402  017424 6350 56        6062 MAC0    EAA       NXT,ID              GET ADDRESS OF NEXT WORD IN MACRO
  017403  035231 0750 00        6063         ADA       T$ZZZ               MAKE ADDRESS ABSOLUTE
  017404  000000 2350 01        6064         LDA       0,AU                GET NEXT WORD OF MACRO IN A REGISTER
  017405  000000 6360 05        6065         EAQ       0,AL                MOVE OPCODE TO QU
  017406  000033 7720 00        6066         QRL       9+18                MOVE OPCODE TO QL
  017407  000760 1760 07        6067         SBQ       OTP,DL              SUBTRACT LOWEST SPECIAL OPCODE
  017410  000006 1160 07        6068         CMPQ      OPTE-OPT,DL
END OF BINARY CARD 00000309
  017411  017433 6020 06        6069         TNC       OPT,QL              TRANSFER TO SPECIAL ROUTINE IF SPECIAL OP CODE
  017412  760337 6200 01        6070         EAX0      -OTAB,AU            GET ADDRESS MINUS LOWEST SPECIAL ADDRESS IN X - 0
  017413  000003 1000 03        6071         CMPX0     OTABE-OTAB,DU       SEE IF ADDRESS IS SPECIAL
  017414  017441 6020 10        6072         TNC       OTAB,0              TRANSFER TO SPECIAL ADDRESS ROUTINE IF SPECIAL
```

```
           3                                                 PASS 3

     017415  000100 3150 07          6073 MAC1    CANA      =0100,DL         CHECK INSTRUCTION FOR STOP BIT
     017416  017421 6010 00          6074         TNZ       MAC2             TRANSFER IF STOP BIT IS ON
     017417  017474 7000 00          6075         TSX0      GAD              ADD CONTENTS OF A REGISTER TO OUTPUT CODE
     017420  017402 7100 00          6076         TRA       MAC0             AND LOOP TO PICK UP NEXT WORD
     017421  000100 6750 07          6077 MAC2    ERA       =0100,DL         TURN OFF STOP BIT IN A REGISTER
     017422  017474 7000 00          6078         TSX0      GAD              ADD CONTENTS OF A REGISTER TO OUTPUT CODE
     017423  000000 7100 00          6079 MACX    TRA       **               AND RETURN
     017424  000000 000000           6080 NXT     ZERO
     017425  000000 000000           6081 AFLAG   ZERO
     017426  000000 000000           6082 BFLAG   ZERO
     017427  000000 000000           6083 TFLAG   ZERO
     017430  000000 000000           6084 AADDR   ZERO
     017431  000000 000000           6085 BADDR   ZERO
     017432  000000 000000           6086 TADDR   ZERO
                        017433       6087 OPT     EQU       *
     017433  017444 7100 00          6088 IFA     TRA       IFA1             BRANCH IF A OPERAND IS IN ACCUMULATOR
     017434  017447 7100 00          6089 INA     TRA       INA1             BRANCH IF A OPERAND IS NOT IN ACCUMULATOR
     017435  017452 7100 00          6090 IFB     TRA       IFB1             BRANCH IF B OPERAND IS IN ACCUMULATOR
     017436  017455 7100 00          6091 INB     TRA       INB1             BRANCH IF B OPERAND IS NOT IN ACCUMULATOR
 END OF BINARY CARD 00000310
     017437  017460 7100 00          6092 JMP     TRA       JMP1             UNCONDITIONAL JUMP
     017440  017423 7100 00          6093 STOP    TRA       MACX             EXIT MACRO PROCESSOR
                        017441       6094 OPTE    EQU       *
                        017441       6095 OTAB    EQU       *
                                     6096         HEAD      3,Z
     017441  017463 7100 00          6097 A       TRA       A1               LEFT OPERAND IF NOT IN ACCUMULATOR
     017442  017466 7100 00          6098 B       TRA       B1               RIGHT OPERAND IF NOT IN ACCUMULATOR
     017443  017471 7100 00          6099 T       TRA       T1               TEMPORARY STORAGE IF NOT IN ACCUMULATOR
                                     6100         HEAD      3
                        017444       6101 OTABE   EQU       *
     017444  017425 2340 00          6102 IFA1    SZN       AFLAG            SET INDICATORS ON A ACCUMULATOR FLAG
     017445  017402 6000 00          6103         TZE       MAC0             TRANSFER IF NOT IN ACCUMULATOR
     017446  017460 7100 00          6104         TRA       JMP1             GO TO JUMP ROUTINE
     017447  017425 2340 00          6105 INA1    SZN       AFLAG            SET INDICATORS ON A ACCUMULATOR FLAG
     017450  017402 6010 00          6106         TNZ       MAC0             TRANSFER IF IN ACCUMULATOR
     017451  017460 7100 00          6107         TRA       JMP1             GO TO JUMP ROUTINE
     017452  017426 2340 00          6108 IFB1    SZN       BFLAG            SET INDICATORS ON B ACCUMULATOR FLAG
     017453  017402 6000 00          6109         TZE       MAC0             TRANSFER IF NOT IN ACCUMULATOR
     017454  017460 7100 00          6110         TRA       JMP1             GO TO JUMP ROUTINE
     017455  017426 2340 00          6111 INB1    SZN       BFLAG            SET INDICATORS ON B ACCUMULATOR FLAG
     017456  017402 6010 00          6112         TNZ       MAC0             TRANSFER IF IN ACCUMULATOR
     017457  017460 7100 00          6113         TRA       JMP1             GO TO JUMP ROUTINE
     017460  777777 6350 01          6114 JMP1    EAA       -1,AU            CORRECT FOR INCREMENTING TALLY WORD AFTER FETCH
     017461  017424 0550 00          6115         ASA       NXT              ADD TO TALLY TO CAUSE RELATIVE JUMP
     017462  017402 7100 00          6116         TRA       MAC0             AND CONTINUE
     017463  777777 3750 07          6117 A1      ANA       -1,DL            ZERO OUT ADDRESS FIELD
     017464  017430 0750 00          6118         ADA       AADDR            ADD A ADDRESS WITH INDEX REGISTER
 END OF BINARY CARD 00000311
     017465  017415 7100 00          6119         TRA       MAC1             TRANSFER TO CONTINUE PROCESSING
     017466  777777 3750 07          6120 B1      ANA       -1,DL            ZERO OUT ADDRESS FIELD
     017467  017431 0750 00          6121          ADA      BADDR           ADD B ADDRESS WITH INDEX REGISTER
     017470  017415 7100 00          6122          TRA      MAC1            TRANSFER TO CONTINUE PROCESSING
     017471  777777 3750 07          6123 T1       ANA      -1,DL           ZERO OUT ADDRESS FIELD
     017472  017432 0750 00          6124          ADA      TADDR           ADD T ADDRESS WITH INDEX REGISTER
     017473  017415 7100 00          6125          TRA      MAC1            TRANSFER TO CONTINUE PROCESSING
     017474  017505 7400 00          6126 GAD      STX0     GADX            SAVE RETURN
     017475  017506 7550 00          6127          STA      GADT            SAVE WORD TO BE ADDED TO GENERATED CODE
     017476  035214 1660 00          6128          SBX6     T$WORK          MAKE WORKING STACK POINTER RELATIVE
     017477  000000 6350 00          6129          EAA      1-1             GET NUMBER OF WORDS TO ADD MINUS ONE IN AU
     017500  035226 2210 03          6130          LDX1     T$GEN,DU        GET POINTER TO GENERATED CODE TABLE CONTROL WORD
     017501  005663 7000 00          6131          TSX0     T$ALOC          ALLOCATE ONE WORD IN GENERATED CODE TABLE
     017502  035214 0660 00          6132          ADX6     T$WORK          MAKE WORKING STACK POINTER ABSOLUTE
     017503  017506 2350 00          6133          LDA      GADT            GET WORD TO BE ADDED
     017504  777777 7550 11          6134          STA      -1,1            AND STORE AT END OF GENERATED CODE TABLE
     017505  000000 7100 00          6135 GADX     TRA      **              AND RETURN
     017506  000000 000000           6136 GADT     ZERO
     017507  017513 7400 00          6137 GADL     STX0     GADLX           SAVE RETURN
     017510  017516 7550 00          6138          STA      GADLT           STORE TALLY WORD TO CODE
     017511  017516 2350 56          6139 GADL0    LDA      GADLT,ID        GET NEXT WORD TO ADD TO GENERATED CODE
     017512  017514 6070 00          6140          TTF      GADL1           TRANSFER IF THERE IS ANOTHER WORD
 END OF BINARY CARD 00000312
     017513  000000 7100 00          6141 GADLX    TRA      **              NO MORE WORDS TO ADD SO EXIT
     017514  017474 7000 00          6142 GADL1    TSX0     GAD             ADD WORD TO GENERATED CODE
     017515  017511 7100 00          6143          TRA      GADL0           AND LOOP

     017516  000000 000000           6144 GADLT    ZERO
     017517  017600 7400 00          6145 MOVEB    STX0     MOVEX           SAVE RETURN
     017520  017602 7570 00          6146          STAQ     MOVET           SAVE SOURCE AND DESTINATION ADDRESSES
     017521  010630 7000 00          6147          TSX0     ASXFER          MAKE MODE POINTER UNIQUE AND ABSOLUTE
     017522  777777 7270 17          6148          LXL7     -1,7            GET LENGTH OF VALUE IN XR - 7
     017523  017600 6000 00          6149          TZE      MOVEX           GO TO EXIT IF NOTHING TO MOVE
     017524  777777 6200 17          6150          EAX0     -1,7            GET NUMBER OF WORDS TO MOVE -1 IN XR - 0
     017525  017602 0400 00          6151          ASX0     MOVET           MAKE SOURCE POINTER POINT TO END OF VALUE
     017526  017603 0400 00          6152          ASX0     MOVET+1         MAKE DESTINATION POINTER POINT TO END OF VALUE
     017527  777777 2200 03          6153          LDX0     -1,DU           GET ADDRESS INCREMENT FOR MOVE
     017530  017541 7100 00          6154          TRA      MOVB1           AND GO TO MOVE ROUTINE
     017531  017600 7400 00          6155 MOVE     STX0     MOVEX           SAVE RETURN
     017532  017602 7570 00          6156          STAQ     MOVET           SAVE SOURCE AND DESTINATION ADDRESSES
     017533  017602 1160 00          6157          CMPQ     MOVET           ARE SOURCE AND DESTINATION ADDRESSES THE SAME
     017534  017600 6000 00          6158          TZE      MOVEX           YES - RETURN
     017535  010630 7000 00          6159          TSX0     ASXFER          MAKE MODE POINTER UNIQUE AND ABSOLUTE
     017536  777777 7270 17          6160          LXL7     -1,7            GET LENGTH OF VALUE IN XR - 7
     017537  017600 6000 00          6161          TZE      MOVEX           GO TO EXIT IF NOTHING TO MOVE
     017540  000001 2200 03          6162          LDX0     1,DU            GET ADDRESS INCREMENT FOR MOVE
 END OF BINARY CARD 00000313
     017541  017604 7400 00          6163 MOVB1    STX0     MOVED           AND STORE IN MEMORY
     017542  017605 4500 00          6164          STZ      MOVEF           INITIALIZE RESTORE A REGISTER FLAG
     017543  017053 2200 00          6165          LDX0     REG+5           GET Q REGISTER USE WORD
     017544  017552 6010 00          6166          TNZ      MOVE1           TRANSFER IF Q REGISTER IS IN USE
     017545  236000 2350 07          6167          LDA      LDQ,DL          GET LDQ INSTRUCTION
     017546  017602 0550 00          6168          ASA      MOVET           AND PUT IN SOURCE ADDRESS WORD
```

```
             3                                              PASS 3

     017547   756000 2350 07        6169        LDA     STQ,DL          GET STQ INSTRUCTION
     017550   017603 0550 00        6170        ASA     MOVET+1         AND PUT IN DESTINATION ADDRESS WORD
     017551   017563 7100 00        6171        TRA     MOVE3           GO GENERATE MOVE INSTRUCTIONS
     017552   017052 2200 00        6172 MOVE1  LDX0    REG+4           GET A REGISTER USE WORD
     017553   017557 6000 00        6173        TZE     MOVE2           TRANSFER IF A REGISTER IS FREE
     017554   017606 2350 00        6174        LDA     STAT            GET STA TEMP INSTRUCTION
     017555   017474 7000 00        6175        TSX0    GAD             AND PUT IN GENERATED CODE
     017556   017605 7500 00        6176        STC2    MOVEF           SET FLAG TO RESTORE A REGISTER
     017557   235000 2350 07        6177 MOVE2  LDA     LDA,DL          GET LDA INSTRUCTION
     017560   017602 0550 00        6178        ASA     MOVET           AND PUT IN SOURCE ADDRESS WORD
     017561   755000 2350 07        6179        LDA     STA,DL          GET STA INSTRUCTION
     017562   017603 0550 00        6180        ASA     MOVET+1         AND PUT IN DESTINATION ADDRESS WORD
     017563   017602 2350 00        6181 MOVE3  LDA     MOVET           GET SOURCE FETCH INSTRUCTION
     017564   017474 7000 00        6182        TSX0    GAD             AND ADD TO GENERATED CODE
     017565   017603 2350 00        6183        LDA     MOVET+1         GET DESTINATION STORE INSTRUCTION
     017566   017474 7000 00        6184        TSX0    GAD             AND ADD TO GENERATED CODE
   END OF BINARY CARD 00000314
     017567   017604 2200 00        6185        LDX0    MOVED           GET MOVE INCREMENT IN XR - 0
     017570   017602 0400 00        6186        ASX0    MOVET           INCREMENT SOURCE FETCH INSTRUCTION
     017571   017603 0400 00        6187        ASX0    MOVET+1         INCREMENT DESTINATION STORE INSTRUCTION
     017572   000001 1670 03        6188        SBX7    1,DU            DECREMENT NUMBER OF WORDS LEFT TO MOVE
     017573   017563 6010 00        6189        TNZ     MOVE3           TRANSFER IF MORE WORDS TO MOVE
     017574   017605 2340 00        6190        SZN     MOVEF           CHECK RESTORE A REGISTER FLAG
     017575   017600 6000 00        6191        TZE     MOVEX           TRANSFER IF NOTHING TO RESTORE
     017576   017607 2350 00        6192        LDA     LDAT            GET LDA TEMP INSTRUCTION
     017577   017474 7000 00        6193        TSX0    GAD             AND ADD TO GENERATED CODE
     017600   000000 7100 00        6194 MOVEX  TRA     **
     017601   000000011007
                           017602   6195        EVEN
     017602   000000000000          6196 MOVET  OCT     0,0
     017603   000000000000
     017604   000000 000000         6197 MOVED  ZERO
     017605   000000 000000         6198 MOVEF  ZERO
     017606   000046 7550 00        6199 STAT   STA     38              #### TEMPORARY LOCATION
     017607   000046 2350 00        6200 LDAT   LDA     38              #### TEMPORARY LOCATION
     017610   000004 6220 00        6201 GA     EAX2    4               GET POINTER TO A REGISTER CONTROL WORD
     017611   017637 7100 00        6202        TRA     GET             AND GO ALLOCATE REGISTER
     017612   000005 6220 00        6203 GQ     EAX2    5               GET POINTER TO Q REGISTER CONTROL WORD
     017613   017637 7100 00        6204        TRA     GET             AND GO ALLOCATE REGISTER
     017614   000006 6220 00        6205 GAQ    EAX2    6               GET POINTER TO AQ REGISTER CONTROL WORD
   END OF BINARY CARD 00000315
     017615   017637 7100 00        6206        TRA     GET             AND GO ALLOCATE REGISTER
     017616   000007 6220 00        6207 GEA    EAX2    7               GET POINTER TO EA(Q) REGISTER CONTROL WORD
     017617   017637 7100 00        6208        TRA     GET             AND GO ALLOCATE REGISTER
     017620   000010 6220 00        6209 GEAQ   EAX2    8               GET POINTER TO EAQ REGISTER CONTROL WORD
     017621   017637 7100 00        6210        TRA     GET             AND GO ALLOCATE REGISTER
     017622   000000 2210 03        6211 GXR    LDX1    0,DU            GET POINTER TO FIRST X REGISTER CONTROL WORD
     017623   777777 2230 03        6212        LDX3    -1,DU           GET MAX NUMBER IN XR - 3
     017624   017046 1030 11        6213 GXR1   CMPX3   REG,1           CHECK CURRENT REGISTER FOR LEAST RECENT USE
     017625   017630 6020 00        6214        TNC     GXR2            TRANSFER IF NOT LEAST RECENT USE
     017626  017046 2230 11        6215         LDX3    REG,1             GET NEW REGISTER TO COMPARE AGAINST
     017627  000000 6220 11        6216         EAX2    0,1               SAVE CONTROL WORD POINTER IN XR - 2
     017630  000001 0610 03        6217 GXR2    ADX1    1,DU              STEP TO NEXT X REGISTER
     017631  000004 1010 03        6218         CMPX1   4,DU              CHECK IF ALL REGISTERS CHECKED
     017632  017624 6010 00        6219         TNZ     GXR1              TRANSFER IF MORE X REGISTERS TO CHECK
     017633  017101 2350 12        6220         LDA     MOD,2             GET MODIFICATION FOR CHOSEN REGISTER
     017634  017664 7550 00        6221         STA     XREGM             AND STORE FOR OTHER ROUTINES
     017635  017112 2350 12        6222         LDA     INST,2            GET INSTRUCTION FOR CHOSEN REGISTER
     017636  017663 7550 00        6223         STA     XREGI             AND STORE FOR OTHER ROUTINES
     017637  017662 7400 00        6224 GET     STX0    GXRX              SAVE RETURN
     017640  017046 6200 12        6225         EAX0    REG,2             GET ABSOLUTE POINTER TO REGISTER CONTROL WORD
     017641  017651 7400 00        6226         STX0    GXR4              AND SAVE FOR FUTURE UPDATE
     017642  017123 6200 12        6227         EAX0    CHAN,2            GET POINTER TO CHAIN FOR THIS REGISTER
 END OF BINARY CARD 00000316
     017643  017652 7400 00        6228         STX0    GXR5              AND STORE FOR FUTURE USE
     017644  017046 2230 12        6229         LDX3    REG,2             GET CURRENT REGISTER CONTROL WORD
     017645  017647 6000 00        6230         TZE     GXR3              TRANSFER IF REGISTER IS FREE
     017646  017730 7000 00        6231         TSX0    PURGE             MAKE REGISTER FREE
     017647  000000 6200 16        6232 GXR3    EAX0    0,6               GET POINTER TO CURRENT BLOCK IN XR - 0
     017650  035214 1600 00        6233         SBX0    T$WORK            MAKE POINTER RELATIVE
     017651  000000 7400 00        6234 GXR4    STX0    **                STORE POINTER IN CONTROL WORD
     017652  000000 2210 00        6235 GXR5    LDX1    **                GET POINTER TO FIRST WORD IN CHAIN
     017653  000000 7220 11        6236 GXR6    LXL2    0,1               GET CONFLICTING REGISTER POINTER IN XR - 2
     017654  017046 2340 12        6237         SZN     REG,2             IS THIS REGISTER IN USE
     017655  017660 6010 00        6238         TNZ     GXR7              TRANSFER IF IN USE
     017656  400000 2200 03        6239         LDX0    =O400000,DU       GET A CONFLICT FLAG
     017657  017046 7400 12        6240         STX0    REG,2             AND STORE CONFLICT FLAG
     017660  000000 2210 11        6241 GXR7    LDX1    0,1               STEP TO NEXT WORD IN CHAIN
     017661  017653 6010 00        6242         TNZ     GXR6              TRANSFER IF THERE ARE MORE WORDS IN CHAIN
     017662  000000 7100 00        6243 GXRX    TRA     **                AND EXIT
     017663  000000 000000         6244 XREGI   ZERO
     017664  000000 000000         6245 XREGM   ZERO
     017665  017672 7400 00        6246 CL1     STX0    CL1X              SAVE RETURN
     017666  200000 3050 03        6247         CANX5   B$FB,DU           IS DEREFERENCED VALUE STORE COMMAND IN 1,6
     017667  017671 6000 00        6248         TZE     CL11              TRANSFER IF NOT
     017670  017707 7000 00        6249         TSX0    CREG              DEALLOCATE REGISTERS USED BY STORE COMMAND
 END OF BINARY CARD 00000317
     017671  177777 3650 03        6250 CL11    ANX5    -1-B$FB-B$FA,DU MAKE 1,6 FREE
     017672  000000 7100 00        6251 CL1X    TRA     **                AND RETURN
     017673  017706 7400 00        6252 CL3     STX0    CL3X              SAVE RETURN
     017674  004000 3050 03        6253         CANX5   B$FG,DU           IS THIS A STORE REGISTER COMMAND
     017675  017700 6000 00        6254         TZE     CL31              TRANSFER IF NOT
     017676  017707 7000 00        6255         TSX0    CREG              DEALLOCATE REGISTERS USED BY STORE COMMAND
     017677  017705 7100 00        6256         TRA     CL32              AND GO CLEAN UP
     017700  003000 3050 03        6257 CL31    CANX5   B$FH+B$FI,DU      IS THIS OF THE FORM (A,B)
     017701  017705 6000 00        6258         TZE     CL32              TRANSFER IF NOT
     017702  000017 3760 07        6259         ANQ     =O17,DL           GET REGISTER IN QL
     017703  777767 6220 06        6260         EAX2    -9,QL             GET REGISTER CONTROL WORD POINTER IN XR - 2
     017704  017721 7000 00        6261         TSX0    DR                AND DELETE REGISTER REFERENCES
     017705  770777 3650 03        6262 CL32    ANX5    -1-B$FG-B$FH-B$FI,DU MAKE 3,6 FREE
```

```
              3                                          PASS 3

      017706  000000 7100 00        6263 CL3X    TRA     **                AND RETURN
      017707  017720 7400 00        6264 CREG    STX0    CREGX             SAVE RETURN
      017710  777000 3760 07        6265         ANQ     =0777000,DL       ZERO OUT ALL BUT OP CODE
      017711  000000 6210 06        6266         EAX1    0,QL              PUT OP CODE IN XR - 1
      017712  017147 6220 00        6267         EAX2    STR               PUT POINTER TO STORE COMMAND TABLE IN XR - 2
      017713  022300 5202 01        6268         RPT     STRE-STR,1,TZE    SEARCH FOR
      017714  000000 1010 12        6269         CMPX1   0,2               OP CODE IN TABLE
      017715  777777 6010 00        6270         TNZ     $ERROR            NOT THERE - COMPILER ERROR
      017716  777777 7220 12        6271         LXL2    -1,2              GET REGISTER CONTROL WORD POINTER IN XR - 2
   END OF BINARY CARD 00000318
      017717  017721 7000 00        6272         TSX0    DR                AND DELETE REGISTER REFERENCES
      017720  000000 7100 00        6273 CREGX   TRA     **                AND RETURN
      017721  017727 7400 00        6274 DR      STX0    DRX               SAVE RETURN
      017722  017123 2230 12        6275         LDX3    CHAN,2            GET IN XR - 3 POINTER TO REGISTER TO RELEASE
      017723  000000 7220 13        6276 DR1     LXL2    0,3               GET POINTER TO CURRENT REGISTER TO RELEASE
      017724  017046 4500 12        6277         STZ     REG,2             MAKE REGISTER FREE
      017725  000000 2230 13        6278         LDX3    0,3               GET POINTER TO NEXT REGISTER TO FREE
      017726  017723 6010 00        6279         TNZ     DR1               TRANSFER IF ANY MORE TO DELETE
      017727  000000 7100 00        6280 DRX     TRA     **                AND RETURN
      017730  020050 7400 00        6281 PURGE   STX0    PX                SAVE RETURN
      017731  017123 2230 12        6282         LDX3    CHAN,2            GET IN XR - 3 POINTER TO REGISTER TO PURGE
      017732  000000 7220 13        6283 P1      LXL2    0,3               GET IN XR - 2 CURRENT REGISTER TO PURGE
      017733  017046 2240 12        6284         LDX4    REG,2             GET RELATIVE POINTER TO BLOCK USING REGISTER
      017734  020045 6040 00        6285         TMI     P9                TRANSFER IF IN USE BY DIFFERENT NAME
      017735  020045 6000 00        6286         TZE     P9                TRANSFER IF NOT IN USE
      017736  035214 0640 00        6287         ADX4    TSWORK            MAKE POINTER TO BLOCK ABSOLUTE
      017737  020051 4500 00        6288         STZ     PF                INITIALIZE ERROR FLAG
      017740  000000 2250 14        6289         LDX5    0,4               GET FLAGS IN BLOCK
      017741  200000 3050 03        6290         CANX5   BSFB,DU           IS DEREFERENCED VALUE IN REGISTERS
      017742  017751 6000 00        6291         TZE     P2                TRANSFER IF NOT
      017743  000001 2350 14        6292         LDA     1,4               GET REGISTER USED FOR VALUE IN AL
      017744  777777 3750 07        6293         ANA     =1,DL             ZERO OUT AU
   END OF BINARY CARD 00000319
      017745  017070 1150 12        6294         CMPA    ST,2              COMPARE WITH A STORE OF CURRENT REGISTER
      017746  017751 6010 00        6295         TNZ     P2                TRANSFER IF NOT THE SAME REGISTER
      017747  577777 3650 03        6296         ANX5    =1-BSFB,DU        RESET DEREFERENCED VALUEBIT
      017750  020051 7500 00        6297         STC2    PF                SET PURGE FLAG
      017751  004000 3050 03        6298 P2      CANX5   BSFG,DU           CHECK VALUE IN REGISTER BIT
      017752  017770 6000 00        6299         TZE     P4                TRANSFER IF VALUE IS NOT IN A REGISTER
      017753  000003 2350 14        6300         LDA     3,4               GET STORE REGISTER COMMAND FROM BLOCK
      017754  017070 1150 12        6301         CMPA    ST,2              IS IT A STORE OF THE CURRENT REGISTER
      017755  017770 6010 00        6302         TNZ     P4                NO - TRANSFER
      017756  110000 3050 03        6303         CANX5   BSFC+BSFF,DU      IS VALUE STACKED OR LOCAL
      017757  017766 6010 00        6304         TNZ     P3                TRANSFER IF YES - DO NOT HAVE TO SAVE VALUE
      017760  000001 2350 14        6305         LDA     1,4               GET ADDRESS OF STACKED TEMP IN AU
      017761  000000 6350 01        6306         EAA     0,AU              ZERO OUT AL
      017762  000003 0750 14        6307         ADA     3,4               ADD IN STORE COMMAND
      017763  000017 0750 07        6308         ADA     D,DL              ADD MODIFICATION BY XR - D
      017764  017474 7000 00        6309         TSX0    GAD               AND ADD STORE TEMP,D TO GENERATED CODE
      017765  100000 2650 03        6310         ORX5    BSFC,DU           SET VALUE IS STACKED FLAG
```

```
    017766  773777 3650 03        6311 P3     ANX5     -1-B$FG,DU       RESET VALUE IN REGISTER BIT
    017767  020051 7500 00        6312        STC2     PF               SET PURGE FLAG
    017770  003000 3050 03        6313 P4     CANX5    B$FH+B$FI,DU     IS A REGISTER USED TO SPECIFY THE VALUE (A,B)
    017771  020042 6000 00        6314        TZE      P8               TRANSFER IF NOT
    017772  000003 2350 14        6315        LDA      3,4              GET (A,B) IN A REGISTER
 END OF BINARY CARD 00000320
    017773  777777 3750 07        6316        ANA      -1,DL            GET MODIFICATION ONLY IN A REGISTER
    017774  017101 1150 12        6317        CMPA     MOD,2            IS MODIFICATION BY CURRENT REGISTER
    017775  020042 6010 00        6318        TNZ      P8               TRANSFER IF NOT
    017776  110000 3050 03        6319        CANX5    B$FC+B$FF,DU     IS VALUE STACKED OR LOCAL
    017777  020036 6010 00        6320        TNZ      P7               TRANSFER IF YES - DO NOT HAVE TO SAVE REGISTER
    020000  002000 3050 03        6321        CANX5    B$FH,DU          IS (A,B) VALUE
    020001  020026 6000 00        6322        TZE      P6               TRANSFER IF NOT VALUE
    020002  000003 2200 14        6323        LDX0     3,4              GET VALUE OF OFFSET TO VALUE
    020003  020010 6000 00        6324        TZE      P5               TRANSFER IF NO OFFSET
    020004  000003 2350 14        6325        LDA      3,4              GET (A,B) IN A REGISTER
    020005  620000 0750 07        6326        ADA      EAX,DL           ADD EAX INSTRUCTION
    020006  017112 0750 12        6327        ADA      INST,2           MAKE RESULT GO TO CURRENT REGISTER
    020007  017474 7000 00        6328        TSX0     GAD              AND ADD TO GENERATED CODE
    020010  000001 2350 14        6329 P5     LDA      1,4              GET LOCATION OF ALLOCATED TEMP IN A
    020011  000000 6350 01        6330        EAA      0,AU             ZERO OUT AL
    020012  017070 0750 12        6331        ADA      ST,2             ADD STORE CURRENT REGISTER COMMAND
    020013  000017 0750 07        6332        ADA      D,DL             ADD MODIFICATION BY XR - D
    020014  017474 7000 00        6333        TSX0     GAD              AND ADD TO GENERATED CODE
    020015  000000 7270 14        6334        LXL7     0,4              GET MODE OF VALUE IN XR - 6
    020016  022103 7000 00        6335        TSX0     MTL              GET TYPE ADDRESS IN AU
    020017  620000 0750 07        6336        ADA      EAX,DL           ADD EAX0 INSTRUCTION
    020020  017474 7000 00        6337        TSX0     GAD              AND ADD TO GENERATED CODE
 END OF BINARY CARD 00000321
    020021  000001 2350 14        6338        LDA      1,4              GET POINTER TO ALLOCATED TEMP
    020022  000000 6350 01        6339        EAA      0,AU             ZERO OUT AL
    020023  440017 0750 07        6340        ADA      SXL+D,DL         ADD SXL0 0,D COMMAND
    020024  017474 7000 00        6341        TSX0     GAD              AND ADD TO GENERATED CODE
    020025  020036 7100 00        6342        TRA      P7               TRANSFER TO UPDATE FLAG BITS
    020026  001000 3050 03        6343 P6     CANX5    B$FI,DU          IS (A,B) REFERENCE TO VALUE
    020027  777777 6000 00        6344        TZE      $ERROR           NO - ERROR   MUST BE ONE OR THE OTHER
    020030  000000 7270 14        6345        LXL7     0,4              GET MODE OF VALUE IN XR - 7
    020031  000003 2350 14        6346        LDA      3,4              GET SOURCE ADDRESS IN A
    020032  000001 2360 14        6347        LDQ      1,4              GET ALLOCATED TEMP ADDRESS AS DESTINATION IN Q
    020033  000000 6360 02        6348        EAQ      0,QU             ZERO OUT QL
    020034  000017 0760 07        6349        ADQ      D,DL             ADD MODIFICATION BY XR - D
    020035  017531 7000 00        6350        TSX0     MOVE             MOVE VALUE TO STACK
    020036  020051 7500 00        6351 P7     STC2     PF               SET PURGE FLAG
    020037  000003 4500 14        6352        STZ      3,4              CLEAR OUT (A,B) WORD IN BLOCK
    020040  774777 3650 03        6353        ANX5     -1-B$FH-B$FI,DU  RESET (A,B) FLAG BITS
    020041  100000 2650 03        6354        ORX5     B$FC,DU          SET STACKED BIT
    020042  020051 2340 00        6355 P8     SZN      PF               CHECK PURGE FLAG
    020043  777777 6000 00        6356        TZE      $ERROR           ERROR IF NOT SET - PURGE UNSUCCESSFUL
    020044  000000 7450 14        6357        STX5     0,4              RESTORE FLAGS IN BLOCK
    020045  017046 4500 12        6358 P9     STZ      REG,2            RESET REGISTER USE WORD
```

```
     020046  000000 2230 13        6359          LDX3    0,3               GET POINTER TO NEXT REGISTER TO PURGE
   END OF BINARY CARD 00000322
     020047  017732 6010 00        6360          TNZ     P1                TRANSFER IF MORE TO DO
     020050  000000 7100 00        6361 PX       TRA     **                AND RETURN
     020051  000000 000000         6362 PF       ZERO
     020052  020065 7400 00        6363 PALL     STX0    PALLX             SAVE RETURN
     020053  000000 2220 03        6364          LDX2    0,DU              GET REGISTER TO PURGE IN XR - 2
     020054  017730 7000 00        6365          TSX0    PURGE             AND PURGE REGISTER
     020055  000001 2220 03        6366          LDX2    1,DU              GET REGISTER TO PURGE IN XR - 2
     020056  017730 7000 00        6367          TSX0    PURGE             AND PURGE REGISTER
     020057  000002 2220 03        6368          LDX2    2,DU              GET REGISTER TO PURGE IN XR - 2
     020060  017730 7000 00        6369          TSX0    PURGE             AND PURGE REGISTER
     020061  000003 2220 03        6370          LDX2    3,DU              GET REGISTER TO PURGE IN XR - 2
     020062  017730 7000 00        6371          TSX0    PURGE             AND PURGE REGISTER
     020063  000006 2220 03        6372          LDX2    6,DU              GET REGISTER TO PURGE IN XR - 2
     020064  017730 7000 00        6373          TSX0    PURGE             AND PURGE REGISTER
     020065  000000 7100 00        6374 PALLX    TRA     **                AND RETURN
     020066  020203 7400 00        6375 MNA      STX0    MNAX              SAVE RETURN
     020067  000000 2250 16        6376          LDX5    0,6               GET FLAGS IN BLOCK
     020070  400000 3050 03        6377          CANX5   BSFA,DU           IS VALUE A DENOTATION
     020071  777777 6010 00        6378          TNZ     SERROR            YES - ERROR - NAME CANNOT BE DENOTATION
     020072  010000 3050 03        6379          CANX5   BSFF,DU           IS IT A LOCAL NAME
     020073  020102 6000 00        6380          TZE     MNA2              TRANSFER IF NOT
     020074  040000 3050 03        6381          CANX5   BSFD,DU           IS VALUE LOCAL NAME
   END OF BINARY CARD 00000323
     020075  020102 6000 00        6382          TZE     MNA2              TRANSFER IF LOCAL NAME REFERS TO VALUE
     020076  000002 2350 16        6383          LDA     2,6               GET OFFSET OF NAME RELATIVE TO D REGISTER
     020077  000000 6350 01        6384          EAA     0,AU              ZERO OUT AL
     020100  000017 0750 07        6385          ADA     D,DL              ADD D REGISTER MODIFICATION
     020101  020202 7100 00        6386          TRA     MNA9              GO CLEAN UP FOR EXIT
     020102  002000 3050 03        6387 MNA2     CANX5   BSFH,DU           IS [A,B] VALUE
     020103  020106 6000 00        6388          TZE     MNA3              TRANSFER IF NOT
     020104  000003 2350 16        6389          LDA     3,6               GET [A,B] IN A REGISTER
     020105  020202 7100 00        6390          TRA     MNA9              GO CLEAN UP FOR EXIT
     020106  017622 7000 00        6391 MNA3     TSX0    GXR               ALLOCATE AN INDEX REGISTER
     020107  000000 2250 16        6392          LDX5    0,6               RESTORE FLAGS
     020110  004000 3050 03        6393          CANX5   BSFG,DU           IS VALUE IN A REGISTER
     020111  020117 6000 00        6394          TZE     MNA1              TRANSFER IF NOT IN REGISTER
     020112  620001 2350 07        6395          LDA     EAX+AU,DL         GET EAX  0,AU COMMAND
     020113  017663 0750 00        6396          ADA     XREGI             MAKE IT EAXR 0,AU
     020114  017474 7000 00        6397          TSX0    GAD               AND ADD TO GENERATED CODE
     020115  017664 2350 00        6398          LDA     XREGM             GET [0,R] IN A REGISTER
     020116  020175 7100 00        6399          TRA     MNA8              GO CLEAN UP FOR EXIT
     020117  001000 3050 03        6400 MNA1     CANX5   BSFI,DU           IS [A,B] A POINTER TO VALUE
     020120  020127 6000 00        6401          TZE     MNA4              TRANSFER IF NOT
     020121  000003 2350 16        6402          LDA     3,6               GET [A,B] IN A REGISTER
     020122  220000 0750 07        6403          ADA     LDX,DL            ADD LDX COMMAND
   END OF BINARY CARD 00000324
     020123  017663 0750 00        6404          ADA     XREGI             ADD INDEX REGISTER INTO COMMAND
     020124  017474 7000 00        6405          TSX0    GAD               ADD LDXR A,B TO GENERATED CODE
     020125  017664 2350 00          6406         LDA      XREGM              GET (0,R) IN A REGISTER
     020126  020175 7100 00          6407         TRA      MNA8               GO CLEAN UP FOR EXIT
     020127  100000 3050 03          6408 MNA4    CANX5    B$FC,DU            IS VALUE STACKED
     020130  020141 6000 00          6409         TZE      MNA5               TRANSFER IF NOT
     020131  000001 2350 16          6410         LDA      1,6                GET ADDRESS RELATIVE TO D REGISTER OF VALUE
     020132  000000 6350 01          6411         EAA      0,AU               ZERO OUT AL
     020133  000017 0750 07          6412         ADA      D,DL               ADD D REGISTER MODIFICATION
     020134  220000 0750 07          6413         ADA      LDX,DL             ADD LDX COMMAND
     020135  017663 0750 00          6414         ADA      XREGI              ADD REGISTER TO COMMAND
     020136  017474 7000 00          6415         TSX0     GAD                ADD LDXR OFFSET,D TO GENERATED CODE
     020137  017664 2350 00          6416         LDA      XREGM              GET (0,R) IN A REGISTER
     020140  020175 7100 00          6417         TRA      MNA8               GO CLEAN UP FOR EXIT
     020141  020000 3050 03          6418 MNA5    CANX5    B$FE,DU            DOES OFFSET,LL POINT TO VALUE
     020142  020155 6000 00          6419         TZE      MNA6               TRANSFER IF NOT
     020143  010000 3050 03          6420         CANX5    B$FF,DU            IS LL CURRENT RANGE
     020144  020155 6000 00          6421         TZE      MNA6               TRANSFER IF NOT
     020145  000002 2350 16          6422         LDA      2,6                GET OFFSET IN AU
     020146  000000 6350 01          6423         EAA      0,AU               ZERO OUT AL
     020147  000017 0750 07          6424         ADA      D,DL               ADD D REGISTER MODIFICATION
     020150  220000 0750 07          6425         ADA      LDX,DL             ADD LDX COMMAND
 END OF BINARY CARD 00000325
     020151  017663 0750 00          6426         ADA      XREGI              ADD X REGISTER TO COMMAND
     020152  017474 7000 00          6427         TSX0     GAD                ADD LDXR OFFSET,R TO GENERATED CODE
     020153  017664 2350 00          6428         LDA      XREGM              GET (0,R) IN A REGISTER
     020154  020175 7100 00          6429         TRA      MNA8               GO CLEAN UP FOR EXIT
     020155  060000 3050 03          6430 MNA6    CANX5    B$FD+B$FE,DU       IS VALUE EITHER IN STACK OR A STACK POINTER
     020156  777777 6000 00          6431         TZE      $ERROR             NO - VALUE MUST BE SOMETHING
     020157  020701 7000 00          6432         TSX0     NLOC               COMPILE CODE FOR NONLOCAL DISPLAY
     020160  040000 3050 03          6433         CANX5    B$FD,DU            IS (OFFSET,LL) VALUE
     020161  020166 6000 00          6434         TZE      MNA7               TRANSFER IF NOT
     020162  000002 2350 16          6435         LDA      2,6                GET OFFSET IN AU
     020163  000000 6350 01          6436         EAA      0,AU               ZERO OUT AL
     020164  017664 0750 00          6437         ADA      XREGM              ADD REGISTER MODIFICATION TO GET (OFFSET,R)
     020165  020175 7100 00          6438         TRA      MNA8               GO CLEAN UP FOR EXIT
     020166  000002 2350 16          6439 MNA7    LDA      2,6                GET OFFSET IN AU
     020167  000000 6350 01          6440         EAA      0,AU               ZERO OUT AL
     020170  017664 0750 00          6441         ADA      XREGM              ADD X REGISTER MODIFICATION
     020171  017663 0750 00          6442         ADA      XREGI              ADD X REGISTER TO COMMAND
     020172  220000 0750 07          6443         ADA      LDX,DL             ADD LDX COMMAND
     020173  017474 7000 00          6444         TSX0     GAD                ADD LDXR OFFSET,R TO GENERATED CODE
     020174  017664 2350 00          6445         LDA      XREGM              GET (0,R) IN A REGISTER
     020175  000003 2360 16          6446 MNA8    LDQ      3,6                GET PREVIOUS CONTENTS OF (A,B)
     020176  000003 7550 16          6447         STA      3,6                STORE NEW NAME IN BLOCK
 END OF BINARY CARD 00000326
     020177  017673 7000 00          6448         TSX0     CL3                DEALLOCATE PREVIOUS (A,B)
     020200  002000 2650 03          6449         ORX5     B$FH,DU            SET (A,B) IS VALUE BIT
     020201  000003 2350 16          6450         LDA      3,6                GET (A,B) FOR RETURN
     020202  000000 7450 16          6451 MNA9    STX5     0,6                RESTORE FLAGS IN BLOCK
     020203  000000 7100 00          6452 MNAX    TRA      **                 AND EXIT
     020204  020336 7400 00          6453 MVA     STX0     MVAX               SAVE RETURN
     020205  000000 2250 16          6454           LDX5     0,6                GET FLAGS FROM BLOCK IN XR - 5
     020206  004000 3050 03          6455           CANX5    B$FG,DU            IS VALUE IN A REGISTER
     020207  020212 6000 00          6456           TZE      MVA1               TRANSFER IF NOT
     020210  000003 2350 16          6457           LDA      3,6                GET STORE REGISTER COMMAND
     020211  020335 7100 00          6458           TRA      MVAR1              GO TO REGISTER RETURN
     020212  002000 3050 03          6459 MVA1      CANX5    B$FH,DU            IS VALUE (A,B)
     020213  020231 6000 00          6460           TZE      MVA3               TRANSFER IF NOT
     020214  017610 7000 00          6461           TSX0     GA                 ALLOCATE A REGISTER
     020215  000000 2250 16          6462           LDX5     0,6                RESTORE FLAGS
     020216  000003 2350 16          6463           LDA      3,6                GET (A,B) VALUE IN A REGISTER
     020217  635000 0750 07          6464           ADA      EAA,DL             ADD EAA COMMAND
     020220  017474 7000 00          6465           TSX0     GAD                ADD EAA A,B TO GENERATED CODE
     020221  000000 7270 16          6466 MVA2      LXL7     0,6                GET MODE OF VALUE
     020222  010630 7000 00          6467           TSX0     A$XFER             MAKE MODE POINTER UNIQUE AND ABSOLUTE
     020223  000001 2270 17          6468           LDX7     1,7                GET DEREFERENCED MODE
     020224  022103 7000 00          6469           TSX0     MTL                GET TYPE ADDRESS IN AU
 END OF BINARY CARD 00000327
     020225  075007 0750 07          6470           ADA      ADA+DL,DL          ADD ADA ,DL COMMAND
     020226  017474 7000 00          6471           TSX0     GAD                ADD ADA TYPE,DL TO GENERATED CODE
     020227  755000 2350 07          6472           LDA      STA,DL             GET STA COMMAND IN A REGISTER
     020230  020330 7100 00          6473           TRA      MVAR               GO TO REGISTER CLEANUP
     020231  001000 3050 03          6474 MVA3      CANX5    B$FI,DU            IS VALUE POINTED TO BY (A,B)
     020232  020235 6000 00          6475           TZE      MVA4               TRANSFER IF NOT
     020233  000003 2350 16          6476           LDA      3,6                GET (A,B) IN A REGISTER
     020234  020344 7100 00          6477           TRA      MVAP1              GO TO POINTER RETURN
     020235  010000 3050 03          6478 MVA4      CANX5    B$FF,DU            IS VALUE STORED IN LOCAL RANGE
     020236  020257 6000 00          6479           TZE      MVA6               TRANSFER IF NOT
     020237  040000 3050 03          6480           CANX5    B$FD,DU            IS VALUE (OFFSET,LL)
     020240  020251 6000 00          6481           TZE      MVA5               TRANSFER IF NOT
     020241  017610 7000 00          6482           TSX0     GA                 ALLOCATE A REGISTER
     020242  000000 2250 16          6483           LDX5     0,6                RESTORE FLAGS
     020243  000002 2350 16          6484           LDA      2,6                GET OFFSET IN AU
     020244  000000 6350 01          6485           EAA      0,AU               ZERO OUT AL
     020245  635000 0750 07          6486           ADA      EAA,DL             ADD EAA COMMAND
     020246  000017 0750 07          6487           ADA      D,DL               ADD D REGISTER MODIFICATION
     020247  017474 7000 00          6488           TSX0     GAD                ADD EAA OFFSET,D TO GENERATED CODE
     020250  020221 7100 00          6489           TRA      MVA2               GO TO ADD TYPE TO VALUE
     020251  020000 3050 03          6490 MVA5      CANX5    B$FE,DU            IS VALUE POINTED TO BY (OFFSET,D)
     020252  777777 6000 00          6491           TZE      $ERROR             CONSISTANCY CHECK
 END OF BINARY CARD 00000328
     020253  000002 2350 16          6492           LDA      2,6                GET OFFSET IN AU
     020254  000000 6350 01          6493           EAA      0,AU               ZERO OUT AL
     020255  000017 0750 07          6494           ADA      D,DL               ADD D REGISTER MODIFICATION
     020256  020344 7100 00          6495           TRA      MVAP1              GO TO POINTER CLEANUP ROUTINE
     020257  100000 3050 03          6496 MVA6      CANX5    B$FC,DU            IS VALUE IN STACK
     020260  020265 6000 00          6497           TZE      MVA7               TRANSFER IF NOT
     020261  000001 2350 16          6498           LDA      1,6                GET OFFSET TO STACK LOCATION
     020262  000000 6350 01          6499           EAA      0,AU               ZERO OUT AL
     020263  000017 0750 07          6500           ADA      D,DL               ADD D REGISTER MODIFICATION
     020264  020337 7100 00          6501           TRA      MVAP               GO TO POINTER CLEANUP ROUTINE
```

```
        020265  400000 3050 03          6502 MVA7    CANX5    B$FA,DU          IS VALUE A DENOTATION
        020266  020300 6000 00          6503         TZE      MVA8             TRANSFER IF NOT
        020267  017622 7000 00          6504         TSX0     GXR              ALLOCATE AN INDEX REGISTER
        020270  000001 2350 16          6505         LDA      1,6              GET ADDRESS OF POINTER IN AU
        020271  000000 6350 01          6506         EAA      0,AU             ZERO OUT AL
        020272  000017 0750 07          6507         ADA      D,DL             ADD D REGISTER MODIFICATION
        020273  220000 0750 07          6508         ADA      LDX,DL           ADD LDX COMMAND
        020274  017663 0750 00          6509         ADA      XREGI            ADD ALLOCATED INDEX REGISTER TO INSTRUCTION
        020275  017474 7000 00          6510         TSX0     GAD              AND ADD LDXX PTR,D TO GENERATED OUTPUT
        020276  017664 2350 00          6511         LDA      XREGM            GET ALLOCATED X REGISTER MODIFICATION IN A
        020277  020337 7100 00          6512         TRA      MVAP             GO TO POINTER CLEANUP ROUTINE
        020300  040000 3050 03          6513 MVA8    CANX5    B$FD,DU          IS VALUE A NONLOCAL NAME
    END OF BINARY CARD 00000329
        020301  020317 6000 00          6514         TZE      MVA9             TRANSFER IF NOT
        020302  017610 7000 00          6515         TSX0     GA               ALLOCATE A REGISTER
        020303  000000 2250 16          6516         LDX5     0,6              RESTORE FLAGS
        020304  020701 7000 00          6517         TSX0     NLOC             GET NAME IN A REGISTER
        020305  000002 2350 16          6518         LDA      2,6              GET OFFSET TO DISPLAY
        020306  000000 6350 01          6519         EAA      0,AU             ZERO OUT AL
        020307  635001 0750 07          6520         ADA      EAA+AU,DL        ADD EAA 0,AU COMMAND
        020310  017474 7000 00          6521         TSX0     GAD              ADD EAA OFFSET,AU TO GENERATED CODE
        020311  000000 7270 16          6522         LXL7     0,6              GET MODE OF VALUE IN XR - 7
        020312  022103 7000 00          6523         TSX0     MTL              GET TYPE ADDRESS IN AU
        020313  075007 0750 07          6524         ADA      ADA+DL,DL        ADD ADA 0,AL COMMAND
        020314  017474 7000 00          6525         TSX0     GAD              ADD ADA TYPE,DL TO GENERATED CODE
        020315  755000 2350 07          6526         LDA      STA,DL           GET STA COMMAND IN A REGISTER
        020316  020330 7100 00          6527         TRA      MVAR             GO TO REGISTER CLEANUP
        020317  020000 3050 03          6528 MVA9    CANX5    B$FE,DU          IS NONLOCAL NAME POINTER TO VALUE
        020320  777777 6000 00          6529         TZE      $ERROR           NO - MUST BE SOMETHING
        020321  017622 7000 00          6530         TSX0     GXR              ALLOCATE AN INDEX REGISTER
        020322  000000 2250 16          6531         LDX5     0,6              RESTORE FLAGS
        020323  020701 7000 00          6532         TSX0     NLOC             GET NONLOCAL DISPLAY IN XR
        020324  000002 2350 16          6533         LDA      2,6              GET OFFSET IN AU
        020325  000000 6350 01          6534         EAA      0,AU             ZERO OUT AL
        020326  017664 0750 00          6535         ADA      XREGM            ADD X REGISTER MODIFICATION
    END OF BINARY CARD 00000330
        020327  020337 7100 00          6536         TRA      MVAP             GO TO POINTER CLEANUP ROUTINE
        020330  000003 2360 16          6537 MVAR    LDQ      3,6              GET PREVIOUS CONTENTS OF (A,B)
        020331  000003 7550 16          6538         STA      3,6              STORE NEW NAME IN BLOCK
        020332  017673 7000 00          6539         TSX0     CL3              DEALLOCATE PREVIOUS (A,B)
        020333  000003 2350 16          6540         LDA      3,6              GET NEW (A,B) FOR RETURN
        020334  004000 2650 03          6541         ORX5     B$FG,DU          SET VALUE IN REGISTER BIT
        020335  000000 7450 16          6542 MVAR1   STX5     0,6              RESTORE FLAGS IN BLOCK
        020336  000000 7100 00          6543 MVAX    TRA      **               AND EXIT
        020337  000003 2360 16          6544 MVAP    LDQ      3,6              GET PREVIOUS CONTENTS OF (A,B)
        020340  000003 7550 16          6545         STA      3,6              STORE NEW (A,B) IN BLOCK
        020341  017673 7000 00          6546         TSX0     CL3              DEALLOCATE PREVIOUS (A,B)
        020342  000003 2350 16          6547         LDA      3,6              GET NEW (A,B) FOR RETURN
        020343  001000 2650 03          6548         ORX5     B$FI,DU          SET (A,B) IS REF TO VALUE BIT
        020344  000000 7450 16          6549 MVAP1   STX5     0,6              RESTORE FLAGS IN BLOCK
```

```
              3                                                   PASS 3

       020345  020336 2200 00          6550         LDX0    MVAX              GET RETURN ADDRESS IN XR - 0
       020346  000001 7100 10          6551         TRA     1,0               AND RETURN
       020347  000000 0000 04          6552 MVAT    ARG     0,IC
       020350  020353 7400 00          6553 MAX     STX0    MAXX              SAVE RETURN
       020351  017040 2200 00          6554         LDX0    MAXS              GET CURRENT STACK POINTER MAXIMUM
       020352  017037 7400 00          6555         STX0    SP                AND MAKE IT CURRENT STACK POINTER
       020353  000000 7100 00          6556 MAXX    TRA     **                AND RETURN
       020354  020376 7400 00          6557 GTMP    STX0    GTMPX             SAVE RETURN
   END OF BINARY CARD 00000331
       020355  010562 7000 00          6558         TSX0    ASRCHK            SEE IF MODE OF VALUE CONTAINS DESCRIPTORS
       020356  020364 7100 00          6559         TRA     GTMP1             TRANSFER IF NO ROW MODES TO WORRY ABOUT
       020357  017037 2200 00          6560         LDX0    SP                GET CURRENT STACK POINTER
       020360  000001 4400 16          6561         SXL0    1,6               STORE AS ADDRESS FOR STACK POINTER
       020361  000001 0600 03          6562         ADX0    1,DU              INCREMENT CURRENT STACK POINTER BY POINTER LENGTH
       020362  017037 7400 00          6563         STX0    SP                AND RESTORE IN CURRENT STACK POINTER
       020363  000000 7270 16          6564         LXL7    0,6               RESTORE XR - 7
       020364  017037 2200 00          6565 GTMP1   LDX0    SP                GET CURRENT STACK POINTER
       020365  000001 7400 16          6566         STX0    1,6               STORE AS ADDRESS FOR STATIC TEMP
       020366  000000 7270 16          6567         LXL7    0,6               GET MODE OF VALUE
       020367  010630 7000 00          6568         TSX0    ASXFER            MAKE MODE POINTER UNIQUE AND ABSOLUTE
       020370  777777 7200 17          6569         LXL0    -1,7              GET LENGTH OF VALUE IN XR - 0
       020371  017037 0600 00          6570         ADX0    SP                ADD LOCATION OF VALUE
       020372  017037 7400 00          6571         STX0    SP                STORE AS NEW STACK POINTER
       020373  017040 1000 00          6572         CMPX0   MAXS              COMPARE WITH MAXIMUM S
       020374  020376 6040 00          6573         TMI     GTMPX             TRANSFER IF NOT NEW MAXIMUM
       020375  017040 7400 00          6574         STX0    MAXS              STORE NEW MAXIMUM
       020376  000000 7100 00          6575 GTMPX   TRA     **                AND RETURN
       020377  020414 7400 00          6576 MBLK    STX0    MBLKX             SAVE RETURN
                         020400        6577         DUP     2,WL              GENERATE BLOCK WL ENTRIES LONG
       020400  035234 4500 56          6578         STZ     ASWORK,ID         ZERO OUT CREATED BLOCK
       020401  005754 7170 00          6579         XED     TSWOVF            CHECK FOR STACK OVERFLOW
       020402  035234 4500 56                       STZ     ASWORK,ID         ZERO OUT CREATED BLOCK
   END OF BINARY CARD 00000332
       020403  005754 7170 00                       XED     TSWOVF            CHECK FOR STACK OVERFLOW
       020404  035234 4500 56                       STZ     ASWORK,ID         ZERO OUT CREATED BLOCK
       020405  005754 7170 00                       XED     TSWOVF            CHECK FOR STACK OVERFLOW
       020406  035234 4500 56                       STZ     ASWORK,ID         ZERO OUT CREATED BLOCK
       020407  005754 7170 00                       XED     TSWOVF            CHECK FOR STACK OVERFLOW
       020410  035234 2260 00          6580         LDX6    ASWORK            GET POINTER TO END OF WORKING STACK
       020411  000004 1660 03          6581         SBX6    WL,DU             GET POINTER TO NEWLY CREATED BLOCK
       020412  000000 4470 16          6582         SXL7    0,6               STORE MODE IN BLOCK
       020413  020354 7000 00          6583         TSX0    GTMP              AND ALLOCATE STACK SPACE FOR VALUE
       020414  000000 7100 00          6584 MBLKX   TRA     **                AND RETURN
       020415  020420 7400 00          6585 MREF    STX0    MREFX             SAVE RETURN
       020416  000037 6270 00          6586         EAX7    MSPTR             GET POINTER MODE IN XR - 7
       020417  020377 7000 00          6587         TSX0    MBLK              MAKE A BLOCK FOR REFERENCE
       020420  000000 7100 00          6588 MREFX   TRA     **                AND EXIT
       020421  020450 7400 00          6589 MR      STX0    MRX               SAVE RETURN
       020422  000000 2250 14          6590         LDX5    0,4               GET FLAGS FROM ORIGINAL BLOCK
       020423  677377 3650 03          6591         ANX5    -1-B$FC-B$FJ,DU   RESET VALUE IS STACKED AND S IS STORED BITS
```

```
     020424  000000 7450 16       6592          STX5     0,6              AND STORE FLAGS IN NEW BLOCK
     020425  000002 2350 14       6593          LDA      2,4              GET OFFSET/LL WORD
     020426  000002 7550 16       6594          STA      2,6              AND STORE IN NEW BLOCK
     020427  000003 2350 14       6595          LDA      3,4              GET REGISTER WORD
     020430  000003 7550 16       6596          STA      3,6              AND STORE IN NEW BLOCK
 END OF BINARY CARD 00000333
     020431  000000 6230 14       6597          EAX3     0,4              GET POINTER TO OLD BLOCK IN XR - 3
     020432  035214 1630 00       6598          SBX3     T$WORK           MAKE POINTER RELATIVE
     020433  000000 6240 16       6599          EAX4     0,6              GET POINTER TO NEW BLOCK IN XR - 4
     020434  035214 1640 00       6600          SBX4     T$WORK           MAKE POINTER RELATIVE
     020435  000000 2220 03       6601          LDX2     0,DU             INITIALIZE REGISTER POINTER TO ZERO
     020436  017046 1030 12       6602 MR1      CMPX3    REG,2            IS THIS REGISTER USED BY OLD BLOCK
     020437  020441 6010 00       6603          TNZ      MR2              NO - TRANSFER
     020440  017046 7440 12       6604          STX4     REG,2            YES - CHANGE TO POINT TO NEW BLOCK
     020441  000001 0620 03       6605 MR2      ADX2     1,DU             STEP TO NEXT REGISTER
     020442  000011 1020 03       6606          CMPX2    STRE-STR,DU      IS THIS THE LAST REGISTER
     020443  020436 6010 00       6607          TNZ      MR1              NO - TRANSFER TO LOOP
     020444  000000 2250 14       6608          LDX5     0,4              GET FLAGS FROM OLD BLOCK
     020445  000400 3650 03       6609          ANX5     B$FJ,DU          ZERO ALL BITS EXCEPT S IS STORED BIT
     020446  100000 2650 03       6610          ORX5     B$FC,DU          SET VALUE IS STACKED BIT
     020447  000000 7450 14       6611          STX5     0,4              AND RESTORE FLAGS IN BLOCK
     020450  000000 7100 00       6612 MRX      TRA      **               AND RETURN
                       020451     6613 MA       EQU      *
     020451  020524 7400 00       6614 FS       STX0     FSX              SAVE RETURN
     020452  000000 2250 16       6615          LDX5     0,6              GET FLAGS FOR CURRENT BLOCK IN XR - 5
     020453  100000 3050 03       6616          CANX5    B$FC,DU          IS VALUE ALREADY STACKED
     020454  020514 6010 00       6617          TNZ      FS2              TRANSFER IF VALUE ALREADY STACKED
     020455  020204 7000 00       6618          TSX0     MVA              MAKE VALUE AVAILABLE
     020456  020507 7100 00       6619          TRA      FS1              TRANSFER IF VALUE IS IN A REGISTER
 END OF BINARY CARD 00000334
     020457  000000 7270 16       6620          LXL7     0,6              GET MODE OF VALUE IN XR - 7
     020460  000001 2360 16       6621          LDQ      1,6              GET DESTINATION ADDRESS IN QU
     020461  000000 6360 02       6622          EAQ      0,QU             ZERO OUT QL
     020462  000017 0760 07       6623          ADQ      D,DL             ADD D REGISTER MODIFICATION
     020463  017531 7000 00       6624          TSX0     MOVE             MOVE VALUE TO STACK
     020464  000000 7270 16       6625          LXL7     0,6              GET MODE OF VALUE IN XR - 7
     020465  010562 7000 00       6626          TSX0     A$RCHK           SEE IF ANY ARRAYS TO MOVE TO STACK
     020466  020514 7100 00       6627          TRA      FS2              TRANSFER IF NO ARRAYS TO MOVE
     020467  000001 7200 16       6628          LXL0     1,6              GET ADDRESS OF LOCATION TO SAVE S REGISTER
     020470  020525 7400 00       6629          STX0     FSI              AND STORE IN INSTRUCTION
     020471  020525 2350 00       6630          LDA      FSI              GET STX S,(SAVE LOCATION),D INSTRUCTION
     020472  017474 7000 00       6631          TSX0     GAD              AND ADD TO GENERATED OUTPUT
     020473  000400 2250 03       6632          LDX5     B$FJ,DU          GET S REGISTER IS SAVED BIT
     020474  000000 2450 16       6633          ORSX5    0,6              AND OR INTO BLOCK FLAG WORD
     020475  000000 7270 16       6634          LXL7     0,6              GET MODE OF VALUE IN XR - 7
     020476  020377 7000 00       6635          TSX0     MBLK             AND MAKE A BLOCK FOR ANOTHER SIMILAR VALUE
     020477  777774 6240 16       6636          EAX4     -WL,6            GET POINTER TO OLD BLOCK IN XR - 4
     020500  020421 7000 00       6637          TSX0     MR               MOVE REGISTER POINTERS TO NEW BLOCK
     020501  000000 7270 16       6638          LXL7     0,6              GET MODE OF VALUE IN XR - 7
     020502  025741 2200 03       6639          LDX0     R$AFS,DU         GET ADDRESS OF RUNTIME FS ROUTINE
     020503  021524 7400 00        6640          STX0    ASI8              AND STORE IN INSTRUCTION SEQUENCE
     020504  021041 7000 00        6641          TSX0    ASG               MOVE ARRAYS TO STACK
 END OF BINARY CARD 00000335
     020505  021007 7000 00        6642          TSX0    DELV              DELETE BLOCK ADDED FOR FS
     020506  020514 7100 00        6643          TRA     FS2               TRANSFER TO CLEAN UP
     020507  000001 2350 16        6644 FS1      LDA     1,6               GET ADDRESS OF STACK VALUE IN AU
     020510  000000 6350 01        6645          EAA     0,AU              ZERO OUT AL
     020511  000017 0750 07        6646          ADA     D,DL              ADD D REGISTER MODIFICATION
     020512  000003 0750 16        6647          ADA     3,6               ADD STORE COMMAND
     020513  017474 7000 00        6648          TSX0    GAD               AND ADD TO GENERATED OUTPUT
     020514  000000 2250 16        6649 FS2      LDX5    0,6               GET FLAGS FROM BLOCK IN XR - 5
     020515  000001 2360 16        6650          LDQ     1,6               GET REGISTER TO DEALLOCATE
     020516  017665 7000 00        6651          TSX0    CL1               DEALLOCATE DEREFERENCED VALUE
     020517  000003 2360 16        6652          LDQ     3,6               GET REGISTER TO DEALLOCATE
     020520  017673 7000 00        6653          TSX0    CL3               DEALLOCATE ALL OTHER REGISTERS
     020521  100000 2650 03        6654          ORX5    BSFC,DU           SET STACKED BIT
     020522  707777 3650 03        6655          ANX5    -1-BSFD-BSFE-BSFF,DU RESET ANY RANGE INFORMATION
     020523  000000 7450 16        6656          STX5    0,6               RESTORE FLAGS IN BLOCK
     020524  000000 7100 00        6657 FSX      TRA     **                AND RETURN
     020525  000000 7460 17        6658 FSI      STX     S,0,D
     020526  020674 7400 00        6659 STYPE    STX0    STYPX             SAVE RETURN
     020527  020675 4500 00        6660          STZ     STYPR             INITIALIZE REGISTER INDEX
     020530  020675 7220 00        6661          LXL2    STYPR             GET NEXT REGISTER TO PURGE
     020531  017730 7000 00        6662 STYP1    TSX0    PURGE             AND PURGE REGISTER
     020532  020675 0540 00        6663          AOS     STYPR             STEP TO NEXT REGISTER
 END OF BINARY CARD 00000336
     020533  020675 7220 00        6664          LXL2    STYPR             GET NEXT REGISTER TO PURGE
     020534  000011 1020 03        6665          CMPX2   STRE-STR,DU       SEE IF LAST REGISTER IS PURGED

     020535  020531 6010 00        6666          TNZ     STYP1             TRANSFER IF MORE REGISTERS TO PURGE
     020536  035227 2210 03        6667          LDX1    TSTYPE,DU         GET POINTER TO TYPE TABLE CONTROL WORD
     020537  000000 6350 00        6668          EAA     1-1               GET NUMBER OF WORDS NEEDED - 1 IN AU
     020540  005663 7000 00        6669          TSX0    TSALOC            AND ALLOCATE SPACE IN TYPE TABLE
     020541  000001 1610 03        6670          SBX1    1,DU              GET POINTER TO ALLOCATED WORD IN XR - 1
     020542  035227 1610 00        6671          SBX1    TSTYPE            MAKE POINTER RELATIVE
     020543  020676 7410 00        6672          STX1    STYPP             AND SAVE POINTER IN MEMORY
     020544  035234 2260 00        6673          LDX6    ASWORK            GET POINTER TO END OF WORKING STACK
     020545  035214 1660 00        6674          SBX6    TSWORK            MAKE POINTER RELATIVE
     020546  000004 1660 03        6675          SBX6    WL,DU             GET RELATIVE POINTER TO LAST BLOCK IN WORK
     020547  020677 7460 00        6676          STX6    STYE              AND SAVE
     020550  017044 2260 00        6677          LDX6    LSTMK             GET POINTER TO LAST TIME WORK WAS MARKED
     020551  035214 0660 00        6678 STYP2    ADX6    TSWORK            MAKE BLOCK POINTER ABSOLUTE
     020552  000000 7270 16        6679          LXL7    0,6               GET MODE OF BLOCK IN XR - 7
     020553  010562 7000 00        6680          TSX0    ASRCHK            SEE IF MODE IS A ROW TYPE MODE
     020554  020572 7100 00        6681          TRA     STYP7             TRANSFER IF NOT ROW TYPE
     020555  000000 6350 00        6682          EAA     1-1               ALLOCATE ONE WORD
     020556  035227 2210 03        6683          LDX1    TSTYPE,DU         IN THE TYPE TABLE
     020557  035214 1660 00        6684          SBX6    TSWORK            MAKE BLOCK POINTER RELATIVE
     020560  005663 7000 00        6685          TSX0    TSALOC            ALLOCATE MEMORY
 END OF BINARY CARD 00000337
     020561  035214 0660 00        6686          ADX6    TSWORK            MAKE BLOCK POINTER ABSOLUTE
```

```
             3                                              PASS 3

      020562  000000 2250 16          6687          LDX5     0,6                GET FLAGS FROM BLOCK IN XR - 5
      020563  000400 3050 03          6688          CANX5    B$FJ,DU            SEE IF POINTER TO STACK IS STORED
      020564  020567 6000 00          6689          TZE      STYP5              TRANSFER IF NOT
      020565  027215 2350 03          6690          LDA      T$PTR,DU           GET POINTER TYPE IN AU
      020566  020571 7100 00          6691          TRA      STYP6              AND CONTINUE
      020567  027222 2350 03          6692 STYP5    LDA      T$SKIP,DU          GET SKIP TYPE IN AU
      020570  000001 0750 07          6693          ADA      1,DL               ADD ONE AS NUMBER OF WORDS TO SKIP IN AL
      020571  777777 7550 11          6694 STYP6    STA      -1,1               AND STORE TYPE WORD IN TYPE TABLE
      020572  000000 7270 16          6695 STYP7    LXL7     0,6                GET MODE OF BLOCK IN XR - 7
      020573  010630 7000 00          6696          TSX0     A$XFER             MAKE MODE POINTER ABSOLUTE
      020574  000000 2250 16          6697          LDX5     0,6                GET FLAGS FROM BLOCK IN XR - 5
      020575  100000 3050 03          6698          CANX5    B$FC,DU            IS VALUE STACKED
      020576  020623 6000 00          6699          TZE      STYP3              TRANSFER IF VALUE IS NOT STACKED
      020577  000000 7240 17          6700          LXL4     0,7                GET POINTER TO TYPE FOR MODE IN XR - 4
      020600  035227 0640 00          6701          ADX4     T$TYPE             MAKE TYPE POINTER ABSOLUTE
      020601  777777 2350 14          6702          LDA      -1,4               GET NUMBER OF WORDS IN TYPE IN AU
      020602  777777 6350 01          6703          EAA      -1,AU              GET NUMBER OF WORDS - 1 TO ALLOCATE IN AU
      020603  035227 2210 03          6704          LDX1     T$TYPE,DU          GET POINTER TO TYPE TABLE CONTROL WORD IN XR - 1
      020604  035227 1640 00          6705          SBX4     T$TYPE             MAKE TYPE POINTER RELATIVE
      020605  035214 1660 00          6706          SBX6     T$WORK             MAKE BLOCK POINTER RELATIVE
      020606  005663 7000 00          6707          TSX0     T$ALOC             ALLOCATE SPACE IN TYPE TABLE
   END OF BINARY CARD 00000338
      020607  035227 0640 00          6708          ADX4     T$TYPE             MAKE TYPE TABLE POINTER ABSOLUTE
      020610  035214 0660 00          6709          ADX6     T$WORK             MAKE BLOCK POINTER ABSOLUTE
      020611  000001 1610 03          6710          SBX1     1,DU               GET POINTER TO START OF ALLOCATED SPACE
      020612  777777 2350 14          6711          LDA      -1,4               GET NUMBER OF WORDS IN TYPE IN AU
      020613  000000 6350 01          6712          EAA      0,AU               ZERO OUT AL
      020614  000010 7710 00          6713          ARL      8                  POSITION WORD COUNT FOR REPEAT
      020615  001400 6200 05          6714          EAX0     768,AL             GET COUNT AND A AND B BITS IN XR - 0
      020616  000000011007
      020617  000000 5602 01          6715          RPDX     ,1                 MOVE
      020620  000000 2350 14          6716          LDA      0,4                FROM TYPE
      020621  000000 7550 11          6717          STA      0,1                TO NEWLY ALLOCATED SPACE IN TYPE TABLE
      020622  020634 7100 00          6718          TRA      STYP4              AND CONTINUE
      020623  777777 7240 17          6719 STYP3    LXL4     -1,7               GET LENGTH OF VALUE IN XR - 4
      020624  035227 2210 03          6720          LDX1     T$TYPE,DU          GET POINTER TO TYPE TABLE CONTROL WORD IN XR - 1
      020625  000000 6350 00          6721          EAA      1-1                PREPARE TO ALLOCATE ONE WORD
      020626  035214 1660 00          6722          SBX6     T$WORK             MAKE BLOCK POINTER RELATIVE
      020627  005663 7000 00          6723          TSX0     T$ALOC             ALLOCATE ONE WORD IN TYPE TABLE
      020630  035214 0660 00          6724          ADX6     T$WORK             MAKE BLOCK POINTER ABSOLUTE
      020631  027222 2200 03          6725          LDX0     T$SKIP,DU          GET SKIP TYPE IN XR - 0
      020632  777777 7400 11          6726          STX0     -1,1               AND STORE IN TYPE TABLE
      020633  777777 4440 11          6727          SXL4     -1,1               STORE LENGTH TO SKIP IN SAME WORD OF TYPE TABLE
      020634  000004 0660 03          6728 STYP4    ADX6     WL,DU              STEP TO NEXT BLOCK IN WORK
   END OF BINARY CARD 00000339
      020635  035214 1660 00          6729          SBX6     T$WORK             MAKE POINTER RELATIVE
      020636  020677 1060 00          6730          CMPX6    STYE               SEE IF AT END OF WORKING STACK
      020637  020551 6010 00          6731          TNZ      STYP2              TRANSFER IF MORE BLOCKS IN WORK
      020640  035227 1610 00          6732          SBX1     T$TYPE             GET RELATIVE POINTER TO END OF TYPE TABLE ENTRY
      020641  020676 1610 00          6733          SBX1     STYPP              SUBTRACT BEGINNING LOCATION OF TYPE TABLE ENTRY
```

```
         3                                                      PASS 3

      020642  777777 6350 11        6734          EAA      -1,1             GET LENGTH OF ENTRY IN AU
      020643  020676 2210 00        6735          LDX1     STYPP            GET POINTER TO START OF TYPE TABLE ENTRY
      020644  035227 0610 00        6736          ADX1     T$TYPE           MAKE POINTER ABSOLUTE
      020645  000000 7550 11        6737          STA      0,1              STORE LENGTH IN FRONT OF ENTRY
      020646  035227 1610 00        6738          SBX1     T$TYPE           MAKE POINTER RELATIVE
      020647  000001 0610 03        6739          ADX1     1,DU             STEP OVER LINK WORD
      020650  020700 7410 00        6740          STX1     STYPT            SAVE TYPE POINTER
      020651  017044 2260 00        6741          LDX6     LSTMK            GET POINTER TO BLOCK OF LAST MARK
      020652  035214 0660 00        6742          ADX6     T$WORK           MAKE POINTER ABSOLUTE
      020653  000001 2200 16        6743          LDX0     1,6              GET ADDRESS OF LAST MARK IN XR - 0
      020654  000000 6350 10        6744          EAA      0,0              GET LAST MARK ADDRESS IN AU
      020655  635017 0750 07        6745          ADA      EAA+D,DL         MAKE EAA LSTMK,D INSTRUCTION
      020656  017474 7000 00        6746          TSX0     GAD              AND ADD TO GENERATED OUTPUT
      020657  020700 2210 00        6747          LDX1     STYPT            GET TYPE POINTER IN XR - 1
      020660  022110 7000 00        6748          TSX0     TL               GET LINKED TYPE ADDRESS IN AU
      020661  075007 0750 07        6749          ADA      ADA+DL,DL        MAKE ADA TYPE,DL INSTRUCTION
      020662  017474 7000 00        6750          TSX0     GAD              AND ADD TO GENERATED OUTPUT
   END OF BINARY CARD 00000340
      020663  035234 2260 00        6751          LDX6     A$WORK           GET POINTER TO END OF WORKING STACK
      020664  000004 1660 03        6752          SBX6     WL,DU            GET POINTER TO MS OR MSCW BLOCK
      020665  100000 2250 03        6753          LDX5     B$FC,DU          GET VALUE IS STACKED BIT
      020666  000000 7450 16        6754          STX5     0,6              AND STORE IN BLOCK
      020667  000001 2350 16        6755          LDA      1,6              GET ADDRESS OF VALUE IN AU
      020670  000000 6350 01        6756          EAA      0,AU             ZERO OUT AL
      020671  755017 0750 07        6757          ADA      STA+D,DL         MAKE STA MS,D INSTRUCTION
      020672  017474 7000 00        6758          TSX0     GAD              AND ADD TO GENERATED OUTPUT
      020673  023652 7000 00        6759          TSX0     PUSH             PUSH OLD MS AND MAKE NEW ONE CURRENT
      020674  000000 7100 00        6760 STYPX    TRA      **               AND RETURN WITH TYPE IN XR - 1
      020675  000000 000000         6761 STYPR    ZERO
      020676  000000 000000         6762 STYPP    ZERO
      020677  000000 000000         6763 STYE     ZERO
      020700  000000 000000         6764 STYPT    ZERO
      020701  020710 7400 00        6765 NLOC     STX0     NLOCX            SAVE RETURN
      020702  000002 7220 16        6766          LXL2     2,6              GET LEXICOGRAPHICAL LEVEL DIFFERENCE IN XR - 2
      020703  017663 2350 00        6767          LDA      XREGI            GET ALLOCATED X-REGISTER FOR COMMAND
      020704  220017 0750 07        6768          ADA      LDX+D,DL         ADD LDX0 0,D COMMAND
      020705  000002 0750 03        6769          ADA      2,DU             MAKE IT LDXR 2,D
      020706  017474 7000 00        6770          TSX0     GAD              AND ADD LDXR 2,D TO GENERATED CODE
      020707  000001 1620 03        6771 NLOCL    SBX2     1,DU             DECREMENT NUMBER OF LEVELS LEFT TO EXIT
      020710  000000 6000 00        6772 NLOCX    TZE      **               EXIT IF NO MORE TO EXIT
   END OF BINARY CARD 00000341
      020711  017663 2350 00        6773          LDA      XREGI            GET ALLOCATED X-REGISTER FOR COMMAND
      020712  017664 0750 00        6774          ADA      XREGM            ADD ALLOCATED X-REGISTER MODIFICATION
      020713  220000 0750 07        6775          ADA      LDX,DL           ADD LDX COMMAND
      020714  000002 0750 03        6776          ADA      2,DU             MAKE IT LDXR 2,R
      020715  017474 7000 00        6777          TSX0     GAD              AND ADD LDXR 2,R TO GENERATED CODE
      020716  020707 7100 00        6778          TRA      NLOCL            AND LOOP
      020717  020740 7400 00        6779 IDFY     STX0     IDFYX            SAVE RETURN
      020720  017045 7210 00        6780          LXL1     PARAM            GET DEFINITION POINTER IN XR - 1
      020721  035220 0610 00        6781          ADX1     T$DEF            MAKE DEFINITION POINTER ABSOLUTE
```

```
        020722  000000 2350 11         6782         LDA     0,1              GET LL OF IDENTIFIER IN AL
        020723  020746 4500 00         6783 IDFYB   STZ     IDFYC            INITIALIZE LL DIFFERENCE
        020724  020747 4500 00         6784         STZ     IDFYF            INITIALIZE LOCAL FLAG
        020725  777777 2360 03         6785         LDQ     -1,DU            MASK OUT UPPER HALF OF A REGISTER
        020726  017042 2200 03         6786         LDX0    LLINK,DU         GET POINTER TO CURRENT ENVIRONMENT
        020727  035215 1600 00         6787         SBX0    T$STACK          MAKE PSEUDO-RELATIVE FOR LOOP ENTRY
        020730  035215 0600 00         6788 IDFY1   ADX0    T$STACK          MAKE ENVIRONMENT POINTER ABSOLUTE
        020731  000000 2340 10         6789         SZN     0,0              CHECK IF LEXICOGRAPHICAL LEVEL CHANGE
        020732  020737 6050 00         6790         TPL     IDFY2            TRANSFER IF NO CHANGE
        020733  020746 0540 00         6791         AOS     IDFYC            INCREMENT LEVEL DIFFERENCE
        020734  000000 2200 10         6792         LDX0    0,0              GET POINTER TO SURROUNDING ENVIRONMENT
        020735  400000 6600 03         6793         ERX0    =0400000,DU      MAKE POINTER CORRECT
        020736  020730 7100 00         6794         TRA     IDFY1            AND LOOP
    END OF BINARY CARD 00000342
        020737  000000 2110 10         6795 IDFY2   CMK     0,0              SEE IF IDENTIFIER IS DECLARED IN THIS RANGE
        020740  000000 6000 00         6796 IDFYX   TZE     **               RETURN IF YES
        020741  020747 0540 00         6797         AOS     IDFYF            SET LL CHANGE FLAG
        020742  000000 2340 10         6798         SZN     0,0              SEE IF THIS IS THE OUTERMOST RANGE
        020743  777777 6000 00         6799         TZE     $ERROR           YES - COMPILER ERROR
        020744  000000 2200 10         6800         LDX0    0,0              GET POINTER TO SURROUNDING ENVIRONMENT
        020745  020730 7100 00         6801         TRA     IDFY1            AND LOOP
        020746  000000 000000          6802 IDFYC   ZERO
        020747  000000 000000          6803 IDFYF   ZERO
        020750  021005 7400 00         6804 VOID    STX0    VOIDX            SAVE RETURN
        020751  035234 2260 00         6805         LDX6    A$WORK           GET POINTER TO END OF WORKING STACK
        020752  000004 1660 03         6806         SBX6    1*WL,DU          GET POINTER TO LAST BLOCK IN WORKING STACK
        020753  000000 2250 16         6807         LDX5    0,6              GET FLAGS FROM BLOCK IN XR - 5
        020754  000001 2360 16         6808         LDQ     1,6              GET REGISTER TO DEALLOCATE
        020755  017665 7000 00         6809         TSX0    CL1              DEALLOCATE ANY DEREFERENCED VALUE
        020756  000003 2360 16         6810         LDQ     3,6              GET REGISTER TO DEALLOCATE
        020757  017673 7000 00         6811         TSX0    CL3              DEALLOCATE ANY OTHER REGISTERS
        020760  000400 3050 03         6812         CANX5   BSFJ,DU          IS ARRAY IN STACK BIT SET
        020761  020766 6000 00         6813         TZE     VOID1            NO - TRANSFER
        020762  000001 7200 16         6814         LXL0    1,6              GET ADDRESS OF OLD STACK POINTER IN STACK
        020763  021006 7400 00         6815         STX0    VOIDI            AND STORE IN INSTRUCTION
        020764  021006 2350 00         6816         LDA     VOIDI            GET INSTRUCTION IN A REGISTER
    END OF BINARY CARD 00000343
        020765  017474 7000 00         6817         TSX0    GAD              AND ADD TO GENERATED OUTPUT
        020766  000001 2200 16         6818 VOID1   LDX0    1,6              GET ADDRESS ALLOCATED FOR VALUE IN STACK
        020767  017037 7400 00         6819         STX0    SP               RESET STACK POINTER
        020770  000000 7270 16         6820         LXL7    0,6              GET MODE OF VALUE IN XR - 7
        020771  010562 7000 00         6821         TSX0    A$RCHK           SEE IF MODE CONTAINS ARRAYS
        020772  020777 7100 00         6822         TRA     VOID2            TRANSFER IF NO ARRAYS
        020773  035234 2260 00         6823         LDX6    A$WORK           GET POINTER TO END OF WORKING STACK
        020774  000004 1660 03         6824         SBX6    WL,DU            GET POINTER TO LAST BLOCK IN WORKING STACK
        020775  000001 7200 16         6825         LXL0    1,6              GET ADDRESS ASSIGNED FOR SAVED STACK POINTER
        020776  017037 7400 00         6826         STX0    SP               AND DEALLOCATE STACK POINTER WORD
        020777  035234 2260 00         6827 VOID2   LDX6    A$WORK           GET POINTER TO END OF WORKING STACK
        021000  000004 1660 03         6828         SBX6    WL,DU            GET POINTER TO LAST BLOCK IN WORKING STACK
        021001  000001 6200 00         6829         EAX0    M$VOID           GET VOID MODE IN XR - 0
```

```
          3                                                PASS 3

     021002  000000 4400 16         6830        SXL0    0,6              AND STORE IN BLOCK
     021003  100000 2250 03         6831        LDX5    BSFC,DU          GET VALUE IS STACKED BIT
     021004  000000 7450 16         6832        STX5    0,6              CLAIM THAT THE VOID IS IN THE STACK
     021005  000000 7100 00         6833 VOIDX  TRA     **               AND RETURN
     021006  000000 2260 17         6834 VOIDI  LDX     S,0,D
     021007  021014 7400 00         6835 DELV   STX0    DELVX            SAVE RETURN
     021010  020750 7000 00         6836        TSX0    VOID             VOID VALUE
     021011  035234 0110 54         6837 DELV1  NOP     ASWORK,DI        DELETE A WORD FROM THE WORKING STACK
     021012  035234 1060 00         6838        CMPX6   ASWORK           HAVE WE DELETED ENOUGH
   END OF BINARY CARD 00000344
     021013  021011 6010 00         6839        TNZ     DELV1            TRANSFER IF MORE TO DELETE
     021014  000000 7100 00         6840 DELVX  TRA     **               AND RETURN
     021015  021025 7400 00         6841 PRIM   STX0    PRIMX            SAVE RETURN
     021016  017045 7270 00         6842        LXL7    PARAM            GET MODE OF RESULT OF COERCION
     021017  000323 2210 03         6843        LDX1    IR,DU            GET POINTER TO INT TO REAL COERCION MACRO
     021020  000011 1070 03         6844        CMPX7   MSREAL,DU        IS RESULT MODE REAL
     021021  021024 6000 00         6845        TZE     PRIM1            YES - ALL SET UP SO TRANSFER
     021022  000330 2210 03         6846        LDX1    LIR,DU           GET POINTER TO LINT TO LREAL COERCION MACRO
     021023  777777 6010 00         6847        TNZ     SERROR           TRANSFER IF COERCION NOT IMPLEMENTED YET
     021024  017256 7000 00         6848 PRIM1  TSX0    OPE0             DO MACRO
     021025  000000 7100 00         6849 PRIMX  TRA     **               AND RETURN
     021026  000000 7100 10         6850 ASGN   TRA     0,0              NOTHING TO DO TO SET UP ASSIGNATION
     021027  021040 7400 00         6851 ASGNE  STX0    ASGNX            SAVE RETURN
     021030  021554 4500 00         6852        STZ     ASL              ZERO OUT OFFSET
     021031  025621 2210 03         6853        LDX1    RSACHEK,DU       GET ASSIGN SUBROUTINE
     021032  021524 7410 00         6854        STX1    ASI8             AND STORE FOR ASSIGN ROUTINE
     021033  035234 2260 00         6855        LDX6    ASWORK           GET POINTER TO END OF WORKING STACK
     021034  000004 1660 03         6856        SBX6    WL,DU            GET POINTER TO LAST BLOCK IN WORKING STACK
     021035  000000 7270 16         6857        LXL7    0,6              GET MODE OF BLOCK IN XR - 7
     021036  021041 7000 00         6858        TSX0    ASG              ASSIGN VALUE TO NAME
     021037  021007 7000 00         6859        TSX0    DELV             DELETE SOURCE BLOCK FROM WORKING STACK
     021040  000000 7100 00         6860 ASGNX  TRA     **               AND RETURN
   END OF BINARY CARD 00000345
     021041  035235 7400 56         6861 ASG    STX0    ASSTACK,ID       SAVE RETURN
     021042  005742 7170 00         6862        XED     TSSOVF           CHECK FOR STACK OVERFLOW
     021043  021553 7470 00         6863        STX7    ASGM             SAVE MODE OF ASSIGNATION
     021044  010630 7000 00         6864        TSX0    ASXFER           MAKE MODE POINTER ABSOLUTE
     021045  000000 2200 17         6865        LDX0    0,7              GET TYPE OF MODE IN XR - 0
     021046  016757 1000 03         6866        CMPX0   MSSTRCT,DU       IS IT A STRUCTURED MODE
     021047  021072 6010 00         6867        TNZ     ASG1             TRANSFER IF NOT STRUCTURED
     021050  000001 2210 03         6868        LDX1    1,DU             GET A RELATIVE POINTER TO FIRST STRUCTURE FIELD
     021051  035216 1670 00         6869 AST1   SBX7    TSMODE           MAKE STRUCTURE MODE POINTER RELATIVE
     021052  035235 7470 56         6870        STX7    ASSTACK,ID       SAVE STRUCTURE MODE IN STACK
     021053  005742 7170 00         6871        XED     TSSOVF           CHECK FOR STACK OVERFLOW
     021054  035235 7410 56         6872        STX1    ASSTACK,ID       SAVE FIELD POINTER IN STACK
     021055  005742 7170 00         6873        XED     TSSOVF           CHECK FOR STACK OVERFLOW
     021056  035216 0670 00         6874        ADX7    TSMODE           GET ABSOLUTE STRUCTURE MODE POINTER IN XR - 7
     021057  021060 7410 00         6875        STX1    AST2             STORE FIELD POINTER FOR ADDING
     021060  000000 0670 03         6876 AST2   ADX7    **,DU            GET POINTER TO CURRENT FIELD OF STRUCTURE
     021061  000000 2270 17         6877        LDX7    0,7              GET MODE OF FIELD IN XR - 7
```

```
      021062  021041 7000 00        6878          TSX0     ASG              ASSIGN FIELD
      021063  035235 2210 54        6879          LDX1     A$STACK,DI       GET FIELD POINTER IN XR - 1
      021064  035235 2270 54        6880          LDX7     A$STACK,DI       GET STRUCTURE MODE IN XR - 7
      021065  035216 0670 00        6881          ADX7     T$MODE           MAKE MODE POINTER ABSOLUTE
      021066  000001 0610 03        6882          ADX1     1,DU             STEP FIELD POINTER TO NEXT FIELD
   END OF BINARY CARD 00000346
      021067  777777 1010 17        6883          CMPX1    -1,7             SEE IF ANY MORE FIELDS
      021070  021051 6010 00        6884          TNZ      AST1             TRANSFER IF MORE FIELDS IN STRUCTURE
      021071  021511 7100 00        6885          TRA      ASGR             AND GO TO RETURN
      021072  016770 1000 03        6886 ASG1     CMPX0    M$ROW,DU         IS IT A ROW VALUE
      021073  021076 6000 00        6887          TZE      ASG2             TRANSFER IF ROW VALUE
      021074  016776 1000 03        6888          CMPX0    M$ROWE,DU        IS IT A ROW VALUE
      021075  021465 6010 00        6889          TNZ      ASG3             TRANSFER IF NOT A ROW VALUE
      021076  000033 6270 00        6890 ASG2     EAX7     M$MS             GET MODE OF MARK STACK IN XR - 7
      021077  020377 7000 00        6891          TSX0     MBLK             AND MAKE A BLOCK FOR MARK STACK
      021100  000001 2350 16        6892          LDA      1,6              GET ADDRESS OF MARK STACK IN AU
      021101  021520 7510 70        6893          STCA     ASI5,70          AND STORE IN INSTRUCTION SEQUENCE
      021102  020526 7000 00        6894          TSX0     STYPE            MARK THE STACK
      021103  035234 2260 00        6895          LDX6     A$WORK           GET POINTER TO END OF WORKING STACK
      021104  000004 1660 03        6896          SBX6     WL,DU            GET POINTER TO MARK STACK BLOCK
      021105  000001 2350 16        6897          LDA      1,6              GET ADDRESS OF MARK STACK IN AU
      021106  000001 0750 03        6898          ADA      1,DU             GET POINTER TO SAVED S WORD
      021107  021513 7510 70        6899          STCA     ASI0,70          AND STORE IN INSTRUCTION SEQUENCE
      021110  000004 1660 03        6900          SBX6     WL,DU            GET POINTER TO SOURCE BLOCK
      021111  020204 7000 00        6901          TSX0     MVA              MAKE SOURCE DESCRIPTOR AVAILABLE
      021112  777777 7100 00        6902          TRA      $ERROR           CAN NOT BE IN ACCUMULATOR
      021113  021554 0750 00        6903          ADA      ASL              ADD OFFSET
      021114  021516 7510 71        6904          STCA     ASI3,71          STORE DESCRIPTOR ADDRESS IN INSTRUCTION SEQUENCE
   END OF BINARY CARD 00000347
      021115  000004 1660 03        6905          SBX6     WL,DU            GET POINTER TO DESTINATION BLOCK
      021116  021524 2200 00        6906          LDX0     ASI8             GET TYPE OF ASSIGNATION FLAG IN XR - 0
      021117  025621 1000 03        6907          CMPX0    R$ACHEK,DU       IS IT A REGULAR ASSIGNATION
      021120  021123 6010 00        6908          TNZ      ASG22            TRANSFER IF NOT
      021121  020066 7000 00        6909          TSX0     MNA              MAKE DESTINATION NAME AVAILABLE
      021122  021125 7100 00        6910          TRA      ASG23            AND CONTINUE
      021123  020204 7000 00        6911 ASG22    TSX0     MVA              MAKE DESTINATION POINTER AVAILABLE
      021124  777777 7100 00        6912          TRA      $ERROR           COMPILER ERROR - NEVER IN ACCUMULATOR
      021125  021554 0750 00        6913 ASG23    ADA      ASL              ADD OFFSET
      021126  021514 7510 71        6914          STCA     ASI1,71          STORE DESTINATION ADDRESS IN INSTRUCTION SEQUENCE
      021127  020052 7000 00        6915          TSX0     PALL             PURGE ALL REGISYERS
      021130  000037 6270 00        6916          EAX7     M$PTR            GET MODE OF POINTER IN XR - 7
      021131  020377 7000 00        6917          TSX0     MBLK             MAKE A BLOCK FOR A POINTER
      021132  100000 2250 03        6918          LDX5     B$FC,DU          GET VALUE IS STACKED BIT
      021133  000000 7450 16        6919          STX5     0,6              AND STORE IN FLAGS OF BLOCK
      021134  000001 2350 16        6920          LDA      1,6              GET ADDRESS OF POINTER IN AU
      021135  021515 7510 70        6921          STCA     ASI2,70          AND STORE IN INSTRUCTION SEQUENCE
      021136  000037 6270 00        6922          EAX7     M$PTR            GET MODE OF POINTER IN XR - 7
      021137  020377 7000 00        6923          TSX0     MBLK             MAKE A BLOCK FOR A POINTER
      021140  100000 2250 03        6924          LDX5     B$FC,DU          GET VALUE IS STACKED BIT
      021141  000000 7450 16        6925          STX5     0,6              AND STORE IN FLAGS OF BLOCK
```

```
     021142  000001 2350 16        6926       LDA     1,6               GET ADDRESS OF POINTER IN AU
 END OF BINARY CARD 00000348
     021143  021517 7510 70        6927       STCA    ASI4,70           AND STORE IN INSTRUCTION SEQUENCE
     021144  021553 2270 00        6928       LDX7    ASGM              GET MODE OF ROW VALUE IN XR - 7
     021145  000000 2210 03        6929       LDX1    0,DU              INITIALIZE COUNT TO ZERO
     021146  000001 0610 03        6930 ASR1  ADX1    1,DU              INCREMENT DIMENSION OF ROW MODE
     021147  010630 7000 00        6931       TSX0    A$XFER            MAKE MODE ABSOLUTE
     021150  000000 2200 17        6932       LDX0    0,7               GET TYPE OF MODE IN XR - 0
     021151  000001 2270 17        6933       LDX7    1,7               GET DEROWED MODE IN XR - 7
     021152  016776 1000 03        6934       CMPX0   M$ROWE,DU         IS THIS THE LAST ROW MODE
     021153  021146 6010 00        6935       TNZ     ASR1              NO - LOOP
     021154  021555 7470 00        6936       STX7    ASE               SAVE MODE OF ELEMENT
     021155  021522 7410 00        6937       STX1    ASI7              STORE DIMENSION IN INSTRUCTION SEQUENCE
     021156  021556 4410 00        6938       SXL1    ASD               SAVE DIMENSION OF ARRAY
     021157  010630 7000 00        6939       TSX0    A$XFER            MAKE MODE POINTER ABSOLUTE
     021160  777777 7200 17        6940       LXL0    -1,7              GET LENGTH OF ELEMENT VALUE IN XR - 0
     021161  021521 7400 00        6941       STX0    ASI6              AND STORE IN INSTRUCTION SEQUENCE
     021162  021523 2350 00        6942       LDA     ASIT1             GET TALLY WORD FOR INSTRUCTION SEQUENCE
     021163  017507 7000 00        6943       TSX0    GADL              AND ADD INSTRUCTIONS TO GENERATED OUTPUT
     021164  021555 2270 00        6944       LDX7    ASE               GET MODE OF ELEMENT IN XR - 7
     021165  022103 7000 00        6945       TSX0    MTL               GET TYPE OF ELEMENT IN AU
     021166  624000 0750 07        6946       ADA     EAX4,DL           MAKE EAX4 TYPE INSTRUCTION
     021167  017474 7000 00        6947       TSX0    GAD               AND ADD TO GENERATED OUTPUT
     021170  021524 2350 00        6948       LDA     ASI8              GET TSX0 TO FIX/FLEX CLEANUP ROUTINE
 END OF BINARY CARD 00000349
     021171  017474 7000 00        6949       TSX0    GAD               AND ADD TO GENERATED OUTPUT
     021172  021007 7000 00        6950       TSX0    DELV              DELETE POINTER BLOCK
     021173  021007 7000 00        6951       TSX0    DELV              DELETE POINTER BLOCK
     021174  035225 7220 00        6952       LXL2    TSLBL             GET NEXT LABEL TO BE ASSIGNED
     021175  017045 4420 00        6953       SXL2    PARAM             STORE AS PARAMETER
     021176  021557 7420 00        6954       STX2    ASLBL             STORE DONE LABEL
     021177  035225 2210 03        6955       LDX1    TSLBL,DU          GET POINTER TO LABEL TABLE CONTROL WORD IN XR - 1
     021200  000000 6350 00        6956       EAA     1-1               ALLOCATE ONE WORD
     021201  005663 7000 00        6957       TSX0    TSALOC            ALLOCATE ONE WORD IN LABEL TABLE
     021202  777777 4500 11        6958       STZ     -1,1              ZERO OUT ALLOCATED WORD
     021203  022017 7000 00        6959       TSX0    TRAD              GET CHAIN ADDRESS IN AU
     021204  710004 0750 07        6960       ADA     TRA+IC,DL         ADD TRA 0,IC INSTRUCTION
     021205  017474 7000 00        6961       TSX0    GAD               AND ADD TO GENERATED OUTPUT
     021206  021556 3350 00        6962       LCA     ASD               GET MINUS NUMBER OF DIMENSIONS FOR COUNT
     021207  021560 7550 00        6963       STA     ASTP              AND STORE FOR COUNTING
     021210  000001 2200 03        6964       LDX0    1,DU              GET OFFSET FOR QUADRUPLE
     021211  021561 7400 00        6965       STX0    ASRX1             AND STORE
     021212  000042 6270 00        6966 ASR2  EAX7    M$QUAD            GET MODE OF QUADRUPLE IN XR - 7
     021213  020377 7000 00        6967       TSX0    MBLK              AND MAKE A QUADRUPLE BLOCK
     021214  021561 2350 00        6968       LDA     ASRX1             GET ADDRESS OF CURRENT QUADRUPLE
     021215  000001 2360 16        6969       LDQ     1,6               GET ADDRESS TO MOVE TO
     021216  000000 6360 02        6970       EAQ     0,QU              ZERO OUT QL
 END OF BINARY CARD 00000350
     021217  000017 0760 07        6971       ADQ     D,DL              ADD XR - D MODIFICATION
     021220  000042 6270 00        6972       EAX7    M$QUAD            GET MODE OF QUAD IN XR - 7
```

```
          3                                            PASS 3

     021221  017531 7000 00          6973          TSX0     MOVE             MOVE QUADRUPLE
     021222  000004 2200 03          6974          LDX0     4,DU             GET LENGTH OF QUADRUPLE IN XR - 0
     021223  021561 0400 00          6975          ASX0     ASRX1            STEP OFFSET TO NEXT QUADRUPLE
     021224  021560 0540 00          6976          AOS      ASTP             COUNT REMAINING DIMENSIONS
     021225  021212 6010 00          6977          TNZ      ASR2             TRANSFER IF MORE DIMENSIONS
     021226  035234 2260 00          6978          LDX6     A$WORK           GET ADDRESS AT END OF WORKING STACK
     021227  035214 1660 00          6979          SBX6     T$WORK           MAKE POINTER RELATIVE
     021230  021563 7460 00          6980          STX6     ASQ1             AND SAVE
     021231  021556 3350 00          6981          LCA      ASD              GET MINUS NUMBER OF DIMENSIONS FOR COUNT
     021232  021560 7550 00          6982          STA      ASTP             AND STORE FOR COUNTING
     021233  000001 2200 03          6983          LDX0     1,DU             GET OFFSET FOR QUADRUPLE
     021234  021562 7400 00          6984          STX0     ASRX2            AND STORE
     021235  000042 6270 00          6985 ASR3     EAX7     M$QUAD           GET MODE OF QUADRUPLE IN XR - 7
     021236  020377 7000 00          6986          TSX0     MBLK             AND MAKE A QUADRUPLE BLOCK
     021237  021562 2350 00          6987          LDA      ASRX2            GET ADDRESS OF CURRENT QUADRUPLE
     021240  000001 2360 16          6988          LDQ      1,6              GET ADDRESS TO MOVE TO
     021241  000000 6360 02          6989          EAQ      0,QU             ZERO OUT QL
     021242  000017 0760 07          6990          ADQ      D,DL             ADD XR - D MODIFICATION
     021243  000042 6270 00          6991          EAX7     M$QUAD           GET MODE OF QUAD IN XR - 7
     021244  017531 7000 00          6992          TSX0     MOVE             MOVE QUADRUPLE
   END OF BINARY CARD 00000351
     021245  000004 2200 03          6993          LDX0     4,DU             GET LENGTH OF QUADRUPLE IN XR - 0
     021246  021562 0400 00          6994          ASX0     ASRX2            STEP OFFSET TO NEXT QUADRUPLE
     021247  021560 0540 00          6995          AOS      ASTP             COUNT REMAINING DIMENSIONS
     021250  021235 6010 00          6996          TNZ      ASR3             TRANSFER IF MORE DIMENSIONS
     021251  035234 2260 00          6997          LDX6     A$WORK           GET ADDRESS AT END OF WORKING STACK
     021252  035214 1660 00          6998          SBX6     T$WORK           MAKE POINTER RELATIVE
     021253  021564 7460 00          6999          STX6     ASQ2             AND SAVE
     021254  021556 3350 00          7000          LCA      ASD              GET MINUS NUMBER OF DIMENSIONS FOR COUNTING
     021255  021560 7550 00          7001          STA      ASTP             AND STORE FOR COUNTING
     021256  000007 6270 00          7002 ASR4     EAX7     M$INT            GET MODE OF PLACE IN XR - 7
     021257  020377 7000 00          7003          TSX0     MBLK             MAKE A BLOCK FOR PLACE
     021260  000001 2350 16          7004          LDA      1,6              GET ADDRESS OF PLACE IN AU
     021261  000000 6350 01          7005          EAA      0,AU             ZERO OUT AL
     021262  450017 0750 07          7006          ADA      STZ+D,DL         ADD STZ 0,D INSTRUCTION
     021263  017474 7000 00          7007          TSX0     GAD              AND ADD TO GENERATED OUTPUT
     021264  021560 0540 00          7008          AOS      ASTP             COUNT REMAINING DIMENSIONS
     021265  021256 6010 00          7009          TNZ      ASR4             TRANSFER IF MORE DIMENSIONS
     021266  000037 6270 00          7010          EAX7     M$PTR            GET MODE OF POINTER IN XR - 7
     021267  020377 7000 00          7011          TSX0     MBLK             AND MAKE A BLOCK
     021270  100000 2250 03          7012          LDX5     BSFC,DU          GET VALUE IS STACKED BIT
     021271  000000 7450 16          7013          STX5     0,6              AND STORE IN BLOCK
     021272  000001 2350 16          7014          LDA      1,6              GET ADDRESS OF POINTER IN AU
   END OF BINARY CARD 00000352
     021273  021526 7510 70          7015          STCA     ASI11,70         AND STORE IN INSTRUCTION SEQUENCE
     021274  000037 6270 00          7016          EAX7     M$PTR            GET MODE OF POINTER IN XR - 7
     021275  020377 7000 00          7017          TSX0     MBLK             AND MAKE A BLOCK
     021276  400000 2250 03          7018          LDX5     BSFA,DU          GET VALUE IS POINTED TO BY STACKED POINTER BIT
     021277  000000 7450 16          7019          STX5     0,6              AND STORE IN BLOCK
     021300  000001 2350 16          7020          LDA      1,6              GET ADDRESS OF POINTER IN AU
     021301  021530 7510 70        7021        STCA      ASI12,70          AND STORE IN INSTRUCTION SEQUENCE
     021302  021531 2350 00        7022        LDA       ASIT2             GET TALLY WORD FOR INSTRUCTION SEQUENCE
     021303  017507 7000 00        7023        TSX0      GADL              AND ADD SEQUENCE TO GENERATED OUTPUT
     021304  035226 7200 00        7024        LXL0      T$GEN             GET ADDRESS OF NEXT GENERATED INSTRUCTION
     021305  035235 7400 56        7025        STX0      A$STACK,ID        AND STORE IN STACK
     021306  005742 7170 00        7026        XED       T$SOVF            CHECK FOR STACK OVERFLOW
     021307  021563 2200 00        7027        LDX0      ASQ1              GET POINTER TO END OF DESTINATION QUADS
     021310  035235 7400 56        7028        STX0      A$STACK,ID        AND STORE IN STACK
     021311  005742 7170 00        7029        XED       T$SOVF            CHECK FOR STACK OVERFLOW
     021312  021564 2200 00        7030        LDX0      ASQ2              GET POINTER TO END OF SOURCE QUADS
     021313  035235 7400 56        7031        STX0      A$STACK,ID        AND STORE IN STACK
     021314  005742 7170 00        7032        XED       T$SOVF            CHECK FOR STACK OVERFLOW
     021315  021557 2200 00        7033        LDX0      ASLBL             GET DONE LABEL IN XR = 0
     021316  035235 7400 56        7034        STX0      A$STACK,ID        AND STORE IN STACK
     021317  005742 7170 00        7035        XED       T$SOVF            CHECK FOR STACK OVERFLOW
     021320  021553 2270 00        7036        LDX7      ASGM              GET MODE OF ROW VALUE
   END OF BINARY CARD 00000353
     021321  035235 7470 56        7037        STX7      A$STACK,ID        AND STORE IN STACK
     021322  005742 7170 00        7038        XED       T$SOVF            CHECK FOR STACK OVERFLOW
     021323  021554 2350 00        7039        LDA       ASL               GET CURRENT OFFSET
     021324  035235 7550 56        7040        STA       A$STACK,ID        AND STORE IN STACK
     021325  005742 7170 00        7041        XED       T$SOVF            CHECK FOR STACK OVERFLOW
     021326  021554 4500 00        7042        STZ       ASL               RESET OFFSET FOR ELEMENT ROUTINES
     021327  021555 2270 00        7043        LDX7      ASE               GET MODE OF ELEMENT IN XR = 7
     021330  035235 7470 56        7044        STX7      A$STACK,ID        AND STORE IN STACK
     021331  005742 7170 00        7045        XED       T$SOVF            CHECK FOR STACK OVERFLOW
     021332  021556 2350 00        7046        LDA       ASD               GET DIMENSION OF ROW VALUE
     021333  035235 7550 56        7047        STA       A$STACK,ID        AND STORE IN STACK
     021334  005742 7170 00        7048        XED       T$SOVF            CHECK FOR STACK OVERFLOW
     021335  021041 7000 00        7049        TSX0      ASG               ASSIGN AN ELEMENT OF ARRAY
     021336  035235 2350 54        7050        LDA       A$STACK,DI        GET DIMENSION OF ARRAY
     021337  021556 7550 00        7051        STA       ASD               AND RESTORE
     021340  035235 2270 54        7052        LDX7      A$STACK,DI        GET MODE OF ELEMENT
     021341  021555 7470 00        7053        STX7      ASE               AND RESTORE
     021342  035235 2350 54        7054        LDA       A$STACK,DI        GET OFFSET
     021343  021554 7550 00        7055        STA       ASL               AND RESTORE
     021344  035235 2270 54        7056        LDX7      A$STACK,DI        GET MODE OF ROW VALUE
     021345  021553 7470 00        7057        STX7      ASGM              AND RESTORE
     021346  035235 2200 54        7058        LDX0      A$STACK,DI        GET DONE LABEL
   END OF BINARY CARD 00000354
     021347  021557 7400 00        7059        STX0      ASLBL             AND RESTORE
     021350  035235 2200 54        7060        LDX0      A$STACK,DI        GET POINTER TO END OF SOURCE QUADS
     021351  021564 7400 00        7061        STX0      ASQ2              AND RESTORE
     021352  035235 2200 54        7062        LDX0      A$STACK,DI        GET POINTER TO END OF DESTINATION QUADS
     021353  021563 7400 00        7063        STX0      ASQ1              AND RESTORE
     021354  035235 2200 54        7064        LDX0      A$STACK,DI        GET ADDRESS OF MOVE ELEMENT ROUTINE
     021355  021565 7400 00        7065        STX0      ASMLB             AND STORE
     021356  035234 2260 00        7066        LDX6      A$WORK            GET POINTER TO END OF WORKING STACK
     021357  000004 1660 03        7067        SBX6      WL,DU             GET POINTER TO SOURCE POINTER
     021360  000001 2350 16        7068        LDA       1,6               GET ADDRESS OF SOURCE POINTER IN AU
```

```
      021361  021536 7510 70        7069        STCA      ASI24,70          STORE IN INSTRUCTION SEQUENCE
      021362  021550 7510 70        7070        STCA      ASI36,70          STORE IN INSTRUCTION SEQUENCE
      021363  000004 1660 03        7071        SBX6      WL,DU             GET POINTER TO DESTINATION POINTER
      021364  000001 2350 16        7072        LDA       1,6               GET ADDRESS OF DESTINATION POINTER IN AU
      021365  021534 7510 70        7073        STCA      ASI22,70          STORE IN INSTRUCTION SEQUENCE
      021366  021545 7510 70        7074        STCA      ASI33,70          STORE IN INSTRUCTION SEQUENCE
      021367  000004 1660 03        7075        SBX6      WL,DU             GET POINTER TO LAST PLACE BLOCK
      021370  000001 2350 16        7076        LDA       1,6               GET ADDRESS OF LAST PLACE IN AU
      021371  021532 7510 70        7077        STCA      ASI20,70          STORE IN INSTRUCTION SEQUENCE
      021372  021541 7510 70        7078        STCA      ASI27,70          STORE IN INSTRUCTION SEQUENCE
      021373  021543 7510 70        7079        STCA      ASI31,70          STORE IN INSTRUCTION SEQUENCE
      021374  021546 7510 70        7080        STCA      ASI34,70          STORE IN INSTRUCTION SEQUENCE
  END OF BINARY CARD 00000355
      021375  021551 7510 70        7081        STCA      ASI37,70          STORE IN INSTRUCTION SEQUENCE
      021376  021563 2260 00        7082        LDX6      ASQ1              GET RELATIVE POINTER TO END OF DESTINATION QUADS
      021377  035214 0660 00        7083        ADX6      T$WORK            MAKE POINTER ABSOLUTE
      021400  000004 1660 03        7084        SBX6      WL,DU             GET POINTER TO LAST DESTINATION QUAD IN XR - 6
      021401  000001 2350 16        7085        LDA       1,6               GET ADDRESS OF LAST DESTINATION QUAD IN AU
      021402  021540 7510 70        7086        STCA      ASI26,70          STORE IN INSTRUCTION SEQUENCE
      021403  000001 0750 03        7087        ADA       1,DU              GET UPPER BOUND ADDRESS
      021404  021537 7510 70        7088        STCA      ASI25,70          STORE IN INSTRUCTION SEQUENCE
      021405  000001 0750 03        7089        ADA       1,DU              GET STRIDE ADDRESS
      021406  021533 7510 70        7090        STCA      ASI21,70          STORE IN INSTRUCTION SEQUENCE
      021407  021544 7510 70        7091        STCA      ASI32,70          STORE IN INSTRUCTION SEQUENCE
      021410  021564 2260 00        7092        LDX6      ASQ2              GET RELATIVE POINTER TO LAST SOURCE QUAD
      021411  035214 0660 00        7093        ADX6      T$WORK            MAKE POINTER ABSOLUTE
      021412  000004 1660 03        7094        SBX6      WL,DU             GET POINTER TO LAST SOURCE QUAD IN XR - 6
      021413  000001 2350 16        7095        LDA       1,6               GET LOWER BOUND ADDRESS
      021414  000002 0750 03        7096        ADA       2,DU              GET STRIDE ADDRESS
      021415  021535 7510 70        7097        STCA      ASI23,70          STORE IN INSTRUCTION SEQUENCE
      021416  021547 7510 70        7098        STCA      ASI35,70          STORE IN INSTRUCTION SEQUENCE
      021417  035226 7200 00        7099        LXL0      T$GEN             GET ADDRESS OF NEXT INSTRUCTION TO BE GENERATED
      021420  000010 0600 03        7100        ADX0      ASI30-ASI20,DU    GET ADDRESS OF TPL INSTRUCTION
      021421  021542 7400 00        7101        STX0      ASI30             STORE IN INSTRUCTION SEQUENCE
      021422  021565 2200 00        7102        LDX0      ASMLB             GET ADDRESS WHERE TO TRANSFER
  END OF BINARY CARD 00000356
      021423  021542 1400 00        7103        SSX0      ASI30             SET TRANSFER ADDRESS IN INSTRUCTION SEQUENCE
      021424  021556 3350 00        7104        LCA       ASD               GET MINUS NUMBER OF DIMENSIONS
      021425  021560 7550 00        7105        STA       ASTP              AND STORE FOR COUNTING
      021426  021552 2350 00        7106 ASR5   LDA       ASIT3             GET TALLY WORD FOR INSTRUCTION SEQUENCE
      021427  017507 7000 00        7107        TSX0      GADL              AND ADD TO GENERATED OUTPUT
      021430  021560 0540 00        7108        AOS       ASTP              COUNT REMAINING DIMENSIONS
      021431  021452 6000 00        7109        TZE       ASR6              TRANSFER IF DONE
      021432  000004 3200 03        7110        LCX0      4,DU              GET MINUS LENGTH OF QUAD IN XR - 0
      021433  021533 0400 00        7111        ASX0      ASI21             UPDATE CODE FOR NEXT DIMENSION
      021434  021535 0400 00        7112        ASX0      ASI23             UPDATE CODE FOR NEXT DIMENSION
      021435  021537 0400 00        7113        ASX0      ASI25             UPDATE CODE FOR NEXT DIMENSION
      021436  021540 0400 00        7114        ASX0      ASI26             UPDATE CODE FOR NEXT DIMENSION
      021437  021544 0400 00        7115        ASX0      ASI32             UPDATE CODE FOR NEXT DIMENSION
      021440  021547 0400 00        7116        ASX0      ASI35             UPDATE CODE FOR NEXT DIMENSION
```

```
     021441  000001 3200 03        7117         LCX0     1,DU              GET MINUS ONE IN XR - 0
     021442  021532 0400 00        7118         ASX0     ASI20             UPDATE CODE FOR NEXT DIMENSION
     021443  021541 0400 00        7119         ASX0     ASI27             UPDATE CODE FOR NEXT DIMENSION
     021444  021543 0400 00        7120         ASX0     ASI31             UPDATE CODE FOR NEXT DIMENSION
     021445  021546 0400 00        7121         ASX0     ASI34             UPDATE CODE FOR NEXT DIMENSION
     021446  021551 0400 00        7122         ASX0     ASI37             UPDATE CODE FOR NEXT DIMENSION
     021447  000020 3200 03        7123         LCX0     ASIT3-ASI20,DU    GET MINUS LENGTH OF INSTRUCTION SEQUENCE
     021450  021542 0400 00        7124         ASX0     ASI30             UPDATE CODE FOR NEXT DIMENSION
  END OF BINARY CARD 00000357
     021451  021426 7100 00        7125         TRA      ASR5              AND LOOP
     021452  021556 3350 00        7126 ASR6    LCA      ASD               GET MINUS NUMBER OF DIMENSIONS
     021453  021560 7550 00        7127         STA      ASTP              AND STORE FOR COUNTING
     021454  023663 7000 00        7128 ASR7    TSX0     POP               DELETE WORKING STACK BACK TO MARK STACK
     021455  021553 2270 00        7129         LDX7     ASGM              GET MODE OF ARRAY VALUE IN XR - 7
     021456  010630 7000 00        7130         TSX0     A$XFER            MAKE MODE POINTER ABSOLUTE
     021457  777777 7200 17        7131         LXL0     -1,7              GET LENGTH OF VALUE IN XR - 0
     021460  021554 0400 00        7132         ASX0     ASL               INCREMENT OFFSET BY DESCRIPTOR LENGTH
     021461  021557 2200 00        7133         LDX0     ASLBL             GET DONE LABEL
     021462  017045 4400 00        7134         SXL0     PARAM             AND STORE AS PARAMETER
     021463  022006 7000 00        7135         TSX0     LBL               DEFINE LABEL
     021464  021511 7100 00        7136         TRA      ASGR              AND GO TO RETURN
     021465  035234 2260 00        7137 ASG3    LDX6     A$WORK            GET POINTER TO END OF WORKING STACK
     021466  000010 1660 03        7138         SBX6     2*WL,DU           GET POINTER TO DESTINATION BLOCK
     021467  020066 7000 00        7139         TSX0     MNA               MAKE DESTINATION NAME AVAILABLE
     021470  021566 7550 00        7140         STA      ASGT              AND SAVE IN TEMP
     021471  000004 0660 03        7141         ADX6     WL,DU             GET POINTER TO SOURCE BLOCK
     021472  020204 7000 00        7142         TSX0     MVA               MAKE SOURCE VALUE AVAILABLE
     021473  021506 7100 00        7143         TRA      ASG4              TRANSFER IF VALUE IS IN A REGISTER
     021474  021566 2360 00        7144         LDQ      ASGT              GET DESTINATION IN Q REGISTER
     021475  021553 2270 00        7145         LDX7     ASGM              GET MODE IN XR - 7
     021476  021554 0750 00        7146         ADA      ASL               ADD OFFSET TO SOURCE ADDRESS
  END OF BINARY CARD 00000358
     021477  021554 0760 00        7147         ADQ      ASL               ADD OFFSET TO DESTINATION ADDRESS
     021500  017531 7000 00        7148         TSX0     MOVE              MOVE VALUE FROM SOURCE TO DESTINATION
     021501  021553 2270 00        7149         LDX7     ASGM              GET MODE OF VALUE IN XR - 7
     021502  010630 7000 00        7150         TSX0     A$XFER            MAKE MODE POINTER ABSOLUTE
     021503  777777 7200 17        7151         LXL0     -1,7              GET LENGTH OF VALUE IN XR - 0
     021504  021554 0400 00        7152         ASX0     ASL               INCREMENT OFFSET BY LENGTH OF VALUE
     021505  021511 7100 00        7153         TRA      ASGR              AND GO TO RETURN
     021506  021566 0750 00        7154 ASG4    ADA      ASGT              ADD ADDRESS OF DESTINATION TO STORE COMMAND
     021507  021554 0750 00        7155         ADA      ASL               ADD OFFSET
     021510  017474 7000 00        7156         TSX0     GAD               AND ADD TO GENERATED OUTPUT CODE
     021511  035235 2200 54        7157 ASGR    LDX0     A$STACK,DI        RESTORE RETURN ADDRESS IN XR - 0
     021512  000000 7100 10        7158         TRA      0,0               AND RETURN
     021513  000000 4500 17        7159 ASI0    STZ      0,D
     021514  000000 6350 00        7160 ASI1    EAA      0
     021515  000000 7550 17        7161 ASI2    STA      0,D
     021516  000000 6350 00        7162 ASI3    EAA      0
     021517  000000 7550 17        7163 ASI4    STA      0,D
     021520  000000 6210 17        7164 ASI5    EAX1     0,D
```

```
              3                                          PASS 3

     021521  000000 6220 00        7165 ASI6    EAX2     0
     021522  000000 6230 00        7166 ASI7    EAX3     0
     021523  021513 0011 00        7167 ASIT1   TALLY    ASI0,*-ASI0+1
     021524  025621 7000 00        7168 ASI8    TSX0     R$ACHEK
 END OF BINARY CARD 00000359
     021525  000000 2350 11        7169 ASI10   LDA      0,1
     021526  000000 7550 17        7170 ASI11   STA      0,D
     021527  000000 2350 12        7171         LDA      0,2
     021530  000000 7550 17        7172 ASI12   STA      0,D
     021531  021525 0005 00        7173 ASIT2   TALLY    ASI10,*-ASI10+1
     021532  000000 0540 17        7174 ASI20   AOS      0,D
     021533  000000 2360 17        7175 ASI21   LDQ      0,D
     021534  000000 0560 17        7176 ASI22   ASQ      0,D
     021535  000000 2360 17        7177 ASI23   LDQ      0,D
     021536  000000 0560 17        7178 ASI24   ASQ      0,D
     021537  000000 2360 17        7179 ASI25   LDQ      0,D
     021540  000000 1760 17        7180 ASI26   SBQ      0,D
     021541  000000 1160 17        7181 ASI27   CMPQ     0,D
     021542  000000 6050 04        7182 ASI30   TPL      0,IC
     021543  000000 3360 17        7183 ASI31   LCQ      0,D
     021544  000000 4020 17        7184 ASI32   MPY      0,D
     021545  000000 0560 17        7185 ASI33   ASQ      0,D
     021546  000000 3360 17        7186 ASI34   LCQ      0,D
     021547  000000 4020 17        7187 ASI35   MPY      0,D
     021550  000000 0560 17        7188 ASI36   ASQ      0,D
     021551  000000 4500 17        7189 ASI37   STZ      0,D
     021552  021532 0021 00        7190 ASIT3   TALLY    ASI20,*-ASI20+1
 END OF BINARY CARD 00000360
     021553  000000 000000         7191 ASGM    ZERO
     021554  000000 000000         7192 ASL     ZERO
     021555  000000 000000         7193 ASE     ZERO
     021556  000000 000000         7194 ASD     ZERO
     021557  000000 000000         7195 ASLBL   ZERO
     021560  000000 000000         7196 ASTP    ZERO
     021561  000000000011          7197 ASRX1   OCT      11
     021562  000000000012          7198 ASRX2   OCT      12
     021563  000000 000000         7199 ASQ1    ZERO
     021564  000000 000000         7200 ASQ2    ZERO
     021565  000000 000000         7201 ASMLB   ZERO
     021566  000000 000000         7202 ASGT    ZERO
     021567  021642 7400 00        7203 SELCT   STX0     SELX                SAVE RETURN
     021570  035234 2260 00        7204         LDX6     A$WORK              GET POINTER TO END OF WORKING STACK
     021571  000004 1660 03        7205         SBX6     1*WL,DU             GET POINTER TO LAST BLOCK IN WORKING STACK
     021572  000000 7270 16        7206         LXL7     0,6                 GET MODE OF VALUE IN XR - 7
     021573  021705 7000 00        7207         TSX0     SLCT                CALCULATE OFFSET OF DESIRED FIELD
     021574  020204 7000 00        7208         TSX0     MVA                 MAKE STRUCTURED VALUE AVAILABLE
     021575  777777 7100 00        7209         TRA      $ERROR              STRUCTURED VALUE CANNOT BE IN REGISTER
     021576  000000 6200 05        7210         EAX0     0,AL                GET MODIFICATION OF VALUE ADDRESS IN XR - 0
     021577  000017 1000 03        7211         CMPX0    D,DU                IS IT D REGISTER MODIFICATION
     021600  021633 6010 00        7212         TNZ      SEL2                TRANSFER IF NOT
 END OF BINARY CARD 00000361
     021601  000000 6200 01        7213           EAX0    0,AU              GET ADDRESS PART OF VALUE POINTER IN XR - 0
     021602  000001 1000 16        7214           CMPX0   1,6               SEE IF IT IS A STACKED VALUE
     021603  021621 6010 00        7215           TNZ     SEL1              TRANSFER IF NOT A STACKED VALUE
     021604  021643 7550 00        7216           STA     SELT              SAVE VALUE POINTER IN TEMP
     021605  021643 2360 00        7217           LDQ     SELT              AND PUT IT IN Q AS DESTINATION
     021606  021730 0750 00        7218           ADA     DELTA             ADD DELTA TO SOURCE ADDRESS
     021607  021731 2270 00        7219           LDX7    FMODE             GET MODE OF FIELD IN XR - 7
     021610  017531 7000 00        7220           TSX0    MOVE              MOVE FIELD TO STACK
     021611  000000 2250 16        7221           LDX5    0,6               GET FLAGS FROM BLOCK IN XR - 5
     021612  000001 2360 16        7222           LDQ     1,6               GET REGISTER TO DEALLOCATE
     021613  017665 7000 00        7223           TSX0    CL1               DEALLOCATE ANY DEREFERENCED VALUES
     021614  000003 2360 16        7224           LDQ     3,6               GET REGISTER TO DEALLOCATE
     021615  017673 7000 00        7225           TSX0    CL3               DEALLOCATE ALL OTHER REGISTERS
     021616  707777 3650 03        7226           ANX5    -1-B$FD-B$FE-B$FF,DU RESET ALL DISPLAY RELATIVE NAME FLAGS
     021617  000000 7450 16        7227           STX5    0,6               RESTORE FLAGS IN BLOCK
     021620  021642 7100 00        7228           TRA     SELX              GO TO RETURN
     021621  021730 2200 00        7229 SEL1      LDX0    DELTA             GET OFFSET FOR DESIRED FIELD
     021622  000002 0400 16        7230           ASX0    2,6               INCREMENT NAME POINTER IF ANY
     021623  000000 2250 16        7231           LDX5    0,6               GET FLAGS FROM BLOCK
     021624  000001 2360 16        7232           LDQ     1,6               GET REGISTER TO DEALLOCATE
     021625  017665 7000 00        7233           TSX0    CL1               DEALLOCATE ANY DEREFERENCED VALUES
     021626  677777 3650 03        7234           ANX5    -1-B$FC,DU        RESET STACKED FLAG
 END OF BINARY CARD 00000362
     021627  000003 2360 16        7235           LDQ     3,6               GET REGISTER TO DEALLOCATE
     021630  017673 7000 00        7236           TSX0    CL3               DEALLOCATE ANY REGISTER USED TO POINT TO VALUE
     021631  000000 7450 16        7237           STX5    0,6               RESTORE FLAGS IN BLOCK
     021632  021642 7100 00        7238           TRA     SELX              GO TO RETURN
     021633  021730 2200 00        7239 SEL2      LDX0    DELTA             GET OFFSET FOR DESIRED FIELD
     021634  000003 0400 16        7240           ASX0    3,6               INCREMENT REGISTER BASED POINTER
     021635  000000 2250 16        7241           LDX5    0,6               GET FLAGS FROM BLOCK IN XR - 5
     021636  000001 2360 16        7242           LDQ     1,6               GET REGISTER TO DEALLOCATE
     021637  017665 7000 00        7243           TSX0    CL1               DEALLOCATE ANY DEREFERENCED VALUES
     021640  607777 3650 03        7244           ANX5    -1-B$FC-B$FD-B$FE-B$FF,DU RESET STACKED AND D REGISTER FLAGS
     021641  000000 7450 16        7245           STX5    0,6               RESTORE FLAGS IN BLOCK
     021642  000000 7100 00        7246 SELX      TRA     **                AND RETURN
     021643  000000 000000         7247 SELT      ZERO
     021644  021704 7400 00        7248 RSLCT     STX0    RSLX              SAVE RETURN
     021645  035234 2260 00        7249           LDX6    A$WORK            GET POINTER TO END OF WORKING STACK
     021646  000004 1660 03        7250           SBX6    1*WL,DU           GET POINTER TO LAST BLOCK IN WORKING STACK
     021647  000000 7270 16        7251           LXL7    0,6               GET MODE OF VALUE IN XR - 7
     021650  010630 7000 00        7252           TSX0    A$XFER            MAKE MODE POINTER UNIQUE AND ABSOLUTE
     021651  000000 2200 17        7253           LDX0    0,7               GET TYPE OF MODE IN XR - 0
     021652  016762 1000 03        7254           CMPX0   M$REF,DU          IS IT A REFERENCE

     021653  777777 6010 00        7255           TNZ     $ERROR            COMPILER ERROR
     021654  000001 2270 17        7256           LDX7    1,7               GET DEREFERENCED MODE IN XR - 7
 END OF BINARY CARD 00000363
     021655  021705 7000 00        7257           TSX0    SLCT              CALCULATE OFFSET OF DESIRED FIELD
     021656  020066 7000 00        7258           TSX0    MNA               MAKE THE STRUCTURE NAME AVAILABLE
     021657  000000 6200 05        7259           EAX0    0,AL              GET MODIFICATION IN XR - 0
```

```
    021660  000017 1000 03        7260          CMPX0   D,DU              IS IT D REGISTER MODIFICATION
    021661  021674 6010 00        7261          TNZ     RSL1              TRANSFER IF NOT
    021662  021730 2200 00        7262          LDX0    DELTA             GET OFFSET IN XR - 0
    021663  000002 0400 16        7263          ASX0    2,6               INCREMENT NAME BY OFFSET
    021664  000000 2250 16        7264          LDX5    0,6               GET FLAGS FROM BLOCK IN XR - 5
    021665  000001 2360 16        7265          LDQ     1,6               GET REGISTER TO DEALLOCATE
    021666  017665 7000 00        7266          TSX0    CL1               DEALLOCATE ANY DEREFERENCED VALUES
    021667  677777 3650 03        7267          ANX5    -1-B$FC,DU        RESET STACKED VALUE FLAG
    021670  000003 2360 16        7268          LDQ     3,6               GET REGISTER TO DEALLOCATE
    021671  017673 7000 00        7269          TSX0    CL3               DEALLOCATE ANY REGISTERS USED
    021672  000000 7450 16        7270          STX5    0,6               RESTORE FLAGS IN BLOCK
    021673  021704 7100 00        7271          TRA     RSLX              GO TO EXIT
    021674  021730 2200 00        7272 RSL1     LDX0    DELTA             GET OFFSET IN XR - 0
    021675  000003 0400 16        7273          ASX0    3,6               ADD TO REGISTER RELATIVE POINTER
    021676  000002 0400 16        7274          ASX0    2,6               ADD TO ANY DISPLAY RELATIVE POINTER
    021677  000000 2250 16        7275          LDX5    0,6               GET FLAGS FROM BLOCK IN XR - 5
    021700  000001 2360 16        7276          LDQ     1,6               GET REGISTER TO DEALLOCATE
    021701  017665 7000 00        7277          TSX0    CL1               DEALLOCATE ANY DEREFERENCED VALUES
    021702  607777 3650 03        7278          ANX5    -1-B$FC-B$FD-B$FE-B$FF,DU RESET STACKED AND D REGISTER FLAGS
END OF BINARY CARD 00000364
    021703  000000 7450 16        7279          STX5    0,6               RESTORE FLAGS IN BLOCK
    021704  000000 7100 00        7280 RSLX     TRA     **                AND RETURN
    021705  021727 7400 00        7281 SLCT     STX0    SLCTX             SAVE RETURN
    021706  010630 7000 00        7282          TSX0    A$XFER            MAKE MODE POINTER UNIQUE AND ABSOLUTE
    021707  000000 2200 17        7283          LDX0    0,7               GET TYPE OF MODE IN XR - 0
    021710  016757 1000 03        7284          CMPX0   M$STRCT,DU        IS IT A STRUCTURED MODE
    021711  777777 6010 00        7285          TNZ     $ERROR            NO - COMPILER ERROR
    021712  000001 6220 17        7286          EAX2    1,7               GET POINTER TO FIRST FIELD OF STRUCTURE
    021713  021730 4500 00        7287          STZ     DELTA             INITIALIZE DELTA OFFSET
    021714  017045 7210 00        7288          LXL1    PARAM             GET FIELD NUMBER IN XR - 1
    021715  000001 1610 03        7289 SLCT1    SBX1    1,DU              DECREMENT NUMBER OF FIELDS TO GO
    021716  021725 6000 00        7290          TZE     SLCT2             TRANSFER IF ALL FIELDS CONSIDERED
    021717  000000 2270 12        7291          LDX7    0,2               GET MODE OF FIELD IN XR - 7
    021720  010630 7000 00        7292          TSX0    A$XFER            MAKE MODE POINTER UNIQUE AND ABSOLUTE
    021721  777777 7200 17        7293          LXL0    -1,7              GET LENGTH OF VALUE OF MODE IN XR - 0
    021722  021730 0400 00        7294          ASX0    DELTA             AND ADD IT TO DELTA OFFSET
    021723  000001 0620 03        7295          ADX2    1,DU              STEP TO NEXT FIELD
    021724  021715 7100 00        7296          TRA     SLCT1             AND LOOP
    021725  000000 2270 12        7297 SLCT2    LDX7    0,2               GET MODE OF DESIRED FIELD
    021726  021731 7470 00        7298          STX7    FMODE             AND STORE
    021727  000000 7100 00        7299 SLCTX    TRA     **                AND RETURN
    021730  000000 000000         7300 DELTA    ZERO
END OF BINARY CARD 00000365
    021731  000000 000000         7301 FMODE    ZERO
    021732  021742 7400 00        7302 ETC      STX0    ETCX              SAVE RETURN
    021733  035234 2260 00        7303          LDX6    A$WORK            GET POINTER TO END OF WORKING STACK
    021734  000004 1660 03        7304          SBX6    1*WL,DU           GET POINTER TO LAST BLOCK IN WORKING STACK
    021735  017045 7270 00        7305          LXL7    PARAM             GET MODE OF RESULT OF SELECTION
    021736  000000 4470 16        7306          SXL7    0,6               AND STORE IN BLOCK
    021737  000001 2200 16        7307          LDX0    1,6               GET TEMP CURRENTLY ALLOCATED FOR BLOCK
```

```
          3                                                PASS 3

     021740  017037 7400 00        7308         STX0    SP              AND DEALLOCATE IT
     021741  020354 7000 00        7309         TSX0    GTMP            REALLOCATE STORAGE FOR NEW MODE
     021742  000000 7100 00        7310 ETCX    TRA     **              AND EXIT
     021743  021773 7400 00        7311 CASE    STX0    CASEX           SAVE RETURN
     021744  035234 2260 00        7312         LDX6    A$WORK          GET POINTER TO END OF WORKING STACK
     021745  000004 1660 03        7313         SBX6    1*WL,DU         GET POINTER TO LAST BLOCK IN WORKING STACK
     021746  020204 7000 00        7314         TSX0    MVA             MAKE VALUE AVAILABLE
     021747  021757 7100 00        7315         TRA     CASE1           TRANSFER IF VALUE IS IN REGISTER
     021750  021774 7550 00        7316         STA     CASET           SAVE POINTER TO VALUE IN TEMP
     021751  017612 7000 00        7317         TSX0    GQ              GET Q REGISTER
     021752  021774 2350 00        7318         LDA     CASET           GET POINTER TO VALUE IN A
     021753  236000 0750 07        7319         ADA     LDQ,DL          ADD LDQ COMMAND
     021754  017474 7000 00        7320         TSX0    GAD             AND ADD TO GENERATED CODE
     021755  000005 2220 03        7321         LDX2    5,DU            GET POINTER TO Q REGISTER CONTROL WORD
     021756  017721 7000 00        7322         TSX0    DR              RELEASE Q REGISTER
   END OF BINARY CARD 00000366
     021757  017045 2350 00        7323 CASE1   LDA     PARAM           GET MAXIMUM VALID INDEX IN AL
     021760  000022 7350 00        7324         ALS     18              MOVE IT TO AU
     021761  000001 0750 03        7325         ADA     1,DU            AND ADD ONE TO MAXIMUM
     021762  116007 0750 07        7326         ADA     CMPQ+DL,DL      GET CMPQ MAX+1,DL INSTRUCTION
     021763  017474 7000 00        7327         TSX0    GAD             AND ADD TO GENERATED CODE
     021764  021775 2350 00        7328         LDA     CASI1           GET TRC 3,IC INSTRUCTION
     021765  017474 7000 00        7329         TSX0    GAD             AND ADD TO GENERATED CODE
     021766  021776 2350 00        7330         LDA     CASI2           GET STC2 1,IC INSTRUCTION
     021767  017474 7000 00        7331         TSX0    GAD             AND ADD TO GENERATED CODE
     021770  021777 2350 00        7332         LDA     CASI3           GET TRA 0,QL INSTRUCTION
     021771  017474 7000 00        7333         TSX0    GAD             AND ADD TO GENERATED CODE
     021772  021007 7000 00        7334         TSX0    DELV            RELEASE ALL REGISTERS
     021773  000000 7100 00        7335 CASEX   TRA     **              AND RETURN
     021774  000000 000000         7336 CASET   ZERO
     021775  000003 6030 04        7337 CASI1   TRC     3,IC
     021776  000001 7500 04        7338 CASI2   STC2    1,IC
     021777  000000 7100 06        7339 CASI3   TRA     0,QL
     022000  022005 7400 00        7340 HIP     STX0    HIPX            SAVE RETURN
     022001  017045 7270 00        7341         LXL7    PARAM           GET MODE OF LABEL IN XR - 7
     022002  020377 7000 00        7342         TSX0    MBLK            AND MAKE A BLOCK FOR IT
     022003  100000 2250 03        7343         LDX5    BSFC,DU         GET STACKED VALUE BIT
     022004  000000 7450 16        7344         STX5    0,6             AND STORE IN BLOCK
   END OF BINARY CARD 00000367
     022005  000000 7100 00        7345 HIPX    TRA     **              AND RETURN
     022006  022016 7400 00        7346 LBL     STX0    LBLX            SAVE RETURN
     022007  020052 7000 00        7347         TSX0    PALL            PURGE ALL REGISTERS
     022010  017045 7210 00        7348         LXL1    PARAM           GET LABEL NUMBER IN XR - 1
     022011  035225 0610 00        7349         ADX1    T$LBL           ADD BASE OF LABEL TABLE
     022012  000000 7200 11        7350         LXL0    0,1             GET DEFINED VALUE FOR LABEL
     022013  777777 6010 00        7351         TNZ     $ERROR          ERROR - LABEL IS DEFINED TWICE
     022014  035226 7220 00        7352         LXL2    T$GEN           GET ADDRESS OF NEXT GENERATED CODE WORD
     022015  000000 4420 11        7353         SXL2    0,1             AND STORE AS DEFINITION OF LABEL
     022016  000000 7100 00        7354 LBLX    TRA     **              AND RETURN
     022017  022026 7400 00        7355 TRAD    STX0    TRADX           SAVE RETURN
```

```
              3                                                    PASS 3

       022020   017045 7210 00          7356          LXL1     PARAM             GET LABEL TO TRANSFER TO
       022021   035225 0610 00          7357          ADX1     T$LBL             ADD BASE OF LABEL TABLE
       022022   000000 2220 11          7358          LDX2     0,1               GET ADDRESS OF LAST TRANSFER TO THIS LABEL
       022023   035226 7230 00          7359          LXL3     T$GEN             GET ADDRESS OF CURRENT TRANSFER INSTRUCTION
       022024   000000 7430 11          7360          STX3     0,1               STORE ADDRESS OF CURRENT TRANSFER IN TABLE
       022025   000000 6350 12          7361          EAA      0,2               GET LINK TO LAST TRANSFER IN AU
       022026   000000 7100 00          7362 TRADX    TRA      **                AND RETURN
       022027   022034 7400 00          7363 JUMP     STX0     JUMPX             SAVE RETURN
       022030   020052 7000 00          7364          TSX0     PALL              PURGE ALL REGISTERS
       022031   022017 7000 00          7365          TSX0     TRAD              GET ADDRESS FOR TRANSFER INSTRUCTION IN AU
       022032   710004 0750 07          7366          ADA      TRA+IC,DL         ADD TRA 0,IC
   END OF BINARY CARD 00000368
       022033   017474 7000 00          7367          TSX0     GAD               AND ADD TO GENERATED CODE
       022034   000000 7100 00          7368 JUMPX    TRA      **                AND RETURN
       022035   022076 7400 00          7369 TF       STX0     TFX               SAVE RETURN
       022036   035234 2260 00          7370          LDX6     A$WORK            GET POINTER TO END OF WORKING STACK
       022037   000004 1660 03          7371          SBX6     WL,DU             GET POINTER TO LAST BLOCK IN WORKING STACK
       022040   020204 7000 00          7372          TSX0     MVA               MAKE BOOLEAN VALUE AVAILABLE
       022041   022045 7100 00          7373          TRA      TF1               TRANSFER IF VALUE IS IN A REGISTER
       022042   234000 0750 07          7374          ADA      SZN,DL            GET SZN A,B IN A REGISTER
       022043   017474 7000 00          7375          TSX0     GAD               AND ADD TO GENERATED CODE
       022044   022072 7100 00          7376          TRA      TF2               TRANSFER TO CONTINUE
       022045   035226 7240 00          7377 TF1      LXL4     T$GEN             GET ADDRESS OF NEXT WORD TO BE GENERATED
       022046   035226 0640 00          7378          ADX4     T$GEN             MAKE END OF CODE POINTER ABSOLUTE
       022047   777774 2350 14          7379          LDA      -4,4              GET FOURTH TO LAST INSTRUCTION IN OUTPUT
       022050   777000 2360 07          7380          LDQ      =0777000,DL       GET MASK TO COMPARE ALL BUT OP CODE
       022051   022077 2110 00          7381          CMK      TFI1              COMPARE WITH ARG 3,IC INSTRUCTION
       022052   022070 6010 00          7382          TNZ      TF3               TRANSFER IF NOT EQUAL TO NORMAL SEQUENCE
       022053   022100 2220 03          7383          LDX2     TFI2,DU           GET POINTER TO REST OF INSTRUCTION SEQUENCE
       022054   000000011007
       022055   007640 5602 01          7384          RPD      3,1,TNZ           SEARCH GENERATED CODE FOR A MISMATCH
       022056   777775 2350 14          7385          LDA      -3,4              GET GENERATED OUTPUT WORD
       022057   000000 1150 12          7386          CMPA     0,2               COMPARE WITH SEQUENCE TO CHANGE INDICATOR TO BOOL
       022060   022070 6010 00          7387          TNZ      TF3               TRANSFER IF NOT EXPECTED SEQUENCE
   END OF BINARY CARD 00000369
       022061   000004 3350 07          7388          LCA      4,DL              GET A MINUS 4 IN A REGISTER
       022062   035226 0550 00          7389          ASA      T$GEN             AND DELETE LAST FOUR WORDS OF GENERATED OUTPUT
       022063   022017 7000 00          7390          TSX0     TRAD              GET ADDRESS OF TRANSFER INSTRUCTION IN AU
       022064   000000 6200 01          7391          EAX0     0,AU              GET ADDRESS IN XR - 0
       022065   777774 7400 14          7392          STX0     -4,4              AND STORE IN CONDITIONAL TRANSFER INSTRUCTION
       022066   035226 0540 00          7393          AOS      T$GEN             ADD CONDITIONAL TRANSFER INSTRUCTION TO OUTPUT
       022067   022075 7100 00          7394          TRA      TF4               GO TO DELETE BOOLEAN VALUE
       022070   736000 2350 07          7395 TF3      LDA      QLS,DL            GET QLS 0 INSTRUCTION
       022071   017474 7000 00          7396          TSX0     GAD               AND ADD TO GENERATED CODE
       022072   022017 7000 00          7397 TF2      TSX0     TRAD              GET ADDRESS FOR TRANSFER INSTRUCTION IN AU
       022073   600004 0750 07          7398          ADA      TZE+IC,DL         GET TZE ADDR,IC INSTRUCTION IN A
       022074   017474 7000 00          7399          TSX0     GAD               AND ADD TO GENERATED CODE
       022075   021007 7000 00          7400 TF4      TSX0     DELV              DELETE BOOLEAN VALUE FROM STACK
       022076   000000 7100 00          7401 TFX      TRA      **                AND RETURN
       022077   000003 0000 04          7402 TFI1     ARG      3,IC
```

```
    022100  000001 2360 07       7403 TFI2    LDQ     1,DL
    022101  000002 7100 04       7404         TRA     2,IC
    022102  000000 2360 07       7405         LDQ     0,DL
    022103  022107 7400 00       7406 MIL     STX0    MTLX          SAVE RETURN
    022104  010630 7000 00       7407         TSX0    A$XFER        MAKE MODE POINTER ABSOLUTE
    022105  000000 7210 17       7408         LXL1    0,7           GET TYPE POINTER FOR MODE IN XR - 1
    022106  022110 7000 00       7409         TSX0    TL            GET ADDRESS FOR TYPE IN AU
 END OF BINARY CARD 00000370
    022107  000000 7100 00       7410 MTLX    TRA     **            AND RETURN
    022110  022126 7400 00       7411 TL      STX0    TLX           SAVE RETURN
    022111  022124 7420 00       7412         STX2    TL3           SAVE XR - 2
    022112  022125 7430 00       7413         STX3    TL4           SAVE XR - 3
    022113  035227 0610 00       7414         ADX1    T$TYPE        MAKE TYPE POINTER ABSOLUTE
    022114  777777 7220 11       7415         LXL2    -1,1          GET POINTER TO LAST USE OF THIS TYPE IN XR - 2
    022115  035226 7230 00       7416         LXL3    T$GEN         GET ADDRESS OF NEXT INSTRUCTION IN XR - 3
    022116  022122 7430 00       7417         STX3    TL1           AND STORE FOR RELATIVE CALCULATION
    022117  777777 4430 11       7418         SXL3    -1,1          AND STORE AS POINTER TO THIS USE OF THIS TYPE
    022120  000000 1020 03       7419         CMPX2   0,DU          IS PREVIOUS LINK POINTER ZERO
    022121  022123 6000 00       7420         TZE     TL2           YES - TRANSFER TO LEAVE ALONE
    022122  000000 1620 03       7421 TL1     SBX2    **,DU         MAKE PREVIOUS LINK POINTER RELATIVE
    022123  000000 6350 12       7422 TL2     EAA     0,2           GET POINTER TO LAST USE IN AU
    022124  000000 2220 03       7423 TL3     LDX2    **,DU         RESTORE XR - 2
    022125  000000 2230 03       7424 TL4     LDX3    **,DU         RESTORE XR - 3
    022126  000000 7100 00       7425 TLX     TRA     **            AND RETURN
    022127  022233 7400 00       7426 DEREF   STX0    DERX          SAVE RETURN
    022130  035234 2260 00       7427         LDX6    A$WORK        GET POINTER TO END OF WORKING STACK
    022131  000004 1660 03       7428         SBX6    WL,DU         GET POINTER TO LAST BLOCK IN WORKING STACK
    022132  000000 7270 16       7429         LXL7    0,6           GET MODE OF VALUE IN XR - 7
    022133  010630 7000 00       7430         TSX0    A$XFER        MAKE IT ABSOLUTE
    022134  000000 2200 17       7431         LDX0    0,7           GET TYPE OF MODE IN XR - 0
 END OF BINARY CARD 00000371
    022135  016762 1000 03       7432         CMPX0   M$REF,DU      IS IT A REFERENCE MODE
    022136  777777 6010 00       7433         TNZ     $ERROR        NO - COMPILER ERROR
    022137  000001 2270 17       7434         LDX7    1,7           GET DEREFERENCED MODE IN XR - 7
    022140  022240 7470 00       7435         STX7    DERM          SAVE DEREFERENCED MODE
    022141  010562 7000 00       7436         TSX0    A$RCHK        SEE IF IT IS AN ARRAY MODE
    022142  022167 7100 00       7437         TRA     DER0          NO - TRANSFER
    022143  020204 7000 00       7438         TSX0    MVA           MAKE ARRAY VALUE AVAILABLE
    022144  777777 7100 00       7439         TRA     $ERROR        COMPILER ERROR - NEVER IN ACCUMULATOR
    022145  022234 7510 71       7440         STCA    DERI1,71      STORE ADDRESS IN INSTRUCTION SEQUENCE
    022146  022236 7510 71       7441         STCA    DERI2,71      STORE ADDRESS IN INSTRUCTION SEQUENCE
    022147  022237 2350 00       7442         LDA     DERIT         GET TALLY WORD FOR INSTRUCTION SEQUENCE
    022150  017507 7000 00       7443         TSX0    GADL          AND ADD TO GENERATED OUTPUT
    022151  000003 2360 16       7444         LDQ     3,6           GET POSSIBLE POINTER IN Q
    022152  017673 7000 00       7445         TSX0    CL3           AND DEALLOCATE IT
    022153  000010 2350 07       7446         LDA     =O10,DL       GET (0,0) ADDRESS IN A REGISTER
    022154  000001 2360 16       7447         LDQ     1,6           GET STACK ADDRESS IN QU
    022155  777777 3760 03       7448         ANQ     -1,DU         ZERO OUT QL
    022156  000002 7560 16       7449         STQ     2,6           AND STORE IN BLOCK
    022157  000017 0760 07       7450         ADQ     D,DL          ADD D REGISTER MODIFICATION
```

```
         3                                             PASS 3

      022160  022240 2270 00        7451          LDX7     DERM             GET MODE OF DEREFERENCED VALUE IN XR - 7
      022161  000000 4470 16        7452          SXL7     0,6              AND STORE IT IN BLOCK
      022162  017531 7000 00        7453          TSX0     MOVE             MOVE VALUE TO STACK
   END OF BINARY CARD 00000372
      022163  030000 2250 03        7454          LDX5     B$FE+B$FF,DU     CLAIM VALUE IS IN STACK
      022164  000000 7450 16        7455          STX5     0,6              AND STORE FLAGS IN BLOCK
      022165  020451 7000 00        7456          TSX0     FS               FORCE ARRAYS TO STACK
      022166  022233 7100 00        7457          TRA      DERX             AND RETURN
      022167  000000 2250 16        7458 DER0     LDX5     0,6              GET FLAGS IN XR - 5
      022170  242000 3050 03        7459          CANX5    B$FB+B$FD+B$FH,DU CHECK IF VALUE IS IMMEDIATELY ACCESSABLE
      022171  022173 6010 00        7460          TNZ      DER1             TRANSFER IF IMMEDIATELY ACCESSABLE
      022172  020066 7000 00        7461          TSX0     MNA              MAKE NAME AVAILABLE
      022173  200000 3050 03        7462 DER1     CANX5    B$FB,DU          IS DEREFERENCED VALUE IN A REGISTER
      022174  022204 6000 00        7463          TZE      DER2             TRANSFER IF NOT
      022175  000003 2360 16        7464          LDQ      3,6              GET POSSIBLE NAME POINTER IN Q
      022176  017673 7000 00        7465          TSX0     CL3              AND DEALLOCATE IT
      022177  000003 4500 16        7466          STZ      3,6              RESET POSSIBLE POINTER
      022200  000001 7200 16        7467          LXL0     1,6              GET STORE COMMAND FOR DEREFERENCED VALUE
      022201  000003 4400 16        7468          SXL0     3,6              AND MAKE IT STORE COMMAND FOR VALUE
      022202  577777 3650 03        7469          ANX5     -1-B$FB,DU       CLEAR DEREFERENCED VALUE IN REGISTER BIT
      022203  004000 2650 03        7470          ORX5     B$FG,DU          SET VALUE IN REGISTER BIT
      022204  040000 3050 03        7471 DER2     CANX5    B$FD,DU          IS NAME VALUE OFFSET,LL
      022205  022210 6000 00        7472          TZE      DER3             TRANSFER IF NOT
      022206  020000 2650 03        7473          ORX5     B$FE,DU          SET OFFSET,LL IS REFERENCE TO VALUE BIT
      022207  022211 7100 00        7474          TRA      DER4             AND CONTINUE
      022210  747777 3650 03        7475 DER3     ANX5     -1-B$FE-B$FF,DU  RESET ALL BITS ABOUT 2,6
   END OF BINARY CARD 00000373
      022211  002000 3050 03        7476 DER4     CANX5    B$FH,DU          IS (A,B) VALUE
      022212  022216 6000 00        7477          TZE      DER5             TRANSFER IF (A,B) IS NOT VALUE
      022213  775777 3650 03        7478          ANX5     -1-B$FH,DU       RESET (A,B) IS VALUE BIT
      022214  001000 2650 03        7479          ORX5     B$FI,DU          SET (A,B) IS REFERENCE TO VALUE BIT
      022215  022220 7100 00        7480          TRA      DER6             CONTINUE
      022216  000003 2360 16        7481 DER5     LDQ      3,6              GET (A,B) WORD
      022217  017673 7000 00        7482          TSX0     CL3              DEALLOCATE IT
      022220  237777 3650 03        7483 DER6     ANX5     -1-B$FA-B$FC-B$FD,DU RESET DENOT, STACK, 2,6 IS VALUE BITS
      022221  022240 2270 00        7484          LDX7     DERM             GET DEREFERENCED MODE IN XR - 7
      022222  000000 4470 16        7485          SXL7     0,6              AND STORE AS MODE OF NEW VALUE
      022223  000000 7450 16        7486          STX5     0,6              RESTORE FLAGS IN BLOCK
      022224  000001 2200 16        7487          LDX0     1,6              GET PREVIOUS ALLOCATED TEMP
      022225  017037 7400 00        7488          STX0     SP               RESTORE IT
      022226  020354 7000 00        7489          TSX0     GTMP             ALLOCATE TEMP FOR NEW VALUE
      022227  000000 7270 16        7490          LXL7     0,6              GET MODE OF VALUE IN XR - 7
      022230  010562 7000 00        7491          TSX0     A$RCHK           SEE IF IT IS A ROW TYPE MODE
      022231  022233 7100 00        7492          TRA      DERX             TRANSFER IF NOT
      022232  020451 7000 00        7493          TSX0     FS               FORCE THE ARRAY VALUES TO THE STACK
      022233  000000 7100 00        7494 DERX     TRA      **               AND RETURN
      022234  000000 2200 00        7495 DERI1    LDX0     0
      022235  000002 6010 04        7496          TNZ      2,IC
      022236  000000 6200 00        7497 DERI2    EAX0     0
   END OF BINARY CARD 00000374
```

```
     022237   022234 0004 00         7498 DERIT   TALLY     DERI1,*-DERI1+1
     022240   000000 000000          7499 DERM    ZERO
     022241   022260 7400 00         7500 IDENT   STX0      IDENX               SAVE RETURN
                                     7501                   INE       WL,4
                                     7502                   IERROR                    THIS ROUTINE NEEDS WL TO EQUAL 4
     022242   017045 7210 00         7503         LXL1      PARAM               GET POINTER TO DEF TABLE ENTRY FOR IDENTIFIER
     022243   035220 0610 00         7504         ADX1      T$DEF               MAKE POINTER ABSOLUTE
     022244   000002 2270 11         7505         LDX7      2,1                 GET MODE OF IDENTIFIER IN XR - 7
     022245   020377 7000 00         7506         TSX0      MBLK                CREATE BLOCK FOR IDENTIFIER
     022246   020717 7000 00         7507         TSX0      IDFY                IDENTIFY IDENTIFIER AND GET LL DIFFERENCE
     022247   000003 2350 11         7508         LDA       3,1                 GET OFFSET OF SYMBOL IN AU
     022250   000000 6350 01         7509         EAA       0,AU                ZERO OUT AL
     022251   020746 0750 00         7510         ADA       IDFYC               PUT LL DIFFERENCE IN AL
     022252   020000 2250 03         7511         LDX5      B$FE,DU             GET OFFSET, LEVEL IS REFERENCE TO VALUE BIT
     022253   020746 2340 00         7512         SZN       IDFYC               SEE IF IDENTIFIER IS IN CURRENT LEVEL
     022254   022256 6010 00         7513         TNZ       IDEN6               TRANSFER IF NOT IN CURRENT LEVEL
     022255   010000 2650 03         7514         ORX5      B$FF,DU             SET OFFSET, LEVEL IS LOCAL BIT
     022256   000002 7550 16         7515 IDEN6   STA       2,6                 STORE ADDRESS IN BLOCK
     022257   000000 7450 16         7516         STX5      0,6                 STORE FLAGS IN IDENTIFIER BLOCK
     022260   000000 7100 00         7517 IDENX   TRA       **                  AND RETURN
     022261   022321 7400 00         7518 GOTO    STX0      GOTOX               SAVE RETURN
     022262   020717 7000 00         7519         TSX0      IDFY                IDENTIFY LABEL
     022263   020746 2360 00         7520         LDQ       IDFYC               GET LL DIFFERENCE IF ANY IN Q
     022264   020747 0760 00         7521         ADQ       IDFYF               ADD ANY CHANGES DETECTED IN ENVIRONMENT
 END OF BINARY CARD 00000375
     022265   022311 6000 00         7522         TZE       GOTO3               TRANSFER IF NO CHANGE IN LL
     022266   035215 1600 00         7523         SBX0      T$STACK             MAKE ENVIRONMENT POINTER RELATIVE
     022267   022277 7400 00         7524         STX0      GOTO2               AND SAVE
     022270   000000 2360 07         7525         LDQ       0,DL                GET A ZERO IN Q
     022271   020746 1560 00         7526         SSQ       IDFYC               AND NEGATE LL DIFFERENCE IN MEMORY
     022272   022277 6000 00         7527 GOTO1   TZE       GOTO2               TRANSFER IF NO MORE LL CHANGE
     022273   022322 2350 00         7528         LDA       GOTI1               GET LDX D,2,D INSTRUCTION
     022274   017474 7000 00         7529         TSX0      GAD                 AND ADD TO GENERATED OUTPUT
     022275   020746 0540 00         7530         AOS       IDFYC               DECREMENT AMOUNT OF LL CHANGE LEFT
     022276   022272 7100 00         7531         TRA       GOTO1               AND LOOP
     022277   000000 2200 03         7532 GOTO2   LDX0      **,DU               GET ENVIRONMENT POINTER IN XR - 0
     022300   035215 0600 00         7533         ADX0      T$STACK             MAKE POINTER ABSOLUTE
     022301   000000 2200 10         7534         LDX0      0,0                 GET PREVIOUS ENVIRONMENT POINTER IN XR - 0
     022302   035215 0600 00         7535         ADX0      T$STACK             AND MAKE IT ABSOLUTE
     022303   000002 2260 10         7536         LDX6      2,0                 GET POINTER TO MS BLOCK
     022304   035214 0660 00         7537         ADX6      T$WORK              MAKE POINTER ABSOLUTE
     022305   000001 2350 16         7538         LDA       1,6                 GET ADDRESS OF MS IN AU
     022306   000001 6350 01         7539         EAA       1,AU                GET ADDRESS OF SAVED S IN AU
     022307   022323 0750 00         7540         ADA       GOTI2               ADD LDX S,0,D INSTRUCTION
     022310   017474 7000 00         7541         TSX0      GAD                 AND ADD TO GENERATED CODE
     022311   017045 7210 00         7542 GOTO3   LXL1      PARAM               GET DEF POINTER IN XR - 1
     022312   035220 0610 00         7543         ADX1      T$DEF               MAKE POINTER ABSOLUTE
 END OF BINARY CARD 00000376
     022313   000003 2220 11         7544         LDX2      3,1                 GET LABEL IN XR - 2
     022314   017045 4420 00         7545         SXL2      PARAM               AND STORE IN PARAM
```

```
             3                                        PASS 3

      022315  022017 7000 00         7546        TSX0    TRAD          GET LINKED TRANSFER ADDRESS IN AU
      022316  710004 0750 07         7547        ADA     TRA+IC,DL     ADD TRA 0,IC INSTRUCTION
      022317  017474 7000 00         7548        TSX0    GAD           AND ADD TO GENERATED OUTPUT
      022320  020052 7000 00         7549        TSX0    PALL          PURGE ALL REGISTERS
      022321  000000 7100 00         7550 GOTOX  TRA     **            AND RETURN
      022322  000002 2270 17         7551 GOTI1  LDX     D,2,D
      022323  000000 2260 17         7552 GOTI2  LDX     S,0,D
      022324  022355 7400 00         7553 DENOT  STX0    DENX          SAVE RETURN
      022325  017045 7210 00         7554        LXL1    PARAM         GET POINTER TO DEF TABLE ENTRY FOR DENOTATION
      022326  035220 0610 00         7555        ADX1    T$DEF         MAKE POINTER ABSOLUTE
      022327  000002 2270 11         7556        LDX7    2,1           GET MODE OF DENOTATION
      022330  020377 7000 00         7557        TSX0    MBLK          AND MAKE A BLOCK FOR IT
      022331  017045 7210 00         7558        LXL1    PARAM         GET POINTER TO DEF TABLE ENTRY FOR DENOTATION
      022332  035220 0610 00         7559        ADX1    T$DEF         MAKE POINTER ABSOLUTE
      022333  000003 2350 11         7560        LDA     3,1           GET VALUE OF DENOTATION IN A REGISTER
      022334  022357 7550 00         7561        STA     DEN2          AND STORE IN INSTRUCTION SEQUENCE
      022335  000000 7270 16         7562        LXL7    0,6           GET MODE OF DENOTATION IN XR - 7
      022336  017355 2220 03         7563        LDX2    OPET,DU       GET POINTER TO MODE COMMAND TABLE
      022337  024300 5202 02         7564        RPT     OPETE/2-OPET/2,2,TZE AND SEARCH FOR MODE IN TABLE
      022340  000000 1070 12         7565        CMPX7   0,2           LOOK IN XR - 7
  END OF BINARY CARD 00000377
      022341  777777 6010 00         7566        TNZ     $ERROR        ERROR MODE IS NOT IN TABLE
      022342  777776 2350 12         7567        LDA     -2,2          GET STORE COMMAND FOR MODE
      022343  777777 3750 07         7568        ANA     -1,DL         MAKE IT CLEAN
      022344  000003 7550 16         7569        STA     3,6           AND STORE IN BLOCK
      022345  777777 2350 12         7570        LDA     -1,2          GET LOAD COMMAND FOR MODE
      022346  022360 5510 10         7571        STBA    DEN3,10       AND STORE IN COMMAND SEQUENCE
      022347  777777 2220 12         7572        LDX2    -1,2          GET REGISTER ALLOCATION ROUTINE NUMBER
      022350  017637 7000 00         7573        TSX0    GET           AND ALLOCATE REGISTER FOR VALUE
      022351  004000 2250 03         7574        LDX5    B$FG,DU       GET VALUE IS IN REGISTER FLAG
      022352  000000 7450 16         7575        STX5    0,6           AND STORE IN BLOCK
      022353  022361 2350 00         7576        LDA     DENT          GET TALLY WORD FOR INSTRUCTION SEQUENCE
      022354  017507 7000 00         7577        TSX0    GADL          AND ADD INSTRUCTION SEQUENCE TO GENERATED OUTPUT
      022355  000000 7100 00         7578 DENX   TRA     **            AND RETURN
      022356  000002 7100 04         7579 DEN1   TRA     2,IC
      022357  000000 000000          7580 DEN2   ZERO
      022360  777777 0000 04         7581 DEN3   ARG     -1,IC
      022361  022356 0004 00         7582 DENT   TALLY   DEN1,*-DEN1+1
      022362  022364 2350 00         7583 TRUE   LDA     TRUET         GET LDQ TRUE INSTRUCTION
      022363  022366 7100 00         7584        TRA     BOOL          AND GO PROCESS IT
      022364  000001 2360 07         7585 TRUET  LDQ     1,DL
      022365  236007 2350 07         7586 FALSE  LDA     LDQ+DL,DL     GET LDQ FALSE COMMAND
      022366  022402 7550 00         7587 BOOL   STA     BOOLT         STORE BOOLEAN LOAD COMMAND
  END OF BINARY CARD 00000378
      022367  022401 7400 00         7588        STX0    BOOLX         SAVE RETURN
      022370  000003 6270 00         7589        EAX7    M$BOOL        GET MODE OF VALUE IN XR - 7
      022371  020377 7000 00         7590        TSX0    MBLK          MAKE A BLOCK FOR VALUE
      022372  017612 7000 00         7591        TSX0    GQ            ALLOCATE Q REGISTER FOR BOOLEAN VALUE
      022373  004000 2250 03         7592        LDX5    B$FG,DU       GET VALUE IN REGISTER FLAG
      022374  000000 7450 16         7593        STX5    0,6           AND STORE IT IN BLOCK
```

```
         3                                            PASS 3

    022375  756000 2350 07         7594         LDA     STQ,DL          GET BOOLEAN STORE COMMAND
    022376  000003 7550 16         7595         STA     3,6             AND STORE IN BLOCK
    022377  022402 2350 00         7596         LDA     BOOLT           GET BOOLEAN LOAD VALUE COMMAND
    022400  017474 7000 00         7597         TSX0    GAD             AND ADD TO GENERATED OUTPUT CODE
    022401  000000 7100 00         7598 BOOLX   TRA     **              AND EXIT
    022402  000000 000000          7599 BOOLT   ZERO
    022403  026130 2210 03         7600 LGEN    LDX1    R$LGEN,DU       GET ADDRESS OF RUN-TIME LOCAL GENERATOR ROUTINE
    022404  022452 2350 00         7601         LDA     RLGEN           GET TSX0 R$RLGEN INSTRUCTION IN A REGISTER
    022405  022410 7100 00         7602         TRA     GEN             AND GO TO GENERATOR ROUTINE
    022406  026157 2210 03         7603 HGEN    LDX1    R$HGEN,DU       GET ADDRESS OF RUN-TIME HEAP GENERATOR ROUTINE
    022407  022453 2350 00         7604         LDA     RHGEN           GET TSX0 R$RHGEN INSTRUCTION IN A REGISTER
    022410  022450 7410 00         7605 GEN     STX1    GEN2            STORE IN INSTRUCTION SEQUENCE
    022411  023073 7550 00         7606         STA     BGEN            STORE ROW GENERATOR INSTRUCTION FOR BOUND ROUTINE
    022412  022445 7400 00         7607         STX0    GENX            SAVE RETURN
    022413  000033 6270 00         7608         EAX7    M$MS            GET MODE OF MARK STACK IN XR - 7
    022414  020377 7000 00         7609         TSX0    MBLK            AND MAKE A BLOCK FOR MARK STACK
  END OF BINARY CARD 00000379
    022415  020526 7000 00         7610         TSX0    STYPE           MARK THE STACK IN MARK STACK WORDS
    022416  035234 2260 00         7611         LDX6    A$WORK          GET POINTER TO END OF WORKING STACK
    022417  000004 1660 03         7612         SBX6    WL,DU           GET POINTER TO LAST BLOCK IN WORKING STACK
    022420  000001 2200 16         7613         LDX0    1,6             GET ADDRESS OF MARK STACK
    022421  022446 7400 00         7614         STX0    GEN0            AND STORE IN INSTRUCTION SEQUENCE
    022422  023663 7000 00         7615         TSX0    POP             POP STACK BACK THROUGH MARK STACK
    022423  017045 7270 00         7616         LXL7    PARAM           GET MODE OF GENERATOR
    022424  020377 7000 00         7617         TSX0    MBLK            MAKE A BLOCK FOR VALUE OF GENERATOR
    022425  017610 7000 00         7618         TSX0    GA              GET A REGISTER
    022426  004000 2250 03         7619         LDX5    B$FG,DU         GET A VALUE IN ACCUMULATOR FLAG
    022427  000000 7450 16         7620         STX5    0,6             AND STORE IN BLOCK
    022430  755000 2350 07         7621         LDA     STA,DL          GET A STORE VALUE COMMAND
    022431  000003 7550 16         7622         STA     3,6             AND STORE IN BLOCK
    022432  000001 2270 17         7623         LDX7    1,7             GET DEREFERENCED MODE
    022433  023074 7470 00         7624         STX7    BDCLR           STORE DEREFERENCED MODE FOR BOUND ROUTINE
    022434  010630 7000 00         7625         TSX0    A$XFER          MAKE MODE POINTER UNIQUE AND ABSOLUTE
    022435  777777 7200 17         7626         LXL0    -1,7            GET LENGTH OF VALUE REFERED TO BY GENERATOR
    022436  022447 7400 00         7627         STX0    GEN1            AND STORE IN INSTRUCTION SEQUENCE
    022437  035216 1670 00         7628         SBX7    T$MODE          MAKE MODE POINTER RELATIVE
    022440  022451 2350 00         7629         LDA     GENT            GET TALLY WORD FOR INSTRUCTION SEQUENCE
    022441  017507 7000 00         7630         TSX0    GADL            AND ADD SEQUENCE TO GENERATED OUTPUT
    022442  022103 7000 00         7631         TSX0    MTL             GET TYPE ADDRESS IN AU
  END OF BINARY CARD 00000380
    022443  075007 0750 07         7632         ADA     ADA+DL,DL       ADD ADA ,DL COMMAND
    022444  017474 7000 00         7633         TSX0    GAD             AND ADD TO GENERATED OUTPUT
    022445  000000 7100 00         7634 GENX    TRA     **              AND RETURN
    022446  000000 6210 17         7635 GEN0    EAX1    0,D
    022447  000000 6220 00         7636 GEN1    EAX2    0
    022450  000000 7000 00         7637 GEN2    TSX0    0
    022451  022446 0004 00         7638 GENT    TALLY   GEN0,*-GEN0+1
    022452  026015 7000 00         7639 RLGEN   TSX0    R$RLGEN
    022453  026017 7000 00         7640 RHGEN   TSX0    R$RHGEN
    022454  022506 7400 00         7641 BOUND   STX0    BNDX            SAVE RETURN
```

```
               3                                        PASS 3

    022455  020052 7000 00       7642          TSX0     PALL            PURGE ALL REGISTERS
    022456  023074 2270 00       7643          LDX7     BDCLR           GET MODE OF VALUE IN XR - 7
    022457  010630 7000 00       7644          TSX0     A$XFER          MAKE MODE POINTER ABSOLUE
    022460  000000 2200 17       7645          LDX0     0,7             GET TYPE OF MODE IN XR - 0
    022461  016757 1000 03       7646          CMPX0    M$STRCT,DU      IS IT A STRUCTURED MODE
    022462  022476 6000 00       7647          TZE      BND1            TRANSFER IF STRUCTURED MODE
    022463  000037 6270 00       7648          EAX7     M$PTR           GET MODE OF POINTER IN XR - 7
    022464  020377 7000 00       7649          TSX0     MBLK            MAKE A BLOCK FOR A POINTER
    022465  100000 2250 03       7650          LDX5     B$FC,DU         GET A VALUE IS STACKED FLAG
    022466  000000 7450 16       7651          STX5     0,6             AND STORE IN BLOCK
    022467  000001 2350 16       7652          LDA      1,6             GET ADDRESS OF POINTER IN AU
    022470  022510 7510 70       7653          STCA     BNDI2,70        AND STORE IN INSTRUCTION SEQUENCE
  END OF BINARY CARD 00000381
    022471  000004 1660 03       7654          SBX6     WL,DU           GET POINTER TO GENERATOR BLOCK
    022472  000001 2350 16       7655          LDA      1,6             GET ADDRESS OF DESCRIPTOR IN AU
    022473  022507 7510 70       7656          STCA     BNDI1,70        AND STORE IN INSTRUCTION SEQUENCE
    022474  022511 2350 00       7657          LDA      BNDT            GET TALLY WORD FOR INSTRUCTION SEQUENCE
    022475  017507 7000 00       7658          TSX0     GADL            AND ADD INSTRUCTION SEQUENCE TO GENERATED OUTPUT
    022476  023075 4500 00       7659 BND1     STZ      BPOS            INITIALIZE CURRENT POSITION TO ZERO
    022477  022512 7000 00       7660          TSX0     BD              SET UP ALL BOUNDS IN GENERATED VALUE
    022500  023074 2270 00       7661          LDX7     BDCLR           GET MODE OF GENERATED VALUE IN XR - 7
    022501  010630 7000 00       7662          TSX0     A$XFER          MAKE MODE POINTER ABSOLUTE
    022502  000000 2200 17       7663          LDX0     0,7             GET TYPE OF MODE IN XR - 0
    022503  016757 1000 03       7664          CMPX0    M$STRCT,DU      IS IT A STRUCTURED VALUE
    022504  022506 6000 00       7665          TZE      BNDX            TRANSFER IF STRUCTURED VALUE - ALL DONE
    022505  021007 7000 00       7666          TSX0     DELV            DELETE POINTER BLOCK
    022506  000000 7100 00       7667 BNDX     TRA      **              AND RETURN
    022507  000000 6350 17       7668 BNDI1    EAA      0,D
    022510  000000 7550 17       7669 BNDI2    STA      0,D
    022511  022507 0003 00       7670 BNDT     TALLY    BNDI1,*-BNDI1+1
    022512  023076 7400 00       7671 BD       STX0     BDX             SAVE RETURN
    022513  017045 7200 00       7672          LXL0     PARAM           GET POINTER TO BOUND TABLE ENTRY FOR DECLARER
    022514  023074 4400 00       7673          SXL0     BDCLR           AND STORE IN CURRENT DECLARER
    022515  023075 0540 00       7674 BD1      AOS      BPOS            STEP TO NEXT FIELD
    022516  023074 2270 00       7675          LDX7     BDCLR           GET MODE OF DECLARER IN XR - 7
  END OF BINARY CARD 00000382
    022517  010630 7000 00       7676          TSX0     A$XFER          MAKE MODE POINTER ABSOLUTE
    022520  023075 7200 00       7677          LXL0     BPOS            GET POINTER TO CURRENT FIELD NUMBER
    022521  777777 1000 17       7678          CMPX0    -1,7            IS CURRENT POSITION AT END OF CURRENT MODE
    022522  023076 6000 51       7679          TZE      BDX,I           RETURN IF SO
    022523  023077 7400 00       7680          STX0     BPOS1           SAVE CURRENT POSITION NUMBER
    022524  000000 2200 17       7681          LDX0     0,7             GET TYPE OF MODE IN XR - 0
    022525  016757 1000 03       7682          CMPX0    M$STRCT,DU      IS IT A STRUCTURED MODE
    022526  022602 6010 00       7683          TNZ      BD5             TRANSFER IF NOT STRUCTURED MODE
    022527  023077 0670 00       7684          ADX7     BPOS1           GET POINTER TO CURRENT FIELD
    022530  000000 2270 17       7685          LDX7     0,7             GET MODE OF FIELD IN XR - 7
    022531  010630 7000 00       7686          TSX0     A$XFER          MAKE MODE POINTER ABSOLUTE
    022532  000000 2200 17       7687          LDX0     0,7             GET TYPE OF FIELD MODE IN XR - 0
    022533  016757 1000 03       7688          CMPX0    M$STRCT,DU      IS IT A STRUCTURED MODE
    022534  022541 6000 00       7689          TZE      BD2             TRANSFER IF YES
```

```
          3                                          PASS 3

     022535  016770 1000 03       7690        CMPX0     M$ROW,DU         IS IT A ROWED MODE
     022536  022541 6000 00       7691        TZE       BD2              YES - TRANSFER
     022537  016776 1000 03       7692        CMPX0     M$ROWE,DU        IS IT A END ROW MODE
     022540  022577 6010 00       7693        TNZ       BD4              NO - TRANSFER
     022541  035216 1670 00       7694 BD2    SBX7      T$MODE           MAKE MODE POINTER RELATIVE
     022542  023076 2350 00       7695        LDA       BDX              GET RETURN
     022543  035235 7550 56       7696        STA       A$STACK,ID       AND STORE IN CONTROL STACK
     022544  005742 7170 00       7697        XED       T$SOVF           CHECK FOR STACK OVERFLOW
   END OF BINARY CARD 00000383
     022545  023074 2350 00       7698        LDA       BDCLR            GET CURRENT DECLARER
     022546  035235 7550 56       7699        STA       A$STACK,ID       AND STORE IN CONTROL STACK
     022547  005742 7170 00       7700        XED       T$SOVF           CHECK FOR STACK OVERFLOW
     022550  023075 2350 00       7701        LDA       BPOS             GET CURRENT POSITION
     022551  035235 7550 56       7702        STA       A$STACK,ID       AND STORE IN CONTROL STACK
     022552  005742 7170 00       7703        XED       T$SOVF           CHECK FOR STACK OVERFLOW
     022553  023074 7470 00       7704        STX7      BDCLR            SET MODE OF FIELD IN CURRENT DECLARER
     022554  023074 7210 00       7705        LXL1      BDCLR            GET BOUND TABLE POINTER OF CURRENT DECLARER
     022555  035217 0610 00       7706        ADX1      T$BOUND          MAKE POINTER ABSOLUTE
     022556  000000 2200 11       7707        LDX0      0,1              GET TYPE OF BOUNT TABLE ENTRY IN XR - 0
     022557  016757 1000 03       7708        CMPX0     B$STRCT,DU       IS IT A STRUCTURE ENTRY
     022560  777777 6010 00       7709        TNZ       $ERROR           COMPILER ERROR
     022561  023077 0610 00       7710        ADX1      BPOS1            GET POINTER TO FIELD BOUND TABLE ENTRY
     022562  000000 2210 11       7711        LDX1      0,1              GET FIELD BOUND POINTER IN XR - 1
     022563  023074 4410 00       7712 BD3    SXL1      BDCLR            AND STORE IN CURRENT DECLARER
     022564  000000 2200 03       7713        LDX0      0,DU             GET A ZERO
     022565  023075 4400 00       7714        SXL0      BPOS             AND RESET FIELD COUNT
     022566  022512 7000 00       7715        TSX0      BD               DO ANY BOUNDS IN FIELD
     022567  023074 2270 00       7716        LDX7      BDCLR            GET MODE OF FIELD
     022570  035235 2350 54       7717        LDA       A$STACK,DI       POP A WORD FROM THE CONTROL STACK
     022571  023075 7550 00       7718        STA       BPOS             AND RESTORE CURRENT POSITION
     022572  035235 2350 54       7719        LDA       A$STACK,DI       POP A WORD FROM THE CONTROL STACK
   END OF BINARY CARD 00000384
     022573  023074 7550 00       7720        STA       BDCLR            AND RESTORE CURRENT DECLARER
     022574  035235 2350 54       7721        LDA       A$STACK,DI       POP A WORD FROM THE CONTROL STACK
     022575  023076 7550 00       7722        STA       BDX              AND RESTORE RETURN
     022576  010630 7000 00       7723        TSX0      A$XFER           MAKE FIELD MODE POINTER ABSOLUTE
     022577  777777 7200 17       7724 BD4    LXL0      -1,7             GET GENGTH OF FIELD VALUE
     022600  023075 0400 00       7725        ASX0      BPOS             AND INCREMENT CURRENT POSITION
     022601  022515 7100 00       7726        TRA       BD1              AND LOOP
     022602  016770 1000 03       7727 BD5    CMPX0     M$ROW,DU         IS IT A ROW MODE
     022603  022606 6000 00       7728        TZE       BD6              TRANSFER IF YES
     022604  016776 1000 03       7729        CMPX0     M$ROWE,DU        IS IT A END ROW MODE
     022605  777777 6010 00       7730        TNZ       $ERROR           NO - ERROR
     022606  000035 6270 00       7731 BD6    EAX7      M$MSCW           GET MODE OF MARK STACK IN XR - 7
     022607  020377 7000 00       7732        TSX0      MBLK             AND MAKE A BLOCK FOR A MARK STACK WORD
     022610  100000 2250 03       7733        LDX5      B$FC,DU          GET A VALUE IS STACKED BIT
     022611  000000 7450 16       7734        STX5      0,6              AND CLAIM MARK STACK VALUE IS STACKED
     022612  000001 2210 16       7735        LDX1      1,6              GET ADDRESS OF MARK STACK IN AU
     022613  023102 7410 00       7736        STX1      BDI1             AND STORE IN INSTRUCTION SEQUENCE
     022614  023116 7410 00       7737        STX1      BDI20            AND STORE IN INSTRUCTION SEQUENCE
```

```
                3                                          PASS 3

       022615   000001 0610 03         7738          ADX1     1,DU              GET ADDRESS OF SECOND WORD OF MARK STACK
       022616   023104 7410 00         7739          STX1     BDI3              AND STORE IN INSTRUCTION SEQUENCE
       022617   000001 0610 03         7740          ADX1     1,DU              GET ADDRESS OF THIRD WORD OF MARK STACK
       022620   023106 7410 00         7741          STX1     BDI4              AND STORE IN INSTRUCTION SEQUENCE
    END OF BINARY CARD 00000385
       022621   000001 0610 03         7742          ADX1     1,DU              GET ADDRESS OF FOURTH WORD OF MARK STACK
       022622   023110 7410 00         7743          STX1     BDI6              AND STORE IN INSTRUCTION SEQUENCE
       022623   023102 2350 00         7744          LDA      BDI1              GET FIRST INSTRUCTION IN SEQUENCE
       022624   017474 7000 00         7745          TSX0     GAD               AND ADD TO INSTRUCTION SEQUENCE
       022625   023074 7210 00         7746          LXL1     BDCLR             GET BOUND POINTER OF ROW MODE
       022626   035217 0610 00         7747          ADX1     T$BOUND           MAKE POINTER ABSOLUTE
       022627   000000 2200 11         7748          LDX0     0,1               GET TYPE OF ENTRY IN XR - 0
       022630   016770 1000 03         7749          CMPX0    B$ROW,DU          IS IT A ROW ENTRY
       022631   777777 6010 00         7750          TNZ      $ERROR            NO - COMPILER ERROR
       022632   000001 2200 11         7751          LDX0     1,1               GET BOUND POINTER FOR ELEMENT OF ROW VALUE
       022633   023100 4400 00         7752          SXL0     BDED              AND STORE IN BOUND ELEMENT DECLARER
       022634   000001 7200 11         7753          LXL0     1,1               GET POINTER TO PROG TABLE FOR DECLARER
       022635   035221 0600 00         7754          ADX0     T$PROG            MAKE POINTER ABSOLUTE
       022636   000004 7220 10         7755          LXL2     4,0               GET LABEL FOR BOUND PROC ST LENGTH
       022637   017045 4420 00         7756          SXL2     PARAM             AND STORE AS PARAMETER
       022640   000003 7220 10         7757          LXL2     3,0               GET LABEL FOR PROCEDURE ENTRANCE
       022641   022674 7420 00         7758          STX2     BD11              AND SAVE
       022642   000002 2350 10         7759          LDA      2,0               GET ENVIRONMENT OF BOUNDS
       022643   000022 7710 00         7760          ARL      18                MOVE TO AL
       022644   020740 7500 00         7761          STC2     IDFYX             SAVE RETURN
       022645   020723 7100 00         7762          TRA      IDFYB             AND GET RANGE DIFFERENCE
       022646   022017 7000 00         7763          TSX0     TRAD              GET LINKED ADDRESS OF ST LEN OF PROC
    END OF BINARY CARD 00000386
       022647   023103 7510 70         7764          STCA     BDI2,70           AND STORE IN INSTRUCTION SEQUENCE
       022650   023105 2350 00         7765          LDA      BDIT1             GET TALLY WORD FOR INSTRUCTION SEQUENCE
       022651   017507 7000 00         7766          TSX0     GADL              AND ADD SEQUENCE TO GENERATED OUTPUT
       022652   020746 2340 00         7767          SZN      IDFYC             CHECK LL DIFFERENCE TO BOUND PROCEDURE
       022653   022657 6010 00         7768          TNZ      BD7               TRANSFER IF NOT ZERO
       022654   635017 2350 07         7769          LDA      EAA+D,DL          GET EAA 0,D INSTRUCTION IN A
       022655   017474 7000 00         7770          TSX0     GAD               AND ADD TO INSTRUCTION SEQUENCE
       022656   022672 7100 00         7771          TRA      BD10              AND CONTINUE
       022657   000000 2350 07         7772 BD7      LDA      0,DL              GET A ZERO IN A
       022660   020746 1550 00         7773          SSA      IDFYC             NEGATE LL DIFFERENCE
       022661   235017 2350 07         7774          LDA      LDA+D,DL          GET LDA 0,D INSTRUCTION
       022662   000002 0750 03         7775 BD8      ADA      2,DU              SET ADDRESS OF INSTRUCTION TO 2
       022663   017474 7000 00         7776          TSX0     GAD               AND ADD TO INSTRUCTION SEQUENCE
       022664   020746 0540 00         7777          AOS      IDFYC             DECREMENT NUMBER OF LEVELS YET TO CROSS
       022665   022670 6000 00         7778          TZE      BD9               TRANSFER IF DONE
       022666   235001 2350 07         7779          LDA      LDA+AU,DL         GET LDA 0,AU INSTRUCTION
       022667   022662 7100 00         7780          TRA      BD8               TRANSFER TO LOOP
       022670   635001 2350 07         7781 BD9      LDA      EAA+AU,DL         GET EAA 0,AU INSTRUCTION
       022671   017474 7000 00         7782          TSX0     GAD               AND ADD TO INSTRUCTION SEQUENCE
       022672   023106 2350 00         7783 BD10     LDA      BDI4              GET STORE IN THIRD MSCW WORD INSTRUCTION
       022673   017474 7000 00         7784          TSX0     GAD               AND ADD TO INSTRUCTION SEQUENCE
       022674   000000 2200 03         7785 BD11     LDX0     **,DU             GET LABEL FOR PROCEDURE ENTRANCE IN XR - 0
```

```
 END OF BINARY CARD 00000387
     022675  017045 4400 00       7786           SXL0     PARAM               AND STORE FOR LABEL ROUTINES
     022676  022017 7000 00       7787           TSX0     TRAD                GET LINKED LABEL IN AU
     022677  023107 7510 70       7788           STCA     BDI5,70             AND STORE IN INSTRUCTION SEQUENCE
     022700  023111 2350 00       7789           LDA      BDIT2               GET TALLY WORD FOR INSTRUCTION SEQUENCE
     022701  017507 7000 00       7790           TSX0     GADL                AND ADD TO INSTRUCTION SEQUENCE
     022702  023516 7000 00       7791           TSX0     MSCW                MARK THE STACK FOR PROCEDURE ENTRANCE
     022703  035234 2260 00       7792           LDX6     A$WORK              GET POINTER TO END OF WORKING STACK
     022704  000010 1660 03       7793           SBX6     2*WL,DU             GET POINTER TO CURRENT FIELD OR ELEMENT
     022705  000001 2350 16       7794           LDA      1,6                 GET ADDRESS OF POINTER IN AU
     022706  023112 7510 70       7795           STCA     BDI7,70             AND STORE IN INSTRUCTION SEQUENCE
     022707  023117 7510 70       7796           STCA     BDI21,70            AND STORE IN INSTRUCTION SEQUENCE
     022710  023075 2350 00       7797           LDA      BPOS                GET CURRENT POSITION
     022711  000001 0750 03       7798           ADA      1,DU                STEP OVER FLAG WORD
     022712  023113 7510 70       7799           STCA     BDI8,70             MAKE EAA OFFSET,AU INSTRUCTION
     022713  000037 6270 00       7800           EAX7     M$PTR               GET MODE OF A POINTER IN XR - 7
     022714  020377 7000 00       7801           TSX0     MBLK                MAKE A BLOCK FOR BOUND PROCEDURE PARAMETER
     022715  100000 2250 03       7802           LDX5     B$FC,DU             CLAIM ARGUMENT IS STACKED
     022716  000000 7450 16       7803           STX5     0,6                 AND STORE IN BLOCK
     022717  000001 2350 16       7804           LDA      1,6                 GET ADDRESS OF ACTUAL PARAMETER IN AU
     022720  023114 7510 70       7805           STCA     BDI9,70             AND STORE IN INSTRUCTION SEQUENCE
     022721  023115 2350 00       7806           LDA      BDIT3               GET TALLY WORD FOR INSTRUCTION SEQUENCE
     022722  017507 7000 00       7807           TSX0     GADL                AND ADD TO GENERATED OUTPUT
 END OF BINARY CARD 00000388
     022723  000033 6270 00       7808           EAX7     M$MS                GET MODE OF PSEUDO RESULT OF BOUND PROCEDURE
     022724  017045 4470 00       7809           SXL7     PARAM               AND STORE FOR ENTER ROUTINE
     022725  023572 7000 00       7810           TSX0     ENTER               AND ENTER BOUND PROCEDURE
     022726  023101 4500 00       7811           STZ      BDD                 INITIALIZE DIMENSION COUNT TO ZERO
     022727  023074 2270 00       7812           LDX7     BDCLR               GET ROW MODE IN XR - 7
     022730  023101 0540 00       7813 BD115     AOS      BDD                 INCREMENT DIMENSION COUNT
     022731  010630 7000 00       7814           TSX0     A$XFER              MAKE MODE POINTER ABSOLUTE
     022732  000000 2200 17       7815           LDX0     0,7                 GET TYPE OF MODE IN XR - 0
     022733  000001 2270 17       7816           LDX7     1,7                 GET DEROWED MODE IN XR - 7
     022734  016776 1000 03       7817           CMPX0    M$ROWE,DU           IS THIS AN END ROW MODE
     022735  022730 6010 00       7818           TNZ      BD115               NO - TRANSFER TO KEEP COUNTING DIMENSIONS
     022736  023101 7200 00       7819           LXL0     BDD                 GET DIMENSION OF ARRAY IN XR - 0
     022737  023120 7400 00       7820           STX0     BDI22               AND STORE IN INSTRUCTION SEQUENCE
     022740  023121 2350 00       7821           LDA      BDIT5               GET TALLY WORD FOR INSTRUCTION SEQUENCE
     022741  017507 7000 00       7822           TSX0     GADL                AND ADD SEQUENCE TO GENERATED OUTPUT
     022742  022103 7000 00       7823           TSX0     MTL                 GET TYPE OF ELEMENT MODE IN AU
     022743  624000 0750 07       7824           ADA      EAX4,DL             MAKE EAX4 TYPE INSTRUCTION
     022744  017474 7000 00       7825           TSX0     GAD                 AND ADD TO INSTRUCTION SEQUENCE
     022745  023073 2350 00       7826           LDA      BGEN                GET TSX0 R$RLGEN OR R$RHGEN INSTRUCTION
     022746  017474 7000 00       7827           TSX0     GAD                 AND ADD TO GENERATED OUTPUT
     022747  021007 7000 00       7828           TSX0     DELV                DELETE MARK STACK BLOCK
     022750  100000 2250 03       7829           LDX5     B$FC,DU             GET VALUE IS STACKED BIT
 END OF BINARY CARD 00000389
     022751  000007 6270 00       7830           EAX7     M$INT               GET INTEGER MODE IN XR - 7
     022752  020377 7000 00       7831           TSX0     MBLK                MAKE A BLOCK FOR MINUS ELEMENT COUNT
     022753  000000 7450 16       7832           STX5     0,6                 CLAIM VALUE IS STACKED
```

```
       022754  000001 2350 16        7833          LDA       1,6               GET ADDRESS OF COUNT IN AU
       022755  023122 7510 70        7834          STCA      BDI10,70          AND STORE IN INSTRUCTION SEQUENCE
       022756  000007 6270 00        7835          EAX7      M$INT             GET INTEGER MODE IN XR - 7
       022757  020377 7000 00        7836          TSX0      MBLK              MAKE A BLOCK FOR LENGTH OF ELEMENT
       022760  000000 7450 16        7837          STX5      0,6               CLAIM VALUE IS STACKED
       022761  000037 6270 00        7838          EAX7      M$PTR             GET POINTER MODE IN XR - 7
       022762  020377 7000 00        7839          TSX0      MBLK              MAKE A BLOCK FOR ELEMENT POINTER
       022763  000000 7450 16        7840          STX5      0,6               CLAIM VALUE IS STACKED
       022764  023074 2270 00        7841          LDX7      BDCLR             GET MODE OF ARRAY IN XR - 7
       022765  010630 7000 00        7842 BD12     TSX0      A$XFER            MAKE MODE POINTER ABSOLUTE
       022766  000000 2200 17        7843          LDX0      0,7               GET TYPE OF MODE IN XR - 0
       022767  000001 2270 17        7844          LDX7      1,7               GET DEROWED MODE IN XR - 7
       022770  016776 1000 03        7845          CMPX0     M$ROWE,DU         SEE IF TYPE OF MODE IS END OF ROW MODE
       022771  022765 6010 00        7846          TNZ       BD12              TRANSFER IF NOT
       022772  023100 7470 00        7847          STX7      BDED              STORE MODE OF ELEMENT IN ELEMENT DECLARER
       022773  010562 7000 00        7848          TSX0      A$RCHK            SEE IF MORE BOUND PROCEDURES TO CALL
       022774  023065 7100 00        7849          TRA       BD13              TRANSFER IF NO MORE PROCEDURES TO CALL
       022775  023122 2350 00        7850          LDA       BDI10             GET SZN COUNT,D INSTRUCTION
       022776  017474 7000 00        7851          TSX0      GAD               AND ADD TO GENERATED OUTPUT
    END OF BINARY CARD 00000390
       022777  035225 7200 00        7852          LXL0      T$LBL             GET ADDRESS OF NEXT LABEL TO BE GENERATED
       023000  017045 4400 00        7853          SXL0      PARAM             AND STORE AS PARAMETER
       023001  035225 2210 03        7854          LDX1      T$LBL,DU          GET POINTER TO LABEL TABLE CONTROL WORD
       023002  000000 6350 00        7855          EAA       1-1               PREPARE TO ALLOCATE ONE WORD
       023003  005663 7000 00        7856          TSX0      T$ALOC            ALLOCATE WORD IN LABEL TABLE
       023004  777777 4500 11        7857          STZ       -1,1              ZERO OUT ALLOCATED WORD
       023005  022017 7000 00        7858          TSX0      TRAD              GET LINKED LABEL IN AU
       023006  600004 0750 07        7859          ADA       TZE+IC,DL         GET TZE LBL,IC INSTRUCTION
       023007  017474 7000 00        7860          TSX0      GAD               AND ADD TO GENERATED OUTPUT
       023010  017045 2350 00        7861          LDA       PARAM             GET LABEL IN A
       023011  035235 7550 56        7862          STA       A$STACK,ID        AND STORE IN CONTROL STACK
       023012  005742 7170 00        7863          XED       T$SOVF            CHECK FOR STACK OVERFLOW
       023013  035226 2350 00        7864          LDA       T$GEN             GET ADDRESS FOR NEXT GENERATED INSTRUCTION
       023014  035235 7550 56        7865          STA       A$STACK,ID        AND STORE IN CONTROL STACK
       023015  005742 7170 00        7866          XED       T$SOVF            CHECK FOR STACK OVERFLOW
       023016  023076 2350 00        7867          LDA       BDX               GET RETURN
       023017  035235 7550 56        7868          STA       A$STACK,ID        AND STORE IN CONTROL STACK
       023020  005742 7170 00        7869          XED       T$SOVF            CHECK FOR STACK OVERFLOW
       023021  023074 2350 00        7870          LDA       BDCLR             GET CURRENT DECLARER
       023022  035235 7550 56        7871          STA       A$STACK,ID        AND STORE IN CONTROL STACK
       023023  005742 7170 00        7872          XED       T$SOVF            CHECK FOR STACK OVERFLOW
       023024  023075 2350 00        7873          LDA       BPOS              GET CURRENT POSITION
    END OF BINARY CARD 00000391
       023025  035235 7550 56        7874          STA       A$STACK,ID        AND STORE IN CONTROL STACK
       023026  005742 7170 00        7875          XED       T$SOVF            CHECK FOR STACK OVERFLOW
       023027  023100 2350 00        7876          LDA       BDED              GET DECLARER FOR ARRAY ELEMENT
       023030  023074 7550 00        7877          STA       BDCLR             MAKE CURRENT DECLARER
       023031  023075 4500 00        7878          STZ       BPOS              ZERO OUT CURRENT POSITION
       023032  022512 7000 00        7879          TSX0      BD                AND COMPILE CODE FOR ELEMENT
       023033  035235 2350 54        7880          LDA       A$STACK,DI        GET OLD POSITION
```

```
     023034  023075 7550 00          7881          STA     BPOS            AND RESTORE
     023035  035235 2350 54          7882          LDA     A$STACK,DI      GET OLD DECLARER
     023036  023074 7550 00          7883          STA     BDCLR           AND RESTORE
     023037  035235 2350 54          7884          LDA     A$STACK,DI      GET OLD RETURN
     023040  023076 7550 00          7885          STA     BDX             AND RESTORE
     023041  035234 2260 00          7886          LDX6    A$WORK          GET POINTER TO END OF WORKING STACK
     023042  000004 1660 03          7887          SBX6    WL,DU           GET POINTER TO POINTER BLOCK
     023043  000001 2350 16          7888          LDA     1,6             GET ADDRESS OF ELEMENT POINTER
     023044  023124 7510 70          7889          STCA    BDI12,70        AND STORE IN INSTRUCTION SEQUENCE
     023045  000004 1660 03          7890          SBX6    WL,DU           GET POINTER TO ELEMENT LENGTH BLOCK
     023046  000001 2350 16          7891          LDA     1,6             GET ADDRESS OF ELEMENT LENGTH IN AU
     023047  023123 7510 70          7892          STCA    BDI11,70        AND STORE IN INSTRUCTION SEQUENCE
     023050  000004 1660 03          7893          SBX6    WL,DU           GET POINTER TO COUNT BLOCK
     023051  000001 2350 16          7894          LDA     1,6             GET ADDRESS OF COUNT IN AU
     023052  023125 7510 70          7895          STCA    BDI13,70        AND STORE IN INSTRUCTION SEQUENCE
 END OF BINARY CARD 00000392
     023053  035226 7200 00          7896          LXL0    T$GEN           GET ADDRESS OF NEXT GENERATED INSTRUCTION
     023054  023126 7400 00          7897          STX0    BDI14           AND STORE IN INSTRUCTION SEQUENCE
     023055  035235 7200 54          7898          LXL0    A$STACK,DI      GET ADDRESS FOR LOOP IN XR = 0
     023056  000003 0600 03          7899          ADX0    3,DU            ADD THREE FOR THIRD FROM START IN SEQUENCE
     023057  023126 1400 00          7900          SSX0    BDI14           AND STORE CORRECT RELATIVE ADDRESS
     023060  023127 2350 00          7901          LDA     BDIT4           GET TALLY WORD FOR INSTRUCTION SEQUENCE
     023061  017507 7000 00          7902          TSX0    GADL            AND ADD TO GENERATED OUTPUT
     023062  035235 7200 54          7903          LXL0    A$STACK,DI      GET LABEL FOR LOOP EXIT
     023063  017045 4400 00          7904          SXL0    PARAM           AND STORE AS PARAMETER
     023064  022006 7000 00          7905          TSX0    LBL             DEFINE LABEL AT CURRENT ADDRESS
     023065  021007 7000 00          7906 BD13     TSX0    DELV            DELETE POINTER BLOCK
     023066  021007 7000 00          7907          TSX0    DELV            DELETE ELEMENT LENGTH BLOCK
     023067  021007 7000 00          7908          TSX0    DELV            DELETE COUNT BLOCK
     023070  023074 2270 00          7909          LDX7    BDCLR           GET MODE OF ARRAY IN XR = 7
     023071  010630 7000 00          7910          TSX0    A$XFER          MAKE MODE POINTER ABSOLUTE
     023072  022577 7100 00          7911          TRA     BD4             AND LOOP
     023073  000000 000000           7912 BGEN     ZERO
     023074  000000 000000           7913 BDCLR    ZERO
     023075  000000 000000           7914 BPOS     ZERO
     023076  000000 000000           7915 BDX      ZERO
     023077  000000 000000           7916 BPOS1    ZERO
     023100  000000 000000           7917 BDED     ZERO
 END OF BINARY CARD 00000393
     023101  000000 000000           7918 BDB      ZERO
     023102  000000 4500 17          7919 BDI1     STZ     0,D
     023103  000000 2350 04          7920 BDI2     LDA     0,IC
     023104  000000 7550 17          7921 BDI3     STA     0,D
     023105  023103 0003 00          7922 BDIT1    TALLY   BDI2,*-BDI2+1
     023106  000000 7550 17          7923 BDI4     STA     0,D
     023107  000000 6350 04          7924 BDI5     EAA     0,IC
     023110  000000 7550 17          7925 BDI6     STA     0,D
     023111  023107 0003 00          7926 BDIT2    TALLY   BDI5,*-BDI5+1
     023112  000000 2350 17          7927 BDI7     LDA     0,D
     023113  000000 6350 01          7928 BDI8     EAA     0,AU
```

```
          3                                                PASS 3

      023114   000000 7550 17        7929 BDI9     STA      0,D
      023115   023112 0004 00        7930 BDIT3    TALLY    BDI7,*-BDI7+1
      023116   000000 6210 17        7931 BDI20    EAX1     0,D
      023117   000000 2220 17        7932 BDI21    LDX2     0,D
      023120   000000 6230 00        7933 BDI22    EAX3     0
      023121   023116 0004 00        7934 BDIT5    TALLY    BDI20,*-BDI20+1
      023122   000000 2340 17        7935 BDI10    SZN      0,D
      023123   000000 3350 17        7936 BDI11    LCA      0,D
      023124   000000 0550 17        7937 BDI12    ASA      0,D
      023125   000000 0540 17        7938 BDI13    AOS      0,D
      023126   000000 6040 04        7939 BDI14    TMI      0,IC
   END OF BINARY CARD 00000394
      023127   023123 0005 00        7940 BDIT4    TALLY    BDI11,*-BDI11+1
      023130   023212 7400 00        7941 SRNGE    STX0     SRNGX              SAVE RETURN
      023131   017042 2350 00        7942          LDA      LLINK              GET CURRENT ENVIRONMENT
      023132   035235 7550 56        7943          STA      A$STACK,ID         AND STORE IN CONTROL STACK
      023133   005742 7170 00        7944          XED      T$SOVF             CHECK FOR STACK OVERFLOW
      023134   017040 2350 00        7945          LDA      MAXS               GET MAXIMUM EXTENT OF ALLOCATED TEMP
      023135   000022 7710 00        7946          ARL      18                 MOVE IT TO AL
      023136   017037 0750 00        7947          ADA      SP                 GET TEMPORARY STACK POINTER IN AU
      023137   035235 7550 56        7948          STA      A$STACK,ID         AND STORE IN CONTROL STACK
      023140   005742 7170 00        7949          XED      T$SOVF             CHECK FOR STACK OVERFLOW
      023141   035234 2200 00        7950          LDX0     A$WORK             GET POINTER TO END OF WORKING STACK
      023142   035214 1600 00        7951          SBX0     T$WORK             MAKE END POINTER RELATIVE
      023143   000000 6350 10        7952          EAA      0,0                GET LENGTH OF WORKING STACK IN AU
      023144   035235 7550 56        7953          STA      A$STACK,ID         AND STORE IN CONTROL STACK
      023145   005742 7170 00        7954          XED      T$SOVF             CHECK FOR STACK OVERFLOW
      023146   035235 2200 00        7955          LDX0     A$STACK            GET POINTER TO END OF CONTROL STACK
      023147   000003 1600 03        7956          SBX0     3,DU               GET POINTER TO RANGE MARK JUST PUT IN STACK
      023150   035215 1600 00        7957          SBX0     T$STACK            MAKE POINTER RELATIVE
      023151   017042 7400 00        7958          STX0     LLINK              AND STORE AS NEW LINK TO RANGE MARK
      023152   017044 2260 00        7959          LDX6     LSTMK              GET POINTER TO LAST MARK IN STACK
      023153   035214 0660 00        7960          ADX6     T$WORK             MAKE POINTER ABSOLUTE
      023154   000001 2350 16        7961          LDA      1,6                GET ADDRESS OF LAST MS IN AU
   END OF BINARY CARD 00000395
      023155   000001 6350 01        7962          EAA      1,AU               GET ADDRESS OF SAVED S REGISTER IN AU
      023156   023213 0750 00        7963          ADA      SRNGI              ADD STX S,0,D INSTRUCTION
      023157   017474 7000 00        7964          TSX0     GAD                AND ADD TO GENERATED OUTPUT
      023160   000033 2270 03        7965          LDX7     M$MS,DU            GET MODE OF MARK STACK IN XR - 7
      023161   020377 7000 00        7966          TSX0     MBLK               AND MAKE A BLOCK FOR IT
      023162   020526 7000 00        7967          TSX0     STYPE              STORE TYPE OF STACK IN MS
      023163   035234 2260 00        7968          LDX6     A$WORK             GET POINTER TO END OF WORKING STACK
      023164   000004 1660 03        7969          SBX6     WL,DU              GET POINTER TO LAST BLOCK IN WORK
      023165   000001 2350 16        7970          LDA      1,6                GET ADDRESS OF MS IN AU
      023166   000001 6350 01        7971          EAA      1,AU               GET ADDRESS OF SAVED S REGISTER IN AU
      023167   450017 0750 07        7972          ADA      STZ+D,DL           ADD STZ 0,D INSTRUCTION
      023170   017474 7000 00        7973          TSX0     GAD                AND ADD TO GENERATED OUTPUT
      023171   017045 7210 00        7974          LXL1     PARAM              GET POINTER TO RANGE JUST ENTERED
      023172   017042 4410 00        7975          SXL1     LLINK              STORE RANGE POINTER IN LLINK
      023173   035221 0610 00        7976          ADX1     T$PROG             MAKE PROG TABLE POINTER ABSOLUTE
```

```
           3                                                 PASS 3

     023174  000002 7210 11        7977          LXL1     2,1              GET POINTER TO DEFINITION CHAIN FOR THIS RANGE
     023175  023212 6040 00        7978 SRNG1    TMI      SRNGX            TRANSFER IF NO MORE DEFINITIONS IN THIS RANGE
     023176  035220 0610 00        7979          ADX1     T$DEF            MAKE DEFINITION POINTER ABSOLUTE
     023177  000001 7220 11        7980          LXL2     1,1              GET POINTER TO NEXT DEFINITION IN CHAIN IN XR - 2
     023200  000002 2270 11        7981          LDX7     2,1              GET MODE OF IDENTIFIER IN XR - 7
     023201  000031 1070 03        7982          CMPX7    M$LBL,DU         SEE IF IT IS A LABEL
     023202  023210 6000 00        7983          TZE      SRNG2            TRANSFER IF IT IS A LABEL
   END OF BINARY CARD 00000396
     023203  017037 2200 00        7984          LDX0     SP               GET CURRENT STACK POINTER
     023204  000003 7400 11        7985          STX0     3,1              AND STORE AS ADDRESS OF IDENTIFIER
     023205  020377 7000 00        7986          TSX0     MBLK             MAKE A BLOCK FOR IDENTIFIER
     023206  100000 2250 03        7987          LDX5     B$FC,DU          GET A VALUE IS STACKED BIT
     023207  000000 7450 16        7988          STX5     0,6              AND STORE IN BLOCK
     023210  000000 6210 12        7989 SRNG2    EAX1     0,2              GET POINTER TO NEXT DEFINITION IN XR - 1
     023211  023175 7100 00        7990          TRA      SRNG1            AND LOOP
     023212  000000 7100 00        7991 SRNGX    TRA      **               AND RETURN
     023213  000000 7460 17        7992 SRNGI    STX      S,0,D
     023214  023260 7400 00        7993 ERNGE    STX0     ERNGX            SAVE RETURN
     023215  035234 2260 00        7994          LDX6     A$WORK           GET POINTER TO END OF WORKING STACK
     023216  000004 1660 03        7995          SBX6     WL,DU            GET POINTER TO LAST BLOCK IN WORKING STACK
     023217  000000 7270 16        7996          LXL7     0,6              GET MODE OF RESULT OF RANGE IN XR - 7
     023220  023233 7470 00        7997          STX7     ERNGM            AND SAVE
     023221  023261 4500 00        7998          STZ      ERNGF            INITIALIZE RANGE VALUE FLAG
     023222  020204 7000 00        7999          TSX0     MVA              MAKE RANGE VALUE AVAILABLE
     023223  023261 7500 00        8000          STC2     ERNGF            SET FLAG INDICATING VALUE IN ACCUMULATOR
     023224  023262 7550 00        8001          STA      ERNGT            SAVE ADDRESS OF VALUE
     023225  023663 7000 00        8002          TSX0     POP              POP STACK BACK THROUGH LAST MARK
     023226  035235 0110 54        8003          NOP      A$STACK,DI       DELETE POINTER WHERE WORK WAS MARKED
     023227  035235 2210 54        8004          LDX1     A$STACK,DI       GET SAVED STACK POINTER
     023230  017037 7410 00        8005          STX1     SP               AND RESTORE STACK POINTER
   END OF BINARY CARD 00000397
     023231  035235 2350 54        8006          LDA      A$STACK,DI       GET PREVIOUS ENVIRONMENT
     023232  017042 7550 00        8007          STA      LLINK            AND RESTORE
     023233  000000 2270 03        8008 ERNGM    LDX7     **,DU            GET MODE OF RANGE IN XR - 7
     023234  020377 7000 00        8009          TSX0     MBLK             MAKE A BLOCK FOR RANGE VALUE
     023235  100000 2250 03        8010          LDX5     B$FC,DU          GET VALUE IS STACKED BIT
     023236  000000 7450 16        8011          STX5     0,6              AND STORE IN BLOCK
     023237  023262 2350 00        8012          LDA      ERNGT            GET ADDRESS OF VALUE IN A
     023240  023261 2340 00        8013          SZN      ERNGF            SEE IF VALUE IS IN ACCUMULATOR
     023241  023247 6010 00        8014          TNZ      ERNG1            TRANSFER IF VALUE IS IN ACCUMULATOR
     023242  000001 2360 16        8015          LDQ      1,6              GET ADDRESS OF DESTINATION OF MOVE IN Q
     023243  000017 0760 07        8016          ADQ      D,DL             ADD D REGISTER MODIFICATION
     023244  000000 7270 16        8017          LXL7     0,6              GET MODE OF VALUE IN XR - 7
     023245  017531 7000 00        8018          TSX0     MOVE             AND MOVE RANGE VALUE TO STACK
     023246  023252 7100 00        8019          TRA      ERNG2            AND CONTINUE
     023247  000001 0750 16        8020 ERNG1    ADA      1,6              ADD ADDRESS WHERE VALUE IS TO BE STORED
     023250  000017 0750 07        8021          ADA      D,DL             ADD D REGISTER MODIFICATION
     023251  017474 7000 00        8022          TSX0     GAD              AND ADD STORE COMMAND TO GENERATED OUTPUT
     023252  017044 2260 00        8023 ERNG2    LDX6     LSTMK            GET POINTER TO LAST MARK
     023253  035214 0660 00        8024          ADX6     T$WORK           MAKE POINTER ABSOLUTE
```

```
              3                                            PASS 3

       023254  000001 2350 16          8025          LDA      1,6               GET ADDRESS OF LAST MARK IN AU
       023255  000001 6350 01          8026          EAA      1,AU              GET ADDRESS OF SAVED S IN AU
       023256  023263 0750 00          8027          ADA      ERNGI             ADD LDX S,0,D INSTRUCTION
    END OF BINARY CARD 00000398
       023257  017474 7000 00          8028          TSX0     GAD               AND ADD TO GENERATED OUTPUT
       023260  000000 7100 00          8029 ERNGX    TRA      **                AND EXIT
       023261  000000 000000           8030 ERNGF    ZERO
       023262  000000 000000           8031 ERNGT    ZERO
       023263  000000 2260 17          8032 ERNGI    LDX      S,0,D
       023264  023336 7400 00          8033 LL       STX0     LLX               SAVE RETURN
       023265  017042 2350 00          8034          LDA      LLINK             GET CURRENT ENVIRONMENT
       023266  035235 7550 56          8035          STA      A$STACK,ID        AND STORE IN CONTROL STACK
       023267  005742 7170 00          8036          XED      T$SOVF            CHECK FOR STACK OVERFLOW
       023270  017040 2350 00          8037          LDA      MAXS              GET MAXIMUM TEMP USED IN PREVIOUS ENVIRONMENT
       023271  000022 7710 00          8038          ARL      18                MOVE TO AL
       023272  017037 0750 00          8039          ADA      SP                GET CURRENT EXTENT OF TEMPORARY STORAGE
       023273  035235 7550 56          8040          STA      A$STACK,ID        AND STORE IN CONTROL STACK
       023274  005742 7170 00          8041          XED      T$SOVF            CHECK FOR STACK OVERFLOW
       023275  035234 2200 00          8042          LDX0     A$WORK            GET POINTER TO END OF WORKING STACK
       023276  035214 1600 00          8043          SBX0     T$WORK            MAKE POINTER RELATIVE
       023277  000000 6350 10          8044          EAA      0,0               GET LENGTH OF WORKING STACK IN AU
       023300  035235 7550 56          8045          STA      A$STACK,ID        AND STORE IN CONTROL STACK
       023301  005742 7170 00          8046          XED      T$SOVF            CHECK FOR STACK OVERFLOW
       023302  035235 2200 00          8047          LDX0     A$STACK           GET POINTER TO END OF CONTROL STACK
       023303  000003 1600 03          8048          SBX0     3,DU              GET POINTER TO ENVIRONMENT JUST STACKED
       023304  017045 2350 00          8049          LDA      PARAM             GET POINTER TO NEW ENVIRONMENT IN AL
    END OF BINARY CARD 00000399
       023305  777777 2360 03          8050          LDQ      -1,DU             GET A MASK TO COMPARE AL ONLY
       023306  017042 4500 00          8051          STZ      LLINK             ZERO OUT DELTA LL IN LOWER HALF OF LLINK
       023307  000000 2340 10          8052 LL1      SZN      0,0               SEE IF ENVIRONMENT IS LEXICOGRAPHICAL LEVEL
       023310  023315 6050 00          8053          TPL      LL2               TRANSFER IF NOT
       023311  017042 0540 00          8054          AOS      LLINK             INCREMENT DELTA LL
       023312  000000 2200 10          8055          LDX0     0,0               GET POINTER TO SURROUNDING RANGE IN XR - 0
       023313  400000 6600 03          8056          ERX0     =0400000,DU       FLIP SIGN BIT
       023314  023322 7100 00          8057          TRA      LL3               AND CONTINUE
       023315  000000 2110 10          8058 LL2      CMK      0,0               SEE IF DESIRED ENVIRONMENT HAS BEEN FOUND
       023316  023324 6000 00          8059          TZE      LL4               TRANSFER IF FOUND
       023317  035215 1000 00          8060          CMPX0    T$STACK           SEE IF CURRENT ENVIRONMENT IS GLOBAL
       023320  777777 6000 00          8061          TZE      $ERROR            TRANSFER IF YES - ERROR NO BIGGER ENVIRONMENT
       023321  000000 2200 10          8062          LDX0     0,0               GET SURROUNDING RANGE IN XR - 0
       023322  035215 0600 00          8063 LL3      ADX0     T$STACK           MAKE ENVIRONMENT POINTER ABSOLUTE
       023323  023307 7100 00          8064          TRA      LL1               AND LOOP
       023324  035215 1600 00          8065 LL4      SBX0     T$STACK           MAKE ENVIRONMENT POINTER RELATIVE
       023325  400000 6600 03          8066          ERX0     =0400000,DU       FLIP SIGN BIT
       023326  017042 7400 00          8067          STX0     LLINK             STORE NEW ENVIRONMENT IN LLINK
       023327  017037 4500 00          8068          STZ      SP                RESET STACK POINTER
       023330  017040 4500 00          8069          STZ      MAXS              RESET MAXIMUM STACK POINTER
       023331  000035 6270 00          8070          EAX7     M$MSCW            GET MODE OF MARK STACK CONTROL WORD
       023332  020377 7000 00          8071          TSX0     MBLK              AND MAKE A BLOCK FOR IT
    END OF BINARY CARD 00000400
```

```
     023333  100000 2250 03        8072        LDX5    B$FC,DU          GET VALUE IS STACKED BIT
     023334  000000 7450 16        8073        STX5    0,6              AND STORE FLAGS IN BLOCK
     023335  023652 7000 00        8074        TSX0    PUSH             MARK WORKING STACK IN CONTROL STACK
     023336  000000 7100 00        8075 LLX    TRA     **               AND EXIT
     023337  023355 7400 00        8076 LLE    STX0    LLEX             SAVE RETURN
     023340  017040 2200 00        8077        LDX0    MAXS             GET MAXIMUM LENGTH OF TEMP NEEDED IN PROCEDURE
     023341  017041 7400 00        8078        STX0    MAXST            AND SAVE
     023342  017042 2350 00        8079        LDA     LLINK            GET DELTA LL AND ENVIRONMENT OF PROCEDURE
     023343  017043 7550 00        8080        STA     DLL              AND SAVE
     023344  023663 7000 00        8081        TSX0    POP              RESTORE WORKING STACK TO CONDITION WHEN MARKED
     023345  035235 0110 54        8082        NOP     A$STACK,DI       DELETE POINTER TO WHERE WORK WAS MARKED
     023346  035235 2350 54        8083        LDA     A$STACK,DI       GET SP AND MAXS
     023347  000000 6200 01        8084        EAX0    0,AU             GET SAVED SP IN XR = 0
     023350  017037 7400 00        8085        STX0    SP               AND RESTORE STACK POINTER
     023351  000000 6200 05        8086        EAX0    0,AL             GET SAVED MAXS IN XR = 0
     023352  017040 7400 00        8087        STX0    MAXS             AND RESTORE MAX STACK POINTER
     023353  035235 2350 54        8088        LDA     A$STACK,DI       GET PREVIOUS ENVIRONMENT
     023354  017042 7550 00        8089        STA     LLINK            AND MAKE IT CURRENT ENVIRONMENT
     023355  000000 7100 00        8090 LLEX   TRA     **               AND EXIT
     023356  023400 7400 00        8091 FORMP  STX0    FORMX            SAVE RETURN
     023357  017045 7210 00        8092        LXL1    PARAM            GET DEFINITION OF FORMAL PARAMETER IN XR - 1
     023360  035220 0610 00        8093        ADX1    T$DEF            MAKE POINTER ABSOLUTE
 END OF BINARY CARD 00000401
     023361  000003 2220 11        8094        LDX2    3,1              GET ASSIGNED ADDRESS OF FORMAL PARAMETER
     023362  035234 2260 00        8095        LDX6    A$WORK           GET ADDRESS OF END OF WORK IN XR - 6
     023363  000004 1660 03        8096 FORM1  SBX6    WL,DU            STEP BACK TO LAST BLOCK
     023364  000001 1020 16        8097        CMPX2   1,6              IS THIS THE FORMAL PARAMETER BEING ASSIGNED
     023365  023363 6010 00        8098        TNZ     FORM1            LOOP IF NOT
     023366  100000 2250 03        8099        LDX5    B$FC,DU          GET VALUE IS STACKED BIT
     023367  000000 2450 16        8100        ORSX5   0,6              SET STACKED BIT IN FORMAL PARAMETER DEFINITION
     023370  000002 2270 11        8101        LDX7    2,1              GET MODE OF PARAMETER IN XR - 7
     023371  000003 2350 11        8102        LDA     3,1              GET ASSIGNED LOCATION OF FORMAL PARAMETER
     023372  000003 2360 11        8103        LDQ     3,1              GET ASSIGNED LOCATION OF FORMAL PARAMETER
     023373  777776 6350 01        8104        EAA     -M$MSL,AU        ZERO OUT AL AND SUBTRACT MARK STACK LENGTH
     023374  000011 0750 07        8105        ADA     =011,DL          ADD X REGISTER 1 MODIFICATION
     023375  000000 6360 02        8106        EAQ     0,QU             ZERO OUT QL
     023376  000012 0760 07        8107        ADQ     =012,DL          ADD X REGISTER 2 MODIFICATION
     023377  017531 7000 00        8108        TSX0    MOVE             COMPILE MOVE FROM ACTUAL TO FORMAL PARAMETERS
     023400  000000 7100 00        8109 FORMX  TRA     **               AND RETURN
     023401  023475 7400 00        8110 EPBV   STX0    EPDVX            SAVE RETURN
     023402  017045 7270 00        8111        LXL7    PARAM            GET MODE OF PROCEDURE IN XR - 7
     023403  020377 7000 00        8112        TSX0    MBLK             PUSH A BLOCK IN THE WORKING STACK FOR THIS VALUE
     023404  100000 2250 03        8113        LDX5    B$FC,DU          GET STACKED BIT
     023405  000000 7450 16        8114        STX5    0,6              AND STORE IN BLOCK
     023406  000001 2200 16        8115        LDX0    1,6              GET ASSIGNED LOCATION FOR VALUE
 END OF BINARY CARD 00000402
     023407  023476 7400 00        8116        STX0    EPDV0            AND STORE IN INSTRUCTION SEQUENCE
     023410  000001 0600 03        8117        ADX0    1,DU             GET SECOND ASSIGNED LOCATION FOR VALUE
     023411  023501 7400 00        8118        STX0    EPDV2            AND STORE IN INSTRUCTION SEQUENCE
     023412  000001 0600 03        8119        ADX0    1,DU             GET THIRD ASSIGNED LOCATION FOR VALUE
```

```
      023413  023503 7400 00        8120          STX0      EPDV6           AND STORE IN INSTRUCTION SEQUENCE
      023414  000001 0600 03        8121          ADX0      1,DU            GET FOURTH ASSIGNED LOCATION FOR VALUE
      023415  023505 7400 00        8122          STX0      EPDV5           AND STORE IN INSTRUCTION SEQUENCE
      023416  017041 2200 00        8123          LDX0      MAXST           GET AMOUNT OF STATIC TEMP REQUIRED BY PROCEDURE
      023417  023477 7400 00        8124          STX0      EPDV1           AND STORE IN INSTRUCTION SEQUENCE
      023420  023476 2350 00        8125          LDA       EPDV0           GET NEXT INSTRUCTION
      023421  017474 7000 00        8126          TSX0      GAD             AND ADD TO GENERATED OUTPUT
      023422  023477 2350 00        8127          LDA       EPDV1           GET NEXT INSTRUCTION
      023423  017474 7000 00        8128          TSX0      GAD             AND ADD TO GENERATED OUTPUT
      023424  000005 6350 00        8129          EAA       5               PREPARE TO ALLOCATE A 5 WORD TYPE
      023425  035227 2210 03        8130          LDX1      T$TYPE,DU       GET POINTER TO TYPE TABLE CONTROL WORD IN XR - 1
      023426  005663 7000 00        8131          TSX0      T$ALOC          ALLOCATE SPACE IN THE TYPE TABLE
      023427  027215 6350 00        8132          EAA       T$PTR           GET A POINTER TYPE IN AU
      023430  000000 7550 11        8133          STA       0,1             AND STORE IN TYPE
      023431  000001 7550 11        8134          STA       1,1             AND STORE IN TYPE
      023432  000002 7550 11        8135          STA       2,1             AND STORE IN TYPE
      023433  000003 7550 11        8136          STA       3,1             AND STORE IN TYPE
      023434  017041 2350 00        8137          LDA       MAXST           GET LENGTH OF STATIC TEMP REQUIRED IN AU
   END OF BINARY CARD 00000403
      023435  000004 1750 03        8138          SBA       4,DU            SUBTRACT LENGTH OF BASE MSCW
      023436  000022 7710 00        8139          ARL       18              MOVE LENGTH TO AL
      023437  027222 0750 03        8140          ADA       T$SKIP,DU       ADD SKIP TYPE
      023440  000004 7550 11        8141          STA       4,1             AND STORE IN TYPE
      023441  035227 1610 00        8142          SBX1      T$TYPE          MAKE TYPE TABLE POINTER RELATIVE
      023442  022110 7000 00        8143          TSX0      TL              GET TYPE POINTER IN AU
      023443  023500 7510 70        8144          STCA      EPDV3,70        AND STORE IN INSTRUCTION SEQUENCE
      023444  023502 2350 00        8145          LDA       EPDVT           GET TALLY WORD TO INSTRUCTION SEQUENCE
      023445  017507 7000 00        8146          TSX0      GADL            AND ADD SEQUENCE TO GENERATED CODE
      023446  017043 2220 00        8147          LDX2      DLL             GET PROCEDURE DENOTATIONS'S LL
      023447  023453 6010 00        8148          TNZ       EPV1            TRANSFER IF NOT GLOBAL
      023450  635000 2350 07        8149          LDA       EAA,DL          GET AN EAA 0,DL INSTRUCTION
      023451  017474 7000 00        8150          TSX0      GAD             AND ADD TO GENERATED CODE
      023452  023473 7100 00        8151          TRA       EPV5            CONTINUE
      023453  017043 7220 00        8152 EPV1     LXL2      DLL             GET DELTA LL OF PROCEDURE DENOTATION
      023454  023460 6010 00        8153          TNZ       EPV2            TRANSFER IF NOT THE SURROUNDING RANGE
      023455  635017 2350 07        8154          LDA       EAA+D,DL        GET EAA 0,D INSTRUCTION
      023456  017474 7000 00        8155          TSX0      GAD             AND ADD TO GENERATED CODE
      023457  023473 7100 00        8156          TRA       EPV5            CONTINUE
      023460  235017 2350 07        8157 EPV2     LDA       LDA+D,DL        GET LDA 0,D INSTRUCTION
      023461  000002 0750 03        8158          ADA       2,DU            MAKE IT LDA 2,D
      023462  017474 7000 00        8159          TSX0      GAD             AND ADD TO GENERATED CODE
   END OF BINARY CARD 00000404
      023463  000001 1620 03        8160 EPV3     SBX2      1,DU            SEE IF IN NEXT SURROUNDING RANGE
      023464  023471 6000 00        8161          TZE       EPV4            TRANSFER IF YES
      023465  235001 2350 07        8162          LDA       LDA+AU,DL       GET LDA 0,AU INSTRUCTION
      023466  000002 0750 03        8163          ADA       2,DU            MAKE IT LDA 2,AU
      023467  017474 7000 00        8164          TSX0      GAD             AND ADD TO GENERATED CODE
      023470  023463 7100 00        8165          TRA       EPV3            AND LOOP
      023471  635001 2350 07        8166 EPV4     LDA       EAA+AU,DL       GET EAA 0,AU INSTRUCTION
      023472  017474 7000 00        8167          TSX0      GAD             AND ADD TO GENERATED CODE
```

```
              3                                              PASS 3

     023473  023503 2350 00          8168 EPV5    LDA     EPDV6              GET FOLLOWING INSTRUCTION
     023474  017474 7000 00          8169         TSX0    GAD                AND ADD TO GENERATED CODE
     023475  000000 7100 00          8170 EPDVX   TRA     **                 AND RETURN
     023476  000000 4500 17          8171 EPDV0   STZ     0,D
     023477  000000 6350 00          8172 EPDV1   EAA     0
     023500  000000 0750 07          8173 EPDV3   ADA     0,DL
     023501  000000 7550 17          8174 EPDV2   STA     0,D
     023502  023500 0003 00          8175 EPDVT   TALLY   EPDV3,*-EPDV3+1
     023503  000000 7550 17          8176 EPDV6   STA     0,D
     023504  000000 6350 04          8177 EPDV4   EAA     0,IC
     023505  000000 7550 17          8178 EPDV5   STA     0,D
     023506  023504 0003 00          8179 EPDTT   TALLY   EPDV4,*-EPDV4+1
     023507  023515 7400 00          8180 EPDE    STX0    EPDEX              SAVE RETURN
     023510  022017 7000 00          8181         TSX0    TRAD               GET LINKED LABEL VALUE FOR INSTRUCTION
  END OF BINARY CARD 00000405
     023511  000000 6200 01          8182         EAX0    0,AU               GET VALUE IN XR - 0
     023512  023504 7400 00          8183         STX0    EPDV4              AND STORE IN INSTRUCTION SEQUENCE
     023513  023506 2350 00          8184         LDA     EPDTT              GET TALLY WORD FOR SEQUENCE
     023514  017507 7000 00          8185         TSX0    GADL               AND ADD TO GENERATED CODE
     023515  000000 7100 00          8186 EPDEX   TRA     **                 AND RETURN
     023516  023547 7400 00          8187 MSCW    STX0    MSCWX              SAVE RETURN
     023517  035234 2260 00          8188         LDX6    A$WORK             GET POINTER TO END OF WORKING STACK
     023520  000004 1660 03          8189         SBX6    WL,DU              GET POINTER TO LAST BLOCK IN WORKING STACK
     023521  020451 7000 00          8190         TSX0    FS                 FORCE PROCEDURE VALUE TO STACK
     023522  000001 2200 16          8191         LDX0    1,6                GET ADDRESS ASSIGNED TO PROCEDURE VALUE
     023523  023551 7400 00          8192         STX0    MSCW1              AND STORE IN INSTRUCTION SEQUENCE
     023524  023555 7400 00          8193         STX0    MSCW4              AND STORE IN INSTRUCTION SEQUENCE
     023525  000001 0600 03          8194         ADX0    1,DU               GET ADDRESS OF SECOND WORD OF PROCEDURE VALUE
     023526  023552 7400 00          8195         STX0    MSCW2              AND STORE IN INSTRUCTION SEQUENCE
     023527  023566 7400 00          8196         STX0    MSCW7              AND STORE IN INSTRUCTION SEQUENCE
     023530  023570 7400 00          8197         STX0    MSCW9              AND STORE IN INSTRUCTION SEQUENCE
     023531  000001 0600 03          8198         ADX0    1,DU               GET ADDRESS OF THIRD WORD OF PROCEDURE VALUE
     023532  023561 7400 00          8199         STX0    MSCW5              AND STORE IN INSTRUCTION SEQUENCE
     023533  023565 7400 00          8200         STX0    MSCW6              AND STORE IN INSTRUCTION SEQUENCE
     023534  023567 7400 00          8201         STX0    MSCW8              AND STORE IN INSTRUCTION SEQUENCE
     023535  000035 6270 00          8202         EAX7    M$MSCW             GET NEW MODE OF PROCEDURE VALUE IN XR - 7
     023536  000000 4470 16          8203         SXL7    0,6                AND STORE IN PROCEDURE BLOCK
  END OF BINARY CARD 00000406
     023537  017044 2260 00          8204         LDX6    LSTMK              GET POINTER TO LAST MARK IN STACK
     023540  035214 0660 00          8205         ADX6    T$WORK             MAKE BLOCK POINTER ABSOLUTE
     023541  000001 2200 16          8206         LDX0    1,6                GET ADDRESS OF LAST MARK IN STACK
     023542  000001 0600 03          8207         ADX0    1,DU               GET ADDRESS OF SAVED S REGISTER
     023543  023550 7400 00          8208         STX0    MSCW0              AND STORE IN INSTRUCTION SEQUENCE
     023544  020526 7000 00          8209         TSX0    STYPE              GET TYPE OF STACK AND LINKED POINTER IN AU
     023545  023571 2350 00          8210         LDA     MSCWT              GET TALLY WORD FOR INSTRUCTION SEQUENCE
     023546  017507 7000 00          8211         TSX0    GADL               AND ADD SEQUENCE TO GENERATED CODE
     023547  000000 7100 00          8212 MSCWX   TRA     **                 AND RETURN
     023550  000000 7460 17          8213 MSCW0   STX     S,0,D
     023551  000000 6210 17          8214 MSCW1   EAX1    0,D
     023552  000000 2220 17          8215 MSCW2   LDX2    0,D
```

```
              3                                               PASS 3

      023553  026116 7000 00          8216         TSX0     R$PLGEN
      023554  000000 6200 01          8217         EAX0     0,AU
      023555  000000 6350 17          8218 MSCW4   EAA      0,D
      023556  027230 0750 07          8219         ADA      T$MSCWT,DL
      023557  000000 7550 10          8220         STA      0,0
      023560  000001 4500 10          8221         STZ      1,0
      023561  000000 2350 17          8222 MSCW5   LDA      0,D
      023562  000002 7550 10          8223         STA      2,0
      023563  000000 6350 17          8224         EAA      0,D
      023564  000003 7550 10          8225         STA      3,0
   END OF BINARY CARD 00000407
      023565  000000 7400 17          8226 MSCW6   STX0     0,7
      023566  000000 7200 17          8227 MSCW7   LXL0     0,7
      023567  000000 4400 17          8228 MSCW8   SXL0     0,7
      023570  000000 4500 17          8229 MSCW9   STZ      0,7
      023571  023550 0022 00          8230 MSCWT   TALLY    MSCW0,*-MSCW0+1
      023572  023606 7400 00          8231 ENTER   STX0     ENTX                SAVE RETURN
      023573  023663 7000 00          8232         TSX0     POP                 DELETE BOTH STACKS BACK THROUGH MARK
      023574  017037 2200 00          8233         LDX0     SP                  GET ADDRESS OF MSCW IN STACK
      023575  023607 7400 00          8234         STX0     ENT0                AND STORE IN INSTRUCTION SEQUENCE
      023576  000002 0600 03          8235         ADX0     2,DU                GET ADDRESS OF THIRD WORD IN MSCW
      023577  023610 7400 00          8236         STX0     ENT1                AND STORE IN INSTRUCTION SEQUENCE
      023600  023612 2350 00          8237         LDA      ENTT                GET TALLY WORD FOR INSTRUCTION SEQUENCE
      023601  017507 7000 00          8238         TSX0     GADL                AND ADD SEQUENCE TO GENERATED CODE
      023602  017045 7270 00          8239         LXL7     PARAM               GET MODE OF RESULT IN XR - 7
      023603  020377 7000 00          8240         TSX0     MBLK                MAKE A BLOCK FOR THE RESULT OF THE PROCEDURE
      023604  100000 2250 03          8241         LDX5     B$FC,DU             GET FLAGS INDICATING STACKED VALUE

      023605  000000 7450 16          8242         STX5     0,6                 AND STORE IN NEWLY CREATED BLOCK
      023606  000000 7100 00          8243 ENTX    TRA      **                  AND RETURN
      023607  000000 6210 17          8244 ENT0    EAX1     0,D
      023610  000000 2220 17          8245 ENT1    LDX2     0,D
      023611  025365 7000 00          8246         TSX0     R$ENTER
      023612  023607 0004 00          8247 ENTT    TALLY    ENT0,*-ENT0+1
   END OF BINARY CARD 00000408
      023613  023617 7400 00          8248 EPDN    STX0     EPDNX               SAVE RETURN
      023614  022006 7000 00          8249         TSX0     LBL                 DEFINE CURRENT ADDRESS AS START OF PROCEDURE CODE
      023615  023620 2350 00          8250         LDA      EPDN0               GET EAX D,0,2 INSTRUCTION
      023616  017474 7000 00          8251         TSX0     GAD                 AND ADD TO GENERATED OUTPUT
      023617  000000 7100 00          8252 EPDNX   TRA      **                  AND EXIT
      023620  000000 6270 12          8253 EPDN0   EAX      D,0,2
      023621  023642 7400 00          8254 RETN    STX0     RETNX               SAVE RETURN
      023622  220017 2350 07          8255         LDA      LDX+D,DL            GET LDX0 0,D COMMAND
      023623  017474 7000 00          8256         TSX0     GAD                 AND ADD TO GENERATED CODE
      023624  023643 2350 00          8257         LDA      RETN0               GET LDX S,3,0 INSTRUCTION
      023625  017474 7000 00          8258         TSX0     GAD                 AND ADD TO GENERATED OUTPUT
      023626  035234 2260 00          8259         LDX6     A$WORK              GET POINTER TO END OF WORKING STACK
      023627  000004 1660 03          8260         SBX6     WL,DU               GET POINTER TO LAST BLOCK IN WORKING STACK
      023630  020204 7000 00          8261         TSX0     MVA                 MAKE PROCEDURE BODY VALUE AVAILABLE
      023631  023636 7100 00          8262         TRA      RET1                TRANSFER IF VALUE IS IN A REGISTER
      023632  000010 2360 07          8263         LDQ      =010,DL             GET DESTINATION ADDRESS AS 0,0
```

```
     023633   000000 7270 16         8264          LXL7     0,6                 GET MODE OF RESULT IN XR - 7
     023634   017531 7000 00         8265          TSX0     MOVE                MOVE RESULT FROM PROCEDURE TO CALLER
     023635   023640 7100 00         8266          TRA      RET2                AND CONTINUE
     023636   000010 0750 07         8267 RET1     ADA      =010,DL             ADD 0,0 TO STORE COMMAND
     023637   017474 7000 00         8268          TSX0     GAD                 AND ADD STORE VALUE IN CALLER TO GENERATED CODE
     023640   023644 2350 00         8269 RET2     LDA      RETN1               GET TSX0 R$RET INSTRUCTION
   END OF BINARY CARD 00000409
     023641   017474 7000 00         8270          TSX0     GAD                 AND ADD TO INSTRUCTION SEQUENCE
     023642   000000 7100 00         8271 RETNX    TRA      **                  AND RETURN
     023643   000003 2260 10         8272 RETNO    LDX      S,3,0
     023644   025400 7000 00         8273 RETN1    TSX0     R$RET
     023645   023651 7400 00         8274 DLEN     STX0     DLENX               SAVE RETURN
     023646   022006 7000 00         8275          TSX0     LBL                 DEFINE LABEL FOR PARAMETER

     023647   017041 2350 00         8276          LDA      MAXST               GET GREATEST EXTENT OF STACK
     023650   017474 7000 00         8277          TSX0     GAD                 AND ADD TO GENERATED OUTPUT
     023651   000000 7100 00         8278 DLENX    TRA      **                  AND EXIT
     023652   023662 7400 00         8279 PUSH     STX0     PUSHX               SAVE RETURN
     023653   017044 2200 00         8280          LDX0     LSTMK               GET POINTER TO LAST MARK
     023654   035235 7400 56         8281          STX0     A$STACK,ID          AND SAVE IN CONTROL STACK
     023655   005742 7170 00         8282          XED      T$SOVF              CHECK FOR STACK OVERFLOW
     023656   035234 2200 00         8283          LDX0     A$WORK              GET POINTER TO END OF WORKING STACK
     023657   035214 1600 00         8284          SBX0     T$WORK              MAKE POINTER RELATIVE
     023660   000004 1600 03         8285          SBX0     WL,DU               MAKE IT POINT BELOW MSCW BLOCK
     023661   017044 7400 00         8286          STX0     LSTMK               AND STORE AS NEW LAST MARK
     023662   000000 7100 00         8287 PUSHX    TRA      **                  AND RETURN
     023663   023673 7400 00         8288 POP      STX0     POPX                SAVE RETURN
     023664   021007 7000 00         8289 POP1     TSX0     DELV                DELETE BLOCK FROM WORKING STACK
     023665   035234 2200 00         8290          LDX0     A$WORK              GET POINTER TO END OF WORKING STACK
     023666   035214 1600 00         8291          SBX0     T$WORK              MAKE POINTER RELATIVE
   END OF BINARY CARD 00000410
     023667   017044 1000 00         8292          CMPX0    LSTMK               SEE IF ENOUGH BLOCKS HAVE BEEN DELETED
     023670   023664 6010 00         8293          TNZ      POP1                TRANSFER IF MORE TO DELETE
     023671   035235 2200 54         8294          LDX0     A$STACK,DI          GET OLD VALUE OF LAST MARK
     023672   017044 7400 00         8295          STX0     LSTMK               AND RESTORE IT
     023673   000000 7100 00         8296 POPX     TRA      **                  AND EXIT
     023674   023704 7400 00         8297 IDNTY    STX0     IDNTX               SAVE RETURN
     023675   022241 7000 00         8298          TSX0     IDENT               CREATE BLOCK AS FOR IDENTIFIER
     023676   000000 2250 16         8299          LDX5     0,6                 GET FLAGS FROM BLOCK
     023677   040000 3050 03         8300          CANX5    B$FD,DU             IS OFFSET,LL VALUE
     023700   777777 6010 00         8301          TNZ      $ERROR              YES - COMPILER ERROR
     023701   040000 2650 03         8302          ORX5     B$FD,DU             SET OFFSET,LL IS VALUE BIT
     023702   757777 3650 03         8303          ANX5     -1-B$FE,DU          RESET OFFSET,LL IS REFERENCE TO VALUE BIT
     023703   000000 7450 16         8304          STX5     0,6                 RESTORE FLAGS IN BLOCK
     023704   000000 7100 00         8305 IDNTX    TRA      **                  AND RETURN
     023705   023720 7400 00         8306 IDNTE    STX0     IDNX                SAVE RETURN
     023706   021027 7000 00         8307          TSX0     ASGNE               ASSIGN VALUE TO DECLARED VARIABLE
     023707   035234 2260 00         8308          LDX6     A$WORK              GET POINTER TO END OF WORK IN XR - 6
     023710   000004 1660 03         8309          SBX6     WL,DU               GET POINTER TO LAST BLOCK IN WORKING STACK
     023711   000002 2210 16         8310          LDX1     2,6                 GET ADDRESS WHERE VALUE WAS JUST STORED
     023712   000004 1660 03         8311 IDNT1    SBX6     WL,DU               STEP BACK TO NEXT BLOCK
     023713  000001 1010 16          8312          CMPX1    1,6                 WAS VALUE FOR THIS BLOCK JUST STORED
     023714  023712 6010 00          8313          TNZ      IDNT1               TRANSFER IF NOT TO KEEP LOOKING
END OF BINARY CARD 00000411
     023715  100000 2250 03          8314          LDX5     B$FC,DU             GET VALUE IS STACKED BIT
     023716  000000 2450 16          8315          ORSX5    0,6                 AND SET STACKED BIT IN BLOCK
     023717  021007 7000 00          8316          TSX0     DELV                DELETE VALUE OF ASSIGNATION FROM STACK
     023720  000000 7100 00          8317 IDNX     TRA      **                  AND RETURN
     023721  600000 2350 07          8318 ISNT     LDA      TZE,DL              GET TRANSFER IF FALSE COMMAND FOR ISNT
     023722  023724 7100 00          8319          TRA      ISS                 AND CONTINUE
     023723  601000 2350 07          8320 IS       LDA      TNZ,DL              GET TRANSFER IF FALSE COMMAND FOR IS
     023724  023755 5510 10          8321 ISS      STBA     IS2,10              STORE COMMAND IN INSTRUCTION SEQUENCE
     023725  023750 7400 00          8322          STX0     ISX                 SAVE RETURN
     023726  035234 2260 00          8323          LDX6     A$WORK              GET POINTER TO END OF WORKING STACK
     023727  000004 1660 03          8324          SBX6     WL,DU               GET POINTER TO LAST BLOCK IN WOROING STACK
     023730  020066 7000 00          8325          TSX0     MNA                 GET ADDRESS OF VALUE
     023731  023751 7510 71          8326          STCA     IS0,71              AND STORE IN INSTRUCTION SEQUENCE
     023732  000004 1660 03          8327          SBX6     WL,DU               GET POINTER TO SECOND TO TOP BLOCK
     023733  020066 7000 00          8328          TSX0     MNA                 GET ADDRESS OF VALUE
     023734  023753 7510 71          8329          STCA     IS1,71              AND STORE IN INSTRUCTION SEQUENCE
     023735  021007 7000 00          8330          TSX0     DELV                DELETE FIRST BLOCK
     023736  021007 7000 00          8331          TSX0     DELV                DELETE SECOND BLOCK
     023737  000003 6270 00          8332          EAX7     M$BOOL              GET MODE OF RESULT OF IDENTITY IN XR - 7
     023740  020377 7000 00          8333          TSX0     MBLK                AND MAKE A BLOCK FOR IT
     023741  017612 7000 00          8334          TSX0     GQ                  ALLOCATE Q REGISTER
     023742  004000 2250 03          8335          LDX5     B$FG,DU             GET VALUE IS IN REGISTER FLAG
END OF BINARY CARD 00000412
     023743  000000 7450 16          8336          STX5     0,6                 AND STORE IN BLOCK
     023744  756000 2360 07          8337          LDQ      STQ,DL              GET STORE REGISTER COMMAND
     023745  000003 7560 16          8338          STQ      3,6                 AND STORE IN BLOCK
     023746  023761 2350 00          8339          LDA      IST                 GET TALLY WORD FOR INSTRUCTION SEQUENCE
     023747  017507 7000 00          8340          TSX0     GADL                AND ADD INSTRUCTION SEQUENCE TO GENERATED OUTPUT
     023750  000000 7100 00          8341 ISX      TRA      **                  AND RETURN
     023751  000000 6200 00          8342 IS0      EAX0     0
     023752  000002 7400 04          8343          STX0     2,IC
     023753  000000 6200 00          8344 IS1      EAX0     0
     023754  000000 1000 03          8345          CMPX0    0,DU
     023755  000003 0000 04          8346 IS2      ARG      3,IC
     023756  000001 2360 07          8347          LDQ      1,DL
     023757  000002 7100 04          8348          TRA      2,IC
     023760  000000 2360 07          8349          LDQ      0,DL
     023761  023751 0011 00          8350 IST      TALLY    IS0,*-IS0+1
     023762  024000 7400 00          8351 CONF     STX0     CONFX               SAVE RETURN
     023763  017612 7000 00          8352          TSX0     GQ                  ALLOCATE Q REGISTER
     023764  035234 2260 00          8353          LDX6     A$WORK              GET POINTER TO END OF WORKING STACK
     023765  000004 1660 03          8354          SBX6     WL,DU               GET POINTER TO LAST BLOCK IN WORKING STACK
     023766  004000 2250 03          8355          LDX5     B$FG,DU             GET FLAG FOR VALUE IN REGISTER
     023767  000000 7450 16          8356          STX5     0,6                 AND STORE IN BLOCK
     023770  756000 2350 07          8357          LDA      STQ,DL              GET STORE REGISTER COMMAND
END OF BINARY CARD 00000413
     023771  000003 7550 16          8358          STA      3,6                 AND STORE IN BLOCK
```

```
      023772  017045 7270 00         8359          LXL7     PARAM            GET DESIRED MODE IN XR - 7
      023773  022103 7000 00         8360          TSX0     MTL              GET TYPE ADDRESS IN AU
      023774  000000 6200 01         8361          EAX0     0,AU             GET TYPE ADDRESS IN XR - 0
      023775  024001 7400 00         8362          STX0     CONF1            AND STORE IN INSTRUCTION SEQUENCE
      023776  024006 2350 00         8363          LDA      CONFT            GET TALLY WORD FOR INSTRUCTION SEQUENCE
      023777  017507 7000 00         8364          TSX0     GADL             AND ADD TO GENERATED CODE
      024000  000000 7100 00         8365 CONFX    TRA      **               AND RETURN
      024001  000000 1000 03         8366 CONF1    CMPX0    0,DL
      024002  000003 6010 04         8367          TNZ      3,IC
      024003  000001 2360 07         8368          LDQ      1,DL
      024004  000002 7100 04         8369          TRA      2,IC
      024005  000000 2360 07         8370          LDQ      0,DL
      024006  024001 0006 00         8371 CONFT    TALLY    CONF1,*-CONF1+1
      024007  024024 7400 00         8372 CONE     STX0     CONEX            SAVE RETURN
      024010  035234 2260 00         8373          LDX6     A$WORK           GET POINTER TO END OF WORKING STACK IN XR - 6
      024011  000004 1660 03         8374          SBX6     WL,DU            GET POINTER TO LAST BLOCK IN WORKING STACK
      024012  020204 7000 00         8375          TSX0     MVA              MAKE RHS VALUE AVAILABLE
      024013  777777 7100 00         8376          TRA      $ERROR           ERROR - UNITED VALUE CANNOT BE IN A REGISTER
      024014  024025 7510 71         8377          STCA     CONE0,71         STORE ADDRESS AND TAG IN INSTRUCTION SEQUENCE
      024015  000004 1660 03         8378          SBX6     WL,DU            GET POINTER TO REFERENCE BLOCK
      024016  000001 2200 16         8379          LDX0     1,6              GET ADDRESS OF REFERENCE WORD
  END OF BINARY CARD 00000414
      024017  024027 7400 00         8380          STX0     CONE1            AND STORE IN INSTRUCTION SEQUENCE
      024020  024031 2350 00         8381          LDA      CONET            GET TALLY WORD FOR INSTRUCTION SEQUENCE
      024021  017507 7000 00         8382          TSX0     GADL             AND ADD INSTRUCTION SEQUENCE TO GENERATED CODE
      024022  021007 7000 00         8383          TSX0     DELV             DELETE RHS BLOCK
      024023  021007 7000 00         8384          TSX0     DELV             DELETE REFERENCE BLOCK
      024024  000000 7100 00         8385 CONEX    TRA      **               AND RETURN
      024025  000000 6350 00         8386 CONE0    EAA      0
      024026  027240 0750 07         8387          ADA      TSPTRT,DL
      024027  000000 7550 17         8388 CONE1    STA      0,D
      024030  000000 7200 01         8389          LXL0     0,AU
      024031  024025 0005 00         8390 CONET    TALLY    CONE0,*-CONE0+1
      024032  024056 7400 00         8391 CASGN    STX0     CASX             SAVE RETURN
      024033  035234 2260 00         8392          LDX6     A$WORK           GET POINTER TO END OF WORKING STACK
      024034  000004 1660 03         8393          SBX6     WL,DU            GET POINTER TO LASK BLOCK IN WORKING STACK
      024035  000000 7270 16         8394          LXL7     0,6              GET MODE OF DESTINATION
      024036  010630 7000 00         8395          TSX0     A$XFER           MAKE MODE POINTER ABSOLUTE
      024037  000001 2270 17         8396          LDX7     1,7              GET MODE OF SOURCE IN XR - 7
      024040  020066 7000 00         8397          TSX0     MNA              MAKE NAME AVAILABLE FOR ASSIGNING TO
      024041  024057 7550 00         8398          STA      CAST             SAVE TO ADDRESS
      024042  035234 2260 00         8399          LDX6     A$WORK           GET POINTER TO END OF WORKING STACK
      024043  000010 1660 03         8400          SBX6     2*WL,DU          GET POINTER TO REFERENCE BLOCK
      024044  000001 2350 16         8401          LDA      1,6              GET ADDRESS ASSIGNED FOR REFERENCE
  END OF BINARY CARD 00000415
      024045  777777 3750 03         8402          ANA      -1,DU            ZERO OUT AL
      024046  220017 0750 07         8403          ADA      LDX+D,DL         CONSTRUCT LDX0 REFERENCE,D INSTRUCTION
      024047  017474 7000 00         8404          TSX0     GAD              AND ADD TO OUTPUT CODE
      024050  000001 6350 00         8405          EAA      1                GET RELATIVE ADDRESS OF VALUE OF VALUE OF UNION
      024051  000010 0750 07         8406          ADA      =O10,DL          ADD XR - 0 MODIFICATION
```

```
   024052  024057 2360 00      8407        LDQ      CAST           GET POINTER TO DESTINATION
   024053  017531 7000 00      8408        TSX0     MOVE           DO ASSIGNMENT
   024054  021007 7000 00      8409        TSX0     DELV           DELETE DESTINATION FROM STACK
   024055  021007 7000 00      8410        TSX0     DELV           DELETE REFERENCE BLOCK FROM STACK
   024056  000000 7100 00      8411 CASX   TRA      **             AND RETURN
   024057  000000 000000       8412 CAST   ZERO
   024060  024077 7400 00      8413 SKIP   STX0     SKIPX          SAVE RETURN
   024061  017045 7270 00      8414        LXL7     PARAM          GET MODE OF SKIP
   024062  020377 7000 00      8415        TSX0     MBLK           MAKE A NEW BLOCK IN WORKING STACK
   024063  100000 2250 03      8416        LDX5     B$FC,DU        GET STACKED VALUE BIT
   024064  000000 7450 16      8417        STX5     0,6            AND STORE IN SKIP VALUE BLOCK
   024065  000001 2350 16      8418        LDA      1,6            GET ASSIGNED STACK ADDRESS FOR VALUE
   024066  000000 6220 01      8419        EAX2     0,AU           SAVE IN XR - 2
   024067  017037 1020 00      8420        CMPX2    SP             SEE IF ANY WORDS TO TRANSFER
   024070  024077 6000 00      8421        TZE      SKIPX          TRANSFER IF NO WORDS TO TRANSFER
   024071  000000 6350 12      8422 SKIP1  EAA      0,2            GET ADDRESS OF NEXT WORD OF VALUE IN AU
   024072  450017 0750 07      8423        ADA      STZ+D,DL       GET STZ STACK,D INSTRUCTION IN A
 END OF BINARY CARD 00000416
   024073  017474 7000 00      8424        TSX0     GAD            AND ADD TO GENERATED CODE
   024074  000001 0620 03      8425        ADX2     1,DU           STEP TO NEXT WORD OF VALUE
   024075  017037 1020 00      8426        CMPX2    SP             SEE IF BEYOND VALUE
   024076  024071 6010 00      8427        TNZ      SKIP1          TRANSFER IF MORE WORDS TO STORE
   024077  000000 7100 00      8428 SKIPX  TRA      **             AND RETURN
                     024060    8429 NIL    EQU      SKIP           NIL IS A SINGLE ZERO WORD
   024100  024103 7400 00      8430 DEPR   STX0     DEPRX          SAVE RETURN
   024101  023516 7000 00      8431        TSX0     MSCW           MARK THE STACK FOR PROCEDURE ENTRY
   024102  023572 7000 00      8432        TSX0     ENTER          AND ENTER PROCEDURE
   024103  000000 7100 00      8433 DEPRX  TRA      **             AND RETURN
   024104  024150 7400 00      8434 UNION  STX0     UNX            SAVE RETURN
   024105  035234 2260 00      8435        LDX6     A$WORK         GET POINTER TO END OF WORKING STACK
   024106  000004 1660 03      8436        SBX6     WL,DU          GET POINTER TO LAST BLOCK IN WORK
   024107  000000 7270 16      8437        LXL7     0,6            GET MODE OF VALUE TO BE UNITED
   024110  010630 7000 00      8438        TSX0     A$XFER         MAKE MODE POINTER UNIQUE AND ABSOLUTE
   024111  777777 7200 17      8439        LXL0     -1,7           GET LENGTH OF VALUE
   024112  024153 7400 00      8440        STX0     UN1            AND STORE IN INSTRUCTION SEQUENCE
   024113  000000 7200 17      8441        LXL0     0,7            GET TYPE OF VALUE
   024114  024140 7400 00      8442        STX0     UNN3           SAVE POINTER TO TYPE TABLE ENTRY
   024115  020204 7000 00      8443        TSX0     MVA            MAKE VALUE TO BE UNITED AVAILABLE
   024116  024125 7100 00      8444        TRA      UNN1           TRANSFER IF VALUE IS IN A REGISTER
   024117  000001 2360 16      8445        LDQ      1,6            GET ADDRESS OF VALUE IN STACK IN QU
   024120  000001 6360 02      8446        EAQ      1,QU           GET NEW LOCATION OF VALUE IN QU
 END OF BINARY CARD 00000417
   024121  000017 0760 07      8447        ADQ      D,DL           ADD D REGISTER MODIFICATION
   024122  000000 7270 16      8448        LXL7     0,6            GET MODE OF VALUE TO BE MOVED
   024123  017517 7000 00      8449        TSX0     MOVEB          MOVE VALUE TO STACK LEAVING ROOM FOR HEADER
   024124  024133 7100 00      8450        TRA      UNN2           AND CONTINUE
   024125  024151 7550 00      8451 UNN1   STA      UNNT           SAVE STORE COMMAND
   024126  000001 2350 16      8452        LDA      1,6            GET ADDRESS IN STACK FOR VALUE
   024127  000001 6350 01      8453        EAA      1,AU           GET ADDRESS TO MOVE ADDRESS TO
   024130  000017 0750 07      8454        ADA      D,DL           ADD D REGISTER MODIFICATION
```

```
          3                                                   PASS 3

     024131   024151 0750 00          8455         ADA     UNNT                ADD STORE COMMAND
     024132   017474 7000 00          8456         TSX0    GAD                 AND ADD TO GENERATED CODE
     024133   021007 7000 00          8457 UNN2    TSX0    DELV                DELETE UNUNITED VALUE BLOCK FROM WORKING STACK
     024134   017045 7270 00          8458         LXL7    PARAM               GET MODE OF UNITED VALUE IN XR - 7
     024135   020377 7000 00          8459         TSX0    MBLK                AND MAKE A BLOCK FOR THE UNITED VALUE
     024136   000001 2200 16          8460         LDX0    1,6                 GET NEW VALUE ADDRESS IN XR - 0
     024137   024154 7400 00          8461         STX0    UN2                 AND STORE IN INSTRUCTION SEQUENCE
     024140   000000 2210 03          8462 UNN3    LDX1    **,DU               GET TYPE OF VALUE TO BE UNITED IN XR - 1
     024141   022110 7000 00          8463         TSX0    TL                  GET TYPE ADDRESS IN AU
     024142   000000 6200 01          8464         EAX0    0,AU                GET TYPE ADDRESS IN XR - 0
     024143   024152 7400 00          8465         STX0    UN0                 AND STORE IN INSTRUCTION SEQUENCE
     024144   024155 2350 00          8466         LDA     UNT                 GET TALLY WORD TO INSTRUCTION SEQUENCE
     024145   017507 7000 00          8467         TSX0    GADL                AND ADD SEQUENCE TO GENERATED CODE
     024146   100000 2250 03          8468         LDX5    B$FC,DU             GET VALUE IS STACKED FLAG
 END OF BINARY CARD 00000418
     024147   000000 7450 16          8469         STX5    0,6                 AND STORE IN BLOCK
     024150   000000 7100 00          8470 UNX     TRA     **                  AND RETURN
     024151   000000 000000           8471 UNNT    ZERO
     024152   000000 2350 07          8472 UN0     LDA     0,DL
     024153   000000 0750 03          8473 UN1     ADA     0,DU
     024154   000000 7550 17          8474 UN2     STA     0,D
     024155   024152 0004 00          8475 UNT     TALLY   UN0,*-UN0+1
     024156   000000 7100 10          8476 DISP    TRA     0,0                 DISPLAY HEADER IS TREATED AS A NOP
     024157   024211 7400 00          8477 EDISP   STX0    DISPX               SAVE RETURN
     024160   017045 7270 00          8478         LXL7    PARAM               GET MODE OF DISPLAYED VALUE
     024161   010630 7000 00          8479         TSX0    A$XFER              MAKE MODE UNIQUE AND ABSOLUTE
     024162   777777 2350 17          8480         LDA     -1,7                GET NUMBER OF FIELDS + 1 IN AU
     024163   000022 7710 00          8481         ARL     18                  MOVE NUMBER OF FIELDS TO AL
     024164   000001 1750 07          8482         SBA     1,DL                GET NUMBER OF FIELDE IN A
     024165   000000 2200 17          8483         LDX0    0,7                 GET TYPE OF MODE IN XR - 0
     024166   017001 1000 03          8484         CMPX0   M$PROC,DU           IS IT A PROCEDURE MODE
     024167   024171 6010 00          8485         TNZ     DISP0               TRANSFER IF NOT PROCEDURE
     024170   000001 1750 07          8486         SBA     1,DL                DECREMENT COUNT SO NOT TO INCLUDE RESULT
     024171   000000 5310 00          8487 DISP0   NEG                         GET MINUS NUMBER OF FIELDS IN A
     024172   024212 7550 00          8488         STA     DISPC               AND STORE COUNT IN MEMORY
     024173   035234 2260 00          8489         LDX6    A$WORK              GET POINTER TO THE END OF THE WORKING STACK
     024174   000004 1660 03          8490 DISP1   SBX6    WL,DU               GET POINTER TO NEXT BLOCK
 END OF BINARY CARD 00000419
     024175   020451 7000 00          8491         TSX0    FS                  FORCE VALUE TO THE STACK
     024176   024212 0540 00          8492         AOS     DISPC               DECREMENT FIELD COUNT
     024177   024174 6010 00          8493         TNZ     DISP1               TRANSFER IF MORE FIELDS TO FORCE TO STACK
     024200   000001 2200 16          8494         LDX0    1,6                 GET LOCATION OF FIRST FIELD
     024201   017037 7400 00          8495         STX0    SP                  AND STORE IN STACK POINTER
     024202   035234 0110 54          8496 DISP2   NOP     A$WORK,DI           DELETE A WORD FROM THE WORKING STACK
     024203   035234 1060 00          8497         CMPX6   A$WORK              CHECK TO SEE IF ENOUGH HAS BEEN DELETED
     024204   024202 6010 00          8498         TNZ     DISP2               TRANSFER IF MORE TO DELETE
     024205   017045 7270 00          8499         LXL7    PARAM               GET MODE OF DISPLAYED VALUE
     024206   020377 7000 00          8500         TSX0    MBLK                AND MAKE A BLOCK FOR DISPLAYED VALUE
     024207   100000 2250 03          8501         LDX5    B$FC,DU             GET A STACKED BIT
     024210   000000 7450 16          8502         STX5    0,6                 AND STORE AS FLAGS FOR BLOCK
```

```
      024211  000000 7100 00        8503 DISPX   TRA      **             AND EXIT
      024212  000000 000000         8504 DISPC   ZERO
      024213  024221 7100 00        8505 VLWB    TRA      V              LOWER BOUND IN TRIMMER
      024214  024221 7100 00        8506 VUPB    TRA      V              UPPER BOUND IN TRIMMER
      024215  024221 7100 00        8507 VNLWB   TRA      V              NEW LOWER BOUND IN TRIMMER
      024216  024221 7100 00        8508 VSBCT   TRA      V              SUBSCRIPT
      024217  024230 7400 00        8509 VEPTY   STX0     VX             SAVE RETURN
      024220  024225 7100 00        8510         TRA      V1             GO STORE EMPTY TRIMMER FLAG
      024221  024230 7400 00        8511 V       STX0     VX             SAVE RETURN
      024222  035234 2260 00        8512         LDX6     A$WORK         GET POINTER TO END OF WORKING STACK
  END OF BINARY CARD 00000420
      024223  000004 1660 03        8513         SBX6     WL,DU          GET POINTER TO LAST BLOCK IN WORKING STACK
      024224  020451 7000 00        8514         TSX0     FS             FORCE VALUE TO STACK
      024225  017045 2350 00        8515 V1      LDA      PARAM          GET COMMAND FROM PASS 2
      024226  035235 7550 56        8516         STA      A$STACK,ID     AND STORE IN CONTROL STACK
      024227  005742 7170 00        8517         XED      T$SOVF         CHECK FOR STACK OVERFLOW
      024230  000000 7100 00        8518 VX      TRA      **             AND RETURN
      024231  024243 7400 00        8519 SUB     STX0     SUBX           SAVE RETURN
      024232  024533 2350 00        8520         LDA      VMARK          GET OLD SLICE MARK
      024233  035235 7550 56        8521         STA      A$STACK,ID     AND STORE IN CONTROL STACK
      024234  005742 7170 00        8522         XED      T$SOVF         CHECK FOR STACK OVERFLOW
      024235  035235 2200 00        8523         LDX0     A$STACK        GET POINTER TO END OF CONTROL STACK
      024236  035215 1600 00        8524         SBX0     T$STACK        MAKE POINTER RELATIVE
      024237  024533 7400 00        8525         STX0     VMARK          AND STORE IN SLICE MARK
      024240  035234 2200 00        8526         LDX0     A$WORK         GET POINTER TO END OF WORKING STACK
      024241  035214 1600 00        8527         SBX0     T$WORK         MAKE POINTER RELATIVE
      024242  024533 4400 00        8528         SXL0     VMARK          AND STORE IN SLICE MARK
      024243  000000 7100 00        8529 SUBX    TRA      **             AND EXIT
                        024244      8530 RBUS    EQU      *
      024244  024475 7400 00        8531 BUS     STX0     BUSX           SAVE RETURN
      024245  024533 7260 00        8532         LXL6     VMARK          GET POINTER TO WORKING STACK WHEN MARKED
      024246  035214 0660 00        8533         ADX6     T$WORK         MAKE POINTER ABSOLUTE
      024247  000004 1660 03        8534         SBX6     WL,DU          GET POINTER TO ARRAY DESCRIPTOR BLOCK
      024250  020204 7000 00        8535         TSX0     MVA            MAKE DESCRIPTOR VALUE AVAILABLE
  END OF BINARY CARD 00000421
      024251  777777 7100 00        8536         TRA      $ERROR         COMPILER ERROR - CANNOT BE IN ACCUMULATOR
      024252  024476 7510 71        8537         STCA     BUSI,71        STORE IN INSTRUCTION SEQUENCE
      024253  024500 7510 71        8538         STCA     BUI,71         STORE IN INSTRUCTION SEQUENCE
      024254  000001 2210 16        8539         LDX1     1,6            GET POINTER TO FLAG IN DESCRIPTOR
      024255  024534 7410 00        8540         STX1     BUSC           AND SAVE
      024256  000001 0610 03        8541         ADX1     1,DU           GET POINTER TO FIRST QUAD
      024257  024503 7410 00        8542         STX1     BUI1           STORE IN INSTRUCTION SEQUENCE
      024260  024510 7410 00        8543         STX1     BUSI3          STORE IN INSTRUCTION SEQUENCE
      024261  000001 0610 03        8544         ADX1     1,DU           GET POINTER TO FIRST QUAD
      024262  024536 7410 00        8545         STX1     NFF            INITIALIZE NEW QUAD POINTER
      024263  000001 2210 03        8546         LDX1     1,DU           GET RELATIVE OLD QUAD POINTER

      024264  024535 7410 00        8547         STX1     OFF            AND INITIALIZE OLD QUAD POINTER
      024265  024533 2350 00        8548         LDA      VMARK          GET CONTROL AND WORKING STACK MARKS
      024266  024537 7550 00        8549         STA      VM1            AND SAVE
      024267  035235 4500 56        8550         STZ      A$STACK,ID     MARK END OF CONTROL STACK WITH A ZERO
```

```
           3                                              PASS 3

     024270  005742 7170 00        8551         XED     T$SOVF          CHECK FOR STACK OVERFLOW
     024271  024504 2350 00        8552         LDA     BUSTO           GET INITIAL SLICE CODE TALLY WORD
     024272  017507 7000 00        8553         TSX0    GADL            AND ADD INDICATED CODE TO OUTPUT
     024273  024540 4500 00        8554 BUS1    STZ     BUSF            RESET SUBSCRIPT FLAG
     024274  024541 4500 00        8555         STZ     BUSE            RESET EMPTY FLAG
     024275  024542 4500 00        8556         STZ     BLWB            RESET LOWER BOUND
     024276  024543 4500 00        8557         STZ     BUPB            RESET UPPER BOUND
  END OF BINARY CARD 00000422
     024277  024544 4500 00        8558         STZ     BNLWB           RESET NEW LOWER BOUND
     024300  024537 7260 00        8559         LXL6    VM1             GET CURRENT PLACE IN WORKING STACK
     024301  035214 0660 00        8560         ADX6    T$WORK          MAKE POINTER ABSOLUTE
     024302  024537 2210 00        8561         LDX1    VM1             GET CURRENT PLACE IN CONTROL STACK
     024303  035215 0610 00        8562         ADX1    T$STACK         MAKE POINTER ABSOLUTE
     024304  000000 2200 11        8563 BUS2    LDX0    0,1             GET TYPE OF NEXT ENTRY FROM CONTROL STACK
     024305  024452 6000 00        8564         TZE     BU18            TRANSFER IF NO MORE IN CONTROL STACK
     024306  016713 1000 03        8565         CMPX0   0$VEPTY,DU      IS IT AN EMPTY TYPE
     024307  024313 6010 00        8566         TNZ     BUS3            TTRANSFER IF NOT
     024310  024313 6010 00        8567         TNZ     BUS3            TRANSFER IF NOT
     024311  024541 7500 00        8568         STC2    BUSE            SET EMPTY FLAG
     024312  024340 7100 00        8569         TRA     BUS8            AND GO TO END CHECK
     024313  016677 1000 03        8570 BUS3    CMPX0   0$VSBCT,DU      IS IT A SUBSCRIPT
     024314  024320 6010 00        8571         TNZ     BUS4            TRANSFER IF NOT
     024315  024540 7500 00        8572         STC2    BUSF            SET SUBSCRIPT FLAG
     024316  024542 6200 00        8573         EAX0    BLWB            GET POINTER TO LOWER BOUND STORAGE
     024317  024334 7100 00        8574         TRA     BUS7            AND GO STORE SUBSCRIPT
     024320  016702 1000 03        8575 BUS4    CMPX0   0$VLWB,DU       IS IT A LOWER BOUND
     024321  024324 6010 00        8576         TNZ     BUS5            TRANSFER IF NOT
     024322  024542 6200 00        8577         EAX0    BLWB            GET POINTER TO LOWER BOUND STORAGE
     024323  024334 7100 00        8578         TRA     BUS7            AND GO STORE LOWER BOUND
     024324  016705 1000 03        8579 BUS5    CMPX0   0$VUPB,DU       IS IT AN UPPER BOUND
  END OF BINARY CARD 00000423
     024325  024330 6010 00        8580         TNZ     BUS6            TRANSFER IF NOT
     024326  024543 6200 00        8581         EAX0    BUPB            GET POINTER TO UPPER BOUND STORAGE
     024327  024334 7100 00        8582         TRA     BUS7            AND GO STORE UPPER BOUND
     024330  016710 1000 03        8583 BUS6    CMPX0   0$VNLWB,DU      IS IT A NEW LOWER BOUND
     024331  777777 6010 00        8584         TNZ     $ERROR          NO - COMPILER ERROR
     024332  024544 6200 00        8585         EAX0    BNLWB           GET POINTER TO NEW LOWER BOUND STORAGE
     024333  024334 7100 00        8586         TRA     BUS7            AND GO STORE NEW LOWER BOUND
     024334  000001 2350 16        8587 BUS7    LDA     1,6             GET ADDRESS OF QUANTITY
     024335  000000 6350 01        8588         EAA     0,AU            ZERO OUT AL
     024336  000000 7550 10        8589         STA     0,0             AND STORE IN INDICATED STORAGE LOCATION
     024337  000004 0660 03        8590         ADX6    WL,DU           STEP TO NEXT BLOCK IN WORKING STACK
     024340  000001 0610 03        8591 BUS8    ADX1    1,DU            STEP TO NEXT WORD IN CONTROL STACK
     024341  777777 2360 03        8592         LDQ     -1,DU           MASK OUT AU
     024342  777777 2350 11        8593         LDA     -1,1            GET WORD JUST PROCESSED
     024343  000000 2110 11        8594         CMK     0,1             IS NEXT WORD IN SAME SUBSCRIPT POSITION
     024344  024304 6000 00        8595         TZE     BUS2            YES - TRANSFER TO PICK IT UP
     024345  035215 1610 00        8596         SBX1    T$STACK         MAKE CONTROL STACK POINTER RELATIVE
     024346  024537 7410 00        8597         STX1    VM1             AND SAVE
     024347  035214 1660 00        8598         SBX6    T$WORK          MAKE WORKING STACK POINTER RELATIVE
```

```
              3                                              PASS 3

      024350  024537 4460 00        8599          SXL6    VM1                 AND SAVE
      024351  024535 2350 00        8600          LDA     OFF                 GET POINTER TO LOWER BOUND
      024352  024506 7510 70        8601          STCA    BUSI1,70            STORE IN INSTRUCTION SEQUENCE
   END OF BINARY CARD 00000424
      024353  024513 7510 71        8602          STCA    BUSI5,71            STORE IN INSTRUCTION SEQUENCE
      024354  024521 7510 70        8603          STCA    BUS10,70            STORE IN INSTRUCTION SEQUENCE
      024355  000001 0750 03        8604          ADA     1,DU                GET POINTER TO UPPER BOUND
      024356  024512 7510 71        8605          STCA    BUSI4,71            STORE IN INSTRUCTION SEQUENCE
      024357  024523 7510 70        8606          STCA    BUS12,70            STORE IN INSTRUCTION SEQUENCE
      024360  000001 0750 03        8607          ADA     1,DU                GET POINTER TO STRIDE
      024361  024507 7510 70        8608          STCA    BUSI2,70            STORE IN INSTRUCTION SEQUENCE
      024362  024526 7510 70        8609          STCA    BUS14,70            STORE IN INSTRUCTION SEQUENCE
      024363  000001 0750 03        8610          ADA     1,DU                GET POINTER TO STATES
      024364  024530 7510 70        8611          STCA    BUS16,70            STORE IN INSTRUCTION SEQUENCE
      024365  024536 2350 00        8612          LDA     NFF                 GET POINTER TO CALCULATED LOWER BOUND
      024366  024517 7510 70        8613          STCA    BUSI9,70            STORE IN INSTRUCTION SEQUENCE
      024367  024522 7510 70        8614          STCA    BUS11,70            STORE IN INSTRUCTION SEQUENCE
      024370  000001 0750 03        8615          ADA     1,DU                GET POINTER TO UPPER BOUND
      024371  024515 7510 70        8616          STCA    BUSI7,70            STORE IN INSTRUCTION SEQUENCE
      024372  024524 7510 70        8617          STCA    BUS13,70            STORE IN INSTRUCTION SEQUENCE
      024373  000001 0750 03        8618          ADA     1,DU                GET POINTER TO STRIDE
      024374  024527 7510 70        8619          STCA    BUS15,70            STORE IN INSTRUCTION SEQUENCE
      024375  000001 0750 03        8620          ADA     1,DU                GET POINTER TO STATES
      024376  024531 7510 70        8621          STCA    BUS17,70            STORE IN INSTRUCTION SEQUENCE
      024377  024542 2350 00        8622          LDA     BLWB                GET LOWER BOUND
      024400  024404 6000 00        8623          TZE     BUS9                TRANSFER IF NO LOWER BOUND
   END OF BINARY CARD 00000425
      024401  000017 0750 07        8624          ADA     D,DL                ADD D REGISTER MODIFICATION
      024402  024505 7510 70        8625          STCA    BUSI0,70            STORE IN INSTRUCTION SEQUENCE
      024403  024513 7510 71        8626          STCA    BUSI5,71            STORE IN INSTRUCTION SEQUENCE
      024404  024543 2350 00        8627 BUS9     LDA     BUPB                GET UPPER BOUND
      024405  024410 6000 00        8628          TZE     BU10                TRANSFER IF NO UPPER BOUND
      024406  000017 0750 07        8629          ADA     D,DL                ADD D REGISTER MODIFICATION
      024407  024512 7510 71        8630          STCA    BUSI4,71            STORE IN INSTRUCTION SEQUENCE
      024410  024544 2350 00        8631 BU10     LDA     BNLWB               GET NEW LOWER BOUND
      024411  024414 6000 00        8632          TZE     BU11                TRANSFER IF NO NEW LOWER BOUND
      024412  000017 0750 07        8633          ADA     D,DL                ADD D REGISTER MODIFICATION
      024413  024416 7100 00        8634          TRA     BU12                GO TO STORE ADDRESS
      024414  000001 2350 03        8635 BU11     LDA     1,DU                GET A ONE
      024415  000007 0750 07        8636          ADA     DL,DL               MAKE ADDRESS 1,DL
      024416  024514 7510 71        8637 BU12     STCA    BUSI6,71            STORE IN INSTRUCTION SEQUENCE
      024417  024516 7510 71        8638          STCA    BUSI8,71            STORE IN INSTRUCTION SEQUENCE
      024420  024541 2340 00        8639          SZN     BUSE                SEE IF EMPTY POSITION
      024421  024430 6000 00        8640          TZE     BU13                TRANSFER IF NOT EMPTY
      024422  024535 2350 00        8641          LDA     OFF                 GET OLD QUAD ADDRESS
      024423  024536 1150 00        8642          CMPA    NFF                 COMPARE WITH NEW QUAD ADDRESS
      024424  024445 6000 00        8643          TZE     BU16                TRANSFER IF SAME - NO MOVING
      024425  024525 2350 00        8644          LDA     BUST3               GET TALLY WORD FOR MOVE INSTRUCTION SEQUENCE
      024426  017507 7000 00        8645          TSX0    GADL                AND ADD TO OUTPUT CODE
   END OF BINARY CARD 00000426
```

```
          3                                              PASS 3

     024427  024443 7100 00       8646         TRA      BU15          GO TO MOVE REST OF QUAD
     024430  024542 2340 00       8647 BU13    SZN      BLWB          SEE IF THERE WAS A LOWER BOUND
     024431  024434 6000 00       8648         TZE      BU14          TRANSFER IF NO LOWER BOUND
     024432  024511 2350 00       8649         LDA      BUST1         GET TALLY WORD FOR LOWER BOUND SEQUENCE
     024433  017507 7000 00       8650         TSX0     GADL          AND ADD TO GENERATED OUTPUT
     024434  024540 2340 00       8651 BU14    SZN      BUSF          CHECK IF SUBSCRIPT OR TRIMMER
     024435  024447 6010 00       8652         TNZ      BU17          TRANSFER IF SUBSCRIPT
     024436  024520 2350 00       8653         LDA      BUST2         GET TALLY WORD FOR SLICE SEQUENCE
     024437  017507 7000 00       8654         TSX0     GADL          AND ADD TO GENERATED OUTPUT
     024440  024535 2350 00       8655         LDA      OFF           GET OLD QUAD ADDRESS
     024441  024536 1150 00       8656         CMPA     NFF           COMPARE WITH NEW QUAD ADDRESS
     024442  024445 6000 00       8657         TZE      BU16          TRANSFER IF SAME - NO MOVING
     024443  024532 2350 00       8658 BU15    LDA      BUST4         GET TALLY WORD FOR MOVE SEQUENCE
     024444  017507 7000 00       8659         TSX0     GADL          AND ADD TO GENERATED OUTPUT
     024445  000004 2200 03       8660 BU16    LDX0     4,DU          GET LENGTH OF QUAD IN XR - 0
     024446  024536 0400 00       8661         ASX0     NFF           INCREMENT NEW QUAD ADDRESS
     024447  000004 2200 03       8662 BU17    LDX0     4,DU          GET LENGTH OF QUAD IN XR - 0
     024450  024535 0400 00       8663         ASX0     OFF           INCREMENT OLD QUAD ADDRESS
     024451  024273 7100 00       8664         TRA      BUS1          AND LOOP
     024452  021007 7000 00       8665 BU18    TSX0     DELV          DELETE A BLOCK FROM THE WORKING STACK
     024453  024533 7210 00       8666         LXL1     VMARK         GET POINTER TO WORKING STACK WHEN MARKED
     024454  035214 0610 00       8667         ADX1     TSWORK        MAKE POINTER ABSOLUTE
 END OF BINARY CARD 00000427
     024455  035234 1010 00       8668         CMPX1    ASWORK        HAVE WE DELETED BACK TO THE MARK
     024456  024452 6010 00       8669         TNZ      BU18          TRANSFER IF MORE TO DELETE
     024457  024533 2210 00       8670         LDX1     VMARK         GET POINTER TO CONTROL STACK WHEN MARKED
     024460  035215 0610 00       8671         ADX1     TSSTACK       MAKE POINTER ABSOLUTE
     024461  035235 0110 54       8672 BU19    NOP      ASSTACK,DI    DELETE A WORD FROM THE CONTROL STACK
     024462  035235 1010 00       8673         CMPX1    ASSTACK       HAVE WE DELETED BACK TO THE MARK
     024463  024461 6010 00       8674         TNZ      BU19          TRANSFER IF MORE TO DELETE
     024464  035235 2350 54       8675         LDA      ASSTACK,DI    GET SAVED PREVIOUS MARK
     024465  024533 7550 00       8676         STA      VMARK         AND RESTORE IT
     024466  035234 2260 00       8677         LDX6     ASWORK        GET POINTER TO END OF WORKING STACK
     024467  000004 1660 03       8678         SBX6     WL,DU         GET POINTER TO LAST BLOCK IN WORKING STACK
     024470  017045 7270 00       8679         LXL7     PARAM         GET MODE OF RESULT OF SLICE
     024471  000000 4470 16       8680         SXL7     0,6           AND STORE IN BLOCK
     024472  000001 2200 16       8681         LDX0     1,6           GET OLD STACK POINTER
     024473  017037 7400 00       8682         STX0     SP            AND RETURN DESCRIPTOR MEMORY
     024474  020354 7000 00       8683         TSX0     GTMP          REALLOCATE FOR NEW DESCRIPTOR
     024475  000000 7100 00       8684 BUSX    TRA      **            AND RETURN
     024476  000000 2200 17       8685 BUSI    LDX0     0,D
     024477  000002 6010 04       8686         TNZ      2,IC
     024500  000000 6200 17       8687 BUI     EAX0     0,D
     024501  000001 0600 03       8688         ADX0     1,DU
     024502  000000 2350 10       8689         LDA      0,0
 END OF BINARY CARD 00000428
     024503  000000 7550 17       8690 BUI1    STA      0,D
     024504  024476 0007 00       8691 BUSTO   TALLY    BUSI,**-BUSI+1
     024505  000000 2360 17       8692 BUSIO   LDQ      0,D
     024506  000000 1760 10       8693 BUSI1   SBQ      0,0
     024507  000000 4020 10        8694 BUSI2   MPY      0,0
     024510  000000 0560 17        8695 BUSI3   ASQ      0,D
     024511  024505 0005 00        8696 BUST1   TALLY    BUSI0,*-BUSI0+1
     024512  000000 2360 17        8697 BUSI4   LDQ      0,D
     024513  000000 1760 17        8698 BUSI5   SBQ      0,D
     024514  000000 0760 00        8699 BUSI6   ADQ      0
     024515  000000 7560 17        8700 BUSI7   STQ      0,D
     024516  000000 2360 00        8701 BUSI8   LDQ      0
     024517  000000 7560 17        8702 BUSI9   STQ      0,D
     024520  024512 0007 00        8703 BUST2   TALLY    BUSI4,*-BUSI4+1
     024521  000000 2360 10        8704 BUS10   LDQ      0,0
     024522  000000 7560 17        8705 BUS11   STQ      0,D
     024523  000000 2360 10        8706 BUS12   LDQ      0,0
     024524  000000 7560 17        8707 BUS13   STQ      0,D
     024525  024521 0005 00        8708 BUST3   TALLY    BUS10,*-BUS10+1
     024526  000000 2360 10        8709 BUS14   LDQ      0,0
     024527  000000 7560 17        8710 BUS15   STQ      0,D
     024530  000000 2360 10        8711 BUS16   LDQ      0,0
   END OF BINARY CARD 00000429
     024531  000000 7560 17        8712 BUS17   STQ      0,D
     024532  024526 0005 00        8713 BUST4   TALLY    BUS14,*-BUS14+1
     024533  000000 000000         8714 VMARK   ZERO
     024534  000000 000000         8715 BUSC    ZERO
     024535  000000 0000 10        8716 OFF     ARG      0,0
     024536  000000 0000 17        8717 NFF     ARG      0,D
     024537  000000 000000         8718 VM1     ZERO
     024540  000000 000000         8719 BUSF    ZERO
     024541  000000 000000         8720 BUSE    ZERO
     024542  000000 000000         8721 BLWB    ZERO
     024543  000000 000000         8722 BUPB    ZERO
     024544  000000 000000         8723 BNLWB   ZERO
     024545  024562 7400 00        8724 DSUB    STX0     DSUBX              SAVE RETURN
     024546  000037 6270 00        8725         EAX7     MSPTR              GET MODE OF POINTER IN XR - 7
     024547  020377 7000 00        8726         TSX0     MBLK               AND MAKE A BLOCK FOR IT
     024550  100000 2250 03        8727         LDX5     BSFC,DU            GET VALUE IS STACKED BIT
     024551  000000 7450 16        8728         STX5     0,6                AND STORE IN BLOCK
     024552  000007 6270 00        8729         EAX7     MSINT              GET INTEGER MODE
     024553  020377 7000 00        8730         TSX0     MBLK               AND MAKE A BLOCK FOR TEMPORARY
     024554  000007 6270 00        8731         EAX7     MSINT              GET INTEGER MODE
     024555  020377 7000 00        8732         TSX0     MBLK               AND MAKE A BLOCK FOR TEMPORARY
     024556  021007 7000 00        8733         TSX0     DELV               DELETE A BLOCK
   END OF BINARY CARD 00000430
     024557  021007 7000 00        8734         TSX0     DELV               DELETE A BLOCK
     024560  024565 2350 00        8735         LDA      DSUBT              GET TALLY WORD FOR INSTRUCTION SEQUENCE
     024561  017507 7000 00        8736         TSX0     GADL               AND ADD TO INSTRUCTION SEQUENCE
     024562  000000 7100 00        8737 DSUBX   TRA      **                 AND RETURN
     024563  000004 2360 11        8738 DSUB0   LDQ      4,1
     024564  000004 7560 12        8739         STQ      4,2
     024565  024563 0003 00        8740 DSUBT   TALLY    DSUB0,*-DSUB0+1
     024566  740000 2350 07        8741 LWB     LDA      STX,DL             GET STORE LOWER STATE INSTRUCTION
```

```
          3                                                   PASS 3

     024567  024625 7550 00       8742         STA     FFI              AND STORE
     024570  017045 2360 00       8743         LDQ     PARAM            GET CURRENT POSITION IN QL
     024571  000024 7360 00       8744         QLS     18*2             MOVE TO QU AND MULTIPLY BY 4
     024572  000003 1760 03       8745         SBQ     3,DU             GET ADDRESS OF LOWER BOUND IN QU
     024573  024626 7560 00       8746         STQ     DBT              AND STORE
     024574  024603 7100 00       8747         TRA     DB1              GO CONTINUE
     024575  440000 2350 07       8748 UPB     LDA     SXL,DL           GET STORE UPPER STATE INSTRUCTION
     024576  024625 7550 00       8749         STA     FFI              AND STORE
     024577  017045 2360 00       8750         LDQ     PARAM            GET CURRENT POSITION IN QL
     024600  000024 7360 00       8751         QLS     18*2             MOVE TO QU AND MULTIPLY BY 4
     024601  000002 1760 03       8752         SBQ     2,DU             GET ADDRESS OF UPPER BOUND IN QU
     024602  024626 7560 00       8753         STQ     DBT              AND STORE
     024603  024623 7400 00       8754 DB1     STX0    DBX              SAVE RETURN
     024604  035234 2260 00       8755         LDX6    A$WORK           GET POINTER TO END OF WORKING STACK
 END OF BINARY CARD 00000431
     024605  000010 1660 03       8756         SBX6    2*WL,DU          GET POINTER TO FORMAL PARAMETER BLOCK
     024606  020066 7000 00       8757         TSX0    MNA              MAKE POINTER AVAILABLE
     024607  024624 7550 00       8758         STA     FFA              AND STORE POINTER TO DESCRIPTOR
     024610  024626 0550 00       8759         ASA     DBT              AND ADD TO BOUND ADDRESS
     024611  000004 0660 03       8760         ADX6    WL,DU            GET POINTER TO BOUND BLOCK
     024612  020204 7000 00       8761         TSX0    MVA              MAKE BOUND AVAILABLE
     024613  024620 7100 00       8762         TRA     DB2              TRANSFER IF BOUND IN REGISTER
     024614  024626 2360 00       8763         LDQ     DBT              GET ADDRESS WHERE BOUND IS TO BE STORED
     024615  000000 7270 16       8764         LXL7    0,6              GET MODE OF BOUND IN XR - 7
     024616  017531 7000 00       8765         TSX0    MOVE             MOVE BOUND TO DESCRIPTOR
     024617  024622 7100 00       8766         TRA     DB3              AND CONTINUE
     024620  024626 0750 00       8767 DB2     ADA     DBT              ADD ADDRESS OF BOUND IN DESCRIPTOR
     024621  017474 7000 00       8768         TSX0    GAD              AND ADD STORE INSTRUCTION TO GENERATED OUTPUT
     024622  021007 7000 00       8769 DB3     TSX0    DELV             DELETE BOUND BLOCK
     024623  000000 7100 00       8770 DBX     TRA     **               AND EXIT
     024624  000000 000000        8771 FFA     ZERO
     024625  000000 000000        8772 FFI     ZERO
     024626  000000 000000        8773 DBT     ZERO
     024627  220003 2350 07       8774 FLEX    LDA     LDX+DU,DL        GET LDX INSTRUCTION
     024630  000001 0750 03       8775         ADA     1,DU             GET LDX0 1,DU INSTRUCTION
     024631  024633 7100 00       8776         TRA     FF1              GO CONTINUE
     024632  220003 2350 07       8777 FIX     LDA     LDX+DU,DL        GET LDX 0,DU INSTRUCTION
 END OF BINARY CARD 00000432
     024633  024642 7400 00       8778 FF1     STX0    FFX              SAVE RETURN
     024634  017474 7000 00       8779         TSX0    GAD              AND ADD TO GENERATED OUTPU
     024635  017045 2350 00       8780         LDA     PARAM            GET POSITION IN AL
     024636  000024 7350 00       8781         ALS     18*2             MOVE TO AU AND MULTIPLY BY 4
     024637  024624 0750 00       8782         ADA     FFA              ADD POINTER TO DESCRIPTOR
     024640  024625 0750 00       8783         ADA     FFI              ADD STORE STATE INSTRUCTION
     024641  017474 7000 00       8784         TSX0    GAD              ADD STORE STATE INSTRUCTION TO GENERATED OUTPUT
     024642  000000 7100 00       8785 FFX     TRA     **               AND EXIT
     024643  024673 7400 00       8786 DBUS    STX0    DBUSX            SAVE RETURN
     024644  017045 7210 00       8787         LXL1    PARAM            GET POINTER TO RANGE OF BOX
     024645  035221 0610 00       8788         ADX1    T$PROG           MAKE POINTER ABSOLUTE
     024646  000005 7270 11       8789         LXL7    5,1              GET MODE OF ELEMENT IN XR - 7
     024647  010630 7000 00        8790         TSX0    A$XFER          MAKE MODE POINTER ABSOLUTE
     024650  777777 2350 17        8791         LDA     -1,7            GET LENGTH OF ELEMENT VALUE IN AL
     024651  000022 7350 00        8792         ALS     18              MOVE LENGTH TO AU
     024652  024674 7510 70        8793         STCA    DBS0,70         AND STORE IN INSTRUCTION SEQUENCE
     024653  024677 2350 00        8794         LDA     DBST            GET TALLY WORD FOR INSTRUCTION SEQUENCE
     024654  017507 7000 00        8795         TSX0    GADL            AND ADD SEQUENCE TO GENERATED CODE
     024655  024626 2350 00        8796 DBUSL   LDA     DBT             GET POINTER TO LAST UPPER BOUND
     024656  024701 7510 71        8797         STCA    DBUS1,71        STORE ADDRESS IN INSTRUCTION SEQUENCE
     024657  000001 1750 03        8798         SBA     1,DU            GET POINTER TO LOWER BOUND
     024660  024702 7510 71        8799         STCA    DBUS2,71        STORE ADDRESS IN INSTRUCTION SEQUENCE
 END OF BINARY CARD 00000433
     024661  000002 0750 03        8800         ADA     2,DU            GET POINTER TO STRIDE
     024662  024700 7510 71        8801         STCA    DBUS0,71        STORE ADDRESS IN INSTRUCTION SEQUENCE
     024663  024705 7510 71        8802         STCA    DBUS3,71        STORE ADDRESS IN INSTRUCTION SEQUENCE
     024664  024707 2350 00        8803         LDA     DBUST           GET TALLY WORD FOR INSTRUCTION SEQUENCE
     024665  017507 7000 00        8804         TSX0    GADL            AND ADD TO GENERATED OUTPUT
     024666  777774 2350 03        8805         LDA     -4,DU           GET LENGTH OF QUAD IN AU

     024667  024626 0550 00        8806         ASA     DBT             AND STEP QUAD POINTER BACK ONE QUAD
     024670  024655 6050 00        8807         TPL     DBUSL           LOOP IF MORE QUADS
     024671  024717 2350 00        8808         LDA     DBST1           GET TALLY WORD FOR SECOND INSTRUCTION SEQUENCE
     024672  017507 7000 00        8809         TSX0    GADL            AND ADD SEQUENCE TO GENERATED OUTPUT
     024673  000000 7100 00        8810 DBUSX   TRA     **              AND EXIT
                        000005     8811 DBT1    EQU     5
                        000006     8812 DBT2    EQU     6
     024674  000000 6360 00        8813 DBS0    EAQ     0
     024675  000005 4500 17        8814         STZ     DBT1,D
     024676  000006 4500 17        8815         STZ     DBT2,D
     024677  024674 0004 00        8816 DBST    TALLY   DBS0,*-DBS0+1
     024700  000000 7560 00        8817 DBUS0   STQ     0
     024701  000000 2360 00        8818 DBUS1   LDQ     0
     024702  000000 1760 00        8819 DBUS2   SBQ     0
     024703  000001 0760 07        8820         ADQ     1,DL
     024704  000005 2560 17        8821         ORSQ    DBT1,D
     024705  000000 4020 00        8822 DBUS3   MPY     0
     024706  000006 2550 17        8823         ORSA    DBT2,D
 END OF BINARY CARD 00000434
     024707  024700 0010 00        8824 DBUST   TALLY   DBUS0,*-DBUS0+1
     024710  000004 7560 17        8825 DBUS4   STQ     4,D
     024711  000005 2340 17        8826         SZN     DBT1,D
     024712  000003 6050 04        8827         TPL     3,IC
     024713  000004 4500 17        8828         STZ     4,D
     024714  000003 7100 04        8829         TRA     3,IC
     024715  000006 2340 17        8830         SZN     DBT2,D
     024716  777777 6010 00        8831         TNZ     $ERROR
     024717  024710 0010 00        8832 DBST1   TALLY   DBUS4,*-DBUS4+1
     024720  000000 2350 07        8833 ETYP    LDA     0,DL            GET A ZERO
     024721  017474 7000 00        8834         TSX0    GAD             AND ADD TO GENERATED OUTPUT
     024722  035216 2270 00        8835         LDX7    T$MODE          GET POINTER TO START OF MODE TABLE
     024723  000001 0670 03        8836 ETYP1   ADX7    1,DU            STEP OVER LINK WORD
     024724  000000 2200 17        8837         LDX0    0,7             GET TYPE OF MODE IN XR - 0
     024725   017012 1000 03         8838         CMPX0    M$XFER,DU          IS IT A VALID MODE
     024726   024741 6000 00         8839         TZE      ETYP2              TRANSFER IF NOT
     024727   017015 1000 03         8840         CMPX0    M$EMPTY,DU         IS IT A VALID MODE
     024730   024741 6000 00         8841         TZE      ETYP2              TRANSFER IF NOT
     024731   000000 7250 17         8842         LXL5     0,7                GET POINTER TO TYPE FOR MODE IN XR - 5
     024732   035227 0650 00         8843         ADX5     T$TYPE             MAKE TYPE POINTER ABSOLUTE
     024733   777777 7200 15         8844         LXL0     -1,5               GET POINTER TO USES OF TYPE IN PROGRAM
     024734   024741 6000 00         8845         TZE      ETYP2              TRANSFER IF NOT USED
 END OF BINARY CARD 00000435
     024735   035216 1670 00         8846         SBX7     T$MODE             MAKE MODE POINTER RELATIVE
     024736   025130 7000 00         8847         TSX0     MOVT               MOVE TYPE TO GENERATED OUTPUT
     024737   025234 7000 00         8848         TSX0     GUM                MOVE MODE OF TYPE TO GENERATED OUTPUT
     024740   035216 0670 00         8849         ADX7     T$MODE             MAKE MODE POINTER ABSOLUTE
     024741   777777 0670 17         8850 ETYP2   ADX7     -1,7               GET POINTER TO FOLLOWING MODE
     024742   000000 2340 17         8851         SZN      0,7                ARE THERE ANY MORE MODES
     024743   024723 6010 00         8852         TNZ      ETYP1              TRANSFER IF MORE MODES
     024744   035227 2250 00         8853         LDX5     T$TYPE             GET POINTER TO START OF TYPE TABLE
     024745   000001 0650 03         8854 ETYP3   ADX5     1,DU               STEP OVER LINK WORD
     024746   777777 7200 15         8855         LXL0     -1,5               SEE IF TYPE IS USED IN PROGRAM
     024747   024756 6000 00         8856         TZE      ETYP4              TRANSFER IF NOT USED
     024750   025130 7000 00         8857         TSX0     MOVT               MOVE TYPE TO GENERATED OUTPUT
     024751   000001 2350 03         8858         LDA      1,DU               GET A VACUUS MODE
     024752   035227 1650 00         8859         SBX5     T$TYPE             MAKE TYPE POINTER RELATIVE
     024753   017474 7000 00         8860         TSX0     GAD                AND ADD TO GENERATED OUTPUT
     024754   035227 0650 00         8861         ADX5     T$TYPE             MAKE TYPE POINTER ABSOLUTE
     024755   024757 7100 00         8862         TRA      ETYP5              AND CONTINUE
     024756   777777 0650 15         8863 ETYP4   ADX5     -1,5               STEP TO NEXT TYPE
     024757   000000 2340 15         8864 ETYP5   SZN      0,5                ARE THERE ANY MORE TYPES
     024760   024745 6010 00         8865         TNZ      ETYP3              TRANSFER IF MORE TYPES
     024761   000000 2350 07         8866         LDA      0,DL               GET A ZERO
     024762   017474 7000 00         8867         TSX0     GAD                AND ADD TO GENERATED OUTPUT
 END OF BINARY CARD 00000436
     024763   016470 7250 00         8868         LXL5     2$LINK0            GET POINTER TO CHAIN OF SYMREFS IN XR - 5
     024764   025000 6040 00         8869 SYMR1   TMI      SYMD1              TRANSFER IF NO MORE SYMREFS
     024765   025167 7000 00         8870         TSX0     SYM                SET UP HEADER FOR SYMREF IN SYM PROTOTYPE
     024766   010630 7000 00         8871         TSX0     A$XFER             MAKE MODE OF SYMBOL ABSOLUTE
     024767   777777 7200 17         8872         LXL0     -1,7               GET LENGTH OF VALUE IN XR - 0
     024770   025223 4400 00         8873         SXL0     SYMP               AND STORE IN SYM PROTOTYPE
     024771   035216 1670 00         8874         SBX7     T$MODE             MAKE MODE POINTER RELATIVE
     024772   025226 2350 00         8875         LDA      SYMT               GET TALLY WORD FOR SYM PROTOTYPE
     024773   017507 7000 00         8876         TSX0     GADL               AND ADD PROTOTYPE TO GENERATED OUTPUT
     024774   025234 7000 00         8877         TSX0     GUM                ADD MODE TO GENERATED OUTPUT
     024775   035220 0650 00         8878         ADX5     T$DEF              MAKE DEFINITION POINTER IN XR - 5 ABSOLUTE
     024776   000001 7250 15         8879         LXL5     1,5                GET NEXT SYMREF IN CHAIN
     024777   024764 7100 00         8880         TRA      SYMR1              AND LOOP
     025000   025227 4500 00         8881 SYMD1   STZ      SYMDT              INITIALIZE SYMDEF POINTER
     025001   025227 7200 00         8882 SYMD2   LXL0     SYMDT              GET POINTER TO NEXT SYMDEF
     025002   035232 0600 00         8883         ADX0     T$SDEF             MAKE POINTER ABSOLUTE
     025003   000000 2250 10         8884         LDX5     0,0                GET POINTER TO DEFINITION IN XR - 5
     025004   025015 6000 00         8885         TZE      SYMD3              TRANSFER IF NO MORE SYMDEFS
```

```
             3                                      PASS 3

      025005  025167 7000 00       8886        TSX0     SYM              SET UP HEADER FOR SYMDEF IN SYM PROTOTYPE
      025006  000000 2200 03       8887        LDX0     0,DU             GET A SYMDEF FLAG
      025007  025223 4400 00       8888        SXL0     SYMP             AND STORE IN SYM PROTOTYPE
      025010  025226 2350 00       8889        LDA      SYMT             GET TALLY WORD FOR SYM PROTOTYPE
   END OF BINARY CARD 00000437
      025011  017507 7000 00       8890        TSX0     GADL             AND ADD PROTOTYPE TO GENERATED OUTPUT
      025012  025234 7000 00       8891        TSX0     GUM              ADD MODE TO GENERATED OUTPUT
      025013  025227 0540 00       8892        AOS      SYMDT            STEP TO NEXT SYMDEF POINTER
      025014  025001 7100 00       8893        TRA      SYMD2            AND LOOP
      025015  017040 2350 00       8894 SYMD3  LDA      MAXS             GET SIZE OF LLO FOR THE PROGRAM
      025016  017474 7000 00       8895        TSX0     GAD              AND ADD TO GENERATED OUTPUT
      025017  011322 2340 00       8896        SZN      2$LIBF           SEE IF JUST COMPILING
      025020  025102 6010 00       8897        TNZ      LOAD             TRANSFER IF EXECUTING
      025021  000002 6220 00       8898        EAX2     2                GET FILE REFERENCE NUMBER OF USER CATALOG
      025022  025100 2370 00       8899        LDAQ     OBJCT            GET NAME OF OBJECT PROGRAM
      025023  007000 2210 03       8900        LDX1     =0007000,DU      ASK FOR READ, WRITE, AND APPEND
      025024  003662 7000 00       8901        TSX0     $OPEN            TRY TO OPEN ,OBJECT, IN USER CATALOG
      025025  000222 2370 51       8902        LDAQ     ,$STAT,I         GET STATUS RETURN WORDS IN AQ
      025026  000000 6200 01       8903        EAX0     0,AU             GET STATUS OF OPERATION IN XR - 0
      025027  000770 3000 03       8904        CANX0    =0770,DU         SEE IF BAD STATUS
      025030  777777 6010 00       8905        TNZ      $ERROR           ERROR - BAD STATUS
      025031  007000 1000 03       8906        CMPX0    =0007000,DU      DID WE GET DESIRED ACCESS
      025032  025054 6000 00       8907        TZE      LIB2             TRANSFER IF YES
      025033  000003 1000 03       8908        CMPX0    3,DU             WAS FILE NOT FOUND
      025034  025036 6000 00       8909        TZE      LIB1             TRANSFER IF FILE NOT FOUND
      025035  777777 7100 00       8910        TRA      $ERROR           STATUS NOT ACCEPTABLE
      025036  004007 7000 00       8911 LIB1   TSX0     $OPENS           GET A SCRATCH FILE
   END OF BINARY CARD 00000438
      025037  000222 2370 51       8912        LDAQ     ,$STAT,I         GET STATUS IN AQ REGISTER
      025040  000000 6200 01       8913        EAX0     0,AU             GET STATUS OF OPERATION IN XR - 0
      025041  777777 6010 00       8914        TNZ      $ERROR           ERROR - BAD STATUS
      025042  000000 6200 05       8915        EAX0     0,AL             GET FILE REFERENCE NUMBER OF FILE IN XR - 0
      025043  025076 7400 00       8916        STX0     FRN              AND STORE
      025044  000002 6220 00       8917        EAX2     2                GET FILE REFERENCE NUMBER OF CATALOG
      025045  025076 2230 00       8918        LDX3     FRN              GET FILE REFERENCE NUMBER OF FILE
      025046  025100 2370 00       8919        LDAQ     OBJCT            GET NAME IN AQ
      025047  004024 7000 00       8920        TSX0     $CAT             CATALOG FILE IN CATALOG WITH GIVEN NAME
      025050  000222 2370 51       8921        LDAQ     ,$STAT,I         GET STATUS OF OPERATION
      025051  000000 6200 01       8922        EAX0     0,AU             GET STATUS IN XR - 0
      025052  777777 6010 00       8923        TNZ      $ERROR           ERROR - BAD STATUS
      025053  025056 7100 00       8924        TRA      LIB3             AND CONTINUE
      025054  000000 6200 05       8925 LIB2   EAX0     0,AL             GET FILE REFERENCE NUMBER IN XR - 0
      025055  025076 7400 00       8926        STX0     FRN              AND STORE
      025056  035226 2350 00       8927 LIB3   LDA      T$GEN            GET DESCRIPTOR FOR GENERATED CODE
      025057  025076 2360 00       8928        LDQ      FRN              GET FILE REFERENCE NUMBER OF FILE
      025060  003744 7000 00       8929        TSX0     $WRITE           WRITE OBJECT CODE IN FILE
      025061  000222 2370 51       8930        LDAQ     ,$STAT,I         GET STATUS RETURN WORDS IN AQ
      025062  777777 6010 00       8931        TNZ      $ERROR           ERROR - BAD STATUS
      025063  035226 2350 00       8932        LDA      T$GEN            GET LENGTH OF OBJECT CODE IN AL
      025064  777777 3750 07       8933        ANA      -1,DL            ZERO OUT AU
```

```
  END OF BINARY CARD 00000439
      025065  025076 2360 00          8934         LDQ      FRN                 GET FILE REFERENCE NUMBER OF FILE
      025066  004044 7000 00          8935         TSX0     $TRUNC              AND SET FILE LENGTH TO OBJECT CODE LENGTH
      025067  000222 2370 51          8936         LDAQ     ,$STAT,I            GET STATUS WORDS OF OPERATION IN AQ
      025070  777777 6010 00          8937         TNZ      $ERROR              ERROR - BAD STATUS
      025071  025076 2360 00          8938         LDQ      FRN                 GET FILE REFERENCE NUMBER OF FILE
      025072  003702 7000 00          8939         TSX0     $CLOSE              AND CLOSE FILE
      025073  000222 2370 51          8940         LDAQ     ,$STAT,I            GET STATUS WORDS OF OPERATION IN AQ
      025074  777777 6010 00          8941         TNZ      $ERROR              ERROR - BAD STATUS
      025075  000000 000000           8942         ZERO                         DONE WITH COMPILATION
      025076  000000 000000           8943 FRN     ZERO
      025077  000000011007
                             025100   8944         EVEN
      025100  056117102112            8945 OBJCT   UASCI    2,,OBJECT,
      025101  105103124056
      025102  035226 2350 00          8946 LOAD    LDA      T$GEN               GET DESCRIPTOR FOR GENERATED CODE
      025103  030234 7550 00          8947         STA      L$USER              AND STORE IN LOADER TABLE
      025104  000000 6200 01          8948         EAX0     0,AU                GET ADDRESS OF BASE OF CODE SEGMENT
      025105  030131 7400 00          8949         STX0     L$FREE              AND STORE AS POINTER TO FREE STORAGE
      025106  030234 7200 00          8950         LXL0     L$USER              GET LENGTH OF GENERATED CODE
      025107  030234 0600 00          8951         ADX0     L$USER              ADD LOCATION TO GET POINTER TO END OF CODE
      025110  000001 6350 10          8952         EAA      1,0                 GET DESCRIPTOR FOR TYPE TABLE
      025111  777777 4500 01          8953         STZ      -1,AU               ZERO OUT LAST WORD OF PRECEDING TABLE
      025112  030235 7550 00          8954         STA      L$TYPE              STORE DESCRIPTOR FOR TYPE TABLE
  END OF BINARY CARD 00000440
      025113  000001 6350 01          8955         EAA      1,AU                GET DESCRIPTOR FOR SYM TABLE
      025114  777777 4500 01          8956         STZ      -1,AU               ZERO OUT LAST WORD OF PRECEDING TABLE
      025115  030236 7550 00          8957         STA      L$SYM               STORE DESCRIPTOR FOR SYM TABLE
      025116  000001 6350 01          8958         EAA      1,AU                GET DESCRIPTOR FOR CHAIN TABLE
      025117  777777 4500 01          8959         STZ      -1,AU               ZERO OUT LAST WORD OF PRECEDING TABLE
      025120  030237 7550 00          8960         STA      L$CHAIN             STORE DESCRIPTOR FOR CHAIN TABLE
      025121  000001 6350 01          8961         EAA      1,AU                GET DESCRIPTOR FOR PROG TABLE
      025122  777777 4500 01          8962         STZ      -1,AU               ZERO OUT LAST WORD OF PRECEDING TABLE
      025123  030240 7550 00          8963         STA      L$PROG              STORE DESCRIPTOR FOR PROG TABLE
      025124  000001 6350 01          8964         EAA      1,AU                GET DESCRIPTOR FOR END
      025125  777777 4500 01          8965         STZ      -1,AU               ZERO OUT LAST WORD OF PRECEDING TABLE
      025126  030241 7550 00          8966         STA      L$TEND              STORE POINTER TO END OF TABLES
      025127  027324 7100 00          8967         TRA      L$START             AND GO TO LOADER
      025130  025165 7400 00          8968 MOVT    STX0     MOVTX               SAVE RETURN
      025131  777777 2360 15          8969         LDQ      -1,5                GET LINK TO USER USES OF TYPE
      025132  000000 2200 03          8970         LDX0     0,DU                GET A ZERO
      025133  777777 4400 15          8971         SXL0     -1,5                AND CLEAR OUT USER LINK
      025134  000000 6240 02          8972         EAX4     0,QU                GET LENGTH OF TYPE IN XR - 4
      025135  025166 4500 00          8973         STZ      MOVTT               INITIALIZE TYPE TERMINATION WORD
      025136  025166 4440 00          8974         SXL4     MOVTT               STORE LENGTH OF TYPE IN TERMINATION WORD
      025137  000000 2350 15          8975 MOVT1   LDA      0,5                 GET NEXT WORD OF TYPE
      025140  750556 6200 01          8976         EAX0     -T$SKIP,AU          SEE IF TYPE WORD IS A SKIP
  END OF BINARY CARD 00000441
      025141  025144 6010 00          8977         TNZ      MOVT2               TRANSFER IF NOT A SKIP
      025142  000000 6200 05          8978         EAX0     0,AL                GET NUMBER OF WORDS SKIPPED IN XR - 0
```

```
    025143  025145 7100 00        8979          TRA      MOVT3            AND CONTINUE
    025144  000001 2200 03        8980 MOVT2    LDX0     1,DU             GET NUMBER OF WORDS ASSOCIATED WITH TYPE
    025145  025166 0400 00        8981 MOVT3    ASX0     MOVTT            AND ADD INTO TYPE TERMINATION WORD
    025146  035227 1650 00        8982          SBX5     T$TYPE           MAKE TYPE POINTER RELATIVE
    025147  017474 7000 00        8983          TSX0     GAD              ADD TYPE WORD TO GENERATED OUTPUT
    025150  035227 0650 00        8984          ADX5     T$TYPE           MAKE TYPE POINTER ABSOLUTE
    025151  000001 0650 03        8985          ADX5     1,DU             STEP POINTER TO NEXT TYPE WORD
    025152  000001 1640 03        8986          SBX4     1,DU             DECREMENT WORD COUNT OF TYPE
    025153  025137 6010 00        8987          TNZ      MOVT1            TRANSFER IF MORE WORDS TO ADD TO GENERATED OUTPUT
    025154  025166 3350 00        8988          LCA      MOVTT            GET TERMINATION WORD IN A REGISTER
    025155  035227 1650 00        8989          SBX5     T$TYPE           MAKE TYPE TABLE POINTER RELATIVE
    025156  017474 7000 00        8990          TSX0     GAD              ADD TERMINATION WORD TO TYPE
    025157  035227 0650 00        8991          ADX5     T$TYPE           MAKE TYPE TABLE POINTER ABSOLUTE
    025160  000044 7370 00        8992          LLS      36               MOVE USER POINTER TO AL
    025161  000002 0750 03        8993          ADA      2,DU             ALLOW FOR TERMINATION WORD AND USE WORD
    025162  035227 1650 00        8994          SBX5     T$TYPE           MAKE TYPE POINTER RELATIVE
    025163  017474 7000 00        8995          TSX0     GAD              AND ADD TO GENERATED OUTPUT
    025164  035227 0650 00        8996          ADX5     T$TYPE           MAKE TYPE POINTER ABSOLUTE
    025165  000000 7100 00        8997 MOVTX    TRA      **               AND RETURN
    025166  000000 000000         8998 MOVTT    ZERO
END OF BINARY CARD 00000442
    025167  025222 7400 00        8999 SYM      STX0     SYMX             SAVE RETURN
    025170  035220 0650 00        9000          ADX5     T$DEF            MAKE DEFINITION POINTER IN XR - 5 ABSOLUTE
    025171  000003 2350 15        9001          LDA      3,5              GET ADDRESS OF VALUE IN A
    025172  025223 7550 00        9002          STA      SYMP             AND STORE IN SYM PROTOTYPE
    025173  000002 2270 15        9003          LDX7     2,5              GET MODE OF SYMBOL IN XR - 7
    025174  000000 2210 15        9004          LDX1     0,5              GET POINTER TO NEXT DEFINITION OF THIS SYMBOL
    025175  025201 6040 00        9005          TMI      SYM2             TRANSFER IF THERE ARE NO MORE
    025176  035220 0610 00        9006 SYM1     ADX1     T$DEF            MAKE DEFINITION POINTER IN XR - 1 ABSOLUTE
    025177  000000 2210 11        9007          LDX1     0,1              GET POINTER TO DEFINITION FOLLOWING THIS SYMBOL
    025200  025176 6050 00        9008          TPL      SYM1             TRANSFER IF THERE IS ONE
    025201  400000 6210 11        9009 SYM2     EAX1     131072,1         MAKE XR - 1 POINTER TO SYMBOL OF DEFINITIONS
    025202  035222 0610 00        9010          ADX1     T$STAB           MAKE STAB POINTER ABSOLUTE
    025203  000000 2350 11        9011          LDA      ASSC,1           GET SC TALLY WORD FOR SYMBOL IN A REGISTER
    025204  025230 7550 00        9012          STA      TAL1             AND STORE
    025205  035223 2200 00        9013          LDX0     T$ITAB           GET POINTER TO BASE OF ITAB IN XR - 0
    025206  025230 0400 00        9014          ASX0     TAL1             AND MAKE TALLY WORD ABSOLUTE
    025207  025231 2350 00        9015          LDA      TAL2I            GET TALLY WORD TO SYM PROTOTYPE
    025210  025232 7550 00        9016          STA      TAL2             AND STORE
    025211  025233 2350 00        9017          LDA      BLNKS            GET FOUR ASCII BLANKS
    025212  025224 7550 00        9018          STA      SYMP+1           AND INITIALIZE SYMBOL TO BLANKS
    025213  025225 7550 00        9019          STA      SYMP+2           AND INITIALIZE SYMBOL TO BLANKS
    025214  025230 2360 52        9020 SYM3     LDQ      TAL1,SC          GET NEXT CHARACTER OF SYMBOL
END OF BINARY CARD 00000443
    025215  025217 6070 00        9021          TTF      SYM4             TRANSFER IF THERE ARE MORE CHARACTERS
    025216  025221 7100 00        9022          TRA      SYM5             TRANSFER IF NO MORE SYMBOLS

    025217  025232 7560 52        9023 SYM4     STQ      TAL2,SC          AND STORE IN SYM PROTOTYPE
    025220  025214 6070 00        9024          TTF      SYM3             TRANSFER IF MORE ROOM IN SYM PROTOTYPE
    025221  035220 1650 00        9025 SYM5     SBX5     T$DEF            MAKE DEFINITION POINTER IN XR - 5 RELATIVE
    025222  000000 7100 00        9026 SYMX     TRA      **               AND RETURN
```

```
           3                                              PASS 3

     025223  000000000000          9027 SYMP    OCT      0,0,0
     025224  000000000000
     025225  000000000000
     025226  025223 0004 00        9028 SYMT    TALLY    SYMP,*-SYMP+1
     025227  000000 000000         9029 SYMDT   ZERO
     025230  000000 000000         9030 TAL1    ZERO
     025231  025224 0010 40        9031 TAL2I   TALLYB   SYMP+1,8
     025232  000000 000000         9032 TAL2    ZERO
     025233  040040040040          9033 BLNKS   UASCI    1,
     025234  025357 7400 00        9034 GUM     STX0     GUMX           SAVE RETURN
     025235  035235 4500 56        9035         STZ      A$STACK,ID     PAD BOTTOM OF CONTROL STACK WITH A ZERO
     025236  005742 7170 00        9036         XED      T$SOVF         CHECK FOR STACK OVERFLOW
     025237  035226 7210 00        9037         LXL1     T$GEN          GET ADDRESS OF FIRST WORD IN GENERATED MODE
     025240  025360 7410 00        9038         STX1     GUMP           INITIALIZE CORRECTION POINTER
     025241  025361 7410 00        9039         STX1     GUMS           STORE ADDRESS OF FIRST WORD IN GENERATED MODE
     025242  025361 1610 00        9040         SBX1     GUMS           MAKE POINTER RELATIVE
   END OF BINARY CARD 00000444
     025243  035235 4410 51        9041 GUM1    SXL1     A$STACK,I      STORE NEW MODE POINTER
     025244  035235 7470 56        9042         STX7     A$STACK,ID     STORE OLD MODE POINTER
     025245  005742 7170 00        9043         XED      T$SOVF         CHECK FOR STACK OVERFLOW
     025246  010630 7000 00        9044         TSX0     A$XFER         MAKE MODE POINTER ABSOLUTE
     025247  777777 2350 17        9045         LDA      -1,7           GET LENGTH OF MODE IN AU
     025250  777777 6350 01        9046         EAA      -1,AU          SET UP REQUEST FOR AU WORDS
     025251  000000 2200 17        9047         LDX0     0,7            GET TYPE OF MODE IN XR - 0
     025252  016754 1000 03        9048         CMPX0    M$PRIM,DU      IS IT A PRIMITIVE MODE
     025253  025255 6010 00        9049         TNZ      GUM11          TRANSFER IF NOT PRIMITIVE
     025254  000000 6350 00        9050         EAA      1-1            ALLOCATE ONE WORD FOR PRIMITIVE MODE
     025255  035216 1670 00        9051 GUM11   SBX7     T$MODE         MAKE MODE POINTER RELATIVE
     025256  035226 2210 03        9052         LDX1     T$GEN,DU       GET POINTER TO GENERATED CODE TABLE WORD
     025257  005663 7000 00        9053         TSX0     T$ALOC         AND ALLOCATE SPACE IN THE GENERATED OUTPUT
     025260  035216 0670 00        9054         ADX7     T$MODE         MAKE MODE POINTER ABSOLUTE
     025261  777777 2200 17        9055         LDX0     -1,7           GET LENGTH OF MODE TABLE ENTRY IN XR - 0
     025262  000000 2350 17        9056         LDA      0,7            GET FIRST WORD OF ENTRY IN A REGISTER
     025263  000000 6220 01        9057         EAX2     0,AU           GET TYPE OF MODE IN XR - 2
     025264  016754 1020 03        9058         CMPX2    M$PRIM,DU      IS IT A PRIMITIVE TYPE
     025265  025267 6000 00        9059         TZE      GUM2           YES - TRANSFER
     025266  000000 6350 01        9060         EAA      0,AU           OTHERWISE ZERO OUT AL
     025267  400000 2750 03        9061 GUM2    ORA      =0400000,DU    SET SIGN BIT
     025270  025272 7100 00        9062         TRA      GUM4           AND CONTINUE
   END OF BINARY CARD 00000445
     025271  000000 2350 17        9063 GUM3    LDA      0,7            GET NEXT WORD OF MODE TABLE ENTRY
     025272  777777 7550 11        9064 GUM4    STA      -1,1           AND STORE IN GENERATED OUTPUT
     025273  400000 6230 01        9065         EAX3     131072,AU      GET TYPE OF MODE IN XR - 3
     025274  016754 1030 03        9066         CMPX3    M$PRIM,DU      IS IT A PRIMITIVE MODE
     025275  025323 6000 00        9067         TZE      GUM5           TRANSFER IF YES - STORE ONLY ONE WORD
     025276  000000 6230 05        9068         EAX3     0,AL           GET TAG IF ANY IN XR - 3
     025277  025317 6000 00        9069         TZE      GUM48          TRANSFER IF NO TAG TO FIX UP
     025300  035222 0630 00        9070         ADX3     T$STAB         MAKE TAG POINTER ABSOLUTE
     025301  000000 2350 13        9071         LDA      A$SC,3         GET TALLY WORD FOR TAG IN A REGISTER
     025302  025362 7550 00        9072         STA      GUMT           AND STORE
```

```
              3                                                  PASS 3

      025303  035223 2230 00          9073         LDX3    T$ITAB          GET BASE OF IDENTIFIER TABLE IN XR - 3
      025304  025362 0430 00          9074         ASX3    GUMT            MAKE TALLY WORD ABSOLUTE
      025305  025363 2350 00          9075         LDA     GUMTL           GET OUTPUT TALLY WORD
      025306  025364 7550 00          9076         STA     GUMT1           AND STORE
      025307  025362 2350 52          9077 GUM42   LDA     GUMT,SC         GET NEXT CHARACTER IN TAG
      025310  025313 6070 00          9078         TTF     GUM43           TRANSFER IF THERE IS A CHARACTER
      025311  025362 0110 54          9079         NOP     GUMT,DI         RESET TALLY FIELD IN TALLY WORD
      025312  000000 2350 07          9080         LDA     0,DL            GET A ZERO CHARACTER
      025313  025364 7550 52          9081 GUM43   STA     GUMT1,SC        AND STORE CHARACTER
      025314  025307 6070 00          9082         TTF     GUM42           TRANSFER IF MORE CHARACTERS NEEDED
      025315  000000 2230 03          9083 GUM44   LDX3    **,DU           GET TWO CHARACTERS OF TAG IN XR - 3
      025316  777777 4430 11          9084         SXL3    -1,1            AND STORE IN GENERATED MODE
    END OF BINARY CARD 00000446
      025317  000001 0610 03          9085 GUM48   ADX1    1,DU            STEP OUTPUT POINTER
      025320  000001 0670 03          9086         ADX7    1,DU            STEP MODE POINTER
      025321  000001 1600 03          9087         SBX0    1,DU            DECREMENT NUMBER OF WORDS LEFT TO TRANSFER
      025322  025271 6010 00          9088         TNZ     GUM3            TRANSFER IF MORE WORDS TO TRANSFER
      025323  035226 7200 00          9089 GUM5    LXL0    T$GEN           GET ADDRESS OF LAST WORD IN GENERATED OUTPUT
      025324  025360 1000 00          9090         CMPX0   GUMP            COMPARE WITH FIXUP POINTER
      025325  025350 6000 00          9091         TZE     GUM8            TRANSFER IF FIXUP IS DONE
      025326  025360 6200 56          9092         EAX0    GUMP,ID         GET ADDRESS OF NEXT WORD TO FIX UP
      025327  035226 0600 00          9093         ADX0    T$GEN           MAKE POINTER ABSOLUTE
      025330  000000 2210 10          9094         LDX1    0,0             GET OLD MODE POINTER THAT REQUIRES FIXING UP
      025331  025323 6040 00          9095         TMI     GUM5            TRANSFER IF NO FIXUP IS NEEDED
      025332  035235 2220 00          9096         LDX2    A$STACK         GET POINTER TO END OF CONTROL STACK
      025333  025362 7420 00          9097         STX2    GUMT            AND STORE
      025334  025362 2340 54          9098 GUM6    SZN     GUMT,DI         SEE IF THERE IS AN EQUIVALENCE STORED HERE
      025335  025343 6000 00          9099         TZE     GUM7            TRANSFER IF POINTER MODE NOT STORED YET
      025336  025362 1010 51          9100         CMPX1   GUMT,I          DOES THIS MATCH THE OLD POINTER TO BE FIXED UP
      025337  025334 6010 00          9101         TNZ     GUM6            TRANSFER IF NO
      025340  025362 7210 51          9102         LXL1    GUMT,I          GET NEW MODE POINTER IN XR - 1
      025341  000000 7410 10          9103         STX1    0,0             AND STORE TO FIX UP OLD MODE POINTER
      025342  025323 7100 00          9104         TRA     GUM5            GO TO FIX UP NEXT WORD
      025343  000000 6270 11          9105 GUM7    EAX7    0,1             GET MODE POINTER TO FIX UP IN XR - 7
      025344  035226 7210 00          9106         LXL1    T$GEN           GET POINTER TO MODE ABOUT TO BE GENERATED
    END OF BINARY CARD 00000447
      025345  025361 1610 00          9107         SBX1    GUMS            MAKE POINTER RELATIVE
      025346  000000 7410 10          9108         STX1    0,0             AND STORE AS NEW MODE POINTER
      025347  025243 7100 00          9109         TRA     GUM1            AND GO GENERATE THIS MODE
      025350  035235 2270 51          9110 GUM8    LDX7    A$STACK,I       RESTORE XR - 7
      025351  035235 2340 54          9111         SZN     A$STACK,DI      DELETE A WORD FROM THE CONTROL STACK
      025352  025350 6010 00          9112         TNZ     GUM8            TRANSFER IF MORE WORDS TO DELETE
      025353  035226 7200 00          9113         LXL0    T$GEN           GET CURRENT LENGTH OF GENERATED OUTPUT IN XR - 0
      025354  025361 1600 00          9114         SBX0    GUMS            SUBTRACT LENGTH WHEN ROUTINE WAS ENTERED
      025355  000001 6350 10          9115         EAA     1,0             GET LENGTH OF MODE + LENGTH WORD IN AU
      025356  017474 7000 00          9116         TSX0    GAD             AND ADD TO GENERATED OUTPUT
      025357  000000 7100 00          9117 GUMX    TRA     **              AND RETURN
      025360  000000 000000           9118 GUMP    ZERO
      025361  000000 000000           9119 GUMS    ZERO
      025362  000000 000000           9120 GUMT    ZERO
```

```
         3                                      PASS 3

     025363  025315 0002 40     GUMTL  TALLYB  GUM44,2,0
     025364  000000 000000      GUMT1  ZERO
                                       HEAD    R
     025365  025375 7400 00     ENTER  STX0    ENT1           SAVE RETURN
     025366  026540 2360 00            LDQ     GD             GET CURRENT ENVIRONMENT DESCRIPTOR IN Q
     025367  000003 7560 12            STQ     3,2            AND STORE IN NEW ENVIRONMENT
     025370  000002 2360 11            LDQ     2,1            GET NEW ENVIRONMENT DESCRIPTOR IN Q
     025371  026540 7560 00            STQ     GD             AND STORE AS CURRENT ENVIRONMENT DESCRIPTOR
     025372  000002 4500 11            STZ     2,1            ZERO OUT DESCRIPTOR IN MSCW FOR GARBAGE COLLECTOR
  END OF BINARY CARD 00000448
     025373  000003 2200 11            LDX0    3,1            GET ENTRY ADDRESS IN XR - 0
     025374  025377 7400 00            STX0    ENT2           AND SAVE
     025375  000000 2200 03     ENT1   LDX0    **,DU          GET RETURN ADDRESS IN XR - 0
     025376  000003 7400 11            STX0    3,1            AND SAVE IN OLD MSCW
     025377  000000 7100 00     ENT2   TRA     **             GO TO ENTRY LOCATION
     025400  000000 6210 16     RET    EAX1    0,S            SAVE RETURN ADDRESS IN XR - 1
     025401  000000 6260 17            EAX     S,0,D          CUT BACK STACK TO D REGISTER LOCATION
     025402  000003 2350 17            LDA     3,D            GET OLD ENVIRONMENT DESCRIPTOR IN A
     025403  026540 7550 00            STA     GD             AND MAKE IT CURRENT ENVIRONMENT DESCRIPTOR
     025404  000003 2270 16            LDX     D,3,S          RESTORE OLD D REGISTER CONTENTS
     025405  000000 7100 11            TRA     0,1            AND RETURN
     025406  025411 7400 00     CRLF   STX0    CRLFX          SAVE RETURN
     025407  002623 7000 00            TSX0    STTB           OUTPUT CARRIAGE RETURN / LINE FEED
     025410  025412 0003 40            TALLYB  CRLFC,3
     025411  000000 7100 00     CRLFX  TRA     **             AND RETURN
     025412  015012000000       CRLFC  OCT     015012000000
     025413  000060 0760 07     BOOL   ADQ     =0060,DL       TURN BOOLEAN VALUE INTO AN INTEGER CHARACTER
     025414  025420 7400 00     CHAR   STX0    CHARX          SAVE RETURN
     025415  025421 7560 00            STQ     CHART          SAVE CHARACTER
     025416  002623 7000 00            TSX0    STTB           OUTPUT CHARACTER
     025417  025421 0002 43            TALLYB  CHART,2,3
     025420  000000 7100 00     CHARX  TRA     **             AND RETURN
  END OF BINARY CARD 00000449
     025421  000000 000000      CHART  ZERO
     025422  025463 7400 00     INT    STX0    INTX           SAVE RETURN
     025423  025612 2350 00            LDA     OUTI           GET INITIAL TALLY WORD IN A REGISTER
     025424  025611 7550 00            STA     OUT            AND INITIALIZE OUTPUT TALLY WORD
     025425  000044 7370 00            LLS     36             MOVE ARGUMENT TO A REGISTER
     025426  106000 4110 03            LDE     =35B25,DU      MAKE IT FLOATING POINT
     025427  000000 5730 00            FNO                    AND NORMALIZE IT
     025430  025436 6050 00            TPL     INT1           TRANSFER IF POSITIVE NUMBER
     025431  000055 2360 07            LDQ     =0055,DL       GET A MINUS SIGN
     025432  025611 7560 52            STQ     OUT,SC         AND STORE IN OUTPUT BUFFER
     025433  000000 2360 07            LDQ     0,DL           RESTORE Q REGISTER
     025434  000000 5130 00            FNEG                   MAKE INTEGER POSITIVE
     025435  025441 7100 00            TRA     INT2           AND CONTINUE
     025436  000040 2360 07     INT1   LDQ     =0040,DL       GET A SPACE
     025437  025611 7560 52            STQ     OUT,SC         AND STORE IN OUTPUT BUFFER
     025440  000000 2360 07            LDQ     0,DL           RESTORE Q REGISTER
     025441  000400 4750 03     INT2   FAD     =0,5,DU        ROUND INTEGER TO AVOID ROUNDOFF
```

```
      025442  025464 5670 00        9169          DFDV     TLVN               DIVIDE BY 10*11
      025443  025606 7570 00        9170          STAQ     DN                 AND STORE AS NUMBER TO CONVERT
      025444  000013 2210 03        9171          LDX1     11,DU              PREPARE TO OUTPUT ELEVEN DIGITS
      025445  025567 7000 00        9172 INT3     TSX0     DIG                CONVERT A DIGIT
      025446  000001 1610 03        9173          SBX1     1,DU               DECREMENT NUMBER LEFT TO CONVERT
   END OF BINARY CARD 00000450
      025447  025445 6010 00        9174          TNZ      INT3               TRANSFER IF MORE TO CONVERT
      025450  025612 2360 00        9175          LDQ      OUTI               GET INITIAL TALLY WORD
      025451  025611 7560 00        9176          STQ      OUT                AND INITIALIZE OUTPUT TALLY WORD
      025452  000040 2350 07        9177          LDA      =0040,DL           GET A SPACE IN A REGISTER
      025453  000060 2360 07        9178          LDQ      =0060,DL           GET A ZERO IN Q REGISTER
      025454  025611 0110 52        9179          NOP      OUT,SC             SKIP OVER SIGN
      025455  025611 1160 50        9180 INT4     CMPQ     OUT,CI             SEE IF LEADING ZERO
      025456  025461 6010 00        9181          TNZ      INT5               TRANSFER IF NOT
      025457  025611 7550 52        9182          STA      OUT,SC             CHANGE LEADING ZERO TO A SPACE
      025460  025455 7100 00        9183          TRA      INT4               AND LOOP
      025461  002623 7000 00        9184 INT5     TSX0     STTB               PRINT INTEGER
      025462  025613 0016 40        9185          TALLYB   OUTB,14
      025463  000000 7100 00        9186 INTX     TRA      **                 AND RETURN
                       025464       9187          EVEN
      025464  112564416672          9188 TLVN     DEC      1D11
      025465  000000000000
      025466  000000 000000         9189          ZERO
      025467  025561 7400 00        9190 REAL     STX0     REALX              SAVE RETURN
      025470  025612 2360 00        9191          LDQ      OUTI               GET INITIAL TALLY WORD
      025471  025611 7560 00        9192          STQ      OUT                AND INITIALIZE OUTPUT TALLY WORD
      025472  000000 2360 07        9193          LDQ      0,DL               RESET Q REGISTER
      025473  025562 4500 00        9194          STZ      RE1                ZERO OUT NEGATIVE EXPONENT COUNT
      025474  025563 4500 00        9195          STZ      RE2                ZERO OUT POSITIVE EXPONENT COUNT
   END OF BINARY CARD 00000451
      025475  000000 5730 00        9196          FNO                         CHECK SIGN OF NUMBER
      025476  025513 6000 00        9197          TZE      REAL3              TRANSFER IF ZERO
      025477  002400 4250 03        9198 REAL1    FCMG     =1,0,DU            CHECK SIZE OF NUMBER
      025500  025504 6050 00        9199          TPL      REAL2              TRANSFER IF GREATER THAN ONE
      025501  010500 4610 03        9200          FMP      =10.0,DU           MULTIPLY BY TEN
      025502  025562 0540 00        9201          AOS      RE1                INCREMENT NEGATIVE EXPONENT COUNT
      025503  025477 7100 00        9202          TRA      REAL1              AND LOOP
      025504  025564 5670 00        9203 REAL2    DFDV     RTEN               DIVIDE BY TEN
      025505  025563 0540 00        9204          AOS      RE2                INCREMENT POSITIVE EXPONENT COUNT
      025506  002400 4250 03        9205          FCMG     =1,0,DU            CHECK SIZE OF NUMBER
      025507  025504 6050 00        9206          TPL      REAL2              LOOP IF BIGGER THAN ONE
      025510  730414 4750 03        9207          FAD      =0,0000005,DU      ADD ROUND CONSTANT
      025511  002400 4250 03        9208          FCMG     =1,0,DU            SEE IF ROUNDING OVERFLOWED NUMBER
      025512  025504 6050 00        9209          TPL      REAL2              TRANSFER IF OVERFLOWED
      025513  025606 7570 00        9210 REAL3    STAQ     DN                 SAVE NUMBER TO BE CONVERTED
      025514  000000 5730 00        9211          FNO                         CHECK SIGN OF NUMBER
      025515  025524 6050 00        9212          TPL      REAL4              TRANSFER IF POSITIVE
      025516  000055 2350 07        9213          LDA      =0055,DL           GET A MINUS SIGN
      025517  025611 7550 52        9214          STA      OUT,SC             AND STORE IN OUTPUT BUFFER

      025520  025606 2370 00        9215          LDAQ     DN                 GET NUMBER TO CONVERT
```

```
          R                                                    PASS 3

    025521  000000 5130 00          9216         FNEG                         MAKE IT POSITIVE
    025522  025606 7570 00          9217         STAQ     DN                  AND STORE NUMBER TO CONVERT
  END OF BINARY CARD 00000452
    025523  025526 7100 00          9218         TRA      REAL5               AND CONTINUE
    025524  000040 2350 07          9219 REAL4   LDA      =0040,DL            GET A SPACE
    025525  025611 7550 52          9220         STA      OUT,SC              AND STORE IN OUTPUT BUFFER
    025526  000056 2350 07          9221 REAL5   LDA      =0056,DL            GET A DECIMAL POINT
    025527  025611 7550 52          9222         STA      OUT,SC              AND STORE IN OUTPUT BUFFER
    025530  000006 2210 03          9223         LDX1     6,DU                GET NUMBER OF DIGITS TO CONVERT IN XR - 1
    025531  025567 7000 00          9224 REAL6   TSX0     DIG                 CONVERT A DIGIT
    025532  000001 1610 03          9225         SBX1     1,DU                DECREMENT NUMBER OF DIGITS LEFT TO CONVERT
    025533  025531 6010 00          9226         TNZ      REAL6               TRANSFER IF MORE TO CONVERT
    025534  000040 2360 07          9227         LDQ      =0040,DL            GET A SPACE
    025535  025611 7560 52          9228         STQ      OUT,SC              AND STORE IN OUTPUT BUFFER
    025536  000105 2360 07          9229         LDQ      =0105,DL            GET A LETTER 'E'
    025537  025611 7560 52          9230         STQ      OUT,SC              AND STORE IN OUTPUT BUFFER
    025540  025563 2360 00          9231         LDQ      RE2                 GET POSITIVE EXPONENT COUNT
    025541  025562 1760 00          9232         SBQ      RE1                 SUBTRACT NEGATIVE EXPONENT COUNT
    025542  025545 6040 00          9233         TMI      REAL7               TRANSFER IF NEGATIVE EXPONENT
    025543  000040 2350 07          9234         LDA      =0040,DL            GET A SPACE
    025544  025547 7100 00          9235         TRA      REAL8               AND CONTINUE
    025545  000000 5330 00          9236 REAL7   NEGL                         MAKE EXPONENT POSITIVE
    025546  000055 2350 07          9237         LDA      =0055,DL            GET A MINUS SIGN
    025547  025611 7550 52          9238 REAL8   STA      OUT,SC              AND STORE IN OUTPUT BUFFER
    025550  000044 7370 00          9239         LLS      36                  GET EXPONENT IN A REGISTER
  END OF BINARY CARD 00000453

    025551  106000 4110 03          9240         LDE      =35B25,DU           MAKE IT FLOATING POINT
    025552  000400 4750 03          9241         FAD      =0.5,DU             ADD ROUND CONSTANT
    025553  016620 5650 03          9242         FDV      =100.0,DU           DIVIDE BY 100
    025554  025606 7570 00          9243         STAQ     DN                  AND STORE NUMBER TO CONVERT
    025555  025567 7000 00          9244         TSX0     DIG                 CONVERT A DIGIT
    025556  025567 7000 00          9245         TSX0     DIG                 CONVERT A DIGIT
    025557  002623 7000 00          9246         TSX0     STTB                OUTPUT FLOATING POINT NUMBER
    025560  025613 0017 40          9247         TALLYB   OUTB,15
    025561  000000 7100 00          9248 REALX   TRA      **                  AND RETURN
    025562  000000 000000           9249 RE1     ZERO
    025563  000000 000000           9250 RE2     ZERO
                       025564       9251         EVEN
    025564  010500000000            9252 RTEN    DEC      1D1
    025565  000000000000
    025566  000000 000000           9253         ZERO
    025567  025606 2370 00          9254 DIG     LDAQ     DN                  GET NUMBER TO CONVERT IN EAQ REGISTER
    025570  010500 4610 03          9255         FMP      =10.0,DU            GET NEXT DIGIT AS INTEGRAL PART
    025571  025606 7570 00          9256         STAQ     DN                  SAVE NUMBER
    025572  025610 4560 00          9257         STE      DE                  SAVE EXPONENT
    025573  216000 4350 03          9258         UFA      =71B25,DU           GET DIGIT IN QL
    025574  000060 0760 07          9259         ADQ      =0060,DL            ADD ASCII ZERO
    025575  025611 7560 52          9260         STQ      OUT,SC              STORE DIGIT IN OUTPUT BUFFER
    025576  000060 1760 07          9261         SBQ      =0060,DL            RESTORE DIGIT IN Q REGISTER
  END OF BINARY CARD 00000454
```

```
              R                                                    PASS 3

      025577  000000 5730 00        9262          FNO                         GET FLOATING INTEGER PART
      025600  025606 6770 00        9263          ERAQ      DN                GET FRACTION PART IN EAQ
      025601  025610 4110 00        9264          LDE       DE                FIX EXPONENT IF DIGIT WAS ZERO
      025602  000000 5730 00        9265          FNO                         NORMALIZE NUMBER
      025603  025606 7570 00        9266          STAQ      DN                AND STORE NUMBER TO CONVERT
      025604  000000 7100 10        9267          TRA       0,0               AND RETURN
      025605  000000011007
                          025606    9268          EVEN
      025606  000000000000          9269 DN       OCT       0,0
      025607  000000000000
      025610  000000 000000         9270 DE       ZERO
      025611  000000 000000         9271 OUT      ZERO
      025612  025613 0000 41        9272 OUTI     TALLYB    OUTB,0,1
      025613  040040040040          9273 OUTB     UASCI     6,
      025614  040040040040
      025615  040040040040
      025616  040040040040
      025617  040040040040
      025620  040040040040
      025621  025732 7400 00        9274 ACHEK    STX0      ACHX              SAVE RETURN
      025622  000002 2200 11        9275          LDX0      2,1               CHECK DESCRIPTOR FLAG WORD
      025623  025646 6010 00        9276          TNZ       ACH2              TRANSFER IF DESTINATION IS FLEXIBLE
      025624  000000 6240 11        9277          EAX4      0,1               GET POINTER TO MARK STACK WORD IN XR - 4
    END OF BINARY CARD 00000455
      025625  000002 2210 14        9278          LDX1      2,4               GET POINTER TO DESTINATION DESCRIPTOR IN XR - 1
      025626  000003 2220 14        9279          LDX2      3,4               GET POINTER TO SOURCE DESCRIPTOR IN XR - 2

      025627  000002 2360 11        9280 ACH1     LDQ       2,1               GET LOWER BOUND IN DESTINATION
      025630  000002 1160 12        9281          CMPQ      2,2               COMPARE WITH LOWER BOUND IN SOURCE
      025631  777777 6010 00        9282          TNZ       $ERROR            ERROR - BOUNDS ARE DIFFERENT
      025632  000003 2360 11        9283          LDQ       3,1               GET UPPER BOUND IN DESTINATION
      025633  000003 1160 12        9284          CMPQ      3,2               COMPARE WITH UPPER BOUND IN SOURCE
      025634  777777 6010 00        9285          TNZ       $ERROR            ERROR - BOUNDS ARE DIFFERENT
      025635  000004 0610 03        9286          ADX1      4,DU              STEP TO NEXT QUAD IN DESTINATION DESCRIPTOR
      025636  000004 0620 03        9287          ADX2      4,DU              STEP TO NEXT QUAD IN SOURCE DESCRIPTOR
      025637  000001 1630 03        9288          SBX3      1,DU              DECREMENT NUMBER OF DIMENSIONS LEFT
      025640  025627 6010 00        9289          TNZ       ACH1              TRANSFER IF MORE DIMENSIONS TO CHECK
      025641  000002 2210 14        9290          LDX1      2,4               GET POINTER TO DESTINATION DESCRIPTOR
      025642  000001 0610 03        9291          ADX1      1,DU              MAKE IT POINT TO FIRST ELEMENT POINTER
      025643  000003 2220 14        9292          LDX2      3,4               GET POINTER TO SOURCE DESCRIPTOR
      025644  000001 0620 03        9293          ADX2      1,DU              MAKE IT POINT TO FIRST ELEMENT POINTER
      025645  025732 7100 00        9294 ACHKX    TRA       ACHX              AND EXIT
      025646  025733 7410 00        9295 ACH2     STX1      ACHP              SAVE POINTER TO MARK STACK WORD
      025647  025734 7420 00        9296          STX2      ACHEL             SAVE ELEMENT LENGTH
      025650  025735 7430 00        9297          STX3      ACHD              SAVE DIMENSION OF ARRAYS
      025651  025736 4440 00        9298          SXL4      ACHTP             SAVE TYPE OF ELEMENT OF ARRAYS
      025652  025735 2360 00        9299          LDQ       ACHD              GET DIMENSION OF ARRAYS IN QU
    END OF BINARY CARD 00000456
      025653  000002 7360 00        9300          QLS       2                 MULTIPLY BY 4
      025654  000001 6360 02        9301          EAQ       1,QU              AND ADD ONE
      025655  025737 7560 00        9302          STQ       ACHT              AND SAVE IN MEMORY
```

```
             R                                              PASS 3

       025656  000000 6360 12        9303        EAQ     0,2              GET LENGTH OF ELEMENT IN QU
       025657  025740 7560 00        9304        STQ     ACHT1            AND STORE IN MEMORY
       025660  000003 2240 11        9305        LDX4    3,1              GET POINTER TO SOURCE DESCRIPTOR IN XR - 4
       025661  000003 2360 14        9306 ACH3   LDQ     3,4              GET UPPER BOUND IN Q REGISTER
       025662  000002 1760 14        9307        SBQ     2,4              SUBTRACT LOWER BOUND
       025663  000001 0760 07        9308        ADQ     1,DL             AND CALCULATE (U-L+1)
       025664  025740 4020 00        9309        MPY     ACHT1            MULTIPLY BY SIZE OF LAST DIMENSION
       025665  025740 7560 00        9310        STQ     ACHT1            AND STORE AS SIZE OF THIS DIMENSION
       025666  000004 0640 03        9311        ADX4    4,DU             STEP TO NEXT QUAD
       025667  000001 1630 03        9312        SBX3    1,DU             DECREMENT NUMBER OF DIMENSIONS LEFT
       025670  025661 6010 00        9313        TNZ     ACH3             TRANSFER IF MORE DIMENSIONS
       025671  025737 0760 00        9314        ADQ     ACHT             ADD TO LENGTH OF NEW ARRAY LENGTH OF DESCRIPTOR
       025672  027211 7560 00        9315        STQ     REQ              AND STORE SUM AS LENGTH TO REQUEST
       025673  025733 2360 00        9316        LDQ     ACHP             GET POINTER TO MARK STACK WITH TYPE
       025674  026162 7000 00        9317        TSX0    HEAP             ALLOCATE SPACE ON THE HEAP
       025675  000000 6210 01        9318        EAX1    0,AU             GET POINTER TO NEW MEMORY IN XR - 1
       025676  025736 0750 00        9319        ADA     ACHTP            ADD TYPE TO POINTER TO NEW MEMORY
       025677  025737 0750 00        9320        ADA     ACHT             ADD LENGTH OF DESCRIPTOR TO POINTER
       025700  000000 7550 11        9321        STA     0,1              STORE POINTER TO NEW ARRAY IN NEW DESCRIPTOR
    END OF BINARY CARD 00000457
       025701  025733 2230 00        9322        LDX3    ACHP             GET POINTER TO MARK STACK WORD
       025702  000002 2240 13        9323        LDX4    2,3              GET POINTER TO DESTINATION DESCRIPTOR IN XR - 3
       025703  000000 7410 14        9324        STX1    0,4              STORE POINTER TO NEW DESCRIPTOR IN OLD FLAG WORD
       025704  000003 2220 13        9325        LDX2    3,3              GET POINTER TO SOURCE DESCRIPTOR IN XR - 2
       025705  000001 0620 03        9326        ADX2    1,DU             STEP OVER FLAG WORD TO ELEMENT POINTER
       025706  025735 2360 00        9327        LDQ     ACHD             GET DIMENSION OF ARRAYS IN QU
       025707  000002 7360 00        9328        QLS     2                MULTIPLY BY 4
       025710  025737 7560 00        9329        STQ     ACHT             AND STORE
       025711  025737 0610 00        9330        ADX1    ACHT             GET POINTER TO END OF DESTINATION DESCRIPTOR IN 1
       025712  025737 0620 00        9331        ADX2    ACHT             GET POINTER TO END OF SOURCE DESCRIPTOR IN XR - 2
       025713  025734 2360 00        9332        LDQ     ACHEL            GET LENGTH OF ELEMENT IN QU
       025714  025735 2240 00        9333        LDX4    ACHD             GET DIMENSION OF ARRAYS IN XR - 4
       025715  000004 1610 03        9334 ACH4   SBX1    4,DU             DECREMENT DESTINATION POINTER BY ONE QUAD
       025716  000004 1620 03        9335        SBX2    4,DU             DECREMENT SOURCE POINTER BY ONE QUAD
       025717  000004 4500 11        9336        STZ     4,1              ZERO OUT DESTINATION STATE WORD
       025720  000003 7560 11        9337        STQ     3,1              STORE STRIDE IN DESTINATION QUAD
       025721  000001 2360 12        9338        LDQ     1,2              GET LOWER BOUND OF SOURCE
       025722  000001 7560 11        9339        STQ     1,1              AND STORE IT IN DESTINATION
       025723  000002 2360 12        9340        LDQ     2,2              GET UPPER BOUND OF SOURCE
       025724  000002 7560 11        9341        STQ     2,1              AND STORE IT IN DESTINATION
       025725  000001 1760 11        9342        SBQ     1,1              GET DIFFERENCE BETWEEN LOWER AND UPPER BOUNDS
       025726  000001 0760 07        9343        ADQ     1,DL             CALCULATE (U-L+1)
    END OF BINARY CARD 00000458
       025727  000003 4020 11        9344        MPY     3,1              MULTIPLY BY STRIDE
       025730  000001 1640 03        9345        SBX4    1,DU             DECREMENT NUMBER OF DIMENSIONS LEFT
       025731  025715 6010 00        9346        TNZ     ACH4             TRANSFER IF MORE DIMENSIONS
       025732  000000 7100 00        9347 ACHX   TRA     **               AND RETURN WITH XR - 1 AND XR - 2 SET
       025733  000000 027235         9348 ACHP   ZERO    0,T$MS
       025734  000000 000000         9349 ACHEL  ZERO
       025735  000000 000000         9350 ACHD   ZERO
```

```
     025736  000000 000000         9351 ACHTP   ZERO
     025737  000000 000000         9352 ACHT    ZERO
     025740  000000 000000         9353 ACHT1   ZERO
     025741  026007 7400 00        9354 AFS     STX0      AFSX           SAVE RETURN
     025742  026010 7410 00        9355         STX1      AFSP           SAVE POINTER TO MARK STACK WORD
     025743  026011 7420 00        9356         STX2      AFSEL          SAVE LENGTH OF ARRAY ELEMENT
     025744  026012 7430 00        9357         STX3      AFSD           SAVE DIMENSION OF ARRAY
     025745  026013 4440 00        9358         SXL4      AFSTP          SAVE TYPE OF ELEMENT
     025746  000000 6360 12        9359         EAQ       0,2            GET ELEMENT LENGTH IN QU
     025747  000000 6200 11        9360         EAX0      0,1            GET POINTER TO MARK STACK WORD IN XR - 0
     025750  000002 2210 10        9361         LDX1      2,0            GET POINTER TO DESTINATION DESCRIPTOR IN XR - 1
     025751  000003 2220 10        9362         LDX2      3,0            GET POINTER TO SOURCE DESCRIPTOR IN XR - 2
     025752  000000 6350 13        9363         EAA       0,3            GET DIMENSION OF ARRAY IN AU
     025753  000002 7350 00        9364         ALS       2              MULTIPLY BY 4
     025754  000001 6350 01        9365         EAA       1,AU           AND ADD ONE
 END OF BINARY CARD 00000459
     025755  026014 7550 00        9366         STA       AFST           AND STORE IN MEMORY
     025756  026014 0610 00        9367         ADX1      AFST           GET POINTER TO END OF DESTINATION DESCRIPTOR IN 1
     025757  026014 0620 00        9368         ADX2      AFST           GET POINTER TO END OF SOURCE DESCRIPTOR IN XR - 2
     025760  000004 1610 03        9369 AFS1    SBX1      4,DU           BACK UP ONE QUAD IN DESTINATION DESCRIPTOR
     025761  000004 1620 03        9370         SBX2      4,DU           BACK UP ONE QUAD IN SOURCE DESCRIPTOR
     025762  000004 4500 11        9371         STZ       4,1            SET STATE WORD TO FIXED
     025763  000003 7560 11        9372         STQ       3,1            INITIALIZE STRIDE
     025764  000001 2360 12        9373         LDQ       1,2            GET SOURCE LOWER BOUND
     025765  000001 7560 11        9374         STQ       1,1            AND STORE IN DESTINATION LOWER BOUND
     025766  000002 2360 12        9375         LDQ       2,2            GET SOURCE UPPER BOUND
     025767  000002 7560 11        9376         STQ       2,1            AND STORE IN DESTINATION UPPER BOUND
     025770  000001 1760 11        9377         SBQ       1,1            SUBTRACT LOWER BOUND
     025771  000001 0760 07        9378         ADQ       1,DL           CALCULATE (U-L+1)
     025772  000003 4020 11        9379         MPY       3,1            MULTIPLY BY STRIDE
     025773  000001 1630 03        9380         SBX3      1,DU           DECREMENT NUMBER OF DIMENSIONS LEFT
     025774  025741 6010 00        9381         TNZ       AFS            TRANSFER IF MORE DIMENSIONS LEFT
     025775  027211 7560 00        9382         STQ       REQ            STORE LENGTH OF ARRAY AS NUMBER OF WORDS NEEDED
     025776  026010 2360 00        9383         LDQ       AFSP           GET POINTER TO MS WORD WITH TYPE IN Q
     025777  026133 7000 00        9384         TSX0      LOC            ALLOCATE SPACE FOR ARRAY ON STACK
     026000  026013 0750 00        9385         ADA       AFSTP          ADD TYPE OF ELEMENT
     026001  026010 2230 00        9386         LDX3      AFSP           GET POINTER TO MARK STACK WORD
     026002  000002 2210 10        9387         LDX1      2,0            GET POINTER TO DESTINATION DESCRIPTOR IN XR - 1
 END OF BINARY CARD 00000460
     026003  000001 0610 03        9388         ADX1      1,DU           STEP OVER FLAG WORD
     026004  000003 2220 10        9389         LDX2      3,0            GET POINTER TO SOURCE DESCRIPTOR IN XR - 2

     026005  000001 0620 03        9390         ADX2      1,DU           STEP OVER FLAG WORD
     026006  000000 7550 11        9391         STA       0,1            STORE POINTER TO FIRST ELEMENT IN DESCRIPTOR
     026007  000000 7100 00        9392 AFSX    TRA       **             AND RETURN WITH XR - 1 AND XR - 2 SET
     026010  000000 027235         9393 AFSP    ZERO      0,TSMS
     026011  000000 000000         9394 AFSEL   ZERO
     026012  000000 000000         9395 AFSD    ZERO
     026013  000000 000000         9396 AFSTP   ZERO
     026014  000000 000000         9397 AFST    ZERO
     026015  026104 4500 00        9398 RLGEN   STZ       RGNF           SET FLAG INDICATING LOCAL GENERATOR
```

```
          R                                               PASS 3

     026016  026020 7100 00         9399          TRA       RGN                 CONTINUE
     026017  026104 7500 00         9400 RHGEN    STC2      RGNF                SET FLAG INDICATING HEAP GENERATOR
     026020  026102 7400 00         9401 RGN      STX0      RGNX                SAVE RETURN
     026021  026110 7410 00         9402          STX1      RGNP                SAVE MARK STACK WORD POINTER
     026022  026111 7420 00         9403          STX2      RGNA                SAVE POINTER TO FLAG WORD OF ARRAY DESCRIPTOR
     026023  026105 7430 00         9404          STX3      RGND                SAVE DIMENSION OF ARRAY
     026024  026106 4440 00         9405          SXL4      RGNTP               SAVE TYPE OF ELEMENT OF ARRAY
     026025  000003 2360 12         9406          LDQ       3,2                 GET FIRST UPPER BOUND OF ARRAY
     026026  000002 1760 12         9407          SBQ       2,2                 SUBTRACT FIRST LOWER BOUND OF ARRAY
     026027  000001 0760 07         9408          ADQ       1,DL                CALCULATE (U-L+1)
     026030  000004 4020 12         9409          MPY       4,2                 MULTIPLY BY FIRST STRIDE TO GET LENGTH OF ARRAY
  END OF BINARY CARD 00000461
     026031  027211 7560 00         9410          STQ       REQ                 AND STORE AS LENGTH TO REQUEST
     026032  000000 2350 03         9411          LDA       0,DU                INITIALIZE A WITH A ZERO
     026033  000005 2750 12         9412 RGN1     ORA       5,2                 OR TO A STATE IN QUAD
     026034  000004 0620 03         9413          ADX2      4,DU                STEP TO NEXT QUAD
     026035  000001 1630 03         9414          SBX3      1,DU                DECREMENT NUMBER OF DIMENSIONS LEFT
     026036  026033 6010 00         9415          TNZ       RGN1                TRANSFER IF MORE DIMENSIONS TO DO
     026037  026103 7550 00         9416          STA       RGNS                STORE STATE TYPE OF ARRAY IN MEMORY
     026040  026112 2360 00         9417          LDQ       RGNTG               GET POINTER TO LIST STRUCTURE TO COLLECT
     026041  026104 2340 00         9418          SZN       RGNF                SEE IF LOCAL OR HEAP GENERATOR
     026042  026045 6010 00         9419          TNZ       RGN2                TRANSFER IF HEAP
     026043  026133 7000 00         9420          TSX0      LOC                 CALL LOCAL GENERATOR
     026044  026046 7100 00         9421          TRA       RGN3                AND CONTINUE
     026045  026162 7000 00         9422 RGN2     TSX0      HEAP                CALL HEAP GENERATOR
     026046  026106 0750 00         9423 RGN3     ADA       RGNTP               ADD ELEMENT TYPE TO ARRAY POINTER
     026047  026111 2220 00         9424          LDX2      RGNA                GET POINTER TO DESCRIPTOR IN XR - 2
     026050  000001 7550 12         9425          STA       1,2                 AND STORE POINTER TO ARRAY IN DESCRIPTOR
     026051  026103 2340 00         9426          SZN       RGNS                CHECK STATE TYPE OF ARRAY
     026052  026057 6010 00         9427          TNZ       RGN4                TRANSFER IF FLEXIBLE
     026053  026111 2220 00         9428          LDX2      RGNA                GET POINTER TO CALCULATED DESCRIPTOR
     026054  000000 2200 03         9429          LDX0      0,DU                GET A HALF WORD ZERO
     026055  000000 7400 12         9430          STX0      0,2                 AND STORE IN FLAG WORD OF DESCRIPTOR
     026056  026102 7100 00         9431          TRA       RGNX                AND EXIT
  END OF BINARY CARD 00000462
     026057  026111 2220 00         9432 RGN4     LDX2      RGNA                GET POINTER TO CALCULATED DESCRIPTOR
     026060  000000 2210 12         9433          LDX1      0,2                 GET POINTER TO ALLOCATED DESCRIPTOR
     026061  000000 4500 11         9434          STZ       0,1                 INITIALIZE FLAG WORD IN ALLOCATED DESCRIPTOR
     026062  000001 0610 03         9435          ADX1      1,DU                STEP OVER FLAG WORD
     026063  000001 0620 03         9436          ADX2      1,DU                MAKE BOTH INDEX REGISTERS POINTER TO C WORD
     026064  026105 2230 00         9437          LDX3      RGND                GET DIMENSION OF ARRAY IN XR - 3
     026065  000000 2360 12         9438          LDQ       0,2                 GET ORIGINAL C WORD
     026066  000000 7560 11         9439          STQ       0,1                 AND MOVE TO NEW DESCRIPTOR
     026067  000000 4500 12         9440          STZ       0,2                 ZERO OUT ORIGINAL POINTER WORD
     026070  000001 2360 12         9441 RGN5     LDQ       1,2                 GET OLD LOWER BOUND
     026071  000001 7560 11         9442          STQ       1,1                 AND STORE IN NEW LOWER BOUND

     026072  000002 2360 12         9443          LDQ       2,2                 GET OLD UPPER BOUND
     026073  000002 7560 11         9444          STQ       2,1                 AND STORE IN NEW UPPER BOUND
     026074  000003 2360 12         9445          LDQ       3,2                 GET OLD STRIDE
     026075  000003 7560 11         9446          STQ       3,1                 AND STORE IN NEW STRIDE
```

```
           R                                                     PASS 3

     026076  000004 2360 12        9447        LDQ     4,2             GET OLD STATE
     026077  000004 7560 11        9448        STQ     4,1             AND STORE IN NEW STATE
     026100  000001 1630 03        9449        SBX3    1,DU            DECREMENT NUMBER OF DIMENSIONS LEFT TO DO
     026101  026070 6010 00        9450        TNZ     RGN5            TRANSFER IF MORE TO DO
     026102  000000 7100 00        9451 RGNX   TRA     **              AND RETURN
     026103  000000 000000         9452 RGNS   ZERO
     026104  000000 000000         9453 RGNF   ZERO
 END OF BINARY CARD 00000463
     026105  000000 000000         9454 RGND   ZERO
     026106  000000 000000         9455 RGNTP  ZERO
     026107  000000 000000         9456 RGNT   ZERO
     026110  000000 027235         9457 RGNP   ZERO    0,T$MS
     026111  000000 000000         9458 RGNA   ZERO
     026112  026110 026113         9459 RGNTG  ZERO    RGNP,*+1
     026113  027216 000000         9460        ZERO    T$WPTR
     026114  027216 000000         9461        ZERO    T$WPTR
     026115  777775 777776         9462        ZERO    -2-1,-2
     026116  026122 7410 00        9463 PLGEN  STX1    PLGNP           STORE POINTER TO MSCW WORD
     026117  027211 7420 00        9464        STX2    REQ             STORE NUMBER OF WORDS REQUESTED
     026120  026122 2360 00        9465        LDQ     PLGNP           GET POINTER TO MEMORY TO MARK IN Q
     026121  026133 7100 00        9466        TRA     LOC             GO ALLOCATE MEMORY
     026122  000000 026123         9467 PLGNP  ZERO    0,**1
     026123  027215 000000         9468        ZERO    T$PTR
     026124  027217 000000         9469        ZERO    T$OCT
     026125  027215 000000         9470        ZERO    T$PTR
     026126  027215 000000         9471        ZERO    T$PTR
     026127  777773 777774         9472        ZERO    -4-1,-4
     026130  026150 7410 00        9473 LGEN   STX1    LGENP           STORE POINTER TO MARK STACK WORD
     026131  027211 7420 00        9474        STX2    REQ             STORE NUMBER OF WORDS REQUESTED
     026132  026150 2360 00        9475        LDQ     LGENP           GET POINTER TO MEMORY TO MARK IN Q
 END OF BINARY CARD 00000464
     026133  026225 7400 00        9476 LOC    STX0    ZERX            SAVE RETURN
     026134  000000 6350 16        9477        EAA     0,S             GET STACK POINTER IN AU
     026135  027211 0660 00        9478        ADX     S,REQ           INCREMENT STACK POINTER BY AMOUNT OF REQUEST
     026136  027204 1060 00        9479        CMPX    S,SMAX          SEE IF STACK POINTER OVERLAPS HEAP
     026137  026213 6020 00        9480        TNC     ZER             NO - POINTER IN AU - GO ZERO IT OUT
     026140  027211 1660 00        9481        SBX     S,REQ           RESTORE STACK POINTER
     026141  026537 7560 00        9482        STQ     GP              AND STORE
     026142  026226 7000 00        9483        TSX0    G               GARBAGE COLLECT AND ALLOCATE MEMORY
     026143  000000 6350 16        9484        EAA     0,S             GET STACK POINTER IN AU
     026144  027211 0660 00        9485        ADX     S,REQ           INCREMENT STACK POINTER BY AMOUNT OF REQUEST
     026145  027204 1060 00        9486        CMPX    S,SMAX          SEE IF STACK POINTER OVERLAPS HEAP
     026146  777777 6030 00        9487        TRC     $ERROR          YES - GARBAGE COLLECTOR ERROR
     026147  026213 7100 00        9488        TRA     ZER             GO ZERO OUT MEMORY
     026150  000000 027240         9489 LGENP  ZERO    0,T$PTRT
     026151  000000 000000         9490 LOCP   ZERO
     026152  000000 000000         9491 LOCD   ZERO
     026153  026151 026154         9492 LOCT   ZERO    LOCP,*+1
     026154  027216 000000         9493        ZERO    T$WPTR
     026155  027216 000000         9494        ZERO    T$WPTR
```

```
          R                                                 PASS 3

     026156  777775 777776          9495         ZERO      -2-1,-2
     026157  026204 7410 00         9496 HGEN    STX1      HGENP              STORE POINTER TO MARK STACK WORD
     026160  027211 7420 00         9497         STX2      REQ                STORE NUMBER OF WORDS REQUESTED
  END OF BINARY CARD 00000465
     026161  026204 2360 00         9498         LDQ       HGENP              GET POINTER TO MEMORY TO MARK IN Q
     026162  026225 7400 00         9499 HEAP    STX0      ZERX               SAVE RETURN
     026163  027204 2200 00         9500         LDX0      SMAX               GET POINTER TO BASE OF HEAP IN XR - 0
     026164  027211 1600 00         9501         SBX0      REQ                DECREMENT BASE BY AMOUNT OF REQUEST
     026165  027204 7400 00         9502         STX0      SMAX               AND STORE DECREMENTED POINTER IN BASE POINTER
     026166  000000 6350 10         9503         EAA       0,0                GET POINTER TO NEW MEMORY IN AU
     026167  027204 1060 00         9504         CMPX      S,SMAX             SEE IF STACK POINTER OVERLAPS HEAP
     026170  026213 6020 00         9505         TNC       ZER                NO OVERLAP - GO ZERO OUT MEMORY
     026171  027211 0600 00         9506         ADX0      REQ                RESTORE BASE OF HEAP POINTER
     026172  027204 7400 00         9507         STX0      SMAX               AND RESTORE IN MEMORY
     026173  026537 7560 00         9508         STQ       GP                 AND STORE
     026174  026226 7000 00         9509         TSX0      G                  GARBAGE COLLECT AND ALLOCATE MEMORY
     026175  027204 2200 00         9510         LDX0      SMAX               GET POINTER TO BASE OF HEAP IN XR - 0
     026176  027211 1600 00         9511         SBX0      REQ                DECREMEQT BASE BY AMOUNT OF REQUEST
     026177  027204 7400 00         9512         STX0      SMAX               AND STORE DECREMENTED POINTER IN BASE POINTER
     026200  000000 6350 10         9513         EAA       0,0                GET POINTER TO NEW MEMORY IN AU
     026201  027204 1060 00         9514         CMPX      S,SMAX             SEE IF STACK POINTER OVERLAPS HEAP
     026202  777777 6030 00         9515         TRC       $ERROR             YES - GARBAGE COLLECTOR ERROR
     026203  026213 7100 00         9516         TRA       ZER                GO ZERO OUT MEMORY
     026204  000000 027240          9517 HGENP   ZERO      0,T$PTRT
     026205  000000 000000          9518 HEAPP   ZERO
     026206  000000 000000          9519 HEAPD   ZERO
  END OF BINARY CARD 00000466
     026207  026205 026210          9520 HEAPT   ZERO      HEAPP,*+1
     026210  027216 000000          9521         ZERO      T$WPTR
     026211  027216 000000          9522         ZERO      T$WPTR
     026212  777775 777776          9523         ZERO      -2-1,-2
     026213  000000 6210 01         9524 ZER     EAX1      0,AU               GET POINTER TO START OF MEMORY IN XR - 1
     026214  027211 2360 00         9525         LDQ       REQ                GET LENGTH OF MEMORY IN QU
     026215  000010 7720 00         9526         QRL       8                  POSITION COUNT FOR REPEAT
     026216  777777 6000 00         9527         TZE       $ERROR             LENGTH IS ZERO - ERROR SOMEWHERE
     026217  000000 6200 06         9528         EAX0      0,QL               GET INITIAL COUNT IN XR - 0
     026220  000001 1760 07         9529         SBQ       1,DL               MAKE LOOP TERMINATE PROPERLY
     026221  000000 5202 01         9530 ZER1    RPTX      ,1                 ZERO OUT
     026222  000000 4500 11         9531         STZ       0,1                WORDS IN MEMORY JUST ALLOCATED
     026223  000001 1760 03         9532         SBQ       1,DU               ANY MORE TO DO
     026224  026221 6050 00         9533         TPL       ZER1               TRANSFER FOR ANOTHER 256 WORDS TO ZERO
     026225  000000 7100 00         9534 ZERX    TRA       **                 RETURN TO CALLER
     026226  027214 7400 00         9535 G       STX0      GX                 SAVE RETURN
     026227  026536 4500 00         9536         STZ       LF                 INITIALIZE FLAG INDICATING MARKING PASS
     026230  000000 6230 00         9537         EAX3      0                  INITIALIZE ARRAY CHAIN POINTER
     026231  026541 2360 00         9538 L1      LDQ       GMARK              GET POINTER TO MEMORY TO MARK
     026232  026545 4500 00         9539         STZ       A                  INITIALIZE BACK POINTER
     026233  027200 2210 00         9540         LDX1      BITT               GET POINTER TO START OF BIT TABLE IN XR - 1
     026234  000001 3350 07         9541         LCA       1,DL               GET ALL ONES IN A REGISTER
  END OF BINARY CARD 00000467
```

```
           R                                                 PASS 3

    026235  000000 7550 11        9542         STA      0,1             AND STORE AT START OF BIT TABLE
    026236  000001 0610 03        9543         ADX1     1,DU            STEP TO NEXT WORD IN BIT TABLE
    026237  000000 4500 11        9544 L1,5    STZ      0,1             ZERO OUT NEXT WORD IN BIT TABLE
    026240  000001 0610 03        9545         ADX1     1,DU            STEP TO NEXT WORD IN BIT TABLE
    026241  027207 1010 00        9546         CMPX1    MTOP            SEE IF AT END OF BIT TABLE
    026242  026237 6010 00        9547         TNZ      L1,5            TRANSFER IF MORE BIT TABLE
    026243  026536 2340 00        9548 L2      SZN      LF              SEE IF SECOND TIME THROUGH MARKING ROUTINE
    026244  026275 6000 00        9549         TZE      L2,4            TRANSFER IF FIRST TIME
    026245  026546 7560 00        9550         STQ      B               SAVE CURRENT POINTER IN MEMORY
    026246  027203 2210 00        9551         LDX1     HP              GET POINTER TO RELOCATION TABLE IN XR - 1
    026247  000000 6220 02        9552         EAX2     0,QU            GET OLD VALUE OF CURRENT POINTER IN XR - 2
    026250  026750 2350 00        9553         LDA      CN              GET NUMBER OF WORDS IN RELOCATION TABLE IN AL
    026251  000012 7350 00        9554         ALS      18-8            POSITION COUNT FOR REPEAT
    026252  000002 6200 05        9555         EAX0     2,AL            GET INITIAL COUNT AND TNC IN XR - 0
    026253  000001 1750 07        9556         SBA      1,DL            MAKE LOOP END TEST WORK PROPERLY
    026254  000002 5202 01        9557 L2,1    RPTX     ,1,TNC          SEARCH FOR
    026255  000000 1020 11        9558         CMPX2    0,1             OLD ADDRESS IN RELOCATION TABLE
    026256  026261 6020 00        9559         TNC      L2,2            TRANSFER IF PROPER ENTRY FOUND
    026257  000001 1750 03        9560         SBA      1,DU            SEE IF ANOTHER 256 WORDS TO SEARCH
    026260  026254 6050 00        9561         TPL      L2,1            TRANSFER IF YES
    026261  777776 7220 11        9562 L2,2    LXL2     -2,1            GET NEW ADDRESS IN XR - 2
    026262  777776 1620 11        9563         SBX2     -2,1            SUBTRACT OLD ADDRESS TO GET OFFSET
 END OF BINARY CARD 00000468
    026263  026546 0420 00        9564         ASX2     B               UPDATE CURRENT POINTER TO NEW VALUE
    026264  777777 7220 11        9565         LXL2     -1,1            GET STARTING ADDRESS OF NEXT MOVED BLOCK
    026265  026546 1020 00        9566         CMPX2    B               SEE IF RELOCATED POINTER IS IN NEXT BLOCK
    026266  026274 6030 00        9567         TRC      L2,3            TRANSFER IF NOT - OK
    026267  026546 7420 00        9568         STX2     B               SET POINTER TO START OF NEXT BLOCK
    026270  027203 2220 00        9569         LDX2     HP              GET POINTER TO START OF HEAP BEFORE MOVE UP
    026271  026546 1020 00        9570         CMPX2    B               SEE IF POINTER WAS INTO THE HEAP
    026272  026274 6030 00        9571         TRC      L2,3            TRANSFER IF NOT - OK
    026273  026546 7420 00        9572         STX2     B               SET POINTER TO JUST ABOVE STACK
    026274  026546 2360 00        9573 L2,3    LDQ      B               AND RESTORE POINTER IN Q REGISTER
    026275  000000 6220 06        9574 L2,4    EAX2     0,QL            SEE IF TYPE POINTER IS ZERO
    026276  026407 6000 00        9575         TZE      L11,5           TRANSFER IF POINTER HAS NO TYPE POINTER
    026277  000000 2220 06        9576         LDX2     0,QL            GET PATTERN TYPE IN XR - 2
    026300  026407 6000 00        9577         TZE      L11,5           TRANSFER IF POINTER HAS VACUOUS TEMPLATE
    026301  027216 1020 03        9578         CMPX2    TSWPTR,DU       IS TYPE A WORKING POINTER THAT IS NOT MARKED
    026302  026374 6000 00        9579         TZE      L6              TRANSFER IF YES TO MARK STRUCTURE IT POINTS TO
    026303  000000 6350 02        9580         EAA      0,QU            GET POINTER TO NEXT WORD TO MARK IN AU
    026304  026407 6000 00        9581         TZE      L11,5           TRANSFER #### IF NIL
    026305  027210 1750 00        9582         SBA      MBASE           SUBTRACT BASE OF COLLECTABLE MEMORY
    026306  777777 6040 00        9583         TMI      $ERROR          TRANSFER IF POINTER IS TOO SMALL
    026307  000015 7750 00        9584         ALR      18-5            GET BIT WORD IN AL AND BIT IN AU
    026310  000001 6210 05        9585         EAX1     1,AL            GET WORD ADDRESS IN XR - 1
 END OF BINARY CARD 00000469
    026311  027200 0610 00        9586         ADX1     BITT            ADD ADDRESS OF BASE OF BIT TABLE
    026312  000015 7710 00        9587         ARL      18-5            GET BIT POSITION IN AU
    026313  000000 6200 01        9588         EAX0     0,AU            GET BIT POSITION IN XR - 0
    026314  020000 2350 03        9589         LDA      =O020000,DU     GET SINGLE BIT IN A REGISTER
```

```
           R                                              PASS 3

     026315  000000 7710 10          ARL     0,0              SHIFT BIT TO BIT POSITION
     026316  027222 1020 03          CMPX2   T$SKIP,DU        IS PATTERN A SKIP TYPE
     026317  026332 6010 00          TNZ     L2,5             NO - TRANSFER
     026320  000000 7200 06          LXL0    0,QL             GET NUMBER OF WORDS TO MARK IN XR - 3
     026321  000000 2550 11    L2,7  ORSA    0,1              MARK A WORD
     026322  000001 1600 03          SBX0    1,DU             DECREMENT NUMBER OF WORDS LEFT TO MARK
     026323  026401 6000 00          TZE     L8               TRANSFER IF ALL MARKED
     026324  000001 0760 03          ADQ     1,DU             STEP TO NEXT WORD TO MARK
     026325  000001 7710 00          ARL     1                SHIFT MARKING BIT TO NEXT WORD
     026326  026321 6010 00          TNZ     L2,7             TRANSFER IF STILL IN SAME WORD
     026327  020000 2350 03          LDA     =0020000,DU      GET BIT FOR FIRST BIT IN WORD
     026330  000001 0610 03          ADX1    1,DU             AND STEP TO NEXT WORD IN BIT TABLE
     026331  026321 7100 00          TRA     L2,7             AND LOOP
     026332  000000 3150 11    L2,5  CANA    0,1              SEE IF BIT IS ALREADY SET
     026333  026401 6010 00          TNZ     L8               TRANSFER IF QU POINTS TO MARKED WORD
     026334  000000 2550 11    L3    ORSA    0,1              SET BIT IN BIT TABLE
     026335  027217 1020 03          CMPX2   T$OCT,DU         SEE IF MARKED WORD REFERS TO ANYTHING
     026336  026401 6000 00          TZE     L8               TRANSFER IF WORD DOESN'T REFER TO ANYTHING
  END OF BINARY CARD 00000470
     026337  027220 1020 03          CMPX2   T$ULEN,DU        SEE IF POINTER TO UNION
     026340  026352 6010 00          TNZ     L5,3             TRANSFER IF NOT POINTER TO UNITED MODE
     026341  026545 7200 00          LXL0    A                GET TYPE OF BACK POINTER IN XR - 0
     026342  026545 7520 07          STCQ    A,07             STORE TYPE OF CURRENT POINTER IN BACK POINTER
     026343  000022 7370 00          LLS     18               SAVE CURRENT POINTER IN AL
     026344  000000 2360 05          LDQ     0,AL             GET TYPE OF CURRENT WORD IN QL
     026345  000022 7360 00          QLS     18               MOVE TO QU
     026346  000022 7330 00          LRS     18               RESTORE CURRENT POINTER WITH NEW TYPE
     026347  000000 4400 02          SXL0    0,QU             STORE BACK POINTER TYPE IN CURRENT WORD
     026350  000001 0760 03          ADQ     1,DU             MAKE CURRENT POINTER POINT TO VALUE OF UNION
     026351  026275 7100 00          TRA     L2,4             AND GO MARK UNITED VALUE
     026352  027221 1020 03    L5,3  CMPX2   T$ROW,DU         IS PATTERN A ROW PATTERN
     026353  026372 6010 00          TNZ     L5,8             NO - TRANSFER
     026354  026536 2340 00          SZN     LF               IS THIS FIRST TIME THROUGH MARKING ROUTINE
     026355  026374 6010 00          TNZ     L6               TRANSFER IF SECOND TIME THROUGH
     026356  000000 7200 06          LXL0    0,QL             GET NUMBER OF DIMENSIONS IN ROW VALUE IN XR - 0
     026357  000004 6210 02          EAX1    4,QU             GET ADDRESS OF FIRST QUAD IN XR - 1
     026360  000000 2350 11    L5,7  LDA     0,1              GET STATES IN QUAD
     026361  000043 7750 00          ALR     35               GET LOWER STATE IN SIGN POSITION
     026362  000021 7310 00          ARS     17               SET UPPER HALF OF A REGISTER TO LOWER STATE
     026363  000043 7750 00          ALR     35               GET BOTH STATES IN TOP TWO BITS OF A REGISTER
     026364  600000 3750 03          ANA     =0600000,DU      MASK OUT ALL OTHER BITS
  END OF BINARY CARD 00000471
     026365  000000 7550 11          STA     0,1              AND RESTORE IN STATE WORD
     026366  000004 0610 03          ADX1    4,DU             STEP TO NEXT QUAD
     026367  000001 1600 03          SBX0    1,DU             DECREMENT NUMBER OF QUADS LEFT TO DO
     026370  026360 6010 00          TNZ     L5,7             TRANSFER IF MORE TO DO
     026371  026374 7100 00          TRA     L6               GO PROCESS FIRST ELEMENT OF ARRAY
     026372  027215 1020 03    L5,8  CMPX2   T$PTR,DU         IS PATTERN TYPE A POINTER TYPE
     026373  777777 6010 00          TNZ     $ERROR           NO - BAD PATTERN POINTER
     026374  026545 2350 00    L6    LDA     A                GET BACK POINTER IN A REGISTER
```

```
                                                    PASS 3

      026375  026545 7560 00       9638           STQ     A              STORE CURRENT POINTER AS BACK POINTER
      026376  000000 2360 02       9639           LDQ     0,QU           GET NEW POINTER AS CURRENT POINTER
      026377  026545 7550 51       9640           STA     A,I            STORE NEW HEAD OF BACK POINTER CHAIN
      026400  026243 7100 00       9641 L7        TRA     L2             GO CONSIDER NEW STRUCTURE
      026401  026547 0760 00       9642 L8        ADQ     STMSK          STEP BOTH ADDRESS AND TAG OF CURRENT POINTER
      026402  000000 2340 06       9643 L9        SZN     0,QL           SEE IF AT END OF PATTERN
      026403  026275 6050 00       9644           TPL     L2,4           TRANSFER IF NOT AT END OF PATTERN
      026404  026545 2340 00       9645 L10       SZN     A              SEE IF BACK POINTER IS ZERO
      026405  026500 6000 00       9646           TZE     LA1            TRANSFER IF MARKING IS COMPLETE
      026406  000000 0760 06       9647 L11       ADQ     0,QL           RESTORE POINTER TO BEGINNING OF STRUCTURE
      026407  026545 7200 00       9648 L11.5     LXL0    A              GET TYPE POINTER IN BACK POINTER IN XR - 0
      026410  000000 2210 10       9649           LDX1    0,0            GET TYPE IN XR - 1
      026411  027220 1010 03       9650           CMPX1   T$ULEN,DU      IS IT A UNITED TYPE
      026412  026424 6010 00       9651           TNZ     L12            TRANSFER IF NOT UNITED TYPE
 END OF BINARY CARD 00000472
      026413  000001 1760 03       9652           SBQ     1,DU           MAKE CURRENT POINTER POINT TO HEAD OF UNION VALUE
      026414  000000 2350 02       9653           LDA     0,QU           GET CURRENT VALUE TYPE IN AL
      026415  026545 7510 07       9654           STCA    A,07           STORE TYPE IN BACK POINTER TYPE
      026416  000000 6210 06       9655           EAX1    0,QL           GET TYPE OF CURRENT POINTER IN XR - 1
      026417  000000 4410 02       9656           SXL1    0,QU           AND STORE TYPE IN CURRENT VALUE
      026420  000022 7370 00       9657           LLS     18             SAVE CURRENT POINTER IN AL
      026421  000000 6360 10       9658           EAQ     0,0            GET TYPE FROM BACK POINTER IN QU
      026422  000022 7330 00       9659           LRS     18             GET CURRENT POINTER AND TYPE IN Q REGISTER
      026423  026401 7100 00       9660           TRA     L8             GO TO CONTINUE MARKING STRUCTURE
      026424  026545 2350 51       9661 L12       LDA     A,I            GET OLD BACK POINTER IN A REGISTER
      026425  026545 7560 51       9662           STQ     A,I            RESTORE OLD FOREWARD POINTER
      026426  026545 2360 00       9663           LDQ     A              MAKE BACK POINTER CURRENT POINTER
      026427  026545 7550 00       9664           STA     A              SET NEW HEAD OF BACK POINTER CHAIN
      026430  000000 2200 06       9665 L13       LDX0    0,QL           GET PATTERN FOR CURRENT WORD IN XR - 0
      026431  027221 1000 03       9666           CMPX0   T$ROW,DU       IS IT A ROW TYPE PATTERN
      026432  026401 6010 00       9667           TNZ     L8             NO - STEP TO NEXT WORD TO MARK
      026433  000000 7200 06       9668           LXL0    0,QL           GET DIMENSION IN XR - 0 OF ARRAY
      026434  000004 6210 02       9669           EAX1    4,QU           GET POINTER TO FIRST QUADRUPLE IN XR - 1
      026435  000001 1600 03       9670           SBX0    1,DU           DECREMENT NUMBER OF QUADRUPLES LEFT
      026436  026401 6040 00       9671           TMI     L8             TRANSFER IF NO QUADRUPLES LEFT
      026437  000000 0540 11       9672 LX        AOS     0,1            INCREMENT PLACE BY ONE
      026440  777777 2220 11       9673           LDX2    -1,1           GET STRIDE IN XR - 2
 END OF BINARY CARD 00000473
      026441  000000 0420 02       9674           ASX2    0,QU           INCREMENT CURRENT POINTER BY STRIDE
      026442  777776 2350 11       9675           LDA     -2,1           GET UPPER BOUND IN A REGISTER
      026443  777775 1750 11       9676           SBA     -3,1           SUBTRACT LOWER BOUND
      026444  000000 1750 11       9677           SBA     0,1            SUBTRACT CURRENT PLACE
      026445  000002 7350 00       9678           ALS     2              SHIFT OUT STATE BITS
      026446  026374 6050 00       9679           TPL     L6             TRANSFER IF PLACE IS IN BOUNDS
      026447  026546 7560 00       9680           STQ     B              SAVE FOREWARD POINTER
      026450  000000 3350 11       9681           LCA     0,1            GET -(U-L+1) IN Q REGISTER
      026451  777777 4020 11       9682           MPY     -1,1           MULTIPLY BY STRIDE
      026452  600000 2350 03       9683           LDA     =0600000,DU    GET MASK FOR STATE BITS
      026453  000000 3550 11       9684           ANSA    0,1            ZERO OUT PLACE EXCEPT FOR STATE BITS
      026454  026546 0560 51       9685           ASQ     B,I            RESTORE ARRAY POINTER TO START OF COLUMN
```

```
          R                                              PASS 3

     026455  026546 2360 00          9686         LDQ      B                   RESTORE CURRENT POINTER IN Q REGISTER
     026456  000004 0610 03          9687         ADX1     4,DU                STEP TO NEXT QUADRUPLE
     026457  000001 1600 03          9688         SBX0     1,DU                DECREMENT NUMBER OF QUADRUPLES LEFT
     026460  026437 6050 00          9689         TPL      LX                  TRANSFER IF MORE QUADRUPLES
     026461  026536 2340 00          9690         SZN      LF                  SEE IF FIRST TIME MARKING
     026462  026466 6010 00          9691         TNZ      LX1                 TRANSFER IF SECOND TIME THROUGH
     026463  000004 4430 02          9692         SXL3     4,QU                STORE LINK TO THIS DESCRIPTOR
     026464  000004 6230 02          9693         EAX3     4,QU                AND GET NEW HEAD CHAIN POINTER IN XR - 3
     026465  026401 7100 00          9694         TRA      L8                  GO CONTINUE MARKING
     026466  000000 7200 06          9695 LX1     LXL0     0,QL                GET DIMENSION OF ARRAY IN XR - 0
   END OF BINARY CARD 00000474
     026467  000004 6210 02          9696         EAX1     4,QU                GET POINTER TO FIRST QUAD IN XR - 1
     026470  000001 1610 03          9697 LX2     SBX1     1,DU                DECREMENT NUMBER OF QUADS LEFT
     026471  026401 6040 00          9698         TMI      L8                  TRANSFER IF NO MORE QUADS
     026472  000000 2350 11          9699         LDA      0,1                 GET STATES OF QUAD IN A REGISTER
     026473  000042 7310 00          9700         ARS      36-2                RESTORE IN PROPER PLACE IN STATE WORD
     026474  026547 3750 00          9701         ANA      STMSK               ZERO OUT OTHER BITS WITH 000001000001 MASK

     026475  000000 7550 11          9702         STA      0,1                 AND RESTORE IN STATE WORD IN QUAD
     026476  000004 0610 03          9703         ADX1     4,DU                STEP TO NEXT QUAD IN ARRAY
     026477  026470 7100 00          9704         TRA      LX2                 AND LOOP
     026500  026536 2340 00          9705 LA1     SZN      LF                  IS THIS FIRST TIME THROUGH MARKING ROUTINE
     026501  027212 6010 00          9706         TNZ      G1                  TRANSFER TO CLEANUP IF SECOND TIME THROUGH
     026502  000000 6230 13          9707         EAX3     0,3                 SEE IF ANY ARRAYS TO MARK
     026503  026550 6000 00          9708 LA2     TZE      C,5                 TRANSFER IF NO MORE ARRAYS TO MARK
     026504  000002 2360 13          9709         LDQ      2,3                 GET UPPER BOUND OF ARRAY
     026505  000001 1760 13          9710         SBQ      1,3                 SUBTRACT LOWER BOUND OF ARRAY
     026506  000001 0760 07          9711         ADQ      1,DL                GET NUMBER OF VALID SUBSCRIPTS IN ARRAY
     026507  000003 4020 13          9712         MPY      3,3                 MULTIPLY BY STRIDE
     026510  026534 6000 00          9713         TZE      LA4                 TRANSFER IF ARRAY IS VACUOUS
     026511  000000 2350 13          9714         LDA      0,3                 GET POINTER TO FIRST ELEMENT OF ARRAY IN AU
     026512  000000 6350 01          9715         EAA      0,AU                ZERO OUT AL
     026513  027210 1750 00          9716         SBA      MBASE               SUBTRACT BASE OF MARKABLE MEMORY
     026514  777777 6040 00          9717         TMI      $ERROR              POINTER IS OUT OF RANGE
   END OF BINARY CARD 00000475
     026515  000015 7750 00          9718         ALR      18-5                GET BIT TABLE WORD IN AL
     026516  000000 6210 05          9719         EAX1     0,AL                GET WORD ADDRESS IN XR - 1
     026517  027200 0610 00          9720         ADX1     BITT                MAKE ADDRESS ABSOLUTE
     026520  000015 7710 00          9721         ARL      18-5                GET BIT POSITION IN AU
     026521  000000 6200 01          9722         EAX0     0,AU                AND PUT IN XR - 0
     026522  020000 2350 03          9723         LDA      =0020000,DU         GET A BIT 32 BITS FROM RIGHT END OF WORD
     026523  000000 7710 10          9724         ARL      0,0                 SHIFT BY BIT POSITION
     026524  000000 2550 11          9725 LA3     ORSA     0,1                 AND OR INTO BIT TABLE
     026525  000001 1760 03          9726         SBQ      1,DU                DECREMENT NUMBER OF WORDS LEFT TO MARK
     026526  026534 6000 00          9727         TZE      LA4                 TRANSFER IF NO MORE WORDS TO MARK
     026527  000001 7710 00          9728         ARL      1                   SHIFT TO NEXT BIT
     026530  026524 6010 00          9729         TNZ      LA3                 TRANSFER IF STILL IN SAME WORD
     026531  020000 2350 03          9730         LDA      =0020000,DU         GET BIT AT START OF NEW WORD
     026532  000001 0610 03          9731         ADX1     1,DU                AND STEP TO NEXT WORD IN BIT TABLE
     026533  026524 7100 00          9732         TRA      LA3                 GO CONTINUE MARKING
     026534  000004 7230 13          9733 LA4     LXL3     4,3                 GET POINTER TO NEXT ARRAY TO MARK
```

```
              R                                                  PASS 3

        026535  026503 7100 00        9734          TRA      LA2               AND LOOP
        026536  000000 000000         9735 LF       ZERO
        026537  000000 000000         9736 GP       ZERO
        026540  000000 000000         9737 GD       ZERO
        026541  026537 026542         9738 GMARK    ZERO     GP,*+1
        026542  027216 000000         9739          ZERO     T$WPTR
    END OF BINARY CARD 00000476
        026543  027216 000000         9740          ZERO     T$WPTR
        026544  777775 777776         9741          ZERO     -2-1,-2
        026545  000000 000000         9742 A        ZERO
        026546  000000 000000         9743 B        ZERO
        026547  000001 000001         9744 STMSK    ZERO     1,1
        026550  004000 6340 07        9745 C,5      LDI      =04000,DL         MASK OUT OVERFLOW FAULTS
        026551  026753 4500 00        9746          STZ      CF                INITIALIZE FIRST TIME FLAG
        026552  026750 4500 00        9747          STZ      CN                ZERO LENGTH OF RELOCATION TABLE
        026553  027200 2210 00        9748          LDX1     BITT              GET POINTER TO START OF BIT TABLE
        026554  000001 1610 03        9749          SBX1     1,DU              ENTER LOOP PROPERLY
        026555  026602 7100 00        9750          TRA      C4                AND GO COMPACT
        026556  026761 2210 00        9751 C1       LDX1     CP2               RESTORE POINTER TO PLACE IN BIT TABLE
        026557  026760 4110 00        9752          LDE      CE2               RESTORE E REGISTER
        026560  026754 2370 00        9753          LDAQ     CAQ               RESTORE AQ REGISTER
        026561  026573 7100 00        9754          TRA      C3                AND GO ENTER BIT SCAN LOOP
        026562  000002 0610 03        9755 C2       ADX1     2,DU              STEP TO NEXT WORD PLUS ONE IN BIT TABLE
        026563  027207 1010 00        9756          CMPX1    MTOP              IS THIS THE TOP WORD IN THE BIT TABLE
        026564  026763 6000 00        9757          TZE      A1                TRANSFER IF YES - DONE COMPACTING
        026565  000001 1610 03        9758          SBX1     1,DU              MAKE XR - 1 POINT TO NEXT WORD IN BIT TABLE
        026566  000000 2350 11        9759          LDA      0,1               GET BIT TABLE WORD IN A REGISTER
        026567  000001 2360 11        9760          LDQ      1,1               GET FOLLOWING WORD IN Q REGISTER
        026570  000004 7360 00        9761          QLS      4                 MOVE BITS IN Q NEXT TO BITS IN A
    END OF BINARY CARD 00000477
        026571  000004 7370 00        9762          LLS      4                 LEFT JUSTIFY BITS IN AQ REGISTER
        026572  476000 4110 03        9763          LDE      =-97B25,DU        GET E REGISTER SET TO UNDERFLOW IN 32 BITS
        026573  000000 5730 00        9764 C3       FNO                        SHIFT AQ LEFT TO NEXT BIT TRANSITION
        026574  026562 6150 00        9765          TEU      C2                TRANSFER IF NO TRANSITION IN WORD
        026575  026562 6000 00        9766          TZE      C2                TRANSFER IF NO TRANSITION IN WORD BECAUSE ZERO
        026576  026756 4560 00        9767          STE      CE1               STORE BIT POINTER FOR ZERO/ONE TRANSITION
        026577  026757 7410 00        9768          STX1     CP1               STORE WORD POINTER FOR ZERO/ONE TRANSITION
        026600  400000 6750 03        9769          ERA      =0400000,DU       FLIP SIGN BIT TO ELIMINATE TRANSITION
        026601  026610 7100 00        9770          TRA      C5                AND GO ENTER BIT SCAN LOOP
        026602  000001 0610 03        9771 C4       ADX1     1,DU              STEP TO NEXT WORD IN BIT TABLE
        026603  000000 2350 11        9772          LDA      0,1               GET BIT TABLE WORD IN A REGISTER
        026604  000001 2360 11        9773          LDQ      1,1               GET FOLLOWING WORD IN Q REGISTER
        026605  000004 7360 00        9774          QLS      4                 MOVE BITS IN Q NEXT TO BITS IN A
        026606  000004 7370 00        9775          LLS      4                 LEFT JUSTIFY BITS IN AQ REGISTER
        026607  476000 4110 03        9776          LDE      =-97B25,DU        GET E REGISTER SET TO UNDERFLOW IN 32 BITS
        026610  000000 5730 00        9777 C5       FNO                        SHIFT AQ LEFT TO NEXT BIT TRANSITION
        026611  026602 6150 00        9778          TEU      C4                TRANSFER IF NO TRANSITION IN WORD
        026612  026602 6000 00        9779          TZE      C4                TRANSFER IF NO TRANSITION IN WORD BECAUSE ZERO
        026613  026760 4560 00        9780          STE      CE2               STORE BIT POINTER FOR ONE/ZERO TRANSITION
        026614  026761 7410 00        9781          STX1     CP2               STORE WORD POINTER FOR ONE/ZERO TRANSITION
```

```
              R                                                         PASS 3

     026615  400000 6750 03         9782          ERA       =0400000,DU       FLIP SIGN BIT TO ELIMINATE TRANSITION
     026616  026754 7570 00         9783          STAQ      CAQ               SAVE CURRENT AQ REGISTER
  END OF BINARY CARD 00000478
     026617  026761 2360 00         9784          LDQ       CP2               GET WORD POINTER FOR ONE/ZERO TRANSITION
     026620  027200 1760 00         9785          SBQ       BITT              SUBTRACT ADDRESS OF BASE OF BIT TABLE
     026621  000005 7360 00         9786          QLS       5                 MULTIPLY BY 32
     026622  777600 0760 03         9787          ADQ       -97-32+1,DU       ADD CORRECTION FACTOR
     026623  026747 7560 00         9788          STQ       CC                AND STORE IN END OF NEEDED AREA POINTER
     026624  026760 2360 00         9789          LDQ       CE2               GET BIT POINTER FOR ONE/ZERO TRANSITION
     026625  000012 7320 00         9790          QRS       18-8              MOVE TO QU
     026626  000000 5330 00         9791          NEGL                        AND NEGATE
     026627  027210 0760 00         9792          ADQ       MBASE             ADD BASE OF MARKABLE MEMORY
     026630  026753 2340 00         9793          SZN       CF                IS THIS THE FIRST TRANSITION FOUND
     026631  026644 6010 00         9794          TNZ       C5,5              NO - TRANSFER
     026632  026747 0760 00         9795          ADQ       CC                ADD TO GET END OF NEEDED AREA
     026633  026745 7560 00         9796          STQ       CA                YES - STORE AS END OF COMPACTED MEMORY
     026634  000000 6240 02         9797          EAX4      0,QU              AND PUT IN RELOCATION TABLE END POINTER
     026635  027204 1160 00         9798          CMPQ      SMAX              SEE IF STACK/HEAP BOUNDARY IS IN FIRST BLOCK
     026636  026640 6020 00         9799          TNC       C5,2              TRANSFER IF NOT
     026637  027204 2360 00         9800          LDQ       SMAX              GET OLD SMAX AS NEW SMAX
     026640  027202 7560 00         9801 C5.2     STQ       SHDIV             AND STORE GUESS FOR STACK/HEAP DIVISION
     026641  027203 7560 00         9802          STQ       HP                AND STORE AS BASE OF HEAP
     026642  026753 7500 00         9803          STC2      CF                SET FLAG INDICATING NOT FIRST TIME THROUGH
     026643  026556 7100 00         9804          TRA       C1                AND LOOP
     026644  026747 0560 00         9805 C5.5     ASQ       CC                AND ADD TO END OF NEEDED AREA POINTER
  END OF BINARY CARD 00000479
     026645  026757 2360 00         9806          LDQ       CP1               GET WORD POINTER FOR ZERO/ONE TRANSITION
     026646  027200 1760 00         9807          SBQ       BITY              SUBTRACT ADDRESS OF BASE OF BIT TABLE
     026647  000005 7360 00         9808          QLS       5                 MULTIPLY BY 32
     026650  777600 0760 03         9809          ADQ       -97-32+1,DU       ADD CORRECTION FACTOR
     026651  026746 7560 00         9810          STQ       CB                AND STORE IN START OF NEEDED AREA POINTER
     026652  026756 2360 00         9811          LDQ       CE1               GET BIT POINTER FOR ZERO/ONE TRANSITION
     026653  000012 7320 00         9812          QRS       18-8              MOVE TO QU
     026654  000000 5330 00         9813          NEGL                        AND NEGATE
     026655  027210 0760 00         9814          ADQ       MBASE             ADD BASE OF MARKABLE MEMORY
     026656  026746 0560 00         9815          ASQ       CB                AND ADD TO START OF NEEDED AREA POINTER
     026657  026746 2220 00         9816          LDX2      CB                GET POINTER TO ADDRESS OF BLOCK TO BE MOVED
     026660  000000 7420 14         9817          STX2      0,4               AND STORE IN TABLE OF OLD AND NEW ADDRESSES
     026661  026745 2210 00         9818          LDX1      CA                GET NEW ADDRESS OF BLOCK IN XR - 1
     026662  000000 4410 14         9819          SXL1      0,4               AND STORE IN TABLE OF OLD AND NEW ADDRESSES
     026663  026750 0540 00         9820          AOS       CN                INCREMENT LENGTH OF TABLE
     026664  000000 6230 12         9821          EAX3      0,2               SAVE ADDRESS OF BLOCK TO BE MOVED IN XR - 3
     026665  000000 2350 11         9822 C6       LDA       0,1               GET WORD AT NEW LOCATION IN A
     026666  000000 2360 12         9823          LDQ       0,2               GET WORD AT OLD LOCATION IN Q
     026667  000000 7560 11         9824          STQ       0,1               STORE OLD WORD IN NEW LOCATION
     026670  000000 7550 12         9825          STA       0,2               STORE NEW WORD IN OLD LOCATION
     026671  000001 0610 03         9826          ADX1      1,DU              STEP TO NEXT WORD
     026672  000001 0620 03         9827          ADX2      1,DU              STEP TO NEXT WORD
  END OF BINARY CARD 00000480
     026673  026747 1020 00         9828          CMPX2     CC                SEE IF ALL OF BLOCK HAS BEEN SWAPPED DOWN
```

```
            R                                                      PASS 3

        026674  026665 6010 00          9829           TNZ      C6                  TRANSFER IF MORE TO SWAP
        026675  027204 2200 00          9830           LDX0     SMAX                GET OLD STACK/HEAP DIVISION
        026676  026746 1000 00          9831           CMPX0    CB                  SEE IF BELOW CURRENT BLOCK THAT WAS MOVED
        026677  026706 6020 00          9832           TNC      C6,4                TRANSFER IF YES
        026700  027202 7420 00          9833           STX2     SHDIV               STORE GUESS FOR OLD STACK/HEAP DIVISION
        026701  027203 7410 00          9834           STX1     HP                  STORE GUESS FOR NEW STACK/HEAP DIVISION
        026702  026747 1600 00          9835           SBX0     CC                  SEE IF STACK HEAP DIVISION WAS JUST MOVED
        026703  026706 6030 00          9836           TRC      C6,4                TRANSFER IF NOT
        026704  027202 0400 00          9837           ASX0     SHDIV               ADJUST GUESS FOR OLD STACK/HEAP DIVISION
        026705  027203 0400 00          9838           ASX0     HP                  ADJUST GUESS FOR NEW STACK/HEAP DIVISION
        026706  026746 2220 00          9839 C6,4      LDX2     CB                  GET ADDRESS OF START OF BLOCK IN XR - 2
        026707  026745 1620 00          9840           SBX2     CA                  SUBTRACT MOVE ADDRESS TO GET DISTANCE MOVED
        026710  026762 7420 00          9841           STX2     CBA                 AND STORE DISTANCE MOVED
        026711  026747 2360 00          9842           LDQ      CC                  GET ADDRESS OF TOP OF OLD BLOCK
        026712  026745 1760 00          9843           SBQ      CA                  SUBTRACT MOVE ADDRESS
        026713  026762 5060 00          9844           DIV      CBA                 DIVIDE BY MOVE DISTANCE TO SEE HOW TABLE MOVED
        026714  026751 7550 00          9845           STA      CR                  SAVE REMAINDER
        026715  026762 4020 00          9846           MPY      CBA                 GET DISTANCE FIRST WORD OF TABLE WAS MOVED
        026716  026745 0760 00          9847           ADQ      CA                  GET ABSOLUTE ADDRESS OF NEW START OF TABLE
        026717  026745 7410 00          9848           STX1     CA                  STORE ADDRESS OF END OF MOVED BLOCK
        026720  000000 6240 11          9849           EAX4     0,1                 GET ADDRESS OF NEW BASE OF TABLE IN XR - 4
    END OF BINARY CARD 00000481
        026721  000000 6250 02          9850           EAX5     0,QU                GET ADDRESS OF START OF TABLE IN XR - 5
        026722  026750 7200 00          9851           LXL0     CN                  GET LENGTH OF TABLE IN XR - 0
        026723  026751 1600 00          9852           SBX0     CR                  SUBTRACT LENGTH OF FRAGMENT AFTER FIRST WORD
        026724  026730 6050 00          9853           TPL      C7                  TRANSFER IF TABLE IN TWO FRAGMENTS
        026725  026750 2350 00          9854           LDA      CN                  GET LENGTH OF TABLE IN AL
        026726  000022 7350 00          9855           ALS      18                  MOVE TO AU
        026727  026733 7100 00          9856           TRA      C8                  GO TO MOVE ENTIRE TABLE DOWN
        026730  026752 7400 00          9857 C7        STX0     CT                  STORE SECOND FRAGMENT LENGTH IN MEMORY
        026731  026751 2350 00          9858           LDA      CR                  GET NUMBER OF WORDS TO MOVE IN AU
        026732  026752 0640 00          9859           ADX4     CT                  GET POINTER IN XR - 4 WHERE TO MOVE TO
        026733  000010 7710 00          9860 C8        ARL      8                   POSITION WORD COUNT FOR REPEAT
        026734  026556 6000 00          9861           TZE      C1                  TRANSFER IF NOTHING TO MOVE
        026735  001400 6200 05          9862           EAX0     768,AL              GET LENGTH OF INITIAL MOVE IN XR - 0
        026736  000001 1750 07          9863           SBA      1,DL                MAKE LOOP COUNT PROPERLY
        026737  000000 5602 01          9864 C9        RPDX     ,1                  MOVE
        026740  000000 2360 15          9865           LDQ      0,5                 FROM FIRST TABLE FRAGMENT
        026741  000000 7560 14          9866           STQ      0,4                 TO OTHER FRAGMENT IF ANY
        026742  000001 1750 03          9867           SBA      1,DU                SEE IF ANOTHER 256 WORD BLOCK TO MOVE
        026743  026737 6050 00          9868           TPL      C9                  TRANSFER IF MORE TO MOVE
        026744  026556 7100 00          9869           TRA      C1                  GO TO CONTINUE COMPACTING
        026745  000000 000000           9870 CA        ZERO
        026746  000000 000000           9871 CB        ZERO
    END OF BINARY CARD 00000482
        026747  000000 000000           9872 CC        ZERO
        026750  000000 000000           9873 CN        ZERO
        026751  000000 000000           9874 CR        ZERO
        026752  000000 000000           9875 CT        ZERO
        026753  000000 000000           9876 CF        ZERO
```

```
            R                                                     PASS 3

                         026754        9877        EVEN
     026754  000000000000              9878 CAG    OCT        0,0
     026755  000000000000
     026756  000000 000000             9879 CE1    ZERO
     026757  000000 000000             9880 CP1    ZERO
     026760  000000 000000             9881 CE2    ZERO
     026761  000000 000000             9882 CP2    ZERO
     026762  000000 000000             9883 CBA    ZERO
     026763  000000 6340 07            9884 A1     LDI        0,DL          RESET OVERFLOW MASK
     026764  026745 2350 00            9885        LDA        CA            GET POINTER TO END OF COMPACTED STORAGE
     026765  027210 1750 00            9886        SBA        MBASE         SUBTRACT BASE OF COLLECTABLE MEMORY
     026766  027176 7550 00            9887        STA        AT            SAVE LENGTH OF COLLECTABLE MEMORY
     026767  000005 7710 00            9888        ARL        5             DIVIDE LENGTH BY 32
     026770  777777 0750 07            9889        ADA        -1,DL         ROUND UP
     026771  000002 6350 01            9890        EAA        2,AU          AND ADD 2 FOR HEAD AND TAIL OF BIT TABLE
     026772  027176 0550 00            9891        ASA        AT            AND ADD BIT TABLE LENGTH TO COLLECTABLE LENGTH
     026773  027211 2350 00            9892        LDA        REQ           GET NUMBER OF WORDS REQUESTED IN AU
     026774  000005 7710 00            9893        ARL        5             DIVIDE BY 32 FOR ASSOCIATED BIT TABLE LENGTH
   END OF BINARY CARD 00000483
     026775  777777 0750 07            9894        ADA        -1,DL         ROUND UP
     026776  027211 0750 00            9895        ADA        REQ           ADD NUMBER OF WORDS REQUESTED TO BIT TABLE LENGTH
     026777  000022 7710 00            9896        ARL        18            MOVE TOTAL LENGTH TO AL
     027000  026750 1150 00            9897        CMPA       CN            IS THIS LARGER THAN SPACE FOR RELOCATION TABLE
     027001  027003 6050 00            9898        TPL        A2            TRANSFER IF LARGER
     027002  026750 2350 00            9899        LDA        CN            GET LENGTH NEEDED FOR RELOCATION TABLE IN AL
     027003  000002 6250 05            9900 A2     EAX5       2,AL          GET MAXIMUM LENGTH IN XR - 5
     027004  027176 0650 00            9901        ADX5       AT            ADD LENGTH NEEDED FOR COMPACT MEMORY AND BITS
     027005  027210 0650 00            9902        ADX5       MBASE         ADD BASE ADDRESS
     027006  003777 0650 03            9903        ADX5       2047,DU       ADD 1K AND ROUND UP
     027007  776000 3650 03            9904        ANX5       -1024,DU      ROUND TO 1K BOUNDARY
     027010  027207 1050 00            9905        CMPX5      MTOP          IS THIS CURRENT MEMORY SIZE
     027011  027021 6000 00            9906        TZE        A3            YES - TRANSFER - NO REQUEST
     027012  002000 1650 03            9907        SBX5       1024,DU       GET SIZE NEEDED IN XR - 5
     027013  027207 1050 00            9908        CMPX5      MTOP          IS THIS CURRENT MEMORY SIZE
     027014  027021 6000 00            9909        TZE        A3            YES - TRANSFER - NO REQUEST
     027015  027207 7450 00            9910        STX5       MTOP          STORE NEW MEMORY SIZE
     027016  500006 0010 00            9911        MME        M$MREQ        AND REQUEST MORE OR LESS MEMORY
     027017  000000 6250 15            9912        EAX5       0,5           SEE IF WE GOT IT
     027020  777777 6010 00            9913        TNZ        $ERROR        NO - FATAL ERROR RAN OUT OF MEMORY
     027021  027207 2360 00            9914 A3     LDQ        MTOP          GET NEW TOP OF MEMORY IN QU
     027022  027210 1760 00            9915        SBQ        MBASE         SUBTRACT BASE OF COLLECTABLE MEMORY
   END OF BINARY CARD 00000484
     027023  000300 0760 03            9916        ADQ        5*32+32,DU    ADD FIVE BIT TABLE WORDS AND ROUND UP
     027024  000041 5060 07            9917        DIV        33,DL         GET NEW BIT TABLE LENGTH IN QU
     027025  777775 6210 02            9918        EAX1       -3,QU         PUT NEW BIT TABLE LENGTH IN XR - 1
     027026  027201 7410 00            9919        STX1       BITL          AND STORE IN BIT TABLE LENGTH
     027027  027207 2210 00            9920        LDX1       MTOP          GET ADDRESS OF TOP OF MEMORY
     027030  027201 1610 00            9921        SBX1       BITL          SUBTRACT BIT TABLE LENGTH TO GET TOP OF HEAP
     027031  027200 7410 00            9922        STX1       BITT          AND STORE AS BIT TABLE BASE
     027032  000003 1610 03            9923        SBX1       3,DU          GET NEW TOP OF HEAP ADDRESS IN XR - 1
```

```
         R                                              PASS 3

    027033  026745 2220 00       9924        LDX2     CA              GET ADDRESS OF TOP OF COMPACTED MEMORY IN XR - 2
    027034  027203 1020 00       9925 A4     CMPX2    HP              HAS ALL OF HEAP BEEN SWAPPED UP
    027035  027045 6000 00       9926        TZE      A5              TRANSFER IF YES
    027036  000001 1610 03       9927        SBX1     1,DU            DECREMENT NEW POINTER
    027037  000001 1620 03       9928        SBX2     1,DU            DECREMENT OLD POINTER
    027040  000000 2350 11       9929        LDA      0,1             GET NEW WORD
    027041  000000 2360 12       9930        LDQ      0,2             GET OLD WORD
    027042  000000 7560 11       9931        STQ      0,1             AND SWAP
    027043  000000 7550 12       9932        STA      0,2             AND SWAP
    027044  027034 7100 00       9933        TRA      A4              AND GO LOOP
    027045  027204 7410 00       9934 A5     STX1     SMAX            STORE LAST ADDRESS MOVED TO AS NEW HEAP
    027046  027204 2360 00       9935        LDQ      SMAX            GET OLD HEAP ADDRESS IN Q
    027047  027203 1760 00       9936        SBQ      HP              SUBTRACT NEW ADDRESS TO GET DISTANCE SWAPPED
    027050  027176 7560 00       9937        STQ      AT              STORE DISTANCE SWAPPED
 END OF BINARY CARD 00000485
    027051  026745 2360 00       9938        LDQ      CA              GET TOP OF OLD COMPACT SPACE
    027052  027203 1760 00       9939        SBQ      HP              SUBTRACT BASE OF PART MOVED
    027053  027176 5060 00       9940        DIV      AT              GET RELATIVE ADDRESS OF START OF RELOCATION TABLE
    027054  027177 7550 00       9941        STA      AR              AND STORE IN AR
    027055  027203 0750 00       9942        ADA      HP              MAKE STARTING ADDRESS ABSOLUTE
    027056  000000 6210 01       9943        EAX1     0,AU            GET STARTING ADDRESS FOR MOVE IN XR - 1
    027057  027176 2350 00       9944        LDA      AT              GET DISTANCE SWAPPED IN A
    027060  027177 1750 00       9945        SBA      AR              SUBTRACT OFFSET FOR START OF RELOCATION TABLE
    027061  000022 7710 00       9946        ARL      18              MOVE POSSIBLE NUMBER OF WORDS TO MOVE TO AL
    027062  026750 1150 00       9947        CMPA     CN              SEE IF MORE THAN LENGTH OF RELOCATION TABLE
    027063  027065 6040 00       9948        TMI      A6              TRANSFER IF LESS
    027064  026750 2350 00       9949        LDA      CN              GET NUMBER OF WORDS TO MOVE ON AL
    027065  000022 7350 00       9950 A6     ALS      18              MOVE MOVE LENGTH TO AU
    027066  027176 7550 00       9951        STA      AT              AND STORE IN TEMP
    027067  026750 7220 00       9952        LXL2     CN              GET LENGTH OF RELOCATION TABLE IN XR - 2
    027070  027203 0620 00       9953        ADX2     HP              GET POINTER TO END OF MOVE IN XR - 2
    027071  027176 1620 00       9954        SBX2     AT              SUBTRACT MOVE LENGTH TO GET DESTINATION ADDRESS
    027072  000010 7710 00       9955        ARL      8               POSITION COUNT FOR REPEAT
    027073  027104 6000 00       9956        TZE      A8              TRANSFER IF NOTHING TO MOVE
    027074  001400 6200 05       9957        EAX0     768,AL          GET COUNT AND A AND B BITS IN XR - 0 FOR MOVE
    027075  000001 1750 07       9958        SBA      1,DL            MAKE LOOP COUNT PROPERLY
    027076  000000011007
 END OF BINARY CARD 00000486
    027077  000000 5602 01       9959 A7     RPDX     ,1              MOVE
    027100  000000 2360 11       9960        LDQ      0,1             FROM TOP FRAGMENT
    027101  000000 7560 12       9961        STQ      0,2             NEXT TO LOWER FRAGMENT
    027102  000001 1750 03       9962        SBA      1,DU            SEE IF ANOTHER BLOCK OF 256 WORDS TO MOVE
    027103  027077 6050 00       9963        TPL      A7              TRANSFER IF MORE TO MOVE
    027104  026750 7210 00       9964 A8     LXL1     CN              GET NUMBER OF WORDS IN RELOCATION TABLE
    027105  027203 0610 00       9965        ADX1     HP              ADD ORIGIN OF TABLE
    027106  027202 2220 00       9966        LDX2     SHDIV           GET OLD STACK/HEAP DIVISION ADDRESS
    027107  000000 7420 11       9967        STX2     0,1             AND STORE IN RELOCATION TABLE
    027110  027203 2220 00       9968        LDX2     HP              GET NEW STACK/HEAP DIVISION ADDRESS
    027111  000000 4420 11       9969        SXL2     0,1             AND STORE IN RELOCATION TABLE
    027112  600000 3350 07       9970        LCA      =0600000,DL     GET AN END OF TABLE WORD
```

```
           R                                                          PASS 3

     027113  000001 7550 11        9971         STA       1,1                AND STORE IN TABLE
     027114  000002 4500 11        9972         STZ       2,1                GET A BEGINNING OF TABLE WORD
     027115  026750 0540 00        9973         AOS       CN                 INCREMENT TABLE LENGTH
     027116  026750 0540 00        9974         AOS       CN                 INCREMENT TABLE LENGTH
     027117  026750 0540 00        9975         AOS       CN                 INCREMENT TABLE LENGTH
     027120  026750 2350 00        9976 S1      LDA       CN                 GET NUMBER OF WORDS IN RELOCATION TABLE TO SORT
     027121  000022 7350 00        9977         ALS       18                 PUT COUNT IN AU
     027122  027205 7550 00        9978         STA       SD                 AND STORE IN SORT DISPLACEMENT
     027123  027203 0750 00        9979         ADA       HP                 ADD BASE OF RELOCATION TABLE ADDRESS
     027124  027206 7550 00        9980         STA       SE                 AND STORE IN END RELOCATION TABLE
   END OF BINARY CARD 00000487
     027125  027205 2350 00        9981 S2      LDA       SD                 GET CURRENT SORT DISPLACEMENT
     027126  000001 7710 00        9982         ARL       1                  DIVIDE BY ABOUT 2
     027127  000000 6350 01        9983         EAA       0,AU               MAKE IT AN INTEGER
     027130  027155 6000 00        9984         TZE       S6                 TRANSFER IF DONE SORTING
     027131  027205 7550 00        9985         STA       SD                 AND STORE AS NEW SORT DISPLACEMENT
     027132  027203 2230 00        9986         LDX3      HP                 GET BASE OF TABLE IN XR - 3
     027133  027205 0630 00        9987         ADX3      SD                 ADD DISPLACEMENT
     027134  000000 6210 13        9988 S3      EAX1      0,3                GET STARTING ADDRESS FOR SWEEP IN XR - 1
     027135  000000 6220 11        9989         EAX2      0,1                ALSO PUT IN XR - 2
     027136  027205 1620 00        9990         SBX2      SD                 GO BACK BY DISPLACEMENT
     027137  000000 2350 11        9991         LDA       0,1                GET CURRENT TABLE ELEMENT TO BE RELOCATED
     027140  000000 1150 12        9992 S4      CMPA      0,2                COMPARE AGAINST ELEMENT IN SORTED PART OF TABLE
     027141  027150 6030 00        9993         TRC       S5                 TRANSFER IF PLACE FOR RELOCATED ELEMENT FOUND
     027142  000000 2360 12        9994         LDQ       0,2                GET ELEMENT IN COMPARED LOCATION
     027143  000000 7560 11        9995         STQ       0,1                AND MOVE IT UP BY DISPLACEMENT
     027144  027205 1610 00        9996         SBX1      SD                 DECREMENT XR - 1 BY DISPLACEMENT
     027145  027205 1620 00        9997         SBX2      SD                 DECREMENT XR - 2 BY DISPLACEMENT
     027146  027203 1020 00        9998         CMPX2     HP                 SEE IF OFF BOTTOM OF TABLE
     027147  027140 6050 00        9999         TPL       S4                 TRANSFER IF STILL WITHIN TABLE
     027150  000000 7550 11       10000 S5      STA       0,1                STORE RELOCATED ELEMENT IN ITS PROPER PLACE
     027151  000001 0630 03       10001         ADX3      1,DU               STEP TO NEXT ELEMENT TO BE RELOCATED
     027152  027206 1030 00       10002         CMPX3     SE                 SEE IF ALL DONE WITH THIS SWEEP
   END OF BINARY CARD 00000488
     027153  027134 6040 00       10003         TMI       S3                 TRANSFER IF MORE TO DO
     027154  027125 7100 00       10004         TRA       S2                 TRANSFER TO SET UP NEXT SWEEP
     027155  026750 7210 00       10005 S6      LXL1      CN                 GET LENGTH OF RELOCATION TABLE IN XR - 1
     027156  027203 2220 00       10006         LDX2      HP                 GET BASE OF RELOCATION TABLE IN XR - 2
     027157  027202 2230 00       10007         LDX3      SHDIV              GET OLD BASE OF HEAP IN XR - 3
     027160  027204 2350 00       10008         LDA       SMAX               GET OLD HEAP BASE IN A
     027161  027203 1750 00       10009         SBA       HP                 SUBTRACT TO GET DISTANCE HEAP WAS MOVED
     027162  000022 7310 00       10010         ARS       18                 MOVE TO AL DISTANCE HEAP WAS MOVED
     027163  000000 1030 12       10011 S7      CMPX3     0,2                IS THIS THE FIRST HEAP ENTRY
     027164  027170 6000 00       10012         TZE       S8                 TRANSFER IF YES
     027165  000001 0620 03       10013         ADX2      1,DU               STEP TO NEXT TABLE ENTRY
     027166  000001 1610 03       10014         SBX1      1,DU               DECREMENT NUMBER LEFT
     027167  027163 7100 00       10015         TRA       S7                 AND LOOP
     027170  000000 0550 12       10016 S8      ASA       0,2                CORRECT RELOCATION TABLE ENTRY FOR MOVED HEAP
     027171  000001 0620 03       10017         ADX2      1,DU               STEP TO NEXT TABLE ENTRY
     027172  000001 1610 03       10018         SBX1      1,DU               DECREMENT NUMBER LEFT
```

```
           R                                                  PASS 3

      027173  027170 6010 00       10019          TNZ       S8             TRANSFER IF MORE TABLE ENTRIES
      027174  026536 7500 00       10020          STC2      LF             SET SECOND TIME THROUGH MARKING ROUTINE FLAG
      027175  026231 7100 00       10021          TRA       L1             AND TRANSFER TO MARKING ROUTINE
      027176  000000 000000        10022 AI       ZERO
      027177  000000 000000        10023 AR       ZERO
      027200  000000 000000        10024 BITT     ZERO
   END OF BINARY CARD 00000489
      027201  000000 000000        10025 BITL     ZERO
      027202  000000 000000        10026 SHDIV    ZERO
      027203  000000 000000        10027 HP       ZERO
      027204  000000 000000        10028 SMAX     ZERO
      027205  000000 000000        10029 SD       ZERO
      027206  000000 000000        10030 SE       ZERO
      027207  000000 000000        10031 MTOP     ZERO
      027210  000000 000000        10032 MBASE    ZERO
      027211  000000 000000        10033 REG      ZERO
      027212  027203 2260 00       10034 G1       LDX       S,HP           LOAD NEW VALUE OF STACK POINTER
      027213  026540 2270 00       10035          LDX       D,GD           LOAD NEW VALUE FOR DISPLAY REGISTER
      027214  000000 7100 00       10036 GX       TRA       **             AND RETURN FROM GARBAGE COLLECTOR
                                   10037          HEAD      T
      027215  000000 000000        10038 PTR      ZERO
      027216  000000 000000        10039 WPTR     ZERO
      027217  000000 000000        10040 OCT      ZERO
      027220  000000 000000        10041 ULEN     ZERO
      027221  000000 000000        10042 ROW      ZERO
      027222  000000 000000        10043 SKIP     ZERO
      027223  027222 000001        10044 PROC     ZERO      SKIP,1
      027224  027222 000001        10045          ZERO      SKIP,1
      027225  027215 000000        10046          ZERO      PTR
      027226  027215 000000        10047          ZERO      PTR
   END OF BINARY CARD 00000490
      027227  777773 777774        10048          ZERO      -4-1,-4
      027230  027215 000000        10049 MSCWT    ZERO      PTR
      027231  027215 000000        10050          ZERO      PTR
      027232  027215 000000        10051          ZERO      PTR
      027233  027215 000000        10052          ZERO      PTR
      027234  777773 777774        10053          ZERO      -4-1,-4
      027235  027215 000000        10054 MS       ZERO      T$PTR
      027236  027215 000000        10055          ZERO      T$PTR
      027237  777775 777776        10056          ZERO      -2-1,-2
      027240  027215 000000        10057 PTRT     ZERO      T$PTR
      027241  777776 777777        10058          ZERO      -1-1,-1
                                   10059          HEAD      L
      027242  030044 2350 00       10060 RD       LDA       SRTI           GET INITIAL SEARCH RULE TALLY WORD
      027243  030045 7550 00       10061          STA       SRT            AND STORE TO DIRECT SEARCH
      027244  030045 2270 51       10062 RD1      LDX7      SRT,I          GET POINTER TO FIRST CATALOG BLOCK
      027245  000001 2220 17       10063          LDX2      1,7            GET FILE REFERENCE NUMBER OF CATALOG
      027246  027267 6050 00       10064          TPL       RD3            TRANSFER IF THERE WAS A FILE REFERENCE NUMBER
      027247  000002 2370 17       10065 RD2      LDAQ      2,7            GET NAME OF CATALOG
      027250  000000 6260 17       10066          EAX6      0,7            SAVE POINTER TO CATALOG CONTROL BLOCK
```

```
          L                                                  PASS 3

     027251   000000 2270 17        10067          LDX7     0,7                 GET CATALOG WHERE CATALOG IS CATALOGED
     027252   000001 2220 17        10068          LDX2     1,7                 GET FILE REFERENCE NUMBER OF CATALOG
     027253   027247 6040 00        10069          TMI      RD2                 TRANSFER IF NO FILE REFERENCE NUMBER
     027254   410000 2210 03        10070          LDX1     =0410000,DU         GET ACCESS NEEDED ON CATALOG
  END OF BINARY CARD 00000491
     027255   003662 7000 00        10071          TSX0     SOPEN               ATTEMPT TO OPEN CATALOG
     027256   000222 2370 51        10072          LDAQ     ,SSTAT,I            GET STATUS RETURN WORDS
     027257   000000 6200 01        10073          EAX0     0,AU                GET STATUS OF OPERATION
     027260   000770 3000 03        10074          CANX0    =0770,DU            WAS IT ALL RIGHT
     027261   777777 6010 00        10075          TNZ      SERROR              NO - ERROR SOMEWHERE
     027262   410000 1000 03        10076          CMPX0    =0410000,DU         IS IT THE DESIRED STATUS
     027263   027300 6010 00        10077          TNZ      RD4                 NO - TRANSFER
     027264   000000 6200 05        10078          EAX0     0,AL                GET FILE REFERENCE NUMBER OF CATALOG JUST OPENED
     027265   000001 7400 16        10079          STX0     1,6                 AND STORE IN CATALOG CONTROL BLOCK
     027266   027244 7100 00        10080          TRA      RD1                 TRANSFER TO TRY AGAIN
     027267   030126 2370 00        10081 RD3      LDAQ     NAME                GET DESIRED FILE NAME
     027270   001000 2210 03        10082          LDX1     =0001000,DU         ASK FOR READ PERMISSION
     027271   003662 7000 00        10083          TSX0     SOPEN               ATTEMPT TO OPEN DESIRED FILE
     027272   000222 2370 51        10084          LDAQ     ,SSTAT,I            GET STATUS RETURN WORDS OF OPERATION
     027273   000000 6200 01        10085          EAX0     0,AU                GET STATUS IN XR - 0
     027274   000770 3000 03        10086          CANX0    =0770,DU            WAS OPERATION ALL RIGHT
     027275   777777 6010 00        10087          TNZ      SERROR              NO - SOMETHING WRONG SOMEWHERE
     027276   001000 1000 03        10088          CMPX0    =0001000,DU         IS IT THE DESIRED STATUS
     027277   027303 6000 00        10089          TZE      RD5                 YES - TRANSFER
     027300   030045 0110 56        10090 RD4      NOP      SRT,ID              STEP TO NEXT CATALOG TO TRY
     027301   027244 6070 00        10091          TTF      RD1                 TRANSFER IF MORE CATALOGS TO TRY
     027302   777777 7100 00        10092          TRA      SERROR              TRANSFER IF FILE CANNOT BE FOUND
  END OF BINARY CARD 00000492
     027303   000000 6200 05        10093 RD5      EAX0     0,AL                GET FILE REFERENCE NUMBER OF FILE IN XR - 0
     027304   030130 7400 00        10094          STX0     FRN                 AND SAVE
     027305   000000 6200 02        10095          EAX0     0,QU                GET UPPER HALF OF LENGTH OF FILE
     027306   777777 6010 00        10096          TNZ      SERROR              TOO BIG - ERROR
     027307   000000 6220 06        10097          EAX2     0,QL                GET LENGTH OF FILE IN XR - 2
     027310   030234 2210 03        10098          LDX1     USER,DU             GET POINTER TO USER TABLE CONTROL WORD
     027311   030160 7000 00        10099          TSX0     ALOC                ALLOCATE NEEDED ADDITIONAL SPACE IN USER AREA
     027312   030131 0760 00        10100          ADQ      FREE                ADD READ LOCATION TO READ LENGTH
     027313   000044 7370 00        10101          LLS      36                  GET LOC/LEN IN A REGISTER
     027314   030130 2360 00        10102          LDQ      FRN                 GET FILE REFERENCE NUMBER IN QU
     027315   003721 7000 00        10103          TSX0     SREAD               READ IN NEW FILE
     027316   000222 2370 51        10104          LDAQ     ,SSTAT,I            GET STATUS RETURN WORDS OF OPERATION
     027317   777777 6010 00        10105          TNZ      SERROR              TRANSFER IF NOT CORRECT
     027320   030130 2360 00        10106          LDQ      FRN                 GET FILE REFERENCE NUMBER OF FILE
     027321   003702 7000 00        10107          TSX0     SCLOSE              AND CLOSE IT
     027322   000222 2370 51        10108          LDAQ     ,SSTAT,I            GET STATUS RETURN WORDS OF OPERATION
     027323   777777 6010 00        10109          TNZ      SERROR              CLOSE DIDN'T WORK - ERROR
     027324   030240 7200 00        10110 START    LXL0     PROG                GET ADDRESS OF NEXT ALLOCATED PROG TABLE ENTRY
     027325   030132 7400 00        10111          STX0     CEL                 AND STORE IN CHAIN TABLE ENTRY PROTOTYPE
     027326   030240 2210 03        10112          LDX1     PROG,DU             GET POINTER TO PROG TABLE CONTROL WORD
     027327   000002 2220 03        10113          LDX2     2,DU                GET LENGTH OF ENTRY IN PROG TABLE IN XR - 2
     027330   030160 7000 00        10114          TSX0     ALOC                ALLOCATE 2 WORD ENTRY IN PROG TABLE
```

```
END OF BINARY CARD 00000493
     027331  030131 2350 00      10115         LDA      FREE          GET POINTER TO BASE OF CURRENT PROGRAM SEGMENT
     027332  000000 7550 11      10116         STA      0,1           AND STORE IN PROG TABLE ENTRY
     027333  000001 4500 11      10117         STZ      1,1           ZERO OUT SECOND WORD OF ENTRY
     027334  030234 7270 00      10118         LXL7     USER          GET LENGTH OF NEW PROGRAM
     027335  030234 0670 00      10119         ADX7     USER          ADD TO LOCATION TO INITIALIZE XR - 7 TO END
     027336  777777 2200 17      10120         LDX0     -1,7          GET REQUIRED LENGTH OF ENVIRONMENT
     027337  000001 4400 11      10121         SXL0     1,1           AND STORE IN PROG TABLE ENTRY
     027340  000001 1670 03      10122         SBX7     1,DU          STEP XR - 7 OVER LENGTH WORD
     027341  777777 2340 17      10123 ESYM    SZN      -1,7          ARE THERE ANY SYMDEFS OR SYMREFS
     027342  027456 6000 00      10124         TZE      ETYP          NO - GO TO ENTER TYPES IN TABLE
     027343  000000 6210 17      10125         EAX1     0,7           GET POINTER TO END OF DEFINITION IN XR - 1
     027344  777777 1610 17      10126         SBX1     -1,7          GET POINTER TO START OF MODE IN DEFINITION
     027345  777775 2350 11      10127         LDA      -3,1          GET DEFINITION OF SYMBOL IN A
     027346  000000 6200 01      10128         EAX0     0,AU          GET ADDRESS PART IN XR - 0
     027347  030132 4400 00      10129         SXL0     CEL           AND STORE IN ENTRY PROTOTYPE
     027350  000000 6200 05      10130         EAX0     0,AL          GET LENGTH OF VALUE OR ZERO IN XR - 0
     027351  030133 4400 00      10131         SXL0     CEL+1         AND STORE IN ENTRY PROTOTYPE
     027352  030236 2260 00      10132         LDX6     SYM           GET POINTER TO START OF SYMBOL TABLE
     027353  000000 2340 16      10133 ESYM1   SZN      0,6           ARE THERE ANY MORE SYMBOL TABLE ENTRIES
     027354  027403 6010 00      10134         TNZ      ESYM2         YES - GO TO COMPARE WITH CURRENT SYMBOL
     027355  777777 2220 17      10135         LDX2     -1,7          GET LENGTH OF MODE IN XR - 2
     027356  000003 0620 03      10136         ADX2     3,DU          GET LENGTH OF NEW ENTRY IN XR - 2
END OF BINARY CARD 00000494
     027357  030236 2210 03      10137         LDX1     SYM,DU        GET POINTER TO SYM TABLE CONTROL WORD
     027360  030160 7000 00      10138         TSX0     ALOC          ALLOCATE SPACE IN SYM TABLE
     027361  000000 6350 12      10139         EAA      0,2           GET LENGTH OF ENTRY IN AU
     027362  000000 7550 11      10140         STA      0,1           AND STORE IN FIRST WORD OF ENTRY
     027363  000001 6230 11      10141         EAX3     1,1           GET DESTINATION OF MOVE IN XR - 3
     027364  000000 6240 17      10142         EAX4     0,7           GET POINTER TO END OF NEW ELEMENT IN XR - 4
     027365  777777 1640 17      10143         SBX4     -1,7          GET POINTER TO START OF MODE IN XR - 4
     027366  000002 1640 03      10144         SBX4     2,DU          GET SOURCE OF MOVE IN XR - 4
     027367  777777 6350 12      10145         EAA      -1,2          GET LENGTH OF MOVE IN AU
     027370  000010 7710 00      10146         ARL      8             SHIFT FOR REPEAT
     027371  001400 6200 05      10147         EAX0     768,AL        GET COUNT AND A AND B BITS IN XR - 0
     027372  000000011007
     027373  000000 5602 01      10148         RPDX     ,1            MOVE
     027374  000000 2360 14      10149         LDQ      0,4           FROM PROGRAM DEFINITION
     027375  000000 7560 13      10150         STQ      0,3           TO NEW TABLE ENTRY
     027376  030237 7200 00      10151         LXL0     CHAIN         GET ADDRESS OF NEXT ENTRY IN CHAIN TABLE
     027377  000000 4400 11      10152         SXL0     0,1           AND STORE AS LINK IN TABLE ENTRY
     027400  000000 2200 03      10153         LDX0     0,DU          GET A ZERO
     027401  030133 7400 00      10154         STX0     CEL+1         AND STORE LINK IN PROTOTYPE
     027402  027444 7100 00      10155         TRA      ESYM5         GO TO ENTER PROTOTYPE IN CHAIN TABLE
     027403  000000 6240 17      10156 ESYM2   EAX4     0,7           GET POINTER TO END OF MODE IN XR - 4
     027404  777777 1640 17      10157         SBX4     -1,7          GET POINTER TO START OF MODE IN XR - 4
END OF BINARY CARD 00000495
     027405  000002 1640 03      10158         SBX4     2,DU          GET POINTER TO START OF NAME BEFORE MODE
     027406  000001 6230 16      10159         EAX3     1,6           GET POINTER TO START OF KEY IN TABLE ENTRY
     027407  000000 2350 16      10160         LDA      0,6           GET LENGTH + 1 IN AU
```

```
                                                     PASS 3

     027410  777777 6350 01      10161        EAA     -1,AU           GET LENGTH OF COMPARE IN AU
     027411  000010 7710 00      10162        ARL     8               POSITION COUNT FOR REPEAT
     027412  001440 6200 05      10163        EAX0    768+32,AL       GET COUNT, A AND B BITS, AND TNZ BIT IN XR - 0
     027413  000040 5602 01      10164        RPDX    ,1,TNZ          COMPARE
     027414  000000 2360 14      10165        LDQ     0,4             PROGRAM SYMBOL
     027415  000000 1160 13      10166        CMPQ    0,3             WITH TABLE ENTRY
     027416  027421 6000 00      10167        TZE     ESYM3           TRANSFER IF CORRECT TABLE ENTRY FOUND
     027417  000000 0660 16      10168        ADX6    0,6             STEP TO NEXT TABLE ENTRY
     027420  027353 7100 00      10169        TRA     ESYM1           AND LOOP
     027421  000000 7210 16      10170 ESYM3  LXL1    0,6             GET IN XR - 1 POINTER TO DEFINITION CHAIN
     027422  030237 0610 00      10171        ADX1    CHAIN           MAKE POINTER ABSOLUTE
     027423  000001 7200 11      10172        LXL0    1,1             GET TYPE OF CHAIN ENTRY IN XR - 0
     027424  027433 6000 00      10173        TZE     ESYM4           TRANSFER IF IT IS A SYMDEF
     027425  030237 7200 00      10174        LXL0    CHAIN           GET ADDRESS OF NEXT CHAIN TABLE ENTRY
     027426  030133 7400 00      10175        STX0    CEL+1           AND STORE IN LINK IN ENTRY PROTOTYPE
     027427  030237 1610 00      10176        SBX1    CHAIN           MAKE CURRENT TABLE ENTRY POINTER RELATIVE
     027430  030133 1410 00      10177        SSX1    CEL+1           MAKE LINK REFER TO CURRENT TABLE ENTRY
     027431  000000 4400 16      10178        SXL0    0,6             STORE POINTER TO NEXT CHAIN ENTRY AS DEF LINK
     027432  027444 7100 00      10179        TRA     ESYM5           GO INSERT PROTOTYPE IN CHAIN TABLE
 END OF BINARY CARD 00000496
     027433  000001 0610 11      10180 ESYM4  ADX1    1,1             STEP TO NEXT ELEMENT OF THE CHAIN
     027434  000001 2200 11      10181        LDX0    1,1             SEE IF AT THE END OF THE CHAIN
     027435  027433 6010 00      10182        TNZ     ESYM4           TRANSFER IF MORE ELEMENTS IN CHAIN
     027436  000000 2200 03      10183        LDX0    0,DU            GET A ZERO IN XR - 0
     027437  030133 7400 00      10184        STX0    CEL+1           STORE ZERO IN PROTOTYPE LINK TO END CHAIN
     027440  000001 7410 11      10185        STX1    1,1             STORE LAST ENTRY ADDRESS IN LINK
     027441  030237 7200 00      10186        LXL0    CHAIN           GET ADDRESS OF NEXT ENTRY IN CHAIN TABLE
     027442  030237 0600 00      10187        ADX0    CHAIN           MAKE IT ABSOLUTE
     027443  000001 1400 11      10188        SSX0    1,1             MAKE CHAIN END REFER TO ENTRY TO BE ADDED
     027444  030237 2210 03      10189 ESYM5  LDX1    CHAIN,DU        GET POINTER TO CHAIN TABLE CONTROL WORD
     027445  000002 2220 03      10190        LDX2    2,DU            GET ENTRY LENGTH IN XR - 2
     027446  030160 7000 00      10191        TSX0    ALOC            ALLOCATE 2 WORDS IN CHAIN TABLE
     027447  030132 2350 00      10192        LDA     CEL             GET FIRST WORD OF PROTOTYPE
     027450  000000 7550 11      10193        STA     0,1             AND STORE IN TABLE ENTRY
     027451  030133 2350 00      10194        LDA     CEL+1           GET SECOND WORD OF PROTOTYPE
     027452  000001 7550 11      10195        STA     1,1             AND STORE IN TABLE ENTRY
     027453  777777 1670 17      10196        SBX7    -1,7            MAKE XR - 7 POINT TO MODE
     027454  000003 1670 03      10197        SBX7    3,DU            MAKE XR - 7 POINT TO FRONT OF DEFINITION
     027455  027341 7100 00      10198        TRA     ESYM            AND LOOP
     027456  000001 1670 03      10199 ETYP   SBX7    1,DU            STEP XR - 7 BEFORE ZERO FLAG WORD
     027457  777777 2340 17      10200 ETYP1  SZN     -1,7            ARE THERE ANY MORE TYPE DEFINITIONS IN PROGRAM
     027460  027566 6000 00      10201        TZE     ETYP6           TRANSFER IF NO MORE TYPES
 END OF BINARY CARD 00000497
     027461  777777 2200 17      10202        LDX0    -1,7            GET LENGTH OF MODE IN XR - 0
     027462  000001 1000 03      10203        CMPX0   1,DU            SEE IF ANY MODE
     027463  027522 6000 00      10204        TZE     ETYP5           TRANSFER IF NO MODE - GO ENTER IN TYPE TABLE
     027464  030235 2260 00      10205        LDX6    TYPE            GET POINTER TO START OF TYPE TABLE
     027465  000000 2340 16      10206 ETYP2  SZN     0,6             ARE THERE ANY MORE ENTRIES IN TYPE TABLE
     027466  027522 6000 00      10207        TZE     ETYP5           TRANSFER IF NOT - GO ENTER IN TYPE TABLE
     027467  000000 6240 17      10208        EAX4    0,7             GET POINTER TO END OF MODE IN PROGRAM
```

```
              L                                          PASS 3

         027470  777777 1640 17       10209        SBX4     -1,7          GET POINTER TO START OF MODE IN PROGRAM
         027471  000001 6230 16       10210        EAX3     1,6           GET POINTER TO START OF MODE IN TABLE ENTRY
         027472  000000 2350 16       10211        LDA      0,6           GET LENGTH+1 OF MODE IN AU
         027473  777777 6350 01       10212        EAA      -1,AU         GET LENGTH OF MODE IN AU
         027474  000010 7710 00       10213        ARL      8             POSITION COUNT FOR REPEAT
         027475  001440 6200 05       10214        EAX0     768+32,AL     GET COUNT, A AND B BITS, AND TNZ BIT IN XR - 0
         027476  000000011007
         027477  000040 5602 01       10215        RPDX     ,1,TNZ        COMPARE
         027500  000000 2360 14       10216        LDQ      0,4           PROGRAM TYPE MODE
         027501  000000 1160 13       10217        CMPQ     0,3           WITH TABLE ENTRY MODE
         027502  027506 6000 00       10218        TZE      ETYP3         TRANSFER IF MATCH
         027503  000000 0660 16       10219        ADX6     0,6           STEP TO TYPE IN TABLE ENTRY
         027504  000000 0660 16       10220        ADX6     0,6           STEP TO NEXT TABLE ENTRY
         027505  027465 7100 00       10221        TRA      ETYP2         AND LOOP
         027506  000000 0660 16       10222 ETYP3  ADX6     0,6           GET POINTER TO TYPE IN TABLE ENTRY
     END OF BINARY CARD 00000498
         027507  000000 7210 16       10223        LXL1     0,6           GET POINTER TO CHAINED ADDRESSES IN XR - 1
         027510  000000 0610 11       10224 ETYP4  ADX1     0,1           STEP TO NEXT ADDRESS IN CHAIN
         027511  000000 2200 11       10225        LDX0     0,1           SEE IF AT END OF CHAIN
         027512  027510 6010 00       10226        TNZ      ETYP4         TRANSFER IF MORE IN CHAIN
         027513  777777 1670 17       10227        SBX7     -1,7          STEP TO TYPE PART OF TYPE DEFINITION IN PROGRAM
         027514  000000 7410 11       10228        STX1     0,1           STORE ADDRESS OF END OF CHAIN IN END OF CHAIN
         027515  777777 7200 17       10229        LXL0     -1,7          GET ADDRESS OF NEW CHAIN TO BE DEFINED
         027516  030131 0600 00       10230        ADX0     FREE          MAKE ADDRESS ABSOLUTE
         027517  000000 1400 11       10231        SSX0     0,1           AND LINK NEW CHAIN ON END OF OLD CHAIN
         027520  777777 1670 17       10232        SBX7     -1,7          STEP XR - 7 TO NEXT TYPE DEFINITION IN PROGRAM
         027521  027457 7100 00       10233        TRA      ETYP1         AND LOOP
         027522  777777 2220 17       10234 ETYP5  LDX2     -1,7          GET LENGTH OF MODE IN XR - 2
         027523  000001 0620 03       10235        ADX2     1,DU          ADD ONE FOR LENGTH WORD
         027524  030235 2210 03       10236        LDX1     TYPE,DU       GET POINTER TO TYPE TABLE CONTROL WORD IN XR - 1
         027525  030160 7000 00       10237        TSX0     ALOC          ALLOCATE SPACE FOR MODE IN TYPE TABLE
         027526  000000 6350 12       10238        EAA      0,2           GET LENGTH OF ENTRY IN AU
         027527  000000 7550 11       10239        STA      0,1           AND STORE AS FIRST WORD OF TYPE TABLE ENTRY
         027530  000000 6240 17       10240        EAX4     0,7           GET POINTER TO END OF MODE IN PROGRAM IN XR - 4
         027531  777777 1640 17       10241        SBX4     -1,7          GET POINTER TO START OF PROGRAM MODE IN XR - 4
         027532  000001 6230 11       10242        EAX3     1,1           GET POINTER TO START OF MODE IN NEW TABLE ENTRY
         027533  777777 6350 01       10243        EAA      -1,AU         GET LENGTH OF MODE IN AU
         027534  000010 7710 00       10244        ARL      8             POSITION COUN FOR REPEAT
     END OF BINARY CARD 00000499
         027535  001400 6200 05       10245        EAX0     768,AL        GET COUNT AND A AND B BITS IN XR - 0
         027536  000000011007
         027537  000000 5602 01       10246        RPDX     ,1            MOVE
         027540  000000 2360 14       10247        LDQ      0,4           FROM PROGRAM MODE
         027541  000000 7560 13       10248        STQ      0,3           TO TABLE ENTRY MODE
         027542  777777 1670 17       10249        SBX7     -1,7          GET POINTER TO END OF PROGRAM TYPE IN XR - 7
         027543  777777 2220 17       10250        LDX2     -1,7          GET LENGTH OF TYPE IN XR - 2
         027544  030235 2210 03       10251        LDX1     TYPE,DU       GET POINTER TO TYPE TABLE CONTROL WORD
         027545  030160 7000 00       10252        TSX0     ALOC          ALLOCATE SPACE FOR TYPE IN TYPE TABLE
         027546  000000 7420 11       10253        STX2     0,1           STORE LENGTH OF ENTRY IN FIRST WORD OF ENTRY
         027547  777777 7200 17       10254        LXL0     -1,7          GET CHAIN POINTER IN XR - 0
         L                                         PASS 3

   027550  030131 0600 00       10255            ADX0      FREE              ADD PROGRAM BASE TO MAKE POINTER ABSOLUTE
   027551  000000 4400 11       10256            SXL0      0,1               AND STORE IN TABLE ENTRY
   027552  000000 6240 17       10257            EAX4      0,7               GET POINTER TO END OF TYPE IN XR - 4
   027553  777777 1640 17       10258            SBX4      -1,7              GET POINTER TO START OF TYPE IN XR - 4
   027554  000001 6230 11       10259            EAX3      1,1               GET POINTER TO START OF TYPE IN NEW TABLE ENTRY
   027555  777777 6350 12       10260            EAA       -1,2              GET LENGTH OF TYPE IN AU
   027556  000010 7710 00       10261            ARL       8                 POSITION COUNT FOR REPEAT
   027557  001400 6200 05       10262            EAX0      768,AL            GET COUNT AND A AND B BITS IN XR - 0
   027560  000000011007
   027561  000000 5602 01       10263            RPDX      ,1                MOVE
   027562  000000 2360 14       10264            LDQ       0,4               FROM PROGRAM TYPE
 END OF BINARY CARD 00000500
   027563  000000 7560 13       10265            STQ       0,3               TO NEW TABLE ENTRY
   027564  777777 1670 17       10266            SBX7      -1,7              STEP XR - 7 TO NEXT PROGRAM TYPE
   027565  027457 7100 00       10267            TRA       ETYP1             AND LOOP
   027566  000001 1670 03       10268 ETYP6      SBX7      1,DU              STEP OVER ZERO FLAG WORD
   027567  030131 7470 00       10269            STX7      FREE              STORE ADDRESS OF FREE LOCATION ABOVE PROGRAM
   027570  030234 1670 00       10270            SBX7      USER              GET AMOUNT OF USER AREA IN USE
   027571  030234 4470 00       10271            SXL7      USER              AND STORE AS LENGTH OF USER SEGMENT
   027572  030134 2260 00       10272            LDX6      SYMP              GET CURRENT SEARCH LOCATION IN SYM TABLE
   027573  030236 0660 00       10273            ADX6      SYM               MAKE POINTER ABSOLUTE
   027574  000000 2340 16       10274 SEA1       SZN       0,6               ARE THERE ANY MORE TABLE ENTRIES
   027575  027613 6000 00       10275            TZE       BIND              NO - GO BIND ALL PROGRAMS TOGETHER
   027576  000000 7210 16       10276            LXL1      0,6               GET POINTER TO DEFINITION CHAIN IN XR - 1
   027577  030237 0610 00       10277            ADX1      CHAIN             MAKE POINTER ABSOLUTE
   027600  000001 7200 11       10278            LXL0      1,1               GET TYPE OF ENTRY IN XR - 0
   027601  027604 6010 00       10279            TNZ       SEA2              TRANSFER IF NO SYMDEF
   027602  000000 0660 16       10280            ADX6      0,6               STEP TO NEXT TABLE ENTRY IN SYM TABLE
   027603  027574 7100 00       10281            TRA       SEA1              AND LOOP
   027604  000001 2350 16       10282 SEA2       LDA       1,6               GET FIRST HALF OF NAME IN A REGISTER
   027605  000002 2360 16       10283            LDQ       2,6               GET SECOND HALF OF NAME IN Q REGISTER
   027606  030126 7570 00       10284            STAQ      NAME              STORE NAME FOR READ ROUTINE
   027607  000000 0660 16       10285            ADX6      0,6               STEP TO NEXT ENTRY IN SYMBOL TABLE
   027610  030236 1660 00       10286            SBX6      SYM               MAKE POINTER TO CURRENT ENTRY RELATIVE
 END OF BINARY CARD 00000501
   027611  030134 7460 00       10287            STX6      SYMP              AND SAVE FOR NEXT TIME
   027612  027242 7100 00       10288            TRA       RD                AND GO TO READ ROUTINE
   027613  030131 2210 00       10289 BIND       LDX1      FREE              GET POINTER TO AREA AFTER PROGRAM
   027614  000001 0610 03       10290            ADX1      1,DU              STEP POINTER FOR FINAL TRANSFER INSTRUCTION
   027615  030131 7410 00       10291            STX1      FREE              AND STORE NEW FREE POINTER
   027616  030234 2210 03       10292            LDX1      USER,DU           GET POINTER TO USER TABLE CONTROL WORD
   027617  000001 2220 03       10293            LDX2      1,DU              GET NUMBER OF WORDS TO ADD IN XR - 2
   027620  030160 7000 00       10294            TSX0      ALOC              ALLOCATE SPACE FOR FINAL TRANSFER INSTRUCTION
   027621  030135 2350 00       10295            LDA       BINDI             GET TRANSFER INSTRUCTION FOR END OF CODE
   027622  000000 7550 11       10296            STA       0,1               AND STORE RETURN TRANSFER AT END OF CODE
   027623  030235 2240 00       10297            LDX4      TYPE              GET TYPE TABLE BASE IN XR - 4
   027624  000000 2340 14       10298 BIND0      SZN       0,4               ARE THERE ANY MORE TYPE ENTRIES
   027625  027662 6000 00       10299            TZE       BIND2             TRANSFER IF NO MORE TYPES
   027626  000000 0640 14       10300            ADX4      0,4               STEP TO TYPE PART OF TABLE ENTRY
   027627  030131 2210 00       10301            LDX1      FREE              GET POINTER TO NEXT AVAILABLE WORD IN XR - 1
```

```
     027630  000000 7220 14        10302          LXL2      0,4               GET POINTER TO ADDRESS CHAIN FOR THIS TYPE
     027631  777777 6000 00        10303          TZE       $ERROR            COMPILER ERROR - NO CHAIN
     027632  000000 2230 12        10304 BIND1    LDX3      0,2               GET POINTER TO NEXT WORD IN CHAIN IN XR - 3
     027633  030141 7430 00        10305          STX3      TEMP              AND STORE IN TEMPORARY LOCATION
     027634  000000 7410 12        10306          STX1      0,2               STORE POINTER TO TYPE TEMPLATE IN CURRENT WORD
     027635  030141 0620 00        10307          ADX2      TEMP              MAKE XR - 2 POINT TO NEXT WORD IN CHAIN
     027636  030141 2340 00        10308          SZN       TEMP              SEE IF THERE ARE MORE WORDS IN CHAIN
 END OF BINARY CARD 00000502
     027637  027632 6010 00        10309          TNZ       BIND1             TRANSFER IF MORE WORDS TO CHAIN
     027640  000000 2220 14        10310          LDX2      0,4               GET LENGTH + 1 OF TEMPLATE
     027641  000001 1620 03        10311          SBX2      1,DU              GET LENGTH OF TEMPLATE IN XR - 2
     027642  030131 0420 00        10312          ASX2      FREE              AND ADD TO FREE TO ALLOCATE MEMORY
     027643  030235 1640 00        10313          SBX4      TYPE              MAKE TYPE TABLE POINTER RELATIVE
     027644  030234 2210 03        10314          LDX1      USER,DU           GET POINTER TO USER TABLE CONTROL WORD
     027645  030160 7000 00        10315          TSX0      ALOC              ALLOCATE SPACE IN USER AREA
     027646  030235 0640 00        10316          ADX4      TYPE              MAKE TYPE TABLE POINTER ABSOLUTE
     027647  000001 6220 14        10317          EAX2      1,4               GET ADDRESS OF TEMPLATE IN XR - 2
     027650  000000 2350 14        10318          LDA       0,4               GET LENGTH OF TEMPLATE + 1 IN AU
     027651  777777 6350 01        10319          EAA       -1,AU             GET LENGTH OF TEMPLATE IN AU - ZERO IN AL
     027652  000010 7710 00        10320          ARL       8                 POSITION COUNT FOR REPEAT
     027653  001400 6200 05        10321          EAX0      768,AL            GET COUNT AND A AND B BITS IN XR - 0
     027654  00000001100/
     027655  000000 5602 01        10322          RPDX      ,1                MOVE
     027656  000000 2360 12        10323          LDQ       0,2               FROM TEMPLATE IN TABLE
     027657  000000 7560 11        10324          STQ       0,1               TO NEWLY ALLOCATED MEMORY
     027660  000000 0640 14        10325          ADX4      0,4               STEP XR - 4 TO NEXT TABLE ENTRY
     027661  027624 7100 00        10326          TRA       BIND0             AND LOOP
     027662  030131 2350 00        10327 BIND2    LDA       FREE              GET POINTER TO CURRENT END OF ALLOCATED MEMORY
     027663  030136 7550 00        10328          STA       PROG1             SAVE LOCATION OF ENVIRONMENT OF MAIN PROGRAM
     027664  027210 7550 00        10329          STA       RSMBASE           STORE POINTER TO BASE OF COLLECTABLE STORAGE
 END OF BINARY CARD 00000503
     027665  000022 7710 00        10330          ARL       18                PUT POINTER IN AL
     027666  030240 2210 00        10331          LDX1      PROG              GET POINTER TO START OF PROGRAM TABLE IN XR - 1
     027667  000000 2340 11        10332          SZN       0,1               SEE IF THERE IS AT LEAST ONE ENTRY
     027670  777777 6000 00        10333          TZE       $ERROR            NO - COMPILER ERROR
     027671  000000 6200 05        10334 BIND3    EAX0      0,AL              GET FREE MEMORY POINTER IN XR - 0
     027672  000001 7400 11        10335          STX0      1,1               AND STORE AS ENVIRONMENT OF CURRENT PROGRAM
     027673  000000 2220 11        10336          LDX2      0,1               GET ADDRESS OF BASE OF CODE IN XR - 2
     027674  000000 7400 12        10337          STX0      0,2               STORE POINTER TO ENVIRONMENT IN FIRST INSTRUCTION
     027675  000001 0350 11        10338          ADLA      1,1               ADD LENGTH OF ENVIRONMENT TO AL
     027676  000002 0610 03        10339          ADX1      2,DU              STEP TO NEXT TABLE ENTRY IN PROGRAM TABLE
     027677  000000 2340 11        10340          SZN       0,1               ARE THERE ANY MORE ENTRIES IN PROGRAM TABLE
     027700  027671 6010 00        10341          TNZ       BIND3             TRANSFER IF THERE ARE MORE ENTRIES
     027701  000000 6220 05        10342          EAX2      0,AL              GET NEW FREE POINTER IN XR - 2
     027702  030131 1620 00        10343          SBX2      FREE              SUBTRACT OLD FREE POINTER TO GET LENGTH NEEDED
     027703  000000 6200 05        10344          EAX0      0,AL              GET POINTER TO FREE AREA IN XR - 0
     027704  030131 7400 00        10345          STX0      FREE              ALLOCATE ALL MEMORY USED FOR PROGRAM ENVIRONMENTS
     027705  030234 2210 03        10346          LDX1      USER,DU           GET POINTER TO USER AREA CONTROL WORD
     027706  030160 7000 00        10347          TSX0      ALOC              AND ALLOCATE MEMORY IN USER AREA
     027707  030234 7100 51        10348          TRA       USER,I            EXECUTE ALL PROGRAMS TO DEFINE SYMBOLS
```

```
         L                                                   PASS 3

     027710  030236 2250 00       10349 BINDR  LDX5    SYM             GET BASE OF SYMBOL TABLE TABLE IN XR - 5
     027711  000000 7210 15       10350 BIND4  LXL1    0,5             GET POINTER TO DEF/REF CHAIN IN XR - 1
     027712  030237 0610 00       10351        ADX1    CHAIN           MAKE POINTER ABSOLUTE
 END OF BINARY CARD 00000504
     027713  000001 7200 11       10352        LXL0    1,1             GET DEF/REF FLAG IN XR - 0
     027714  777777 6010 00       10353        TNZ     $ERROR          ERROR - NO SYMDEF FOR THIS SYMBOL
     027715  030240 2220 00       10354        LDX2    PROG            GET POINTER TO BASE OF PROGRAM TABLE IN XR - 2
     027716  000000 0620 11       10355        ADX2    0,1             ADD PROGRAM NUMBER WHERE SYMBOL IS DEFINED
     027717  000000 7240 11       10356        LXL4    0,1             GET ADDRESS OF SYMDEF IN XR - 4
     027720  000001 0640 12       10357        ADX4    1,2             ADD BASE OF ENVIRONMENT TO MAKE IT ABSOLUTE
     027721  030137 2350 00       10358        LDA     BLNKS           GET NAME OF ENTRY POINT IN A REGISTER
     027722  000001 1150 15       10359        CMPA    1,5             IS CURRENT SYMBOL THE ENTRY POINT
     027723  027727 6010 00       10360        TNZ     BIND5           TRANSFER IF NOT
     027724  000002 1150 15       10361        CMPA    2,5             IS CURRENT SYMBOL THE ENTRY POINT
     027725  027727 6010 00       10362        TNZ     BIND5           TRANSFER IF NOT
     027726  030140 7440 00       10363        STX4    ENTRY           SAVE ABSOLUTE ENTRY ADDRESS
     027727  000001 2200 11       10364 BIND5  LDX0    1,1             GET POINTER TO NEXT SYMREF IN XR - 0
     027730  027751 6000 00       10365        TZE     BIND6           TRANSFER IF NO MORE SYMREFS IN THIS CHAIN
     027731  000001 0610 11       10366        ADX1    1,1             STEP TO NEXT SYMREF
     027732  030240 2220 00       10367        LDX2    PROG            GET POINTER TO BASE OF PROGRAM TABLE IN XR - 2
     027733  000000 0620 11       10368        ADX2    0,1             ADD SYMREF PROGRAM POINTER TO MAKE IT ABSOLUTE
     027734  000000 7230 11       10369        LXL3    0,1             GET ENVIRONMENT ADDRESS OF SYMBOL
     027735  000001 0630 12       10370        ADX3    1,2             ADD BASE OF ENVIRONMENT TO MAKE IT ABSOLUTE
     027736  000000 6220 14       10371        EAX2    0,4             GET ADDRESS OF VALUE DEFINED BY SYMDEF IN XR - 2
     027737  000001 2350 11       10372        LDA     1,1             GET LENGTH OF VALUE IN AL
     027740  000000 6350 05       10373        EAA     0,AL            MOVE LENGTH TO AU
 END OF BINARY CARD 00000505
     027741  777777 6000 00       10374        TZE     $ERROR          ERROR - TOO MANY SYMDEFS FOR THIS SYMBOL
     027742  000010 7710 00       10375        ARL     8               POSITION COUNT FOR REPEAT
     027743  001400 6200 05       10376        EAX0    768,AL          GET COUNT AND A AND B BITS IN XR - 0
     027744  000000011007
     027745  000000 5602 01       10377        RPDX    ,1              MOVE
     027746  000000 2360 12       10378        LDQ     0,2             FROM VALUE OF SYMDEF
     027747  000000 7560 13       10379        STQ     0,3             TO ADDRESS OF SYMREF
     027750  027727 7100 00       10380        TRA     BIND5           AND LOOP
     027751  000000 0650 15       10381 BIND6  ADX5    0,5             STEP TO NEXT SYMBOL
     027752  000000 2340 15       10382        SZN     0,5             SEE IF THERE ARE ANY MORE SYMBOLS
     027753  027711 6010 00       10383        TNZ     BIND4           TRANSFER IF MORE SYMBOLS
                                  10384 *      THIS IS A KLUDGE TO SET RUN TIME ENVIRONMENT
     027754  027755 5500 00       10385        SBAR    *+1
     027755  000000 2350 03       10386        LDA     **,DU
     027756  000011 7350 00       10387        ALS     9
     027757  002000 6250 01       10388        EAX5    1024,AU
     027760  500006 0010 00       10389        MME     M$MREQ
     027761  027762 5500 00       10390        SBAR    *+1
     027762  000000 2350 03       10391        LDA     **,DU
     027763  000011 7350 00       10392        ALS     9
     027764  027207 7550 00       10393        STA     R$MTOP
     027765  027207 2360 00       10394        LDQ     R$MTOP
     027766  027210 1760 00       10395        SBQ     R$MBASE
```

```
    END OF BINARY CARD 00000506
         027767  000300 0760 03       10396         ADQ      5*32+32,DU
         027770  000041 5060 07       10397         DIV      33,DL
         027771  777776 6210 02       10398         EAX1     -2,QU
         027772  027201 7410 00       10399         STX1     R$BITL
         027773  027207 2210 00       10400         LDX1     R$MTOP
         027774  027201 1610 00       10401         SBX1     R$BITL
         027775  027200 7410 00       10402         STX1     R$BITT
         027776  000002 1610 03       10403         SBX1     2,DU
         027777  027204 7410 00       10404         STX1     R$SMAX
         030000  030131 2350 00       10405         LDA      FREE
         030001  030140 2220 00       10406         LDX2     ENTRY
         030002  000001 2220 12       10407         LDX2     1,2
         030003  027211 7420 00       10408         STX2     R$REQ
         030004  026225 7500 00       10409         STC2     R$ZERX
         030005  026213 7100 00       10410         TRA      R$ZER
         030006  030140 2210 00       10411         LDX1     ENTRY             GET MAIN ENTRY POINTER IN XR - 1
         030007  777777 6000 00       10412         TZE      $ERROR            ERROR - NO BLANK SYMDEF
         030010  030131 2220 00       10413         LDX2     FREE              GET POINTER TO BASE OF STACK IN XR - 2
         030011  030131 2260 00       10414         LDX      S,FREE            INITIALIZE STACK POINTER
         030012  000001 0660 11       10415         ADX      S,1,1             ADD AMOUNT OF LOCAL STORAGE NEEDED BY PROGRAM
         030013  027204 1060 00       10416         CMPX     S,R$SMAX          DOES THIS OVERLAP THE HEAP
         030014  000002 6020 04       10417         TNC      2,IC              TRANSFER IF OK
    END OF BINARY CARD 00000507
         030015  777777 7000 00       10418         TSX0     $ERROR            GARBAGE COLLECT (LINK UP LLO'S)
         030016  000000 6350 11       10419         EAA      0,1               GET ADDRESS OF PROCEDURE VALUE IN AU
         030017  027223 0750 07       10420         ADA      T$PROC,DL         ADD TYPE IN AL
         030020  000000 7550 12       10421         STA      0,2               AND STORE AT BASE OF NEW STACK
         030021  000001 4500 12       10422         STZ      1,2               INITIALIZE S REGISTER SAVE WORD
         030022  000002 2350 11       10423         LDA      2,1               GET ENVIRONMENT OF PROCEDURE
         030023  000002 7550 12       10424         STA      2,2               AND STORE IN STACK
         030024  027210 2350 00       10425         LDA      R$MBASE           GET BASE OF STORAGE AS ENVIRONMENT
         030025  030042 0750 07       10426         ADA      TP,DL             ADD TYPE OF GLOBAL STORAGE
         030026  026540 7550 00       10427         STA      R$GD              AND STORE AS CURRENT ENVIRONMENT
         030027  000002 7420 11       10428         STX2     2,1               STORE OLD S REGISTER IN OLD STACK
         030030  000001 7200 11       10429         LXL0     1,1               GET TYPE OF NEW ENVIRONMENT IN XR - 0
         030031  000002 4400 11       10430         SXL0     2,1               AND MOVE IN MSCW
         030032  000001 4500 11       10431         STZ      1,1               ZERO OUT OLD TYPE FOR GARBAGE COLLECTOR
         030033  000000 6200 12       10432         EAX0     0,2               GET BASE OF NEW ENVIRONMENT IN XR - 0
         030034  027210 1600 00       10433         SBX0     R$MBASE           GET LENGTH OF CURRENT ENVIRONMENT IN XR - 0
         030035  030042 4400 00       10434         SXL0     TP                AND STORE AS LENGTH IN TYPE PATTERN
         030036  777777 6600 03       10435         ERX0     -1,DU             GET 1'S COMPLEMENT
         030037  030043 7400 00       10436         STX0     TP+1              AND STORE IN TERMINATOR WORD IN TYPE PATTERN
         030040  025365 7000 00       10437         TSX0     R$ENTER           AND ENTER PROGRAM
         030041  005653 7100 00       10438 TERM    TRA      P$STOP            PRINT READY MESSAGE AND STOP
         030042  027222 000000        10439 TP      ZERO     T$SKIP,0
    END OF BINARY CARD 00000508
         030043  000000 777777        10440         ZERO     0,-1
         030044  030046 0004 00       10441 SRTI    TALLY    SR,SRE-SR
         030045  000000 000000        10442 SRT     ZERO
```

```
           L                                                 PASS 3

     030046  030052 000000        10443 SR       ZERO      UC
     030047  030062 000000        10444          ZERO      XSYSC
     030050  030056 000000        10445          ZERO      US
     030051  030072 000000        10446          ZERO      LC
                       030052     10447 SRE      EQU       *
                       030052     10448          EVEN
     030052  030056 000000        10449 UC       ZERO      US
     030053  777777 000000        10450          ZERO      -1
     030054  056101114107         10451          UASCI     2,,ALGOL68
     030055  117114066070
     030056  777777 000000        10452 US       ZERO      -1
     030057  000002 000000        10453          ZERO      2
     030060  074125123105         10454          UASCI     2,<USER>
     030061  122076040040
     030062  030066 000000        10455 XSYSC    ZERO      *+4
     030063  777777 000000        10456          ZERO      -1
     030064  056101114107         10457          UASCI     2,,ALGOL68
     030065  117114066070
     030066  030102 000000        10458          ZERO      MFD
     030067  777777 000000        10459          ZERO      -1
     030070  130123131123         10460          UASCI     2,XSYSCAT
 END OF BINARY CARD 00000509
     030071  103101124040
     030072  030076 000000        10461 LC       ZERO      *+4
     030073  777777 000000        10462          ZERO      -1
     030074  056101114107         10463          UASCI     2,,ALGOL68
     030075  117114066070
     030076  030102 000000        10464          ZERO      MFD
     030077  777777 000000        10465          ZERO      -1
     030100  104114111102         10466          UASCI     2,DLIBRARY
     030101  122101122131
     030102  777777 000000        10467 MFD      ZERO      -1
     030103  000000 000000        10468          ZERO      0
     030104  074115106104         10469          UASCI     2,<MFD>
     030105  076040040040
                       030106     10470          BSS       16
                       030126     10471          EVEN
     030126  000000000000         10472 NAME     OCT       0,0
     030127  000000000000
     030130  000000 000000        10473 FRN      ZERO
     030131  000000 000000        10474 FREE     ZERO
     030132  000000000000         10475 CEL      OCT       0,0
     030133  000000000000
     030134  000000 000000        10476 SYMP     ZERO
     030135  027710 7100 00       10477 BINDI    TRA       BINDR
 END OF BINARY CARD 00000510
     030136  000000 000000        10478 PROG1    ZERO
     030137  040040040040         10479 BLNKS    UASCI     1,
     030140  000000 000000        10480 ENTRY    ZERO
     030141  000000 000000        10481 TEMP     ZERO
```

```
           L                                        PASS 3

      030142  000006710004
                          030150     10482          EIGHT
      030150  000000000000           10483 REG      OCT      0,0,0,0,0,0,0,0
      030151  000000000000
      030152  000000000000
      030153  000000000000
      030154  000000000000
      030155  000000000000
      030156  000000000000
      030157  000000000000
                                     10484 *
                                     10485 *        THE TABLE ALLOCATION ROUTINE ALLOCATES DYNAMIC SPACE
                                     10486 *        FOR THE SEVERAL TABLES REQUIRED DURING COMPILATION.
                                     10487 *        THE CALLING SEQUENCE IS A TSX0 WITH THE A REGISTER
                                     10488 *        CONTAINING THE LENGTH,DU OF THE NEW TABLE ENTRY
                                     10489 *        REQUESTED AND XR - 1 CONTAINING A POINTER TO THE
                                     10490 *        TABLE CONTROL WORD OF THE APPROPRIATE TABLE. THE
                                     10491 *        SUBROUTINE RETURNS IN XR - 1 A POINTER TO THE
                                     10492 *        NEWLY CREATED TABLE ENTRY. THE FORMAT OF A TABLE
                                     10493 *        CONTROL WORD IS UPPER HALF IS ABSOLUTE LOCATION
                                     10494 *        AND LOWER HALF IS THE CURRENT LENGTH OF THE TABLE.
                                     10495 *
      030160  030150 7530 00         10496 ALOC     SREG     REG              SAVE REGISTERS
      030161  030141 7420 00         10497          STX2     TEMP             SAVE LENGTH REQUESTED
      030162  000000 7220 11         10498          LXL2     0,1              GET CURRENT LENGTH OF TABLE
      030163  030141 0620 00         10499          ADX2     TEMP             ADD REQUESTED LENGTH TO GET NEW LENGTH
      030164  000000 4420 11         10500          SXL2     0,1              SAVE NEW LENGTH OF TABLE
      030165  000000 0620 11         10501          ADX2     0,1              ADD CURRENT LOCATION OF TABLE
      030166  000001 1620 11         10502          SBX2     1,1              SUBTRACT LOCATION OF ADJACIENT TABLE
      030167  030225 6040 00         10503          TMI      ALOC4            TRANSFER IF NO TABLE MOVING HAS TO BE DONE
   END OF BINARY CARD 00000511
      030170  000145 0620 03         10504          ADX2     1+100,DU         ADD ONE PLUS EXTRA SPACE TO BE ALLOCATED
      030171  030173 7420 00         10505          STX2     AINC             SAVE ADDITIONAL LENGTH NEEDED
      030172  030241 2230 00         10506          LDX3     TEND             GET ADDRESS OF END OF TABLES
      030173  000000 6240 13         10507 AINC     EAX4     **,3             GET ADDRESS OF NEW END OF TABLE IN XR - 4
      030174  030175 5500 00         10508          SBAR     *+1              STORE CURRENT LENGTH OF CORE
      030175  000000 2360 03         10509          LDQ      **,DU            AND PUT IT IN Q
      030176  000011 7360 00         10510          QLS      9                GET LENGTH IN QU
      030177  000000 6250 02         10511          EAX5     0,QU             PUT IN XR - 5 FOR HALFWORD OPERATION
      030200  030201 7450 00         10512          STX5     *+1              AND STORE FOR COMPARE
      030201  000000 1040 03         10513          CMPX4    **,DU            IS THERE ENOUGH CORE
      030202  030207 6020 00         10514          TNC      ALOC1            TRANSFER IF OK
      030203  000001 6250 14         10515          EAX5     1,4              GET LENGTH OF CORE NEEDED IN XR - 5
      030204  500006 0010 00         10516          MME      MSMREQ           AND TRY TO GET MORE CORE
      030205  000000 6250 15         10517          EAX5     0,5              DID WE GET IT
      030206  777777 6010 00         10518          TNZ      SERROR           NO - ERROR
      030207  000001 1040 11         10519 ALOC1    CMPX4    1,1              IS THERE ANY MORE TO MOVE
      030210  030216 6000 00         10520          TZE      ALOC2            TRANSFER IF DONE MOVING
      030211  000000 2360 13         10521          LDQ      0,3              FETCH FROM OLD TABLE LOCATION
      030212  000000 7560 14         10522          STQ      0,4              STORE IN NEW TABLE LOCATION
```

```
              L                                            PASS 3

        030213  000001 1630 03      10523          SBX3     1,DU              DECREMENT SOURCE POINTER
        030214  000001 1640 03      10524          SBX4     1,DU              DECREMENT DESTINATION POINTER
        030215  030207 7100 00      10525          TRA      ALOC1             AND TRY TO MOVE SOME MORE
   END OF BINARY CARD 00000512
        030216  030173 2230 00      10526 ALOC2    LDX3     AINC              GET DISPLACEMENT IN XR - 3
        030217  000001 6240 11      10527          EAX4     1,1               GET POINTER TO CONTROL WORD OF FIRST MOVED TABLE
        030220  000000 0430 14      10528 ALOC3    ASX3     0,4               MODIFY DESCRIPTOR TO POINT TO NEW TABLE LOCATION
        030221  000001 0640 03      10529          ADX4     1,DU              STEP TO NEXT DESCRIPTOR
        030222  030241 1040 03      10530          CMPX4    TEND,DU           ARE WE DONE INCREMENTING DESCRIPTORS
        030223  030220 6040 00      10531          TMI      ALOC3             TRANSFER IF MORE TO DO
        030224  030220 6000 00      10532          TZE      ALOC3             TRANSFER IF MORE TO DO
        030225  000000 7220 11      10533 ALOC4    LXL2     0,1               GET LENGTH OF TABLE
        030226  000000 0620 11      10534          ADX2     0,1               ADD CURRENT LOCATION
        030227  000000 4500 12      10535          STZ      0,2               ZERO OUT WORD FOLLOWING ENTRY
        030230  030141 1620 00      10536          SBX2     TEMP              SUBTRACT LENGTH OF NEW ELEMENT FOR ITS LOCATION
        030231  030150 4420 00      10537          SXL2     REG               PUT LOCATION OF NEW ELEMENT IN XR - 1
        030232  030150 0730 00      10538          LREG     REG               RESTORE REGISTERS
        030233  000000 7100 10      10539          TRA      0,0               AND RETURN
        030234  000000 000000       10540 USER     ZERO
        030235  000000 000000       10541 TYPE     ZERO
        030236  000000 000000       10542 SYM      ZERO
        030237  000000 000000       10543 CHAIN    ZERO
        030240  000000 000000       10544 PROG     ZERO
        030241  000000 000000       10545 TEND     ZERO
                                    10546          HEAD     1
                          030242    10547          BSS      100               PATCH SPACE
                          10548         TTLS    SYNTAX          PASS 1
                 000001   10549 SETE    SET     1
                          10550         PMC     OFF
                          10551         REF     OFF
                 030406   10552 PROG    SYNTAX  (PRO,A$PASS2,,2$PRO,2$DONE,)
 END OF BINARY CARD 00000513
                 030416   10553 PRO     SYNTAX  (PDEN,1$0DEL,),
                 030416   10554         ETC     (1$SRNGE,CLOR,1$CLEAR,IDNT,1$MRNGE,,SER,1$DECI,,*),
                 030416   10555         ETC     (1$ERNGE,CLOR)
                 030434   10556 COND    SYNTAX  (A$MATCH,S$BAR,1$DECI,,1$ERNGE,BARR,1$SRNGE,BARR,*),
                 030434   10557         ETC     (1$MRNGE,,SER,ELSE),
                 030434   10558         ETC     (A$MATCH,S$BARF,1$DECI,,1$ERNGE,BARR,1$SRNGE,*),
                 030434   10559         ETC     (BARR,1$MRNGE,,SER,THEN),
                 030434   10560         ETC     (1$EMPTY,)
 END OF BINARY CARD 00000514
                 030463   10561 THEN    SYNTAX  (A$MATCH,S$BAR,1$DECI,,1$ERNGE,BARR,1$SRNGE,BARR,*),
                 030463   10562         ETC     (1$MRNGE,,SER,ELSE),
                 030463   10563         ETC     (A$MATCH,S$BARF,1$DECI,,1$ERNGE,BARR,1$SRNGE,*),
                 030463   10564         ETC     (BARR,1$MRNGE,,SER,THEN),
                 030463   10565         ETC     (A$0ERR,A$ER8)
 END OF BINARY CARD 00000516
                 030512   10566 ELSE    SYNTAX  (A$MATCH,S$BAR,1$DECI,,1$ERNGE,BARR,1$SRNGE,BARR,*),
                 030512   10567         ETC     (1$MRNGE,,SER),
                 030512   10568         ETC     (A$MATCH,S$BARF,1$DECI,,1$ERNGE,BARR,1$SRNGE,*),
                 030512   10569         ETC     (BARR,1$MRNGE,,SER,THEN),
                 030512   10570         ETC     (1$EMPTY,)
 END OF BINARY CARD 00000517
                 030540   10571 SER     SYNTAX  (UNIT,UTAIL)
                 030546   10572 UTAIL   SYNTAX  (A$MATCH,S$SEMI,1$0ADO,SEMIC,UNIT,UTAIL),
                 030546   10573         ETC     (A$MATCH,S$COLON,1$0ADO,CLNC,1$DECLC,,UNIT,*),
                 030546   10574         ETC     (UTAIL),
                 030546   10575         ETC     (A$MATCH,S$PER,1$0ADO,SEMIC,1$IDENT,,*),
                 030546   10576         ETC     (A$MATCH,S$COLON,1$STID,,1$DECL,,UNIT,UTAIL),
                 030546   10577         ETC     (UTAIL)
 END OF BINARY CARD 00000518
                 030602   10578 UNIT    SYNTAX  (1$0STZ,IDNT,DEC)
                 030610   10579 JUNKE   SYNTAX  (JUNK),
                 030610   10580         ETC     (1$EMPTY,)
 END OF BINARY CARD 00000519
                 030620   10581 JUNK    SYNTAX  (1$IDENT,,1$STIDB,,JUNKE),
                 030620   10582         ETC     (A$MATCH,S$EQ,1$0SMO,IDNT,JUNK),
                 030620   10583         ETC     (A$MATCH,S$ASGN,1$0SMO,IDNT,JUNKE),
                 030620   10584         ETC     (PDEN,1$0DEL,,1$0SMO,IDNT,JUNKE),
                 030620   10585         ETC     (BOX,1$0DEL,,1$0SMO,IDNT,JUNKE),
                 030620   10586         ETC     (A$MATCH,S$STR,SUNIT,1$0DEL,,1$0SMO,IDNT,JUNKE),
                 030620   10587         ETC     (A$MATCH,S$UNION,UUNIT,1$0DEL,,1$0SMO,IDNT,*),
                 030620   10588         ETC     (JUNKE),
                 030620   10589         ETC     (A$MATCH,S$PROC,ARGE,1$0SMO,IDNT,JUNKE),
                 030620   10590         ETC     (A$MATCH,S$REF,1$0SMO,IDNT,JUNK),
                 030620   10591         ETC     (A$MATCH,S$LOC,1$0SMO,IDNT,JUNK),
          1                              SYNTAX           PASS 1

                     030620      10592         ETC      (A$MATCH,S$HEAP,1$0SMO,IDNT,JUNK),
                     030620      10593         ETC      (A$MATCH,S$OF,1$0SMO,IDNT,JUNK)
  END OF BINARY CARD 00000522
                     030725      10594 FJ      SYNTAX   (FORS),
                     030725      10595         ETC      (JUNKE)
                     030735      10596 JTAIL   SYNTAX   (A$MATCH,S$COMMA,1$0ADO,COMAC,1$0SMO,IDNT,JTL1),
                     030735      10597         ETC      (1$EMPTY,)
  END OF BINARY CARD 00000523
                     030750      10598 JTL1    SYNTAX   (FORS,JTAIL),
                     030750      10599         ETC      (JUNKE,JTAIL)
                     030762      10600 SUNIT   SYNTAX   (A$MATCH,S$LPAR,1$PUSH,IDNT,1$MARK,,FSEQ,*),
                     030762      10601         ETC      (A$MATCH,S$RPAR,1$0CST,,1$0POP,IDNT)
  END OF BINARY CARD 00000524
                     030775      10602 UUNIT   SYNTAX   (A$MATCH,S$LPAR,1$PUSH,IDNT,1$MARK,,MSEQ,*),
                     030775      10603         ETC      (A$MATCH,S$RPAR,1$0CUN,,1$0POP,IDNT)
                     031010      10604 FORS    SYNTAX   (A$MATCH,S$FOR,1$IDENT,,FROMS),
                     031010      10605         ETC      (FROMS)
  END OF BINARY CARD 00000525
                     031022      10606 FROMS   SYNTAX   (A$MATCH,S$FROM,JUNK,BYS),
                     031022      10607         ETC      (BYS)
                     031034      10608 BYS     SYNTAX   (A$MATCH,S$BY,JUNK,TOS),
                     031034      10609         ETC      (TOS)
  END OF BINARY CARD 00000526
                     031046      10610 TOS     SYNTAX   (A$MATCH,S$TO,JUNK,WHLES),
                     031046      10611         ETC      (WHLES)
                     031060      10612 WHLES   SYNTAX   (A$MATCH,S$WHLE,SER,DOS),
                     031060      10613         ETC      (DOS)
                     031072      10614 DOS     SYNTAX   (A$MATCH,S$DO,1$0SMO,IDNT,FJ)
  END OF BINARY CARD 00000527
                     031101      10615 BOX     SYNTAX   (A$MATCH,S$SUB,1$SRNGE,BOXR,BELEM,BTAIL,*),
                     031101      10616         ETC      (A$MATCH,S$BUS,1$CBOX,,1$ERNGE,BOXR,1$REST,CBOXT)
                     031115      10617 BTAIL   SYNTAX   (A$MATCH,S$COMMA,1$0ADO,COMAC,BELEM,BTAIL),
                     031115      10618         ETC      (1$EMPTY,)
  END OF BINARY CARD 00000528
                     031130      10619 BELEM   SYNTAX   (1$0STZ;IDNT,JUNKE,BEL2)
                     031137      10620 BEL2    SYNTAX   (A$MATCH,S$COLON,1$0ADO,CLNC,1$0STZ,IDNT,JUNKE,*),
                     031137      10621         ETC      (BEL3),
                     031137      10622         ETC      (A$MATCH,S$AT,1$0ADO,CLNC,1$0STZ,IDNT,JUNK),
                     031137      10623         ETC      (A$MATCH,S$FLEX,A$MATCH,S$COLON,1$0ADO,CLNC,*),
                     031137      10624         ETC      (1$0STZ;IDNT,JUNKE,BEL4),
                     031137      10625         ETC      (A$MATCH,S$EITH,A$MATCH,S$COLON,1$0ADO,CLNC,*),
                     031137      10626         ETC      (1$0STZ;IDNT,JUNKE,BEL4),
                     031137      10627         ETC      (1$0STZ;IDNT)
  END OF BINARY CARD 00000530
                     031201      10628 BEL3    SYNTAX   (A$MATCH,S$AT,JUNK),
                     031201      10629         ETC      (BEL4)
                     031212      10630 BEL4    SYNTAX   (A$MATCH,S$FLEX),
                     031212      10631         ETC      (A$MATCH,S$EITH),
                     031212      10632         ETC      (1$EMPTY,)
  END OF BINARY CARD 00000531
```

```
                    1                        SYNTAX          PASS 1
),
                              031225      10633 DEC     SYNTAX   (A$MATCH,S$PRIOR,1$IDENT,,1$STID,,A$MATCH,S$EQ,*),
),                            031225      10634         ETC      (PRT3),
                              031225      10635         ETC      (A$MATCH,S$OP,OPDEC),
                              031225      10636         ETC      (A$MATCH,S$MODE,1$IDENT,,1$STID,,A$MATCH,S$EQ,*),
                              031225      10637         ETC      (MT3),
                              031225      10638         ETC      (A$MATCH,S$STR,SDEC),
                              031225      10639         ETC      (A$MATCH,S$UNION,UDEC),
                              031225      10640         ETC      (A$MATCH,S$PROC,PDEC),
                              031225      10641         ETC      (A$MATCH,S$REF,RESLT,1$EREF,,IDEC),
                              031225      10642         ETC      (BOX,RESLT,1$EBOX,,IDEC),
                              031225      10643         ETC      (A$MATCH,S$LOC,RESLT,IDEC),
                              031225      10644         ETC      (A$MATCH,S$HEAP,RESLT,IDEC),
                              031225      10645         ETC      (A$MATCH,S$LIBRY,RESLT,1$SAVE,FMODE,1$DECLB,,*),
                              031225      10646         ETC      (LIBL),
                              031225      10647         ETC      (1$IDENT,,1$STID,,ID),
                              031225      10648         ETC      (JTL1)
            END OF BINARY CARD 00000534
                              031332      10649 PRT     SYNTAX   (A$MATCH,S$COMMA,1$0ADO;COMAC,PRT1),
                              031332      10650         ETC      (1$EMPTY,)
                              031344      10651 PRT1    SYNTAX   (1$IDENT,,1$STID,,PRT2),
                              031344      10652         ETC      (1$0SMO;IDNT,DEC)
            END OF BINARY CARD 00000535
                              031357      10653 PRT2    SYNTAX   (A$MATCH,S$EQ,PRT3),
                              031357      10654         ETC      (ID)
                              031370      10655 PRT3    SYNTAX   (1$PUSH;IDNT,1$IDENT,,1$CNVRT,,1$0POP,IDNT,*),
                              031370      10656         ETC      (1$DECP;,PRT)
            END OF BINARY CARD 00000536
                              031402      10657 OPDEC   SYNTAX   (A$MATCH,S$LPAR,1$MARK,,MSEQ,A$MATCH,S$RPAR,*),
                              031402      10658         ETC      (FIELD,1$0EPR,,1$DECO,,1$DELS,,A$MATCH,S$EQ,FJ,OPTL),
                              031402      10659         ETC      (1$IDENT,,1$STID,,1$EVOID,,1$DECO,,A$MATCH,S$EQ,*),
                              031402      10660         ETC      (PDEN,1$DECOP,,OPTL),
                              031402      10661         ETC      (A$MATCH,S$EQ,1$STID,,1$EVOID,,1$DECO,,*),
                              031402      10662         ETC      (A$MATCH,S$EQ,PDEN,1$DECOP,,OPTL),
                              031402      10663         ETC      (A$0ERR;A$ER3)
            END OF BINARY CARD 00000537
                              031450      10664 OPTL    SYNTAX   (A$MATCH,S$COMMA,1$0ADO,COMAC,OPT1),
                              031450      10665         ETC      (1$EMPTY,)
            END OF BINARY CARD 00000538
                              031462      10666 OPT1    SYNTAX   (1$IDENT,,1$STID,,OPT2);
                              031462      10667         ETC      (OPDEC);
                              031462      10668         ETC      (1$0SMO,IDNT,DEC)
                              031500      10669 OPT2    SYNTAX   (A$MATCH,S$EQ,JUNK,OPTL),
                              031500      10670         ETC      (ID)
            END OF BINARY CARD 00000539
                              031512      10671 MTAIL   SYNTAX   (A$MATCH,S$COMMA,1$0ADO,COMAC,MT1),
                              031512      10672         ETC      (1$EMPTY,)
                              031524      10673 MT1     SYNTAX   (1$IDENT,,1$STID,,MT2),
                              031524      10674         ETC      (1$0SMO;IDNT,DEC)
            END OF BINARY CARD 00000540
                              031537      10675 MT2     SYNTAX   (A$MATCH,S$EQ,MT3),
          1                         SYNTAX           PASS 1

                    031537     10676         ETC     (ID)
                    031550     10677 MT3     SYNTAX  (1$PUSH,IDNT,MOD,1$0POP,IDNT,1$DECM,,MTAIL)
  END OF BINARY CARD 00000541
                    031561     10678 SDEC    SYNTAX  (SUNIT,IDM),
                    031561     10679         ETC     (1$IDENT,,1$STID,,A$MATCH,S$EQ,SUNIT,1$DECM,,*),
                    031561     10680         ETC     (STAIL),
                    031561     10681         ETC     (A$0ERR,A$ER5)
                    031602     10682 STAIL   SYNTAX  (A$MATCH,S$COMMA,1$0ADO,COMAC,ST1),
                    031602     10683         ETC     (1$EMPTY,)
  END OF BINARY CARD 00000542
                    031614     10684 ST1     SYNTAX  (1$IDENT,,1$STID,,ST2),
                    031614     10685         ETC     (1$0SMO,IDNT,DEC)
                    031627     10686 ST2     SYNTAX  (A$MATCH,S$EQ,SUNIT,1$DECM,,STAIL),
                    031627     10687         ETC     (ID)
  END OF BINARY CARD 00000543
                    031642     10688 UDEC    SYNTAX  (UUNIT,IDM),
                    031642     10689         ETC     (1$IDENT,,1$STID,,A$MATCH,S$EQ,UUNIT,1$DECM,,*),
                    031642     10690         ETC     (UNT),
                    031642     10691         ETC     (A$0ERR,A$ER6)
  END OF BINARY CARD 00000544
                    031663     10692 UNT     SYNTAX  (A$MATCH,S$COMMA,1$0ADO,COMAC,UT1),
                    031663     10693         ETC     (1$EMPTY,)
                    031675     10694 UT1     SYNTAX  (1$IDENT,,1$STID,,UT2),
                    031675     10695         ETC     (1$0SMO,IDNT,DEC)
  END OF BINARY CARD 00000545
                    031710     10696 UT2     SYNTAX  (A$MATCH,S$EQ,UUNIT,1$DECM,,UNT),
                    031710     10697         ETC     (ID)
                    031723     10698 PDEC    SYNTAX  (1$IDENT,,1$STID,,PDEC1),
                    031723     10699         ETC     (1$MARK,,ARGE,RESLT,1$0EPR,,IDEC)
  END OF BINARY CARD 00000546
                    031741     10700 PDEC1   SYNTAX  (A$MATCH,S$EQ,1$PUSH,IDNT,PDEC2,1$0POP,IDNT,*),
                    031741     10701         ETC     (1$0TAG,,PTE),
                    031741     10702         ETC     (A$MATCH,S$ASGN,1$PUSH,IDNT,PDEC2,1$0POP,IDNT,*),
                    031741     10703         ETC     (1$EREF,,1$0TAG,,PTA),
                    031741     10704         ETC     (1$IDENT,,1$MARK,,1$0MMI,,1$0EPR,,1*STID,,IDEC),
                    031741     10705         ETC     (1$MARK,,1$EVOID,,1$0EPR,,IDEC)
  END OF BINARY CARD 00000547
                    032002     10706 PDEC2   SYNTAX  (PDEN),
                    032002     10707         ETC     (JUNKE,1$MARK,,1$EVOID,,1$0EPR,)
  END OF BINARY CARD 00000548
                    032015     10708 RESLT   SYNTAX  (A$MATCH,S$PROC,1$MARK,,ARGE,RESLT,1$0EPR,),
                    032015     10709         ETC     (BOX,RESLT,1$EBOX,),
                    032015     10710         ETC     (A$MATCH,S$REF,RESLT,1$EREF,),
                    032015     10711         ETC     (A$MATCH,S$STR,SUNIT,IDE),
                    032015     10712         ETC     (A$MATCH,S$UNION,UUNIT,IDE),
                    032015     10713         ETC     (1$IDENT,,1$STID,,RES1)
  END OF BINARY CARD 00000549
                    032057     10714 RES1    SYNTAX  (1$IDENT,,1$0MMI,,1$STID,),
                    032057     10715         ETC     (1$EVOID,)
  END OF BINARY CARD 00000550
```

```
                      032071      10716 IDE     SYNTAX   (1$IDENT,,1$STID,),
                      032071      10717         ETC      (1$0SMO,IDNT)
                      032102      10718 PTE     SYNTAX   (A$MATCH,S$COMMA,1$0ADD,COMAC,PTE1),
                      032102      10719         ETC      (1$EMPTY,)
 END OF BINARY CARD 00000551
                      032114      10720 PTE1    SYNTAX   (1$IDENT,,1$STID,,PTE2),
                      032114      10721         ETC      (1$0SMO,IDNT,DEC)
                      032127      10722 PTE2    SYNTAX   (A$MATCH,S$EQ,1$PUSH,IDNT,PDEN,1$0POP,IDNT,*),
                      032127      10723         ETC      (1$0TAG,,PTE),
                      032127      10724         ETC      (ID)
 END OF BINARY CARD 00000552
                      032144      10725 PTA     SYNTAX   (A$MATCH,S$COMMA,1$0ADD,COMAC,PTA1),
                      032144      10726         ETC      (1$EMPTY,)
                      032156      10727 PTA1    SYNTAX   (1$IDENT,,1$STID,,PTA2),
                      032156      10728         ETC      (1$0SMO,IDNT,DEC)
 END OF BINARY CARD 00000553
                      032171      10729 PTA2    SYNTAX   (A$MATCH,S$ASGN,1$PUSH,IDNT,PDEN,1$0POP,IDNT,*),
                      032171      10730         ETC      (1$EREF,,1$0TAG,,PTA),
                      032171      10731         ETC      (A$MATCH,S$COMMA,1$0ADD,COMAC,1$MARK,,1$EVOID,,*),
                      032171      10732         ETC      (1$0EPR,,1$0TAG,),
                      032171      10733         ETC      (ID)
 END OF BINARY CARD 00000554
                      032217      10734 MOD     SYNTAX   (A$MATCH,S$REF,MOD,1$EREF,),
                      032217      10735         ETC      (BOX,MOD1,1$EBOX,),
                      032217      10736         ETC      (A$MATCH,S$STR,SUNIT),
                      032217      10737         ETC      (A$MATCH,S$UNION,UUNIT),
                      032217      10738         ETC      (1$IDENT,,1$STID,,1$0MMI,),
                      032217      10739         ETC      (A$MATCH,S$PROC,1$MARK,,ARGE,MOD1,1$0EPR,)
 END OF BINARY CARD 00000555
                      032257      10740 ARGE    SYNTAX   (A$MATCH,S$LPAR,MSEQ,A$MATCH,S$RPAR),
                      032257      10741         ETC      (1$EMPTY,)
                      032271      10742 MOD1    SYNTAX   (MOD),
                      032271      10743         ETC      (1$EVOID,)
 END OF BINARY CARD 00000556
                      032301      10744 FIELD   SYNTAX   (A$MATCH,S$REF,1$0SMO,FMODE,FIELD,1$EREF,,1$SAVE,*),
                      032301      10745         ETC      (FMODE),
                      032301      10746         ETC      (BOX,1$0SMO,FMODE,FIELD,1$EBOX,,1$SAVE,FMODE),
                      032301      10747         ETC      (A$MATCH,S$STR,SUNIT,1$IDENT,,1$STID,,1$SAVE,*),
                      032301      10748         ETC      (FMODE),
                      032301      10749         ETC      (A$MATCH,S$UNION,UUNIT,1$IDENT,,1$STID,,1$SAVE,*),
                      032301      10750         ETC      (FMODE),
                      032301      10751         ETC      (A$MATCH,S$PROC,1$MARK,,ARGE,FLD2,1$0EPR,,1$SAVE,*),
                      032301      10752         ETC      (FMODE),
                      032301      10753         ETC      (1$IDENT,,1$STID,,FLD4,1$SAVE,FMODE),
                      032301      10754         ETC      (A$0ERR,A$ER7)
 END OF BINARY CARD 00000558
                      032360      10755 FLD2    SYNTAX   (1$IDENT,,1$STID,,FLD3),
                      032360      10756         ETC      (FIELD)
                      032372      10757 FLD3    SYNTAX   (1$IDENT,,1$0MMI,,1$STID,),
                      032372      10758         ETC      (1$EVOID,)
```

```
  END OF BINARY CARD 00000559
                         032404     10759 FLD4    SYNTAX   (1$IDENT,,1$0MMI,,1$STID,),
                         032404     10760         ETC      (1$REST,FMODE)
                         032416     10761 MSEQ    SYNTAX   (MOD,MSEQ1)
  END OF BINARY CARD 00000560
                         032424     10762 MSEQ1   SYNTAX   (A$MATCH,S$COMMA,MOD,MSEQ1),
                         032424     10763         ETC      (1$EMPTY,)
                         032436     10764 FSEQ    SYNTAX   (FIELD,1$0TAG,,FSEQ1)
                         032445     10765 FSEQ1   SYNTAX   (A$MATCH,S$COMMA,FIELD,1$0TAG,,FSEQ1),
                         032445     10766         ETC      (1$EMPTY,)
  END OF BINARY CARD 00000561
                         032460     10767 IDM     SYNTAX   (1$IDENT,,1$STID,,IDEC),
                         032460     10768         ETC      (JUNKE,1$0DEL,,JTAIL)
                         032474     10769 ID      SYNTAX   (1$IDENT,,1$0MMI,,1$STID,,IDEC),
                         032474     10770         ETC      (JUNKE,JTAIL)
  END OF BINARY CARD 00000562
                         032510     10771 CTAIL   SYNTAX   (A$MATCH,S$COMMA,1$0ADO,COMAC,DEC,CTAIL),
                         032510     10772         ETC      (1$EMPTY,)
                         032523     10773 IDEC    SYNTAX   (A$MATCH,S$EQ,1$SAVE,FMODE,1$0TAG,,JUNKE,ETL),
                         032523     10774         ETC      (A$MATCH,S$ASGN,1$EREF,,1$SAVE,FMODE,1$0TAG,,*),
                         032523     10775         ETC      (JUNKE,ATL),
                         032523     10776         ETC      (JUNK,1$0DEL,,JTAIL),
                         032523     10777         ETC      (1$EREF,,1$SAVE,FMODE,1$0TAG,,ATL)
  END OF BINARY CARD 00000564
                         032557     10778 ETL     SYNTAX   (A$MATCH,S$COMMA,1$0ADO,COMAC,ET1),
                         032557     10779         ETC      (1$0SMO,IDNT)
                         032571     10780 ET1     SYNTAX   (1$IDENT,,1$STID,,ET2),
                         032571     10781         ETC      (1$0SMO,IDNT,DEC)
  END OF BINARY CARD 00000565
                         032604     10782 ET2     SYNTAX   (A$MATCH,S$EQ,1$REST,FMODE,1$0TAG,,JUNKE,ETL),
                         032604     10783         ETC      (ID)
                         032620     10784 ATL     SYNTAX   (A$MATCH,S$COMMA,1$0ADO,COMAC,AT1),
                         032620     10785         ETC      (1$0SMO,IDNT)
  END OF BINARY CARD 00000566
                         032632     10786 AT1     SYNTAX   (1$IDENT,,1$STID,,AT2),
                         032632     10787         ETC      (1$0SMO,IDNT,DEC)
                         032645     10788 AT2     SYNTAX   (A$MATCH,S$ASGN,1$REST,FMODE,1$0TAG,,JUNKE,ATL),
                         032645     10789         ETC      (A$MATCH,S$COMMA,1$0ADO,COMAC,1$REST,FMODE,*),
                         032645     10790         ETC      (1$0TAG,,AT1),
                         032645     10791         ETC      (1$IDENT,,1$0MMI,,1$STID,,IDEC),
                         032645     10792         ETC      (JUNK,JTAIL),
                         032645     10793         ETC      (1$REST,FMODE,1$0TAG,)
  END OF BINARY CARD 00000568
                         032703     10794 LIBL    SYNTAX   (A$MATCH,S$COMMA,1$0ADO,COMAC,LIB1),
                         032703     10795         ETC      (1$0SMO,IDNT)
                         032715     10796 LIB1    SYNTAX   (1$IDENT,,1$STID,,LIB2),
                         032715     10797         ETC      (1$0SMO,IDNT,DEC)
  END OF BINARY CARD 00000569
                         032730     10798 LIB2    SYNTAX   (A$MATCH,S$COMMA,1$0ADO,COMAC,1$REST,FMODE,*),
                         032730     10799         ETC      (1$DECLB,,LIB1),
                       032730    10800         ETC      (1$IDENT,,1$0MMI,,1$STID,,IDEC),
                       032730    10801         ETC      (JUNK,JTAIL),
                       032730    10802         ETC      (1$REST,FMODE,1$DECLB,)
 END OF BINARY CARD 00000570
                       032757    10803 PDEN    SYNTAX   (A$MATCH,S$LPAR,1$SRNGE,CLOR,1$PUSH,MK,1$MRNGE,,*),
                       032757    10804         ETC      (SER,COND,A$MATCH,S$RPAR,MOD1,PBOD,1$EPDEN,)
                       032775    10805 PBOD    SYNTAX   (A$MATCH,S$COLON,JUNKE,1$0STZ,PDPOS),
                       032775    10806         ETC      (1$0SMO,PDPOS)
 END OF BINARY CARD 00000571
```

```
                          10807        TTLS     SYNTAX          PASS 2
                          10808        HEAD     2
                000001    10809 SETE   SET      1
                033007    10810 PRO    SYNTAX   (2$START,,CLO,2$VOID,,2$REST,FMODE,2$SVAL,,*),
                033007    10811        ETC      (2$STRNG,,DECX),
                033007    10812        ETC      (2$SRNGE,,2$0STZ,RNGE,SER,2$VOID,,2$EVOID,,*),
                033007    10813        ETC      (2$DELVY,2$ERNGE,)
  END OF BINARY CARD 00000572
                033033    10814 CLO    SYNTAX   (2$LPAR,PDEN,2$SRNGE,,2$INLL,,2$PUSH,T$CODE,*),
                033033    10815        ETC      (2$MARK,,FORMP,A$MATCH,S$RPAR,PMOD,2$PUSEA,*),
                033033    10816        ETC      (DECLR,2$0EPR,,2$PUSH,DECLR,A$MATCH,S$COLON,*),
                033033    10817        ETC      (QUAT,2$EPDEN,,2$ORLL,,2$ERNGE,,2$EPDNE,),
                033033    10818        ETC      (2$LPAR,FP+MCOM+ZCOL+ZSEM,2$SRNGE,,2$INLL,,*),
                033033    10819        ETC      (2$PUSH,T$CODE,QS,A$MATCH,S$RPAR,2$PARN,,*),
                033033    10820        ETC      (2$ORLL,,2$ERNGE,),
                033033    10821        ETC      (2$LPAR,FP+ZCOM+MCOL+ZSEM,2$SRNGE,,2$INLL,,PMOD,*),
                033033    10822        ETC      (A$MATCH,S$COLON,QUAT,A$MATCH,S$RPAR,2$SVAL,,*),
                033033    10823        ETC      (2$STRNG,,2$ORLL,,2$ERNGE,),
                033033    10824        ETC      (2$LPAR,FP,2$SRNGE,,2$INLL,,SER,A$MATCH,S$RPAR,*),
                033033    10825        ETC      (2$ORLL,,2$ERNGE,),
                033033    10826        ETC      (2$LPAR,,2$SRNGE,,2$INLL,,IF,A$MATCH,S$RPAR,*),
                033033    10827        ETC      (2$ORLL,,2$ERNGE,)
  END OF BINARY CARD 00000574
                033132    10828 RLTR   SYNTAX   (A$MATCH,S$CT,2$CSCT,),
                033132    10829        ETC      (A$MATCH,S$CTAB,2$CSCTB,)
  END OF BINARY CARD 00000575
                033144    10830 SER    SYNTAX   (2$PUSH,T$CODE,TRAIN,TRTL,2$BALN,)
                033154    10831 TRTL   SYNTAX   (A$MATCH,S$RER,2$0ADO,CNT,SER),
                033154    10832        ETC      (A$0OK,)
  END OF BINARY CARD 00000576
                033166    10833 TRAIN  SYNTAX   (UNIT,UTAIL)
                033174    10834 UTAIL  SYNTAX   (A$MATCH,S$SEMI,2$VOID,,2$DELV,,2$0DELV,,TRAIN),
                033174    10835        ETC      (A$0OK,)
  END OF BINARY CARD 00000577
                033210    10836 IF     SYNTAX   (2$CPAR,,2$PUSH,T$CODE,2$PUSH,T$CODE,2$CONF,,*),
                033210    10837        ETC      (TSEQ,RLTR,TERT,2$ENTL,B1,2$CONE,,2$CLEAR,CNT,*),
                033210    10838        ETC      (2$BALZR,,2$SVAL,,2$FIRM,,2$DELV,,2$0DELV,,THEN,*),
                033210    10839        ETC      (ELSE,2$BAL2,),
                033210    10840        ETC      (2$PUSH,T$CODE,SER,2$0DIF,,THEN,ELSE,2$BAL2,)
  END OF BINARY CARD 00000578
                033246    10841 THEN   SYNTAX   (2$0BAR,MCOM+ZCOL+ZSEM,2$XRNGE,,2$PUSH,T$CODE,*),
                033246    10842        ETC      (QSEQ,2$BALZR,),
                033246    10843        ETC      (2$0BAR,,2$XRNGE,,SER),
                033246    10844        ETC      (2$BARF,,2$XRNGE,,IF,2$ESKIP,)
  END OF BINARY CARD 00000579
                033272    10845 ELSE   SYNTAX   (2$0BAR,,2$XRNGE,,SER,2$ENTL,B2),
                033272    10846        ETC      (2$BARF,,2$XRNGE,,2$PUSH,B2X,IF,2$0POP,B2X,*),
                033272    10847        ETC      (2$ENTL,B2),
                033272    10848        ETC      (2$ESKIP,,2$ENTL,B2)
  END OF BINARY CARD 00000580
```

```
                     033316       10849 QSEQ   SYNTAX   (QUAT,2$ENTL,B2,QSEQT)
                     033325       10850 QSEQT  SYNTAX   (A$MATCH,S$COMMA,2$0ADO,CNT,QSEQ),
                     033325       10851        ETC      (A$OOOK,)
                     033337       10852 QS     SYNTAX   (QUAT,QST)
   END OF BINARY CARD 00000581
                     033345       10853 QST    SYNTAX   (A$MATCH,S$COMMA,2$0ADO,CNT,QS),
                     033345       10854        ETC      (A$OOOK,)
                     033357       10855 TSEQ   SYNTAX   (TERT,2$ENTL,B1,2$ENTC,,TSEQT)
   END OF BINARY CARD 00000582
                     033367       10856 TSEQT  SYNTAX   (A$MATCH,S$COMMA,2$0ADO,CNT,TSEQ),
                     033367       10857        ETC      (A$OOOK,)
                     033401       10858 UNIT   SYNTAX   (DEC),
                     033401       10859        ETC      (LSEQ,QUAT)
                     033412       10860 LSEQ   SYNTAX   (2$LABEL,,LSEQ),
                     033412       10861        ETC      (A$OOOK,)
   END OF BINARY CARD 00000583
                     033423       10862 QUAT   SYNTAX   (A$MATCH,S$FOR),
                     033423       10863        ETC      (TERT,QTAIL)
                     033434       10864 QTAIL  SYNTAX   (A$MATCH,S$ASGN,2$SASGN,,QUAT,2$DASGN,),
                     033434       10865        ETC      (A$MATCH,S$CT,2$0SCT,,TERT,2$0DCT,),
                     033434       10866        ETC      (A$MATCH,S$CTAB,2$SCTAB,,TERT,2$DCTAB,),
                     033434       10867        ETC      (A$MATCH,S$IS,2$0SIS,,TERT,2$IDNTY,),
                     033434       10868        ETC      (A$MATCH,S$ISNT,2$SISNT,,TERT,2$IDNTY,),
                     033434       10869        ETC      (A$OOOK,)
   END OF BINARY CARD 00000585
                     033477       10870 TERT   SYNTAX   (MFOR,STAIL)
                     033505       10871 STAIL  SYNTAX   (2$OPER,,2$0FOP,,TERT,STAIL),
                     033505       10872        ETC      (2$FOPS,)
   END OF BINARY CARD 00000586
                     033520       10873 MFOR   SYNTAX   (SEC),
                     033520       10874        ETC      (2$OPER,,2$WMOP,,MFOR,2$0MOP,)
                     033533       10875 SEC    SYNTAX   (2$TAGOF,,SEC,2$SELCT,),
                     033533       10876        ETC      (AMOD,2$SHEAP,,2$EREF,,2$WGEN,),
                     033533       10877        ETC      (A$MATCH,S$HEAP,AMOD,2$SHEAP,,2$EREF,,2$WGEN,),
                     033533       10878        ETC      (A$MATCH,S$LOC,AMOD,2$SLOC,,2$EREF,,2$WGEN,),
                     033533       10879        ETC      (PRIM),
                     033533       10880        ETC      (CLO,ACTP)
   END OF BINARY CARD 00000588
                     033575       10881 PRIM   SYNTAX   (A$MATCH,S$SKIP,2$ESKIP,),
                     033575       10882        ETC      (A$MATCH,S$NIL,2$ENIL,),
                     033575       10883        ETC      (A$MATCH,S$TRUE,2$ETRUE,),
                     033575       10884        ETC      (A$MATCH,S$FALSE,2$EFALS,),
                     033575       10885        ETC      (2$DENOT,,2$WDEN,,ITAIL),
                     033575       10886        ETC      (2$IDENT,,2$WIDEN,,ITAIL)
   END OF BINARY CARD 00000589
                     033631       10887 ITAIL  SYNTAX   (A$MATCH,S$SUB,2$SVAL,,2$WEAK,,2$0SUB,,2$SRNGE,,*),
                     033631       10888        ETC      (2$INLL,,INDEX,XTAIL,A$MATCH,S$BUS,2$0BUS,,*),
                     033631       10889        ETC      (2$0RLL,,2$ERNGE,),
                     033631       10890        ETC      (ACTP)
   END OF BINARY CARD 00000590
```

```
                     033654    10891 ACTP   SYNTAX  (2$LPAR,FP+ZCOL+ZSEM,2$SVAL,,2$FIRM,,2$SRNGE,,*),
                     033654    10892        ETC     (2$PUSH,TMODE,2$INLL,,2$PUSH,TSCODE,QS,*),
                     033654    10893        ETC     (A$MATCH,S$RPAR,2$PARN,,2$ORLL,,2$CALL,,*),
                     033654    10894        ETC     (2$ERNGE,,ACTP),
                     033654    10895        ETC     (A$OOOK,)
  END OF BINARY CARD 00000591
                     033701    10896 XTAIL  SYNTAX  (A$MATCH,S$COMMA,2$OADO,CNT,INDEX,XTAIL),
                     033701    10897        ETC     (A$OOOK,)
                     033714    10898 INDEX  SYNTAX  (BOUND,IX1),
                     033714    10899        ETC     (A$MATCH,S$COLON,IX2),
                     033714    10900        ETC     (IX5),
                     033714    10901        ETC     (2$EPTY,)
  END OF BINARY CARD 00000592
                     033734    10902 IX1    SYNTAX  (A$MATCH,S$COLON,2$OLWB,,IX3),
                     033734    10903        ETC     (IX5),
                     033734    10904        ETC     (2$SBCT,,2$OADO,SBCNT)
  END OF BINARY CARD 00000593
                     033752    10905 IX2    SYNTAX  (BOUND,2$OUPB,,IX4),
                     033752    10906        ETC     (IX5),
                     033752    10907        ETC     (2$EPTY,)
                     033767    10908 IX3    SYNTAX  (BOUND,2$OUPB,,IX4),
                     033767    10909        ETC     (IX4)
  END OF BINARY CARD 00000594
                     034001    10910 IX4    SYNTAX  (IX5),
                     034001    10911        ETC     (A$OOOK,)
                     034011    10912 IX5    SYNTAX  (A$MATCH,S$AT,BOUND,2$NLWB,)
                     034020    10913 BOUND  SYNTAX  (TERT,2$SINT,,2$DELV,)
  END OF BINARY CARD 00000595
                     034027    10914 VMOD   SYNTAX  (MOD,2$SVIRT,,2$CLEAR,DFLG)
                     034036    10915 AMOD   SYNTAX  (MOD,2$SACT,,2$CLEAR,DFLG)
                     034045    10916 MOD    SYNTAX  (BOX,MOD,2$EBOX,),
                     034045    10917        ETC     (A$MATCH,S$REF,2$PUSH,DFLG,VMOD,2$EREF,,2$OPOP,*),
                     034045    10918        ETC     (DFLG),
                     034045    10919        ETC     (A$MATCH,S$STR,SUNIT),
                     034045    10920        ETC     (A$MATCH,S$UNION,UUNIT),
                     034045    10921        ETC     (A$MATCH,S$PROC,2$PUSH,DFLG,PROCT,2$OPOP,DFLG),
                     034045    10922        ETC     (2$MIND,,2$SDCLR,)
  END OF BINARY CARD 00000597
                     034105    10923 PROCT  SYNTAX  (PROCM),
                     034105    10924        ETC     (PMOD,2$MARK,,2$PUSEA,DECLR,2$OEPR,)
                     034120    10925 PROCM  SYNTAX  (2$LPAR,,2$PUSH,DFLG,2$CLEAR,DFLG,2$MARK,,MSEQ,*),
                     034120    10926        ETC     (A$MATCH,S$RPAR,PMOD,2$PUSEA,DECLR,2$OEPR,,*),
                     034120    10927        ETC     (2$OPOP,DFLG)
  END OF BINARY CARD 00000598
                     034136    10928 PMOD   SYNTAX  (VMOD),
                     034136    10929        ETC     (2$EVOID,)
                     034146    10930 BOX    SYNTAX  (A$MATCH,S$SUB,2$SRNGE,,2$DSUB,,BOXER,BTAIL,*),
                     034146    10931        ETC     (2$CBOX,,A$MATCH,S$BUS,2$DBUS,,2$ERNGE,)
  END OF BINARY CARD 00000599
                     034163    10932 BOXER  SYNTAX  (BOUND,2$OLWB,,FOPT,A$MATCH,S$COLON,BOUND,*),
                    034163     10933           ETC      (2$OUPB,,FOPT,2$SACT,),
                    034163     10934           ETC      (A$MATCH,S$COLON,2$SVIRT,),
                    034163     10935           ETC      (2$SVIRT,)
 END OF BINARY CARD 00000600
                    034206     10936 FOPT      SYNTAX   (A$MATCH,S$FLEX,2$OFLEX,),
                    034206     10937           ETC      (2$OFIX,)
                    034217     10938 BTAIL     SYNTAX   (A$MATCH,S$COMMA,2$OADO,CNT,BOXER,BTAIL),
                    034217     10939           ETC      (A$OOOK,)
 END OF BINARY CARD 00000601
                    034232     10940 SUNIT     SYNTAX   (2$LPAR,,2$EVOID,,2$MARK,,FSEQ,A$MATCH,S$RPAR,*),
                    034232     10941           ETC      (2$OCST,)
                    034244     10942 UUNIT     SYNTAX   (2$LPAR,,2$PUSH,DFLG,2$CLEAR,DFLG,2$MARK,,MSEQ,*),
                    034244     10943           ETC      (A$MATCH,S$RPAR,2$OCUN,,2$OPOP,DFLG)
 END OF BINARY CARD 00000602
                    034260     10944 MSEQ      SYNTAX   (VMOD,2$PUSEA,DECLR,MODT)
                    034267     10945 MODT      SYNTAX   (A$MATCH,S$COMMA,MSEQ),
                    034267     10946           ETC      (A$OOOK,)
                    034300     10947 FSEQ      SYNTAX   (FIELD,2$PUSHW,DECLR,2$PUSHW,TG,FSEQ1)
 END OF BINARY CARD 00000603
                    034310     10948 FSEQ1     SYNTAX   (A$MATCH,S$COMMA,FSEQ),
                    034310     10949           ETC      (A$OOOK,)
                    034321     10950 FIELD     SYNTAX   (2$MIND,,FLD1),
                    034321     10951           ETC      (MOD,2$OTAG,),
                    034321     10952           ETC      (2$OTAG,)
 END OF BINARY CARD 00000604
                    034336     10953 FLD1      SYNTAX   (2$OTAG,,2$SDCLR,),
                    034336     10954           ETC      (A$OOOK,)
                    034347     10955 PARAM     SYNTAX   (QUAT)
                    034354     10956 PTAIL     SYNTAX   (A$MATCH,S$COMMA,2$OADO,CNT,PARAM,PTAIL),
                    034354     10957           ETC      (A$OOOK,)
 END OF BINARY CARD 00000605
                    034367     10958 FORMP     SYNTAX   (VMOD,2$IDENT,,2$PUSEA,DECLR,2$OFORM,,FTL)
                    034400     10959 FTL       SYNTAX   (A$MATCH,S$COMMA,FT1),
                    034400     10960           ETC      (A$OOOK,)
 END OF BINARY CARD 00000606
                    034411     10961 FT1       SYNTAX   (2$IDENT,,2$PUSEA,DECLR,2$OFORM,,FTL),
                    034411     10962           ETC      (FORMP)
                    034424     10963 DEC       SYNTAX   (A$MATCH,S$STR,2$CLEAR,DFLG,ST1),
                    034424     10964           ETC      (A$MATCH,S$UNION,UN1),
                    034424     10965           ETC      (A$MATCH,S$MODE,MD1),
                    034424     10966           ETC      (A$MATCH,S$PRIOR,PR1),
                    034424     10967           ETC      (A$MATCH,S$OP,OPDEC),
                    034424     10968           ETC      (A$MATCH,S$PROC,PDT),
                    034424     10969           ETC      (MOD,2$SBLNK,,IDEC),
                    034424     10970           ETC      (A$MATCH,S$HEAP,MOD,2$SHEAP,,IDEC),
                    034424     10971           ETC      (A$MATCH,S$LOC,MOD,2$SLOC,,IDEC),
                    034424     10972           ETC      (A$MATCH,S$LIBRY,VMOD,2$IDENT,,LIBL)
 END OF BINARY CARD 00000608
                    034506     10973 PDT       SYNTAX   (2$IDENT,,2$STID,,2$GMOD,,IDEC1),
                    034506     10974           ETC      (PROCT,2$SBLNK,,IDEC)
 END OF BINARY CARD 00000609
                        034523     10975 ST1     SYNTAX  (SUNIT,2$SBLNK,,IDEC),
                        034523     10976         ETC     (2$MIND,,A$MATCH,S$EQ,SUNIT,2$SACT,,2$CLEAR,DFLG,*),
                        034523     10977         ETC     (STL)
 END OF BINARY CARD 00000610
                        034542     10978 STL     SYNTAX  (A$MATCH,S$COMMA,ST2),
                        034542     10979         ETC     (A$MATCH,S$SEMI,TRAIN)
                        034554     10980 ST2     SYNTAX  (2$MIND,,A$MATCH,S$EQ,SUNIT,2$SACT,,2$CLEAR,DFLG,*),
                        034554     10981         ETC     (STL),
                        034554     10982         ETC     (DEC)
 END OF BINARY CARD 00000611
                        034571     10983 UN1     SYNTAX  (UUNIT,2$SBLNK,,IDEC),
                        034571     10984         ETC     (2$MIND,,A$MATCH,S$EQ,UUNIT,UTL)
                        034606     10985 UTL     SYNTAX  (A$MATCH,S$COMMA,UT2),
                        034606     10986         ETC     (A$MATCH,S$SEMI,TRAIN)
 END OF BINARY CARD 00000612
                        034620     10987 UT2     SYNTAX  (2$MIND,,A$MATCH,S$EQ,UUNIT,UTL),
                        034620     10988         ETC     (DEC)
                        034633     10989 MD1     SYNTAX  (2$MIND,,A$MATCH,S$EQ,AMOD,MTAIL)
 END OF BINARY CARD 00000613
                        034643     10990 MTAIL   SYNTAX  (A$MATCH,S$COMMA,MD2),
                        034643     10991         ETC     (A$MATCH,S$SEMI,TRAIN)
                        034655     10992 MD2     SYNTAX  (MD1),
                        034655     10993         ETC     (DEC)
                        034665     10994 PR1     SYNTAX  (2$0TAG,,A$MATCH,S$EQ,2$DENOT,,PRT)
 END OF BINARY CARD 00000614
                        034675     10995 PRT     SYNTAX  (A$MATCH,S$COMMA,PR2),
                        034675     10996         ETC     (A$MATCH,S$SEMI,TRAIN)
                        034707     10997 PR2     SYNTAX  (PR1),
                        034707     10998         ETC     (DEC)
 END OF BINARY CARD 00000615
                        034717     10999 OPDEC   SYNTAX  (PROCM,2$OPER,,2$STID,,2$OIDNY,,2$OIDN,,*),
                        034717     11000         ETC     (A$MATCH,S$EQ,QUAT,2$REST,FMODE,2$SVAL,,*),
                        034717     11001         ETC     (2$STRNG,,2$EVOID,,2$OENTR,,2$DELV,,OPTL),
                        034717     11002         ETC     (2$OPER,,2$STID,,2$OIDNY,,2$OIDN,,A$MATCH,S$EQ,*),
                        034717     11003         ETC     (QUAT,2$REST,FMODE,2$SVAL,,2$STRNG,,2$EVOID,,*),
                        034717     11004         ETC     (2$OENTR,,2$DELV,,OPTL)
 END OF BINARY CARD 00000616
                        034760     11005 OPTL    SYNTAX  (A$MATCH,S$COMMA,OPT1),
                        034760     11006         ETC     (A$MATCH,S$SEMI,TRAIN)
 END OF BINARY CARD 00000617
                        034772     11007 OPT1    SYNTAX  (OPDEC),
                        034772     11008         ETC     (DEC)
                        035002     11009 IDEC    SYNTAX  (2$IDENT,,2$STID,,2$SAVE,FMODE,IDEC1),
                        035002     11010         ETC     (2$SHPBK,,2$SACT,,2$CLEAR,DFLG,2$EREF,,2$WGEN,,*),
                        035002     11011         ETC     (STAIL,QTAIL)
 END OF BINARY CARD 00000618
                        035023     11012 IDEC1   SYNTAX  (A$MATCH,S$EQ,2$SVIRT,,2$CLEAR,DFLG,2$OIDN,,QUAT,*),
                        035023     11013         ETC     (2$REST,FMODE,2$SVAL,,2$STRNG,,DECX,ETL),
                        035023     11014         ETC     (2$SACT,,2$CLEAR,DFLG,2$EREF,,2$SAVE,FMODE,AT2)
```

```
           2                          SYNTAX          PASS 2

   END OF BINARY CARD 00000619
                            035050     11015 ETL     SYNTAX  (A$MATCH,S$COMMA,ET1),
                            035050     11016         ETC     (A$MATCH,S$SEMI,TRAIN)
                            035062     11017 ET1     SYNTAX  (2$IDENT,,2$STID,,A$MATCH,S$EQ,2$OIDN,,QUAT,*),
                            035062     11018         ETC     (2$REST,FMODE,2$SVAL,,2$STRNG,,DECX,ETL),
                            035062     11019         ETC     (DEC)
   END OF BINARY CARD 00000620
                            035103     11020 ATL     SYNTAX  (A$MATCH,S$COMMA,AT1),
                            035103     11021         ETC     (A$MATCH,S$SEMI,TRAIN)
                            035115     11022 AT1     SYNTAX  (2$IDENT,,2$STID,,AT2),
                            035115     11023         ETC     (DEC)
   END OF BINARY CARD 00000621
                            035127     11024 AT2     SYNTAX  (A$MATCH,S$ASGN,2$OIDN,,2$REST,FMODE,2$OASGN,,*),
                            035127     11025         ETC     (2$SLCBK,,2$WGEN,,QUAT,2$DEREF,,2$SVAL,,*),
                            035127     11026         ETC     (2$STRNG,,2$ASGNE,,2$DELV,,DECX,ATL),
                            035127     11027         ETC     (2$OIDN,,2$REST,FMODE,2$SLCBK,,2$WGEN,,DECX,ATL)
   END OF BINARY CARD 00000622
                            035161     11028 LIBL    SYNTAX  (A$MATCH,S$COMMA,LIB1),
                            035161     11029         ETC     (A$MATCH,S$SEMI,TRAIN)
                            035173     11030 LIB1    SYNTAX  (2$IDENT,,LIBL),
                            035173     11031         ETC     (DEC)
   END OF BINARY CARD 00000623
                            035204     11032 DECX    SYNTAX  (2$EVOID,,2$OENTR,,2$DELV,)
```

```
                 11033        TTLS    TABLES
                 11034 DEF    MACRO                     TABLE LETTER, ENTRY NAME, CONTENTS
                 11035        HEAD    T
                 11036        USE     #1
                 11037 SET    SET     0
                 11038        IDRP    #3
                 11039 SET    SET     SET+1
                 11040        IDRP
                 11041        ZERO    SET,#4
                 11042        HEAD    #1
                 11043        INE     '#2','
                 11044 #2     EQU     *-T$#1
                 11045        IDRP    #3
                 11046        VFD     #3
                 11047        IDRP
                 11048        USE     PREVIOUS
                 11049        ENDM    DEF
                 11050 DEFTYP MACRO   NAME,(,PARAM),(TYPE),LEN
                 11051        HEAD    T
                 11052        USE     T
                 11053 SET    SET     0
                 11054        IDRP    #3
                 11055 SET    SET     SET+1
                 11056        IDRP
                 11057        ZERO    SET
                 11058 SETSET SET     *-T
                 11059        IDRP    #3
                 11060        VFD     #3
                 11061        IDRP
                 11062        DEF     M,#1,(36/262144*PRIM+SETSET#2),#4
                 11063        ENDM    DEFTYP
                 11064        HEAD    T
  035237         11065        USE     WK
  035237         11066 WK     EQU     *
  035241         11067        USE     SK
  035241         11068 SK     EQU     *
  035243         11069        USE     M
  035243         11070 M      EQU     *
  036071         11071        USE     B
  036071         11072 B      EQU     *
  036072         11073        USE     D
  036072         11074 D      EQU     *
  037011         11075        USE     P
  037011         11076 P      EQU     *
  037012         11077        USE     S
  037012         11078 S      EQU     *
  037265         11079        USE     I
  037265         11080 I      EQU     *
  037404         11081        USE     C
  037404         11082 C      EQU     *
```

```
                         037405      11083            USE      L
                         037405      11084 L          EQU      *
                         037406      11085            USE      G
                         037406      11086 G          EQU      *
                         037407      11087            USE      T
                         037407      11088 T          EQU      *
                         037454      11089            USE      N
                         037454      11090 N          EQU      *
                         037455      11091            USE      Z
                         037455      11092 Z          EQU      *
                         040066      11093            USE      SD
                         040066      11094 SD         EQU      *
                         040067      11095            USE      E
                         040067      11096 E          EQU      *
                         035213      11097            USE
                         035213      11098            DEFTYP   VOID,,18/$ERROR,0
                         037411      11099            DEFTYP   BOOL,,18/OCT,1
   END OF BINARY CARD 00000624
                         037413      11100            DEFTYP   CHAR,,18/OCT,1
                         037415      11101            DEFTYP   INT,,18/OCT,1
                         037417      11102            DEFTYP   REAL,(,18/INT),18/OCT,1
   END OF BINARY CARD 00000625
                         037421      11103            DEFTYP   BITS,,18/OCT,1
                         037423      11104            DEFTYP   BYTES,,18/OCT,1
                         037425      11105            DEFTYP   LINT,,(18/OCT,18/OCT),2
                         037430      11106            DEFTYP   LREAL,(,18/LINT),(18/OCT,18/OCT),2
   END OF BINARY CARD 00000626
                         037433      11107            DEFTYP   LBITS,,(18/OCT,18/OCT),2
                         037436      11108            DEFTYP   LBYTS,,(18/OCT,18/OCT),2
                         037441      11109            DEFTYP   LBL,,18/$ERROR,0
   END OF BINARY CARD 00000627
                         037443      11110            DEFTYP   MS,,(18/PTR,18/PTR),2
                         000002      11111 MSL        EQU      2                  LENGTH OF MS VALUE
                         037446      11112            DEFTYP   MSCW,,(18/PTR,18/PTR,18/PTR,18/PTR),4
                         037453      11113            DEF      M,PTR,(18/REF,18/VOID)
   END OF BINARY CARD 00000628
                         037453      11114            DEF      M,QUAD,(18/STRCT,18/INT,18/INT,18/INT,18/INT)
                         037453      11115            DEF      M,PROCV,(18/PROC,18/VOID)
                         037453      11116            DEF      M,STRNG,(18/ROWE,18/CHAR)
                         037453      11117            DEF      M,COMPL,
                         037453      11118            ETC      (18/STRCT,36/262144*REAL+S$RE,36/262144*REAL+S$IM)
   END OF BINARY CARD 00000629
                         037453      11119            DEF      M,LCOMP,
                         037453      11120            ETC      (18/STRCT,36/262144*LREAL+S$RE,36/262144*LREAL+S$IM)
                         037453      11121            DEF      M,RBOOL,(18/ROWE,18/BOOL)
                         037453      11122            DEF      M,M20,(18/PROC,18/BOOL,18/BOOL)
                         037453      11123            DEF      M,M21,(18/PROC,18/BOOL,18/INT)
                         037453      11124            DEF      M,M22,(18/PROC,18/BOOL,18/BOOL,18/BOOL)
   END OF BINARY CARD 00000630
                         037453      11125            DEF      M,M23,(18/PROC,18/CHAR,18/INT)
```

```
           M                             TABLES

                        037453     11126          DEF       M,M24,(18/PROC,18/CHAR,18/CHAR,18/BOOL)
                        037453     11127          DEF       M,M25,(18/PROC,18/CHAR,18/CHAR,18/STRNG)
                        037453     11128          DEF       M,M26,(18/PROC,18/LREAL,18/LREAL)
 END OF BINARY CARD 00000631
                        037453     11129          DEF       M,M27,(18/PROC,18/LREAL,18/INT)
                        037453     11130          DEF       M,M28,(18/PROC,18/LREAL,18/LINT)
                        037453     11131          DEF       M,M29,(18/PROC,18/LREAL,18/REAL)
                        037453     11132          DEF       M,M30,(18/PROC,18/LREAL,18/LREAL,18/BOOL)
 END OF BINARY CARD 00000632
                        037453     11133          DEF       M,M31,(18/PROC,18/LREAL,18/LREAL,18/LREAL)
                        037453     11134          DEF       M,M32,(18/PROC,18/LREAL,18/LREAL,18/LCOMP)
                        037453     11135          DEF       M,M33,(18/PROC,18/LREAL,18/INT,18/LREAL)
                        037453     11136          DEF       M,M34,(18/PROC,18/INT,18/BOOL)
 END OF BINARY CARD 00000633
                        037453     11137          DEF       M,M35,(18/PROC,18/INT,18/CHAR)
                        037453     11138          DEF       M,M36,(18/PROC,18/INT,18/INT)
                        037453     11139          DEF       M,M37,(18/PROC,18/INT,18/LINT)
                        037453     11140          DEF       M,M38,(18/PROC,18/INT,18/BITS)
 END OF BINARY CARD 00000634
                        037453     11141          DEF       M,M39,(18/PROC,18/INT,18/INT,18/BOOL)
                        037453     11142          DEF       M,M40,(18/PROC,18/INT,18/INT,18/INT)
                        037453     11143          DEF       M,M41,(18/PROC,18/INT,18/INT,18/REAL)
                        037453     11144          DEF       M,M42,(18/PROC,18/INT,18/LBYTS,18/CHAR)
 END OF BINARY CARD 00000635
                        037453     11145          DEF       M,M43,(18/PROC,18/INT,18/BITS,18/BOOL)
                        037453     11146          DEF       M,M44,(18/PROC,18/INT,18/LBITS,18/BOOL)
                        037453     11147          DEF       M,M45,(18/PROC,18/INT,18/BYTES,18/CHAR)
                        037453     11148          DEF       M,M46,(18/PROC,18/LINT,18/BOOL)
 END OF BINARY CARD 00000636
                        037453     11149          DEF       M,M47,(18/PROC,18/LINT,18/INT)
                        037453     11150          DEF       M,M48,(18/PROC,18/LINT,18/LINT)
                        037453     11151          DEF       M,M49,(18/PROC,18/LINT,18/LBITS)
                        037453     11152          DEF       M,M50,(18/PROC,18/LINT,18/INT,18/LINT)
 END OF BINARY CARD 00000637
                        037453     11153          DEF       M,M51,(18/PROC,18/LINT,18/LINT,18/BOOL)
                        037453     11154          DEF       M,M52,(18/PROC,18/LINT,18/LINT,18/LREAL)
                        037453     11155          DEF       M,M53,(18/PROC,18/LINT,18/LINT,18/LINT)
                        037453     11156          DEF       M,M54,(18/PROC,18/LBYTS,18/LBYTS,18/BOOL)
 END OF BINARY CARD 00000638
                        037453     11157          DEF       M,M55,(18/PROC,18/REAL,18/LREAL)
                        037453     11158          DEF       M,M56,(18/PROC,18/REAL,18/INT)
                        037453     11159          DEF       M,M57,(18/PROC,18/REAL,18/REAL)
                        037453     11160          DEF       M,M58,(18/PROC,18/REAL,18/INT,18/REAL)
 END OF BINARY CARD 00000639
                        037453     11161          DEF       M,M59,(18/PROC,18/REAL,18/REAL,18/BOOL)
                        037453     11162          DEF       M,M60,(18/PROC,18/REAL,18/REAL,18/REAL)
                        037453     11163          DEF       M,M61,(18/PROC,18/REAL,18/REAL,18/COMPL)
                        037453     11164          DEF       M,M62,(18/PROC,18/STRNG,18/LBYTS)
 END OF BINARY CARD 00000640
                        037453     11165          DEF       M,M63,(18/PROC,18/STRNG,18/BYTES)
```

```
                     037453     11166         DEF       M,M64,(18/PROC,18/STRNG,18/STRNG,18/BOOL)
                     037453     11167         DEF       M,M65,(18/PROC,18/STRNG,18/STRNG,18/STRNG)
                     037453     11168         DEF       M,M66,(18/PROC,18/BITS,18/INT)
 END OF BINARY CARD 00000641
                     037453     11169         DEF       M,M67,(18/PROC,18/BITS,18/BITS)
                     037453     11170         DEF       M,M68,(18/PROC,18/BITS,18/INT,18/BITS)
                     037453     11171         DEF       M,M69,(18/PROC,18/BITS,18/BITS,18/BOOL)
                     037453     11172         DEF       M,M70,(18/PROC,18/BITS,18/BITS,18/BITS)
 END OF BINARY CARD 00000642
                     037453     11173         DEF       M,M71,(18/PROC,18/LBITS,18/LINT)
                     037453     11174         DEF       M,M72,(18/PROC,18/LBITS,18/LBITS)
                     037453     11175         DEF       M,M73,(18/PROC,18/LBITS,18/INT,18/LBITS)
                     037453     11176         DEF       M,M74,(18/PROC,18/LBITS,18/LBITS,18/BOOL)
 END OF BINARY CARD 00000643
                     037453     11177         DEF       M,M75,(18/PROC,18/LBITS,18/LBITS,18/LBITS)
                     037453     11178         DEF       M,M76,(18/PROC,18/COMPL,18/REAL)
                     037453     11179         DEF       M,M77,(18/PROC,18/COMPL,18/COMPL)
                     037453     11180         DEF       M,M78,(18/PROC,18/COMPL,18/LCOMP)
 END OF BINARY CARD 00000644
                     037453     11181         DEF       M,M79,(18/PROC,18/COMPL,18/INT,18/COMPL)
                     037453     11182         DEF       M,M80,(18/PROC,18/COMPL,18/COMPL,18/BOOL)
                     037453     11183         DEF       M,M81,(18/PROC,18/COMPL,18/COMPL,18/COMPL)
                     037453     11184         DEF       M,M82,(18/PROC,18/BYTES,18/BYTES,18/BOOL)
 END OF BINARY CARD 00000645
                     037453     11185         DEF       M,M83,(18/PROC,18/LCOMP,18/LREAL)
                     037453     11186         DEF       M,M84,(18/PROC,18/LCOMP,18/COMPL)
                     037453     11187         DEF       M,M85,(18/PROC,18/LCOMP,18/LCOMP)
                     037453     11188         DEF       M,M86,(18/PROC,18/LCOMP,18/INT,18/LCOMP)
 END OF BINARY CARD 00000646
                     037453     11189         DEF       M,M87,(18/PROC,18/LCOMP,18/LCOMP,18/BOOL)
                     037453     11190         DEF       M,M88,(18/PROC,18/LCOMP,18/LCOMP,18/LCOMP)
                     037453     11191         DEF       M,M89,(18/PROC,18/RBOOL,18/BITS)
                     037453     11192         DEF       M,M90,(18/PROC,18/RBOOL,18/LBITS)
 END OF BINARY CARD 00000647
                     037453     11193         DEF       M,M91,(18/PROC,18/REAL)
                     037453     11194         DEF       M,M92,(18/PROC,18/INT,18/REAL)
                     037453     11195         DEF       M,M93,(18/PROC,18/LINT,18/LREAL)
                     037453     11196         DEF       M,M94,(18/PROC,18/BOOL,18/BOOL)
 END OF BINARY CARD 00000648
                     037453     11197         DEF       M,M95,(18/PROC,18/CHAR,18/BOOL)
                     037453     11198         DEF       M,M96,(18/PROC,18/INT,18/BOOL)
                     037453     11199         DEF       M,M97,(18/PROC,18/REAL,18/BOOL)
                     037453     11200         DEF       D,INT,(36/0,18/MODE,18/MSINT,36/0)
                     037453     11201         DEF       D,REAL,(36/0,18/MODE,18/MSREAL,36/0)
 END OF BINARY CARD 00000649
                     037453     11202         DEF       D,BOOL,(36/0,18/MODE,18/MSBOOL,36/0)
                     037453     11203         DEF       D,CHAR,(36/0,18/MODE,18/MSCHAR,36/0)
                     037453     11204         DEF       D,BITS,(36/0,18/MODE,18/MSBITS,36/0)
 END OF BINARY CARD 00000650
                     037453     11205         DEF       D,BYTES,(36/0,18/MODE,18/MSBYTES,36/0)
```

```
                   037453              DEF      D,STRNG,(36/0,18/MODE,18/M$STRNG,36/0)
                   037453              DEF      D,TRUE,(36/0,18/IDENT,18/M$BOOL,36/0)
                   037453              DEF      D,FALSE,(36/0,18/IDENT,18/M$BOOL,36/0)
 END OF BINARY CARD 00000651
                   037453              DEF      D,PI,(18/0,18/IDENT,18/M$REAL,36/0)
                   037453              DEF      D,SQRT,(18/0,18/IDENT,18/M$M57,36/0)
                   037453              DEF      D,EXP,(18/0,18/IDENT,18/M$M57,36/0)
                   037453              DEF      D,LN,(18/0,18/IDENT,18/M$M57,36/0)
 END OF BINARY CARD 00000652
                   037453              DEF      D,COS,(18/0,18/IDENT,18/M$M57,36/0)
                   037453              DEF      D,ACOS,(18/0,18/IDENT,18/M$M57,36/0)
                   037453              DEF      D,SIN,(18/0,18/IDENT,18/M$M57,36/0)
 END OF BINARY CARD 00000653
                   037453              DEF      D,ASIN,(18/0,18/IDENT,18/M$M57,36/0)
                   037453              DEF      D,TAN,(18/0,18/IDENT,18/M$M57,36/0)
                   037453              DEF      D,ATAN,(18/0,18/IDENT,18/M$M57,36/0)
                   037453              DEF      D,RND,(18/0,18/IDENT,18/M$M91,36/0)
 END OF BINARY CARD 00000654
                   037453              DEF      D,LRND,(18/0,18/IDENT,18/M$REAL,36/0)
                   037453              DEF      D,LIBI,(36/0,18/IDENT,18/M$BOOL,36/0)
                                       LIST     ON
                                       HEAD     A
                   000002        ELEN  EQU      2
                   000000        SETW  SET      0
                   000000        SETC  SET      0
                                 DEFINE MACRO   (OCTAL STRING), NAME
                                       HEAD     S
                                       USE      S
                                 #2    EQU      *-T$S
                                       HEAD     A
                                       STAB     (#1)
                                       ENDM     DEFINE
                                 ENTER MACRO    (OCTAL STRING), NAME
                                       USE      D
                                 SETSET SET     *-T$D+1
                                       HEAD     S
                                       USE      S
                                 #2    EQU      *-T$S
                                       HEAD     A
                                       STAB     (#1),SETSET
                                       ENDM     ENTER
                                 STAB  MACRO    (OCTAL STRING), DEFINITION LINK
                                       REF      SAVE,OFF
                                 SETN  SET      0                INITIALIZE CHARACTER COUNT
                                 SETCI SET      SETC             SAVE INITIAL CHARACTER POSITION
                                 SETWI SET      SETW             SAVE INITIAL WORD POSITION
                                       IDRP     #1
                                 SETN  SET      SETN+1           INCREMENT CHARACTER COUNTER
                                       IFE      SETC,0
                                 SETCO SET      #1-#1/10*2-#1/100*16 PUT OCTAL CHARACTER IN SET SYMBOL
```

```
                            11252         IFE      SETC,1
                            11253 SETC1   SET      #1-#1/10*2-#1/100*16 PUT OCTAL CHARACTER IN SET SYMBOL
                            11254         IFE      SETC,2
                            11255 SETC2   SET      #1-#1/10*2-#1/100*16 PUT OCTAL CHARACTER IN SET SYMBOL
                            11256         IFE      SETC,3
                            11257 SETC3   SET      #1-#1/10*2-#1/100*16 PUT OCTAL CHARACTER IN SET SYMBOL
                            11258 SETC    SET      SETC+1            STEP CHARACTER POSITION
                            11259         IFE      SETC,4,7
                            11260         USE      I
                            11261         LIST     SAVE,ON
                            11262         VFD      9/SETC0,9/SETC1,9/SETC2,9/SETC3
                            11263         LIST     RESTORE
                            11264         USE      S
                            11265 SETW    SET      SETW+1
                            11266 SETC    SET      0
                            11267         IDRP
                            11268         LIST     SAVE,ON
                            11269         TALLYB   SETWI,SETN+1,SETCI
                            11270         IFE      '#2','
                            11271         ZERO     131072+*-T$S-1
                            11272         INE      '#2','
                            11273         ZERO     #2
                            11274         LIST     RESTORE
                            11275         REF      RESTORE
                            11276         ENDM     STAB
                            11277 FORCE   MACRO
                            11278         INE      SETC,0,10
                            11279 SETC3   SET      0
                            11280         INE      SETC,3,3
                            11281 SETC2   SET      0
                            11282         INE      SETC,2,1
                            11283 SETC1   SET      0
                            11284         USE      I
                            11285         LIST     SAVE,ON
                            11286         VFD      9/SETC0,9/SETC1,9/SETC2,9/SETC3
                            11287         LIST     RESTORE
                            11288         USE      S
                            11289         ENDM     FORCE
                   037453   11290         DEFINE   (074,117,120,105,122,101,124,117,122,076),ERM1
                   037014   11291         DEFINE   (074,111,116,104,111,103,101,124,111,117,116,076),ERM2
 END OF BINARY CARD 00000655
                   037016   11292         DEFINE   (074,111,104,105,116,124,111,106,111,105,122,076),ERM3
                   037020   11293         DEFINE   (074,104,105,116,117,124,101,124,111,117,116,076),ERM4
 END OF BINARY CARD 00000656
                   037022   11294         DEFINE   (074,124,101,107,040,117,106,076),ERM5
                   037024   11295         DEFINE   (056),PER
                   037026   11296         DEFINE   (072,075),ASGN
                   037030   11297         DEFINE   (072),COLON
                   037032   11298         DEFINE   (114,117,116,107),LONG
 END OF BINARY CARD 00000657
```

```
          A                             TABLES

                    037034     11299         DEFINE  (123,124,122,125,103,124),STR
                    037036     11300         DEFINE  (122,105,106),REF
                    037040     11301         DEFINE  (106,114,105,130),FLEX
                    037042     11302         DEFINE  (105,111,124,110,105,122),EITH
  END OF BINARY CARD 00000658
                    037044     11303         DEFINE  (120,122,117,103),PROC
                    037046     11304         DEFINE  (125,116,111,117,116),UNION
                    037050     11305         DEFINE  (115,117,104,105),MODE
                    037052     11306         DEFINE  (120,122,111,117,122,111,124,131),PRIOR
  END OF BINARY CARD 00000659
                    037054     11307         DEFINE  (114,117,103),LOC
                    037056     11308         DEFINE  (117,120),OP
                    037060     11309         DEFINE  (050),LPAR
                    037062     11310         DEFINE  (051),RPAR
                    037064     11311         DEFINE  (054),COMMA
  END OF BINARY CARD 00000660
                    037066     11312         DEFINE  (133),SUB
                    037070     11313         DEFINE  (135),BUS
                    037072     11314         DEFINE  (101,124),AT
                    037074     11315         DEFINE  (134),BAR
                    037076     11316         DEFINE  (117,106),OF
                    037100     11317         DEFINE  (073),SEMI
  END OF BINARY CARD 00000661
                    037102     11318         DEFINE  (110,105,101,120),HEAP
                    037104     11319         DEFINE  (114,111,102,122,101,122,131),LIBRY
                    037106     11320         DEFINE  (106,117,122),FOR
                    037110     11321         DEFINE  (106,122,117,115),FROM
  END OF BINARY CARD 00000662
                    037112     11322         DEFINE  (102,131),BY
                    037114     11323         DEFINE  (124,117),TO
                    037116     11324         DEFINE  (127,110,111,114,105),WHLE
                    037120     11325         DEFINE  (104,117),DO
                    037122     11326         DEFINE  (134,072),BARF
                    000112     11327 TABLE   EQU     *-T$S
                    037124     11328         DEFINE  (124,122,125,105),TRUE
  END OF BINARY CARD 00000663
                    037126     11329         DEFINE  (106,101,114,123,105),FALSE
                    037130     11330         DEFINE  (072,072),CT
                    037132     11331         DEFINE  (072,072,075),CTAB
                    037134     11332         DEFINE  (072,075,072),IS
                    037136     11333         DEFINE  (072,057,075,072),ISNT
  END OF BINARY CARD 00000664
                    037140     11334         DEFINE  (123,113,111,120),SKIP
                    037142     11335         DEFINE  (116,111,114),NIL
                    037144     11336         DEFINE  (122,105),RE
                    037146     11337         DEFINE  (111,115),IM
              760000 401003    11338 IFA     OPD     09/760,09/0,018/401003
              761000 401003    11339 INA     OPD     09/761,09/0,018/401003
              762000 401003    11340 IFB     OPD     09/762,09/0,018/401003
              763000 401003    11341 INB     OPD     09/763,09/0,018/401003
          A                          TABLES

          764000 401003       11342 JMP    OPD     09/764,09/0,018/401003
          765000 400003       11343 STOP   OPD     09/765,09/0,018/400003
          236100 401003       11344 LDQ,   OPD     036/236100401003
          336100 401003       11345 LCQ,   OPD     036/336100401003
          076100 401003       11346 ADQ,   OPD     036/076100401003
          176100 401003       11347 SBQ,   OPD     036/176100401003
          773100 401003       11348 LRL,   OPD     036/773100401003
          402100 401003       11349 MPY,   OPD     036/402100401003
          506100 401003       11350 DIV,   OPD     036/506100401003
          276100 401003       11351 ORQ,   OPD     036/276100401003
          376100 401003       11352 ANQ,   OPD     036/376100401003
          676100 401003       11353 ERQ,   OPD     036/676100401003
          431100 401003       11354 FLD,   OPD     036/431100401003
          513100 400003       11355 FNEG,  OPD     036/513100400003
          573100 400003       11356 FNO,   OPD     036/573100400003
          475100 401003       11357 FAD,   OPD     036/475100401003
          575100 401003       11358 FSB,   OPD     036/575100401003
          461100 401003       11359 FMP,   OPD     036/461100401003
          565100 401003       11360 FDV,   OPD     036/565100401003
          525100 401003       11361 FDI,   OPD     036/525100401003
          700100 401003       11362 TSX0,  OPD     036/700100401003
                              11363 DEFOP  MACRO   MODE
                              11364        USE     Z
                              11365 SETSET SET     *-T$Z
                              11366        DEF     D,,(18/*+5-T$D,18/OP,36/M$#1*262144+1,18/SETSET)
                              11367        USE     Z
                              11368        HEAD    Z
                              11369        ENDM    DEFOP
                              11370 IDENT  MACRO   MODE
                              11371        USE     S
                              11372 SETSET SET     *-T$S-A$ELEN+131072
                              11373        DEF     D,,(18/SETSET,18/IDENT,18/M$#1,36/0)
                              11374        ENDM    IDENT
                              11375 MODE   MACRO   MODE
                              11376        USE     S
                              11377 SETSET SET     *-T$S-A$ELEN+131072
                              11378        DEF     D,,(18/SETSET,18/MODE,18/M$#1,36/0)
                              11379        ENDM    MODE
                              11380 PRIOR  MACRO   PRIORITY
                              11381        USE     S
                              11382 SETSET SET     *-T$S-A$ELEN+131072
                              11383        DEF     D,,(18/SETSET,18/PRIOR,18/#1)
                              11384        ENDM    PRIOR
                              11385 OP     MACRO   MODE
                              11386        USE     S
                              11387 SETSET SET     *-T$S-A$ELEN+131072
                              11388        DEF     D,,(18/SETSET,18/OP,18/M$#1,36/0)
                              11389        ENDM    OP
                    037150    11390        ENTER   (040),BLANK
   END OF BINARY CARD 00000665
```

```
                          036250     11391           USE      D
                                     11392           HEAD     D
                          000157     11393 BLANK     EQU      *-T$D+1
                          036250     11394           IDENT    PROCV
                          037152     11395           ENTER    (111,116,124),INT
                          037154     11396           MODE     INT
                          037154     11397           ENTER    (122,105,101,114),REAL
  END OF BINARY CARD 00000666
                          037156     11398           MODE     REAL
                          037156     11399           ENTER    (102,117,117,114),BOOL
                          037160     11400           MODE     BOOL
                          037160     11401           ENTER    (103,110,101,122),CHAR
  END OF BINARY CARD 00000667
                          037162     11402           MODE     CHAR
                          037162     11403           ENTER    (102,111,124,123),BITS
                          037164     11404           MODE     BITS
                          037164     11405           ENTER    (102,131,124,105,123),BYTES
  END OF BINARY CARD 00000668
                          037166     11406           MODE     BYTES
                          037166     11407           ENTER    (123,124,122,111,116,107),STRNG
                          037170     11408           MODE     STRNG
                          037170     11409           ENTER    (120,111),PI
  END OF BINARY CARD 00000669
                          037172     11410           IDENT    REAL
                          037172     11411           ENTER    (123,121,122,124),SQRT
                          037174     11412           IDENT    M57
                          037174     11413           ENTER    (105,130,120),EXP
  END OF BINARY CARD 00000670
                          037176     11414           IDENT    M57
                          037176     11415           ENTER    (114,116),LN
                          037200     11416           IDENT    M57
                          037200     11417           ENTER    (103,117,123),COS
                          037202     11418           IDENT    M57
  END OF BINARY CARD 00000671
                          037202     11419           ENTER    (101,122,103,103,117,123),ACOS
                          037204     11420           IDENT    M57
                          037204     11421           ENTER    (123,111,116),SIN
                          037206     11422           IDENT    M57
  END OF BINARY CARD 00000672
                          037206     11423           ENTER    (101,122,103,123,111,116),ASIN
                          037210     11424           IDENT    M57
                          037210     11425           ENTER    (124,101,116),TAN
  END OF BINARY CARD 00000673
                          037212     11426           IDENT    M57
                          037212     11427           ENTER    (101,122,103,124,101,116),ATAN
                          037214     11428           IDENT    M57
                          037214     11429           ENTER    (122,101,116,104,117,115),RND
  END OF BINARY CARD 00000674
                          037216     11430           IDENT    M91
                          037216     11431           ENTER    (114,101,123,124,122,101,116,104,117,115),LRND
```

```
                  037220     11432           IDENT    REAL
                  036414     11433           USE      D
                             11434           HEAD     3
                  000323     11435 IR        EQU      *+1-TSD
                  036414     11436           DEFOP    M92
 END OF BINARY CARD 00000675
     037455  000004 7620 04 3746111437           IFB      *+4,$
     037456  216000 4310 03      11438           FLD      =71B25,DU
     037457  017442 2360 00      11439           LDQ      B
     037460  000003 7640 04 3746311440           JMP      *+3,$
     037461  000000 2350 07      11441           LDA      0,DL
     037462  216000 4110 03      11442           LDE      =71B25,DU
     037463  000000 5731 00      11443           FNO,
                  036421     11444           USE      D
                             11445           HEAD     3
                  000330     11446 LIR       EQU      *+1-TSD
                  036421     11447           DEFOP    M93
     037464  000002 7620 04 3746611448           IFB      *+2,$
     037465  017442 2370 00      11449           LDAQ     B
 END OF BINARY CARD 00000676
     037466  216000 4110 03      11450           LDE      =71B25,DU
     037467  000000 5731 00      11451           FNO,
                  037470     11452           ENTER    (053),PSGN
                  037222     11453           DEFOP    M36
     037470  000002 7620 04 3747211454           IFB      *+2,$
     037471  017442 2361 00      11455           LDQ,     B
     037472  000000 7650 00      11456           STOP
                  037473     11457           DEFOP    M40
 END OF BINARY CARD 00000677
     037473  000002 7630 04 3747511458           INB      *+2,$
     037474  017441 0761 00      11459           ADQ,     A
     037475  000002 7600 04 3747711460           IFA      *+2,$
     037476  017441 2360 00      11461           LDQ      A
     037477  017442 0761 00      11462           ADQ,     B
                  037500     11463           DEFOP    M57
     037500  000002 7620 04 3750211464           IFB      *+2,$
     037501  017442 4311 00      11465           FLD,     B
     037502  000000 7650 00      11466           STOP
                  037503     11467           DEFOP    M60
 END OF BINARY CARD 00000678
     037503  000002 7630 04 3750511468           INB      *+2,$
     037504  017441 4751 00      11469           FAD,     A
     037505  000002 7600 04 3750711470           IFA      *+2,$
     037506  017441 4310 00      11471           FLD      A
     037507  017442 4751 00      11472           FAD,     B
                  037510     11473           PRIOR    6
                  037222     11474           ENTER    (055),MSGN
                  037224     11475           DEFOP    M36
 END OF BINARY CARD 00000679
     037510  000002 7620 04 3751211476           IFB      *+2,$
```

```
                Z                          TABLES

     037511  017442 3361 00        11477           LCQ,     B
     037512  017443 7560 00        11478           STQ      T
     037513  017443 3361 00        11479           LCQ,     T
                        037514     11480           DEFOP    M40
     037514  000004 7630 04 3752011481           INB      *+4,$
     037515  017443 7560 00        11482           STQ      T
     037516  017441 2360 00        11483           LDQ      A
     037517  017443 1761 00        11484           SBQ,     T
     037520  000002 7600 04 3752211485           IFA      *+2,$
     037521  017441 2360 00        11486           LDQ      A
     037522  017442 1761 00        11487           SBQ,     B
                        037523     11488           DEFOP    M57
  END OF BINARY CARD 00000680
     037523  000002 7620 04 3752511489           IFB      *+2,$
     037524  017442 4310 00        11490           FLD      B
     037525  000000 5131 00        11491           FNEG,
                        037526     11492           DEFOP    M60
     037526  000004 7630 04 3753211493           INB      *+4,$
     037527  017443 4550 00        11494           FST      T
     037530  017441 4310 00        11495           FLD      A
     037531  017443 5751 00        11496           FSB,     T
     037532  000002 7600 04 3753411497           IFA      *+2,$
     037533  017441 4310 00        11498           FLD      A
     037534  017442 5751 00        11499           FSB,     B
                        037535     11500           PRIOR    6
  END OF BINARY CARD 00000681
                        037224     11501           ENTER    (052),ASTER
                        037226     11502           DEFOP    M40
     037535  000002 7630 04 3753711503           INB      *+2,$

     037536  017441 4021 00        11504           MPY,     A
     037537  000002 7600 04 3754111505           IFA      *+2,$
     037540  017441 2360 00        11506           LDQ      A
     037541  017442 4021 00        11507           MPY,     B
                        037542     11508           DEFOP    M60
  END OF BINARY CARD 00000682
     037542  000002 7630 04 3754411509           INB      *+2,$
     037543  017441 4611 00        11510           FMP,     A
     037544  000002 7600 04 3754611511           IFA      *+2,$
     037545  017441 4310 00        11512           FLD      A
     037546  017442 4611 00        11513           FMP,     B
                        037547     11514           PRIOR    7
                        037226     11515           ENTER    (057),SLASH
                        037230     11516           DEFOP    M41
  END OF BINARY CARD 00000683
     037547  000015 7600 04 3756411517           IFA      *+13,$
     037550  000004 7620 04 3755411518           IFB      *+4,$
     037551  216000 4310 03        11519           FLD      =71B25,DU
     037552  017442 2360 00        11520           LDQ      B
     037553  000003 7640 04 3755611521           JMP      *+3,$
     037554  000000 2350 07        11522           LDA      0,DL
```

```
      037555  216000 4110 03       11523          LDE      =71B25,DU
      037556  000000 5730 00       11524          FNO
      037557  017443 4700 00       11525          FSTR     T
      037560  216000 4310 03       11526          FLD      =71B25,DU
      037561  017441 2360 00       11527          LDQ      A
      037562  000000 5730 00       11528          FNO
      037563  017443 5651 00       11529          FDV,     T
      037564  000000 2350 07       11530          LDA      0,DL
      037565  216000 4110 03       11531          LDE      =71B25,DU
      037566  000000 5730 00       11532          FNO
      037567  017443 4700 00       11533          FSTR     T
      037570  216000 4310 03       11534          FLD      =71B25,DU
      037571  017442 2360 00       11535          LDQ      B
      037572  000000 5730 00       11536          FNO
      037573  017443 5251 00       11537          FDI,     T
                          037574   11538          DEFOP    M60
    END OF BINARY CARD 00000684
      037574  000002 7630 04 3757611539          INB      *+2,$
      037575  017441 5251 00       11540          FDI,     A
      037576  000002 7600 04 3760011541          IFA      *+2,$
      037577  017441 4310 00       11542          FLD      A
      037600  017442 5651 00       11543          FDV,     B
                          037601   11544          PRIOR    7
                          037230   11545          ENTER    (101,116,104),AND
                          037232   11546          DEFOP    M22
    END OF BINARY CARD 00000685
      037601  000002 7630 04 3760311547          INB      *+2,$
      037602  017441 3761 00       11548          ANQ,     A
      037603  000002 7600 04 3760511549          IFA      *+2,$
      037604  017441 2360 00       11550          LDQ      A
      037605  017442 3761 00       11551          ANQ,     B
                          037606   11552          PRIOR    3
                          037232   11553          ENTER    (117,122),OR
                          037234   11554          DEFOP    M22
    END OF BINARY CARD 00000686
      037606  000002 7630 04 3761011555          INB      *+2,$
      037607  017441 2761 00       11556          ORQ,     A
      037610  000002 7600 04 3761211557          IFA      *+2,$
      037611  017441 2360 00       11558          LDQ      A
      037612  017442 2761 00       11559          ORQ,     B
                          037613   11560          PRIOR    2
                          037234   11561          ENTER    (116,117,124),NOT
                          037236   11562          DEFOP    M20
    END OF BINARY CARD 00000687
      037613  000002 7620 04 3761511563          IFB      *+2,$
      037614  017442 2360 00       11564          LDQ      B
      037615  000001 6761 07       11565          ERQ,     1,DL
                          037616   11566          PRIOR    10
                          037236   11567          ENTER    (075),EQ
                          037240   11568          DEFOP    M22
```

```
  END OF BINARY CARD 00000688
      037616  000003 7630 04 3762111569         INB      *+3,$
      037617  017441 6760 00       11570        ERQ      A
      037620  000001 6761 07       11571        ERQ,     1,DL
      037621  000002 7600 04 3762311572         IFA      *+2,$
      037622  017441 2360 00       11573        LDQ      A
      037623  017442 6760 00       11574        ERQ      B
      037624  000001 6761 07       11575        ERQ,     1,DL
                           037625  11576        DEFOP    M39
      037625  000003 7630 04 3763011577         INB      *+3,$
      037626  017441 1160 00       11578        CMPQ     A
      037627  000004 7640 04 3763311579         JMP      *+4,$
      037630  000002 7600 04 3763211580         IFA      *+2,$
      037631  017441 2360 00       11581        LDQ      A
      037632  017442 1160 00       11582        CMPQ     B
      037633  000003 6010 04 3763611583         TNZ      *+3,$
      037634  000001 2360 07       11584        LDQ      1,DL
  END OF BINARY CARD 00000689
      037635  000002 7100 04 3763711585         TRA      *+2,$
      037636  000000 2361 07       11586        LDQ,     0,DL
                           037637  11587        DEFOP    M59
      037637  000003 7630 04 3764211588         INB      *+3,$
      037640  017441 5150 00       11589        FCMP     A
      037641  000004 7640 04 3764511590         JMP      *+4,$
      037642  000002 7600 04 3764411591         IFA      *+2,$
      037643  017441 4310 00       11592        FLD      A
      037644  017442 5150 00       11593        FCMP     B
      037645  000003 6010 04 3765011594         TNZ      *+3,$
      037646  000001 2360 07       11595        LDQ      1,DL
      037647  000002 7100 04 3765111596         TRA      *+2,$
      037650  000000 2361 07       11597        LDQ,     0,DL
                           037651  11598        PRIOR    4
  END OF BINARY CARD 00000690
                           037240  11599        ENTER    (057,075),NE
                           037242  11600        DEFOP    M22
      037651  000002 7630 04 3765311601         INB      *+2,$
      037652  017441 6761 00       11602        ERQ,     A
      037653  000002 7600 04 3765511603         IFA      *+2,$
      037654  017441 2360 00       11604        LDQ      A
      037655  017442 6761 00       11605        ERQ,     B
                           037656  11606        DEFOP    M39
  END OF BINARY CARD 00000691
      037656  000003 7630 04 3766111607         INB      *+3,$
      037657  017441 1160 00       11608        CMPQ     A
      037660  000004 7640 04 3766411609         JMP      *+4,$
      037661  000002 7600 04 3766311610         IFA      *+2,$
      037662  017441 2360 00       11611        LDQ      A
      037663  017442 1160 00       11612        CMPQ     B
      037664  000003 6000 04 3766711613         TZE      *+3,$
      037665  000001 2360 07       11614        LDQ      1,DL
      037666  000002 7100 04 3767011615         TRA     *+2,$
      037667  000000 2361 07       11616         LDQ,    0,DL
                     037670       11617         DEFOP   M59
      037670  000003 7630 04 3767311618         INB     *+3,$
  END OF BINARY CARD 00000692
      037671  017441 5150 00       11619         FCMP    A
      037672  000004 7640 04 3767611620         JMP     *+4,$
      037673  000002 7600 04 3767511621         IFA     *+2,$
      037674  017441 4310 00       11622         FLD     A
      037675  017442 5150 00       11623         FCMP    B
      037676  000003 6000 04 3770111624         TZE     *+3,$
      037677  000001 2360 07       11625         LDQ     1,DL
      037700  000002 7100 04 3770211626         TRA     *+2,$
      037701  000000 2361 07       11627         LDQ,    0,DL
                     037702       11628         PRIOR   4
                     037242       11629         ENTER   (074),LT
                     037244       11630         DEFOP   M39
  END OF BINARY CARD 00000693
      037702  000005 7630 04 3770711631         INB     *+5,$
      037703  017443 7560 00       11632         STQ     T
      037704  017441 2360 00       11633         LDQ     A
      037705  017443 1160 00       11634         CMPQ    T
      037706  000004 7640 04 3771211635         JMP     *+4,$
      037707  000002 7600 04 3771111636         IFA     *+2,$
      037710  017441 2360 00       11637         LDQ     A
      037711  017442 1160 00       11638         CMPQ    B
      037712  000003 6050 04 3771511639         TPL     *+3,$
      037713  000001 2360 07       11640         LDQ     1,DL
      037714  000002 7100 04 3771611641         TRA     *+2,$
      037715  000000 2361 07       11642         LDQ,    0,DL
                     037716       11643         DEFOP   M59
      037716  000005 7630 04 3772311644         INB     *+5,$
  END OF BINARY CARD 00000694
      037717  017443 4550 00       11645         FST     T
      037720  017441 4310 00       11646         FLD     A
      037721  017443 5150 00       11647         FCMP    T
      037722  000004 7640 04 3772611648         JMP     *+4,$
      037723  000002 7600 04 3772511649         IFA     *+2,$
      037724  017441 4310 00       11650         FLD     A
      037725  017442 5150 00       11651         FCMP    B
      037726  000003 6050 04 3773111652         TPL     *+3,$
      037727  000001 2360 07       11653         LDQ     1,DL
      037730  000002 7100 04 3773211654         TRA     *+2,$
      037731  000000 2361 07       11655         LDQ,    0,DL
                     037732       11656         PRIOR   5
                     037244       11657         ENTER   (076,075),GE
                     037246       11658         DEFOP   M39
  END OF BINARY CARD 00000695
      037732  000005 7630 04 3773711659         INB     *+5,$
      037733  017443 7560 00       11660         STQ     T
```

```
           Z                                 TABLES

     037734  017441 2360 00        11661          LDQ      A
     037735  017443 1160 00        11662          CMPQ     T
     037736  000004 7640 04 3774211663          JMP      *+4,$
     037737  000002 7600 04 3774111664          IFA      *+2,$
     037740  017441 2360 00        11665          LDQ      A
     037741  017442 1160 00        11666          CMPQ     B
     037742  000003 6040 04 3774511667          TMI      *+3,$
     037743  000001 2360 07        11668          LDQ      1,DL
     037744  000002 7100 04 3774611669          TRA      *+2,$
     037745  000000 2361 07        11670          LDQ,     0,DL
                        037746     11671          DEFOP    M59
  END OF BINARY CARD 00000696
     037746  000005 7630 04 3775311672          INB      *+5,$
     037747  017443 4550 00        11673          FST      T
     037750  017441 4310 00        11674          FLD      A
     037751  017443 5150 00        11675          FCMP     T
     037752  000004 7640 04 3775611676          JMP      *+4,$
     037753  000002 7600 04 3775511677          IFA      *+2,$
     037754  017441 4310 00        11678          FLD      A
     037755  017442 5150 00        11679          FCMP     B
     037756  000003 6040 04 3776111680          TMI      *+3,$
     037757  000001 2360 07        11681          LDQ      1,DL
     037760  000002 7100 04 3776211682          TRA      *+2,$
     037761  000000 2361 07        11683          LDQ,     0,DL
                        037762     11684          PRIOR    5
                        037246     11685          ENTER    (076),GT
  END OF BINARY CARD 00000697
                        037250     11686          DEFOP    M39
     037762  000005 7610 04 3776711687          INA      *+5,$
     037763  017443 7560 00        11688          STQ      T
     037764  017442 2360 00        11689          LDQ      B
     037765  017443 1160 00        11690          CMPQ     T
     037766  000004 7640 04 3777211691          JMP      *+4,$
     037767  000002 7620 04 3777111692          IFB      *+2,$
     037770  017442 2360 00        11693          LDQ      B
     037771  017441 1160 00        11694          CMPQ     A
     037772  000003 6050 04 3777511695          TPL      *+3,$
     037773  000001 2360 07        11696          LDQ      1,DL
     037774  000002 7100 04 3777611697          TRA      *+2,$
     037775  000000 2361 07        11698          LDQ,     0,DL
                        037776     11699          DEFOP    M59
  END OF BINARY CARD 00000698
     037776  000005 7610 04 4000311700          INA      *+5,$
     037777  017443 4550 00        11701          FST      T
     040000  017442 4310 00        11702          FLD      B
     040001  017443 5150 00        11703          FCMP     T
     040002  000004 7640 04 4000611704          JMP      *+4,$
     040003  000002 7620 04 4000511705          IFB      *+2,$
     040004  017442 4310 00        11706          FLD      B
     040005  017441 5150 00        11707          FCMP     A
     040006  000003 6050 04 4001111708          TPL     *+3,$
     040007  000001 2360 07       11709         LDQ     1,DL
     040010  000002 7100 04 4001211710          TRA     *+2,$
     040011  000000 2361 07       11711         LDQ,    0,DL
                         040012   11712         PRIOR   5
                         037250   11713         ENTER   (074,075),LE
 END OF BINARY CARD 00000699
                         037252   11714         DEFOP   M39
     040012  000005 7610 04 4001711715          INA     *+5,$
     040013  017443 7560 00       11716         STQ     T
     040014  017442 2360 00       11717         LDQ     B
     040015  017443 1160 00       11718         CMPQ    T
     040016  000004 7640 04 4002211719          JMP     *+4,$
     040017  000002 7620 04 4002111720          IFB     *+2,$
     040020  017442 2360 00       11721         LDQ     B
     040021  017441 1160 00       11722         CMPQ    A
     040022  000003 6040 04 4002511723          TMI     *+3,$
     040023  000001 2360 07       11724         LDQ     1,DL
     040024  000002 7100 04 4002611725          TRA     *+2,$
     040025  000000 2361 07       11726         LDQ,    0,DL
                         040026   11727         DEFOP   M59
 END OF BINARY CARD 00000700
     040026  000005 7610 04 4003311728          INA     *+5,$
     040027  017443 4550 00       11729         FST     T
     040030  017442 4310 00       11730         FLD     B
     040031  017443 5150 00       11731         FCMP    T
     040032  000004 7640 04 4003611732          JMP     *+4,$
     040033  000002 7620 04 4003511733          IFB     *+2,$
     040034  017442 4310 00       11734         FLD     B
     040035  017441 5150 00       11735         FCMP    A
     040036  000003 6040 04 4004111736          TMI     *+3,$
     040037  000001 2360 07       11737         LDQ     1,DL
     040040  000002 7100 04 4004211738          TRA     *+2,$
     040041  000000 2361 07       11739         LDQ,    0,DL
                         040042   11740         PRIOR   5
 END OF BINARY CARD 00000701
                         037252   11741         ENTER   (045),OVER
                         037254   11742         DEFOP   M40
     040042  000004 7630 04 4004611743          INB     *+4,$
     040043  017443 7560 00       11744         STQ     T
     040044  017441 2360 00       11745         LDQ     A
     040045  017443 5061 00       11746         DIV,    T
     040046  000002 7600 04 4005011747          IFA     *+2,$
     040047  017441 2360 00       11748         LDQ     A
     040050  017442 5061 00       11749         DIV,    B
                         040051   11750         PRIOR   7
 END OF BINARY CARD 00000702
                         037254   11751         ENTER   (102,117,117,114,120),BOOLP
                         037256   11752         DEFOP   M94
     040051  000002 7620 04 4005311753          IFB     *+2,$
```

```
    040052  017442 2360 00       11754          LDQ     B
    040053  025413 7001 00       11755          TSX0,   R$BOOL
                   040054        11756          PRIOR   10
  END OF BINARY CARD 00000703
                   037256        11757          ENTER   (103,110,101,122,120),CHARP
                   037260        11758          DEFOP   M95
    040054  000002 7620 04 4005611759           IFB     *+2,$
    040055  017442 2360 00       11760          LDQ     B
    040056  025414 7001 00       11761          TSX0,   R$CHAR
                   040057        11762          PRIOR   10
                   037260        11763          ENTER   (111,116,124,120),INTP
  END OF BINARY CARD 00000704
                   037262        11764          DEFOP   M96
    040057  000002 7620 04 4006111765           IFB     *+2,$
    040060  017442 2360 00       11766          LDQ     B
    040061  025422 7001 00       11767          TSX0,   R$INT
                   040062        11768          PRIOR   10
                   037262        11769          ENTER   (122,105,101,114,120),REALP
  END OF BINARY CARD 00000705
                   037264        11770          DEFOP   M97
    040062  000002 7620 04 4006411771           IFB     *+2,$
    040063  017442 4310 00       11772          FLD     B
    040064  025467 7001 00       11773          TSX0,   R$REAL
                   040065        11774          PRIOR   10
                                 11775          HEAD    A
                   037264        11776          FORCE
                   035213        11777          USE
                   000114        11778 ITABX    EQU     SETW
                   000003        11779 ITACX    EQU     SETC
                                 11780          HEAD    T
                   035237        11781          USE     WK
    035237  000000 000000        11782          ZERO
                   035240        11783 WKE      EQU     *
    035240  000000 000000        11784          ZERO
                   035241        11785          USE     SK
  END OF BINARY CARD 00000706
    035241  000000 000000        11786          ZERO
                   035242        11787 SKE      EQU     *
    035242  000000 000000        11788          ZERO
                   036070        11789          USE     M
                   036070        11790 ME       EQU     *
    036070  000000 000000        11791          ZERO
                   036071        11792          USE     B
                   036071        11793 BE       EQU     *
    036071  000000 000000        11794          ZERO
                   037010        11795          USE     D
                   037010        11796 DE       EQU     *
    037010  000000 000000        11797          ZERO
                   037011        11798          USE     P
                   037011        11799 PE       EQU     *
```

```
                T                                  TABLES

     037011  000000 000000         11800           ZERO
                        037264     11801           USE       S
                        037264     11802 SE        EQU       *
     037264  000000 000000         11803           ZERO
                        037402     11804           USE       I
     037402  000000 000000         11805           ZERO
                        037403     11806 IE        EQU       *
     037403  000000 000000         11807           ZERO
                        037404     11808           USE       C
                        037404     11809 CE        EQU       *
     037404  000000 000000         11810           ZERO
                        037405     11811           USE       L
                        037405     11812 LE        EQU       *
     037405  000000 000000         11813           ZERO
                        037406     11814           USE       G
                        037406     11815 GE        EQU       *
     037406  000000 000000         11816           ZERO
                        037453     11817           USE       T
                        037453     11818 TE        EQU       *
 END OF BINARY CARD 00000707
     037453  000000 000000         11819           ZERO
                        037454     11820           USE       N
                        037454     11821 NE        EQU       *
     037454  000000 000000         11822           ZERO
                        040065     11823           USE       Z
                        040065     11824 ZE        EQU       *
     040065  000000 000000         11825           ZERO
                        040066     11826           USE       SD
                        040066     11827 SDE       EQU       *
     040066  000000 000000         11828           ZERO
                        040067     11829           USE       E
                        040067     11830 EE        EQU       *
     040067  000000 000000         11831           ZERO
                        035213     11832           USE
                                   11833           HEAD      T
     035213  000000 000000         11834 START     ZERO
     035214  035237 000001         11835 WORK      ZERO      WK,WKE-WK
     035215  035241 000001         11836 STACK     ZERO      SK,SKE-SK
     035216  035243 000625         11837 MODE      ZERO      M,ME-M
     035217  036071 000000         11838 BOUND     ZERO      B,BE-B
     035220  036072 000716         11839 DEF       ZERO      D,DE-D
     035221  037011 000000         11840 PROG      ZERO      P,PE-P
     035222  037012 000252         11841 STAB      ZERO      S,SE-S
     035223  037265 000116         11842 ITAB      ZERO      I,IE-I
     035224  037404 000000         11843 CODE      ZERO      C,CE-C
     035225  037405 000000         11844 LBL       ZERO      L,LE-L
     035226  037406 000000         11845 GEN       ZERO      G,GE-G

 END OF BINARY CARD 00000708
     035227  037407 000044         11846 TYPE      ZERO      T,TE-T
     035230  037454 000000         11847 NUM       ZERO      N,NE-N
```

```
              T                               TABLES

     035231  037455 000410          11848 ZZZ     ZERO     Z,ZE-Z
     035232  040066 000000          11849 SDEF    ZERO     SD,SDE-SD
     035233  040067 000000          11850 END     ZERO     E
                                    11851         HEAD     A
     035234  035237 0001 56         11852 WORK    TALLYC   T$WK,1,ID
     035235  035241 0001 51         11853 STACK   TALLYC   T$SK,1,I
                                    11854         HEAD     T
     035236  035214 000023          11855 TABLE   ZERO     T$START+1,*-T$START
                                    11856         REF      LNRSM
 END OF BINARY CARD 00000709
                       000000       11857         END
 END OF BINARY CARD 00000710
40070 IS THE NEXT AVAILABLE LOCATION,   GMAP VERSION  JMPA/030271 JMPB/030271 JMPC/030271
THERE WERE      8 WARNING FLAGS IN THE ABOVE ASSEMBLY
ON PAGE NO,
   17    88    89   122   125   131   135   136
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 30725    1   FJ
 32474    1   ID
  6523    1   MK       1915  1915  2012  2126  2148  2165  2178  2185  2296  2303
 10030    1   ADO      2426  2426
 32632    1   AT1
 32645    1   AT2
 32620    1   ATL
 31101    1   BOX
 31034    1   BYS
  7227    1   CST      2041  2041
  7510    1   CUN      2218  2218
 31225    1   DEC
  7150    1   DEL      1994  1994
 31072    1   DOS
  7505    1   EPR      2215  2215
 10106    1   ER1      2472  2472  2474
 10115    1   ERT      2479  2471  2473  2479
 32571    1   ET1
 32604    1   ET2
  6524    1   ETH      1916  1916
 32557    1   ETL
  6526    1   FIX      1918  1918
  6525    1   FLX      1917  1917
 32071    1   IDE
 32460    1   IDM
  7457    1   MMI      2193  2193
 32217    1   MOD
 31524    1   MT1
 31537    1   MT2
 31550    1   MT3
  7156    1   POP      2000  2000
 30416    1   PRO
 31332    1   PRT
 32144    1   PTA
 32102    1   PTE
 30540    1   SER
     0    1   SET
  7130    1   SMO      1978  1978
 10050    1   SR1      2442  2442  2446
 10063    1   SRT      2453  2441  2442  2445  2453
 31614    1   ST1
 31627    1   ST2
  7125    1   STZ      1975  1975
  7211    1   TAG      2027  2027
 31046    1   TOS
 31663    1   UNT
 31675    1   UT1
 31710    1   UT2
  6612    1   XAQ      1970  1929  1932  1941  1947  1962  1970
  6606    1   XX1      1966  1942  1944  1948  1951  1966
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

  6607    1  XX2     1967  1429  1434  1440  1445  1449  1503  1505  1506  1508  1510  1525  1931  1936  1950  1953
                           1964  1967  2490  2492  2494  2514  2520  2522  2523  2526
  6565    1  XXL     1949  1945  1949
  6554    1  XXS     1940  1686  1696  1853  1862  1940  2039  2555  3163  3359  3410  3444  3449  3474  3479  4968
  6610    1  XXT     1968  1433  1439  1502  1930  1949  1968  2489  2519
  6562    1  XXX     1946  1680  1682  1688  1946  2553  4621  4964
  6540    1  XYZ     1928  1734  1928
 32257    1 ARGE
  6530    1 BARR     1920  1920  3382
 31137    1 BEL2
 31201    1 BEL3
 31212    1 BEL4
  6527    1 BOXR     1919  1919
  7755    1 CBOX     2383  2383
  6533    1 CLNC     1923  1923  2124  2466  2481
  6531    1 CLOR     1921  1921  2186  3386
  7331    1 CNVA     2107  2092  2094  2095  2104  2107
 30434    1 COND
  7265    1 CST1     2071  2071  2077
  7303    1 CSTA     2085  2046  2052  2060  2085
  7305    1 CSTB     2087  2057  2063  2067  2076  2087
  7304    1 CSTM     2086  2051  2059  2065  2073  2074  2086
  7306    1 CSTT     2088  2070  2071  2072  2088
  7536    1 CUN1     2240  2240  2243
  7554    1 CUNA     2254  2227  2231  2233  2234  2239  2242  2254
  7555    1 CUNB     2255  2237  2240  2255
  7556    1 CUNH     2256  2221  2238  2256
  7553    1 CUNX     2253  2222  2253
  6536    1 CURR     1926  1926  2392  2447  2456  2482
  7685    1 DECI     2295  2295
  7676    1 DECL     2336  1993  2336
  7557    1 DECM     2257  2257
  7603    1 DECO     2277  2277
  7571    1 DECP     2267  2267
  7161    1 DELS     2003  2003
  7774    1 EBOX     2398  2398
 30512    1 ELSE
  7367    1 EPD1     2137  2137  2146
  7410    1 EPD2     2154  2154  2157
  7416    1 EPD3     2160  2160  2162
  7417    1 EPD4     2161  2159  2161
  7456    1 EPDD     2192  2128  2130  2135  2153  2156  2158  2167  2171  2192
  7434    1 EPDF     2174  2121  2174
  7455    1 EPDS     2191  2152  2154  2191
  7513    1 EPR1     2221  2214  2216  2219  2221
  7332    1 EREF     2108  2108
 32360    1 FLD2
 32372    1 FLD3
 32404    1 FLD4
 31010    1 FORS
```

```
OCTAL    SYMBOL    REFERENCES BY ALTER NO,

 32436    1 FSEQ
 32523    1 IDEC
  6520    1 IDNT      1912  1912  1983  1986  1988  1990  2027  2198  2207  2329  2339  2356  2358  2430  2450  2460
                            2475  2484  3283
 30750    1 JTL1
 30620    1 JUNK
 32715    1 LIB1
 32730    1 LIB2
 32703    1 LIBL
  7163    1 MARK      2005  2005
 10033    1 MBND      2429  2429
 32271    1 MOD1
 32416    1 MSEQ
 31462    1 OPT1
 31500    1 OPT2
 31450    1 OPTL
 32775    1 PBOD
 31723    1 PDEC
 32757    1 PDEN
 30406    1 PROG     10552  1653

 31344    1 PRT1
 31357    1 PRT2
 31370    1 PRT3
 32156    1 PTA1
 32171    1 PTA2
 32114    1 PTE1
 32127    1 PTE2
  7152    1 PUSH      1996  1996
 32057    1 RES1
  7204    1 REST      2022  2022
 10116    1 RTAB      2480  2452  2478  2480
  7200    1 SAVE      2018  2018
 31561    1 SDEC
     1    1 SETC
  7721    1 SETD      2355  2262  2272  2282  2330  2355  3284
     1    1 SETE     10806 10549
     0    1 SETT
  7142    1 STID      1988  1984  1988
 30463    1 THEN
 31642    1 UDEC
 30602    1 UNIT
  6614    1 XBUF      1971  1954  1961  1968  1971
  6603    1 XRET      1963  1432  1452  1518  1928  1940  1946  1963  2517  2529
  6604    1 XXL1      1964  1952  1964
  6567    1 XXL2      1951  1951  1965
  6571    1 XXL3      1953  1436  1453  1519  1939  1953  2518  2530
  6601    1 XXL4      1961  1959  1961
  6546    1 XYZ1      1934  1934  1938
 31130    1BELEM
 31115    1BTAIL
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

  7773    1CBOXT     2397  2395  2397
  7133    1CLEAR     1981  1981
  7307    1CNVRT     2089  2089
  6532    1COMAC     1922  1922  2388  2468  2480
 32510    1CTAIL
  7630    1DECI1     2298  2298  2301
  7654    1DECI2     2318  2314  2318
  7671    1DECI3     2331  2311  2317  2331
  7634    1DECIA     2302  2147  2177  2295  2302
  7637    1DECIL     2305  2305  2333
  7675    1DECIT     2335  2304  2305  2321  2332  2335
  7674    1DECIX     2334  2302  2308  2334
  7705    1DECLB     2343  2343
  7144    1DECLC     1990  1990
  7620    1DECOP     2290  2290
 10006    1EBOX1     2408  2408  2420
 10027    1EBOXH     2425  2407  2411  2416  2425
 10026    1EBOXM     2424  2401  2413  2418  2424
  6537    1EMPTY     1927  1740  1927
  7346    1EPDEN     2120  2120
  7440    1EPDF0     2178  2176  2178
  7442    1EPDF1     2180  2180  2183
  7446    1EPDF2     2184  2181  2184
  7503    1EPROC     2213  2163  2213
 10066    1ERNGA     2456  2170  2187  2454  2456
 10064    1ERNGE     2454  2454
 10114    1ERTAL     2478  2470  2478
  7174    1EVOID     2014  2014
 32301    1FIELD
  6521    1FMODE     1913  1913
 31022    1FROMS
 32445    1FSEQ1
  7215    1IDENT     2031  1741  2031
 30735    1JTAIL
 30610    1JUNKE
  6535    1LEVEL     1925  1925  2448  2449
  7170    1MRNGE     2010  2010
 32424    1MSEQ1
 31512    1MTAIL
 31402    1OPDEC
 31741    1PDEC1
 32002    1PDEC2
  6522    1PDPOS     1914  1914  2120
 32015    1RESLT
 10123    1RTABE     2485  2452  2478  2485
  6534    1SEMIC     1924  1924  2122  2175  2464  2483
  7727    1SETD1     2361  2361  2369  2375
  7740    1SETD2     2370  2367  2370
  7746    1SETD3     2376  2370  2373  2376
  7747    1SETD4     2377  2362  2377
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

  7723    1SETDE     2357  2350  2357
  7754    1SETDT     2382  2365  2366  2382
  7753    1SETDX     2381  2349  2355  2381
 10040    1SRNGE     2434  2434
 10062    1SRTAL     2452  2440  2452
 31602    1STAIL
  7135    1STIDB     1983  1983
 30762    1SUNIT
 30546    1UTAIL
 30775    1UUNIT
 31060    1WHLES
 12460    2    A     3705  3670  3671  3705
 15253    2    C     5090  3633  3642  4869  4894  4980  4991  5090
 11645    2    D     3310  3299  3302  3310
 12457    2    N     3704  3664  3677  3704

 13652    2   B1     4337  4337  4355
 13655    2   B2     4340  4340  4364  4408  4466
 15114    2   BB     4995  3626  4990  4995
 15250    2   BC     5087  4999  5050  5054  5087
 15252    2   BF     5089  5067  5070  5076  5079  5089
 15251    2   BN     5088  5015  5032  5049  5053  5088
 15247    2   BX     5086  4995  5086
 15261    2   C0     5096  5096  5361  5422  5484
 15323    2   C1     5130  5130  5250
 15332    2   C2     5137  5137  5138  5219
 15334    2   C3     5139  5139  5142  5278
 15346    2   C4     5149  5146  5149  5151  5153
 15361    2   C5     5160  5155  5160
 15362    2   C6     5161  5148  5159  5161
 15363    2   C7     5162  5162  5166
 15364    2   C8     5163  5163
 15366    2   C9     5165  5165
 16176    2   CF     5565  4873  4898  5097  5223  5264  5279  5286  5298  5309  5364  5371  5425  5565
 16200    2   CW     5567  5092  5168  5354  5445  5567
 15530    2   CX     5263  5099  5175  5263  5291  5335  5500
 16204    2   DC     5571  3635  3644  4877  4900  4902  4982  4993  5571
 11647    2   DM     3312  3277  3285  3312
 11640    2   DN     3306  3205  3243  3253  3255  3258  3263  3265  3269  3274  3287  3300  3303  3306
 11646    2   DT     3311  3231  3233  3311
     1    2   FB     5755  3384  5755
 16177    2   FF     5566  5098  5177  5183  5249  5566
     2    2   FP     5756  3388  5756
 33210    2   IF
 13660    2   L1     4343  3149  4343  4380  4394  4412
 15573    2   LC     5298  5117  5298
 11651    2   LK     3314  3314  3350  4701  4941
 16465    2   LL     5751  3788  4671  4676  4683  4728  4748  5751
 16467    2   MK     5753  5753
 33337    2   QS
 11423    2   TG     3165  3159  3165  3354
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 15531    2   VC     5264  5180  5264
 12375    2   ADD    3654  3654  4027  4799  4801  4803  4805  4807  4956  5604  5663  5665  5667  5677  5679  5691
                           5706  5727
 13260    2   ADO    4087  4087
 35115    2   AT1
 35127    2   AT2
 35103    2   ATL
 13654    2   B1S    4339  3146  4339  4389  4420
 13653    2   B1X    4338  3145  4337  4338  4388
 13657    2   B2S    4342  3148  4342  4393  4424  4456  4472  4481
 13656    2   B2X    4341  3147  4340  4341  4392  4425  4455  4473  4480
 15641    2   BAL    5344  5104  5344  5345  5593
 12013    2   BAR    3412  3412
 15122    2   BB1    5001  5001  5082
 15131    2   BB2    5008  5007  5008
 15133    2   BB3    5010  5005  5010
 15143    2   BB4    5018  5018  5026
 15154    2   BB5    5027  5024  5027
 15157    2   BB6    5030  5030  5039  5041
 15173    2   BB7    5042  5021  5028  5036  5042
 15205    2   BB8    5052  5048  5052
 15211    2   BB9    5056  5044  5056
 15106    2   BLC    4989  4985  4987  4989
 34146    2   BOX
 15370    2   C10    5167  5140  5167  5285  5297  5308  5326  5328
 15402    2   C11    5177  5136  5177
 15434    2   C12    5203  5203  5211
 15455    2   C13    5220  5198  5208  5220
 15460    2   C14    5223  5186  5188  5190  5223
 15472    2   C15    5233  5233  5259
 15505    2   C16    5244  5242  5244
 15514    2   C17    5251  5192  5241  5251
 15521    2   C18    5256  5256  5258
 15522    2   C19    5257  5255  5257
 15525    2   C20    5260  5235  5260
 15526    2   C21    5261  5261  5262
 12614    2   CAD    3796  3470  3520  3526  3565  3567  3571  3575  3577  3581  3586  3590  3592  3595  3794  3796
                           3910  3916  3922  3924  3929  3932  3938  3944  3948  4052  4132  4135  4176  4196  4323
                           4363  4367  4369  4397  4399  4419  4423  4460  4465  4469  4485  4602  4613  4654  4698
                           4715  4720  4725  4923  4938
 15025    2   CFS    4952  4878  4901  4903  4952
 33033    2   CLO
 16457    2   CNT    5745  3119  3142  3955  4130  4228  4237  4242  5745
 14620    2   COP    4819  4532  4556  4819  4943
 16474    2   CPR    5767  4530  4544  4554  4823  5767
 16123    2   CPX    5522  5499  5522
 15550    2   CSK    5279  5108  5279
 12661    2   CST    3833  3833
 12733    2   CUN    3875  3875
 16235    2   DCO    5597  5589  5597
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 16243     2   DC1      5603   5580   5581   5582   5583   5584   5585   5586   5603
 16263     2   DC2      5619   5587   5588   5590   5619
 16300     2   DC3      5632   5579   5632
 16303     2   DC4      5635   5591   5635
 16306     2   DC5      5638   5592   5638
 16205     2   DCL      5572   5572   5614   5618   5631   5634   5637   5642   5685   5732
 12263     2   DCT      3580   3580
 16206     2   DCX      5573   5571   5573
 34424     2   DEC
 14010     2   DIF      4431   4431
 11563     2   DST      3261   3195   3197   3261
 12731     2   EPR      3873   3873
 11351     2   ER1      3131   3131   3133
 11363     2   ERT      3141   3130   3132   3141
 35062     2   ET1
 35050     2   ETL
 14137     2   FOP      4518   4518
 34411     2   FT1
 34400     2   FTL
 12403     2   INS      3660   3492   3537   3539   3545   3553   3557   3559   3660   4023   4335   4354   4358   4360   4411   4415
                               4417   4639   4772   4776   4792   4931   5688   5724
 33734     2   IX1
 33752     2   IX2
 33767     2   IX3
 34001     2   IX4
 34011     2   IX5
 11656     2   LK1      3319   3316   3319
 11663     2   LK2      3324   3321   3324   3328
 11670     2   LK3      3329   3325   3329
 11673     2   LKX      3332   3314   3318   3332
 13313     2   LWB      4114   4114
 34633     2   MD1
 34655     2   MD2
 34045     2   MOD
 14212     2   MOP      4561   4561
 13401     2   OB1      4168   4168   4178
 13407     2   OB2      4174   4171   4174
 13414     2   OB3      4179   4173   4179
 13415     2   OB4      4180   4180   4185
 15071     2   OPF      4976   4563   4826   4844   4849   4875   4942   4976
 16475     2   OPT      5768   3150   4537   4540   4819   4828   5768
 15725     2   PAR      5396   5106   5396   5436   5594
 34506     2   PDT
 13235     2   POP      4068   4068
 34665     2   PR1
 34707     2   PR2
 33007     2   PRO
 34675     2   PRT
 33345     2   QST
 13573     2   RLL      4290   4266   4269   4273   4277   4290   4729   4915
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 12200    2  SCT     3529  3529
 33533    2  SEC
 33144    2  SER
     0    2  SET
 12324    2  SIS     3613  3613
 11330    2  SR1     3114  3114  3118
 11361    2  SRT     3139  3113  3114  3117  3139
 34523    2  ST1
 34554    2  ST2
 34542    2  STL
 13262    2  STZ     4089  4089
 13467    2  SV1     4222  4220  4222
 11410    2  TAG     3154  3154
 11642    2  TEN     3307  3250  3307
 34571    2  UN1
 13315    2  UPB     4116  4116
 34620    2  UT2
 34606    2  UTL
 15606    2  VAC     5309  5112  5309
 15532    2  VC0     5265  5265  5266
 15541    2  VC1     5272  5269  5272
 12572    2  WAD     3778  3620  3778  4672  4677  4684  4734  4749
 13333    2 ACNT     4130  4113  4115  4117  4119  4121  4123  4125  4127  4129  4130
 33654    2 ACTP
 12407    2 ADD1     3664  3659  3664
 34036    2 AMOD
 15662    2 BAL1     5361  5361  5369
 13502    2 BAL2     4233  4233
 15704    2 BAL3     5379  5362  5379  5433  5464  5470  5485
 15713    2 BAL4     5386  5386  5388
 13513    2 BALN     4242  4242
 12020    2 BARF     3417  3417
 15226    2 BB10     5069  5059  5061  5064  5069
 15231    2 BB11     5072  5057  5072
 15241    2 BB12     5080  5009  5011  5014  5051  5055  5068  5071  5073  5077  5080
 15244    2 BB13     5083  5003  5075  5083
 15113    2 BLCX     4994  4989  4994
 13445    2 BUSC     4204  4139  4180  4182  4204
 13446    2 BUSD     4205  4141  4164  4175  4177  4205
 13444    2 BUSF     4203  4137  4154  4186  4195  4203
 13447    2 BUSM     4206  4148  4156  4179  4206
 12624    2 CADT     3804  3797  3801  3804
 12623    2 CADX     3803  3796  3803
 14320    2 CALL     4627  4627
 13046    2 CBOX     3950  3950
 16152    2 CDEP     5545  5300  5314  5398  5545
 15035    2 CFSX     4960  4954  4960
 11755    2 CHK1     3382  3379  3382
 11761    2 CHK2     3386  3383  3386
 11764    2 CHK3     3389  3381  3385  3389
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO.

 11770    2 CHK4     3393  3390  3393
 11771    2 CHK5     3394  3392  3394
 11775    2 CHK6     3398  3395  3398
 11776    2 CHK7     3399  3397  3399
 12002    2 CHK8     3403  3400  3403
 12003    2 CHK9     3404  3402  3404
 16450    2 CNDX     5736  5736  5737
 15557    2 CNIL     5286  5110  5286
 13752    2 CONE     4401  4401
 13723    2 CONF     4378  4378
 14641    2 COP1     4836  4836  4845  4951
 15016    2 COP3     4945  4895  4899  4945  4946
 15020    2 COP4     4947  4870  4874  4947  4948
 14662    2 COP5     4853  4850  4853
 14664    2 COP6     4855  4852  4855
 14721    2 COP7     4884  4876  4884
 14756    2 COP8     4913  4883  4913
 14775    2 COP9     4928  4919  4928
 15040    2 COPE     4963  4837  4840  4963
 14633    2 COPM     4830  4565  4830
 15070    2 COPT     4975  4860  4871  4896  4918  4952  4975
 14621    2 COPX     4820  4531  4555  4564  4820  4824  4944
 12026    2 CPAR     3423  3423
 16124    2 CRED     5523  5120  5131  5523
 13716    2 CSCT     4373  4373
 12705    2 CST1     3853  3853  3860
 12720    2 CST2     3864  3864  3866
 12725    2 CSTA     3869  3838  3842  3869
 12726    2 CSTB     3870  3841  3845  3852  3859  3870
 12727    2 CSTC     3871  3849  3853  3856  3871
 12730    2 CSTD     3872  3850  3854  3858  3863  3872
 12734    2 CUN1     3876  3874  3876
 12735    2 CUN2     3877  3868  3877
 12747    2 CUN3     3887  3887  3889
 16313    2 DBAL     5643  5593  5643
 13016    2 DBUS     3926  3926
 11644    2 DCNT     3309  3208  3241  3298  3309
 11621    2 DCNV     3291  3206  3212  3291  3304
 16214    2 DCTB     5579  5574  5575  5579
 12321    2 DCTT     3610  3583  3584  3587  3588  3593  3610
 35204    2 DECX
 14100    2 DELV     4487  4370  4474  4486  4487
 12530    2 DELW     3744  3516  3601  3647  3744  4660  4910
 11603    2 DEND     3277  3257  3260  3275  3277
 11650    2 DENT     3313  3190  3192  3193  3219  3234  3261  3262  3273  3291  3313
 16175    2 DEPT     5564  5555  5556  5557  5564
 13472    2 DFLG     4225  4214  4217  4219  4223  4225
 14064    2 DIF1     4475  4440  4442  4475
 14053    2 DIFL     4466  4466  4471
 11553    2 DINT     3253  3207  3253
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 11472    2 DNUM     3204  3199  3202  3204
 16471    2 DONE     5764  5764
 16377    2 DPAR     5695  5594  5695
 11602    2 DST1     3276  3271  3276
 12753    2 DSUB     3891  3891
 13065    2 EBOX     3965  3965
 33272    2 ELSE
 14377    2 ENIL     4674  4674
 13661    2 ENTC     4344  4344
 13641    2 ENTL     4328  4328
 13321    2 EPTY     4120  4120
 12642    2 EREF     3818  3818
 11643    2 ESGN     3308  3217  3226  3238  3308
 14407    2 ETR1     4682  4680  4682
 15101    2 FIRM     4984  4405  4984
 34336    2 FLD1
 14145    2 FOP1     4524  4524  4528
 14203    2 FOPS     4554  4554
 34206    2 FOPT
 34300    2 FSEQ
 16464    2 GLBL     5750  2863  2864  3534  3535  3540  3541  3554  3555  3568  3569  3572  3895  3896  3900  3901
                           3903  3904  4378  4379  4386  4387  4390  4391  4453  4454  4461  4478  4479  4769  4770
                           4773  4774  4793  5660  5673  5674  5750  5918  5922
 13217    2 GMOD     4054  4054
 35002    2 IDEC
 16462    2 IDNT     5748  4226  5748
 13537    2 INLL     4262  4262
 12416    2 INS1     3671  3671  3676
 12424    2 INS2     3677  3672  3677
 12431    2 INS3     3682  3682  3687  3700  3702
 12454    2 INS4     3701  3698  3701
 12447    2 INS5     3696  3689  3690  3696
 12452    2 INS6     3699  3697  3699
 12456    2 INSX     3703  3654  3660  3684  3703
 35173    2 LIB1
 11322    2 LIBF     3108  3106  3108  8896
 35161    2 LIBL
 11732    2 LPAR     3363  3363
 33412    2 LSEQ
 14363    2 MARK     4662  4662
 12467    2 MAT1     3712  3712  3728
 12507    2 MAT2     3727  3716  3719  3727
 12511    2 MAT3     3729  3726  3729
 12512    2 MATX     3730  3706  3730
   100    2 MCOL     5761  3393  5761
    20    2 MCOM     5759  3398  5759
 33520    2 MFOR
 11431    2 MIND     3171  3171
 16451    2 MNDX     5738  5738  5739
 16452    2 MODE     5740  5350  5441  5592  5740
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO.

 34267    2 MODI
 14220    2 MOPC      4567   4562   4567
   400    2 MSEM      5763   3403   5763
 34260    2 MSEQ
 13317    2 NLWB      4118   4118
 13342    2 OBUS      4137   4137
 13327    2 OFIX      4126   4126
 14424    2 OIDN      4695   3107   4695
 13323    2 OLWB      4122   4122
 11424    2 OPER      3166   3166
 14136    2 OPNT      4517   4502   4514   4517
 34772    2 OPT1
 34760    2 OPTL
 13550    2 ORLL      4271   4271
 13337    2 OSUB      4134   4134
 13325    2 OUPB      4124   4124
 15752    2 PAR1      5417   5417   5432
 15766    2 PAR2      5429   5423   5426   5429
 15773    2 PAR3      5434   5407   5434
 16030    2 PAR4      5463   5460   5463
 16033    2 PAR5      5466   5462   5466
 16041    2 PAR6      5472   5472   5489
 16050    2 PAR7      5479   5455   5457   5476   5479
 16052    2 PAR8      5481   5478   5481
 16054    2 PAR9      5483   5480   5483
 13475    2 PARN      4228   4228
 14530    2 PCDR      4763   4039   4763   5633
     4    2 PDEN      5757   3380   5757
 16451    2 PLOC      5739   5739
 34136    2 PMOD
 33575    2 PRIM
 13251    2 PUSH      4064   4064
 33316    2 QSEQ
 33423    2 QUAT
 16151    2 REDT      5544   5535   5536   5537   5544
 16463    2 RELF      5749   4373   4376   5749
 13254    2 REST      4083   4083
 33152    2 RLTR
 11357    2 RNGE      3137   3109   3120   3125   3126   3135   3137   3333   3373   3423   3457   3891   3911   3917   3926   3933
                             3939   3959   4020   4024   4142   4245   4381   4435   4520   4779   4834
 11364    2 RTAB      3142   3138   3140   3142
 13457    2 SACT      4214   4214
 13251    2 SAVE      4080   4080
 13311    2 SBCT      4112   4112
 14245    2 SEL0      4588   4578   4588
 14251    2 SEL1      4592   4592   4599
 14273    2 SEL2      4610   4608   4610
 14305    2 SELE      4620   4580   4585   4593   4620
 14317    2 SELF      4626   4568   4587   4601   4607   4626
 14316    2 SELT      4625   4597   4598   4625
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

    3     2 SETC
    1     2 SETE
    0     2 SETT
13451     2 SINT     4208  4208
12325     2 SIS1     3614  3612  3614
13272     2 SLOC     4097  4097
15105     2 SOFT     4988  3486  3533  4988
13473     2 STID     4226  4226

13224     2 SVAL     4059  4059
33477     2 TERT
33246     2 THEN
11674     2 TLUG     3333  3170  3175  3180  3333
11730     2 TLUT     3361  3167  3172  3177  3346  3361
33154     2 TRYL
33357     2 TSEQ
33401     2 UNIT
15634     2 VAC1     5331  5331  5333
15632     2 VACT     5329  5312  5329
16446     2 VALP     5734  3485  3508  3532  3546  3560  3596  3624  3632  3641  3655  3661  4007  4009  4028  4047
                           4062  4209  4334  4348  4404  4407  4434  4572  4632  4637  4640  4655  4777  4808  4866
                           4884  4891  4917  4957  4996  5093  5100  5167  5170  5174  5353  5356  5360  5381  5444
                           5447  5451  5512  5591  5597  5605  5619  5636  5638  5643  5654  5657  5659  5670  5680
                           5695  5704  5707  5711  5715  5718  5728  5734
15534     2 VC,5     5267  5267  5271
34027     2 VMOD
13205     2 VOID     4044  4044
12613     2 WADX     3795  3778  3795
14503     2 WDEN     4742  4742
15103     2 WEAK     4986  4573  4986
13574     2 WGEN     4291  4291
14207     2 WMOP     4558  4558
   40     2 ZCOL     5760  3391  5760
   10     2 ZCOM     5758  3396  5758
  200     2 ZSEM     5762  3401  5762
16201     2AMODE     5568  5123  5135  5194  5568
12172     2ASGNE     3523  3523
15673     2BAL2,     5370  5370  5428  5494  5496  5498
13517     2BALN1     4246  4232  4236  4241  4246
13506     2BALZR     4237  4237
16202     2BMODE     5569  3627  3637  4868  4893  4979  4998  5047  5052  5056  5066  5069  5072  5078  5083  5130
                           5134  5246  5254  5260  5280  5287  5292  5299  5313  5317  5334  5348  5367  5383  5397
                           5401  5421  5483  5505  5569
34020     2BOUND
34163     2BOXER
34217     2BTAIL
15406     2C11,3     5181  5178  5181
15420     2C11,7     5191  5184  5191
15461     2C14,1     5224  5222  5224
16153     2CDEP1     5546  5546  5561
16173     2CDEP2     5562  5551  5554  5562
```

```
OCTAL     SYMBOL     REFERENCES BY ALTER NO,

 12005    2CHK10      3406  3372  3406
 13265    2CLEAR      4092  4092
 16450    2CMODE      5737  5737
 15013    2COP10      4942  4927  4942
 15022    2COP41      4949  4949
 14710    2COP61      4875  4872  4875
 14741    2COP71      4900  4897  4900
 15041    2COPE1      4964  4962  4964
 15047    2COPEM      4970  4963  4970
 15036    2COPEP      4961  4525  4961
 14202    2COPER      4553  4518  4541  4553
 15067    2COPET      4974  4519  4830  4965  4974
 15056    2COPPM      4972  4961  4972
 16074    2CPUSH      5499  5344  5396  5499
 16125    2CRED1      5524  5524  5541
 16140    2CRED2      5535  5529  5535
 16147    2CRED3      5542  5531  5534  5542
 13720    2CSCTB      4375  4375
 12141    2DASGN      3498  3498
 12171    2DASGT      3522  3499  3513  3517  3522
 16327    2DBAL1      5655  5655  5668
 16345    2DBAL2      5669  5656  5669
 16375    2DBALC      5693  5651  5655  5693
 16376    2DBALT      5694  5647  5684  5686  5689  5694
 16257    2DC1,1      5615  5612  5615
 11625    2DCNV1      3295  3292  3295
 12262    2DCTAB      3579  3579
 16235    2DCTBE      5596  5575  5596
 16461    2DECL1      5747  5747
 16460    2DECLR      5746  3102  3143  3504  3523  3813  3816  3818  3820  3862  3876  3884  3965  3992  4003  4045
                            4080  4084  4211  4294  4311  4315  4320  4446  4476  4628  4693  4712  4717  4978  5746
 14103    2DELV1      4490  4490  4496
 14112    2DELV2      4497  4497  4499
 14116    2DELVT      4501  4493  4494  4501
 12536    2DELW1      3750  3750  3753
 12561    2DELW2      3769  3769  3773
 12566    2DELW3      3774  3768  3774  3776
 12515    2DELWL      3733  3733  3735
 12513    2DELWW      3731  3731  5672  5717
 12571    2DELWX      3777  3744  3777
 11403    2DENOT      3181  3181
 12625    2DEREF      3805  3805
 14031    2DIFX0      4448  4444  4448
 11505    2DNUM1      3215  3211  3215
 11514    2DNUM2      3222  3220  3222
 11522    2DNUM3      3228  3225  3228  3235
 11530    2DNUM4      3234  3223  3227  3234
 11537    2DNUM5      3241  3213  3239  3241
 11543    2DNUM6      3245  3245  3249  3252
 11550    2DNUM7      3250  3246  3250
```

```
OCTAL     SYMBOL     REFERENCES BY ALTER NO,

 16411    2DPAR1      5705  5705  5713
 16440    2DPAR2      5728  5692  5728
 16445    2DPART      5733  5701  5712  5722  5725  5733
 11560    2DREAL      3258  3245  3258
 12762    2DSUB1      3898  3894  3898
 12767    2DSUB2      3903  3899  3903
 13106    2EBOX1      3982  3982  3989
 13124    2EBOXC      3995  3979  3988  3995
 13122    2EBOXM      3994  3968  3981  3983  3986  3987  3994
 14404    2EFALS      4679  4679
 13714    2ENTCT      4371  4349  4361  4371
 13125    2EPDEN      3996  3996
 13172    2EPDNE      4033  4033
 13203    2EPDNL      4042  4042
 13202    2EPDNM      4041  3997  4017  4038  4041
 13204    2EPDNT      4043  4000  4013  4015  4043
 12646    2EREF1      3822  3819  3822  4188  4609
 12657    2EREFT      3831  3825  3827  3831
 12656    2EREFX      3830  3822  3830
 11347    2ERNGE      3129  3129
 11362    2ERTAL      3140  3129  3140
 14372    2ESKIP      4669  4669
 14406    2ETRUE      4681  4681
 12637    2EVOID      3815  3815
 34331    2FIELD
 13257    2FMODE      4086  3103  3144  3805  4057  4086  4512
 34367    2FORMP
 34310    2FSEQ1
 16466    2GTYPE      5752  4095  4098  4100  4102  4105  4107  4110  4313  5752
 35023    2IDEC1
 11436    2IDENT      3176  3176
 12374    2IDNTT      3653  3628  3636  3653
 12330    2IDNTY      3617  3617
 33734    2INDEX
 33631    2ITAIL
 12063    2LABEL      3452  3452
 11731    2LASTS      3362  3105  3353  3362  4054  4515  4690  4695  4699  4702  4709  4722  4730  4737
 16470    2LINKO      5754  3319  5754  5934  8868
 12025    2LPARP      3422  3364  3371  3404  3413  3418  3422
 12461    2MATCH      3706  3706  3828  3883  3984
 16453    2MODE1      5741  4854  4892  5741
 16454    2MODE2      5742  4856  4867  5742
 16455    2MODE3      5743  4858  4881  4908  4936  5743
 34643    2MTAIL
 14452    2OASGN      4717  4717
 13366    2OBUS1      4157  4157  4166
 13371    2OBUS2      4160  4152  4160
 13375    2OBUS3      4164  4161  4164
 13423    2OBUS6      4186  4181  4186
 13426    2OBUS7      4189  4187  4189
```

```
OCTAL     SYMBOL     REFERENCES BY ALTER NO,

 13213    20DELV      4050  4050
 14445    20ENTR      4712  4712
 13331    20FLEX      4128  4128
 14457    20FORM      4722  4722
 14433    20IDN2      4702  4702
 14117    20IDNY      4502  4502
 14121    20IDY1      4504  4504  4507  4510
 34717    20PDEC
 16456    20PDEF      5744  4862  4920  4928  4939  4949  5744
 13564    20RLL1      4283  4283  4288
 13570    20RLL2      4287  4280  4282  4287
 34347    2PARAM
 11722    2PAROK      3355  3290  3355
 14613    2PCDL1      4814  4768  4806  4814
 14614    2PCDL2      4815  4780  4804  4815
 14615    2PCDL3      4816  4795  4802  4816
 14616    2PCDL4      4817  4764  4800  4817
 14617    2PCDL5      4818  4797  4798  4818
 14554    2PCDR1      4783  4783  4788
 14562    2PCDR2      4789  4781  4783  4786  4789
 14612    2PCDRX      4813  4763  4813
 34120    2PROCM
 34105    2PROCT
 34354    2PTAIL
 13244    2PUSEA      4075  4075
 13240    2PUSHW      4071  4071
 33325    2QSEQT
 33434    2QTAIL
 11376    2RTABE      3152  3138  3140  3152
 12121    2SASGN      3482  3482
 13450    2SBCNT      4207  3151  4138  4207
 13275    2SBLNK      4100  4100
 12177    2SCTAB      3528  3528
 14417    2SDCLR      4690  4690
 14221    2SELCT      4568  4568
 14310    2SELEM      4623  4620  4623
 13267    2SHEAP      4094  4094
 13277    2SHPBK      4102  4102
 12322    2SISNT      3611  3611
 13304    2SLCBK      4107  4107
 11323    2SRNGE      3109  3109
 11360    2SRTAL      3138  3112  3138
 33505    2STAIL
 11307    2START      3097  3097
 15072    2STRNG      4977  3509  4008  4048  4212  4447  4477  4633  4977
 15100    2STRNX      4983  4977  4983
 34232    2SUNIT
 13464    2SVIRT      4219  4219
 12037    2TAGOF      3432  3432
 11717    2TLUG2      3352  3349  3352
```

```
01557 01  05-26-72   18,180      ALGOL68                                                                              PAGE    285

OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 11720    2TLUG3     3353  3351  3353
 16203    2TMODE     5570  3487  4574  5084  5570
 11736    2TPCHK     3367  3367  3416  3421
 33166    2TRAIN
 33367    2TSEQT
 33174    2UTAIL
 34244    2UUNIT
 16447    2VALP1     5735  5644  5669  5696  5714  5735
 13640    2WGENX     4327  4319  4327
 14464    2WIDEN     4727  4727
 11335    2XRNG1     3119  3119  3128
 11346    2XRNGE     3128  3128
 33701    2XTAIL
 17441    3    A     6097  6097
 17442    3    B     6098  6098
    17    3    D     5822  5822  6006  6308  6332  6340  6349  6385  6412  6424  6487  6494  6500  6507  6623  6646
                           6745  6757  6768  6971  6990  7006  7211  7260  7450  7551  7769  7774  7972  8016  8021
                           8154  8157  8253  8255  8403  8423  8447  8454  8624  8629  8633
    16    3    S     5821  5821  6658  6834  7552  7992  8032  8213  8272
 17443    3    T     6099  6099
 24221    3    V     8511  8505  8506  8507  8508  8511
 17463    3   A1     6117  6097  6117
     1    3   AU     5910  5910  6395  6520  7779  7781  8162  8166
 17466    3   B1     6120  6098  6120
 22512    3   BD     7671  7660  7671  7715  7879
     7    3   DL     5912  5912  6470  6524  6749  7326  7586  7632  8636
 17721    3   DR     6274  6011  6261  6272  6274  7322
     3    3   DU     5911  5911  8774  8777
 20451    3   FS     6614  5796  6614  7456  7493  8190  8491  8514
 17610    3   GA     6201  6201  6461  6482  6515  7618
 17612    3   GQ     6203  6203  7317  7591  8334  8352
     4    3   IC     5913  5913  6960  7366  7398  7547  7859
   323    3   IR    11435  6843
 23723    3   IS     8320  5783  8320
 17057    3   LD     5833  5833
 23264    3   LL     8033  5786  8033
 20451    3   MA     6613  5796  6613
 20421    3   MR     6589  6589  6637
 17251    3   OP     5972  5780  5972
 17732    3   P1     6283  6283  6360
 17751    3   P2     6298  6291  6295  6298
 17766    3   P3     6311  6304  6311
 17770    3   P4     6313  6299  6302  6313
 20010    3   P5     6329  6324  6329
 20026    3   P6     6343  6322  6343
 20036    3   P7     6351  6320  6342  6351
 20042    3   P8     6355  6314  6318  6355
 20045    3   P9     6358  6285  6286  6358
 20051    3   PF     6362  6288  6297  6312  6351  6355  6362
 20050    3   PX     6361  6281  6361
```

```
01557 01  05-26-72   18,180       ALGOL68                                                                              PAGE     286

OCTAL     SYMBOL    REFERENCES BY ALTER NO.

 17037    3    SP     5824   5824   5933   6555   6560   6563   6565   6570   6571   6819   6826   7308   7488   7947   7984   8005
                             8039   8058   8085   8233   8420   8426   8495   8682
 17070    3    ST     5842   5842   6294   6301   6331
 17471    3    T1     6123   6099   6123
 22035    3    TF     7369   5782   7369
 22110    3    TL     7411   6748   7409   7411   8143   8463
 24225    3    V1     8515   8510   8515
 24230    3    VX     8518   8509   8511   8518
     4    3    WL     5831   5831   5984   5993   6577   6581   6636   6675   6728   6752   6806   6824   6828   6856   6896   6900
                             6905   7067   7071   7075   7084   7094   7138   7141   7205   7250   7304   7313   7371   7428   7501
                             7612   7654   7793   7887   7890   7893   7969   7995   8096   8189   8260   8285   8309   8311   8324
                             8327   8354   8374   8378   8393   8400   8436   8490   8513   8534   8590   8678   8756   8760
 17134    3    X1     5862   5853   5862
 17135    3    X2     5863   5854   5863
 17136    3    X3     5864   5855   5864
 17137    3    X4     5865   5856   5865
 17140    3    XA     5866   5857   5866
 17141    3    XQ     5867   5858   5867
 75000    3   ADA     5899   5899   6470   6524   6749   7632
 60000    3   ADX     5889   5889
 21556    3   ASD     7194   6938   6962   6981   7000   7046   7051   7104   7126   7194
 21555    3   ASE     7193   6936   6944   7043   7053   7193
 21041    3   ASG     6861   6641   6858   6861   6878   7049
 21554    3   ASL     7192   6852   6903   6913   7039   7042   7055   7132   7146   7147   7152   7155   7192
 22515    3   BD1     7674   7674   7726
 22541    3   BD2     7694   7689   7691   7694
 22563    3   BD3     7712   7712
 22577    3   BD4     7724   7693   7724   7911
 22602    3   BD5     7727   7683   7727
 22606    3   BD6     7731   7728   7731
 22657    3   BD7     7772   7768   7772
 22662    3   BD8     7775   7775   7780
 22670    3   BD9     7781   7778   7781
 23101    3   BDD     7918   7811   7813   7819   7918
 23076    3   BDX     7915   7671   7679   7695   7722   7867   7885   7915
 24500    3   BUI     8687   8538   8687
 24244    3   BUS     8531   5789   8531
 17665    3   CL1     6246   6246   6651   6809   7223   7233   7243   7266   7277
 17673    3   CL3     6252   6252   6448   6539   6546   6653   6811   7225   7236   7269   7445   7465   7482
 24603    3   DB1     8754   8747   8754
 24620    3   DB2     8767   8762   8767
 24622    3   DB3     8769   8766   8769
 24626    3   DBT     8773   8746   8753   8759   8763   8767   8773   8796   8806
 24623    3   DBX     8770   8754   8770
 17043    3   DLL     5828   5828   8080   8147   8152
 17723    3   DR1     6276   6276   6279
 17727    3   DRX     6280   6274   6280
635000    3   EAA     5898   5898   6464   6486   6520   6745   7769   7781   8149   8154   8166
620000    3   EAX     5886   5886   6326   6336   6395
 21732    3   ETC     7302   5783   7302
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 24633    3  FF1     8776   8776   8778
 24624    3  FFA     8771   8758   8771   8782
 24625    3  FFI     8772   8742   8749   8772   8783
 24642    3  FFX     8785   8778   8785
 24632    3  FIX     8777   5789   8777
431000    3  FLD     5901   5901   6047
 25076    3  FRN     8943   8916   8918   8926   8928   8934   8938   8943
 20507    3  FS1     6644   6619   6644
 20514    3  FS2     6649   6617   6627   6643   6649
 20525    3  FSI     6658   6629   6630   6658
 20524    3  FSX     6657   6614   6657
 17474    3  GAD     6126   5931   6075   6078   6126   6142   6175   6182   6184   6193   6309   6328   6333   6337   6341   6397
                            6405   6415   6427   6444   6465   6471   6488   6510   6521   6525   6631   6648   6746   6750   6758
                            6770   6777   6817   6947   6949   6961   7007   7156   7320   7327   7329   7331   7333   7367   7375
                            7396   7399   7529   7541   7548   7597   7633   7745   7770   7776   7782   7784   7825   7827   7851
                            7860   7964   7973   8022   8028   8126   8128   8150   8155   8159   8164   8167   8169   8251   8256
                            8258   8268   8270   8277   8404   8424   8456   8768   8779   8784   8834   8860   8867   8895   8983
                            8990   8995   9116
 17614    3  GAQ     6205   6205
 17616    3  GEA     6207   6207
 22410    3  GEN     7605   7602   7605
 17637    3  GET     6224   6009   6028   6202   6204   6206   6208   6210   6224   7573
 25234    3  GUM     9034   8848   8877   8891   9034
 17622    3  GXR     6211   6211   6391   6504   6530
 22000    3  HIP     7340   5781   7340
 17433    3  IFA     6088   6088
 17435    3  IFB     6090   6090
 17434    3  INA     6089   6089
 17436    3  INB     6091   6091
 23751    3  IS0     8342   8326   8342   8350
 23753    3  IS1     8344   8329   8344
 23755    3  IS2     8346   8321   8346
 23724    3  ISS     8321   8319   8321
 23761    3  IST     8350   8339   8350
 23750    3  ISX     8341   8322   8341
 17437    3  JMP     6092   6092
 22006    3  LBL     7346   5780   7135   7346   7905   8249   8275
235000    3  LDA     5891   5891   6177   7774   7779   8157   8162
236000    3  LDQ     5892   5892   6041   6043   6045   6049   6051   6167   7319   7586
220000    3  LDX     5884   5884   6403   6413   6425   6443   6508   6768   6775   8255   8403   8774   8777
   330    3  LIR    11446   6846
 23307    3  LL1     8052   8052   8064
 23315    3  LL2     8058   8053   8058
 23322    3  LL3     8063   8057   8063
 23324    3  LL4     8065   8059   8065
 23337    3  LLE     8076   5786   8076
 23336    3  LLX     8075   8033   8075
 24566    3  LWB     8741   5789   8741
 17401    3  MAC     6061   6012   6061
 20350    3  MAX     6553   5796   6553
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 17245    3  MMI     5968  5793  5968
 20066    3  MNA     6375  6375  6909  7139  7258  7461  8325  8328  8397  8757
 17101    3  MOD     5851  5851  6220  6317
 20436    3  MR1     6602  6602  6607
 20441    3  MR2     6605  6603  6605
 20450    3  MRX     6612  6589  6612
 22103    3  MTL     7406  6335  6469  6523  6945  7406  7631  7823  8360
 20204    3  MVA     6453  6033  6453  6618  6901  6911  7142  7208  7314  7372  7438  7999  8261  8375  8443  8535
                           8761
 24536    3  NFF     8717  8545  8612  8642  8656  8661  8717
 24060    3  NIL     8429  5783  8429
 11000    3  NOP     5883  5883
 17424    3  NXT     6080  5982  6062  6080  6115
 24535    3  OFF     8716  8547  8600  8641  8655  8663  8716
 17255    3  OPE     5976  5780  5976
 17433    3  OPT     6087  6068  6069  6087
 17254    3  OPX     5975  5972  5975
   760    3  OTP     5914  5914  6067
 17224    3  PL1     5951  5951  5955
 17230    3  PL3     5955  5955  5963
 17234    3  PL4     5959  5956  5959
 17241    3  PL5     5964  5953  5964
 23663    3  POP     8288  7128  7615  8002  8081  8232  8288
736000    3  QLS     5907  5907  7395
 17244    3  REF     5967  5793  5967
 17046    3  REG     5832  5832  6165  6172  6213  6215  6225  6229  6237  6240  6277  6284  6358  6602  6604
755000    3  STA     5895  5877  5895  6179  6472  6526  6757  7621
756000    3  STQ     5896  5878  5896  6040  6042  6044  6048  6050  6169  7594  8337  8357
 17147    3  STR     5873  5873  6267  6268  6606  6665
740000    3  STX     5885  5873  5874  5875  5876  5885  8741
450000    3  STZ     5894  5894  7006  7972  8423
 24231    3  SUB     8519  5789  8519
440000    3  SXL     5888  5888  6340  8748
 25167    3  SYM     8999  8870  8886  8999
234000    3  SZN     5906  5906  7374
 22045    3  TF1     7377  7373  7377
 22072    3  TF2     7397  7376  7397
 22070    3  TF3     7395  7382  7387  7395
 22075    3  TF4     7400  7394  7400
 22076    3  TFX     7401  7369  7401
 22122    3  TL1     7421  7417  7421
 22123    3  TL2     7422  7420  7422
 22124    3  TL3     7423  7412  7423
 22125    3  TL4     7424  7413  7424
 22126    3  TLX     7425  7411  7425
601000    3  TNZ     5909  5909  8320
710000    3  TRA     5905  5905  6960  7366  7547
600000    3  TZE     5908  5908  7398  7859  8318
 24152    3  UN0     8472  8465  8472  8475
 24153    3  UN1     8473  8440  8473
```

```
OCTAL     SYMBOL     REFERENCES BY ALTER NO,

 24154    3  UN2      8474   8461   8474
 24155    3  UNT      8475   8466   8475
 24150    3  UNX      8470   8434   8470
 24575    3  UPB      8748   5789   8748
 24537    3  VM1      8718   8549   8559   8561   8597   8599   8718
 17142    3  XAQ      5868   5859   5860   5861   5868
 21072    3 ASG1      6886   6867   6886
 21076    3 ASG2      6890   6887   6890
 21465    3 ASG3      7137   6889   7137
 21506    3 ASG4      7154   7143   7154
 21553    3 ASGM      7191   6863   6928   7036   7057   7129   7145   7149   7191
 21026    3 ASGN      6850   5785   6850
 21511    3 ASGR      7157   6885   7136   7153   7157
 21566    3 ASGT      7202   7140   7144   7154   7202
 21513    3 ASI0      7159   6899   7159   7167
 21514    3 ASI1      7160   6914   7160
 21515    3 ASI2      7161   6921   7161
 21516    3 ASI3      7162   6904   7162
 21517    3 ASI4      7163   6927   7163
 21520    3 ASI5      7164   6893   7164
 21521    3 ASI6      7165   6941   7165
 21522    3 ASI7      7166   6937   7166
 21524    3 ASI8      7168   6640   6854   6906   6948   7168
 21563    3 ASQ1      7199   6980   7027   7063   7082   7199
 21564    3 ASQ2      7200   6999   7030   7061   7092   7200
 21146    3 ASR1      6930   6930   6935
 21232    3 ASR2      6966   6966   6977
 21235    3 ASR3      6985   6985   6996
 21256    3 ASR4      7002   7002   7009
 21426    3 ASR5      7106   7106   7125
 21452    3 ASR6      7126   7109   7126
 21454    3 ASR7      7128   7128
 21051    3 AST1      6869   6869   6884
 21060    3 AST2      6876   6875   6876
 21560    3 ASTP      7196   6963   6976   6982   6995   7001   7008   7105   7108   7127   7196
 22672    3 BD10      7783   7771   7783
 22674    3 BD11      7785   7758   7785
 22765    3 BD12      7842   7842   7846
 23065    3 BD13      7906   7849   7906
 23100    3 BDED      7917   7752   7847   7876   7917
 23102    3 BDI1      7919   7736   7744   7919
 23103    3 BDI2      7920   7764   7920   7922
 23104    3 BDI3      7921   7739   7921
 23106    3 BDI4      7923   7741   7783   7923
 23107    3 BDI5      7924   7788   7924   7926
 23110    3 BDI6      7925   7743   7925
 23112    3 BDI7      7927   7795   7927   7930
 23113    3 BDI8      7928   7799   7928
 23114    3 BDI9      7929   7805   7929
 23073    3 BGEN      7912   7606   7826   7912
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 24542    3 BLWB     8721   8556   8573   8577   8622   8647   8721
 22476    3 BND1     7659   7647   7659
 22511    3 BNDT     7670   7657   7670
 22506    3 BNDX     7667   7641   7665   7667
 22366    3 BOOL     7587   7584   7587
 23075    3 BPOS     7914   7659   7674   7677   7701   7714   7718   7725   7797   7873   7878   7881   7914
 24410    3 BU10     8631   8628   8631
 24414    3 BU11     8635   8632   8635
 24416    3 BU12     8637   8634   8637
 24430    3 BU13     8647   8640   8647
 24434    3 BU14     8651   8648   8651
 24443    3 BU15     8658   8646   8658
 24445    3 BU16     8660   8643   8657   8660
 24447    3 BU17     8662   8652   8662
 24452    3 BU18     8665   8564   8665   8669
 24461    3 BU19     8672   8672   8674
 24503    3 BUI1     8690   8542   8690
 24543    3 BUPB     8722   8557   8581   8627   8722
 24273    3 BUS1     8554   8554   8664
 24304    3 BUS2     8563   8563   8595
 24313    3 BUS3     8570   8566   8567   8570
 24320    3 BUS4     8575   8571   8575
 24324    3 BUS5     8579   8576   8579
 24330    3 BUS6     8583   8580   8583
 24334    3 BUS7     8587   8574   8578   8582   8586   8587
 24340    3 BUS8     8591   8569   8591
 24404    3 BUS9     8627   8623   8627
 24534    3 BUSC     8715   8540   8715
 24541    3 BUSE     8720   8555   8568   8639   8720
 24540    3 BUSF     8719   8554   8572   8651   8719
 24476    3 BUSI     8685   8537   8685   8691
 24475    3 BUSX     8684   8531   8684
 21743    3 CASE     7311   5782   7311
 24057    3 CAST     8412   8398   8407   8412
 24056    3 CASX     8411   8391   8411

 17123    3 CHAN     5853   5853   6227   6275   6282
 17671    3 CL11     6250   6248   6250
 17672    3 CL1X     6251   6246   6251
 17700    3 CL31     6257   6254   6257
 17705    3 CL32     6262   6256   6258   6262
 17706    3 CL3X     6263   6252   6263
116000    3 CMPQ     5900   5900   7326
100000    3 CMPX     5890   5890
 24007    3 CONE     8372   5796   8372
 23762    3 CONF     8351   5782   8351
 17707    3 CREG     6264   6249   6255   6264
 24674    3 DBS0     8813   8793   8813   8816
 24677    3 DBST     8816   8794   8816
     5    3 DBT1     8811   8811   8814   8821   8826
     6    3 DBT2     8812   8812   8815   8823   8830
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 24643    3 DBUS    8786  5791  8786
 21007    3 DELV    6835  5786  5997  5998  6642  6835  6859  6950  6951  7334  7400  7666  7828  7906  7907  7908
                          8289  8316  8330  8331  8383  8384  8409  8410  8457  8665  8733  8734  8769
 22356    3 DEN1    7579  7579  7582
 22357    3 DEN2    7580  7561  7580
 22360    3 DEN3    7581  7571  7581
 22361    3 DENT    7582  7576  7582
 22355    3 DENX    7578  7553  7578
 24100    3 DEPR    8430  5795  8430
 22167    3 DER0    7458  7437  7458
 22173    3 DER1    7462  7460  7462
 22204    3 DER2    7471  7463  7471
 22210    3 DER3    7475  7472  7475
 22211    3 DER4    7476  7474  7476
 22216    3 DER5    7481  7477  7481
 22220    3 DER6    7483  7480  7483
 22240    3 DERM    7499  7435  7451  7484  7499
 22233    3 DERX    7494  7426  7457  7492  7494
433000    3 DFLD    5903  5903  6055
457000    3 DFST    5904  5881  5904  6054
 24156    3 DISP    8476  5780  8476
 23645    3 DLEN    8274  5788  8274
 24545    3 DSUB    8724  5791  8724
624000    3 EAX4    5887  5887  6946  7824
 23607    3 ENT0    8244  8234  8244  8247
 23610    3 ENT1    8245  8236  8245
 23612    3 ENTT    8247  8237  8247
 23606    3 ENTX    8243  8231  8243
 23507    3 EPDE    8180  5787  8180
 23613    3 EPDN    8248  5787  8248
 23401    3 EPDV    8110  5787  8110
 23453    3 EPV1    8152  8148  8152
 23460    3 EPV2    8157  8153  8157
 23463    3 EPV3    8160  8160  8165
 23471    3 EPV4    8166  8161  8166
 23473    3 EPV5    8168  8151  8156  8168
 21742    3 ETCX    7310  7302  7310
 24720    3 ETYP    8833  5964  8833
 24627    3 FLEX    8774  5789  8774
470000    3 FSTR    5902  5880  5902  6046
 17507    3 GADL    6137  6137  6943  7023  7107  7443  7577  7630  7658  7766  7790  7807  7822  7902  8146  8185
                          8211  8238  8340  8364  8382  8467  8553  8645  8650  8654  8659  8736  8795  8804  8809
                          8876  8890
 17506    3 GADT    6136  6127  6133  6136
 17505    3 GADX    6135  6126  6135
 17620    3 GEAQ    6209  6209
 22446    3 GEN0    7635  7614  7635  7638
 22447    3 GEN1    7636  7627  7636
 22450    3 GEN2    7637  7605  7637
 22451    3 GENT    7638  7629  7638
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 22445    3 GENX      7634   7607   7634
 22261    3 GOTO      7518   5781   7518
 20354    3 GTMP      6557   6557   6583   7309   7489   6683
 25243    3 GUM1      9041   9041   9109
 25267    3 GUM2      9061   9059   9061
 25271    3 GUM3      9063   9063   9088
 25272    3 GUM4      9064   9062   9064
 25323    3 GUM5      9089   9067   9089   9095   9104
 25334    3 GUM6      9098   9098   9101
 25343    3 GUM7      9105   9099   9105
 25350    3 GUM8      9110   9091   9110   9112
 25360    3 GUMP      9118   9038   9090   9092   9118
 25361    3 GUMS      9119   9039   9040   9107   9114   9119
 25362    3 GUMT      9120   9072   9074   9077   9079   9097   9098   9100   9102   9120
 25357    3 GUMX      9117   9034   9117
 17624    3 GXR1      6213   6213   6219
 17630    3 GXR2      6217   6214   6217
 17647    3 GXR3      6232   6230   6232
 17651    3 GXR4      6234   6226   6234
 17652    3 GXR5      6235   6228   6235
 17653    3 GXR6      6236   6236   6242
 17660    3 GXR7      6241   6238   6241
 17662    3 GXRX      6243   6224   6243
 22406    3 HGEN      7603   5782   7603
 22005    3 HIPX      7345   7340   7345
 20717    3 IDFY      6779   6779   7507   7519
 23720    3 IDNX      8317   8306   8317
 17444    3 IFA1      6102   6088   6102
 17452    3 IFB1      6108   6090   6108
 17447    3 INA1      6105   6089   6105
 17455    3 INB1      6111   6091   6111
 17112    3 INST      5852   5852   6222   6327
 23721    3 ISNT      8318   5783   8318
 17460    3 JMP1      6114   6092   6104   6107   6110   6113   6114
 22027    3 JUMP      7363   5780   7363
 22016    3 LBLX      7354   7346   7354
237000    3 LDAQ      5893   5893   6053   6057   6059
 17607    3 LDAT      6200   6192   6200
 22403    3 LGEN      7600   5782   7600
 25036    3 LIB1      8911   8909   8911
 25054    3 LIB2      8925   8907   8925
 25056    3 LIB3      8927   8924   8927
 23355    3 LLEX      8090   8076   8090
 25102    3 LOAD      8946   8897   8946
 17402    3 MAC0      6062   6062   6076   6103   6106   6109   6112   6116
 17415    3 MAC1      6073   6073   6119   6122   6125
 17421    3 MAC2      6077   6074   6077
 17423    3 MACX      6079   6061   6079   6093
 17040    3 MAXS      5825   5825   6554   6572   6574   7945   8037   8069   8077   8087   8894
 20353    3 MAXX      6556   6553   6556
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 20377    3 MBLK     6576   6003   6576   6587   6635   6891   6917   6923   6967   6986   7003   7011   7017   7342   7506   7557
                            7590   7609   7617   7649   7732   7801   7831   7836   7839   7966   7986   8009   8071   8112   8240
                            8333   8415   8459   8500   8726   8730   8732
 20117    3 MNA1     6400   6394   6400
 20102    3 MNA2     6387   6380   6382   6387
 20106    3 MNA3     6391   6388   6391
 20127    3 MNA4     6408   6401   6408
 20141    3 MNA5     6418   6409   6418
 20155    3 MNA6     6430   6419   6421   6430
 20166    3 MNA7     6439   6434   6439
 20175    3 MNA8     6446   6399   6407   6417   6429   6438   6446
 20202    3 MNA9     6451   6386   6390   6451
 20203    3 MNAX     6452   6375   6452
 17531    3 MOVE     6155   6155   6350   6624   6973   6992   7148   7220   7453   8018   8108   8265   8408   8765
 25130    3 MOVT     8968   8847   8857   8968
 20415    3 MREF     6585   5796   6585
 23516    3 MSCW     8187   5784   5974   7791   8187   8431
 22107    3 MTLX     7410   7406   7410
 20212    3 MVA1     6459   6456   6459
 20221    3 MVA2     6466   6466   6489
 20231    3 MVA3     6474   6460   6474
 20235    3 MVA4     6478   6475   6478
 20251    3 MVA5     6490   6481   6490
 20257    3 MVA6     6496   6479   6496
 20265    3 MVA7     6502   6497   6502
 20300    3 MVA8     6513   6503   6513
 20317    3 MVA9     6528   6514   6528
 20337    3 MVAP     6544   6501   6512   6536   6544
 20330    3 MVAR     6537   6473   6527   6537
 20347    3 MVAT     6552   6552
 20336    3 MVAX     6543   6453   6543   6550
 20701    3 NLOC     6765   6432   6517   6532   6765
 17256    3 OPE0     5977   5977   6848
 17303    3 OPE1     5998   5992   5998
 17326    3 OPE2     6017   6014   6017
 17343    3 OPE3     6030   6016   6030
 17354    3 OPEM     6039   5980   5988   5999   6039
 17345    3 OPES     6032   5985   5994   6032
 17355    3 OPET     6040   6020   6021   6040   7563   7564
 17344    3 OPEX     6031   5977   6031
 17441    3 OPTE     6094   6068   6094
 17441    3 OTAB     6095   6070   6071   6072   6095
 20052    3 PALL     6363   6363   6915   7347   7364   7549   7642
 23664    3 POP1     8289   8289   8293
 23673    3 POPX     8296   8288   8296
 21015    3 PRIM     6841   5793   6841
 17246    3 PROC     5969   5795   5969
 23652    3 PUSH     8279   6759   8074   8279
 24244    3 RBUS     8530   5790   8530
 23636    3 RET1     8267   8262   8267
```

```
OCTAL     SYMBOL     REFERENCES BY ALTER NO,

 23640    3 RET2      8269   8266   8269
 23621    3 RETN      8254   5787   8254
 21674    3 RSL1      7272   7261   7272
 21704    3 RSLX      7280   7248   7271   7280
 21621    3 SEL1      7229   7215   7229
 21633    3 SEL2      7239   7212   7239
 21643    3 SELT      7247   7216   7217   7247
 21642    3 SELX      7246   7203   7228   7238   7246
 24060    3 SKIP      8413   5783   8413   8429
 21705    3 SLCT      7281   7207   7257   7281
757000    3 STAQ      5897   5879   5897   6052   6056   6058
 17606    3 STAT      6199   6174   6199
 17440    3 STOP      6093   6093
 17160    3 STRE      5882   5882   6268   6606   6665
 20677    3 STYE      6763   6676   6730   6763
 24243    3 SUBX      8529   8519   8529
 25176    3 SYM1      9006   9006   9008
 25201    3 SYM2      9009   9005   9009
 25214    3 SYM3      9020   9020   9024
 25217    3 SYM4      9023   9021   9023
 25221    3 SYM5      9025   9022   9025
 25223    3 SYMP      9027   8873   8888   9002   9018   9019   9027   9028   9031
 25226    3 SYMT      9028   8875   8889   9028
 25222    3 SYMX      9026   8999   9026
 25230    3 TAL1      9030   9012   9014   9020   9030
 25232    3 TAL2      9032   9016   9023   9032
 22077    3 TFI1      7402   7381   7402
 22100    3 TFI2      7403   7383   7403
 22017    3 TRAD      7355   6959   7355   7365   7390   7397   7546   7763   7787   7858   8181
 22362    3 TRUE      7583   5784   7583
 24125    3 UNN1      8451   8444   8451
 24133    3 UNN2      8457   8450   8457
 24140    3 UNN3      8462   8442   8462
 24151    3 UNNT      8471   8451   8455   8471
 24213    3 VLWB      8505   5788   8505
 20750    3 VOID      6804   5782   6804   6836
 24214    3 VUPB      8506   5788   8506
 17144    3 XAQ1      5870   5866   5867   5870
 17247    3 XFER      5970   5795   5970
 17430    3AADDR      6084   5995   6084   6118
 17425    3AFLAG      6081   5996   6081   6102   6105
 21123    3ASG22      6911   6908   6911
 21125    3ASG23      6913   6910   6913
 21027    3ASGNE      6851   5785   6851   8307
 21040    3ASGNX      6860   6851   6860
 21525    3ASI10      7169   7169   7173
 21526    3ASI11      7170   7015   7170
 21530    3ASI12      7172   7021   7172
 21532    3ASI20      7174   7077   7100   7118   7123   7174   7190
 21533    3ASI21      7175   7090   7111   7175
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 21534    3ASI22     7176  7073  7176
 21535    3ASI23     7177  7097  7112  7177
 21536    3ASI24     7178  7069  7178
 21537    3ASI25     7179  7088  7113  7179
 21540    3ASI26     7180  7086  7114  7180
 21541    3ASI27     7181  7078  7119  7181
 21542    3ASI30     7182  7100  7101  7103  7124  7182
 21543    3ASI31     7183  7079  7120  7183
 21544    3ASI32     7184  7091  7115  7184
 21545    3ASI33     7185  7074  7185
 21546    3ASI34     7186  7080  7121  7186
 21547    3ASI35     7187  7098  7116  7187
 21550    3ASI36     7188  7070  7188
 21551    3ASI37     7189  7081  7122  7189
 21523    3ASIT1     7167  6942  7167
 21501    3ASIT2     7173  7022  7173
 21552    3ASIT3     7190  7106  7123  7190
 21557    3ASLBL     7195  6954  7033  7059  7133  7195
 21565    3ASMLB     7201  7065  7102  7201
 21561    3ASRX1     7197  6965  6968  6975  7197
 21562    3ASRX2     7198  6984  6987  6994  7198
 17431    3BADDR     6085  5986  6085  6121
 22730    3BD115     7813  7813  7818
 23074    3BDCLR     7913  7624  7643  7661  7673  7675  7698  7704  7705  7712  7716  7720  7746  7812  7841  7870
                           7877  7883  7909  7913
 23122    3BDI10     7935  7834  7850  7935
 23123    3BDI11     7936  7892  7936  7940
 23124    3BDI12     7937  7889  7937
 23125    3BDI13     7938  7895  7938
 23126    3BDI14     7939  7897  7900  7939
 23116    3BDI20     7931  7737  7931  7934
 23117    3BDI21     7932  7796  7932
 23120    3BDI22     7933  7820  7933
 23105    3BDIT1     7922  7765  7922
 23111    3BDIT2     7926  7789  7926
 23115    3BDIT3     7930  7806  7930
 23127    3BDIT4     7940  7901  7940
 23121    3BDIT5     7934  7821  7934
 17426    3BFLAG     6082  5987  6082  6108  6111
 25233    3BLNKS     9033  9017  9033
 22507    3BNDI1     7668  7656  7668  7670
 22510    3BNDI2     7669  7653  7669
 24544    3BNLWB     8723  8558  8585  8631  8723
 22402    3BOOLT     7599  7587  7596  7599
 22401    3BOOLX     7598  7588  7598
 22454    3BOUND     7641  5789  7641
 23077    3BPOS1     7916  7680  7684  7710  7916
 24521    3BUS10     8704  8603  8704  8708
 24522    3BUS11     8705  8614  8705
 24523    3BUS12     8706  8606  8706
```

```
OCTAL     SYMBOL      REFERENCES BY ALTER NO,

 24524    3BUS13       8707   8617   8707
 24526    3BUS14       8709   8609   8709   8713
 24527    3BUS15       8710   8619   8710
 24530    3BUS16       8711   8611   8711
 24531    3BUS17       8712   8621   8712
 24505    3BUSI0       8692   8625   8692   8696
 24506    3BUSI1       8693   8601   8693
 24507    3BUSI2       8694   8608   8694
 24510    3BUSI3       8695   8543   8695
 24512    3BUSI4       8697   8605   8630   8697   8703
 24513    3BUSI5       8698   8602   8626   8698
 24514    3BUSI6       8699   8637   8699
 24515    3BUSI7       8700   8616   8700
 24516    3BUSI8       8701   8638   8701
 24517    3BUSI9       8702   8613   8702
 24504    3BUST0       8691   8552   8691
 24511    3BUST1       8696   8649   8696
 24520    3BUST2       8703   8653   8703
 24525    3BUST3       8708   8644   8708
 24532    3BUST4       8713   8658   8713
 21757    3CASE1       7323   7315   7323
 21774    3CASET       7336   7316   7318   7336
 21773    3CASEX       7335   7311   7335
 24032    3CASGN       8391   5782   8391
 21775    3CASI1       7337   7328   7337
 21776    3CASI2       7338   7330   7338
 21777    3CASI3       7339   7332   7339
 24025    3CONE0       8386   8377   8386   8390
 24027    3CONE1       8388   8380   8388
 24031    3CONET       8390   8381   8390
 24024    3CONEX       8385   8372   8385
 24001    3CONF1       8366   8362   8366   8371
 24006    3CONFT       8371   8363   8371
 24000    3CONFX       8365   8351   8365
 17242    3CPNTR       5965   5915   5937   5944   5965
 17720    3CREGX       6273   6264   6273
 24717    3DBST1       8832   8808   8832
 24700    3DBUS0       8817   8801   8817   8824
 24701    3DBUS1       8818   8797   8818
 24702    3DBUS2       8819   8799   8819
 24705    3DBUS3       8822   8802   8822
 24710    3DBUS4       8825   8825   8832
 24675    3DBUSL       8796   8796   8807
 24707    3DBUST       8824   8803   8824
 24673    3DBUSX       8810   8786   8810
 21730    3DELTA       7300   7218   7229   7239   7262   7272   7287   7294   7300
 21011    3DELV1       6837   6837   6839
 21014    3DELVX       6840   6835   6840
 22324    3DENOT       7553   5785   7553
 24103    3DEPRX       8433   8430   8433
```

```
OCTAL     SYMBOL     REFERENCES BY ALTER NO,

 22127    3DEREF      7426   5793   7426
 22234    3DERI1      7495   7440   7495   7498
 22236    3DERI2      7497   7441   7497
 22237    3DERIT      7498   7442   7498
 24171    3DISP0      8487   8485   8487
 24174    3DISP1      8490   8490   8493
 24202    3DISP2      8496   8496   8498
 24212    3DISPC      8504   8488   8492   8504
 24211    3DISPX      8503   8477   8503
 23651    3DLENX      8278   8274   8278
 24563    3DSUB0      8738   8738   8740
 24565    3DSUBT      8740   8735   8740
 24562    3DSUBX      8737   8724   8737
 24157    3EDISP      8477   5780   8477
 17270    3EMPTY      5971   5795   5971
 23572    3ENTER      8231   5780   7810   8231   8432
 23515    3EPDEX      8186   8180   8186
 23620    3EPDNO      8253   8250   8253
 23617    3EPDNX      8252   8248   8252
 23506    3EPDYT      8179   8179   8184
 23476    3EPDV0      8171   8116   8125   8171
 23477    3EPDV1      8172   8124   8127   8172
 23501    3EPDV2      8174   8118   8174
 23500    3EPDV3      8173   8144   8173   8175
 23504    3EPDV4      8177   8177   8179   8183
 23505    3EPDV5      8178   8122   8178
 23503    3EPDV6      8176   8120   8168   8176
 23502    3EPDVT      8175   8145   8175
 23475    3EPDVX      8170   8110   8170
 23247    3ERNG1      8020   8014   8020
 23252    3ERNG2      8023   8019   8023
 23214    3ERNGE      7993   5787   7993
 23261    3ERNGF      8030   7998   8000   8013   8030
 23263    3ERNGI      8032   8027   8032
 23233    3ERNGM      8008   7997   8008
 23262    3ERNGT      8031   8001   8012   8031
 23260    3ERNGX      8029   7993   8029
 24723    3ETYP1      8836   8836   8852
 24741    3ETYP2      8850   8839   8841   8845   8850
 24745    3ETYP3      8854   8854   8865
 24756    3ETYP4      8863   8856   8863
 24757    3ETYP5      8864   8862   8864
 22365    3FALSE      7586   5784   7586
 21731    3FMODE      7301   7219   7298   7301
 23363    3FORM1      8096   8096   8098
 23356    3FORMP      8091   5785   8091
 23400    3FORMX      8109   8091   8109
 17511    3GADL0      6139   6139   6143
 17514    3GADL1      6142   6140   6142
 17516    3GADLT      6144   6138   6139   6144
```

```
01557 01  05-26-72   18,180      ALGOL68                                                              PAGE    298

OCTAL     SYMBOL     REFERENCES BY ALTER NO,

 17513    3GADLX      6141  6137  6141
 22322    3GOTI1      7551  7528  7551
 22323    3GOTI2      7552  7540  7552
 22272    3GOTO1      7527  7527  7531
 22277    3GOTO2      7532  7524  7527  7532
 22311    3GOTO3      7542  7522  7542
 22321    3GOTOX      7550  7518  7550
 20364    3GTMP1      6565  6559  6565
 20376    3GTMPX      6575  6557  6573  6575
 25255    3GUM11      9051  9049  9051
 25307    3GUM42      9077  9077  9082
 25313    3GUM43      9081  9078  9081
 25315    3GUM44      9083  9083  9121
 25317    3GUM48      9085  9069  9085
 25364    3GUMT1      9122  9076  9081  9122
 25363    3GUMTL      9121  9075  9121
 22256    3IDEN6      7515  7513  7515
 22241    3IDENT      7500  5785  5973  7500  8298
 22260    3IDENX      7517  7500  7517
 20730    3IDFY1      6788  6788  6794  6801
 20737    3IDFY2      6795  6790  6795
 20723    3IDFYB      6783  6783  7762
 20746    3IDFYC      6802  6783  6791  6802  7510  7512  7520  7526  7530  7767  7773  7777
 20747    3IDFYF      6803  6784  6797  6803  7521
 20740    3IDFYX      6796  6779  6796  7761
 23712    3IDNT1      8311  8311  8313
 23705    3IDNTE      8306  5784  8306
 23704    3IDNTX      8305  8297  8305
 23674    3IDNTY      8297  5784  8297
 22034    3JUMPX      7368  7363  7368
 17042    3LLINK      5827  5827  6786  7942  7958  7975  8007  8034  8051  8054  8067  8079  8089
 17044    3LSTMK      5829  5829  6677  6741  7959  8023  8204  8280  8286  8292  8295
 17041    3MAXST      5826  5826  8078  8123  8137  8276
 20414    3MBLKX      6584  6576  6584
 17541    3MOVB1      6163  6154  6163
 17552    3MOVE1      6172  6166  6172
 17557    3MOVE2      6177  6173  6177
 17563    3MOVE3      6181  6171  6181  6189
 17517    3MOVEB      6145  6145  8449
 17604    3MOVED      6197  6163  6185  6197
 17605    3MOVEF      6198  6164  6176  6190  6198
 17602    3MOVET      6196  6146  6151  6152  6156  6157  6168  6170  6178  6180  6181  6183  6186  6187  6196
 17600    3MOVEX      6194  6145  6149  6155  6158  6161  6191  6194
 25137    3MOVT1      8975  8975  8987
 25144    3MOVT2      8980  8977  8980
 25145    3MOVT3      8981  8979  8981
 25166    3MOVTT      8998  8973  8974  8981  8988  8998
 25165    3MOVTX      8997  8968  8997
 20420    3MREFX      6588  6585  6588
 23550    3MSCW0      8213  8208  8213  8230
```

```
OCTAL    SYMBOL    REFERENCES BY ALTER NO,

 23551   3MSCW1     8214  8192  8214
 23552   3MSCW2     8215  8195  8215
 23555   3MSCW4     8218  8193  8218
 23561   3MSCW5     8222  8199  8222
 23565   3MSCW6     8226  8200  8226
 23566   3MSCW7     8227  8196  8227
 23567   3MSCW8     8228  8201  8228
 23570   3MSCW9     8229  8197  8229
 23571   3MSCWT     8230  8210  8230
 23547   3MSCWX     8212  8187  8212
 20344   3MVAP1     6549  6477  6495  6549
 20335   3MVAR1     6542  6458  6542
 20707   3NLOCL     6771  6771  6778
 20710   3NLOCX     6772  6765  6772

 25100   3OBJCT     8945  8899  8919  8945
 17352   3OPES1     6037  6034  6037
 17353   3OPESX     6038  6032  6036  6038
 17401   3OPETE     6060  6021  6060  7564
 17444   3OTABE     6101  6071  6101
 20065   3PALLX     6374  6363  6374
 17045   3PARAM     5830  5830  5941  5976  6780  6842  6953  7134  7288  7305  7323  7341  7348  7356  7503  7542
                          7545  7554  7558  7616  7672  7756  7786  7809  7853  7861  7904  7974  8049  8092  8111
                          8239  8359  8414  8458  8478  8499  8515  8679  8743  8750  8780  8787
 17217   3PAS3E     5946  5940  5946
 17206   3PAS3L     5937  5937  5945
 17173   3PAS3R     5926  5926  5929
 17160   3PASS3     5915  5765  5915
 21024   3PRIM1     6848  6845  6848
 21025   3PRIMX     6849  6841  6849
 17730   3PURGE     6281  6231  6281  6365  6367  6369  6371  6373  6662
 23662   3PUSHX     8287  8279  8287
 23643   3RETN0     8272  8257  8272
 23644   3RETN1     8273  8269  8273
 23642   3RETNX     8271  8254  8271
 22453   3RHGEN     7640  7604  7640
 22452   3RLGEN     7639  7601  7639
 21644   3RSLCT     7248  5783  7248
 21567   3SELCT     7203  5783  7203
 24071   3SKIP1     8422  8422  8427
 24077   3SKIPX     8428  8413  8421  8428
 21715   3SLCT1     7289  7289  7296
 21725   3SLCT2     7297  7290  7297
 21727   3SLCTX     7299  7281  7299
 23175   3SRNG1     7978  5936  7978  7990
 23210   3SRNG2     7989  7983  7989
 23130   3SRNGE     7941  5787  7941
 23213   3SRNGI     7992  7963  7992
 23212   3SRNGX     7991  5935  7941  7978  7991
 17243   3STRCT     5966  5793  5966
 20531   3STYP1     6662  6662  6666
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 20551    3STYP2     6678  6678  6731
 20623    3STYP3     6719  6699  6719
 20634    3STYP4     6728  6718  6728
 20567    3STYP5     6692  6689  6692
 20571    3STYP6     6694  6691  6694
 20572    3STYP7     6695  6681  6695
 20526    3STYPE     6659  6659  6894  7610  7967  8209
 20676    3STYPP     6762  6672  6733  6735  6762
 20675    3STYPR     6761  6660  6661  6663  6664  6761
 20700    3STYPT     6764  6740  6747  6764
 20674    3STYPX     6760  6659  6760
 25000    3SYMD1     8881  8869  8881
 25001    3SYMD2     8882  8882  8893
 25015    3SYMD3     8894  8885  8894
 25227    3SYMDT     9029  8881  8882  8892  9029
 24764    3SYMR1     8869  8869  8880
 17432    3TADDR     6086  6007  6086  6124
 25231    3TAL2I     9031  9015  9031
 17427    3TFLAG     6083  6013  6083
 22026    3TRADX     7362  7355  7362
 22364    3TRUET     7585  7583  7585
 24104    3UNION     8434  5795  8434
 24217    3VEPTY     8509  5788  8509
 24533    3VMARK     8714  8520  8525  8528  8532  8548  8666  8670  8676  8714
 24215    3VNLWB     8507  5788  8507
 20766    3VOID1     6818  6813  6818
 20777    3VOID2     6827  6822  6827
 21006    3VOIDI     6834  6815  6816  6834
 21005    3VOIDX     6833  6804  6833
 24216    3VSBCT     8508  5788  8508
 17663    3XREGI     6244  6223  6244  6396  6404  6414  6426  6442  6509  6767  6773
 17664    3XREGM     6245  6221  6245  6398  6406  6416  6428  6437  6441  6445  6511  6535  6774
  6411    A    S     1747  1661  1662  1664  1668  1674  1677  1747
     1    A   DF      298   298  1130  1133  1165  1168  1173  1176  1261  1466  2359  2532  2610  2809  3342  3461
                            4522  4744  4750  4832
  1675    A   IN      222   222  1019  1030  1038  1095  1098  1099  1100  1101  1652  2854
 11305    A   M1     3093  2659  2838  2843  2882  2884  3093
 11306    A   M2     3094  2662  2829  2834  3094
  4774    A   NS     1113  1113  1237
  6210    A   OK     1662  1654  1662  1667  1672  1678  1736  1863  1927  1977  1980  1982  1987  1989  1991  1992
                            1995  1999  2002  2004  2009  2013  2017  2021  2026  2830  2040  2084  2106  2119  2173
                            2190  2212  2217  2220  2266  2276  2289  2294  2299  2342  2354  2396  2423  2428  2433
                            2451  2455  2889  3127  3136  3164  3360  3411  3431  3451  3481  3497  3521  3527  3578
                            3609  3616  3652  3814  3817  3821  3890  3925  3949  3964  3993  4032  4040  4049  4053
                            4058  4063  4067  4070  4074  4079  4082  4085  4088  4891  4093  4096  4099  4101  4103
                            4106  4108  4111  4133  4136  4202  4213  4218  4224  4227  4261  4270  4289  4327  4336
                            4374  4377  4400  4430  4500  4516  4552  4557  4560  4566  4619  4661  4668  4673  4678
                            4689  4694  4705  4711  4716  4721  4726  4741  4762
     0    A   SC      297   297  1118  1126  1129  1179  1180  1184  1276  1471  1685  1852  1861  2038  2091  2554
                            2646  3162  3189  3358  3409  3443  3448  3473  3478  4967  9011  9071
```

```
01557 01  05-26-72   18,180      ALGOL68                                                                    PAGE    301

OCTAL     SYMBOL    REFERENCES BY ALTER NO,

  6470    A  BCD     1890  1890  2513  2550
 10210    A  CL1     2532  2532  2647
 10213    A  CL2     2535  2535  2547
 10215    A  CL3     2537  2537  2558
 10405    A  CL4     2645  2537  2645
 10224    A  CL5     2544  2541  2544
  6461    A  DLL     1883  1865  1879  1883  1885
 10301    A  EM1     2579  2551  2578  2579
  6226    A  END     1676  1676  1737

  1677    A  EOF      224   224  1022  1036  1650  2850
  6274    A  ER1     1705  1705
  6305    A  ER2     1708  1708
  6312    A  ER3     1711  1711
  6323    A  ER4     1714  1714
  6333    A  ER5     1717  1717
  6344    A  ER6     1720  1720
  6354    A  ER7     1723  1723
  6361    A  ER8     1726  1726
  6231    A  ERR     1679  1679
  6462    A  GLL     1884  1884  4727  4913
 10306    A  IDN     2582  2562  2582
  1674    A  INI      221   221  1651  2853
  1676    A  INP      223   223  1029
 10550    A  NEQ     2745  2677  2680  2712  2727  2745  2750
  4666    A  NID     1044  1044  1115
  4764    A  NIX     1105  1102  1105
  5023    A  NS1     1136  1136  1142  1144
  5030    A  NS2     1141  1137  1141
  5034    A  NS3     1145  1119  1145
  5041    A  NS4     1150  1150  1219
  5143    A  NS5     1216  1151  1216
  5146    A  NS6     1219  1217  1219
  5147    A  NS7     1220  1153  1218  1220
  5053    A  NS8     1160  1140  1160
  5057    A  NS9     1164  1164  1204
  5151    A  NSI     1222  1143  1212  1222
  5001    A  NSL     1118  1118  1221
  4776    A  NSR     1115  1115  1158
  4777    A  NSX     1116  1113  1116  1206
 10145    A  PL1     2497  2497  2516
 10162    A  PL2     2510  2510  2525
 10171    A  PL3     2517  2506  2509  2517
 10200    A  PL4     2524  2524  2528

 10205    A  PL5     2529  2504  2529
 10430    A  POP     2665  2665  2674  2717  2720
 11021    A  REF     2914  2900  2914
 11074    A  ROW     2957  2908  2910  2957
 10676    A  SQ1     2831  2831  2846
 10706    A  SQ2     2839  2656  2839
 11122    A  STR     2978  2904  2978
```

```
OCTAL     SYMBOL     REFERENCES BY ALTER NO,

   140    A  TCB        63    63   156   479
  6437    A  TLU      1865  1865  2611  2812  3343  3462  4524  4836
 11252    A  UN1      3066  3066  3079
 11261    A  UN2      3073  3071  3073
 10745    A  UP1      2870  2870  2875
 10750    A  UP2      2873  2873  2880
 10637    A  XFR      2800  2649  2652  2800
  6517    A ASCZ      1910  1904  1910
  6511    A BCDT      1907  1895  1907
  6510    A BCDX      1906  1890  1906
   144    A BLEN       215   215   217   220   223  1024
 10230    A CLER      2548  2548  2564  2566  2569  2571  2573  2574  2575
 10226    A CLOK      2546  2546  2567  2570  2640
  1526    A CRLF       218   218   221   222
 10674    A DONE      2829  2742  2752  2829
     2    A ELEN     11224  1120  1169  1220  1263  1480
  6245    A ERR1      1691  1691  1698
  6255    A ERR2      1699  1692  1699
  6256    A ERRC      1700  1690  1697  1700
  6241    A ERRF      1687  1687  4622  4969
  6233    A ERRS      1681  1670  1681
 10544    A EXIT      2741  2666  2741  2744
  6223    A FAIL      1673  1673  1738
  6467    A GLLX      1889  1884  1889
  1700    A IBUF       225   225   226
 10313    A IDN1      2587  2587  2618
 10343    A IDN2      2611  2611  2635
 10353    A IDN3      2619  2615  2619
 10355    A IDN4      2621  2595  2600  2612  2617  2621
 10365    A IDN5      2629  2591  2629
 10374    A IDN6      2636  2622  2636
 10401    A IDN7      2641  2630  2641
 10331    A IDNG      2601  2593  2601
  6032    A INIT      1646    63  1646
  6457    A LAST      1881  1866  1881  2536  2557  2585  2602  2626  2633  2634  2638  2643
 10727    A LBL1      2856  2856  2868
 10736    A LBL2      2863  2863
 10741    A LBL3      2866  2859  2862  2866
  6436    A MATT      1864  1683  1850  1856  1864  3169  3174  3179  3182  3366  3369  3415  3420  3433
 10247    A MTAB      2563  2559  2563
 10515    A MULT      2718  2707  2709  2711  2718
  4662    A NCH1      1040  1020  1040
  4641    A NCHX      1023  1021  1023  1028
  1753    A NICH       229   229  1050  1074  1103
  4731    A NICR      1078   240  1078
  4733    A NILN      1080   235   237  1080
  4674    A NILP      1050  1050  1062  1065  1071  1073  1083  1087  1089  1093
  4743    A NIQU      1088   244   286   288  1088
  4757    A NIS2      1100  1096  1100
  4672    A NISK      1048   236   241   276  1048
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

  1764    A NIT1       238   238  1048
  2022    A NIT2       253   253  1061
  2032    A NIT3       257   257  1064  1070
  2044    A NIT4       262   262  1066
  2052    A NIT5       265   265  1072
  2064    A NIT6       270   270  1076
  1762    A NIT7       237   237  1078
  1756    A NIT8       235   235  1080
  2072    A NIT9       273   273  1082
  5124    A NS10      1201  1166  1201
  5130    A NS11      1205  1164  1205
  5133    A NS12      1208  1194  1197  1208
  5163    A NSIX      1232  1222  1232
 10251    A OTAB      2565  2560  2565
  5164    A PEEK      1233  1233  1691  1854  2031  3097  3154  3183  3335  3367  3434  3452
 11077    A PROC      2960  2902  2960
 10254    A PTAB      2568  2561  2568
 10534    A PUSH      2733  2714  2716  2729  2731  2733
 10562    A RCHK      2755  2755  4318  6556  6626  6680  6821  7436  7491  7848
 11024    A REF1      2917  2917  2923
 11033    A REF2      2924  2921  2924
 11036    A REF3      2927  2925  2927
 11043    A REF4      2932  2930  2932
 11061    A REF5      2946  2934  2946
 11064    A REF6      2949  2945  2949
 11073    A REFM      2956  2915  2950  2956  2958
 11072    A REFT      2955  2914  2922  2926  2929  2933  2936  2940  2955  2957
 11025    A ROW1      2918  2918  2959
     3    A SETC
 10773    A SETM      2892  2883  2892  2995  3068
     5    A SETN
   114    A SETW
 10510    A SNGL      2713  2701  2703  2705  2713
 11141    A STR1      2993  2993  3011
 11204    A STR2      3028  3028  3038
 11217    A STR3      3039  3039  3041
  6516    A SUPC      1909  1902  1909
 10305    A TEMP      2581  2497  2508  2581  2724  2726
  6440    A TLU0      1866  1866  1874
  6445    A TLU1      1871  1871  1880
  6451    A TLU2      1875  1872  1875
 11304    A UNT1      3092  3048  3092
 35234    A WORK     11852  1625  1629  1635  1994  2005  2010  2015  2018  2022  2024  2028  2041  2078  2079  2081
                           2082  2102  2113  2117  2129  2136  2145  2150  2160  2161  2166  2172  2174  2180  2182
                           2189  2201  2210  2223  2246  2247  2251  2263  2273  2283  2292  2298  2300  2307  2337
                           2340  2351  2398  2402  2421  2431  2443  2472  2586  2601  2619  2621  2629  2631  2657
                           2660  2663  2665  2669  2736  2751  2753  2979  2997  3037  3039  3040  3482  3484  3493
                           3505  3507  3510  3529  3531  3547  3561  3597  3602  3621  3623  3629  3631  3638  3640
                           3645  3648  3656  3662  3681  3762  3774  3775  3779  3781  3786  3789  3792  3833  3864
                           3865  3882  3887  3888  3962  3969  3990  3991  4004  4006  4010  4029  4033  4059  4061
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

                           4072  4077  4145  4189  4197  4246  4250  4254  4256  4259  4262  4271  4281  4292  4296
                           4301  4303  4306  4324  4331  4333  4344  4347  4401  4403  4426  4431  4433  4488  4497
                           4498  4533  4542  4545  4547  4550  4569  4571  4614  4629  4631  4634  4636  4641  4656
                           4662  4666  4685  4735  4758  4778  4809  4863  4865  4879  4890  4904  4911  4916  4924
                           4932  4958  4997  5090  5118  5121  5128  5137  5139  5158  5165  5231  5233  5247  5251
                           5252  5261  5265  5267  5270  5272  5363  5372  5373  5374  5375  5376  5377  5379  5389
                           5390  5391  5392  5393  5394  5402  5424  5429  5468  5486  5501  5503  5506  5510  5513
                           5519  5598  5606  5620  5639  5645  5653  5671  5681  5697  5703  5708  5716  5719  5729
                           5916  5983  6578  6579  6580  6673  6751  6805  6823  6827  6837  6838  6855  6895  6978
                           6997  7066  7137  7204  7249  7303  7312  7370  7427  7611  7792  7886  7950  7968  7994
                           8042  8095  8188  8259  8283  8290  8308  8323  8353  8373  8392  8399  8435  8489  8496
                           8497  8512  8526  8668  8677  8755
  10630   A XFER     2793  1257  2667  2670  2759  2793  2877  2893  2918  3502  3549  3806  3811  3823  3998  4149
                           4158  4168  4183  4316  4350  4575  4582  4605  4645  4765  4842  5018  5033  5181  5195
                           5239  5288  5323  5404  5418  5452  5473  5491  5524  5546  5615  5989  6000  6018  6147
                           6159  6467  6568  6696  6864  6931  6939  7130  7150  7252  7282  7292  7407  7430  7625
                           7644  7662  7676  7686  7723  7814  7842  7910  8395  8438  8479  8790  8871  9044
  10640   A XFR1     2801  2801  2826
  10653   A XFR2     2812  2812
  10660   A XFR3     2817  2813  2817
  10664   A XFR4     2821  2816  2821
  10670   A XFR5     2825  2806  2820  2825
  10672   A XFR6     2827  2803  2827
  10666   A XFRX     2823  2800  2823  2827
   1754   ABACK4      230   230  1094  1097
   6515   ABCDTP     1908  1896  1905  1908
  10304   ABLNKS     2580  2521  2580
  10240   ACLDEL     2556  2556  2644
  10243   ACLTAB     2559  2545  2559
   1524   ACPRAM      217   217  1026
   2615   ADDR1       531   531   535
   2614   ADDR        530   509   513   530
  10300   AEM1SC     2578  2552  2578
 777777   AERROR      214   214   295  1033  1060
   6257   AERRS1     1701  1681  1701
   6267   AERRS2     1703  1687  1703
  10556   AFAIL1     2751  2746  2751  2754
   1751   AIDENT      227   227  1046  1051  1063  1069  1075  1104  1106  1108  1109  1124  1136  1148
  10270   AIDREG     2577  2577  2583  2636  2641
  10260   AIDTAB     2572  2572  2639
   1530   AINBUF      220   217   220   223  1024  1025
    114   AITABX    11778   299
      3   AITACX    11779   299
   6460   ALEVEL     1882  1871  1875  1878  1882  2534  2540  2543  2582  2604  2607  2624  2637  2642  2811  3334
                           3458  4521  4835
   2162   ALNGCT      307   307  1114  1157  1163  1203
   2157   ALONGT      305   305  1185
   6417   AMATCH     1849  1739  1849
  10516   AMULT1     2719  2719  2732
   4635   ANCHAR     1019  1019  1035  1039  1040  1043  1052
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO.

  4751    ANIB2X    1094   263   264   271   272  1094
  4754    ANIBEX    1097   255   256   260   261   268   269   274   275   282   283   284   287   289   290   292
                           294  1097
  1752    ANICHR     228   228   229  1047  1055
  4737    ANICOM    1084   243   277  1084
  4717    ANIDC1    1068   262  1068
  4715    ANIDEC    1066   258  1066
  4710    ANIDEN    1061   238   253   254  1061
  4712    ANIDG1    1063   270  1063
  4713    ANIDIG    1064   239   257   267   273  1064
  4735    ANIDOT    1082   242  1082
  4773    ANIEOF    1112   252   278  1044  1110  1112
  4725    ANIEX1    1074   265   266  1074
  4723    ANIEXP    1072   259  1072
  4747    ANIFOR    1092   245   293  1092
  4676    ANILP1    1052  1049  1052  1067  1077  1079  1081  1085  1091
  4741    ANIOPR    1086   251   279   280   281  1086
  4745    ANIQU1    1090   285  1090
  2100    ANIT10     276   276  1084
  2106    ANIT11     279   279  1086
  2120    ANIT12     284   284  1088
  2130    ANIT13     288   288  1090
  2136    ANIT14     291   291  1092
  1750    ANITAL     226   226  1045  1105
  4762    ANIXIT    1103   246   247   248   249   250   291  1103
  6216    ANOMAT    1668  1668  1675  1855  1857  2032  2034  3098  3100  3155  3157  3184  3187  3203  3336  3339
                          3344  3347  3368  3370  3405  3426  3428  3430  3435  3438  3453  3456  3463  3466
  5052    ANS8.1    1159  1156  1159
 10456    ANSWAP    2687  2683  2687
  2150    ANTMSK     296   296  1128  1176
 10135    APASS2    2489  2489
  5165    APEEK1    1234  1234  1250
  5174    APEEK2    1241  1236  1241
  2152    APEEKF     300   300  1241  1242  1245  1247  1693  1858  2035  2851  3158  3355  3406  3445  3475
  5206    APEEKX    1251  1233  1243  1248  1251  1252
 11116    APROCT    2974  2964  2974
 10543    APUSHX    2740  2733  2740
  2153    AQEEKF     301   301  1234  1235  1239  1240  1244  1246  1249  2852  3337  3436  3446  3450  3454  3476
                          3480
 10566    ARCHK1    2759  2759  2786
 10574    ARCHK2    2765  2762  2765  2766
 10601    ARCHK3    2770  2764  2770
 10607    ARCHK4    2776  2776  2790
 10617    ARCHK5    2784  2782  2784
 10622    ARCHK6    2787  2771  2781  2787
 10626    ARCHK7    2791  2788  2791
 10627    ARCHKX    2792  2755  2767  2792
   101    ASETC0
   114    ASETC1
   120    ASETC2
```

```
OCTAL    SYMBOL    REFERENCES BY ALTER NO,

     0   ASETC3
     2   ASETCI
 10761   ASETML     2882  2882  2891
 10760   ASETMS     2881  2872  2881
   113   ASETWI
 35235   ASTACK    11853  1255  1282  1303  1314  1355  1370  1372  1375  1376  1403  1405  1408  1409  1419  1615
                          1631  1637  1665  1673  1676  1997  2000  2003  2007  2043  2068  2149  2164  2184  2225
                          2235  2244  2287  2290  2658  2690  2692  2741  2743  2745  2747  2757  2765  2774  2776
                          2777  2778  2783  2787  2789  2978  2981  2986  2988  2990  2999  3016  3042  3043  3044
                          3056  3061  3063  3069  3089  3090  3110  3115  3131  3134  3439  3488  3498  3542  3582
                          3614  3617  3835  3847  3878  3886  3905  3945  3996  3999  4065  4068  4252  4264  4267
                          4276  4278  4558  4561  4588  4627  4664  4945  4947  5094  5124  5132  5141  5143  5149
                          5157  5160  5161  5163  5171  5217  5224  5226  5228  5256  5257  5276  5283  5295  5301
                          5306  5310  5315  5321  5331  5332  5346  5351  5357  5386  5387  5399  5415  5437  5442
                          5448  5508  5572  5603  5608  5610  5625  5628  5632  5635  5640  6861  6870  6872  6879
                          6880  7025  7028  7031  7034  7037  7040  7044  7047  7050  7052  7054  7056  7058  7060
                          7062  7064  7157  7696  7699  7702  7717  7719  7721  7862  7865  7868  7871  7874  7880
                          7882  7884  7898  7903  7943  7948  7953  7955  8003  8004  8006  8035  8040  8045  8047
                          8082  8083  8088  8281  8294  8516  8521  8523  8550  8672  8673  8675  9035  9041  9042
                          9096  9110  9111
 10420   ASTART     2657  2657  2837
  2151   ASYMSC      299   299  1127  1135  1139  1141  1177  1188  1199  1208  1224  1229  1231
   112   ATABLE    11327  2033  3156  3186
  2154   ATEMP1      302   302  1145  1147  1150  1181  1189  1196  1211  1214
  2155   ATEMP2      303   303  1149  1152  1186  1193  1216
  2156   ATEMP3      304   304  1191  1200
  6375   ATRAC1     1735  1733  1735
  6367   ATRACE     1729  1663  1729
  6377   ATRACT     1737  1729  1737
 11225   AUNION     3045  2906  3045
 10631   AXFER1     2794  2794  2799
 10634   AXFERX     2797  2793  2797
400000   B   FA     5810  5810  6250  6377  6502  7018  7483
200000   B   FB     5811  5811  6247  6250  6290  6296  7459  7462  7469
100000   B   FC     5812  5812  6015  6303  6310  6319  6354  6408  6496  6591  6610  6616  6654  6698  6753  6831
                          6918  6924  7012  7234  7244  7267  7278  7343  7483  7650  7733  7802  7829  7987  8010
                          8072  8099  8113  8241  8314  8416  8468  8501  8727
 40000   B   FD     5813  5813  6381  6430  6433  6480  6513  6655  7226  7244  7278  7459  7471  7483  8300  8302
 20000   B   FE     5814  5814  6418  6430  6490  6528  6655  7226  7244  7278  7454  7473  7475  7511  8303
 10000   B   FF     5815  5815  6303  6319  6379  6420  6478  6655  7226  7244  7278  7454  7475  7514
  4000   B   FG     5816  5816  6029  6253  6262  6298  6311  6393  6455  6541  7470  7574  7592  7619  8335  8355
  2000   B   FH     5817  5817  6257  6262  6313  6321  6353  6387  6449  6459  7459  7476  7478
  1000   B   FI     5818  5818  6257  6262  6313  6343  6353  6400  6474  6548  7479
   400   B   FJ     5819  5819  6591  6609  6632  6688  6812
 17034   B   FS     5796  5796
 17031   B   MA     5796  5796
 17026   B  MAX     5796  5796
 16773   B  MMI     5793  2205  5793
 16762   B  REF     5793  5793
 16770   B  ROW     5793  2386  3953  5793  7749
```

```
OCTAL     SYMBOL     REFERENCES BY ALTER NO,

   160    B  TCB        64    64   477
 17023    B CONE      5796  5796
 17004    B DEPR      5795  5795
 17020    B MREF      5796  5796
 16754    B PRIM      5793  5793
 17001    B PROC      5795  5795
 16776    B ROWE      5794  5794
 17012    B XFER      5795  5795
 16765    BDEREF      5793  5793
 17015    BEMPTY      5795  5795
   400    BLEN         612   608   609   611   612   614
 17037    BOTBLE      5797  5797
 16757    BSTRCT      5793  2066  3851  5793  7708
  3135    BUF1         576    64   576   579
  3157    BUF2         577   577   605
  3164    BUF3         582   582   585
  3170    BUF4         586   583   586
  3253    BUF5         605   603   605
  3262    BUF          613   608   609   611   613
  3254    BUFS         606   547   553   576   606
 17007    BUNION      5795  5795
  4024    CAT          660   660  8920
  3702    CLOSE        623   623  8939 10107
   360    CREG         125   125   126   127   661   662   663   666
    75    D   LN
  6413    D   OP      1844  1844  2280  3166  4509  4839
    56    D   PI
   102    D  COS
    70    D  EXP
     1    D  INT
   140    D  RND
   114    D  SIN
   126    D  TAN
   107    D ACOS
   121    D ASIN
   133    D ATAN
    25    D BITS
    13    D BOOL
    20    D CHAR
   152    D LIBI
   145    D LRND
  6412    D MODE      1843  1269  1843  2260  2545  2614  2815  3171
     6    D REAL
    63    D SQRT
    44    D TRUE
    17    D                 6658  6834  7159  7161  7163  7164  7170  7172  7174  7175  7176  7177  7178  7179  7180
                            7181  7183  7184  7185  7186  7187  7188  7189  7551  7552  7635  7668  7669  7919  7921
                            7923  7925  7927  7929  7931  7932  7935  7936  7937  7938  7992  8032  8171  8174  8176
                            8178  8213  8214  8215  8218  8222  8224  8244  8245  8388  8474  8685  8687  8690  8692
                            8695  8697  8698  8700  8702  8705  8707  8710  8712  8717  8814  8815  8821  8823  8825
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO.

                              8826  8828  8830  9136  9137
   157    DBLANK    11393  3104
    32    DBYTES
  6416    DDENOT     1847  1847  3281  4756
    51    DFALSE
  6415    DIDENT     1846  1846  2326  2346  2616  2858  3176  3348  3465
  6414    DPRIOR     1845  1845  2270  4527
    37    DSTRNG
777777    ERROR        81    81   470   527   547   553   574   600   604  1345  1367  1400  1499  1501  1513  1585
                           1888  2097  2098  2100  2105  2123  2125  2141  2620  2913  3216  3221  3230  3237  3387
                           3634  3643  3695  3708  3714  3809  4163  4215  4221  4648  4753  4757  4848  4888  4981
                           4992  5046  5085  5114  5595  5627  5966  5967  5968  5969  5970  5971  6023  6270  6344
                           6356  6378  6431  6491  6529  6799  6847  6902  6912  7209  7255  7285  7351  7433  7439
                           7566  7709  7730  7750  8061  8301  8376  8536  8584  8831  8905  8910  8914  8923  8931
                           8937  8941  9282  9285  9487  9515  9527  9583  9636  9717  9913 10075 10087 10092 10096
                          10105 10109 10303 10333 10353 10374 10412 10418 10518
  2525    F1          501   495   501
  2527    F2          503   491   503
  2530    F3          504   500   504
  2564    F4          523   520   523
  2465    FAULT       486   454   486
   330    FREG        107   107   635   638   645   648
  2306    FT          451   451
   412    FTAB        145   145   503
  4063    FTRAP       673   620   626   636   646   653   657   664   670   673
    10    GEBORT
    22    GECALL
    27    GECHEK
    16    GEENDC
     3    GEFADD
    12    GEFCON
    13    GEFILS
     7    GEFINI
    40    GEFRCE
    41    GEFSYE
    35    GEIDSE
    45    GEINFO
     1    GEINOS
     6    GELAPS
    37    GELBAR
    33    GELOOP
    11    GEMORE
    25    GEMREL
    43    GENEWS
    42    GEPRIO
    17    GERELC
     4    GERELS
    15    GERETS
     2    GEROAD
    31    GEROLL
```

```
OCTAL     SYMBOL     REFERENCES BY ALTER NO,

    30    GEROUT
    24    GERSTR
    23    GESAVE
    14    GESETS
     5    GESNAP
    44    GESNUM
    20    GESPEC
    26    GESYOT
    21    GETIME
    32    GEUSER
    34    GEWAKE
  2462    IDLE         481    65   481   483
  2766    IN           557   542   543   557
  3260    IOF          610   545   551   596   601   610
   200    ITCB          65    65   478
   221    ,   IC        70    70   402   406   409   423   440   446   457   464   470   486   521   522   547   553
                             555   558   564   565   566   572   573   574   576   604   621   622   627   628   637
                             638   647   648   654   655   658   659   665   666   671   672   680   937  1543
  2316    ,009,        456   456
    47    ,  NIC        74    74   522   565   573   622   628   638   648   655   659   666   672
   220    ,  REG        69    69   404   442   445   470   486   522   547   553   555   558   565   566   573   574
                             576   604   616   622   623   628   629   638   639   648   649   655   656   659   660
                             666   667   672   680  1544
  2323    ,010,        456   456
  2370    ,011,        470   470
  2405    ,012,        470   470
  2353    ,013,        470   470
  2360    ,014,        470   470
  2375    ,015,        470   470
  2402    ,016,        470   470
  2414    ,017,        476   476
  2421    ,018,        476   476
  2426    ,019,        477   477
  2433    ,020,        477   477
  2440    ,021,        478   478
  2445    ,022,        478   478
  2452    ,023,        479   479
  2457    ,024,        479   479
  2506    ,025,        486   486
  2507    ,026,        486   486
  2476    ,027,        486   486
  2503    ,028,        486   486
  2576    ,029,        527   527
  2612    ,030,        527   527
  2603    ,031,        527   527
  2610    ,032,        527   527
  2657    ,033,        547   547
  2674    ,034,        547   547
  2642    ,035,        547   547
  2647    ,036,        547   547
```

```
OCTAL    SYMBOL    REFERENCES BY ALTER NO,

  2664   ,037,         547   547
  2671   ,038,         547   547
  2725   ,039,         553   553
  2742   ,040,         553   553
  2710   ,041,         553   553
  2715   ,042,         553   553
  2732   ,043,         553   553
  2737   ,044,         553   553
  2764   ,045,         555   555
  2765   ,046,         555   555
  2754   ,047,         555   555
  2761   ,048,         555   555
  3010   ,049,         558   558
  3011   ,050,         558   558
  3000   ,051,         558   558
  3005   ,052,         558   558
  3052   ,053,         566   566
  3053   ,054,         566   566
  3042   ,055,         566   566
  3047   ,056,         566   566
  3117   ,057,         574   574
  3134   ,058,         574   574
  3102   ,059,         574   574
  3107   ,060,         574   574
  3124   ,061,         574   574
  3131   ,062,         574   574
  3156   ,063,         576   576
  3157   ,064,         576   576
  3146   ,065,         576   576
  3153   ,066,         576   576
  3236   ,067,         604   604
  3253   ,068,         604   604
  3221   ,069,         604   604
  3226   ,070,         604   604
  3243   ,071,         604   604
  3250   ,072,         604   604
  2270   , EXIT        434   430   432   434   457   464   470   480   486   522   529   547   553   555   558   565
                             566   573   574   576   604   622   628   638   648   655   659   666   672   680  1547
   223   , LINK         72    72   417   431   433   436   470   486   487   496   501   522   523   547   553   555
                             558   565   566   573   574   576   604   622   628   638   648   655   659   666   672
                             680
    50   , NREG         75    75   522   565   573   622   628   638   648   655   659   666   672
   222   , STAT         71    71   438   488   498   502   598   684  1031  8902  8912  8921  8930  8936  8940 10072
                           10084 10104 10108
   224   , TASK         73    73   414   420   422   433   434   444   456   470   476   477   478   479   527   547
                             553   574   604   930
  2164   , TRAP        402    24    59    60    61    62    63    64    65    76   402
   225   , XEDT         76    76   427
    36   ,EMM
777777   ,ERROR         68    68   435
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

  226     ,FFLAG       77    77   405   419   429
 2203     .,001,      414   414
 2210     .,002,      414   414
 2222     ,,003,      420   420
 2227     .,004,      420   420
 2235     .,005,      422   422
 2242     .,006,      422   422
 2261     ,,007,      433   433
 2266     ,,008,      433   433
    0     ,N355F
    1     ,N635F
    0     ,N655F
    1     ,NIOCF
    0     ,NIOMF
 2172     ,TRAP1      408   408   425   428
 2232     ,TRAP2      422   411   422
 2244     ,TRAP3      423   415   421   423
 2213     ,TRAP4      416   413   416
   17     L     D
   16     L     S           10414 10415 10416
30072     L    LC     10461 10446 10461
27242     L    RD     10060 10060 10288
30046     L    SR     10443 10441 10443
30042     L    TP     10439 10426 10434 10436 10439
30052     L    UC     10449 10443 10449
30056     L    US     10452 10445 10449 10452
30132     L   CEL     10475 10111 10129 10131 10154 10175 10177 10184 10192 10194 10475
30130     L   FRN     10473 10094 10102 10106 10473
30102     L   MFD     10467 10458 10464 10467
27244     L   RD1     10062 10062 10080 10091
27247     L   RD2     10065 10065 10069
27267     L   RD3     10081 10064 10081
27300     L   RD4     10090 10077 10090
27303     L   RD5     10093 10089 10093
30150     L   REG     10483 10483 10496 10537 10538
30052     L   SRE     10447 10441 10447
30045     L   SRT     10442 10061 10062 10090 10442
30236     L   SYM     10542  8957 10132 10137 10273 10286 10349 10542
30173     L AINC      10507 10505 10507 10526
30160     L ALOC      10496 10099 10114 10138 10191 10237 10252 10294 10315 10347 10496
27613     L BIND      10289 10275 10289
27341     L ESYM      10123 10123 10198
27456     L ETYP      10199 10124 10199
30131     L FREE      10474  8949 10100 10115 10230 10255 10269 10289 10291 10301 10312 10327 10343 10345 10405 10413
                            10414 10474
30126     L NAME      10472 10081 10284 10472
30240     L PROG      10544  8963 10110 10112 10331 10354 10367 10544
27574     L SEA1      10274 10274 10281
27604     L SEA2      10282 10279 10282
30044     L SRTI      10441 10060 10441
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 30134    L SYMP    10476 10272 10287 10476
 30141    L TEMP    10481 10305 10307 10308 10481 10497 10499 10536
 30241    L TEND    10545  8966 10506 10530 10545
 30041    L TERM    10438 10438
 30235    L TYPE    10541  8954 10205 10236 10251 10297 10313 10316 10541
 30234    L USER    10540  8947  8950  8951 10098 10118 10119 10270 10271 10292 10314 10346 10348 10540
 30207    LALOC1    10519 10514 10519 10525
 30216    LALOC2    10526 10520 10526
 30220    LALOC3    10528 10528 10531 10532
 30225    LALOC4    10533 10503 10533
 27624    LBIND0    10298 10298 10326
 27632    LBIND1    10304 18304 10309
 27662    LBIND2    10327 10299 10327
 27671    LBIND3    10334 10334 10341
 27711    LBIND4    10350 10350 10383
 27727    LBIND5    10364 10360 10362 10364 10380
 27751    LBIND6    10381 10365 10381
 30135    LBINDI    10477 10295 10477
 27710    LBINDR    10349 10349 10477
 30137    LBLNKS    10479 10358 10479
 30237    LCHAIN    10543  8960 10151 10171 10174 10176 10186 10187 10189 10277 10351 10543
   411    LEN         143   104   110   143   562   570   633   643
 30140    LENTRY    10480 10363 10406 10411 10480
 27353    LESYM1    10133 10133 10169
 27403    LESYM2    10156 10134 10156
 27421    LESYM3    10170 10167 10170
 27433    LESYM4    10180 10173 10180 10182
 27444    LESYM5    10189 10155 10179 10189
 27457    LETYP1    10200 18200 10233 10267
 27465    LETYP2    10206 10206 10221
 27506    LETYP3    10222 10218 10222
 27510    LETYP4    10224 10224 10226
 27522    LETYP5    10234 10204 10207 10234
 27566    LETYP6    10268 10201 10268
   410    LOC         142   101   102   107   108   142   560   568   631   641
 30136    LPROG1    10478 10328 10478
   300    LREG         92    92   522
 27324    LSTART    10110  8967 10110
 30062    LXSYSC    10455 10444 10455
 17054    M   FS
 17051    M   MA
    33    M   MS    11110  6890  7608  7808  7965
500103    M  CAT       15    15   666
     7    M  INT    11101  3256  4210  4445  6044  7002  7830  7835  8729  8731
    31    M  LBL    11109  2336  2861  5013  5116  7982
    71    M  M20
    75    M  M21
   101    M  M22
   106    M  M23
   112    M  M24
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO.

   117    M  M25
   124    M  M26
   130    M  M27
   134    M  M28
   140    M  M29
   144    M  M30
   151    M  M31
   156    M  M32
   163    M  M33
   170    M  M34
   174    M  M35
   200    M  M36
   204    M  M37
   210    M  M38
   214    M  M39
   221    M  M40
   226    M  M41
   233    M  M42
   240    M  M43
   245    M  M44
   252    M  M45
   257    M  M46
   263    M  M47
   267    M  M48
   273    M  M49
   277    M  M50
   304    M  M51
   311    M  M52
   316    M  M53
   323    M  M54
   330    M  M55
   334    M  M56
   340    M  M57
   344    M  M58
   351    M  M59
   356    M  M60
   363    M  M61
   370    M  M62
   374    M  M63
   400    M  M64
   405    M  M65
   412    M  M66
   416    M  M67
   422    M  M68
   427    M  M69
   434    M  M70
   441    M  M71
   445    M  M72
   451    M  M73
   456    M  M74
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

   463    M  M75
   470    M  M76
   474    M  M77
   500    M  M78
   504    M  M79
   511    M  M80
   516    M  M81
   523    M  M82
   530    M  M83
   534    M  M84
   540    M  M85
   544    M  M86
   551    M  M87
   556    M  M88
   563    M  M89
   567    M  M90
   573    M  M91
   576    M  M92
   602    M  M93
   606    M  M94
   612    M  M95
   616    M  M96
   622    M  M97
 17026    M  MAX
 16773    M  MMI              2196  2592  2599  2805
     2    M  MSL       11111  8104
    37    M  PTR       11113  2881  6586  6916  6922  7010  7016  7648  7800  7838  8725
 16762    M  REF              1292  2111  2140  2313  2594  2700  2899  3808  3831  4151  4579  5027  5040  5063  5154
                              5268  5290  5528  7254  7432
 16770    M  ROW              1296  2415  2702  2761  2907  2920  3985  4160  4170  5145  5150  5187  5325  5454  5475
                              5493  5585  6886  7690  7727
    14    M BITS       11103  6048
     3    M BOOL       11099  3599  3650  4475  4687  6040  7589  8332
     5    M CHAR       11100  3272  6042
 17023    M CONE
 17004    M DEPR              5580  5611
    20    M LINT       11105  6052
 17020    M MREF
500006    M MREQ          13    13  1583  9911 10389 10516
    35    M MSCW       11112  7731  8070  8202
500101    M OPEN           7     7   622
 16754    M PRIM              1295  5185  5584  9048  9058  9066
 17001    M PROC              1293  2213  2215  2710  2901  3873  4647  4847  5023  5038  5463  5530  5550  5579  8484
    42    M QUAD       11114  6966  6972  6985  6991
500133    M READ           9     9   565   638
    11    M REAL       11102  3259  6046  6844
 16776    M ROWE              1297  2406  2704  2763  2909  2924  3980  4162  4172  5147  5152  5189  5327  5456  5477
                              5495  6888  6934  7692  7729  7817  7845
500113    M SETP          11    11   655
500000    M TERM          12    12   891
```

```
01557 01  05-26-72   18,180      ALGOL68                                                                       PAGE    315

OCTAL    SYMBOL    REFERENCES BY ALTER NO,

     1   M VOID    11098  2014  2188  2310  2316  3815  3931  4044  4051  5179  6829
 17012   M XFER             2695  2796  2821  8838
    16   MBYTES    11104  6050
500105   MCLOSE        8     8   628
    56   MCOMPL
 16765   MDEREF             5581  5613
 17015   MEMPTY             2590  2817  2911  8840
500012   MJTIME       17    17   795
    25   MLBITS    11107  6056
    27   MLBYTS    11108  6058
    62   MLCOMP
    22   MLREAL    11106  6054
  5234   MODE4,     1274  1270  1274
500100   MOPENS       14    14   659
 17037   MOTBLE
500005   MPAUSE        6     6   482
    50   MPROCV    11115  3101
    66   MRBOOL
 16757   MSTRCT             1294  2064  2708  2770  2903  3867  4577  4584  5459  6866  7284  7646  7664  7682  7688
    53   MSTRNG    11116  3276
  2305   MTASK       450    60   450   457
    60   MTCB         60    60   416   418   420
500107   MTRUNC       16    16   672
 17007   MUNION             1298  2218  2706  2905  3875  5191  5197  5216  5220  5248  5406  5414  5583
500134   MWRITE       10    10   522   573   648
    40   NULL         59    59    74    75
 17034   0   FS             4955  5705
 16575   0   IS     5783  3613  5783
 16641   0   LL     5786  3921  4791  5786
 17031   0   MA             5666  5678
 16476   0   OP     5780  4930  5780
 16545   0   TF     5782  3556  4357  4484  5782
 16735   0  BUS     5789  4153  4194  5789
 16564   0  ETC     5783  4612  5783
 16724   0  FIX     5789  4126  5789
 16526   0  HIP     5781  5305  5589  5781
 16504   0  LBL     5780  3469  3544  3574  3589  3594  3947  4330  4414  4816  5676  5780
 16644   0  LLE     5786  3937  4815  5786
 16716   0  LWB     5789  4122  5789
 17026   0  MAX             3576  4416
 16773   0  MMI
 16572   0  NIL     5783  4675  5294  5588  5783
 16501   0  OPE     5780  4922  5780
 16762   0  REF
 16770   0  ROW
 16732   0  SUB     5789  4134  5789
 16721   0  UPB     5789  4124  5789
 16633   0 ASGN     5785  3491  4719  5785
 16550   0 CASE     5782  4459  5782
 17023   0 CONE             3580  4418
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 16542    0 CONF     5782  3552  4353  5782
 16751    0 DBUS     5791  3928  5791
 16647    0 DELV     5786  4050  5664  5786
 17004    0 DEPR
 16512    0 DISP     5780  5723  5780
 16674    0 DLEN     5788  3943  5788
 16746    0 DSUB     5791  3923  5791
 16671    0 EPDE     5787  4818  5787
 16660    0 EPDN     5787  3915  4775  5787
 16666    0 EPDV     5787  4817  5787
 16727    0 FLEX     5789  4128  5789
 16523    0 GOTO     5781  5601  5781
 16537    0 HGEN     5782  4094  4104  5782
 16600    0 ISNT     5783  3611  5783
 16507    0 JUMP     5780  3536  3570  3585  3909  4366  4396  4410  4422  4464  4468  4771  5662  5780
 16534    0 LGEN     5782  4097  4109  5782
 17020    0 MREF           3538  3558  4359  4368  4398
 16611    0 MSCW     5784  4638  5784
 16476    0 OTBL     5779  1498  5779
 16754    0 PRIM
 17001    0 PROC
 16743    0 RBUS     5790  4153  5790
 16663    0 RETN     5787  3930  4814  5787
 16776    0 ROWE
 16567    0 SKIP     5783  4670  5282  5587  5783
 16603    0 TRUE     5784  3566  4681  5784
 16702    0 VLWB     5788  4114  5788  8575
 16531    0 VOID     5782  5275  5586  5782
 16705    0 VUPB     5788  4116  5788  8579
 17012    0 XFER
 16636    OASGNE     5785  3519  3525  5785
 16740    OBOUND     5789  4322  5789
 16553    OCASGN     5782  3564  4362  5782
 16630    ODENOT     5785  4746  5785
 16765    ODEREF
 16515    OEDISP     5780  5726  5780
 17015    OEMPTY
 16520    OENTER     5780  4653  4937  5780
 16655    OERNGE     5787  4026  5690  5787
 16606    OFALSE     5784  3591  4679  5784
 16622    OFORMP     5785  4724  5785
 16625    OIDENT     5785  4732  5785
 16617    OIDNTE     5784  4714  5784
 16614    OIDNTY     5784  4697  5784
 17037    OOTBLE           1500
  3662    OPEN        616   616  8901 10071 10083
  4007    OPENS       656   656  8911
   340    OREG        113   113   617   618   619   622   625   628
 16561    ORSLCT     5783  4586  5783
 16556    OSELCT     5783  4586  4591  5783
```

```
OCTAL     SYMBOL     REFERENCES BY ALTER NO,

  350     OSREG        119   119   659
16652     OSRNGE      5787  4022  5687  5787
16757     OSTRCT
17007     OUNION
 3257     OUT1         609   577   586   588   593   594   609
 3256     OUT          608   549   578   582   584   587   595   608
16713     OVEPTY      5788  4120  4205  5788  8565
16710     OVNLWB      5788  4118  5788  8583
16677     OVSBCT      5788  4112  5788  8570
 5532     P    T      1462  1438  1441  1450  1454  1462
 5256     P   MT      1292  1286  1287  1292
 5502     P   P1      1438  1438  1461
 5507     P   P2      1443  1443  1447
 5336     P  BUS      1341  1328  1341
 5533     P  DEF      1463  1463  1534
 5255     P  MOD      1291  1259  1272  1283  1291
 5266     P  MTE      1300  1287  1300
 5625     P  OCT      1521  1515  1521
 5305     P  REF      1316  1292  1316
 5312     P  ROW      1321  1296  1321
 5335     P  SUB      1340  1336  1340
 5433     P  UN1      1400  1358  1400
 5434     P  UN2      1401  1401  1416
 5454     P  UN3      1417  1389  1412  1417
 5457     P  UNT      1420  1396  1420
 5640     P CDEF      1532  1517  1532
 5635     P CMOD      1529  1516  1529
 5536     P DEF1      1466  1466  1482
 5537     P DEF2      1467  1467  1479
 5550     P DEF3      1476  1472  1474  1476
 5552     P DEF4      1478  1470  1478
 5554     P DEF5      1480  1468  1480
 5563     P DEFE      1487  1484  1487
 5562     P DEFT      1486  1464  1469  1486
 5551     P DEFX      1477  1463  1477  1485
 5266     P MERR      1301  1299  1301
 5211     P MODE      1255  1255  1374  1407  1435  1451  1530  1540
 5627     P OCT1      1523  1523  1527
 5574     P PCT1      1496  1494  1496
 5622     P PCTE      1518  1518  1528  1531  1541
 5571     P PCTL      1493  1493  1520
 5466     P PRI1      1426  1426  1430
 5471     P PRI2      1429  1427  1429
 5272     P PRIM      1305  1295  1305
 5473     P PRIT      1431  1425  1426  1431
 5337     P PROC      1342  1293  1342
 5311     P REFT      1320  1317  1320
 5317     P ROWE      1326  1297  1326
 5334     P ROWF      1339  1329  1332  1337  1339
 5325     P ROWI      1332  1321  1326  1332
```

```
OCTAL    SYMBOL    REFERENCES BY ALTER NO,

  5463   P RPAR     1423  1418  1423
  5653   P STOP     1543  1458  1495  1543  1699 10438
  5372   P STR1     1368  1368  1393
  5415   P STR2     1387  1383  1385  1387
  5424   P STRT     1394  1363  1394
  5645   PCDEF1     1537  1533  1537
  5652   PCODEP     1542  1492  1493  1542
  5461   PCOMMA     1421  1323  1392  1415  1421
  5271   PMERRT     1304  1302  1304
  5213   PMODE1     1257  1257  1319  1325  1331  1352  1360
  5217   PMODE2     1261  1261  1265
  5220   PMODE3     1262  1262  1275
  5225   PMODE4     1267  1262  1267
  5236   PMODE5     1276  1273  1276
  5243   PMODE6     1281  1277  1279  1281
  5245   PMODE7     1283  1266  1283
  5564   PPCTBL     1488  1488  5766
  5474   PPMODE     1432  1432
  5501   PPMTBL     1437  1437
  5304   PPRIMT     1315  1311  1313  1315
  5404   PPRINT     1424  1280  1301  1312  1316  1322  1327  1335  1347  1349  1353  1362  1378  1386  1391  1395
                          1414  1417  1424  1475  1483  1535
  5352   PPROC1     1353  1346  1353
  5362   PPROCT     1361  1348  1354  1361
  5333   PROWIX     1338  1334  1338
  5462   PSPACE     1422  1350  1379  1422  1536
  5660   PSTOPM     1548  1546  1548
  5364   PSTRCT     1362  1294  1362
  5426   PUNION     1395  1298  1395
 26545   R    A     9742  9539  9610  9611  9637  9638  9640  9645  9648  9654  9661  9662  9663  9664  9742
 26546   R    B     9743  9550  9564  9566  9568  9570  9572  9573  9680  9685  9686  9743
    17   R    D           9139 10035
 26226   R    G     9535  9483  9509  9535
    16   R    S           9136  9478  9479  9481  9485  9486  9504  9514 10034
 26763   R   A1     9884  9757  9884
 27003   R   A2     9900  9898  9900
 27021   R   A3     9914  9906  9909  9914
 27034   R   A4     9925  9925  9933
 27045   R   A5     9934  9926  9934
 27065   R   A6     9950  9948  9950
 27077   R   A7     9959  9959  9963
 27104   R   A8     9964  9956  9964
 27177   R   AR    10023  9941  9945 10023
 27176   R   AT    10022  9887  9891  9901  9937  9940  9944  9951  9954 10022
 26556   R   C1     9751  9751  9804  9861  9869
 26562   R   C2     9755  9755  9765  9766
 26573   R   C3     9764  9754  9764
 26602   R   C4     9771  9750  9771  9778  9779
 26610   R   C5     9777  9770  9777
 26665   R   C6     9822  9822  9829
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 26730    R    C7    9857  9853  9857
 26733    R    C8    9860  9856  9860
 26737    R    C9    9864  9864  9868
 26745    R    CA    9870  9796  9818  9840  9843  9847  9848  9870  9885  9924  9938
 26746    R    CB    9871  9810  9815  9816  9831  9839  9871
 26747    R    CC    9872  9788  9795  9805  9828  9835  9842  9872
 26753    R    CF    9876  9746  9793  9803  9876
 26750    R    CN    9873  9553  9747  9820  9851  9854  9873  9897  9899  9947  9949  9952  9964  9973  9974  9975
                           9976 10005
 26751    R    CR    9874  9845  9852  9858  9874
 26752    R    CT    9875  9857  9859  9875
 25610    R    DE    9270  9257  9264  9270
 25606    R    DN    9269  9170  9210  9215  9217  9243  9254  9256  9263  9266  9269
 27212    R    G1   10034  9706 10034
 26540    R    GD    9737  9125  9128  9138  9737 10035 10427
 26537    R    GP    9736  9482  9508  9736  9738
 27214    R    GX   10036  9535 10036
 27203    R    HP   10027  9551  9569  9802  9834  9838  9925  9936  9939  9942  9953  9965  9968  9979  9986  9998
                          10006 10009 10027 10034
 26231    R    L1    9538  9538 10021
 26243    R    L2    9548  9548  9641
 26334    R    L3    9605  9605
 26374    R    L6    9637  9579  9622  9634  9637  9679
 26400    R    L7    9641  9641
 26401    R    L8    9642  9596  9604  9607  9642  9660  9667  9671  9694  9698
 26402    R    L9    9643  9643
 26536    R    LF    9735  9536  9548  9621  9690  9705  9735 10020
 26437    R    LX    9672  9672  9689
 27120    R    S1    9976  9976
 27125    R    S2    9981  9981 10004
 27134    R    S3    9988  9988 10003
 27140    R    S4    9992  9992  9999
 27150    R    S5   10000  9993 10000
 27155    R    S6   10005  9984 10005
 27163    R    S7   10011 10011 10015
 27170    R    S8   10016 10012 10016 10019
 27205    R    SD   10029  9978  9981  9985  9987  9990  9996  9997 10029
 27206    R    SE   10030  9980 10002 10030
 25741    R   AFS    9354  6639  9354  9381
 26754    R   CAQ    9878  9753  9783  9878
 26762    R   CBA    9883  9841  9844  9846  9883
 26756    R   CE1    9879  9767  9811  9879
 26760    R   CE2    9881  9752  9780  9789  9881
 26550    R   C,5    9745  9708  9745
 26757    R   CP1    9880  9768  9806  9880
 26761    R   CP2    9882  9751  9781  9784  9882
 25567    R   DIG    9254  9172  9224  9244  9245  9254
 25422    R   INT    9153  9153
 26404    R   L10    9645  9645
 26406    R   L11    9647  9647
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 26424    R  L12     9661  9651  9661
 26430    R  L13     9665  9665
 26500    R  LA1     9705  9646  9705
 26503    R  LA2     9708  9708  9734
 26524    R  LA3     9725  9725  9729  9732
 26534    R  LA4     9733  9713  9727  9733
 26133    R  LOC     9476  9384  9420  9466  9476
 26466    R  LX1     9695  9691  9695
 26470    R  LX2     9697  9697  9704
 25611    R  OUT     9271  9155  9161  9166  9176  9179  9180  9182  9192  9214  9220  9222  9228  9230  9238  9260
                           9271
 25562    R  RE1     9249  9194  9201  9232  9249
 25563    R  RE2     9250  9195  9204  9231  9250
 27211    R  REQ    10033  9315  9382  9410  9464  9474  9478  9481  9485  9497  9501  9506  9511  9525  9892  9895
                          10033 10408
 25400    R  RET     9135  8273  9135
 26020    R  RGN     9401  9399  9401
 26213    R  ZER     9524  9480  9488  9505  9516  9524 10410
 25627    R ACH1     9280  9280  9289
 25646    R ACH2     9295  9276  9295
 25661    R ACH3     9306  9306  9313
 25715    R ACH4     9334  9334  9346
 25735    R ACHD     9350  9297  9299  9327  9333  9350
 25733    R ACHP     9348  9295  9316  9322  9348
 25737    R ACHT     9352  9302  9314  9320  9329  9330  9331  9352
 25732    R ACHX     9347  9274  9294  9347
 25760    R AFS1     9369  9369
 26012    R AFSD     9395  9357  9395
 26010    R AFSP     9393  9355  9383  9386  9393
 26014    R AFST     9397  9366  9367  9368  9397
 26007    R AFSX     9392  9354  9392
 27201    R BITL    10025  9919  9921 10025 10399 10401
 27200    R BITT    10024  9540  9586  9720  9748  9785  9807  9922 10024 10402
 25413    R BOOL     9146  9146
 26640    R C5,2     9801  9799  9801
 26644    R C5,5     9805  9794  9805
 26706    R C6,4     9839  9832  9836  9839
 25414    R CHAR     9147  9147
 25406    R CRLF     9141  9141
 25375    R ENT1     9132  9124  9132
 25377    R ENT2     9134  9131  9134
 26162    R HEAP     9499  9317  9422  9499
 26157    R HGEN     9496  7603  9496
 25436    R INT1     9165  9159  9165
 25441    R INT2     9168  9164  9168
 25445    R INT3     9172  9172  9174
 25455    R INT4     9180  9180  9183
 25461    R INT5     9184  9181  9184
 25463    R INTX     9186  9153  9186
 26237    R L1,5     9544  9544  9547
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 26254    R L2,1     9557  9557  9561
 26261    R L2,2     9562  9559  9562
 26274    R L2,3     9573  9567  9571  9573
 26275    R L2,4     9574  9549  9574  9618  9644
 26332    R L2,5     9603  9592  9603
 26321    R L2,7     9594  9594  9599  9602
 26352    R L5,3     9619  9609  9619
 26360    R L5,7     9625  9625  9633
 26372    R L5,8     9635  9620  9635
 26130    R LGEN     9473  7600  9473
 26152    R LOCD     9491  9491
 26151    R LOCP     9490  9490  9492
 26153    R LOCT     9492  9492
 27207    R MTOP    10031  9546  9756  9905  9908  9910  9914  9920 10031 10393 10394 10400
 25613    R OUTB     9273  9185  9247  9272  9273
 25612    R OUTI     9272  9154  9175  9191  9272
 25467    R REAL     9190  9190
 26033    R RGN1     9412  9412  9415
 26045    R RGN2     9422  9419  9422
 26046    R RGN3     9423  9421  9423
 26057    R RGN4     9432  9427  9432
 26070    R RGN5     9441  9441  9450
 26111    R RGNA     9458  9403  9424  9428  9432  9458
 26105    R RGND     9454  9404  9437  9454
 26104    R RGNF     9453  9398  9400  9418  9453
 26110    R RGNP     9457  9402  9457  9459
 26103    R RGNS     9452  9416  9426  9452
 26107    R RGNT     9456  9456
 26102    R RGNX     9451  9401  9431  9451
 25564    R RTEN     9252  9203  9252

 27204    R SMAX    10028  9479  9486  9500  9502  9504  9507  9510  9512  9514  9798  9800  9830  9934  9935 10008
                          10028 10404 10416
 25464    R TLVN     9188  9169  9188
 26221    R ZER1     9530  9530  9533
 26225    R ZERX     9534  9476  9499  9534 10409
 25621    RACHEK     9274  6853  6907  7168  9274
 25734    RACHEL     9349  9296  9332  9349
 25645    RACHKX     9294  9294
 25740    RACHT1     9353  9304  9309  9310  9353
 25736    RACHTP     9351  9298  9319  9351
 26011    RAFSEL     9394  9356  9394
 26013    RAFSTP     9396  9358  9385  9396
 25421    RCHART     9152  9148  9150  9152
 25420    RCHARX     9151  9147  9151
 25412    RCRLFC     9145  9143  9145
 25411    RCRLFX     9144  9141  9144
  3721    READ        629   629  1027 10103
  3255    RELS        607   555   602   604   607
     0    RELZER       19    19   613
 25365    RENTER     9124  8246  9124 10437
```

```
01557 01  05-26-72   18,180       ALGOL68                                                                        PAGE     322

OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 26541    RGMARK     9738  9538  9738
 26206    RHEAPD     9519  9519
 26205    RHEAPP     9518  9518  9520
 26207    RHEAPT     9520  9520
 26204    RHGENP     9517  9496  9498  9517
 26407    RL11,5     9648  9575  9577  9581  9648
 26150    RLGENP     9489  9473  9475  9489
 27210    RMBASE    10032  9582  9716  9792  9814  9886  9902  9915 10032 10329 10395 10425 10433
 26116    RPLGEN     9463  8216  9463
 26122    RPLGNP     9467  9463  9465  9467
 25477    RREAL1     9198  9198  9202
 25504    RREAL2     9205  9199  9203  9206  9209
 25513    RREAL3     9210  9197  9210
 25524    RREAL4     9219  9212  9219
 25526    RREAL5     9221  9218  9221
 25531    RREAL6     9224  9224  9226
 25545    RREAL7     9236  9233  9236
 25547    RREAL8     9238  9235  9238
 25561    RREALX     9248  9190  9248
   320    RREG        100   100   565
 26112    RRGNTG     9459  9417  9459
 26106    RRGNTP     9455  9405  9423  9455
 26017    RRHGEN     9400  7640  9400
 26015    RRLGEN     9398  7639  9398
 27202    RSHDIV    10026  9801  9833  9837  9966 10007 10026
 26547    RSTMSK     9744  9642  9701  9744
    60    S   AT
   100    S   BY
   116    S   CT
   106    S   DO
   224    S   EQ
   232    S   GE
   234    S   GT
   154    S   IM
   122    S   IS
   236    S   LE
   164    S   LN
   230    S   LT
   226    S   NE
    64    S   OF    11316  3338  3437
    44    S   OP
   220    S   OR
   156    S   PI
   132    S   RE
   102    S   TO
   216    S  AND
    62    S  BAR    11315  3414
    56    S  BUS
   166    S  COS
   162    S  EXP
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

   74     S  FOR
  140     S  INT
   42     S  LOC
  130     S  NIL
  222     S  NOT
   12     S  PER
   24     S  REF
  202     S  RND
  172     S  SIN
   22     S  STR
   54     S  SUB
  176     S  TAN
  170     S ACOS
   14     S ASGN
  174     S ASIN
  200     S ATAN
  110     S BARF      11326  3419
  150     S BITS
  144     S BOOL
  146     S CHAR
  120     S CTAB
   30     S EITH
    0     S ERM1      11290  3168
    2     S ERM2      11291  3173
    4     S ERM3      11292  3178
    6     S ERM4      11293  3181
   10     S ERM5      11294  3432
   26     S FLEX
   76     S FROM
   70     S HEAP
  246     S INTP
  124     S ISNT
   20     S LONG      11298  1155
   46     S LPAR      11309  3099  3365
  204     S LRND
   36     S MODE
  210     S MSGN
  240     S OVER
   32     S PROC
  206     S PSGN
  142     S REAL
   50     S RPAR
   66     S SEMI
  126     S SKIP
  160     S SQRT
  112     S TRUE
  104     S WHLE
   16     S                  9135  9139  9477  9484
  212     SASTER
  136     SBLANK
```

```
01557 01  05-26-72   18,180      ALGOL68                                                                   PAGE    324

OCTAL     SYMBOL    REFERENCES BY ALTER NO,

   242    SBOCLP
   152    SBYTES
   244    SCHARP
    16    SCOLON    11297  3455
    52    SCOMMA
  3767    SETP        649    649   1649   2849
400250    SETSET
   114    SFALSE
   246    SIB          89     21     87     89
   244    SIF          87     87    461    678
    72    SLIBRY
  2337    SPEC1       465    462    465
  2330    SPEC        460     61    460    464    467    469    471
    40    SPRIOR
   250    SREALP
   400    SREG        134    134    651    652    655
   214    SSLASH
   154    SSTRNG
  2407    START       474     20    474
   100    STCB         61     61    414
  2406    STRT1       473    473    474
   232    STTYF        84     84    468    470    680
    34    SUNION
  3261    SWAPC       611    592    611
 36071    T     B
 37404    T     C
 36072    T     D
 40067    T     E
 37406    T     G
 37265    T     I
 37405    T     L
 35243    T     M
 37454    T     N
 37011    T     P
 37012    T     S
 37407    T     T
 37455    T     Z
 36071    T    BE
 37404    T    CE
 37010    T    DE
 40067    T    EE
 37406    T    GE
 37403    T    IE
 37405    T    LE
 36070    T    ME
 27235    T    MS     10054   9348   9393   9457 10054
 37454    T    NE
 37011    T    PE
 40066    T    SD
 37264    T    SE
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

 35241    T    SK
 37453    T    TE
 35237    T    WK
 40065    T    ZE
 35220    T   DEF    11839  1267  1478  1538  1869  2258  2268  2278  2286  2291  2319  2344  2363  2376  2379  2538
                            2628  2855  3279  3317  3327  3330  3352  4055  4503  4513  4691  4700  4703  4738  4754
                            4861  4940  4950  5978  6781  7504  7543  7555  7559  7979  8093  8878  9000  9006  9025
 35233    T   END    11850  1573  1597
 35226    T   GEN    11845  5957  6130  7024  7099  7352  7359  7377  7378  7389  7393  7416  7864  7896  8927  8932
                            8946  9037  9052  9089  9093  9106  9113
 35225    T   LBL    11844  5920  5946  5947  5950  6952  6955  7349  7357  7852  7854
 35230    T   NUM
 27217    T   OCT    10040  2974  2975  9469  9606 10040
  5765    T   OVF     1628  1617  1627  1628
 27215    T   PTR    10038  2938  2947  2976  2977  6690  8132  9468  9470  9471  9635 10038 10046 10047 10049 10050
                           10051 10052 10054 10055 10057
  6010    T   REG     1641  1562  1605  1606  1641
 27221    T   ROW    10042  2935  9619  9666 10042
 40066    T   SDE
     3    T   SET
 35242    T   SKE
 35240    T   WKE
 35231    T   ZZZ    11848  6063
  5677    T AINC      1574  1572  1574  1593
  5663    T ALOC      1562  1122  1171  1227  1562  1633  2048  2054  2110  2195  2204  2229  2259  2269  2279  2320
                            2345  2385  2410  2437  2932  2962  3015  3047  3280  3668  3800  3840  3952  4310  4708
                            5921  6131  6669  6685  6707  6723  6957  7856  8131  9053
 35224    T CODE     11843  1488  1496  3606  3665  3667  3669  3679  3783  3799  4035  4199  4298  4309  5600  5624
                            5938
 35223    T ITAB     11842  1134  1138  1146  1187  1198  1210  1213  1223  1226  1230  1278  1384  1473  1943  2093
                            3191  3264  9013  9073
 35216    T MODE     11837  1258  1284  1369  1377  1402  1410  1455  1460  2047  2049  2058  2109  2116  2138  2194
                            2200  2228  2249  2409  2417  2588  2597  2648  2688  2689  2694  2698  2734  2735  2738
                            2739  2748  2772  2779  2794  2830  2833  2835  2839  2842  2844  2869  2878  2885  2890
                            2898  2916  2951  2970  2992  3009  3018  3051  3053  3065  3077  3500  3503  3711  3729
                            3812  3824  4600  4603  4606  5042  5045  5065  5213  5225  5411  5434  5526  5548  6019
                            6869  6874  6881  7628  7694  8835  8846  8849  8874  9051  9054
  6020    T OVFR      1642  1613  1623  1638  1642
 27223    T PROC     10044 10044 10420
 35221    T PROG     11840  1876  2436  2439  2457  2495  2500  2501  2507  2510  2515  2527  3121  3124  3323  3376
                            3424  3892  3912  3918  3934  3940  3973  4143  4382  4436  4443  4449  4784  7754  7976
                            8788
 27240    T PTRT     10057  8387  9489  9517 10057
 35232    T SDEF     11849  4706  8883
 27222    T SKIP     10043  2942  3084  6692  6725  8140  8976  9591 10043 10044 10045 10439
  5742    T SOVF      1609  1256  1356  1371  1373  1404  1406  1609  1666  1998  2008  2288  2691  2693  2758  2775
                            2982  2987  2989  3057  3062  3111  3116  3440  3489  3543  3615  3906  4066  4265  4268
                            4559  4665  5095  5133  5164  5172  5218  5227  5277  5284  5296  5302  5307  5316  5322
                            5347  5352  5358  5400  5416  5438  5443  5449  6862  6871  6873  7026  7029  7032  7035
                            7038  7041  7045  7048  7697  7700  7703  7863  7866  7869  7872  7875  7944  7949  7954
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

                               8036  8041  8046  8282  8517  8522  8551  9036  9043
 35222    T STAB     11841  1117  1121  1132  1154  1159  1161  1167  1170  1175  1190  1202  1205  1260  1381  1465
                            1684  1694  1851  1860  2037  2090  2360  2531  2609  2808  3161  3188  3341  3357  3408
                            3442  3447  3460  3472  3477  4523  4743  4751  4831  4966  9010  9070
  6031    T TEMP      1644  1563  1565  1603  1644
 35227    T TYPE     11846  2927  2949  2960  2968  3002  3014  3019  3024  3029  3045  3050  3081  3086  6667  6671
                            6683  6701  6704  6705  6708  6720  6732  6736  6738  7414  8130  8142  8843  8853  8859
                            8861  8982  8984  8989  8991  8994  8996
 27220    T ULEN     10041  3092  9608  9650 10041
 35214    T WORK     11835  1626  1628  1634  2006  2011  2042  2069  2127  2151  2179  2224  2236  2245  2297  2306
                            2322  2756  2768  2791  2980  3026  3683  3834  3846  3877  3885  4263  4279  4539  4663
                            4821  4885  5091  5101  5515  6128  6132  6233  6287  6598  6600  6674  6678  6684  6686
                            6706  6709  6722  6724  6729  6742  6979  6998  7083  7093  7537  7951  7960  8024  8043
                            8205  8284  8291  8527  8533  8560  8598  8667
  5754    T WOVF      1619  1619  2016  2025  2029  2083  2103  2118  2211  2252  2338  2341  2422  2432  2444  2661
                            2664  2737  2998  3780  3782  3787  3790  3793  3963  4073  4078  4247  4251  4255  4257
                            4260  4293  4297  4302  4304  4307  4534  4543  4546  4548  4551  4667  5119  5122  5129
                            5232  5502  5504  5507  5511  5514  5520  5917  6579
 27216    T WPTR     10039  9460  9461  9493  9494  9521  9522  9578  9739  9740 10039
  5713    TALOC1      1586  1581  1586  1592
  5722    TALOC2      1593  1587  1593
  5724    TALOC3      1595  1595  1598  1599
  5731    TALOC4      1600  1570  1600
 35217    TBOUND     11838  2053  2056  2203  2209  2383  2394  2403  2651  3839  3861  3950  3961  3970  7706  7747
 27230    TMSCWT     10049  8219 10049
  6050    TOVFIC      1643  1611  1621  1639  1643
   370    TRREG        132   132   668   669   672
  4044    TRUNC        667   667  8935
 35215    TSTACK     11836  1616  1630  1636  5125  5229  5237  5253  5311  5330  5385  5509  6787  6788  7523  7533
                            7535  7957  8060  8063  8065  8524  8562  8596  8671
 35213    TSTART     11834   718   765
 35236    TTABLE     11855   709
  2627    TTB1         543   543   550   554   556
  2675    TTB2         549   544   549
  2743    TTB3         555   552   555
  2623    TTB          539   539  1545  1960  9142  9149  9184  9246
  2674    TTBX         548   541   546   548
   230    TTY           82    82   507   511   515   517   519   533
   234    TTYB          86    86    97   504   505
   231    TTYF          83    83   486   527   558   566   574
  2767    TTYR         558   558   683
   272    TTYSI         90    90   466
  3073    TTYTR        574   563   571   574
  3031    TTYW         566   566   597   742   789   908
  4000    W   OP      5807  4535  4887  5807
 20000    W  BAL      5803  3618  4233  4239  4244  5004  5103  5803  5805
100000    W  NIL      5801  4674  5010  5109  5801  5806
 40000    W  PAR      5802  4230  5002  5105  5802  5805
 10000    W  VAC      5804  5010  5111  5590  5804
 60000    W MULT      5805  4701  5805  5806
```

```
OCTAL      SYMBOL     REFERENCES BY ALTER NO,

200000     W SKIP      5800  4669  5010  5107  5800  5806
   310     WLOC          97    92    97
   311     WN            98    95    98
760000     WOBJCT      5806  3686  5806
   320     WREG         101   101   573

  3744     WRITE        639   639  8929
400000     WVALUE      5799  3755  4291  4682  4733  4747  5113  5621  5799  5806
 17441     Z      A
 17442     Z      B
 17443     Z      T
  4564     Z     FG     974   940   946   968   974
  1135     Z     IN     171   171   689   862   941   958   967   972
  4306     Z    BCD     808   797   802   808
  1466     Z    BNF     188   187   188   883
  4351     Z    BRE     843   204   843
  4345     Z    CON     839   203   838   839
  4064     Z    DDT     678   678   686   910   931
  4142     Z    DUM     708   200   708
  4432     Z    ERR     892   209   210   211   701   755   832   834   840   861   892   954   963
  4435     Z    FIN     895    62   705   724   739   771   786   805   842   858   880   889   894   895
  1137     Z    INB     173   171   172   173   198   692
  1136     Z    INP     172   172   688
   457     Z    OUT     168   168   691   731   744   778   780   791   799   814   827   900   902   904   981   990
  4132     Z    PAT     700   199   700
   120     Z    TCB      62    62   476
  4335     Z    TRA     831   202   831
  4400     Z    UNB     866   205   866
  4567     Z ADDR       977   728   775   851   885   977
  4322     Z BCD1       820   810   811   812   815   816   817   818   820
  4334     Z BCDC       830   809   830
  4333     Z BCDT       829   821   824   829
  4514     Z BKST       933   178   933
  1453     Z BPIT       182   181   182   849
  1500     Z BPMS       194   193   194
  4353     Z BRE1       845   845   859   865
  4363     Z BRE2       853   847   853
  4367     Z BRE3       857   852   857
  4372     Z BRE4       860   844   860
  4617     Z BREM      1001   687  1001
  4606     Z BSET       992   895   992
  1303     Z BTAB       176   176   853   856   875   876   878   888   916   994   997  1002  1004  1009  1011
  4626     Z BTLU      1008   494   846   869   914  1008
  1447     Z BXED       178   178   922
  1451     Z CRLF       180   179   180   726   773   794   897
  1507     Z CTAB       199   199   694
   455     Z DTCB       166   166   746   836   863   912   924   930
  4160     Z DUM1       722   722   745   753
  4170     Z DUM2       730   730   738
  4202     Z DUM3       740   735   740
  1475     Z ERRM       191   190   191   893
```

```
OCTAL     SYMBOL    REFERENCES BY ALTER NO,

  4430    Z EXIT      890   207   890
  4442    Z FIN1      900   900   903
  4446    Z FIN2      904   901   904
  4500    Z FIN3      930   915   930
  4522    Z GARG      940   700   704   708   712   716   754   758   763   833   843   857   866   879   940   950
                            952
  4570    Z LOOP      978   976   978   983
   461    Z OUTB      170   168   169   170   740   787   905   907
   460    Z OUTP      169   169   690   743   790
  4136    Z PAT1      704   704   707
  4210    Z RDUM      746   201   746
   452    Z TEMP      163   163   703   706   711   715   721   722   732   736   751   757   762   768   769   779
                            783   921   928
  4267    Z TIME      793   208   793
  4220    Z TYPE      754   206   754
  4402    Z UNB1      868   868   881
  4415    Z UNB2      879   879   886
  4420    Z UNB3      882   870   882
  4425    Z UNB4      887   867   887
  4565    Z WORD      975   733   975
  4521    ZBINST      938   917   919   938
  1504    ZBMESS      197   197   499
  1462    ZBREAK      185   184   185
  4620    ZBREM1     1002  1002  1007
  4610    ZBSET1      994   994  1000
  4627    ZBTLU1     1009   873  1009  1014
  1523    ZCTABE      211   211   693
  4542    ZGARG1      956   956   962
  4552    ZGARG2      964   960   964
  4562    ZGARG3      972   943   972
  4533    ZGARG4      949   945   949
  4541    ZGARG5      955   948   955
  4556    ZGARG6      968   966   968
  4107    Z,073,      680   680
  4110    Z,074,      680   680
  4077    Z,075,      680   680
  4104    Z,076,      680   680
  4503    Z,077,      930   930
  4510    Z,078,      930   930
  4305    ZLTIME      807   801   804   807
   456    ZRETRN      167   167   528   831   839   841   860   909   911
  1506    ZRPRAM      198   198   682
   453    ZTEMP1      164   164   720   727   752   767   774   927   935
  4304    ZTIMET      806   796   800   803   806
  4237    ZTYPE1      769   769   792
  4251    ZTYPE2      779   779   785
  4261    ZTYPE3      787   782   787
  4601    ZWRIT1      987   987   991
  4577    ZWRITE      985   725   772   793   848   882   892   896   985
   454    ZWTEMP      165   165   986   987
```

```
01557 01  05-26-72   18,180      ALGOL68                                                                   PAGE    329
OCTAL     SYMBOL     REFERENCES BY ALTER NO,
**  35220 WORDS OF MEMORY WERE USED BY GMAP FOR THIS ASSEMBLY,
      UNDEFINED SYMBOLS

30ROWE     300ROW     OUOVAC     MREFRW
```

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

@   @ @@@@   @@@   @@@  @@@@  @@@  @@@@  @@@         @@@@@ @      @@@  @@@@@ @      @    @@@
@   @ @   @ @   @ @   @    @ @   @    @ @   @        @   @ @     @   @ @   @ @     @    @   @
@   @ @   @ @        @@  @@@    @@  @@@    @@        @@@@@ @     @     @   @ @     @     @@@
@@@@@ @   @ @      @@      @  @@      @  @@          @   @ @     @ @@@ @   @ @     @@@@ @   @
@   @ @   @ @   @ @@       @ @@       @ @@           @   @ @     @   @ @   @ @     @   @ @   @
@   @ @@@@   @@@  @@@@@ @@@@ @@@@@ @@@@ @@@@@        @   @ @@@@@  @@@  @@@@@ @@@@@  @@@   @@@

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

      05/26/72          01:01:03
```

```
                         A C C O U N T I N G   R E P O R T
                         =================================


$$     01557 ENTERED SDL3,3 AT  00:15:52   FROM  TTY

   1  $        IDENT   C23232,ALGOL68
   2  $        GMAP    ON5,COMDK,DJMP,NGMAC
   3  $        TAPE    G*,A1D,,MINIC,,MINITAPE
   4  $        TAPE    *1,A2R
   5  $        TAPE    K*,A3D,,MINID,,MINITAPE
   6  $        LIMITS  300,64000,0,50000
   7  $        UPDATE  LIST
   8  $        ENDJOB

             TOTAL CARD COUNT:      22

                                  *           *           *
                                * * *       * * *       * * *
                                 * *         * *         * *


* BEGIN ACTIVITY  - 1-  GMAP    05/26/72  SW=   417700000000

 * NORMAL TERMINATION   AT    002421  INDICATORS      4020

START 00:19:02   LINES    16K      PROC 0,1600      I/O      246      IU    MEM      62K
STOP  00:39:31   LIMIT    48K      LIMIT 3,0000     LIMIT             CU    M*T   1,2630E 5

      LAPSE 0,3413      FC     D      TYPE  BUSY  IP/AT FP/RT AS/#C MS/#E ADDRESS     L#/#T

                        A*     D      DISK
                        G*     D      TAPE
                        *1     R      TAPE
                        B*     S      DISK
                        C*     S      SYOUT
                        K*     D      TAPE
                        P*     S      SYOUT
                        R*     S      DISK
      LIST     16K  LINES

                                  *           *           *
                                * * *       * * *       * * *
                                 * *         * *         * *
```

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

@  @   @@@@@  @   @  @@@@@  @  @                                      @@@@@  @       @@@   @@@@@  @       @      @@@
@ @    @   @  @   @  @      @  @                                      @   @  @      @   @  @   @  @      @      @   @
@@     @   @  @   @  @@@@   @  @                                      @@@@@  @      @      @   @  @      @       @@@
@ @    @   @  @@@@@  @      @  @                                      @   @  @      @ @@@  @   @  @      @@@@   @   @
@  @   @   @  @   @  @      @@@@                                      @   @  @      @   @  @   @  @      @   @  @   @
@   @  @@@@@  @   @  @@@@@      @                                     @   @  @@@@@   @@@   @@@@@  @@@@@   @@@    @@@

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

      05/26/72          01:01:10
```

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Dartmouth College Department of Mathematics Hanover, N.H. 03755	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

On the implementation of ALGOL 68

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Scientific Interim

5. AUTHOR(S) *(First name, middle initial, last name)*

Sidney Marshall

6. REPORT DATE June 1972	7a. TOTAL NO. OF PAGES 204	7b. NO. OF REFS 65

8a. CONTRACT OR GRANT NO. F 44620-68-C-0015 b. PROJECT NO. c. 9744 d.	9a. ORIGINATOR'S REPORT NUMBER(S) 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT
This document has been approved for Public Release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES Tech., other	12. SPONSORING MILITARY ACTIVITY Air Force Office of Scientific Research (SRMA) 1400 Wilson Blvd. Arlington, Virginia 22209

13. ABSTRACT

This thesis is concerned with implementing a compiler for the computer language ALGOL 68. The compiler contains two passes that are syntax directed followed by a third code generating pass.

Imbedded in the syntax for pass 1 and pass 2 are "actions" that are subroutine calls that perform the actual compilation. All declarations are analyzed in pass 1 and stored in tables for use by pass 2. Pass 2 rereads the source program and generates a modified Polish postfix intermediate code. Pass 2 also determines the proper sequence of "coercions" to apply. These coercions transform one data type into another and an extensive set of coercions is provided by ALGOL 68. Determining the proper sequence of coercions to apply in a particular case is not trivial and an algorithm that determines this sequence is presented.

A garbage collector is described for use in ALGOL 68 programs. This garbage collector collects all ALGOL 68 data types and requires no push-down stack.

The purpose of this thesis was to see if practical compilers for ALGOL 68 could be written. It was found that such a compiler could be written but that there were some language features that could be modified to simplify the compiler writing task.

DD FORM 1 NOV 65 1473

Security Classification