

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS TMS CHANGE NO. _____ DATE _____ PAGE NO. _____
PRODUCT NAME FORTRAN Extended Version 2
PRODUCT MODEL NO. C012 MACHINE SERIES 64/65/6600
DEPT. NO. 6231 PROJECT NO. 3PC08

SUBMITTED James J. Minnick 3/3/69 REVIEWED DR Young 3/4/69
PROJ. MGR. DATE DEPT. MGR. DATE DIRECTOR DATE
FINAL APPROVAL

DRB CHAIRMAN DATE

PPB DATE
 GEN MGR
 OTHER

INTERNAL MAINTENANCE SPECIFICATIONS

64/65/6600 FORTRAN EXTENDED

VERSION 2

DOCUMENT CLASS IMS PAGE NO. i
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

TABLE OF CONTENTS

Section

1	Introduction
2	FTNSIO
3	PS1CTL
4	SCANNER
5	LSTPROC
6	CONVERT
7	ERPRO and FORMAT
8	DATA
9	DOPROC
10	STOP
11	PAUSE
12	ARITH
13	CALL
14	IF
15	ENDPROC
16	RTNPROC
17	NAMELST
18	PRINT
19	ASSIGN
20	ENTRY
21	GOTOPRC
22	ASFPRO
23	PH2CTL
24	DECPRO

DOCUMENT CLASS IMS PAGE NO. ii
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

Section

25 PH1CTL
26 CODE GENERATION TECHNIQUE
27 PRE
28 APLISTE
29 DOPRE
30 SQUEEZE
31 OPTB
32 POST
33 MACROE
34 BUILDDT
35 PS2CTL
36 FORTRAN EXTENDED ASSEMBLER
37 ORGTAB

DOCUMENT CLASS IMS PAGE NO. iii
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

APPENDIX

- A RLIST DESCRIPTION
- B I/O CALLING SEQUENCES
- C SUBROUTINE LINKAGE
- D COMPILER I/O {SI0}
- E FLOW CHARTS {in following order}
 - E2 FTNSIO
 - E3 PS1CTL
 - E4 SCANNER
 - E5 LSTPROC
 - E6 CONVERT
 - E7 ERPRO
 - E8 DATA
 - E9 DOPROC
 - E10 STOP
 - E11 PAUSE
 - E12 ARITH
 - E13 CALL
 - E14 IF
 - E15 ENDPROC
 - E16 RTNPROC
 - E17 NAMELST
 - E18 PRINT
 - E19 ASSIGN

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. iiii
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

APPENDIX {Continued}

E20 ENTRY
E21 GOTOPRC
E22 ASFPRO
E23 PH2CTL
E24 DECPRO
E25 PH1CTL
E27 PRE
E28 APLISTE
E29 DOPRE
E30 SQUEEZE
E31 OPTB
E32 POST
E33 MACROE
E34 BUILDDT
E35 PS2CTL
E36 FORTRAN EXTENDED ASSEMBLER

DOCUMENT CLASS IMS PAGE NO. 1-1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

IMS Introduction

FORTRAN Extended is a two pass compiler; the input is FORTRAN source card images and the output as assembly language program. Version 1 assembly is by COMPASS, Version 2 assembly by FTNXAS, a one pass assembler which recognizes a subset of the COMPASS language.

PASS 1 is divided into two phases: the FTN control card, the header card, and declarative statements are processed in Phase 1, executable statements in Phase 2.

During Phase 1 the header card is processed; the COMMON, DIMENSION and EQUIVALENCE information is held in linked lists in working storage. These lists are processed at the end of Phase 1 and COMPASS instructions are issued for storage allocation.

Phase 2 converts the executable statements to an intermediate language, R-list, and when the END card is seen storage is issued for usage defined variables and program constants. The symbol table which contains the attributes of the symbolic names and labels in the program is condensed.

Thus Pass 1:

- 1} Converts all source statements to an intermediate language E-list.
- 2} Forms the symbol table.
- 3} Issues COMPASS instructions for program identification, variable initialization and storage allocation, program

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 1.2
PRODUCT NAME F0RTRAN Extended
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

constants, traceback and formal parameter initialization.

- 4} Produce the R-list intermediate file for executable statements.

Pass 1 Task Summaries

FTN is the main controlling routine. It loads the overlays, cracks the FTN control card, contains the I/O buffer area and the general purpose I/O routines.

SCANNER transforms all source statements into the intermediate language, E-list, and determines statement type. Basic syntax errors are diagnosed.

LSTPROC locates in or enters into the symbol table a given symbolic name or label.

CONVERT converts the display code representation of a constant to its internal binary form. Illegal constants such as those containing too many digits, non-octal digits in an octal constant or constants out of range are diagnosed here.

ERPRO saves diagnostic information accumulated during Pass 1 for processing in Pass 2.

DATA processes DATA statements and produces appropriate COMPASS code for data initialization.

DOPROC examines DO statements, DO-implied lists, statement numbers, statement number references and integer variable definitions and references. Determines the characteristics of DO's and index functions, diagnoses nesting and the use of statement numbers and generates R-list defining the beginning and end of each DO loop and DO-implied list.

STOP processes the STOP statement.

PAUSE processes the PAUSE statement.

IF processes all IF statements, the IF expression is translated via ARITH.

ARITH processes replacement statements and translates the arithmetic, logical, relational, or masking expressions that legally appear in any statement.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 1.3
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

ENDPROC generates R-list for exit code, issues storage for usage defined variables, processes EXTERNAL names, condenses the symbol table and sends CONLIST, the program constants, to the assembly code file.

RTNPROC processes RETURN statements.

NAMELIST processes NAMELIST statements into COMPASS instructions.

PRINT processes all I/O statements.

ASSIGN processes the ASSIGN statement.

ENTRY processes the ENTRY statement.

GOTOPRC processes all GOTO type statements.

ASFPRO processes all statement function definitions by saving the text, and processes all statement function references by expanding the E-list and inserting the text.

DECPHI processes declarative and header statements. Declarative information is held in linked lists until Phase 2, at which time COMPASS instructions are issued for storage allocation. Header statements cause COMPASS instructions for program initialization to be issued.

Initially Pass 2 issues the diagnostics to the OUTPUT file. Pass 2 may be divided by function into two principal areas, namely, the pre-processing of R-list and the actual code generation. The former phase basically entails accumulating R-list for optimization, usually one sequencer, and provides for the expansion of all R-list macros. The various optimizing routines are then called for code generation. Control, in turn, reverts back to the first area and continues to fluctuate between the two functions until all R-list on the file has been decoded. The variable dimension and formal parameter code is sent to the COMPASS file. If the R option was selected, Pass 1 is loaded and receives control if more FORTRAN programs are present; otherwise COMPASS is loaded to assemble the contents of the assembly code {COMPS} file. If the R option was not selected, the generated code is assembled by FTNXAS and the binary sent to the binary file.

PRE is the main controlling routine. It calls other Pass 2 routines and also defines a sequence. It puts out inactive label names to COMPASS, passes control to PROSEQ for optimization, and detects the end of R-list.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 1.4
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

READRL obtains R-list file input for Pass 2. It also receives as input macro expansions from MACROE. When called, it returns either a single entry plus descriptor or an entire macro reference.

ADDTOSQ accumulates the sequence in working storage.

ADDTOAP accumulates actual parameter list entries in working storage.

DOPRE examines D0 begin and D0 end macro references, standard index function macro references and all R-list instructions generated within the innermost loop of a D0 nest provided the loop is well behaved. R-list instructions are generated to count D0-loops, reference standard index functions and to materialize the control variable when necessary.

MACROE expands macro references into normal R-list form.

The following routines combine to perform the code optimization functions:

PROSEQ calls the optimizing routines and also handles the cutting down of a sequence should tree complexity or working storage limitations become a problem.

COPY recopies the sequence, generating three separate lists and removing statement markers. SQUEEZE marks redundant instructions for elimination. PURGE physically eliminates the instructions marked by SQUEEZE.

BUILDDT forms a dependency tree from the squeezed sequence. The tree reflects precedence relationships within the sequence.

OPT is the code selector. Having considered timing aspects and register usages, it calls POST with the particular instruction to be issued.

POST transforms the R-list instruction into a COMPASS card image, and eventually issues the code for the sequence to the COMPASS file.

FTNXAS is the Version 2 specialized one pass assembler.

The following routines are involved in closing Pass 2.

VARCLOS transforms the VARDIM sequence, if any exists, to ordinary sequence format and calls PROSEQ.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 1.5
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 3PC0A MACHINE SERIES 64/65/6600

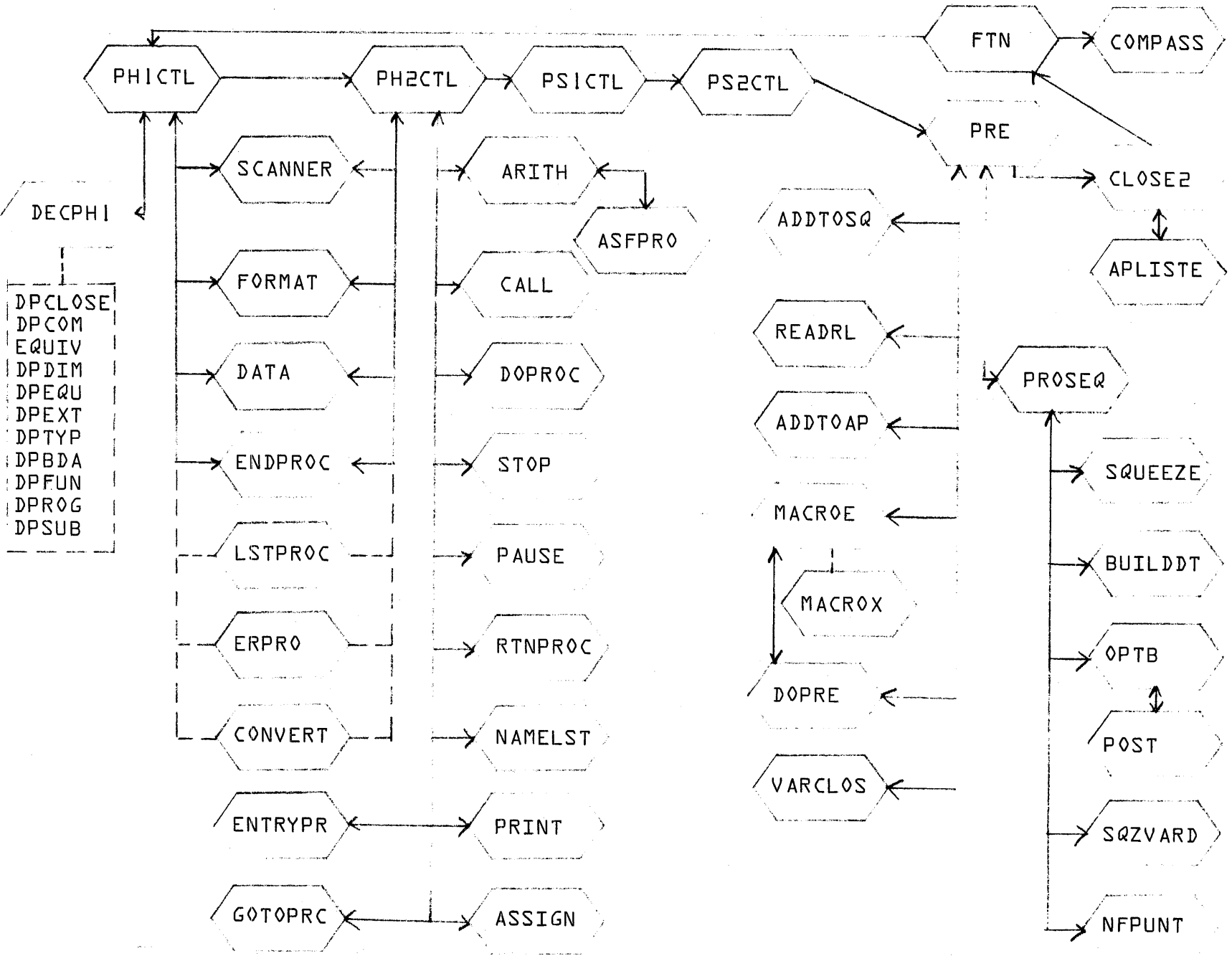
SQZVARD eliminates redundant store operations from the VARDIM initialization sequence and transforms corresponding storage allocation to the COMPASS file.

APLISTE converts APLIST entries to COMPASS card image then puts them in the COMPASS file. A SUB macro reference is generated for any formal parameter not referenced in the program.

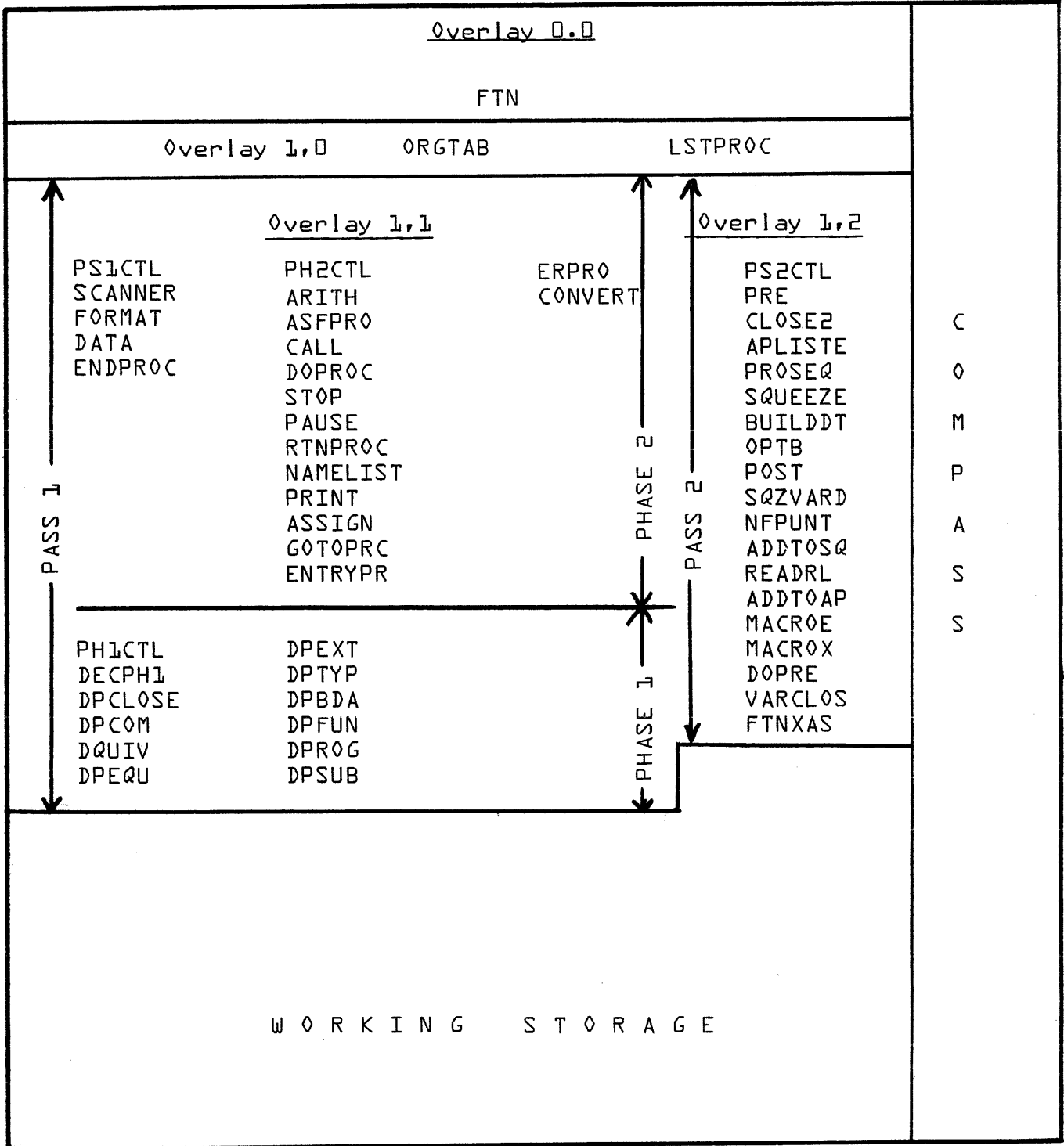
10
PAGE NO. 1-5

DOCUMENT CLASS
PRODUCT NAME
PRODUCT MODEL NO.

MACHINE SERIES



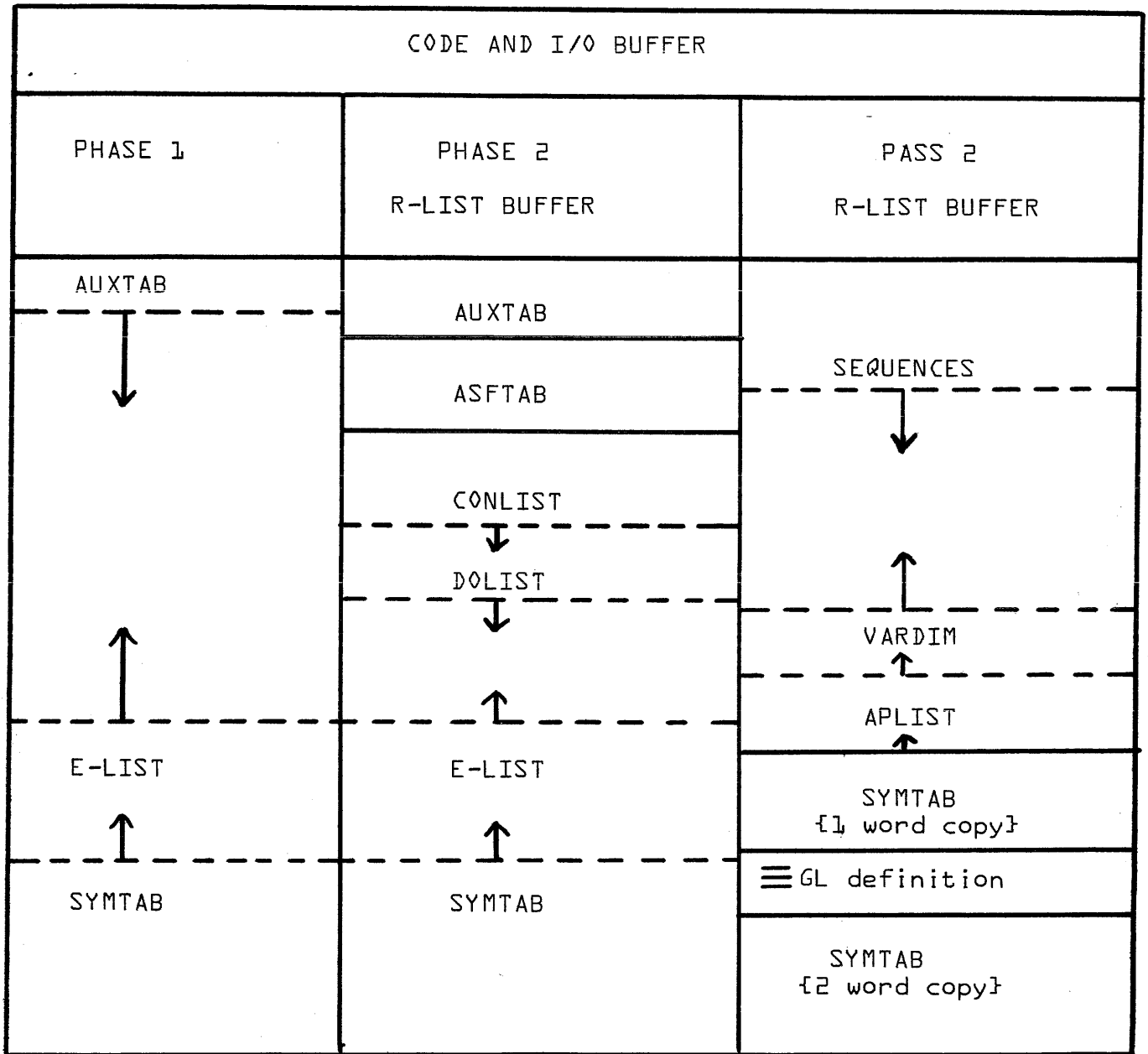
DOCUMENT CLASS IMS PAGE NO. 1.6
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 3PCDB MACHINE SERIES 64/65/6600



CONTROL DATA CORPORATION

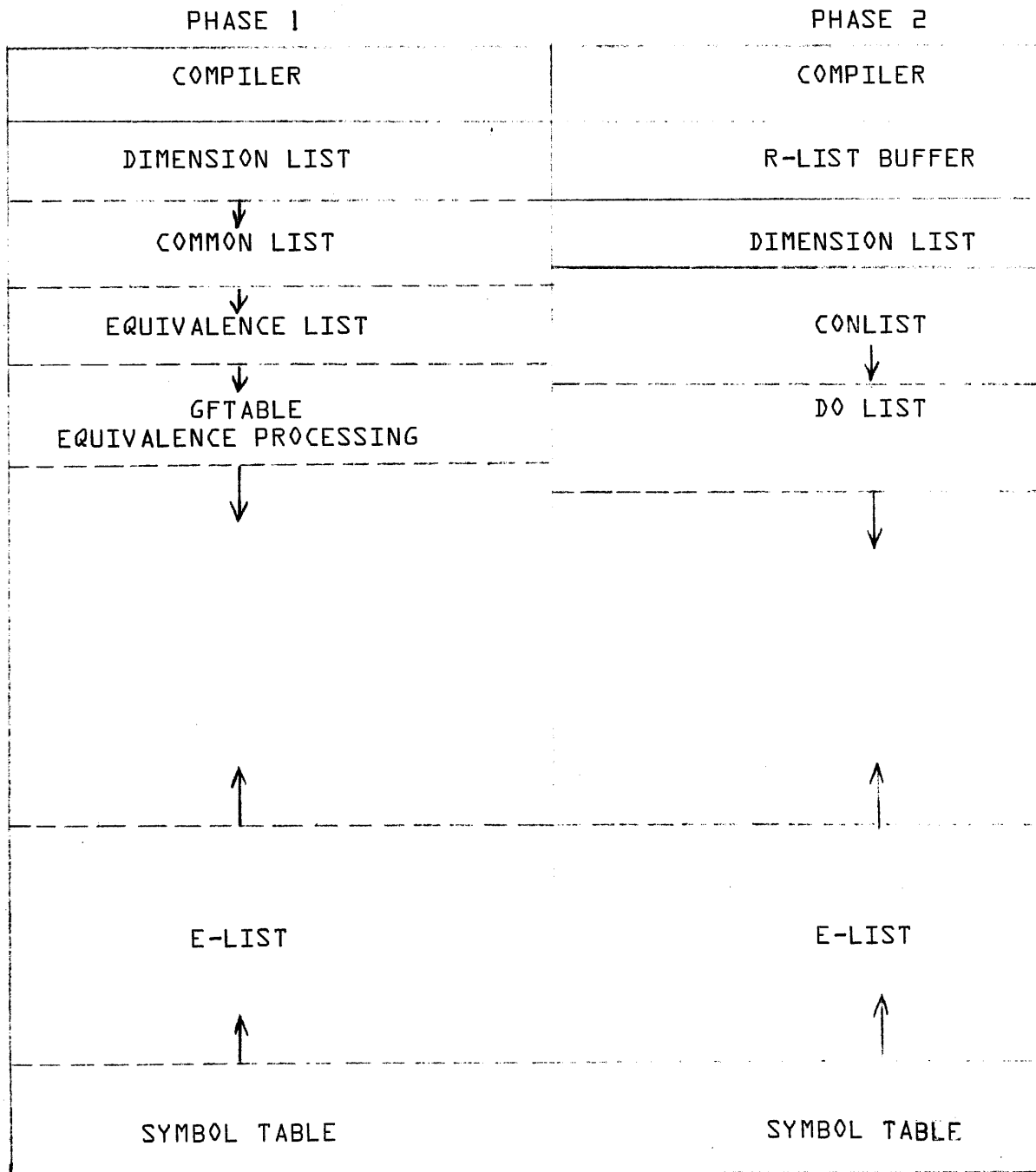
DIVISION

DOCUMENT CLASS TMC PAGE NO. 1.7
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600



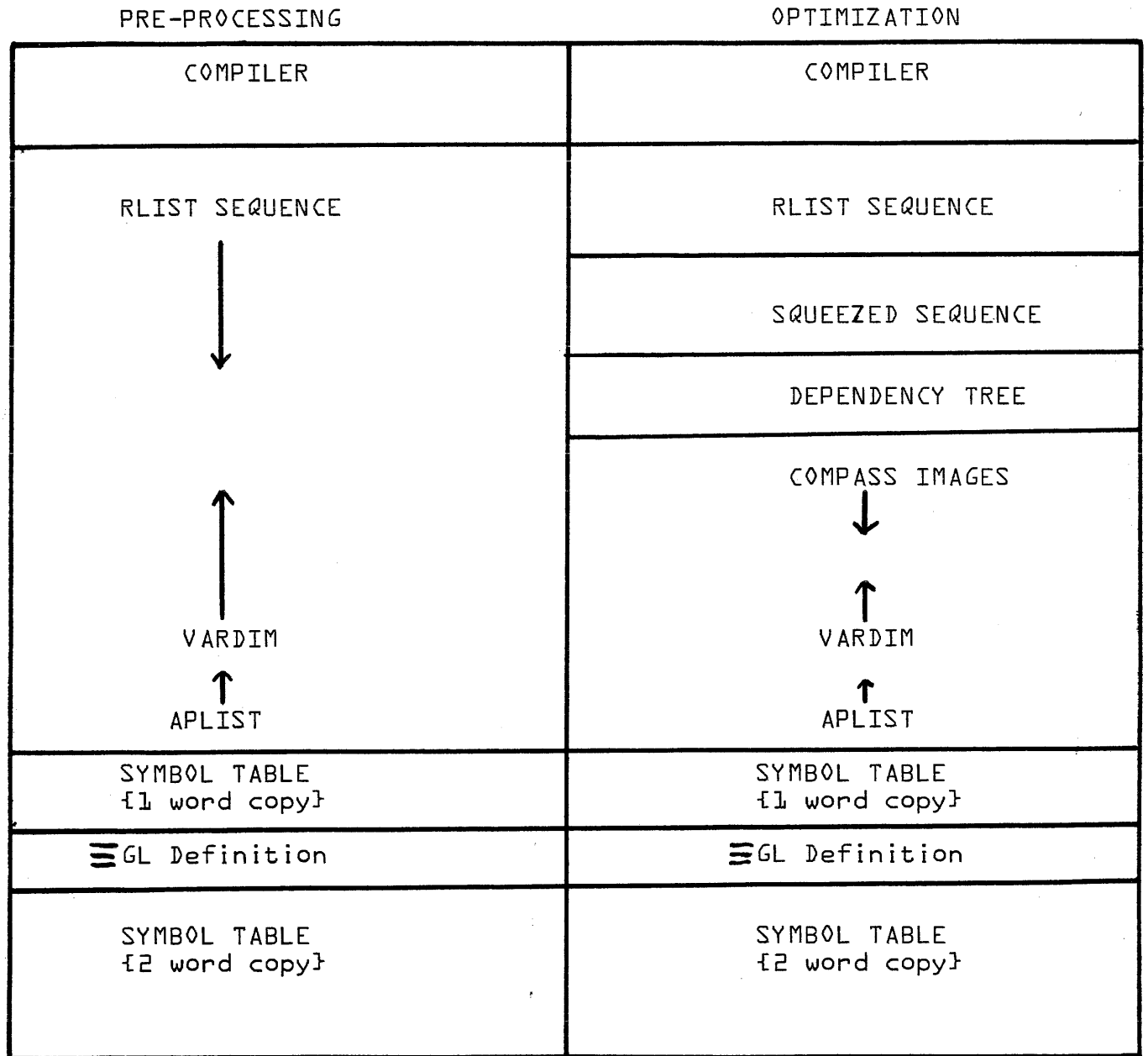
DOCUMENT CLASS _____ PAGE NO. 1.8
 PRODUCT NAME _____
 PRODUCT MODEL NO. _____ MACHINE SERIES _____

PASS 1 MEMORY



DOCUMENT CLASS IMS PAGE NO. 1.9
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

PASS 2 MEMORY



DOCUMENT CLASS IMS PAGE NO. 2.1
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3P C08 MACHINE SERIES 64/65/6600

FTNSIO

1.0 General

1.1 FTNSIO is the 0,0 or primary overlay. It contains: general purpose I/O routines, I/O buffers and FETs, and the code to load the overlays. Assembled into the I/O buffer areas is the code to crack the FTN control card and to initialize necessary files.

The I/O buffers are for the INPUT and OUTPUT files. The FETs are for the INPUT, OUTPUT, LGO or binary, and the COMPS {generated assembly language} file. The COMPS buffer is in the 1, 0 overlay and the LGO buffer is assigned a location by the assembler.

2.0 Entry point names

2.1 CI01. This entry point is used to facilitate the issuing of system I/O calls.

2.1.1. Calling sequence.

SX2. I/O parameter {see SCOPE 3.0 reference manual Chapter 3}

SB7 return address

SX1 FET address

EQ CI01

if X2 <0, the requested function is issued with auto-recall {see page 3-19 of SCOPE 3.0 reference manual for definition of auto-recall}

2.2 RCL1. This entry point is used to facilitate issuing the 'recall' request to the operating system.

2.2.1 Calling sequence SX1 FET address
 SB7 Return address
 SX2 P
 EQ RCL1

If P=0 control will not be returned to the program until bit 0 of word 1 of the FET becomes a 1.

If P=0 the central processor will be relinquished only until the next time around the monitor loop.

See Chapter 3 of the SCOPE 3.0 reference manual for definition of the two types of RECALL.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 2.2
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3P C08 MACHINE SERIES 64/65/6600

2.3 REWIND This entry point is used to rewind output files used by the compiler. An end-of-file write is requested followed by a rewind request.

2.3.1 Calling sequence

SBB file number *
RJ REWIND

*See paragraph 5.0

2.4 TITLE1. This entry point is the first word of a 13 word title which is listed at the beginning of each page.

2.5 LCNT. This one word entry point holds the number of lines left to be printed on a page. If a page eject is desired, LCNT is set to zero.

2.6 LIST. This entry point is used to output compiler information to the output file specified on the control card.

2.7 PAGE. This one word entry point contains the current page number in display code. Setting PAGE to zero will initialize the page number to one.

2.8 LWAERTL. Address of the last word in the error table, used by ERPR0 to detect table overflow.

2.9 LDCOM Entry point which is used to generate a call to the LOADER to load the COMPASS overlay. COMPASS is loaded at 3000B.

2.10 ERACCUM. A one word flag which indicates if any fatal errors have occurred during a run. Zero indicates no errors, non-zero indicates errors.

2.11 LOVER. Routine which calls the LOADER to load overlays. See SCOPE 3.0 reference manual Chapter 4 calling sequence to LOADER.

2.12 PLUG 1, PLUG 2. An entry point {s} in which an RJ WRWDS for {WRWDS2} is stored to an output routine in each pass of the compiler. Since it is not possible to link from the 0,0 overlay to the 1,0 or 1,2 overlays, each pass of the compiler stores a RJ instruction in these locations.

2.13 Processing Flow Description

Control is obtained through the entry point FTN and

DOCUMENT CLASS IMS PAGE NO. 2.3
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3P C08 MACHINE SERIES 64/65/6600

then cracks the control card and opens the I/O files. {See SCOPE 3.0 reference manual, ch.3, page 3.21}. LOVER is then called to load the 1,0 overlay. The 1,0 overlay will allocate storage for the COMPS buffer then return control to LDRPH1 in FTNSI0 which then loads the 1,1 overlay or Pass 1 of the compiler. The 1,0 overlay is loaded only once while the Pass 1 {1,1} and Pass 2 {1,2} overlays once for each sub-program under the standard configuration.

3.0 Diagnostics. No diagnostics are produced by FTN.

4.0 Structure

4.1 Major Subroutine Names

4.1.1 INIT. This routine cracks the FTN card and sets up the appropriate options.

L option selected, set N0LSTFL {loc 50B} to non-zero

X option selected, set NASAFLG {loc 42B} to non-zero

R option selected, set bits 30-59 at loc 33B to non-zero

0 option selected, set bit 0 of location 33B

B option, the file name specified is placed in location 4B

L {output} file name is placed in location 3B

I {input} file name is placed in location 2B

T option selected, location 45B is set to non-zero

E option selected, location 10B is set to non-zero, file name or COMPS placed in location 6B

4.1.2 DONE. This routine makes system calls to 'OPEN' the I/O files and requests the time and date from the operating system.

4.1.3 0TITLE. This routine is used to update the page count when it is necessary to output a title line. The number of lines per page may be changed by modifying an EQU call LMAX in the common file call OPTIONS.

5.0 All files, names {and FET addresses} used by the compiler are placed in RA+2 thru RA+6. All routines which

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 2.4
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3P COB MACHINE SERIES 64/65/6600

wish to reference a particular file do so by number.

File number designations.

- 1 INPUT
- 2 OUTPUT
- 3 LGO
- 4 RLIST
- 5 COMPS

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 3.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

PS1CTL

1.0 General Information

PS1CTL resides in the Pass 1 overlay. Its primary function is to load in the second pass. It contains the compiler output buffer filling routine WRWD1.

2.0 Usage

2.1 LDPS2

2.1.1 Entering at LDPS2 will result in the R-list intermediate file being rewound and a call to LOVER being made to load the overlay READRL\$ which is the name of Pass 2 of the compiler.

2.1.2 Calling sequence:

LDPS2 is entered via a branch to the entry point of the same name.

2.2 WRWDS

2.2.1 This routine is used to output information. The calling sequence is:

SB6 "file number"

SB7 "fwa" of central memory to transfer from

SB1 "number of words to transfer"

RJ WRWD2

7.0 Modification Facilities

The OVERLAY parameter of OPTIONS is interrogated in this routine.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 4.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600

SCANNER

1.0 General information

1.1 SCANNER transforms all source statements into the intermediate language E-list, then determines the statement type so the proper statement processor can be called. SCANNER is called by PH1CTL and PH2CTL and thus resides in overlay 1.0.

2.0 Usage

2.1 SCANNER

2.1.1 Upon initial entry from PH1CTL;

1. TITLE1 and TITLE 1+1 will be initialized to all blanks.
2. Any comment or blank cards are placed in the LIST file via FCRD. If, before a PROGRAM, SUBROUTINE, FUNCTION, or BLOCK DATA statement is seen the characters IDENT appear in columns 11-15, that card image along with all subsequent cards up to and including the first card with END characters in 11-13 and a blank in 14 are sent to the COMPASS file via FCRD.

When a PROGRAM, SUBROUTINE, FUNCTION, or BLOCK DATA statement is seen with a "name" on the first card the "name" is placed in TITLE1+1 and the statement identifier placed in TITLE1. TITLE1 is an entry point in the SIO program. Each source statement is scanned individually and transformed into E-list before control is returned to the phase controller.

2.1.2 Calling Sequence and Returns

The calling sequence is RJ SCANNER. Upon return register B7 and TYPE(RA+24B) will hold the primary statement type. ATYPE (RA+51B) will hold any constant type associated with the statement. If the

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 4.2 _____
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

statement is a logical IF, the type of the statement following the logical expression is found in LTYPE (RA+21) and the starting address in LELIST (RA+34B). The starting address of E-list for the primary statement is found in SELIST. The last location used for E-list is found in ELAST(RA+14B). CLABEL(RA+23B) holds in display code the statement label, if any, left justified and blank filled. NLABEL (RA+60B) holds the label, if any, in the same form for the next statement. If no label is present NLABEL and CLABEL will be zero.

2.3 Processing Flow Description

SCANNER expects SYMEND(RA+13B) and CSTOR1(RA+16B) to be initialized. SIO is called to transfer one card image from the input file to a buffer local to SCANNER. The 80 display code characters are stored one per word, right justified zero filled. The card image is checked for 1) all blanks, 2) a comment card, and 3) an IDENT card. If not one of these a string is formed (left to right scan) containing all initial alphanumeric characters and terminated by sensing the next statement or an operator or delimiter in the statement. The string is checked for PROGRAM, SUBROUTINE, FUNCTION, BLOCK DATA "name" and when found the identifier and "name" is placed in the list file header line and the entire statement is then processed. If not found either an error has occurred or packing of the alphanumeric string will continue. In general a left to right alphanumeric string is formed and terminated by an operator or delimiter or sensing the next statement. Depending upon the condition that terminated the string, control is passed either to determine the statement type or transform the statement to E-list until the condition exists when the statement can be

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 4.3
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

typed. After the statement has been typed the remainder of the statement is simply transformed to E-list until the next statement is sensed. It is sometimes necessary to separate the statement identifier from a symbolic name or constant. This is done when the statement is typed and the SELIST pointer is set. When the next statement is seen the ELAST pointer is set and control is returned to the caller.

3.0 Diagnostics

3.1 Fatal to Compilation

3.1.1 TABLES OVERLAP, MORE MEMORY REQUIRED.

3.2 Fatal to Execution

3.2.1 UNRECOGNIZED STATEMENT

3.2.2 ILLEGAL CHARACTER IN LABEL FIELD

3.2.3 STATEMENT TOO LONG

3.2.4 SYMBOLIC NAME TOO LONG, MAX IS 7

3.2.5 UNMATCHED PARENTHESIS

3.3 Non ASA Diagnostics

3.3.1 7 CHARACTER SYMBOLIC NAME IS NOT ASA

4.0 Environment

The address placed in SELIST is found by subtracting 15 from the address found in SYMEND. The E-list is formed growing into smaller addressed memory. Constants are placed in CONSTOR (initial address is found in (STOR1) left adjusted blank filled and CONSTOR grows into higher addressed memory. Also expected to be available via SIO are the input file, the COMPASS file and the list file.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 4.4
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

5.0 Structure

The subroutines are listed in the order of appearance in SCANNER.

5.1 FRCRD

FRCRD fetches the next source card from the input file and places the 80 characters one to a word, right adjusted zero filled. The first request will not result in a check to see if the card image should be placed in the list file. All subsequent requests will check NOLSTFL(RA+50B) (non-zero means yes) to see if a listing is desired and further check FSTSW (a local flag and non-zero means yes) to list the card image. FSTSW is turned to OFF when a COMPASS program is being sent to the COMPASS file. The line count is updated and inserted in the listing every five lines.

5.2 FCRD

Transfers the last read card image into the list file when FSTSW and NOLSTFL are set non-zero.

5.3 PACK

When expecting a statement identifier PACK will pack alphanumeric characters 10 per word and store in working storage (SELIST). When expecting a symbolic name PACK will make the E-list entry for the symbolic name.

5.4 SEARCH

Using the three tables described in Section 6, SEARCH tries to identify the statement type. If successful the statement type code is placed in TYPE, any arithmetic type in ATYPE, and the number of characters in the statement identifier is in XO.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 4.5
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. _____ 4P616 _____ MACHINE SERIES 64/65/6600

5.5 ADJ

ADJ separates the statement identifier from any symbolic name or constant that was packed with it at the start of statement processing. Either a symbolic name or constant E-list entry is made and SELIST adjusted as necessary.

5.6 GET

GET keeps track of the current column of the source card image being interrogated and upon request returns to the caller the next non-blank character in register B2. The blank squeeze takes place here. After column 72 the routine NEXT is called to transfer the next card image to the local buffer.

5.7 NEXT

NEXT calls FRCRD to transfer the next card image from the input file to location SBUFF an 80 word buffer. When either an all blank card or a comment card is seen, the next card image is requested. For a continuation card return is made to the caller. When a new statement starts, several checks are made: 1) if an error occurred on the statement being processed and the statement was typed return is made to SCANNER's caller via the routine STATED, otherwise processing starts on the new statement, 2) any symbolic name E-list entry is completed, 3) any constant E-list entry is completed. If the statement has been typed return is made to the phase controller via SCANNER's entry point. Otherwise, the necessary statement typing routine is called and an attempt is made to type the statement.

5.8 STATE1

STATE1 makes sure an alphabetic character starts the statement and then

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 4.6
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

calls PACK to pack a statement identifier. PACK returns control to STATE 2.

5.9 STATE 2.

STATE2 is a jump vector which transfers control to the proper routine depending on the condition that terminated the statement identifier packing.

5.9.1 The characters + - *) . cause an unrecognized statement diagnostic.

5.9.2 S. A slash terminates the string. SEARCH is called to check for DATA N/, COMMON/, COMMON N/, or NAMELIST/. After successful typing and adjusting control is returned to STATE3.

5.9.3 Ll. A left parenthesis terminates the string. If the string is FORMAT and the statement was labeled, control is transferred to FORMAT to process the statement. If not, a parenthesis count is started and control is transferred to STATE3.

5.9.4 D1. The typing routine for an all alphanumeric statement. SEARCH is called to look for any form of: CONTINUE, STOP, ECS, GOTO, PAUSE, CALL, READ, REAL, ENTRY, PRINT, PUNCH, RETURN, COMMON, DOUBLE, REWIND, COMPLEX, ENDFILE, INTEGER, LOGICAL, PROGRAM, TYPEECS, EXTERNAL, TYPEREAL, BLOCKDATA, BACKSPACE, SUBROUTINE, TYPEDOUBLE, TYPECOMPLEX, TYPEINTEGER, TYPELOGICAL, ASSIGN, DOUBLE PRECISION, TYPEDOUBLEPRECISION. After successful typing and adjusting control is transferred to STATE0.

5.9.5 E1. An = sign terminates the string. If the string is from 8 to 14 characters long a check is made for a DO statement. If so control is passed to ADJDO. If the string is less than 8, control is passed to STATE6.

5.9.6 CC1. A comma terminates the string. SEARCH is called to look for the forms of: ECS, GOTO, CALL, REAL, DATA, READ, PRINT, PUNCH, COMMON,

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 4.7
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

DOUBLE, COMPLEX, INTEGER, LOGICAL, TYPEECS, EXTERNAL, TYPREAL,
SUBROUTINE, TYPEDOUBLE, TYPECOMPLEX, TYPEINTEGER, TYPELOGICAL,
DOUBLEPRECISION, TYPEDOUBLEPRECISION. After typing and adjusting
control is returned to STATE8.

5.10 STATE 3

STATE 3 transforms symbolic names, constants, operators, and delimiters
into E-list until the parenthesis count is zero, then control is passed
to STATE5.

5.11 STATE 5

STATE5 contains a jump vector to pass control to the processing routine
depending on the character that appears immediately after the parenthesis
count goes to zero.

5.11.1 The characters + - *) blank and . will cause an unrecognized statement
diagnostic.

5.11.2 P5. Is entered when an alphabetic follows when paren count goes to 0.
If the string length before the first left paren is 2 a check is made for
a logical IF. If so, control is passed to STATE8. If not SEARCH is
called to look for any of the forms of: GOTO, READ, WRITE, ENCODE, DECODE.
After successful typing and adjusting control is passed to STATE8.

5.11.3 N3. Is entered when a digit follows as paren count goes to 0. Check
the string before the first left parenthesis for IF and if so, assume
an arithmetic IF then pass control to STATE8.

5.11.4 SD. A slash causes SEARCH to be called to look for any of the forms of
DATA and COMMON. After typing and adjusting control is passed to
STATE8.

CONTROL DATA CORPORATION . COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ TMS. _____ PAGE NO. 4.8
PRODUCT NAME _____ **FORTRAN Extended** _____
PRODUCT MODEL NO. _____ **4P616** _____ MACHINE SERIES 64/65/6600

5.11.5 L3. A left parenthesis causes SEARCH to look for any proper form of READ, WRITE, ENCODE, DECODE, BUFFERIN, BUFFEROUT and after typing and adjusting pass control to STATE8.

5.11.6 D3. The statement is terminated at parenthesis count=0. After checking for WRITE and EQUIVALENCE, SEARCH is called to look for any form of: READ, DATA, ECS, CALL, REAL, COMMON, DOUBLE, COMPLEX, INTEGER, LOGICAL, PROGRAM, TYPEECS, TYPREAL, FUNCTION, DIMENSION, SUBROUTINE, TYPEDOUBLE, TYPECOMPLEX, TYPEINTEGER, TYPELOGICAL, REAL FUNCTION, DOUBLE FUNCTION, COMPLEXFUNCTION, INTEGERFUNCTION, LOGICAL FUNCTION, DOUBLE-PRECISION, TYPEDOUBLEPRECISION, DOUBLEPRECISIONFUNCTION and after typing and adjusting pass control to STATE0.

5.11.7 E3. The = sign here causes the type to be set replacement and control is passed to STATE8 after making the string before the first left paren into a symbolic name entry.

5.11.8 CC3. The comma causes a check made for EQUIVALENCE and then SEARCH is called to look for the forms of: DATA, GOTO, ECS, CALL, REAL, DOUBLE, COMMON, COMPLEX, INTEGER, LOGICAL, TYPEECS, TYPREAL, DIMENSION, SUBROUTINE, TYPEDOUBLE, TYPECOMPLEX, TYPEINTEGER, TYPELOGICAL, DOUBLE-PRECISION, TYPEDOUBLEPRECISION and after typing and adjusting pass control to STATE8.

5.12 STATE6

STATE6 determines if the statement is a replacement or a DO. A jump vector passes control depending upon the first character after the = sign.

5.12.1 A / *) = \$, cause a unrecognized statement diagnostic to be issued.

5.12.2 P6. An alphabetic causes PACK to be called to pack a symbolic name then

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 4.9
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

pass control to STATE 7.

5.12.3 N4. A digit causes DIGIT to be called to make the E-list entry for a constant and then pass control to STATE10.

5.12.4 A + - (or . cause the statement to be typed replacement and control passed to STATE8 after making a symbolic name entry of the string before the = sign.

5.13 STATE7

STATE 7 has a jump vector and passes control depending upon the character that terminated the symbolic name after the = sign.

5.13.1 A + - * (= or . cause the statement to be typed replacement and control passed to STATE8 after making a symbolic name entry of the string before the = sign.

5.13.2 A) will cause an unrecognized statement diagnostic to be issued.

5.13.3 CC4. A , will cause a check of the string before the = sign for a DO. If the first two characters are DO a jump is made to ADJDO to make a constant and symbolic name entry and then pass control to STATE8.

5.14 STATE8

The remaining elements of the statement are transformed into E-list and stored until the statement is terminated either by an error occurring, or a new statement being sensed.

5.15 STATE10

STATE10 contains a jump vector and passes control depending upon the character that terminated the constant that appeared after the first = sign.

5.15.1 A + - * / (or . will cause the statement to be typed replacement and control passed to STATE8.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 4.10
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. 4-P616 _____ MACHINE SERIES 64/65/6600 _____

5.15.2 A) will cause an unrecognized statement to be issued.

5.15.3 CC5. A , will cause the string on the left of the = sign to be checked for a DO statement by calling ADJDO.

5.16 STATEO

STATEO inserts the end-of-statement code into E-list. QLABEL is called to scan the label field (columns 1-5) and make the proper entries in CLABEL and NLABEL. Return is then made to the phase controller via SCANNER's entry point.

5.17 QLABEL

QLABEL scans the label field (SBUFF through SBUFF+4) for a legitimate label. If found, the label is placed in NLABEL left adjusted and blank filled after first transferring the contents of NLABEL to CLABEL.

5.18 ADJDO

E-list and constor entries are made for the u i in the DO u i part of the DO statement. DO u i is stored as a statement identifier at SELIST. A symbolic name entry is made for i and an integer constant entry made for u.

5.19 ASSIGN

The ASSIGN k TO i string is stored at SELIST. ASSIGN makes a symbolic name E-list entry for i and an integer constant entry for k.

5.20 CFSNC

Checks all two character symbolic names for register names (A0-A7, B0-B7, X0-X7) and if so suffixes a currency symbol to the symbolic name E-list entry.

5.21 POINT

Determines the element that starts with a decimal point (constant,

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 4.11
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. VP616n 1.0 MACHINE SERIES 64/65/6600

relational or logical operator) and makes appropriate E-list and CONSTOR entries.

- 5.21.1 L01. Checks one character alphabetic string bracketed by decimal points for A, F, N, O, T.
- 5.21.2 L02. Checks two character alphabetic strings bracketed by decimal points for EQ, GE, GT, LE, LT, NE, OR.
- 5.21.3 L03. Checks three character alphabetic strings bracketed by decimal points for AND, NOT.
- 5.21.4 L04. Checks four character alphabetic strings bracketed by decimal points for TRUE.
- 5.21.5 L05. Checks five character alphabetic strings bracketed by decimal points for FALSE.
- 5.21.6 PACKC. Packs one character at a time into the CONSTOR entry for a constant.
- 5.21.7 PACKT. Fills the last word of the CONSTOR entry with blanks and makes the E-list entry for the constant.
- 5.22 DIGIT
Determines the constant type element that begins with a digit, and calls PACKC and PACKT to make appropriate CONSTOR and E-list entries.
 - 5.22.1 CFD. Checks the alphabetic character that terminates an all digit string for E, D, B, H, L, R.
 - 5.22.2 REXPPP. Inserts a decimal point in the constant string just before the E.
 - 5.22.3 DEXPPP. Inserts a decimal point in the constant string just before the D.
 - 5.22.4 REXP. Sets constant type to real, then expects an exponent.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 4.12
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- 5.22.5 DEXP. Sets constant type to double, then expects an exponent.
- 5.22.6 KD. When a decimal point follows a digit string the constant type is set to real, then expect a fractional part or a D or E.
- 5.22.7 IIIC. When expecting either a fractional part of the constant or a D or E.
- 5.22.8 IIIIC. Expecting an optionally signed exponent that must appear after a D or E.
- 5.22.9 VC. Packs digits in the exponent field until the constant is terminated.
- 5.22.10 OEXP. Sets constant type to octal and terminates constant via PACKT.
- 5.22.11 HEXP. Converts the n in the nH, nR, or nL field to binary, then packs the next n characters into a Hollerith constant.
- 5.23 FORMAT
- Beginning with the first left parenthesis that follows the characters FORMAT, the entire statement is packed ten characters per word and stored beginning at SELIST. The last word is blank filled.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 4.13
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

6.0 FORMATS

6.1 E-list format

<u>Element</u>	<u>E-list format</u>
constant	VFD 12/2000B, 3/t, 6/s, 11/0, 10/n, 18/Pointer
symbolic name	VFD 12/2001B, 48/Name
)	VFD 12/2002B, 48/0
,	VFD 12/2003B, 48/0
end-of-statement	VFD 12/2004B, 48/0
=	VFD 12/2005B, 48/0
(VFD 12/2006B, 48/0
.OR.	VFD 12/2007B, 48/2
.AND.	VFD 12/2010B, 48/3
.NOT.	VFD 12/2011B, 48/4
.LE.	VFD 12/2012B, 48/5
.LT.	VFD 12/2013B, 48/5
.GE.	VFD 12/2014B, 48/5
.GT.	VFD 12/2015B, 48/5
.NE.	VFD 12/2016B, 48/5
.EQ.	VFD 12/2017B, 48/5
-	VFD 12/2020B, 48/6
+	VFD 12/2021B, 48/6
*	VFD 12/2022B, 48/7
/	VFD 12/2023B, 48/8
**	VFD 12/2024B, 48/10

For a constant entry t = 0 for logical, 1 for integer, 2 for real, 3 for double precision, 5 for octal, and 6 for Hollerith. When T = 6, s = 0 for the H form, s = 1 for the L form and s = 2 for the R form.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 4.14
PRODUCT NAME _____ **FORTRAN Extended** _____
PRODUCT MODEL NO. 4F616 _____ MACHINE SERIES 64/65/6600

n is the number of characters in the constant string and Pointer is the starting address of the string in CONSTOR. For logical constants the Pointer field will hold -1 for TRUE and 0 for FALSE and the n field is 0 and no CONSTOR entry is necessary.

6.2 Statement Type Codes

Each statement has an associated type code which has the following significance; it is the ordinal in a jump vector of the statement processing program. The elements that actually appear in E-list are underlined.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 4.15
 PRODUCT NAME _____ FORTRAN Extended _____
 PRODUCT MODEL NO. _____ 4P616 _____ MACHINE SERIES 64/65/6600 _____

Statement	Statement and E-list entries
Code Number	
0	PROGRAM <u>s</u> PROGRAM <u>s (...)</u>
1	BLOCK DATA BLOCK DATA <u>s</u>
2	SUBROUTINE <u>s</u> SUBROUTINE <u>s (a₁, a₂, ..., a_n)</u> SUBROUTINE <u>s, RETURNS (b₁, b₂, ..., b_m)</u> SUBROUTINE <u>s, (a₁, a₂, ..., a_n), RETURNS (b₁, b₂, ..., b_m)</u>
3	t FUNCTION <u>s (a₁, a₂, ..., a_n)</u>
4	COMMON <u>/x₁/a₁/.../x_n/a_n</u>
5	DIMENSION <u>v₁, v₂, ..., v_n</u>
6	EXTERNAL <u>v₁, v₂, ..., v_n</u>
7	EQUIVALENCE <u>(k₁), (k₂), ..., (k_n)</u>
8	INTEGER, TYPE INTEGER, REAL, TYPE REAL, COMPLEX, TYPE COMPLEX, DOUBLE, TYPE DOUBLE, DOUBLE PRECISION, TYPE DOUBLE PRECISION, LOGICAL, TYPE LOGICAL, ECS or TYPE ECS <u>v₁, v₂, ..., v_n</u>
9	FORMAT <u>(...)</u>
10	DATA <u>k₁/d₁/, ..., k_n/d_n/ or (r₁=d₁), ..., (r_n=d_n)</u>
11	NAMelist <u>/y₁/a₁/.../y_n/a_n</u>
12	f <u>(a₁, a₂, ..., a_n) = e or v=e</u>
13	END
14	ASSIGN <u>k TO i</u>

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 4.16
 PRODUCT NAME _____ **FORTRAN Extended** _____
 PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600

15 GO TO k
 GO TO i, (k₁, k₂, ...k_n)

16 IF (e) k₁, k₂, k₃

17 IF (e) S

18 not used at this time

19 CALL s
 CALL s (a₁, a₂, ..., a_n)
 CALL s, RETURNS (b₁, b₂, ...b_m)

20 RETURN
 RETURN i

21 CONTINUE

22 STOP
 STOP n

23 PAUSE
 PAUSE n

24 DO n i = m₁, m₂, m₃

25 READ f, k
 READ (u) k
 READ (u, f) k
 READ (u, f)

26 WRITE (u) k
 WRITE (u, f) k

27 BUFFER IN (u, k) (A, B)

28 BUFFER OUT (u, k) (A, B)

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 4.17
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

29 ENCODE (n, f, A) k
30 DECODE (n, f, A) k
31 REWIND u
32 BACKSPACE u
33 ENDFILE u
34 PRINT f, k
35 PUNCH f, k
36 ENTRY s
37 END card assumed for end-of-record

Statement types 3 and 8 need arithmetic typing information ATYPE (RA+51B) holds 0 for logical, 1 for integer, 2 for real, 3 for double precision 4 for complex, 5 for ECS and -0 means a FUNCTION not typed.

6.3 SEARCH Table Formats

The SEARCH program utilizes three tables. Each condition that requires a search has two distinct tables plus a third table common to all conditions. The conditions that use the search are:

- (1) An all alphanumeric statement.
- (2) A , after an all alphanumeric identifier.
- (3) An identifier, then statement terminated a zero parenthesis count.
- (4) An identifier, parenthesis count equal zero, then a slash.
- (5) An identifier, parenthesis count equals zero, then a left parenthesis.
- (6) An identifier, then a slash.
- (7) An identifier, parenthesis count equal zero, then an alphabetic

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 4.18
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. _____ 4P616 _____ MACHINE SERIES 64/65/6600

character.

- (8) An identifier, then parenthesis count equal zero, then a , .
- (9) The initial statement.

The search keys on the number of alphanumeric characters that appear in the initial string. Table 1 thus has one word containing the number of statement possibilities as determined by the length of the string. In addition to this Table 1 has a pointer to the Table 2 location that contains the following information: The location (in Table 3) of the display code representation of the statement identifier and the location to jump to upon a successful match. The format of Table 1 is:

VFD 12/200nB, 48/Table 2 location

n = the number of identifier possibilities

The format of Table 2 is:

VFD 30/jump address, 30/Table 3 location

The format of Table 3 is:

VFD 12/200mB, 48/statement code

VFD 60/display code picture of identifier

m = the number of characters in picture

Thus for a given condition, the n Table 2 entries (in sequence) are used to find the pictures to compare to the string.

7. Modification Facilities

Changes in the language are easily incorporated into the scanning techniques by adding the pictures to Table 3 and making additions and/or modifications to Tables 1 and 2.

8. Method

Not applicable.

DOCUMENT CLASS IMS PAGE NO. 5.1
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/660

LSTPROC

1.0 General

- 1.1 LSTPROC contains the routines which fetch from or enter into the symbol table a given symbol or label. LSTPROC is called by CONVERT, DATA, DOPROC, STOPP, PAUSEP, ARITH, ENDPROC, RTNPROC, NAMLIST, PRINT, ASSIGN, ENTRYPR, GOTOPRC, DPCOM, DPDIM, DPEQU, DPEXT, DPTYP, DPBDA, DPFUN, DPROG, DPSUB, FORMAT, and FTNXAS.

LSTPROC calls one external routine, ERPRO.

As an instrument for storing data the symbol table is active during Pass 1 only. The two word symbol table is saved for the FTNX assembler during Pass 2, while the rest of Pass 2 references the one word copy of the symbol table directly. The assembler uses only the finding feature of LSTPROC.

Throughout pass 1, symbol table entries are two words in length. Any necessary information which does not fit in the two word entry will be kept in an auxiliary list elsewhere in memory. The symbol table will begin at FL-1 and expand (as new entries are made) into lower addressed consecutive locations, while the auxiliary table is built from the first available location in low core and expanded into higher addressed locations. This auxiliary table contains the dimension information as well as the information required to process COMMON and EQUIVALENCE statements.

At the end of phase 2 (during END processing), the symbol table will be condensed to one word per entry. The entry will consist of the symbol (or label), and other information needed by DOPRE or other processors during pass 2 (see page 8).

The two word copy of the symbol table is retained for use by the FTNX assembler.

2.0 Usage

- 2.1 Entry Point Names: SYMBOL, LABEL

- 2.1.1 SYMBOL searches for a given 7-character symbol in the symbol table. If the symbol is already in the table the entry is loaded and SYMBOL returns to the caller. If the SYMBOL is not presently in the table, it is entered in the table, loaded, and SYMBOL returns to the caller.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 5.2
PRODUCT NAME FORTTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

LABEL searches for a given 6-character statement label in the symbol table in exactly the same manner as SYMBOL searches for symbols.

2.1.2 Calling Sequence and Returns

Entry is made to SYMBOL or LABEL via a direct jump (not a return jump) with the following register requirements:

X1: The symbol (or label) left justified in bits 0-47 with blank fill. The contents of bits 48-59 is insignificant.

B7: The address to which control is to be returned if the symbol was not already in the table.

B7+1: The address to which control is to be returned if the symbol was already in the table.

Control is returned to the caller with:

B1 = ordinal of word 1 of the two word entry.

B2 = double the ordinal of word 1 of the entry (B1+B1).

B5 = 1

X1 = word 1 of the entry.

X2 = word 2 of the entry.

A0 = starting address of the symbol table.

A1 = address of word 1 of the entry.

A2 = address of word 2 of the entry.

3.0 Diagnostics

3.1 One fatal to compilation condition may be detected: "SYMBOL TABLE OVERFLOW" (a maximum of 8192 words is used for the symbol table).

3.2 No fatal to execution errors are detected.

3.3 No information diagnostics are issued.

3.4 No non-ASA errors are detected.

4.0 Environment

When LSTPROC is entered, it is expected to search for a given symbol or label, enter the symbol or label if it is not presently in the table, and return the entry to the caller. Hence, no conditions are expected to be set up by any other processors (with the exception of the common cells noted in section 7.0 of this document).

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 5.3
PRODUCT NAME FORTTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

During Pass 2 the storing feature of LSTPROC is disabled. When the assembler jumps to SYMBOL and the name is not found, the last non-blank character of the name is checked for . or \$, and if so is replaced by a blank and another attempt is made to find the name. If the name has seven non-blank characters and the last character is not . or \$, the last character is exclusive "or"ed with an 06 and another attempt is made to find the name. This procedure is necessary because special characters are suffixed to intrinsic and basic external function names, and although they appear that way in the one word symbol table, in the two word symbol table only the name appears.

5.0 Structure

Note: In many of the subsections of section 5.0, the reader will notice repeated use of the symbols P+, P-, C, C-1, and C-2. Please refer to section 6.1 for definitions of P+ and P- (these are the lower 12 bits of each word of the symbol table entry during Pass 1). C denotes the last encountered symbol in the search path before determining that this new symbol must be entered. Then C-1 and C-2 refer to the next to the last symbol and the 2nd from the last symbol, respectively, encountered in the search path. For further clarification of these ciphers refer to subsections 6.3 and 6.4.

5.1 SYMBOL

5.1.1 SYMBOL hashes the 7-character symbol (to be searched for) into a 7-bit pointer. This value is added to the base address of a table (SLIST) to load a word which contains an ordinal of a symbol table entry which is the head of this particular list. Routine SLCOMM is then entered.

5.2 LABEL

5.2.1 LABEL hashes the 6-character statement label (to be searched for) into a 5-bit pointer. This value is added to the base address of a table (LLIST) to load a word which contains an ordinal of a symbol table entry which is the head of this particular list. A jump is then taken to SLCOMM.

5.3 SLCOMM

5.3.1 SLCOMM transfers the symbol (or label) to the specific register (X0). Then if this particular list is empty the symbol is entered in the symbol table, its ordinal is set as the head of the list and the not-found exit is taken. If the list is not empty, SLCOMM sets the head of the list ordinal in B4 and jumps to TOP.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 5.4
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.4 TOP

- 5.4.1 TOP is the main search loop of LSTPROC. After loading the symbol located at the head of the list, the loop is entered to compare the symbol searched for with each symbol already in the list. The comparison is an integer subtraction. If the result is positive the P+ pointer of the current symbol is examined. If P+ is zero, the list is exhausted with no compare, hence the new symbol must be entered so control goes to routine ENTER. If P+ is not zero, the next symbol is loaded and control loops to TOP to compare the next entry of the list. If the result of the subtraction is negative, P- of the current symbol is examined in the same manner as P+. If the result of the subtraction is zero, the symbol being searched for has been found, and control transfers to FOUND.

As the list is being searched, information is collected in the X2 register to be examined by routine ENTER to determine the course of the search through the last few (maximum of 3) comparisons. See section 6.3 for format of X2.

5.5 ENTER

- 5.5.1 Control transfers to ENTER when it is determined (at TOP) that the current symbol is a new symbol and consequently must be entered in the table. Through examination of the X2 register, ENTER makes available in registers the last few symbols (or labels), their ordinals in the symbol table, and the linking pointers (P+ and P-) contained in these entries. The new symbol is then entered in the next available position in the symbol table with its P+ and P- both zero, and the length of the table is increased by two. ENTER also forms a jump switch in X5, from the examination of X2, which will effect transfer of control to the appropriate linkage manipulating routine after all registers are set up.

5.6 SECOND

- 5.6.1 This point is entered when it is determined that the new symbol is the second entry of this list. If the new symbol is greater than the current symbol, control transfers to routine G2. If it is less, control goes to routine L2.

5.7 L2

- 5.7.1 L2 sets P- of the current symbol (C) equal to the ordinal of the new symbol, and transfers control to RPTRN.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 5.5
PRODUCT NAME FORTTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.8 G2

5.8.1 G2 sets P+ of C = the ordinal of the new symbol, and transfers control to RETRN.

5.9 NWENTL

5.9.1 This routine stores P+ of the new symbol, clears P+ of C-1, sets and stores P- of the new symbol = the ordinal of C-1, stores P+ of C-1, and jumps to RETRN.

5.10 CENTL

5.10.1 This routine stores P- of C-1, sets and stores P- of C = the ordinal of the new symbol, sets and stores P+ of C = the ordinal of C-1, and jumps to RETRN.

5.11 NWENTG

5.11.1 This routine stores P+ of the new symbol, sets and stores P- of the new symbol = the ordinal of C, clears P- of C-1, and jumps to RETRN.

5.12 CNETG

5.12.1 This routine stores P+ of C-1, sets and stores P+ of C = the ordinal of the new symbol, sets and stores P- of C = the ordinal of C-1, and jumps to RETRN.

5.13 LNBASN

5.13.1 This routine sets P+ of the new symbol = the ordinal of C. If rearrangement of the list is necessary, the ordinal of the new symbol is stored as the head of the list and control goes to NWENTL. If rearrangement is not necessary (P- of C-1 is not zero), the list head stays the same and control transfers to L2.

5.14 LNBASC

5.14.1 If P+ of C-1 is not zero rearrangement is not necessary and control goes to L2. If rearrangement is necessary the ordinal of C is stored as the new head of the list, P- of C-1 is set to zero, and control goes to CENTL.

5.15 GNBASN

5.15.1 If P+ of C-1 is not zero rearrangement is not necessary and control goes to G2. If rearrangement is necessary the ordinal of new is stored as the new list head, P+ of new is set = the ordinal of C-1, and control goes to NWENTG.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 5.6
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.16 GNBASC

5.16.1 This routine sets P+ of C-1 to zero, and if rearrangement is necessary (P- of C-1 = zero) the ordinal of C is stored as the new list head. Control then goes to CENTG. If rearrangement is not necessary the list head is not changed and control transfers to G2.

5.17 LLR, LRR

5.17.1 If P- of C-1 is not zero no rearrangement is necessary, and control goes to L2. If rearrangement is necessary, P+ of new is set to the ordinal of C and stored, P (P- for LLR, P+ for LRR) of C-2 is set = the ordinal of new and control goes to NWENTL.

5.18 LLL, LRL

5.18.1 If P+ of C-1 is not zero rearrangement is not necessary, control goes to L2. If rearrangement is necessary P (P- for LLL, P+ for LRL) of C-2 is set = the ordinal of C and stored, P- of C-1 is set = zero, and control transfers to CENTL.

5.19 GLR, GRR

5.19.1 If P- of C-1 is not zero no rearrangement is necessary and control goes to G2. If rearrangement is necessary P (P- for GLR, P+ for GRR) of C-2 is set = the ordinal of C and stored, P+ of C-1 is set to zero, and control transfers to CENTG.

5.20 GLL, GRL

5.20.1 If P+ of C-1 is not zero no rearrangement is necessary and control goes to G2. If rearrangement is necessary P (P- for GLL, P+ for GRL) of C-2 is set = the ordinal of the new symbol and stored, P+ of the new symbol is set = the ordinal of C-1, and control transfers to NWENTG.

5.21 RETRN

5.21.1 RETRN is the not-found exit. B5 is set to 1, B2 is set to twice the ordinal of the current symbol (B1+B1), the first word of the entry is loaded into X2, and a jump is taken to the address specified in the B7 register.

5.22 FOUND

5.22.1 FOUND is the found exit. Its function is exactly the same as RETRN except that the jump is taken to the address specified in B7+1.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 5.7
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

6.0 Formats

6.1 PASS 1 SYMBOL TABLE FORMATS

During phase 1, symbol table entries will occupy two words, A and B:

	59		17	16	15	14	13	12	11	0
			F							
A		NAME	P	U	F	C	D	E	P+	

The first word of the entry (A) will have the above format, with the following meanings:

- NAME: (bits 59-18) the name of the symbol or label, 7 (6 if a LABEL) or less display code characters left justified, with blank fill.
- FP: (bit 17) = 1 if the name is a formal parameter or RETURNS.
- U: (bit 16) = set - 1 when symbol becomes defined.
- F: (bit 15) = 1 if the symbol is the name of a function (unless the subprogram being processed is this function).
- C: (bit 14) = 1 if the name appears in a COMMON statement.
- D: (bit 13) = 1 if the name has been dimensioned.
- E: (bit 12) = 1 if the name has been changed due to EQUIVALENCEing. A "base" and "bias" appear in the first word of the DIMENSION LIST entry for the name.
- P+: (bits 0=11) - the ordinal of the symbol table entry, in this list, which is the next greater entry than this entry, or if none exists, P+ = zero.

If NAME is the name of a symbol, word B will have the following format:

	59	56	55	54	53	52	41	40	39	34	19	18	13	12	11	0
			A	E	E			V						B		
B	T		S	X	N	DIMP		N		OP		S	RB	C		P-
			F	T	T											

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 5.8
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

where T (bits 59-56) is the type of the symbol:

0	= LOGICAL	6	= LABEL
1	= INTEGER	7	= RETURNS parameter
2	= REAL	10	= NAMELIST group variable (namelist name)
3	= DOUBLE	11	= PROGRAM, SUBROUTINE, BLOCKDATA
4	= COMPLEX		
5	= ECS variable		

IF PROGRAM: ASF (BIT 55) = 0
 ENT (BIT 53) = 1

IF SUBROUTINE: ASF (BIT 55) = 1
 ENT (BIT 53) = 1

IF BLOCKDATA: ASF (BIT 55) = 0
 ENT (BIT 53) = 0

12 = ENTRY statement name

Note: (1) FUNCTION subprogram name will have type 0-4 and ENT = 1 in word B.

(2) Ordinal 1 of SYMTAB will always be the name of the routine.

ASF (BIT 55) = Statement Function

EXT (BIT 54) = External symbol (variable in an external statement) or implicit function reference, or object of a call.

ENT (BIT 53) = Entry point.

DIMP (BITS 52-41) = INDEX (the ordinal times 2) of the dimension and/or equivalence information if bit 13 and/or bit 12 of word A = 1.

S (BIT 19) = 1 if BSS storage is to be assigned when DPCLOSE is called. (This is for equivalence processing. If this bit is set, DPCLOSE will assign storage for the variable (or array) and then set the C bit in word A so that duplicate storage is not issued when the END statement is processed.)

VN (BIT 40) = 1 if the name has been used as a variable name.

RB (BITS 18-13) = relocation base indicator (see PASS 2 SYMBOL TABLE FORMATS, section 6.2.).

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 5.9
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

BC (BIT 12) = 1 if BIT 14 of word A = 1 and the name is in blank common.

P - (BITS 11-00) = the ordinal of the symbol table entry, in this list, which is the next lesser entry than this entry, or if none exists, P- = zero.

OP (BITS 34-39) of word A = the order of appearance of the formal parameters or the variables in a RETURNS list.

- Note:
- (1) if bit 15 (FUNCTION) of A = 1, then bits 51-46 of B = the number of arguments. (If NAME is a basic external, then Bit 45 = 1; if NAME is an intrinsic function, then Bit 44 = 1).
 - (2) if bit 17 (FORMAL PARAMETER, DUMMY argument) of A = 1, then bits 39-34 of B = the order of appearance of the dummy argument name in the argument list, such that, if the name is the first in the list, bits 39-34 = 0; if the second, bits 39-34 = 1, etc.
 - (3) if type = 7 (RETURNS parameter), then bits 39-34 of B = the ordinal of the parameter in APLIST. The ordinal is numbered consecutively beginning with zero (as in note #2 above).

If NAME is a statement label, then word B has the following format:

59	56	55	54	53	50	49	38	24	23	12	11	0
		G		R	R	R				L	P-	OR
T	"	R	S	S	F	F	A	BLC				
(=6)	O	Z	N	N	M	M	S					

where T and P- are defined as above, and the other fields are defined as follows:

- 1) If the T field is set to 6 (label indication) then a one bit field called G is needed. A G field of 1 indicates that the label has been generated by the DO processor. A G of 0 indicates a normal label.

a. If G = 0 then the following fields are needed:

- RZ = 1 bit field, set if label is referenced prior to current DO next.
- RSN = 1 bit field, set if label is referenced as statement number.
- SN = 1 bit field, set if label has been defined as a statement number.

DOCUMENT CLASS IMS PAGE NO. 5.10
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

FM = 1 bit field, set if label is defined as a format.
RFM = 1 bit field, set if label is referenced as a format.
L = a 12 bit field containing the ordinal in SYMTAB of
the loop in which the label is defined or referenced.
RAS = 1 bit field, set if label referenced in ASSIGN
statement.

b. If G = 1 then the following fields are necessary:

R = 1 bit field, set if all integer variables are
considered to be re-defined within the loop.
E = 1 bit field, set if loop may be entered at a point
other than the top.

X - 1 bit field, set if loop may be exited at a point
other than the terminating statement of the DO.

M - 1 bit field, set if loop control variable must be
materialized (placed in memory).

V - 1 bit field, set if control variable is equal to
incremental limit. Example: DO 10 K = 1, N, K

J - 1 bit field, set if loop contains an external
reference.

I - 1 bit field, set if loop contains another loop.

BLC - 15 bit field, binary line count.

The seven flags for G = 1 must be ordered as E, X, I, M, V, J, R
from left to right in adjacent bits.

2) If T is something other than six then standard fields of the
symbol table will be observed.

6.2 PASS 2 SYMBOL TABLE FORMATS

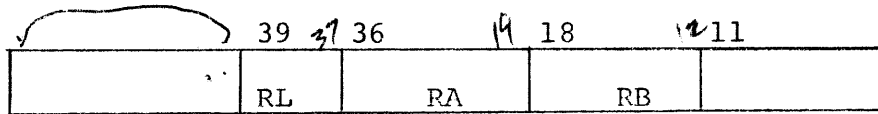
During Pass 2 there is only one word per symbol table entry. The
original bits 59-12 of word A (from Pass 1) occupy bits 59-12 in
the Pass 2 entry. For symbols, the original bits 24-13 of word B
(from Pass 1) occupy bits 11-00 in the Pass 2 entry. For labels,
the original bits 59-48 of word B (from Pass 1) occupy bits 11-00
in the Pass 2 entry.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 5.12
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

During the assembly phase of Pass 2, word B of the symbol table is reformatted to hold:



RL = relocation code:

- 0 = absolute
- 1 = program
- 2 = common
- 3 = external

37-38

RA = relocation address relative to relocation base RL.

RB = relocation base ordinal:

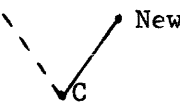
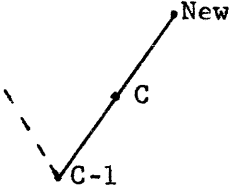
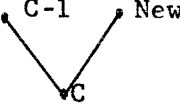
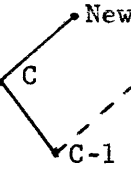

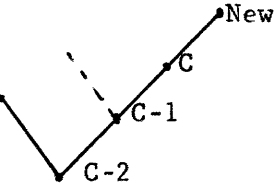
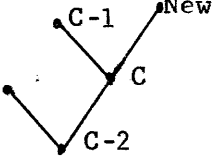
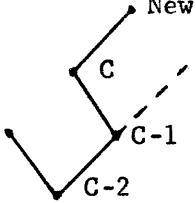
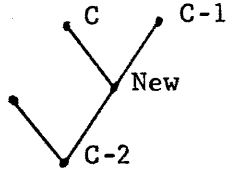
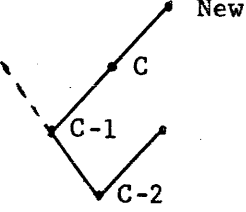
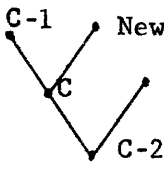
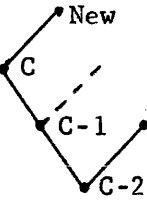
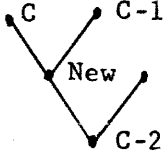
When RL = 1, RB ranges from 0-73 for the maximum number of local blocks.

When RL = 2, RB ranges from 0-59 for the maximum number of common blocks.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 5.12
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600
 6.4 Entry Routines

6.4.1 Entry routines for the possible cases where the new symbol is greater than the symbol at c.

IDENTIFIER	PICTURE	ROUTINE
<u>xab</u>	<u>CURRENT</u>	<u>REARRANGEMENT (if any)</u>
000		NONE
000		
001		
000		
010		
001		
011		

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 5.13
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

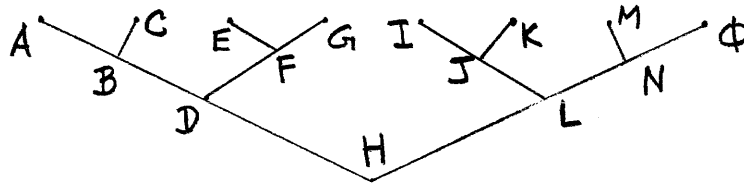
6.4.2 Entry routines for the possible cases where the new symbol is less than the symbol at c.

IDENTIFIER	PICTURE	ROUTINE NAME
<u>xab</u>	<u>CURRENT</u>	<u>REARRANGEMENT (if any)</u>
100		NONE L2
010		 LNBASN
011		 LNBASC
100		 LRR
110		 LRL
101		 LLR
111		 LLL

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 5.14
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- 7.0 When LSTPROC is entered it is assumed that SYM1 (RA+12B) has been set to Field Length - 1. SYMEND (RA+13B) must be initially set to Field Length -1, thereafter LSTPROC updates SYMEND each time a new symbol or label is entered in the symbol table.
- 8.0 In order to find a given symbol in the symbol table (or to determine that the symbol is not yet in the table) with the least number of comparisons, the symbol table is actually broken down into a number of short lists. Each symbol in a list is linked by ordinals to the other symbols in the list, and each time a symbol is added to a list the links are changed such that any symbol within the list can be found with the least number of comparisons. Each symbol contains a pointer (P+) to the next greater symbol in this list and a pointer (P-) to the next lesser symbol in this list, and each list is kept as symmetric from a given starting point as possible. For example, a list reflecting the best possible symmetry could look as follows:



where the symbol H is the start, or head of the list. In this case, where the number of symbols is 15, the maximum number of comparisons to find a given symbol would be four. In this example, symbol H would have a P+ pointer to symbol L and a P- pointer to D. L has a P+ pointer to N and a P- pointer to J, etc. Each of the symbols A, C, E, G, I, K, M, and O would have P+ and P- equal to zero indicating that the particular symbol is the end of this path in this particular list and if a comparison has not been reached by now the symbol being searched for must be entered in the table and linked to the last compared symbol.

Although each list is linked only within itself, this does not mean that the elements of one particular list must be stored consecutively in memory. As each new symbol is encountered it is simply stored in the next available location, and pointers are set up to reflect its location in the table.

Each one of the short lists must have a starting point, or head of the list. Further, we must have a way of determining what list a particular symbol belongs to. This is done by commutatively forming (by the use of shifts and the exclusive OR (logical difference operation) a 7-bit value or a 5-bit value for symbols or labels respectively. This value is an index into one of two local tables (SLIST for symbols, LLIST for labels) which contain symbol table ordinals which point to the head of that list. Initially the SLIST and LLIST tables of list heads are set to zero. If a given cell is loaded that is zero, then we know this is the first symbol in this list and therefore no searching must be done. The symbol is merely

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 5.15
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

entered, set as the head of this list, and a return is made to the caller.

When a list has grown to 3 symbols the head of the list may be changed to point to another symbol of that list in order to maintain symmetry. However, after the list has 3 symbols in it the list head will never again change. Although this method does not guarantee maximum symmetry for a list there is a point where the time required to rearrange the list head throughout the life of the list is greater than the speed gained in subsequent searches.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 6.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

CONVERT

1.0 General Information

1.1 CONVERT converts the display code representation of a constant to its internal binary form. The binary form is placed in a table and the user now refers to the constant by the I, H of the table name and the CA of the location of the constant in the table CON..

2.0 Usage

2.1 CONVERT

2.1.1 Determines which of the three options is desired.

2.1.1.1 The constant is converted to binary form, placed in CONLIST, if not already there and the caller informed of I, H and CA to be used to reference the constant. Conversion and add to CONLIST.

2.1.1.2 The constant is converted to binary form and returned to the caller. Conversion only.

2.1.1.3 The constant in the form supplied by the caller is placed in CONLIST if not already there and the caller informed of I, H and CA. Add to CONLIST only.

2.1.2 Calling Sequence and Returns.

The calling sequence is RJ CONVERT. Case 2.1.1.1 expects register B1 to be +0 and the E-list entry for the constant to be in register X1. Upon successful return register X1 holds H in bits 0=17, I in bits 18-29 and CA in bits 30-47 all other bits being 0.

Case 2.1.1.2 expects register B1 to be negative and the E-list for the constant to be in register X1. Upon return, the converted form of the constant is held in X1, and X2 if the constant is a two word element.

Case 2.1.1.3 expects register B1 to hold 1 or 2 the number of words in the caller supplied constant and X1 and X2 to hold the one or two word element, X1 the first part of the constant and X2 the second part.

2.1.3 Processing Flow Description

CONVERT quickly determines the option desired. The first call for either case 2.1.1.1 or 2.1.1.3 will cause the symbolic name CON. to be placed in the symbol table. For Case 2.1.1.1, the display code of the constant is formed for DEC, DEC is called to convert the constant, then CONLIST is searched for the converted form of the constant. If the constant already appears the I, H and CA is returned to the caller. Otherwise, the constant is placed in CONLIST. For Case 2.1.1.2, DEC is called to convert the constant. DISPLAY is the routine called to process all forms

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 6.2
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

of Hollerith constants. Case 2.1.1.2 the first word of the constant is returned in X1. For cases 2.1.1.1 and 2.1.1.3, the constant is put in the COMPASS file following a USE HOL.. The constant instruction is a DISI, except for the last word which occupies a VFD 60/nH or nL or nR depending upon the form of the Hollerith constant. The first call to DISPLAY will cause the symbolic name HOL. to be placed in the symbol table.

3.0 Diagnostics Produced

3.1 Fatal to Compilation

3.1.1 CONLIST TOO BIG. TOO MANY CONSTANTS. MORE MEMORY REQUIRED.

3.2 Fatal to Execution

3.2.1 CONSTANT CONVERSION ERROR.

4.0 Environment

CONVERT expect CONI (RA+26B), DOI (RA+30B), DOLAST (RA+31B) and ELAST (RA+14B) to be set prior to being called. CONVERT maintains CONI and CONLAST the first and last locations used by CONLIST. 100 locations are initially reserved for CONLIST. If more room is required the DO tables are moved 100 locations, if possible, and the pointers maintained. When 100 more locations are not available ((ELAST) being the highest+1 address that can be used) a fatal to compilation diagnostic is issued via FATALER.

5.0 Structure

5.1 CONVERT determines if the option is "store only" and if so, jumps to PACK. If not, a check is made for the constant being any form of Hollerith and if so, a jump is made to DISPLAY. For the "convert only" option, a jump is made to PRECON to arrange the input to DEC. For the "convert and store" option, a jump is made to PRECON, then PUT.

5.2 PACK determines the first call for a store and calls SYMBOL to put the name CON. in the symbol table and retain its ordinal for use as the H field in the I, H and CA information.

5.3 PRECON arranges the display code of the constant as follows: digits are packed a maximum of seven per word left adjusted to bit 59 and zero filled, +-, or B are stored in bits 0-5 with zero fill, and E or D are stored in bits 54-59 with zero fill.

5.4 PUT places the one or two word converted constant (or caller supplied constant) into CONLIST if the constant is not already in CONLIST. Initially 100 locations are reserved for CONLIST and will be expanded 100 locations at a time moving the DO tables if necessary until the

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 6.3
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

time when 100 locations are not available (CONLIST) or the DO tables running over ELAST when a fatal to compilation diagnostic is issued.

- 5.5 DEC does the actual conversion.
- 5.6 DISPLAY determines the first call for a storing option and calls SYMBOL to place the name HOL. in the symbol table and retain the ordinal to use as H in the I, H and CA information. For the convert only option, the first word of the Hollerith constant is returned to the caller in register X1. For any storing option, the constant is placed in the COMPASS file and the user returned the I, H and CA information. Any ten character part of the constant is packed following a DIS I, and any part of the constant is packed following a VFD 60/n H or L or R with n being the number of characters. The first Hollerith constant put in the COMPASS file will have HOL. in the label field. A DATA instruction is put in the COMPASS file to terminate each constant with a word of zeros. Finally a USE DATA. is put in the COMPASS file.
- 6.0 Formats
- 6.1 I, H and CA word returned to the caller is VFD 12/0, 18/CA, 12/I, 18/H.
- 6.2 CONLIST is the name of the table of converted or user supplied constants. .TRUE. is converted to -1 and .FALSE. is converted as +0.
- 7.0 Modification facilities.
The coding is straight forward

DOCUMENT CLASS IMS PAGE NO. 7.1
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

ERPR0 and F0RMAT

1.0 General Information

1.1 Task Description - Error Processing

The error processing during compilation is divided into two routines: ERPR0 in PASS 1, and PS2CTL in PASS 2. The final output resulting from detecting an error will be the card sequence number {compiler generated} on which the error occurred, the severity {fatal to execution, etc}, an optional symbol, a name or word to further clarify the message, and the actual diagnostic {up to 106 characters}. ERPR0 is located in the 1.0 overlay.

1.2 Task Description - F0RMAT Scanner

The F0RMAT scanner processes F0RMAT statements and checks for errors at compilation time. The scanner squeezes out blanks and redundant commas. Before scanning the F0RMAT for validity, the statement label is checked for validity, recorded in the symbol table, and sent to the COMPS file.

2.0 Usage

2.1 Entry Point Names - Error Processing

In general the calling sequence is:

```
SB6 error number  
Sb7 return  
EQ entry point
```

Using the above calling sequence, ERPR0 would expect an E-LIST entry in X4, or if X4 is zero a display code message in X3 {bits 48-59 zero, bits 0-47 display code}.

Using the following calling sequence, no message is expected in X3 or X4.

```
SB6 -error number  
Sb7 return  
EQ entry point
```

The parameter in B6 is a symbol or number which is equated to the ordinal of the error in the error directory table in PASS 2.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 7.2
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- 2.1.2 ERPR0 This entry is used for all messages which resulted from errors which are fatal to execution.
- 2.1.3 ASAER This entry is used for all messages which denote non-USASI usage.
- 2.1.3.1 A reference to this entry point will check a flag called NASAFL and if the flag is non-zero, the message will be entered into the error table, otherwise, an immediate exit is taken.
- 2.1.4 FATALER This entry is used for all messages which resulted from errors which are fatal compilation.
- 2.1.4.1 A reference to this entry will result in making an entry in the error table and setting the fatal to compilation flag {FX}. Control does not return to the caller. Calls to SCANNER are then made until an END card is encountered, then a request to load PASS 2 is made.
- 2.1.5 ERPR0 I This entry point is used for all diagnostics which are informative in nature.
- 2.1.5.1 Informative diagnostics up until {10 minus the maximum} are placed in the table and at that time an informative diagnostic is issued stating that no more informative diagnostics will be put in the table.
- 2.2 FORMAT has one entry point.
- 2.2.1 Upon entry with a legal statement label, FORMAT scanning takes place. FORMAT is entered by both PH1CTL and PH1CTL, since formats may be among both executable and non-executable statements.
- 2.2.2 Calling Sequence
- FORMAT is entered via a return jump and upon completion of its tasks, exits through its entry point.
- 2.2.3 Flow of Processing
- The characters which comprise a FORMAT, beginning with the left parenthesis, are scanned sequentially until the matching right parenthesis or an irrecoverable error condition is encountered.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 7.3
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

3.0 Diagnostics

3.1 Error Processing

Number 206 Informative
DUE TO THE NUMEROUS ERRORS NOTED, ONLY THOSE WHICH ARE
FATAL TO EXECUTION WILL BE LISTED BEYOND THIS POINT
Number 110 Fatal to Compilation
ERROR TABLE OVERFLOW

3.2 FORMAT

3.2.1 Fatal to Execution

PRECEDING CHARACTER ILLEGAL AT THIS POINT IN CHARACTER
STRING. ERROR SCAN FOR THIS FORMAT STOPS HERE.

ILLEGAL CHARACTER FOLLOWS PRECEDING FLOATING POINT
DESCRIPTOR. ERROR SCAN FOR THIS FORMAT STOPS HERE.

ILLEGAL CHARACTER FOLLOWS PRECEDING A, I, L, O, OR R
DESCRIPTOR. ERROR SCAN FOR THIS FORMAT STOPS HERE.

ILLEGAL CHARACTER FOLLOWS TAB SETTING DESIGNATOR. ERROR
SCAN FOR THIS FORMAT STOPS HERE.

ILLEGAL CHARACTER FOLLOWS PRECEDING SIGN CHARACTER.
ERROR SCANNING FOR THIS FORMAT STOPS HERE.

PRECEDING CHARACTER ILLEGAL, SCALE FACTOR EXPECTED.
ERROR SCANNING FOR THIS FORMAT STOPS HERE.

PRECEDING HOLLERITH COUNT IS EQUAL TO ZERO. ERROR
SCANNING FOR THIS FORMAT STOPS HERE.

FORMAT STATEMENT ENDS BEFORE LAST HOLLERITH COUNT IS
COMPLETE. ERROR SCAN FOR THIS FORMAT STOPS AT H.

FORMAT STATEMENT ENDS BEFORE END OF HOLLERITH STRING.
ERROR SCANNING STOPS HERE.

PRECEDING HOLLERITH INDICATOR IS NOT PRECEDED BY A COUNT.
ERROR SCANNING STOPS HERE WITH FORMAT INCOMPLETE.

ZERO LEVEL RIGHT PARENTHESIS MISSING. SCANNING CONTINUES.

PRECEDING FIELD WIDTH OUTSIDE OUTER LIMITS FOR RECORD
SIZE. SCANNING CONTINUES.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 7.4
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

PRECEDING RECORD OUTSIDE OUTER LIMITS FOR RECORD SIZE.
SCANNING CONTINUES.

TAB SETTING IS OUTSIDE OUTER LIMITS FOR RECORD LENGTH.
SCANNING CONTINUES.

3.2.2 Non-USASI

PLUS SIGN IS AN ILLEGAL CHARACTER.

PRECEDING FIELD DESCRIPTOR IS NON-USASI.

FLOATING POINT DESCRIPTOR EXPECTED FOLLOWING SCALE FACTOR
DESIGNATOR.

TAB SETTING DESIGNATOR IS NON-USASI.

HOLLERITH STRING DELINEATED BY SYMBOLS IS NON-USASI.

3.2.3 Informative

SEPARATOR MISSING, SEPARATOR ASSUMED HERE.

X-FIELD PRECEDED BY A BLANK, 1X ASSUMED.

X-FIELD PRECEDED BY A ZERO, NO SPACING OCCURS.

PRECEDING FIELD WIDTH IS ZERO.

PRECEDING FIELD WIDTH SHOULD BE 7 OR MORE.

FLOATING POINT DESCRIPTOR EXPECTS DECIMAL POINT SPECIFIED.
OUTPUT WILL INCLUDE NO FRACTIONAL PARTS.

FLOATING POINT SPECIFICATION EXPECTS DECIMAL DIGITS TO BE
SPECIFIED. ZERO DECIMAL DIGITS ASSUMED.

REPEAT COUNT FOR PRECEDING FIELD DESCRIPTOR IS ZERO.

FIELD WIDTH IS OUTSIDE INNER LIMITS. CHECK USE OF THIS
FORMAT TO ASSURE DEVICE CAN HANDLE THIS RECORD SIZE.

PRECEDING SCALE FACTOR IS OUTSIDE LIMITS OF REPRESENTATION
WITHIN THE MACHINE.

SUPERFLUOUS SCALE FACTOR ENCOUNTERED PRECEDING CURRENT
SCALE FACTOR.

RECORD SIZE OUTSIDE INNER LIMITS. CHECK USE OF THIS FORMAT
TO ASSURE DEVICE CAN HANDLE THIS RECORD SIZE.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 7.5
PRODUCT NAME FORTAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

FIELD WIDTH OF PRECEDING FLOATING POINT DESCRIPTOR SHOULD BE 7 OR MORE THAN DECIMAL DIGITS SPECIFIED.

NUMERIC FIELD FOLLOWING TAB SETTING DESIGNATOR IS EQUAL TO ZERO, COLUMN ONE IS ASSUMED.

NUMERIC FIELD OMITTED IN PRECEDING SCALE FACTOR. ZERO SCALE ASSUMED.

NON-BLANK CHARACTERS FOLLOW ZERO-LEVEL RIGHT PARENTHESIS. THESE CHARACTERS WILL BE IGNORED.

TAB SETTING MAY EXCEED RECORD SIZE DEPENDING ON USE.

3.2.4 Each error message will be preceded by a 48 bit message stating the card and column number of the error encountered. Computation and the form of this message is described in Section 8.

4.0 Environment

4.1 Error Processing

4.1.1 Information provided by other processors.

4.1.1.1 In location 46B and 47B, SCANNER places information regarding the current card number. Location 46B contains in display code the line number as printed on the listing, location 47B contains in binary an offset count which ranges from 1 to 10.

4.1.1.2 The location of the error table is an entry point name in FTN called ERTABL.

4.1.1.3 NASAFL is contained in location 42B and is set by the control card cracker.

4.1.2 Information generated by ERPR0.

4.1.2.1 Location 40B contains the number of errors in binary encountered during a single compilation.

4.2 FORMAT

FORMAT scanner expects character to be packed ten characters per word in display code, where the first character is a left parenthesis. FORMAT expects the first word of information at the location specified by SELIST, and the last word of information at the location specified by

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP

 DIVISION

DOCUMENT CLASS IMS PAGE NO. 7.6
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

ELAST. FORMAT allows a maximum of three levels of parentheses, an input record length of 150 characters, and an output record length of 136 characters. In general, formats must be in accordance with USASI FORTRAN standards, with the addition of the tab setting and Hollerith string capabilities. Legitimate format field descriptors are of the following form:

$\{ \{ + | - \} \{ n \} P \} \{ r \} \langle D | E | F | G \rangle w \cdot d$

$\{ r \} \langle A | I | L | O | R \rangle w$

$n H h_1 h_2 \dots h_n$

$\{ n \} X$

$\# \dots \#$

$T m$

where:

1. In the above description a vertical bar separates alternatives; angle brackets denote that one and only one of the enclosed alternatives must be chosen; square brackets denote that none or one of the enclosed alternatives may be chosen.
2. The letters D, E, F, G, A, I, L, O, R, H, and X indicate the manner of conversion or editing between the internal and external representations and are called the conversion codes.
3. w and n are integer constants representing the width of the field in the external character string.
4. d is an integer constant representing the number of digits in the fractional part of the external character string. If d is omitted it is assumed to be zero.
5. m is an integer constant representing the tab setting for the external character string.
6. r is the repeat count {an optional, non-zero integer constant} indicating the number of times to repeat the succeeding basic field descriptor.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 7.7
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

7. $\{+|- \} \{n\}$ P is optional and represents a scale factor to be applied to the processing of the succeeding conversion code if a $D|E|F|G$.
8. Each h_i is one of the characters capable of representation by the processor.
9. $*...*$ encloses hollerith information [excluding an asterisk], up to one record in length.
10. For all descriptors other than $*...*$, field width must be specified; for descriptors of the form $\langle D|E|F|G \rangle w.d$, w must be greater than or equal to $d+7$.

Format field separators are the slash and the comma. Field separators are used to delimit field descriptors. Field separators are optional in the following cases:

1. after $*...*$
2. after $nHh_1h_2...h_n$
3. after nX
4. after $\{+|- \} \{n\}$ P
5. after another field separator
6. before or after a right parenthesis

In all other cases a field separator is expected, and a diagnostic is issued if the separator is missing. Scanning of the format will continue in such a case. Blanks and commas, where unnecessary, are squeezed out of the format specification.

5.0 Structure

5.1 Major subroutine names in ERPR0.

- 5.1.1 ERPR0. This subroutine checks if room exists in the table and determines type of parameter that accompanies the message.
- 5.1.2 OPER. This subroutine decodes the E-list element.
- 5.1.3 TAB0FL0. This subroutine issues diagnostic 110 and makes the calls to SCANNER.
- 5.1.4 PK. This subroutine sets up the entries in the error table and updates the cell {ERL0C} which contains the address of the next available cell in the error table.

CONTROL DATA CORPORATION . COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 7.8
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.2 FORMAT

5.2.1 Transition Diagram

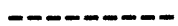
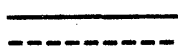
Format scanner has been implemented utilizing transition diagram oriented processing. A transition diagram describes action to be taken for each syntactic type encountered in a string. The transition diagram consists of circles, boxes, unbroken, and broken line segments where:



::= a NODE, or state in the flow which has been reached at some point in the string.



::= a set of intermediate processing on the string between nodes, or states, which can be made analogous to a FORTRAN subroutine.



::= action in processing the string. Over a solid line segment, character advancement takes place; over a broken line segment, character advancement does not take place. The character(s), or group of characters (i.e. digit ::= (0,1,2,3,4,5,6,7,8,9)) which direct the processing to a particular state are inscribed on the line segment.

Character advancement can also occur in intermediate processing.

The transition diagram which traces the flow of processing for the format scanner follows.

DOCUMENT CLASS IMS PAGE NO. 7.9
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.2.2 Micro Definitions for Transition Diagram

Micro definitions for the format transition diagram are formed in the following manner:

```
node micro ::= branch{branch} [otherb] | otherb
branch ::= (char test|mask test) , [ignore] : transfer designation
otherb ::= /{char3}:transfer designation
transfer designation ::= node name [, routine name [ (param) ]]
char test ::= <|=|<|<> (char1|expr1)
char1 ::= <A|B|C|...|8|9|+|-|*|/|$|,|.|.|(|)|[|]|<|<|<|>|=|&|v|↓|↑|:|~>
expr1 ::= compass expression designated by more than one character.
mask test ::= [~][< [|] {char2}|{char2}1>]
char2 ::= <A|B|C|...|8|9|+|-|*|/|$|,|.|.|(|)|[|]|<|<|<|>|=|&|v|↓|↑|:|~>
ignore ::= ,{char3}
char3 ::= <A|B|C|...|8|9|+|-|*|/|$|,|.|.|(|)|[|]|<|<|<|>|=|&|v|↓|↑|~>
node name ::= name
routine name ::= name
name ::= letter {letter|number|$|.}07
letter ::= <A|B|C|...|Z>
number ::= <0|1|2|3|4|5|6|7|8|9>
param ::= compass expression
```

For example, at node 7:

```
= ,NOPACK:NODE7 - a blank is not packed, the flow is advanced
one character and sent back to node 7.
[0123456789]:NODE1,DECIM - a digit is packed, the flow is
advanced one character, and sent to NODE1,
via a set of intermediate processing, DECIM.
/ELSE:NODE1,IERROR(7) - any other character at this node
inhibits character advancement, and flow is
sent to NODE1, via IERROR, the informative
error processor, with a parameter of 7.
```

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 7.10
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.2.3 Table Formed from Definitions and Table Processor

The micro definitions generate one word table entries, which are acted upon by the transition diagram table processor, TRANSIT, all of which is located in FLY. TRANSIT processes the character string along the path defined by the micro definitions of the transition diagram, fetching and storing characters where required.

5.2.4 Intermediate Processing, i.e Subroutines Used

5.2.4.1 NUMBER

Converts a string of display code numerical digits into a binary number which is stored relative to location NUMN, with a displacement of the input parameter {-1,0,1}; the input parameter specifies the number to be decimal digits, a repeat count or skip span, or a field width. Control is returned to the address specified.

5.2.4.2 RANGE

Checks for valid result of NUMBER routine; range to be checked is specified via the calling parameter. If number is out of range, the error processor is called. Control is returned to the address specified.

5.2.4.3 FLDCHEK

Checks range of field elements; computes total field length and checks the range; record length is increased by the length of the total field. Record count is saved in a pushdown table which saves information for the 3 levels of parentheses. If the record count is longer than one record, an informative error is produced. Control is returned to the address specified.

5.2.4.4 WIDTH

Field descriptor width handler; calls NUMBER{0}, RANGE{1}, and FLDCHEK{1}. Parameter {0} implies a floating point descriptor, and if the field width is not 7 or greater, an informative error is produced. Parameter {1} for other descriptors, and no test is made. Control is returned to the address specified.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 7.11
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.2.4.5 DECIM

Handles decimal digits portion of floating point descriptors; calls NUMBER[-1], and if descriptor is D, E, or G, a check is made for field width greater than or equal to 7 + decimal digits specified. If the descriptor fails this test, an informative error is produced. Control is returned to the address specified.

5.2.4.6 FLAGW7D

Called to turn on a flag indicating a D, E, or G type field descriptor. The flag is utilized by DECIM to determine whether or not to perform a test comparing field width with decimal digits specified. Control is returned to the address specified.

5.2.4.7 ONECNT

Initializes temporary count storage for repeat count, field width, and decimal digits, and turns off flag indicating a D, E, or G specification encountered. Control is returned to the address specified.

5.2.4.8 DELCOM

The last character stored in the string is fetched. If the character was a comma, it is squeezed out of the output string. Control is returned to the address specified.

5.2.4.9 XBLANK

An X descriptor was preceded by a blank, and an informative error is issued to that effect. FLDCHEK is then called to update the record length count. Control is returned to the address specified.

5.2.4.10 XZERO

The skip count is tested for zero; if so, an informative error is issued. If the count is non-zero, FLDCHEK is called to update the record length count. Control is returned to the address specified.

5.2.4.11 TSASI

DELCOM is called to squeeze out redundant commas. A non-ASA error is produced, and control is returned to the address specified.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 7.12
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.2.4.12 TCODE

NUMBER{0} is called to convert the tab setting pointer to binary. If the result is zero, an informative error is produced. Otherwise RECCHEK{1} is called, where the record count is accordingly checked and modified. Control is returned to the address specified.

5.2.4.13 SCALE

NUMBER{0} is called to convert scale factor to binary; then RANGE{-1} is called to check for validity of scale factor. Control is returned to the address specified.

5.2.4.14 NULLP

An informative error is initiated and zero scaling is assumed. The scale flag is turned on; if previously on, and unused, another informative error is produced. Control is returned to the address specified.

5.2.4.15 HCOUNTR

The Hollerith count is fetched, each character is checked against an end-of-statement; if an end of statement is encountered, an error exit is taken. Otherwise the character is stored, the count decremented, and the loop continued until the count is depleted to zero. FLDCHEK is then called to add to the record count. Control is returned to the address specified.

5.2.4.16 HSTRNGR

Each character is compared with the end of statement and the Hollerith string indicator. While no match is made, character advancement continues. If an end of statement is encountered, an error exit is taken. When a matching Hollerith indicator is encountered the character count is sent to FLDCHEK where it is added to the record count. Control is returned to the address specified.

5.2.4.17 SLASH

DELCOM is called when the input parameter indicates, and RECCHEK{0} is called to check for legal record size. Values are checked and modified in a pushdown table which saves record size information for each parenthesis level. Current record count is reinitialized. Control is returned to the address specified.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 7.13
PRODUCT NAME FORTTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.2.4.18 RECCHK

Current record count is checked for legal record size. If entry was from SLASH, control is then returned to the address specified. If entry was from RITEPAR because of a first level right parenthesis, control is sent to FINISH where the format is sent to the COMPS file. Otherwise, entry was from TCODE, and the current record count is set to the tab setting. The record saving pushdown table is modified, and control is returned to the address specified.

5.2.4.19 LEFTPAR

The parenthesis level is incremented and checked for validity. An invalid parenthesis level causes an error exit to be taken. If the parenthesis level is valid, the level repeat count is preserved in the pushdown table. Control is returned to the address specified.

5.2.4.20 RITEPAR

DELCOM is called to delete redundant commas where appropriate. Parenthesis level is checked for zero level. If so, RECCHK{-1} is called, and control is sent to close out procedures. Otherwise, appropriate record size updating is performed on the pushdown table. The parenthesis level is decremented by one, and control is returned to the address specified.

5.2.4.21 FINISH

Control is received by scan when a zero-level right parenthesis is encountered. A check is made for extraneous characters. The last word of the format is packed. If no fatal errors were encountered in the process of scanning, the E-LIST string is inverted and 6 word blocks of COMPASS images are sent to the COMPS file. Entry conditions are restored, and control is returned via a jump to FORMAT.

5.2.4.22 IERROR, UERROR, FERROR

All are entries to the error processing routine, depending upon the type of error incurred. The type is preserved, along with the error number. All critical registers are saved; then the card number and column number in which the

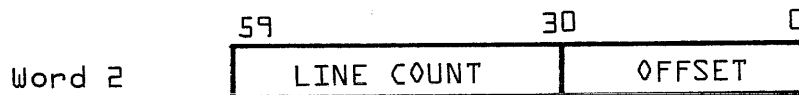
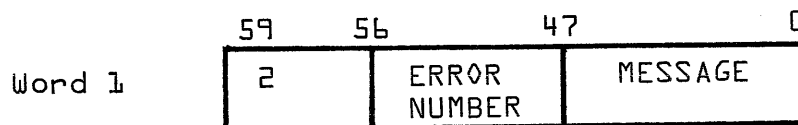
CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 7.14
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

error occurred are computed and merged into the 48 bit message word. Control is then released to the appropriate error processor. On return, the critical registers are restored, and control is returned to the address specified by the caller.

6.0 Table Formats

6.1 Error Table Format



6.2 FORMAT

6.2.1 Memory Pointers and Flags

- DEGFLAG - flag turned on when D, E, or G descriptor is encountered; is used to determine when field width adequacy tests should be made.
- COLCNT - contains count for current record length; is checked in RECCHEK.
- FLAGPON - flag turned on when scale factor is encountered; turned off when utilized. Checked each time scale factor encountered.
- FE - flag turned on when a fatal error condition has been encountered in a format. This flag inhibits packing the format for the COMPS file.
- LEVEL - a counter which keeps track of the parenthesis level, where the first level is level zero.
- NUMD - location which saves the decimal field of floating point descriptors.
- NUMN - location which saves tab settings, and repeat counters.
- NUMW - location which saves the width field of format descriptors.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 7.15
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

PUSHDOWN - a table which contains four fields of information per word, one word per parenthesis level. The information is used to calculate accumulated record length when an end of record is encountered. For each parenthesis level, the following information is saved:

SL indication of presence or absence of slash in level

GR the group repeat count

NL column count following last slash in level

N1 column count preceding first slash in level

The table will be structured as follows:

59	53	35	17	0
SL ₀	GP ₀	NL ₀	N1 ₀	
SL ₁	GP ₁	NL ₁	N1 ₁	

SL _{MAX}	GP _{MAX}	NL _{MAX}	N1 _{MAX}
-------------------	-------------------	-------------------	-------------------

7.0 Modification Facilities

7.1 Error Processing

ERRMAX is an EQU in FTN, controls the size of the error table.

7.2 FORMAT

7.2.1 EQU's

MAXMAX EQU 150 maximum read record length
 MINMAX EQU 136 maximum written record length
 PMAX EQU 615 maximum size scale factor
 LEVMAX EQU 2 maximum parenthesis level
 HOLLER EQU 1R# hollerith string indicator

These limits may all be changed by simply modifying the EQU's.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS TMS PAGE NO. 7.16
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

7.2.2 Allowable Formats

Additions and/or changes to the forms allowable for format descriptors may be made by adding to and/or changing the micro definitions in FLY, and/or adding to and/or modifying the specific subroutine handler{s} involved.

7.2.3 Character Manipulation

Characters are fetched and stored using two macros: GETCH and PUTCH, from words packed ten characters per word to words packed ten characters per word, with a one character delay, SAVECHAR, on storage. These macros may be modified without disturbing the rest of the logic of the scanner.

8.0 Method Used

Format scanner is a left-to-right, character by character, one pass scan, implemented through TRANSIT, the main routine in FLY, which sends the format to the part of code indicated appropriate by the transition diagram. The approved format is packed, ten characters per word, and sent six words per line, to the COMPS file. The scan operates on a character recognition basis. Recognition causes control to be sent to an appropriate set of intermediate processing, which expects a particular combination of characters, previously referred to as field descriptors. Permissible descriptors are itemized in Section 4. At the end of a set of intermediate processing, control is returned to the appropriate state in the flow of the scanner. Scanning terminates when an end of statement is encountered, or an illegal character or character sequence is encountered. A running count is kept of the length, in characters, of the current record described by the format. Calculation of total record length involves utilization of the information stored in the PUSHDOWN table described in Section 6. Calculation and checking is done whenever a slash or a zero-level right parenthesis is encountered. When an error is encountered in the scanning process, the error processor is called, where the card and column number in which the error occurred is calculated. They are computed using the following formulae:

$$\begin{aligned} CD &= 21 - \{CONTS\} \\ COL &= \{COLS - 8\} \end{aligned} \text{ where } \{CONTS\} \text{ and } \{COLS\} \text{ are computed in scanner}$$

$$N = \left\{ \text{fwa format - current address} \right\} * 10 + \{b0 - \{b * \text{character pointer}\}\} / b$$

$$= \text{CURRENT COLUMN POINTER}$$

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 7.17
 PRODUCT NAME FORTAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

$WD = \{COL+N-1\} =$ RELATIVE WORD POINTER

$CDNO = CD+WD =$ CURRENT CARD POINTER

$COLNO = COL+N+5-\{66*WD\} =$ CURRENT COLUMN OF CURRENT CARD
POINTER

This information is packed in the lower 48 bits of the error word in one of the following forms:

NNCDNNNN
 NNCDbNNN
 NNbCDbNN
 NNbbCDbN

where the first field is the column number and the second field is the card number. This information is then sent to the standard error processing routine.

9.0 Restrictions and Other Remarks

9.1 ERPR0

None

9.2 FORMAT

9.2.1 Register Usage

Caution must be taken by the modifier of FORMAT scanner with respect to register usage. The following registers are used by TRANSIT, and must be preserved in FORMAT scanner:

B1=1	A0=mask base	X0=?? ——— 7700B
B2=shift input	A1=input address	X1=input word
B3=node address		X2=input character
B4=return address		X3=subroutine parameter
B7=shift output	A7=output address	X7=output word

Caution must also be taken with respect to TRANSIT utilization of scratch registers. The following registers are used as scratch registers by TRANSIT:

	A3	
	A4	X4
B5		X5
	A6	X6

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 7.18
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

The return mechanism in all cases is via register B4. All intermediate processors save and restore B4 when it is utilized before a return.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 8.1
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

DATA

1.0 General

1.1 DATA is called by PH2CTL whenever SCANNER encounters a DATA statement. DATA send COMPASS pseudo-ops directly to the COMPS file to preset data into variables or arrays. Referenced by DATA are SYMBOL, CONVERT, WRWDS, ERPRO and ASAER.

2.0 Usages

2.1 Entry point name: DATA

2.1.1 DATA processes DATA statements.

2.1.2 PH2CTL enters DATA via a return jump and upon completion returns control to PH2CTL by exiting through its entry point.

2.1.3 DATA translates the FORTRAN statement into COMPASS pseudo-ops.

3.0 Diagnostics

3.1 Fatal to Compilation - none.

3.2 Fatal to Execution

- 1) ILLEGAL IDENTIFIER IN VARIABLE LIST OF DATA STATEMENT. (#112).
- 2) VARIABLE APPEARING IN DATA STATEMENT MAY NOT BE IN BLANK COMMON. (#113).
- 3) VARIABLE APPEARING IN DATA STATEMENT MAY NOT BE A FORMAL PARAMETER. (#114).
- 4) VARIABLE APPEARING IN DATA STATEMENT MAY NOT BE A FUNCTION NAME. (#115).
- 5) ILLEGALLY TYPED VARIABLE IN DATA STATEMENT. MUST BE ONLY INTEGER, REAL, DOUBLE, COMPLEX, OR LOGICAL. (#116).
- 6) ILLEGAL FORMAT OF DATA STATEMENT. (#117).
- 7) ALL ITEMS IN DATA LIST OF DATA STATEMENT MUST BE CONSTANT. (#118).

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 8.2
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- 8) REPEAT FACTOR OF DATA ITEMS AND DO LIMITS MUST BE INTEGER. (#119).
- 9) CONSTANT SUBSCRIPT OF VARIABLE MUST BE INTEGER. (#120).
- 10) NO TERMINATING RIGHT PARENTHESIS AFTER SUBSCRIPT OR DO VARIABLES. (#121).
- 11) DO CONTROL VARIABLES NOT USED AS SUBSCRIPT IN DATA STATEMENT. (#122).
- 12) NO EQUALS SIGN AFTER DO VARIABLE IN DATA STATEMENT. (#123).
- 13) IMPLIED DO LOOP MAY HAVE ONLY 3 LIMITS. (#124).
- 14) VARIABLE APPEARING AS SUBSCRIPT BUT ITS DO LIMITS WERE NEVER DEFINED. (#125).
- 15) NON DIMENSIONED IDENTIFIER APPEARS IN DATA STATEMENT WITH SUBSCRIPTS. (#127).
- 16) UNMATCHED PARENTHESIS IN DATA STATEMENT. (#128).
- 17) ILLEGAL CHARACTER AFTER DATA ITEM. MUST BE COMMA, SLASH, OR RIGHT PAREN. (#129).
- 18) ONLY COMMA OR END OF STATEMENT MAY FOLLOW TERMINATING SLASH OR). (#132).
- 19) SLASH, EQUAL SIGN, OR LEFT PAREN MUST FOLLOW VARIABLE LIST. (#133).

3.3 Information Diagnostic

- 1) MORE DATA ITEMS APPEAR IN DATA LIST THAN ARRAY CAN CONTAIN. EXCESS ITEMS ARE DISCARDED. (#126).

3.4 Non-USASI Diagnostic

- 1) NON-USASI FORM OF DATA STATEMENT. (#129).

4.0 Environment

DATA expects that SCANNER has cracked the statement into E-list format and has set the address of the first element after the word DATA into SELIST (RA+32B). The last element of the statement must be followed by an end of statement. The DIMENSION statement processor must have set the address of the dimension table in DIM1 (RA+17B).

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 8.3
PRODUCT NAME FORTTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.0 Structure

5.1 CONVERG

5.1.1 Sends the COMPASS images "ORG name" to COMPS file. The variable name is in VARBL. If the data is to be stored at a point different from zero of the array, then this index is computed. ORGPNT contains a binary number if prior data has been stored in this array using the "REPI" pseudo-op. If the variable has been equivalenced and a bias introduced, this bias is contained in EQUBIAS. N1, N2, and N3 contain the subscript -1 for the first, second and third subscripts, respectively. If the data is to be stored at a point different from the beginning of the array, the index is converted to display code and suffixed to the array name, the COMPASS line image being "ORG name + index".

5.2 STARRED

5.2.1 Checks the data for a repeated list. The list enclosed in parenthesis or an item alone must be preceded by an asterisk. The repeat count in display code is saved in the upper 30 bits of REPEAT and the binary count is saved in the lower 18 bits of REPEAT. The E-list address of the first data item to be repeated is saved in AGAIN.

5.3 LEAD

5.3.1 Checks the data for logical, Hollerith, complex, double and other.

- a) Logical is represented by false as DATA 0B and true as -1B.
- b) Hollerith constants have all but the last word sent to the COMPS file with the DIS 1, pseudo-op. The last word is sent as VFD 60/n1 where n is the number of characters remaining and l is L, R, or H.
- c) Real, integer, complex and double precision data are converted to internal binary via CONVERT, then to display code and suffixed with a B; thus being issued as octal constants.

5.4 STRING

5.4.1 Alters the repeat count of a data list if the data list exceeds the array length.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 8.4
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.5 FLOW

5.5.1 Checks for data exceeding array length and whether or not there is more data in the list.

5.6 EXIT

5.6.1 Sends COMPASS buffer to COMPS file.

5.7 CLEAR

5.7.1 Clears all temporary flags after a variable and its data list is completely processed.

5.8 LAST

5.8.1 Processes the subscripts of the array name.

- a) Constant subscripts in binary are saved in N1, N2, N3 in their respective order, i.e., an array A(2, J, 3) would have 2, the first subscript, saved in N1, and 3, the third subscript, saved in N3.
- b) The variable subscript in left-adjusted display code is saved in the same manner as the constant subscript except in cells L1, L2, and L3.

5.9 REFORM

5.9.1 Reformats the DO control limits for the variable subscripts and saves the information in L1, L2, L3. The initial value, the terminal value, and increment are saved in the 1st, 2nd and 3rd 18 bit bytes of the word respectively. The order in which the subscripts are to be incremented is saved in the upper 6 bits. Therefore, A(I, J, K), I=1, 10), K=2, 10, 2) J=4,5) would be saved as:

L1	1	1	10	1	for I
L2	3	1	5	4	for J
L3	2	2	10	2	for K
	6	18	18	18	

5.10 PAT

5.10.1 Sets the initial value from the DO control limits from L1, L2, and L3 into their corresponding N1, N2 and N3 cells. If the DO control increment is greater than 1, then SWITCH is set to indicate each piece of data must be sent individually rather than possibly with a "REPI" pseudo-op.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 8.5
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.11 KEEP

5.11.1 Adds the running total of the implied DO loop for the array.

5.12 HELLO

5.12.1 Sets SWITCH to indicate each piece of data must be sent individually if the subscripts are not to be incremented in consecutive order.

5.13 RUNOVER

5.13.1 Computes the difference between the array total and the starting point of the array which has only constant subscripts and uses this difference as the new array total.

5.14 ITEM

5.14.1 Matches a variable in a list to its corresponding data.

5.15 BRUTE

5.15.1 Increments the subscript pointers for an array with DO loop control. The data items must be sent individually because the position within the array must be recalculated each time. Either an increment of a loop control was not 1 or the variable subscripts are not to be incremented in the same order as they appeared after the array name.

5.16 NEWSYM

5.16.1 Entered when a new variable is mentioned in a DATA statement. Calculates the arithmetic type and saves this type along with setting the variable bit, defined bit and common bit in the symbol table. The variable name is stored in the buffer so that its associated data will follow the name in the buffer. The DATA block is increased by 1.

6.0 Formats

6.1 PAREN

Parenthesis count: incremented for left parenthesis
decremented for right parenthesis

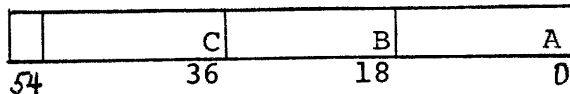
6.2 TEMPL

a) Used in LAST (see section 5.8) contains the order number of the subscript in the lowest 18 bits.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 8.6
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

b) Used in REFORM (see section 5.9).



where:

A = order of variable in implied DO loop.

B = shift count for DO limits.

0 for initial value.
18 for terminal value.
36 for increment.

C = order of variable in the subscript.

6.3 EQUIBIAS - equivalence bias if applicable.

6.4 TOTAL

Maximum number of words which can be stored in the array.
Altered by DO loop control and subscripted reference other
than the first element of the array.

6.5 VARBL

Name of the variable currently being processed in left
adjusted display code. Lowest 18 bits are zero.

6.6 SWITCH

- a) 1 for non-contiguous storage of data as with DO loop control.
- b) For non-subscripted variables in a list, the lowest 18 bits contain the E-list address of the next element in the variable list.
- c) When a subscripted variable is mentioned in a variable list, the lowest 18 bits contain the current E-list address in the data. After the subscript is processed the lowest 18 bits contain the E-list address of the next element in the variable list.

6.7 ORGPNT

Contains the last binary index into the array. If a "REPI" pseudo-op is used to preset data, then in order to have the location counter be positioned at the proper place within the array for the next piece of data this index is used to space over the repeated data.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 8.7
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- 6.8 D1
- a) simple variable = 0
 - b) one dimensioned = 0
 - c) two dimensioned = first dimension in lowest 18 bits
 - d) three dimensioned = first dimension in lowest 18 bits and 1st * 2nd dimensions in bits 30-48.

- 6.9 N1 - 1st subscript -1
 N2 - 2nd subscript -1
 N3 - 3rd subscript -1
 Set only once for a variable with constant subscripts.
 Incremented after each data item for variable with variable subscripts.

- 6.10 L1 - a) in LAST (see section 5.8)
 L2 contains the variable subscript
 L3 name in left adjusted display code.
 b) in REFORM (see section 5.9)

	0	increment or 1	terminal value	initial value
57	54	36	18	0

where:

0 is the order in which the subscripted are to be incremented.

- 6.11 AGAIN
 E-list address of the first data item of a repeated data list.

- 6.12 REPEAT
- | | | |
|--------------|----|--------|
| display code | | binary |
| 59 | 30 | 18 0 |

repeat count

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 8.8
PRODUCT NAME FORTTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

6.13 DBLCMPX

0 for single variable; non-zero for double or complex variable.

6.14 UNIQUE

Unique name which is incremented after every use. Specifies a beginning address for data to be repeated via "REPI" pseudo-op.

7.0 The low core cells, SYM1, DIM1, etc., are set via an EQU. All of the special characters, =,), (, *, etc., are referenced symbolically and set via an EQU.

8.0 Not applicable.

9.0 Throughout the processor A4 contains the current E-list address and A0 contains the address of the next available location in DATA's COMPASS buffer. Whenever an external routine is referenced, these two addresses are saved in bits 0-18 and bits 30-48 respectively in a cell called ELIST.

The beginning address of DATA's COMPASS buffer is obtained from DOLAST (RA+31B).

Examples of generated COMPASS images:

1. DATA A/1, 2, 3, 4, 5/ or DATA (A =1, 2, 3, 4, 5)
where A has been dimensioned 5

```
ORG A
DATA 1B
DATA 2B
DATA 3B
DATA 4B
DATA 5B
```

2. DATA B/27*0/ or DATA (B=27*0)
where B has been dimensioned (3,3,3)

```
ORG B
S SET *
DATA 0B
REPI S/S ,B/1B,C/32B
```

CONTROL DATA CORPORATION . COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 8.9
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

3. DATA A,B,C/4.DO,5.DO,2*3.DO/
where A,B,C are double and C dimensioned 2

```
ORG A
DATA 172240000000000000000B
DATA 164200000000000000000B
ORG B
DATA 172250000000000000000B
DATA 164200000000000000000B
ORG CC
S SET *
DATA 172160000000000000000B
DATA 164100000000000000000B
REPI S/S ,B/2B,C/1B
```

4. DATA (NEW=4LNEW=)
where NEW has not been mentioned before

```
USE DATA.
NEW VFD 60/4LNEW=
```

5. DATA ((A(I,J,K),I=1,2),J=3,4),K=5,6)/8*0/
where A is dimensioned (2,4,6)

```
ORG A+44B
DATA 0B
ORG A+45B
DATA 0B
ORG A+46B
DATA 0B
ORG A+47B
DATA 0B
ORG A+54B
DATA 0B
ORG A+55B
DATA 0B
ORG A+56B
DATA 0B
ORG A+57B
DATA 0B
```

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 8.10
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

3. DATA A, B,C/4.DO, 5.DO,6*3D0/

where A,B,C are double and C has been dimensioned 6

```
VAR$ SET A
      ORG VAR$
      DATA 4.EEO
VAR$ SET B
      ORG VAR$
      DATA 5.EEO
VAR$ SET C
      ORG VAR$
      DATA 3.EEO
.AC SET .AC
XB SET 2
XC SET 6
DATMAC XS , XB , XC
```

4. DATA (NEW=4LNEW=)

where NEW has never been mentioned before

```
USE DATA
NEW VFD 60/4LNEW=
```

5. DATA (((A(I,J,K), I=1,2), J=3,4),K=5,6)/8*0/

where A is dimensioned (2,4,6)

```
VAR$ SET A
      ORG VAR$+0042B
      DATA 0
      ORG VAR$+0043B
      DATA 0
      ORG VAR$+0044B
      DATA 0
      ORG VAR$+0045B
      DATA 0
      ORG VAR$+0054B
      DATA 0
      ORG VAR$+0055B
      DATA 0
      ORG VAR$+0056B
      DATA 0
      ORG VAR$+0057B
      DATA 0
```

DOCUMENT CLASS IMS PAGE NO. 9.1
PRODUCT NAME Fortran Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

DOPROC Description

1.0 General Information

1.1 The DO processor (DOPROC) examines DO statements, DO-implied lists, statement numbers, statement number references, and integer variable definitions. It determines the characteristics of DO's and index functions, diagnoses nesting, syntax, and the use of statement numbers, and generates R-list macro words defining the beginning and end of each DO loop and DO-implied list. The DO statement is the only statement fully processed by DOPROC.

2.0 Entry Points

2.1 DOPROC

2.1.1 DOPROC is referenced by PH2CTL when a DO statement is encountered. DOPROC examines the E-list items of the DO for syntax. If the DO is found to be legal, it generates a label for referencing from the bottom of the DO, sets up the DO table (DOT) with flag and address information concerning the control variable, limits, etc, and generates an R-list macro for processing by the second pass DO processor (DOPRE).

2.1.2 The calling sequence for DOPROC is:

RJ DOPROC

Upon entry to DOPROC, it is expected that low core location SELIST will hold the address of the E-LIST for the DO. Upon a successful exit from DOPROC, the R-list file will contain seven words relating to the DO and if necessary, label information will be filed in SYMTAB. Low core location DOLAST will hold the address of the list entry in the DOLIST (DOL table).

2.2 DODEF

2.2.1 DODEF is referenced by ARITH, ASSIGN and LISTEDIO (on input) whenever an integer variable appears as the object of a replacement statement, ASSIGN statement, or input statement. DODEF sets up an integer variable definition item in the DOLIST (DOL table) and diagnoses illegal redefinition of loop limits.

2.2.2 The Calling Sequence for DODEF is:

RJ DODEF

where B1 has been preset to the ordinal of the integer variable in the symbol table.

2.3 DOSYM

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 9.2
PRODUCT NAME Fortran Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

2.3.1 DOSYM is referenced by ARITH and LISTED10 (output only) when an integer variable appears as an operand. DOSYM causes a search of the DOT table to see if the integer variable is the control variable for any loop and if so marks the control variable for materialization.

2.3.2 The calling sequence for DOSYM is as follows:

RJ DOSYM

where BI has been preset to the ordinal of the integer variable in the symbol table.

2.4 DOCALL

2.4.1 DOCALL is referenced by ARITH, CALL AND LISTED10 when a subroutine or function reference, explicit or implicit, is encountered. DOCALL will set a flag indicating a transfer out of the loop.

2.4.2 The calling sequence for DOCALL is:

RJ DOCALL

2.5 DOIT

2.5.1 DOIT is referenced by LISTED10 when it encounters a DO-implied list. DOIT initializes the loop as described under DOPROC.

2.5.2 The calling sequence to DOIT is:

EQ DOIT

where BI points to the E-list entry for the = sign.

Return is to DOITX.

2.6 DONE

2.6.1 DONE is referenced by LISTED10 after processing the list of a DO-implied loop. DONE closes the loop as described under DOLAB.

2.6.2 The calling sequence for DONE is:

EQ DONE

Return is to DONEX.

2.7 DOGOOF

2.7.1 DOGOOF is referenced by LISTED10 after encountering a fatal error while processing the list of a DO-implied loop. This allows DOGOOF to remove the current nest of DO-implied loops.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 9.3
PRODUCT NAME Fortrand Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

2.7.2 The calling sequence for DOGOOF is:

EQ DOGOOF

Return is to DOGOOFX

2.8 DOENT

2.8.1 DOENT is referenced by ENTRY. DOENT files an error message indicating that external entry to a loop is being attempted.

2.8.2 The calling sequence to DOENT is:

RJ DOENT

2.9 DOEND

2.9.1 DOEND is referenced by END to see if all loops have been terminated, all referenced statement numbers and format numbers have been defined as such, and all loops with entries also have exits.

2.9.2 The calling sequence to DOEND is:

RJ DOEND

2.10 DOLABCN

2.10.1 DOLABCN is referenced by PH2CTL before each labeled statement is processed. The label is checked for prior definition and, if not found, is entered into the symbol table (SYMTAB). The loop in which the label is defined is entered in the symbol table, and whether or not the label is an entry point to a loop.

2.10.2 The calling sequence for DOLABCN is:

RJ DOLABCN

CLABEL of the common block PSICOM contains the current statement label left justified with blank fill.

2.11 DOLAB

2.11.1 DOLAB is referenced by PH2CTL after each labeled statement is processed. If the label terminates one or more loops, DOLAB closes each loop; i.e. it notes exits from the loop, generates an R-list macro, and compresses the DOT and DOL tables.

2.11.2 The calling sequence to DOLAB is:

RJ DOLAB

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 9.4
PRODUCT NAME Fortran Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

CLABEL of the COMMON block PSICOM contains the current statement label left justified with blank fill.

2.12 DOLABR

2.12.1 DOLABR is referenced by IF, CALL, and GO TO processors when reference to a statement label is encountered. If the label is not yet defined it is entered in the DOL table, otherwise entries and exits are noted.

2.12.2 The calling sequence to DOLABR is:

RJ DOLABR

SELIST points to the E-list item for the label, and upon return B1 contains ordinal of SYMTAB entry for that label.

3.0 DOPROC will produce the following diagnostic messages:

- (1) Loops are nested more than 50 deep.
- (2) The terminal label of a D0 must be an integer constant between 0 and 100,000.
- (3) The terminal statement of this D0 precedes it.
- (4) The control variable of a D0 or D0-implied loop must be a simple integer variable.
- (5) The syntax of D0 parameters must be $I=M_1, M_2, M_3$, or $I=M_1, M_2$.
- (6) A constant D0 parameter must be between 0 and 131K.
- (7) A D0 parameter must be an integer constant or variable.
- (8) This statement number has been used before.
- (9) A previous statement in this nest references this statement number illegally.
- (10) This statement references a previous statement number in this nest illegally.
- (11) A D0 loop may not terminate on this type of statement.
- (12) A D0 loop which terminates here includes this unterminated D0 loop.
- (13) This statement redefines a current loop control variable or parameter.
- (14) ENTRY statements may not occur within the range of a D0 statement.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 9.5
PRODUCT NAME Fortran Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- (15) This D0 loop is unterminated at program end.
- (16) This loop is entered from outside its range but has no exit.
- (17) This referenced statement number does not appear on an executable statement.
- (18) This referenced format number does not appear on a format statement.
- (19) More storage required by D0 statement processor for
- (20) The variable upper limit and the control variable of this D0 are the same producing a non-terminating loop.
- (21) Compiler error.
- (22) The constant lower limit is greater than the constant upper limit of a D0.
- (23) The referenced label is greater than five characters.
- (24) Zero statement labels are illegal.

Errors 8,11,13,19,20 and 22 are informative.

Error 21 is fatal to compilation.

The remainder are fatal to execution.

4.0 ENVIRONMENT

DOPROC depends on information from the other processors in varying forms and degrees. It must have the E-list pointer for each D0 statement and D0-implied loop. It files entries in the symbol table (SYMTAB) and stores both temporary and permanent information regarding them. It outputs macros and generates label items into the R-list.

The communications region of COMMON block PSICOM will contain several variables of interest to DOPROC. These are:

- (1) SELIST - contains the address of the current entry in the E-list.
- (2) ELAST - contains the address of the last entry in the E-list.
- (3) D01 - contains the address of the first entry in the DOLIST (DOL table).
- (4) DOLAST - contains the address of the last entry in the DOLIST (DOL table).

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 9.6
PRODUCT NAME Fortran Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- (5) SYMEND - contains the address of the last entry in the symbol table (SYMTAB).
- (6) SYMI - contains the address of the unused entry for ordinal zero in the symbol table (SYMTAB).
- (7) CLABEL - contains the current statement label, left justified with blank fill.
- (8) TYPE - contains the current statement type in binary right justified in the word.

5.0 STRUCTURE

5.1 DOTOP

5.1.1 DOTOP processes each DO statement and DO-implied list and generates the R-list macro words for the top of the loop.

5.2 SYNCHEK

5.2.1 SYNCHEK performs a syntax check on each DO statement and DO-implied list.

5.3 INTVAR

5.3.1 INTVAR makes an integer variable assurance check and returns a verdict of integer variable or not.

5.4 IDEF

5.4.1 IDEF checks for illegal definition of loop variables and makes entries in DOLIST (DOL table).

5.5 LIMIT

5.5.1 LIMIT converts loop limits and determines if constant, variable, or illegal.

5.6 IREF

5.6.1 IREF searches the DOT table to determine if the integer variable referenced in this statement is a control variable. If so it sets flag M.

5.7 GENMAC

5.7.1 GENMAC generates the R-list macro words.

5.8 CHECK

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 9.7
PRODUCT NAME Fortran Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- 5.8.1 CHECK checks the form of the limits and constructs symbolic-constant representation for each.
- 5.9 LABEL and SYMBOL
 - 5.9.1 LABEL and SYMBOL search SYMTAB for the given display code entry and return I and H (R-list description) of entry and address.
- 5.10 ERPRO
 - 5.10.1 ERPRO is used to produce a variety of error messages.
- 5.11 SIO
 - 5.11.1 SIO is called to file R-list items.
- 5.12 ETB
 - 5.12.1 ETB converts E-list integer constant items into the corresponding binary constant.
- 5.13 NAME
 - 5.13.1 Given the ordinal of a SYMTAB entry, NAME checks the E-field of the BYMTAB entry and if zero puts the ordinal into B1 for use by IREF. If E=1, it puts the base and bias from DIMLIST into B1 and B2 respectively.

6.0 FORMATS

The DO processor receives information from SCANNER in E-LIST format and using SYMTAB data generates R-LIST macros for processing by pass two of the compiler. Interim data is stored into two tables:

- (1) DOTABLE (DOT) - information on current loops
- (2) DOLIST (DOL) - references to undefined labels and integer variable definitions.

Each table will be discussed in turn as to content, format, etc.

DOT Table - The DOT table is a rigid table of up to 51 entries consisting of one entry per DO statement. However, each entry requires 3 memory words so that 153 60-bit memory words are consumed. The first entry in the table is not used. Flag and address quantities are defined as follows:

P - an 18-bit index relating the DO statement to the machine address in the DOL table where information about the loop is stored.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 9.8
PRODUCT NAME Fortran Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

CV - A three bit field containing flags b, c, and d where B, C, and D are lower limit, upper limit and increment of the D0 respectively. If b, c, or d is set to 1 then the corresponding limit is variable. Otherwise the limit is constant.

Example:

D0 10 1 = 1, N, 2

B = 1 b = constant or 0
C = N c = variable or 1
D = 2 d = constant or 0

and CV - 010 in binary

Flags - seven bits of loop description broken into seven one-bit fields.

- (1) E - set to 1 if loop may be entered at a point other than the top.
- (2) X - set to 1 if loop may be exited at a point other than the terminating statement.
- (3) I - set to 1 if loop contains another loop.
- (4) M - set to 1 if loop control variable must be materialized (placed in memory)
- (5) V - set to 1 if control variable same as incremental limit.
Example: D0 10 D = 1, N, K
- (6) J - set to 1 if loop contains an implicit or explicit subroutine call.
- (7) R - set to 1 if all integer variables are assumed to be redefined within a loop.

N - a 12-bit field containing the number of integer variable definitions.

S - A 12-bit field holding the symbol table (SYMTAB) ordinal of the statement number referred to in the D0 statement.

IX - A 12-bit field holding the symbol table ordinal for the control variable entry.

L - A 12-bit field holding the symbol table ordinal of the generated label at the top of the loop.

B - An 18-bit field that is either a binary constant for the lower limit of the D0 or the symbol table ordinal of a variable lower limit.

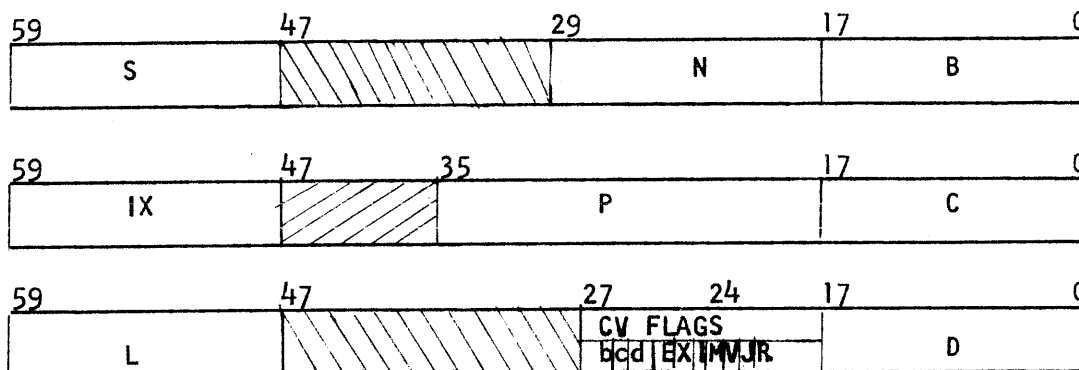
CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 9.9
 PRODUCT NAME Fortran Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

C - An 18-bit field that is either a binary constant for the upper limit of the D0 or the symbol table ordinal of a variable upper limit.

D - An 18-bit field that is either a binary constant for the incremental parameter of the D0 or the symbol table ordinal of a variable incremental parameter.

Information is aligned within the three words as follows:



DOL list - The DOL list is a variable length list residing in working storage and limited by the growth of SYMTAB and/or E-list entries. The list has two forms of entries:

- (1) References to undefined labels.
- (2) Integer variable definitions.

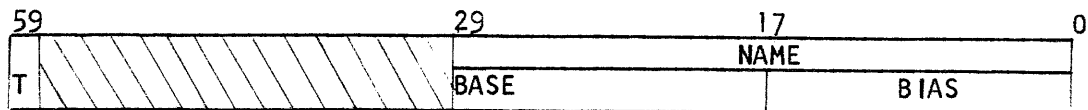
All DOL list entries are one word (60 bits) in length and the fields are these:

- (1) T - a one bit flag indicating a reference to an undefined label (0) or an integer variable definition (1)
- (2) NAME - a 30 bit field with contents in one of three forms:
 - a) T = 0, NAME is the ordinal of the symbol table entry for the undefined label, right justified in the 30 bit field.
 - b) T = 1, E of SYMTAB = 0, then NAME is the ordinal of the symbol table entry for the integer variable definition right adjusted in the upper 12 bits of the 30 bit field (to align with the base of an equivalenced set of variables)
 - c) T = 1, E = 1, then name contains the base and bias of the equivalenced variable as 12 bit and 18 bit fields respectively.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 9.10
 PRODUCT NAME Fortran Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

The DOL list format is:



R-LIST Macros - DOPROC will generate R-LIST macros at the top and bottom of each loop, for processing by DOPRE in pass two of the compilation. Each macro will consist of 6 words (60 bits each). The first word is a standard macro reference with fields for OC, IN, etc. (See R-LIST language description). The second word has the ordinal of SYMTAB where the DO label definition is filed. This field is the rightmost 30 bits and the ordinal is right justified. Beginning with the left half of word two and continuing through word 6 is information relating to the three limiting parameters of the DO (B,C,D where the general form is DO SN I=B,C,D) and the induction variable I.

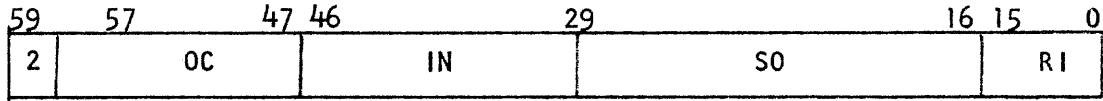
The parameters for words 2-6 are as follows:

- (1) L - a 30-bit label reference of the I and H variety described in R-LIST description. L is the label attached to the DO.
- (2) SB - 30 bits of symbolic (I and H) information related to a variable B or base if variable is equivalenced to another.
- (3) SC - 30 bits of symbolic (I and H) information related to a variable C or base address if variable is equivalenced to another variable. Bias appears in CC.
- (4) SD - 30 bits of symbolic (I and H) information related to another variable. Bias appears in CD.
- (5) SI - 30 bits of symbolic (I and H) information related to a control variable (induction variable) or base address if variable is equivalenced to another variable. Bias would appear in CI.
- (6) CB - 18 bits of binary constant representing the constant B or the bias if B is an equivalenced variable.
- (7) CC - 18 bits of binary constant representing the constant C or the bias if C is an equivalenced variable.
- (8) CD - 18 bits of binary constant representing the constant D or the bias if D is an equivalenced variable.
- (9) CI - 18 bits of bias if the induction variable is equivalenced to another variable. If variable does not appear in an equivalence statement then CI is zero.

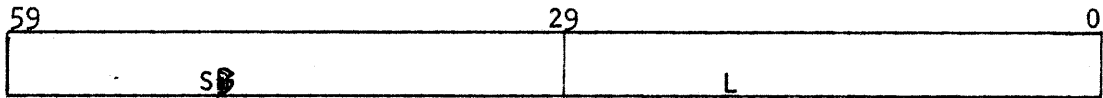
CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 9.11
 PRODUCT NAME Fortran Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

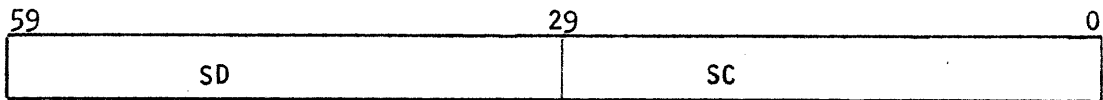
MA + 0



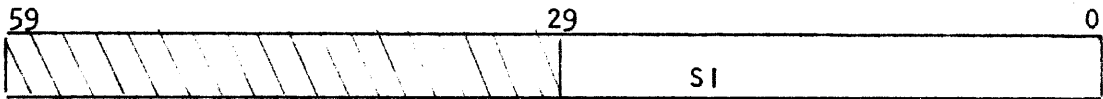
+



+



+



+



+



CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 10.1
PRODUCT NAME Fortran Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

STOP

1.0 General Information

1.1 STOP is called by PH2CTL when SCANNER encounters a STOP statement. STOP then sets up the proper R-list entries to cause a 60 bit jump to STOP. at execution time. When the system routine STOP. is entered X7 will contain up to 5 octal digits that follow the word STOP on the FORTRAN statement. This processor is used only in phase 2 of pass 1.

2.0 Usage

2.1 STOP

2.1.1 R-list is generated to:

1. Load an X register with the octal digits (up to five) given after the word STOP on the FORTRAN statement. The register is loaded from the Hollerith constant region HOL.
2. Transmit the characters from the input register to the output register X7. Characters are left justified, blank fill.
3. Jump to a point in the system to display the message and terminate the job.

2.1.2 The calling sequence is:

RJ STOPP

SELIST must contain the address of the E-list items related to the STOPP.

2.1.3 Flow of the processing goes as follows:

1. The symbol STOPP. is entered into the symbol table, if not there.
2. The external bit is set for STOPP. in SYMTAB.
3. The E-list is checked for End of Statement.
4. If first word of E-list is not an end of statement, it is typed as Hollerith and tabled through CONVERT.
5. R-list macro references are formed from the IH of STOP. and the IH, CA of the Hollerith constant.
6. The R-list macro references are filed in S10 by WRWDS.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 10.2
PRODUCT NAME Fortran Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- 3.0 PAUSEP STOPP produces the following FE diagnostic:
The field following STOP and PAUSE must be 5 or less octal digits
- 4.0 Not applicable
- 5.0 Not applicable
- 8.0 A macro was written for inclusion in pass 2 processing which is exercised by the referencing R-list filed in S10. This macro uses the IH of STOP. and the IH, CA fields for the Hollerith constant stored in HOL. to generate the following execution time code.

SAi	HOL.+CA
BX7	Xi
RJ	STOP.

STOPPcalls on WENDS, CONVERT, and SYMBOL.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 11.1
PRODUCT NAME Fortran Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

PAUSE

1.0 General Information

1.1 PAUSE is called by PH2CTL when SCANNER encounters a PAUSE statement. PAUSE then sets up the proper R-list entries to cause a 60 bit jump to PAUSE. at execution time. When the system routine PAUSE. is entered X7 will contain up to 5 octal digits that follow the word PAUSE on the FORTRAN statement. This processor is used only in phase 2 of pass 1.

2.0 Usage

2.1 PAUSE

2.1.1 R-list is generated to:

1. Load an X register with the characters (up to five) given after the word PAUSE on the FORTRAN statement. The register is loaded from the Hollerith constant region HOL.
2. Transmit the characters from the input register to the output register X7. characters are left justified, blank fill.
3. Jump to a point in the system to display the message and suspend the job.

2.1.2 The calling sequence is:

RJ PAUSEP

SELIST must contain the address of the E-list items related to the PAUSE.

2.1.3 Flow of the processing goes as follows:

1. The symbol PAUSE. is entered into the symbol table, if not there.
2. The external bit is set for PAUSE. in SYMTAB.
3. The E-list is checked for End of Statement.
4. If first word of E-list is not an end of statement it is typed as Hollerith and tabled through CONVERT.
5. R-list macro references are formed from the IH of PAUSE. and the IH, CA of the Hollerith constant.
6. The R-list macro references are filed in S10 by WRWDS.

3.0 PAUSEP produces the following FE diagnostic:
The field following STOP and PAUSE must be 5 or less octal digits.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 11.2
PRODUCT NAME Fortran Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

4.0 Not applicable

5.0 Not applicable

6.0 Not applicable

7.0 Not applicable

8.0 A macro was written for inclusion in pass 2 processing which is exercised by the referencing R-list filed in S10. This macro uses the IH of PAUSER and the IH, CA files for the hollerith constant stored in HOL. to generate the following execution time code.

SAi	HOL.+CA
BX7	Xi
RJ	PAUSE.

PAUSE calls on WRWDS, CONVERT, and SYMBOL.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 12.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

ARITH

1. General information
 - 1.1 ARITH processes replacement statements. It calls ASFDEF to process statement function definitions. ARITH translates any arithmetic, logical, relational, or masking expressions which may legally appear in any type of statement. It is found in phase-two of the first pass of the compiler.
2. Entry points:
 - 2.1 BEFTB
 - 2.1.1 Function: to allow ENDPROC to refer to ARITH's basic-external function name table (BEFTB). This is not an entry into a coding area.
 - 2.2 GTARTH
 - 2.2.1 Function: To translate the expression in a computed GOTO statement.
 - 2.2.2 Calling sequence and returns: GTARTH is entered by an RJ instruction. It returns control to GOTOPRC thru its entry point.
 - 2.2.3 Processing description: ARITH will translate the expression (addressed by SELIST) until an end-of-statement element is encountered or a syntax error occurs. After translating the expression, if it is other than type integer an RLIST macro is output to convert it to integer.
 - 2.3 IXFN
 - 2.3.1 Function: To output a macro to RLIST to set an X register to the address of an I/O list item, preceded by an evaluation of the address

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 12.2
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/6600 FORTRAN Extended

if it is an array reference with a non-standard subscript.

2.3.2 Calling sequence and returns: IXFN is entered by an RJ instruction and exits through its entry point.

2.3.3 Processing description: IXFN enters a special operator (XFLP) into the OPSTAK with zero precedence and enters the beginning of ARITH's main translation loop at NEXTE. SELIST initially addresses the name of the I/O item. The first ELIST element following the complete I/O item should have a precedence of zero (a comma or end of statement, e.g.) and will cause XFLP to be popped from the OPSTAK which results in a jump to IXFN2. IXFN2 checks to make sure that the last entry made to ARLIST is a fetch macro. It changes the fetch macro code to a set-X macro code, sets SELIST to address the comma or end-of-statement in ELIST following the I/O item and sets X2 to the symbol table ordinal of the name of the I/O item.

2.4 ACALL

2.4.1 Function: To process the subroutine name and argument list of a CALL statement.

2.4.2 Calling sequence and returns: ACALL is entered by an RJ instruction and exits through its entry point.

2.4.3 Processing description: ACALL calls the SYMTAB entry routine (SYMBOL) for the subroutine name, and turns on the external bit in the symbol table. If there are no arguments, it exits through its entry point. Otherwise, it sets up ARITH to process the argument list as it would a general external function's argument list. It enters a special operator (SUBRLP) into the OPSTAK rather than the operator

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 12.3
PRODUCT NAME _____ FORTRAN-Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

- 137, ERMSG3: NO MATCHING RIGHT PARENTHESIS.
- 138, ERMSG4: NO MATCHING LEFT PARENTHESIS.
- 139, ERMSG5: THE OPERATOR INDICATED (-, +, *, /, or **) MUST BE FOLLOWED BY A CONSTANT, NAME, OR LEFT PARENTHESIS.
- 140, ERMSG6: A NAME MAY NOT BE FOLLOWED BY A CONSTANT.
- 141, ERMSG7: MORE THAN 63 ARGUMENTS IN ARGUMENT LIST.
- 142, ERMSG8: A CONSTANT MAY NOT BE FOLLOWED BY AN EQUAL SIGN, NAME, OR ANOTHER CONSTANT.
- 143, ERMSG9: EXPRESSION TRANSLATOR TABLE (OPSTAK) OVERFLOWED. SIMPLIFY THE EXPRESSION.
- 144, ERMSG10: LOGICAL OPERAND USED WITH NON-LOGICAL OPERATORS.
- 145, ERMSG11: NO MATCHING RIGHT PARENTHESIS IN SUBSCRIPT.
- 146, ERMSG12: LOCAL ENTRY POINT REFERRED TO AS EXTERNAL FUNCTION.
- 148, ERMSG14: INTRINSIC FUNCTION REFERENCE MAY NOT USE A FUNCTION NAME AS AN ARGUMENT.
- 149, ERMSG15: ARGUMENT NOT FOLLOWED BY COMMA OR RIGHT PARENTHESIS.
- 150, ERMSG16: A FUNCTION REFERENCE REQUIRES AN ARGUMENT LIST.
- 151, ERMSG17: ILLEGAL CALL FORMAT.
- 152, ERMSG18: EXPRESSION TRANSLATOR TABLE (FRSTB) OVERFLOWED. SIMPLIFY THE EXPRESSION.
- 153, ERMSG19: THE OPERATOR INDICATED (.NOT. OR A RELATIONAL) MUST BE FOLLOWED BY A CONSTANT, NAME, LEFT PAREN, -, or +.
- 155, ERMSG21: BASIC INTRINSIC FUNCTION WITH AN INCORRECT ARGUMENT COUNT.
- 156, ERMSG22: EXPRESSION TRANSLATOR TABLE (ARLIST) OVERFLOWED. SIMPLIFY THE EXPRESSION.
- 158, ERMSG24: ILLEGAL INPUT/OUTPUT ADDRESS.
- 159, ERMSG25: RIGHT PARENTHESIS FOLLOWED BY A NAME, CONSTANT, OR LEFT PARENTHESIS.
- 160, ERMSG26: MORE THAN ONE RELATIONAL OPERATOR IN A RELATIONAL

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 12.4
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

entered for the beginning of a function argument list (ARGLP). It then enters the beginning of ARITH's main translation loop at NEXTE. When SUBRLP is popped from the OPSTAK, ACAL7 is entered. ACAL7 then calls ARGP8CR to output loads of any saved function results to RLIST. Then the ARLIST block is output to RLIST and ACALL exits through its entry point.

2.5 ARITH

2.5.1 Function: To translate the expression addressed by SELIST, unless the expression is found to be a Statement Function definition, in which case ASFDEF is entered to process the definition.

2.5.2 Calling sequence and returns: ARITH is entered by an RJ instruction and exits through its entry point, or to ASFDEF in the case of a Statement Function definition.

3. Diagnostics produced:

3.1 Fatal to compilation: none

3.2 Fatal to execution:

Message Ordinal
& Symbolic Name

70,	ERMSG51:	A CONSTANT ARITHMETIC OPERATION WILL GIVE AN INDEFINITE OR OUT-OF-RANGE RESULT.
71,	JAMER4:	EXPRESSION TRANSLATOR TABLE (JAMTB1) OVERFLOWED. SIMPLIFY THE EXPRESSION.
75,	JAMER5:	75 = SIGNS PER REPLACEMENT STATEMENT IS THE MAXIMUM.
76,	ERMSG52:	TYPE ECS NOT AVAILABLE IN THIS VERSION OF FTNX.
135,	ERMSG1:	ILLEGAL USE OF THE EQUAL SIGN.
136,	ERMSG2:	VARIABLE FOLLOWED BY LEFT PARENTHESIS.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 12.5
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

EXPRESSION.

- 161, ERMSG27: A COMMA, LEFT PAREN, =, .OR., OR .AND. MUST BE FOLLOWED BY A NAME, CONSTANT, LEFT PAREN, -, .NOT., OR +.
- 162, ERMSG28: AN ARRAY REFERENCE HAS TOO MANY SUBSCRIPTS.
- 163, ERMSG29: NO MATCHING RIGHT PARENTHESIS IN ARGUMENT LIST.
- 164, ERMSG30: ILLEGAL FORM INVOLVING THE USE OF A COMMA.
- 165, ERMSG31: LOGICAL AND NON-LOGICAL OPERANDS MAY NOT BE MIXED.
- 166, ERMSG32: DIVISION BY CONSTANT ZERO.
- 167, ERMSG33: A COMPLEX BASE MAY ONLY BE RAISED TO AN INTEGER POWER.
- 168, ERMSG34: USE OF THIS SUBROUTINE NAME IN AN EXPRESSION.
- 169, ERMSG35: SUBROUTINE NAME REFERRED TO BY CALL IS USED ELSEWHERE AS A NON-SUBROUTINE NAME.
- 195, ERMSG48: TOO MANY SUBSCRIPTS IN ARRAY REFERENCE.
- 198, ERMSG37: LEFT SIDE OF REPLACEMENT STATEMENT IS ILLEGAL.
- 203, ERMSG49: THE TYPE OF THIS IDENTIFIER IS NOT LEGAL FOR ANY EXPRESSION.
- 204, ERMSG50: A CONSTANT OPERAND OF A REAL OPERATION IS OUT OF RANGE OR INDEFINITE.
- 207, JAMER1: THIS COMBINATION OF OPERAND TYPES IS NOT ALLOWED IN THIS VERSION.
- 211, JAMER2: DOUBLE OR COMPLEX OPERAND IN SUBSCRIPT EXPRESSION NOT ALLOWED.
- 212, JAMER3: DOUBLE OR COMPLEX ARGUMENT NOT LEGAL FOR THIS INTRINSIC FUNCTION.
- 219, ERMSG53: .NOT. MAY NOT BE PRECEDED BY NAME, CONSTANT, OR RIGHT PARENS.

3.3 INFORMATIVE

- 147, ERMSG13: ARRAY NAME OPERAND NOT SUBSCRIPTED. FIRST ELEMENT WILL BE USED.
- 154, ERMSG20: THE NUMBER OF ARGUMENTS IN THE ARGUMENT LIST OF A NON-BASIC EXTERNAL FUNCTION IS INCONSISTENT.
- 170, ERMSG36: THE NUMBER OF ARGUMENTS IN A SUBROUTINE ARGUMENT LIST IS INCONSISTENT.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 12.6
PRODUCT NAME _____ FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

184, ERMSG39: A HOLLERITH CONSTANT IS AN OPERAND OF AN ARITHMETIC OPERATOR.

3.4 NON-USASI

134, ERMSG0: MORE THAN ONE EQUAL SIGN.

157, ERMSG23: ARRAY NAME REFERENCED WITH FEWER SUBSCRIPTS THAN THE DIMENSIONALITY OF THE ARRAY.

162, ERMSG33: HOLLERITH CONSTANT APPEARS OTHER THAN IN AN ARGUMENT LIST OF A CALL STATEMENT OR IN A DATA STATEMENT.

185, ERMSG40: NON-USASI SUBSCRIPT.

186, ERMSG41: MASKING EXPRESSIONS ARE NON-USASI.

187, ERMSG42: THE TYPE COMBINATION OF THE OPERANDS OF AN EXPONENTIAL OPERATOR IS NOT USASI.

188, ERMSG43: A RELATIONAL HAS A COMPLEX OPERAND.

189, ERMSG44: THE TYPE COMBINATION OF THE OPERANDS OF A RELATIONAL OR AN ARITHMETIC OPERATOR (OTHER THAN **) IS NOT USASI.

4. Common data:

4.1 Absolute locations in lower core:

12B, SYM1 = The first-word address of the symbol table. This is used to set the defined bit in SYMTAB via a SYMTAB ordinal.

17B, DIM1 = The first-word address of the DIMENSION information table. This is used to get information for subscripted array references.

22B, EXFLG = 0 until the first executable statement occurs. (Statement functions may not be defined after the first executable occurs).

24B, TYPE = Type code of the current statement. (Different statement types have different legal syntax at the end of expressions).

26B, CON1 = First-word address of CONLIST. ARITH uses this to fetch the values of constants which are involved in compile-time constant sub-expression evaluation.

32B, SELIST= The address of the next ELIST element to be processed.

37B, CDCNT=The card (line) number of the first card of the current statement. Inserted in RJ6 RLIST instructions for trace-back information.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 12.7
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

44B, DOFLAG=0 if the current statement is not in the range of a DO statement. (Dummy variables not in the range of DOs are loaded indirectly rather than using address substitution).

45B, CBNFLG=0 if the call-by-name option (T) has not been selected on the FTN control card. This determines whether Basic External Functions and externally evaluated exponential operation functions are called by value or by name.

64B, NRLN= The number to assign to the next RLIST result, e.g., $R_i = R_{10} + R_{15}$ where i is determined by (NRLN). SELIST and NRLN are also referred to as EPOINT and NARN.

4.2 COMMON Blocks:

NAALN: next available APLIST number. Also used by CALL.

STSORD: next available statement-temporary-store number. Reset to 1 by PH2CTL at the start of each statement.

CLNFO: A block of information used by both CALL and ARITH:

SUBFWA= the symbol table address of the name of the subroutine being called.

SUBH= the SYMTAB ordinal of the subroutine name.

ARGCNT= the number of arguments in the arg list. (These last three cells are set by ARITH for CALL's use).

NARGSF= 0 if there is an argument list.

SUBNAME= holds the name of the subroutine in E-list form. (These two cells are set by CALL for ARITH's use).

5. Subroutines used by ARITH:

External Routines:

5.1 WRWDS: Used to make entries to the RLIST file.

5.2 SYMBOL: SYMTAB search and entry routine.

5.3 CONVERT: Constant conversion and CONLIST entry routine.

5.4 ASFREF: Called as each statement function reference is encountered to insert the statement function with actual arguments replacing dummy arguments into the ELIST block.

5.5 DODEF: Called to inform DOPROC of the definition of a variable or

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 12.8
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 1.0 _____ MACHINE SERIES 64/65/6600 _____

- 5.6 DOCALL: Called to inform DOPROC of an external function reference.
- 5.7 DOSYM: Called to inform DOPROC of a reference to a variable.

Local Routines:

- 5.8 FUNC5RT: This routine is called when a function reference (other than a statement function) is encountered. The reference might occur in an argument list, so a block of cells (FRLW) used to hold information about argument lists is entered into the OPSTAK followed by the ARGLP operator (which will be popped by the right parens which terminates the list) and an ARGCMA operator (which will be popped by the comma after the first argument or the right parens if only one argument). The FRLW block is initialized. DOCALL is called if it is an external function. If the result of a previously referenced external function has not been saved, an instruction is output to RLIST to save the result.

The routine is called by the main line processor and by the exponential operator processor.

- 5.9 CARGPORT This routine is called by the main line processor and the exponential operator processor. It is called after each argument of a non Statement Function argument list has been scanned (it may be an expression). Intrinsic, basic external, and general external arguments are each processed differently. Intrinsic arguments cause the R name of the argument to be added to the R name table (RNTB), basic external arguments cause register-store instructions to be output to ARLIST (which cause particular X-registers to be associated with the arguments), and general external arguments cause a store to APLIST or assemble to APLIST instruction to be sent to ARLIST.

- 5.10 ARGPIRT: This routine is called by the main line processor and the exponential operator processor after the argument list has been processed. If the function is general external, it outputs a call by name macro to ARLIST. It then enters ARGPS8CR to output loads of functions saved during the processing of the list, if any, to RLIST. Then, register define instructions are output to ARLIST, giving R-names to the result register(s), X6 (and X7). Then all of the ARLIST for this function reference is output to RLIST. The next available location in the ARLIST buffer is adjusted. A psuedo-op giving the name of the function result is then output to ARLIST. Finally, the FRLW block is restored to the values it contained before this function reference.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 12.9
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

If the function was intrinsic, the contents of RNTB would be used to set up the parameters of the corresponding macro and the macro would be sent to ARLIST. Finally, the FRLW block would be restored.

If the function was basic external a call by value macro would be sent to ARLIST and ARGP8CR would be entered, as for general externals.

- 5.11 INGEN: Processes binary operations. The input is the address of the operands, and the macro code of the operator. If both operands are real or integer constants and the operator is +, -, *, or / then the operation is made on the constants, the instructions which loaded the constants are no-oped and a macro to load the new constant is output; otherwise a macro is formed and output to ARLIST to operate on the operands and the operand entries in ARLIST are marked as having been used. Finally, the cells holding the addresses of the last two available operands (RL1 & RL2) are reset.
- 5.12 UINGEN: Processes binary operations, similar to INGEN.
- 5.13 MACOUT: Routine to make entries to ARLIST. The input is: type of result (e.g., Double Precision), the macro descriptor (macro number, number of Rs, IHs, CAs), NARN (next available R name), and the macro parameters in a block called PARAMS. MACOUT forms the ARLIST information word and the macro in the next available locations in ARLIST.
- 5.14 MODCH: Used to generate a macro to convert from one data mode to another. The input is the address of the operand in ARLIST which is to be converted, and the data type to which it is to be converted. The type code of the operand and the new type code are combined to form a vector. The vector table is entered: the correct convert-macro code is selected and one of two possible branches are jumped to. A macro is then output to ARLIST.
- 5.15 JAM1: (See seventh section in 8.) This is an extension of INGEN to process DOUBLE and COMPLEX operations. It outputs macros addressing variables rather than R names as results, followed by a macro to store the results of the operation in statement temporary storage.
- 5.16 JAM3: Is an extension to UINGEN, similar to JAM1.
- 5.17 JAM8: This is called by MACOUT just before it exits. If the preceding operand is double length and the entry just made by MACOUT is single length, it outputs a macro to convert the single length operand to DOUBLE. If the preceding operand is single length and the entry just made is double length, it outputs conversions of all preceding unused single length operands.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 12.10
 PRODUCT NAME _____ **FORTRAN Extended** _____
 PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600

6. Formats:

6.1 ARLIST entries: (ARLIST is the block that ARITH forms RLIST subexpressions in).

Word 1:

NOP	Type of result	GP				unused	J	# of wds. in		0
		C	TU	U	XMT			this entry	preceding entry	
59	58	48	47	46	45	44	36	35	18	17

B59 = 1 if this entry is not to be sent to RLIST. If B59-1, the entire contents of word 1 have been complemented.

B58-48 indicate the type of operand as follows:

2000B	Logical
2001B	Integer
2002B	Real
2003B	Double
2004B	Complex
2005B	Octal
2006B	Hollerith

B47 = 1 if the operand is a constant.

B46 = 1 if this entry is temporarily unavailable as an operand.

B45 = 1 if this entry has been used as an operand to a subsequent operation.

B44 = 1 if a transmit instruction should follow this entry if it is the second operand of an equal-sign operator. (e.g., see the RLIST macro definition of the intrinsic function REAL).

B36 = 1 if this entry is a replacement "fetch". (Used by JAMS only).

Word 2: Word 2 is unused at this time. It was initially planned to use this word to further optimize evaluation of logical expressions but it was found that the optimizations could not be made because of a basic design peculiarity.

Word 3: The first word of the RLIST macro (or instruction). If the entry may be used as an operand, B15-0 of this word holds the R-name of the operand. (If a double length operand, R+1 is the name of the second word of the operand).

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 12.11
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

6.2 OPSTAK: This is the operator stack block. Generally, there is one word per operator. The format of that word is given below:

operator code	S.F. type	unused	CGP	S	A	E	GP	operator precedence
59	48 47 44 43		23 22	21	20	19	18	17
								0

B59-48 = the operator code. The lowest operator code is 2003B. The codes used to represent source operators are also used by ARITH although ARITH generates some of its own operators.

B47-44 are used with code 2036B to indicate the type of statement function referred to (0 = logical, 1 = integer, etc.).

B22 = 1 if B13 = 1 and this operator has been compared with one in the stack with equal precedence.

B21-19 are used with codes 2006B, 2026B, and 2036B. (These are operators which represent different types of left parens). B21-19 are used to remember whether a subscript, argument, or normal expression was being translated before the left paren occurred; this is indicated by B21-19=4, 2, or 1 respectively. The information is needed to know whether a comma is a subscript, argument, or complex constant comma.

B13 = 1 if the operator has been compared with one of higher precedence.

B17 - 0 = the precedence of the operator.

<u>Operators</u>	<u>Code in Octal</u>	<u>Precedence</u>
)	2002	0
,	2003	0
E.O.S.	2004	0 (end-of-statement)
=	2005	0
(2006	0
.OR.	2007	2
.AND.	2010	3
.NOT.	2011	4
.LE.	2012	5
.LT.	2013	5
.GE.	2014	5
.GT.	2015	5
.NE.	2016	5
.EQ.	2017	5
-	2020	6
+	2021	6
*	2022	7
/	2023	8
**	2024	10
(A	2025	0 (left parens preceding function argument list (see 2036)

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 12.12
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

<u>Operators</u>	<u>Code in Octal</u>	<u>Precedence</u>	
(S	2026	0	(left parens preceding nonstandard subscript).
,S1	2027	1	(comma following first subscript expression in non-standard subscript).
,S2	2030	1	(comma following second subscript expression in non-standard subscript).
,A	2031	1	(comma separating arguments).
U-	2032	6	(unary minus).
R-	2033	6	(reverse-operand minus).
R/	2034	8	(reverse-operand divide).
*	2035	9	(special multiply, e.g., A/B/C/D A/ (B*C*D)
(S.F.	2036	0	(left parens preceding Statement-Function argument list.
(X	2037	0	(generated left parens entered at start of IXFN).
(SUBR	2040	0	(left parens preceding CALL argument list).

6.3 FRSTB: Function results saved table. Information about functions which have been saved. One word per entry.

B58 = 1 if the function was Double or Complex.

B33-16 = the number of the statement-temporary-storage location in which the function result was saved.

E15-0 = the R name of the function result

6.4 XPNMT: Exponent function name table. This table gives the name of each library function corresponding to the various combinations of operands possible for the $**$ operator. Each entry is one word. The format is:

B59-56 = type of result of the operation (1 = integer, 2 = real, etc.)

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 12.13
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600

B-55-49 (unused).

B-43 = 1 if the combination is non-USASI.

B47-0 = the name of the function.

There are 16 entries for the 16 possible combinations. Entries for illegal combinations are all zero. To make an illegal combination legal, replace the entry with the necessary information as described above.

6.5 INTFTB: Intrinsic function table. Three words per entry, one entry per intrinsic function. The format of the first and second words is the same as the pass-1 format of SYMTAB entries with the exception that BO of the second word = 1 if the function contains an RNM RLIST instruction and therefore may need a transmit before a store. The third word holds the macro descriptor word (see MACOUT), or, if MAX or MIN type functions, special information about the type of MAX or MIN function.

6.6 BEFTB: Basic external function table. Two words per entry, one entry per function. The format is the same as the first pass format of SYMTAB entries.

7. Modification facilities: EQUs are used for diagnostic ordinals, MACROX ordinals, lower memory cell locations, block sizes, codes, etc. Diagnostic macros are used. All explicit operations and intrinsic function references result in RLIST macros rather than separate RLIST instructions.

8. Method:

3.1 There are four kinds of expressions in FORTRAN Extended:

1. Arithmetic
2. Relational
3. Logical
4. Masking

The same translator is used to translate all kinds of expressions. Translation takes place in a single left to right scan of the expression.

Translation is from the 1-list form of the source statement to RLIST language. The RLIST language specifies the machine instructions and registers to evaluate the expression, but the registers are assigned as if there were an infinite number available. The second pass assigns actual registers to the instructions.

**CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION**

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 12.14
 PRODUCT NAME _____ FORTRAN Extended _____
 PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

Arith is called by

1. Phase-2 control for processing of replacement statements.
2. Computed GO TO processor
3. IF
4. CALL

These are the only kinds of statements which may contain expressions. If the replacement statement is actually a statement function definition, ARITH will call ASFDEF to save the statement function for later reference as a macro. Statement functions are expanded in-line at each point of reference.

- 8.2.1 Computed GO TO: ARITH translates the expression, converts to type integer if necessary, outputs the RLIST block, and returns to the GOTO processor with the number + 1 of the result-R in a common location.
- 8.2.2 IF: ARITH translates the expression, outputs the RLIST to the RLIST blok, and returns to the caller with the name and type of the result in a common location.
- 8.2.3 CALL: CALL calls ACALL which is local to arith. ACALL sets up ARITH to process the CALL statement with argument list in much the same way as an external function reference is processed. ARITH outputs all the RLIST needed for the arguments and returns to CALL.
- 8.3 Generalized flow of the translation process.
- 8.3.1 The basic translation algoritm used is similar to that used to produce reverse Polish notation.

For example:

$A*(B+C)-D$

is translated to reverse Polish as follows:

<u>Step</u>	<u>E-list Item</u>	<u>Operator Stack Contents</u>	<u>Reverse Polish String</u>
1	A	.(EOS)	A
2	*	.*	A
3	(.*(A
4	B	.*(AB
5	+	.*(+	ABC
6	C	.*(+	ABC+
7)	.*	ABC+
8	-	.-	ABC+*
9	D	.-	ABC+*D
10	E.O.S.		ABC+*D-.

(E.O.S. = end-of-statement operator)

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 12.15
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

This algorithm has been modified so that RLIST, rather than Polish notation, is produced. The difference is that instead of outputting the name of a variable to a string, an instruction to load the variable is output, and instead of outputting an operator, an operation with operands named is output. For example taking the expression used in the last example, the results are:

<u>RLIST</u>	<u>Corresponding Polish</u>
R1=A	A
R2=B	B
R3=C	C
R4=R2+R3	+
R5=R1*R4	*
R6=D	D
R7+R5-R6	-

Almost all of the RLIST generated by ARITH is in the form of RLIST macro references. RLIST macros are described in detail.

8.4.1.5 Since no provision is made for saving intermediate result registers, all external function calls must be made before the remainder of the expression is evaluated. ARITH does this by forming the RLIST for the expression in a block local to ARITH called ARLIST and outputting each function reference including argument expression evaluation to the RLIST file as each argument list becomes completed.

In general, function results, except for the last function call, are saved in a block called ST for statement temporary. After the entire expression has been scanned, ARITH outputs loads of the saved function results to the RLIST file and then the remainder of the contents of ARLIST are output to RLIST.

8.4.1.6 The exponential operator, **:

For exponential operations which are not done in-line, the ** operator is really an external function reference. Since it has only two arguments, it can be called by value. So, the exponential operator is made to look like a basic external function call and some of the function processing routines are used. A problem arises in an expression like

$$A+(B*(C+D)**E/F$$

because the call to the exponential function must be output first preceded by all of the argument evaluating RLIST to the RLIST file.

The solution for this case is to save a marker with every left parens entered in the OPSTAK to point to the start of the ARLIST for what might be the first operand of an exponential operator.

DOCUMENT CLASS IMS. PAGE NO. 12.16
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

For exponentials with integer or real base expressions, and an integer constant power which is greater than one and less than 7, ARITH selects an RLIST macro code and outputs a macro to do the exponentiation in-line.

8.4.2 Subscripts

8.4.2.1 Standard: ARITH's subscript processor produces two kinds of array references for standard subscripts: a subscript psuedo-macro which is processed by DOPRE in the second pass, and a simple variable load macro with a constant addend.

8.4.2.2 Non-Standard: If the subscript processor finds that the subscript is not in standard form, it resets the ELIST pointer to address the start of the subscript. It then adds a non-standard-subscript operator to the operator stack, and returns control to the general expression scanner (at NEXTE).

There are three kinds of commas that ARITH must deal with: an argument comma, a subscript comma, and a complex-constant comma. To do this, ARITH keeps a cell called EMODE which indicates whether it is in argument mode, normal expression mode or subscript mode. Initially, EMODE is set to normal expression mode. As each left parens is met, the current mode is saved in the left parens entry in the OPSTAK and EMODE is set to normal, or argument, or subscript if the left parens is normal, or follows a function name, or follows a subscript name, respectively. As each left paren is popped from the stack, EMODE is reset to what it was before that left paren was encountered in ELIST.

Argument and subscript commas are psuedo operators and when popped from the OPSTAK they initiate the action necessary to complete the processing of the argument or subscript. As each subscript comma is popped, it causes some of the index function RLIST to be generated. When the subscript operator itself is popped from the stack, the final index function RLIST is produced followed by a macro to load the array element name.

Since a non-standard subscript expression can be any arithmetic expression, it's possible to have subscripted subscripts to any depth. This means that the non-standard subscript processing must be able to operate recursively.

DOCUMENT CLASS IMS. PAGE NO. 12.17
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

So, when the subscript operator is added to the OPSTAK, it is preceded by information about the subscript currently being processed. This is the same way that the function processor worked.

8.4.3 Relational, Logical, and Masking expressions

Processing the four kinds of expressions with the same translator presents no serious problems to the basic algorithm. The relative hierarchy of the explicit operators are:

```

**
/,*
-,+
relationals
.NOT.
.AND.
.OR.
  
```

No distinction is made between the arithmetic operators (/,*,-, and +) and the relationals. The logical operators become masking operators if their operands are non-logical. Since logical operands are only legal for the logical operators, and since .AND. and .OR. must have both operands logical or both non-logical, it is impossible to have an expression that contains both masking and logical subexpressions.

Otherwise, expression types are mixed in any way. For example,

A+B.LE.C.AND.L1 (L1 is logical)

is a logical expression with relational and arithmetic subexpressions. It is translated as follows:

<u>RLIST</u>	<u>OPSTAK</u>
R1=A	+
R2=B	.LE.
R3=R1+R2	.AND.
R4=C	.
R5=R3.LE.R4	
R6=L1	
R7=R5. AND .R6	

8.4.4 Distinctions made between operand types in arithmetic and relational expressions.

When an operator is popped out of the OPSTAK, it enters a jump or vector table where an RLIST macro code is assigned to it and it is sent to the appropriate processor. For the relational and arithmetic operators other than **, if the operands of the operator are type integer, the macro code is increased by one; if they are double precision, the code is increased by two; if complex, by three; and if real the code is used as is.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 12.18
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

Before the macro code incrementation is made, the operand types are compared. If they are not the same, an RLIST macro is output to convert the lower type operand to the same type as the higher operand.

8.5 Optimizations

8.5.1 Compile-time data-type change:

When an operator is popped from the stack, if its operands are of different types, and if the operand of lower type is constant, it will be converted to the higher type and the RLIST instruction which loaded it or set it will be replaced by one that loads the converted constant.

8.5.2 Compile-time constant subexpression evaluation:

Before a macro is formed for an operator, if it is an arithmetic operator and if its operands are integer or real constants, the RLIST for the operands are no-oped, the operation is performed on the operands, and a load or set of the computed value is output to ARLIST. If this load is subsequently used as an operand with another constant operand it will be no-oped just as the loads it replaced were no-oped.

8.5.3 Division by real or complex constants:

If a real or complex constant is preceded by a divide operator, RLIST is output to load its inverted value and the divide is changed to a multiply operator.

8.5.4 Expression transformations:

Some expressions or subexpressions can be transformed to other mathematically equivalent forms which evaluate faster on the 6600 than they would if translated by the basic algorithm.

8.5.4.1 The RLIST produced for

$A*B*C*D$

would compute the product as if it had been written

$((A*B)*C)*D$

and thus not take advantage of the 6600's two multiply units. If it had been written as

$(A*B)*(C*D)$

the two products in parentheses would be computed simultaneously. In order to achieve this effect, Arith keeps a flip-flop for popping multiply operators by operators of equal hierarchy. The flip-flop is flipped for every multiply operator encountered in ELIST, so that for

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 12.19
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

$A*B*C*D+E$

the first multiply is popped by the second, as usual, but the second is not popped by the third. The last two will be popped by the plus thus resulting in

$(A*B)*(C*D)+E$

8.5.4.2 Normally:

$A*B*C/D$

results in

$((A*B)*C)/D$

which gives no parallel execution. But if divide is given a higher priority or hierarchy than multiply, then

$A*B*C/D$

is evaluated as

$(A*B)*(C/D)$

and the divide and multiply units are working simultaneously,

Note: It might be well to note at this time that the rules for carrying out these transformations are general rules and are always in effect in the translation algorithm. The translator never looks at a source item in E-list more than once, except for non-standard subscripts. (see 8.4.2.2)

8.5.4.3 $-A+B$ or, to illustrate the preceding note,

$-(A-B)+C*D$

becomes

$C*D-(A-B)$

which reduces the number of operations from four to three, by the following rule:

If a unary minus is about to be popped from the stack by a plus, remove the unary minus from the stack and replace the plus with a reverse-operand minus operator. The macros associated with the reverse-operand minus operator are the same as those for a normal minus operator except that the first parameter is subtracted from the second instead of the second from the first.

8.5.4.4 $A/B/C$ becomes $A/B*C$, replacing a 29 cycle divide with a 10 cycle multiply (6600), by the following rule:

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 12.20
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

If the current ELIST item is a divide and the last operand in ARLIST is not type integer and the last operator in the stack is a divide or a multiply-D operator, then change the current divide to a multiply-D operator which has higher priority than the divide and specifies a multiplication. Introducing the multiply-D operator allows more than one sequential divide to become a multiply:

A/B/C/D/E becomes
 A/(B*C*D*E), which happens
 to become A/((B*C)*(D*E)) because
 the flip-flop also applies to multiply-D
 operators.

8.5.4.5 The following rules allow a great variety of transformations which allow for more parallel evaluation of expressions. Some examples of the transformations made are:

A*B/C*D	(B/C)*(A*D)
A+B/C+D	(B/C)+(A+D)
A+B*C-D	(B*C)+(A-D)
A*(B+C)/D	(A/D)*(B+C)
A-(B*C+D)-E	(A-E)-(B*C+D)

The rules are as follows:

When a right paren is encountered in E-list, set the GP (greater priority) flag bit in the operator following the right paren. Or, if the current operator pops an operator with a higher priority out of the stack set the GP bit in the current operator word.

If the GP bit of the current operator is set and it is about to pop an operator (other than unary minus) of equal priority, or it is a divide and the last operator in the stack is a multiply, then don't pop the operator from the stack, set the CGP (confirmed GP) bit in the current operator and the GPTU bit in the information word of the last ARLIST entry. If the operator left in the OPSTAK is a minus, change it to a reverse minus. Add the current operator to the stack.

The GPTU bit makes an ARLIST entry temporarily unavailable for use as an operand. After an operator with its CGP bit set is popped, the last ARLIST entry with its GPTU bit set has the bit turned off.

The following example will illustrate the use of these rules.

A*B/C*D (B/C)*(A*D)

ARLIST

R1=A
 R2=B
 R3=C

OPSTAK

*
 /
 CGP, GP,*

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 12.21
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

GPTU R4=R2/R3
R5=D
R6=R1*R5
R7=R4*R6

3.6 Arith table overflow diagnostics

3.6.1 There is a fatal-to-execution diagnostic which says:

"EXPRESSION TRANSLATOR TABLE (table-name) OVERFLOWED. SIMPLIFY THE EXPRESSION."

There are three different tables which may become overflowed.

8.6.1.1 OPSTAK table:

The size of the stack fluctuates as the expression is scanned. For example, it increases as left parentheses and operators of higher priority occur, and decreases as operators of lower priority occur.

Each operator entered in the stack requires one word of space, except for left parens which require two: one to mark the start of a possible exponential base and the other the operator itself.

The start of each function reference requires nine words which includes the recursive function processor information. If the reference occurs in an intrinsic function argument list, then the opstak is also used to save the R-names of the arguments which have been processed so far, which could be up to 62 words if the function is a MAX or MIN type function.

The start of non-standard subscripts requires four words.

The OPSTAK block size may be modified by changing the EQU named MXOSE in the common file called OPTIONS, and reassembling ARITH.

8.6.1.2 FRSTB:

This is the function result-saved table. A one word entry is made for each function that has its results saved. For example, in

$$A=F1(B)+F2(C)+F3(F4(D)+F5(E))+F6(F)$$

F1 and F2 are saved, and then F4 is saved but is reloaded to add to F5 so the size of the table goes down by one, and finally F3 is saved before calling F6.

The FRSTB block size can be modified by changing the EQU named MXFRSTB in the Option file and reassembling Arith.

8.6.1.3 ARLIST:

This is ARITH's RLIST block. The size increases as the expression is

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 12.22
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

scanned, but it decreases after each external function reference is output to the common RLIST file. A variable load entry takes 6 words; an operation takes four words if single length operands, and 8 if double length; a standard subscript psuedo-macro takes fourteen words.

The size of ARLIST is controlled by the EQU named ARLSZ in the Options file.

8.7 The Register Jam Problem

The second pass was designed under the assumption that there would always be a sufficient number of X-registers to be used in the evaluation of expressions, with the following type of exception:

A very long expression can be constructed in such a way that the result registers of enough subexpressions must be saved so that a point is reached where enough registers to continue do not exist.

The assumption was correct for integer and real expressions, but it was found that it was quite easy to run out of registers when evaluating Double or Complex type expressions.

The problem has been partially solved by modifying the expression translator to produce a different kind of output for Double and Complex expressions; partially solved because it is still possible to run out of registers for Real or Integer expressions.

These modifications to ARITH make up over 20% of the total number of source lines in ARITH, so it's important that they be described.

8.7.1 The solution in general:

When the second pass finds that it does not have enough registers to complete a sequence of statements, it reduces the number of statements in the sequence and begins again.

The solution is to break up a double or complex expression into many statements, one statement per operation. For example,

$$D1=D2*D3+D4$$

is made to look to the second pass like

$$\begin{aligned} ST1 &= D2*D3 \\ D1 &= ST1+D4 \end{aligned}$$

where ST1 is statement-temporary -one and is treated as a double variable.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 12.23
PRODUCT NAME FORTTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

8.7.2 The solution in particular:

8.7.2.1 Double length operations:

When ARITH is ready to output a macro for an operation, if the operands are double-word-length loads, it no-ops the load instructions and outputs a macro which will load the operand and do the operation. It then outputs a macro to store the results of the operation in Statement-Temporary storage followed by an end-of-statement psuedo-op. This macro is called a DSTR macro. The DSTR macro is in the same format as a double load macro and will be used as an operand to subsequent operators.

8.7.2.2 Mixed single and double:

An expression with mixed single length and double length operands, for example real and complex, presents a new problem. The expression

$$A+B*(C1+C2) , (C1 \text{ and } C2 \text{ complex})$$

would now result in

```
R1=A
R2=B
R3, 4=C1 (no-oped)
R5, 6=C2 (no-oped)
R7, 10=C1+C2
ST1=R7, 10
EOS (end-of-statement op)
R11, 12=CMPX(R2)
Etc.
```

but at the point of the last line, a reference to R2 is made which is defined in the previous statement which may end up in another sequence and therefore be undefined in this sequence. R1 won't be referred to until after the multiply operation which will occur two statements away, and so the chances that it will be undefined are even greater.

This problem has been solved by converting all unused single length operands to type Double Precision in an expression that contains a double or complex operand. The method of doing this is as follows: As each load or operation is output to ARLIST, the type is compared with the type of the last operand in ARLIST. If one is type integer or real and the other is double or complex, then a flag is set to indicate that mixed single and double has occurred, and the contents of the ARLIST blok are scanned, inserting macros to convert operands which have not already been used in operations to double precision. Thereafter, the type of each entry to ARLIST is checked, and if not double or complex, a macro is output to convert it.

DOCUMENT CLASS IMS. PAGE NO. 12.24
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

8.7.2.2.1 Mixed super-and sub-expressions

A modification to this method of dealing with the mixed single and double problem is necessary because of real or integer argument expressions occurring in a Double or Complex super-expression. Also, index functions in a double expression should not be forced to be computed in double precision. This makes it necessary to treat argument and subscript sub-expression as an autonomous expression with regard to whether mixed single and double has occurred, and with regard to how far back in the ARLIST to go when the first double operand occurs in a mixed single and double expression. And since subscripted subscripts and function references in argument expressions, etc. exist, it is necessary to keep track of where in ARLIST each subexpression starts, and for each subexpression whether mixed single and double has occurred. This introduces another table which can overflow. The name of the table is JAMTB1. The size is controlled by the EQU named JAMTB1MX in the options file. An entry of two words is made at the start of each argument and non-standard subscript. The table is reduced by two at the end of each argument and non-standard subscript.

8.7.2.3 Consider the statement

$$A(I*J)=D1+D2*D1 \quad (D1 \text{ and } D2 \text{ double})$$

The subscript I*J is non-standard. The RLIST produced for the statement would normally consist of the calculation of the index function followed by the evaluation of the expression followed by a store of the result into A(I*J). But since ends-of-statement operators now follow the double plus and double multiply operations, the subscript calculation may end up in another sequence.

So, ARITH now moves the RLIST to compute a non-standard index function from the front of the ARLIST blok to the end before outputting the store macro.

Because of this and multiple replacement statements, it is necessary to remember the starting point in ARLIST of each replacement variable. The limit on the number of replacements per statement is 75.

8.7.3 Subscripts and double length operands:

To make the implementation of this solution to the register depletion problem more feasible, Double or Complex operands in subscripts has been made illegal, and all Double or Complex array subscripts are considered non-standard.

9. Restrictions and other remarks:

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 12.25
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

9.1 Basic syntax checking is done by looking at the ELIST element following the current E-list element. The following table indicates the syntax rules:

<u>E-list element</u>	<u>may be followed by</u>
CON (constant)),,, E.O.S., OPS (2.)
ID (name)),,, E.O.S., OPS, = , (
) (1.)),,, E.O.S., OPS, =
*, =, (, .OR., .AND.	CON, ID, (, -, +, .NOT.
.NOT., relational ops	CON, ID, (, -, +
-, +, *, /, **	CON, ID, (

(1) If) is the closing parons of an IF expression it may be followed by an ID (if Logical IF) or constant (label). If) is in I/O list (IXFN call) it may be followed by (or ID.

(2) OPS = .OR., .AND., relationals, -, +, *, /, **

9.2 The format of ARITH in COMPASS:

Label field starts in column 2, operator in column 11, and symbols or integer constants in column 21. Instructions (other than 50-57) which have more than one result register (e.g., Unpack) are written so that the B register name starts one character after the end of the operation field. Operand registers start in column 18. This format results in all result registers, operand registers, and symbolic references to be found in column 12-16, 18-20, and 21-72 respectively. Comments start in column 31.

Every conditional branch instruction has a comment stating what condition must be met in order to branch.

9.3 At NEXTE, B1 is set to the address of the next Elist item to process. A large part of ARITH assumes that B1 holds this address. Therefore, B1 should be used very carefully.

Except for B1, in general it may be assumed that any register may be destroyed when calling a subroutine (including B1 for external routines). Any exceptions to this are noted in the introductory comments of the exceptional routines.

9.4 Caution to modifier. ARITH is not laid with booby traps but may appear to be so because of the complexity of the task. Transformations, etc., cause unexpected results. Beware of functions, non-standard subscripts, and exponential operators.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 13.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

CALL: IMS Documentation for the CALL Statement Processor

1. General Information:

1.1 CALL has one function: to translate CALL statements to RLIST. CALL is part of PHASE -2 of the compiler.

2. Entry Point:

2.1 CALL

2.1.1 Function: Translate the E-list form of a CALL statement to RLIST.

2.1.2 Calling sequence and returns: CALL is entered by an RJ instruction.

If the statement does not contain a fatal error, CALL exits through its entry point. Otherwise, it returns control to PH2CTL at P2ER.

(See Section 4)

3. Diagnostics Produced:

3.1 Fatal to compile diagnostics: none

3.2 Fatal to execution:

Ordinal
in
Octal

253: "Illegal CALL statement format".

254: "Illegal RETURNS parameter".

3.3 Informative: none

3.4 Non-USASI:

Ordinal
in
Octal

266: "RETURNS parameters in a CALL statement".

4. Common data:

The following are single cell locations in a common block called CLNFO:

SUBFWA: Holds the address of the first word of the symbol table entry for the name of the subroutine being called.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS, PAGE NO. 13.2
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

SUBH: Holds the SYMTAB ordinal of the subroutine name.
ARGCNT: Holds the number of arguments in the argument list.
(These last three cells are set by ARITH for CALL's use).
NARGSF: Is set to zero if there is an argument list.
SUBNAME: Holds the name of the subroutine in E-list form.
The latter two cells are set by CALL for ARITH's use.

The common block NAALN holds a single cell variable called NAALN which holds the next available APLIST (actual parameter list) ordinal. It is updated by Arith.

On entry to CALL, location 32B (SELIST) holds the address of the ELIST block for the CALL statement. The first word in the block is the name of the subroutine being called. Location 37B (CDCNT) holds the number of the first source card in the CALL statement being processed.

Location 64B (NRLN) holds the next available RLIST result number.

5. Subroutines used by CALL:

External routines:

5.1 DOCALL: Called to inform DOPROC that a CALL statement has appeared.

5.2 ACALL: Subroutine local to ARITH called to set up ARITH to process the argument list.

5.3 WRWDS: Called to output RETURNS parameter information and the return-jump instruction to RLIST.

5.4 DOLABR: Called to inform DOPROC of a reference to a statement label the (RETURNS actual arguments).

6. Formats: There are no tables in CALL. The format of the RLIST produced by CALL is given in the RLIST document. Data cells set by CALL are

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 13.3
PRODUCT NAME **FORTRAN Extended**
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600

quantities which are right adjusted. The RLIST formed by CALL consists of a 56 (APL) operation (if there is CALL argument list and a RETURNS argument list) with I = 7, H = 0, and RI = the APLIST ordinal for the subroutine, followed by type 56 (APL) operation (s) giving the returns parameters (if there are any), followed by a 50 (LD) operation with CA= 0, S0 = 15B to specify a load into X1, RI = the contents of NRLN, I = 5, H = the ordinal of the APLIST, followed by a 101 (RJ6) operation with CA = contents of CDCNT and H1 = the ordinal of the subroutine name.

7. Modification facilities:

EQUs and diagnostic macros are used which would facilitate some modifications.

8. Method: CALL calls ARITH through ACALL to process the subroutine name and the argument list, if any. After return from ACALL it processes the RETURNS list, if any, and outputs the final RLIST instructions to call the subroutine.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 14.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600

IF

1. General Information:

- 1.1 IF processes all IF statements. IF is local to ARITH which is in PHASE-2 of the compiler.

2. Entry Points:

2.1 IFE

- 2.1.1 This entry is taken to process two and three branch IF statements.

- 2.1.2 Calling sequence and returns: IFE is entered by an RJ instruction and returns through the entry point, unless a fatal diagnostic is encountered in which case it returns to PH2CTL at P2ER.

- 2.1.3 General Flow: ARITH is called to translate the IF expression. Instruction(s) are then output to RLIST to branch to the appropriate labels.

2.2 IFL

- 2.2.1 This entry is taken to process Logical IF statements.

- 2.2.2 (Same as 2.1.2)

- 2.2.3 General Flow: ARITH is called to translate the IF expression. If the statement following the expression is GOTO k where k is a statement label, IFL outputs a branch-if-.TRUE. instruction to RLIST: otherwise, it outputs a branch-if-.FALSE. to a generated label, calls the appropriate statement processor to process the statement, and then outputs the generated label.

3. Diagnostics Produced

- 3.1 Fatal-to-compile diagnostics: none

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 14.2
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. _____ 4P616 _____ MACHINE SERIES 64/65/6600

3.2 Fatal-to-execution:

Ordinal
in
Octal

- 255 "Illegal labels in IF statement"
- 256: "Logical expression in 3 branch IF"
- 257: "The statement in a Logical IF may be any executable statement other than a DO or another Logical IF"
- 260: "The expression in a Logical IF statement is not type logical"

3.3 Informative:

- 301: "This statement branches to itself"
- 302: "The statement following an IF is not labeled"

3.4 Non-USASI:

- 277: "Two branch IF statements are non-USASI"
- 300: "The expression in an IF statement is type Complex"

4. Common data:

Location 21B (LTYPE) gives the type of statement following the Logical IF expression.

Location 23B (CLABEL) holds the label (if any) of the IF statement.

Location 34B (LELIST) holds the address of the start of the statement following the Logical IF expression.

Location 60B (NLABEL) holds the label (if any) of the next statement.

Location 32B (SELIST) holds the address of the start of the IF expression in the ELIST block.

IFRPF is a flag initialized by IF but used by ARITH to differentiate between the legal and illegal occurrence of a right paren followed by a constant (e.g., IF (A+B) 1, 2, 3 and IF (A+B) 1, 2, 3).

5. Subroutines used by IF:

External routines:

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 14.3
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

5.1 DOLABR: Called to inform DOPROC of a reference to a statement label.

5.2 All executable statement processors other than DO and LOGICAL IF to process the statement following a Logical IF expression.

Local routines used by both ARITH and IF (see ARITH documentation for descriptions):

5.3 MACOUT: (to make entry to ARLIST).

5.4 DARLIST: (to send ARLIST entries to RLIST).

Local routines:

5.5 ARITH: Called to translate the IF expression to RLIST. (Only local so far as being found in the same subprogram).

5.6 IFBRT: Syntax checks the E-element addressed by SELIST which should be a statement label, and returns with the label in BCD form in X6 and IFTS2.

5.7 OUTBR: Calls DOLABR, then MACOUT to output a branch instruction to ARLIST.

6. Formats: There are no tables in IF. IF outputs RLIST macros 321B through 330B. See MACROX for descriptions.

7. Modification facilities: EQUs are used for diagnostic ordinals, MACROX ordinals, and lower memory cell locations. Diagnostic macros are used. Only RLIST macros are output to RLIST.

8. Method:

IFE (two and three branch IFs): ARITH is called to translate the IF expression to RLIST. If it is a three branch IF, the expression type is checked to see that it is not type Logical. A macro to branch-on-zero to the 2nd label is output to ARLIST. If the first label is not the same as the label on the next statement, a macro to branch-on-negative to the first label is output to ARLIST. If the third label is not the same as the label on the next statement, a macro to branch unconditionally to the third label is output to ARLIST. Then, IFOUT is entered: An end-of-sequence instruction is output to ARLIST to insure that none of the following statement is evaluated before all tests for the IF have been completed, i.e., consider

IF(A) 1, 2, 1

1 B =C/A

Finally, ARLIST is output to RLIST.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 14.4
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

If it is a two branch IF, the first label is compared to the label on the next statement. If it is not the same a macro to branch on negative (.TRUE.) or non-zero to the first label is output to ARLIST depending on whether the expression is logical or non-logical respectively. If the second label is not the same as the label on the next statement, an unconditional branch is output to ARLIST. Then IFOUT is entered.

IFL(Logical IFs):

ARITH is called to translate the IF expression. If the statement following the IF expression is

GO TO k where k is a statement label,

then a branch on negative (.TRUE.) to k is output to RLIST and IFL exits through its entry point. If the statement is not GO TO k, a branch-on-plus (.FALSE.) to a generated label is output to RLIST followed by an end-of-sequence macro. Then a vector table is entered (biased by the statement type code found in LTYPE) which results in calling the statement processor if it is an executable statement other than a DO or Logical IF. (Otherwise, ERPRO is called). On return from the called statement processor IFL outputs an RLIST psuedo-op defining the generated label. IFL then exits through its entry point.

The only time a three branch IF will jump to the following statement is when the second label is the same as the label on the next statement.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 15.1
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

ENDPROC

1.0 General

1.1 ENDPROC is called by PH2CTL when SCANNER detects the END card of a subprogram. ENDPROC performs the following functions:

- a. Call DOEND to close out DO lists.
- b. Generates the proper exit code placing it into R-list, then issuing an end-of-R-list operation code.
- c. Issue BSS storage for all usage defined variables.
- d. Issue EXT pseudo ops for all external symbols.
- e. Issue informative diagnostics for undefined variables.
- f. Condense symbol table from two words/entry to one word/entry.
- g. Move CONLIST, terminated by a "USE CODE.", to the COMPS buffer.
- h. Append special character to external library names for co-existence problems with previous versions.

ENDPROC calls DOEND, SYMBOL, WRWDS, STR, and ERPRO.

2.0 Usages

2.1 Entry Point Name: END

2.1.1 ENDPROC processes the END card of a subprogram.

2.1.2 ENDPROC is entered via a return jump and upon completion of its tasks, exits through its entry point.

2.1.3 ENDPROC processes the first entry in the symbol table (Program/Subprogram Name) separately from the rest of the symbol table. After determining the type of program being processed, the proper exit code is issued to R-list; the routine then enters the main loop to process the rest of the tasks listed in Section 1.1. A buffer is built in working storage of the EXT's and the BSS's for usage defined variables.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. _____ 15.2
PRODUCT NAME _____ FORTRAN Extended Version 2.0 _____
PRODUCT MODEL NO. _____ 3PC08 _____ MACHINE SERIES _____ 64/65/6600 _____

2.1.3 (continued)

When the end of the symbol table is reached, CONLIST is converted to display code and also written into the working storage buffer in the form of DATA pseudo ops, along with a "USE CODE." to terminate the list. The buffer is then moved to the COMPS file and the routine exits through its entry point.

2.2 Entry Point Name: SCHBET

2.2.1 SCHBET appends a special character to the FORTRAN Extended external library functions.

2.2.2 SCHBET is entered via a direct jump and expects X6 to contain the name to be searched for left justified in the upper 42 bits, with the lower 18 bits all zero. SCHBET expects the return address in B7.

2.2.3 The routine compares the name in X6 with the names in a table of externals. If the name is found, a special character . or \$ is appended to the name in X6. If the name is not found, X6 is unaltered. In either case the routine exits by jumping to the address specified in B7.

3.0 Diagnostics

3.1 No fatal to compilation errors are detected.

3.2 No fatal to execution errors are detected.

3.3 One informative diagnostic may be issued:

"-name- UNDEFINED VARIABLE, I.E. THIS VARIABLE IS NEVER
INITIALIZED.

3.4 No non-ASA errors are detected.

4.0 Environment

When ENDPROC is called, it is expected that symbols necessary to the sub-program have been entered in the symbol table and all fields in

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. _____ 15.3
PRODUCT NAME _____ FORTRAN Extended Version 2.0 _____
PRODUCT MODEL NO. _____ 3PC08 _____ MACHINE SERIES _____ 64/65/6600 _____

4.0 (continued)

the entry have been set accordingly.

Of particular interest are the entries made by DECPRO when processing a subroutine or function subprogram name.

- a. For a subroutine with parameters, the name is entered and typed as a subroutine; ENTRY. is entered and defined as the entry point; and TEMPAO. is entered and defined in the symbol table.
- b. For a subroutine without parameters, the name, and ENTRY. are entered, but TEMPAO. unless there is a RETURNS list is not entered in the symbol table.
- c. For a function subprogram, the name is entered with the proper type, and the ENT bit is set; ENTRY. is entered and defined as the entry point; TEMPAO. is entered and defined; VALUE. is entered, defined, and typed the same as the function subprogram name.
- d. For main program END. is entered in the symbol table and marked as external.
- d. For all programs ST., QT, and DO. are entered in the symbol table with the U bit set and RB=3.

5.0 Structure

5.1 Major Subroutine Name: END

- 5.1.1 Upon entry to END a return jump is immediately made to DOEND to close out all DO lists. Then the first entry of the symbol table (Program/subprogram name) is loaded into X1 and X2 and the program type is determined. If a function subprogram is being processed and the last statement was not a transfer or a RETURN, X1 and X2 are saved and a call is made to SYMBOL to obtain the symbol table ordinal for VALUE.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 15.4
PRODUCT NAME _____ FORTRAN Extended Version 2.0 _____
PRODUCT MODEL NO. 3PC08 _____ MACHINE SERIES 64/65/6600 _____

5.1.1 (continued)

Then a reference is placed into R-list to expand the RMACRO LDVAL. for a single precision function, or LDVAL1. for a double precision function. X1 and X2 are restored, and the necessary registers are set to go to the main loop to process the rest of the symbol table. If a BLOCKDATA subprogram is being processed, the END line is written to the COMPS file and ENDPROC exits through its entry point. If a subroutine is being processed and the last statement was not a transfer or a RETURN, X1 and X2 are saved and a jump is made to SYMBOL to find TEMP AO. If TEMP AO was not previously in the symbol table, a reference is set into R-list to expand the RMACRO SUBRWO which sets up a jump to ENTRY, in R-list. If TEMP AO, was in the symbol table, a reference is set into R-list to expand the RMACRO SUBRW which generates code to restore A0 and then a jump to ENTRY. After either case X1 and X2 are restored, registers are set, and the main loop is entered. For a subroutine or function, if the last statement in the subprogram was a transfer or a RETURN, it is not necessary to generate any of the RMACROS but merely to issue the end-of-R-list operation code, set necessary registers, and go to the main loop.

If a main program is being processed, a return jump to END. is placed into R-list, the R-list is ended, registers initiated, and entry is made to the main loop.

5.2 Major subroutine Name: PROSYM

5.2.1 This is the start of the main loop for processing and squeezing the symbol table.

Each symbol table entry is examined separately.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS TMS PAGE NO. 15.5
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.2.1 (continued)

Examination of the symbol table entries proceeds as follows: First, if the type field of the entry is greater than 5 flow falls directly through to the squeeze loop, SQUEEZE. If the entry is a label SQUEEZE places bits 59-48 of word B into bit positions 11-00 of word A. If the entry is a symbol bits 24-13 of word B are placed into bit positions 11-00 of word A. This combined word is then stored into the next location of the one word symbol table, the next entry is loaded.

If the type is 0 through 5 other conditions must be checked. If the symbol is external, not a formal parameter, and is a basic external function a decimal point (.) is appended to the name in the symbol table, an EXT pseudo op is put into the COMPS buffer and SQUEEZE is entered. If the symbol is external, not a formal parameter, and is not a function or not a basic external function then SCHBET is entered to determine if a special character (\$) need be appended.

Whether the character is appended or not, the EXT pseudo is put into the COMPS buffer and SQUEEZE is entered.

If a symbol has not been defined in the symbol table, the symbol is placed into X4 in E-list form, necessary registers are saved, and a jump is made to ERPRO to enter the variable and diagnostic in the error table. Upon return, registers are restored and the loop is re-entered. As the diagnostic is only informative, a BSS 1 (or 2) is sent to the buffer for the undefined variable.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. _____ 15.6
PRODUCT NAME _____ FORTRAN Extended Version 2.0 _____
PRODUCT MODEL NO. _____ 3PC08 _____ MACHINE SERIES _____ 64/65/6600 _____

5.2.1 (continued)

If a variable has been defined by usage only, a BSS 1 (or 2 for two word variables) is stored into the buffer for that variable to be output to COMPS later.

5.3 Subroutine NAME: CONS

5.3.1 The one word symbol table is saved in lower addressed memory with enough room between it and the two word symbol table to hold the generated label definition (one word for each generated label).

The head and end addresses of the one word symbol table are held in SYM1 and SYMEND respectively while the head and end of the two word symbol table are held in SYM1A and SYMENDA.

CONLIST is then converted to display code octal constants and put into the buffer area.

A "USE CODE." is placed in the buffer and the entire buffer moved to COMPS via WRWDS.

If at any time the COMPS buffer being formed will exceed the available working storage (at this time the area from CONLAST to SYMEND), a return jump is made to RITCOMP to save registers, dump the buffer to the COMPS file, and restore or reinitialize the registers.

6.0 Not applicable.

7.0 ENDPROC expects the proper addresses in SYM1 (RA+12B). SYMENT (RA+13B), CON1, (RA+26B), and CONLAST (RA+27B). PH2CTL must set LSFLG (RA+44B) to zero if the last statement in the subprogram was not an unconditional transfer or a RETURN, or to non-zero if the above condition does exist. DECPRO must set PROGRAM (RA+56B) equal to the program name packed into

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 15.7
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

7.0 (continued)

the cell with a bias of 2000B if a main program. A subprogram will have the number of arguments in the lower 48 bits and a bias of 2001B for a subroutine, or a bias of 2002B for a function.

M01, M02, M03, and M04 are set equal to the number of the RMACRO for a single precision function subprogram, a double precision function subprogram, a sub-routine with parameters, or a subroutine without parameters respectively.

8.0 The method used in the task is discussed at length in Section 5.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 16.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

RTNPROC

1.0 General Information

1.1 RTNPROC is called by PH2CTL when SCANNER encounters a RETURN statement. RTNPROC performs the following functions:

- a) Checks for illegal use of the RETURN (or RETURN a) statement.
- b) Determines type of return (standard or non-standard), and
- c) Generates proper RETURN coding to R-list.

RTNPROC calls ERPRO, SYMBOL, STR, WRWDS, and ASAER.

2.0 Usage

2.1 Entry Point Name: RETURN

2.1.1 RTNPROC processes all RETURN statements.

2.1.2 RTNPROC is entered via a return jump and upon completion of its task (or upon detecting an error), exits through its entry point.

2.1.3 RTNPROC checks first to see if the RETURN statement has been used legitimately. Then the type of subprogram is determined, and a reference to an RMACRO is placed into R-list to generate the necessary R-list code. See Section 5.0 for detailed flow.

3.0 Diagnostics

3.1 Fatal to compilation: none.

3.2 RTNPROC may issue one of three fatal to execution diagnostics:

- a) RETURN STATEMENT APPEARS IN A MAIN PROGRAM. (#67)
- b) NON-STANDARD RETURN STATEMENT MAY NOT APPEAR IN A FUNCTION SUBPROGRAM (#63).

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 16.2
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- c) PARAMETER ON NON-STANDARD RETURN STATEMENT IS NOT A RETURNS FORMAL
PARAMETER. (#69)

3.3 None

3.4 Non-ASA Diagnostic

- a) THE NON-STANDARD RETURN STATEMENT IS NOT AMERICAN STANDARD FORTRAN,
(#72)

4.0 Environment

When RTNPROC is entered the following entries are expected to have
been made in the symbol table by DECPRO:

- a) For a subroutine with parameters - the name is entered and typed
as a subroutine; ENTRY. is entered and defined as the entry point;
and TEMPAO. is entered and defined.
- b) For a subroutine without parameters - the name, and ENTRY. are
entered, but TEMPAO. is not entered in the symbol table.
- c) For a function subprogram - the name is entered with the proper
type, and the ENT bit is set; ENTRY. is entered and defined as the
entry point; TEMPAO. is entered and defined; VALUE. is entered,
defined, and typed the same as the function subprogram name.

See Section 7.0 for the list of common cells and the information
expected in each.

5.0 Structure

5.1 Major Subroutine Name: RETURN

- 5.1.1 Upon entry a check is immediately made to see if a main program is
being compiled. If it is a main program, ERPRO is called to issue
diagnostic #67, then the routine exits. If we are processing a

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 16.3
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

a standard return in a subroutine which has formal parameters, R-list code is issued to reference the RMACRO SUBRW to restore A0 and jump to the entry point. If the subroutine has no parameters, RMACRO SUBRWO is referenced to set up just a jump to the entry point.

For a standard return in a function subprogram entries are placed in R-list to reference RMACRO LDVAL. to load a single precision function value to X6, or to reference RMACRO LDVAL1. to load a double function value to X6 & X7. Both of these RMACRO's then generate code to restore A0, and jump to the entry point.

If we are processing a non-standard return statement in a subroutine, and the RETURNS parameter is in the symbol table with type equal to 7, the entries are placed in R-list to reference RMACRO NSRTRN to load the RETURNS entry in APLIST into B1, restore A0, and then jump to contents of B1. If the RETURNS parameter is not in the symbol table or if the type is not equal to 7, ERPRO is called to issue diagnostic #69, then the routine exits.

If the non-standard return statement occurred in a function subprogram, ERPRO is called to issue diagnostic #68, then RTNPROC exits.

6.0 Not applicable.

7.0 RTNPROC expects the proper addresses in DOLAST (RA+31B), and SELIST (RA+32B). DECPRO must pack the program name with a bias of 2000B into PROGRAM (RA+56B) if processing a main program. If processing a sub-program DECPRO places the number of arguments in bits 0-47 of PROGRAM

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 16.4
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

with a bias of 2000B for a subroutine, or a bias of 2002B for a function.

MORD1, MORD2, MORD3, MORD4, and MORD5 are set equal to the ordinal of the RMACRO for a single precision function subprogram, a double precision function subprogram, a subroutine with parameters, a subroutine without parameters or a non-standard return statement respectively.

8.0 Method

The method used for the non-standard RETURN is as follows:

Processing of the subroutine header card will cause the RETURNS formal parameter to be placed in the symbol table with type = 7, and numbered consecutively (in bits 39-34 of word 2 of the entry) beginning with zero,

For the nominal case where no formal parameters are declared with the subroutine name, an APLIST of just the RETURNS formal parameters is formed, terminated by a zero entry. If there are formal parameters they are entered in the APLIST followed by a zero entry; then the RETURNS formal parameters are entered (one word per entry) and followed by a zero word.

The number of formal parameters (n) declared in the SUBROUTINE header card is added to the ordinal obtained from word 2 of the symbol table entry for the RETURNS parameter. If n = 0, then this sum is the ordinal (i) passed to the NSRTRN RMACRO. If n = 0, then $i=i+1$ must be passed to the RMACRO to compensate for the extra zero word in APLIST.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 16.5
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

The RMACRO NSRTRN generates:

$$SA_j = AO + i$$

$$SB1 = X_j$$

$$SA_k = TEMPAO.$$

$$SAO = X_k$$

JP B1

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
 _____ DIVISION

DOCUMENT CLASS IMS PAGE NO. 17.1
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

NAMELIST

- 1.0 General
- 1.1 NAMELIST processes the NAMELIST statement, and is part of Phase 2, Pass 1.
- 2.0 NAMELIST has one entry point.
- 2.0.1 NAMELIST processes the NAMELIST statement.
- 2.0.2 NAMELIST is entered from PH2CTL via a return jump and exits through its entry point or to FATALER if there is insufficient memory to process the statement. A flag, MACROIN, is checked to determine whether the NAME macro definition has been written onto the COMPS file. If not, it is written before any NAMELIST processing. The macro is as follows:

```

NAME      MACRO  N,T,BASE,BIAS,FP,D1,D2,D3
LOCAL    Z1,Z2,Z3,Z
VFD      18/0,42/0LN
VFD      6/1,6/0
IFC      NE,#BASE##,1
VFD      18/BASE+BIAS
IFC      EQ,#BASE##,1
VFD      18/N
IFC      NE,#FP##,1
SUB      N
VFD      30/T
IFC      EQ,#D1##,1
Z        SET    0
IFC      NE,#D1##,1
Z        SET    1
IFC      NE,#D3##,4
Z1       SET    D1
Z2       SET    D2
Z3       SET    D3
IFEQ     0,1
IFC      NE,#D2##,4
Z1       SET    0
Z2       SET    D1
Z3       SET    D2
IFEQ     0,1
IFC      NE,#D1##,4
  
```

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 17.2
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

```
Z1 SET 0
Z2 SET 0
Z3 SET D1
    IFEQ 0,1
Z1 SET 0
Z2 SET 0
Z3 SET 1
ENDIF
VFD 6/Z,18/Z1,18/Z2,18/Z3
ENDM
```

2.0.3 The NAMELIST statement is syntax checked and processed at the same time from the E-list form. This process causes BSS storage to be defined for previously undefined variables, and macro calls to be formed, which will reference the NAME macro. The BSS storage code is preserved in NAMELIST, and is written to COMPS when the reserved area in NAMELIST is full, and when a NAMELIST group has been completely processed. The macro reference code is preserved in temporary storage, beginning at DOLAST+1. This information is written onto the COMPS file after a NAMELIST group has been completely processed, and any BSS storage code has been written onto the COMPS file. A NAMELIST group set of information ends with a zero word.

When NAMELIST exits, the following will have been accomplished:

1. Each NAMELIST group name will have been entered into the symbol table with mode set to NAMELIST.
2. Each group variable that was not in the symbol table will have been entered and will have its mode set according to natural typing.
3. Each group variable that was not dimensioned, equivalenced, or in COMMON, will cause BSS storage to be issued, the number of words of storage allocated is added to DATA, and the common and defined bits are set in the symbol table.
4. For each NAMELIST group the following information will have been generated by the NAME macro for the COMPS file.

Word 0: The NAMELIST name in display code, zero-filled, left-justified in the lower 42 bits.

•
•
•

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS TMS PAGE NO. 17.3
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

Word 3K-2: the name of the k^{th} associated variable, in display code, zero-filled, left-justified in the lower 42 bits.

Word 3K-1: 1 in bits 59-54 {indicates FWA_k is the address of a variable's first memory location}.

Zero in bits 53-48

FWA_k in bits 47-30

T_k in bits 29-0 {all right-justified within the allocated bits}

FWA_k is the address of the variable's first memory location.

T_k is the type of the variable: 1=logical
2=integer
4=real
5=double
6=complex

Word 3K: Zero in 59-54 if variable not dimensioned; 1 if dimensioned.

$M2_k$ in 53-36

$M1_k$ in 35-18

LNG in 17-0

$M2_k$ is: the first dimension of a three-dimensional array. 0 otherwise.

$M1_k$ is: the product of the first two dimensions of a three-dimensional array; the first dimension of a two-dimensional array; 0 otherwise.

LNG is: the number of elements {not necessarily the number of computer words} of an array. 1 for a variable.

•
•
•

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 17.4
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

Word 3N+1: 0 {where N is the number of variables associated with the NAMELIST name}.
Formatting NAMELIST information in this manner is to facilitate the implementation of the RUN 2.0 and 2.3 NAMELIST I/O routines.

3.0 Diagnostics

3.1 Fatal to Compilation

None

3.2 Fatal to Execution

3.2.1 THERE IS INSUFFICIENT ROOM TO PROCESS THIS STATEMENT.
Issued if there does not appear to be enough room to process the statement and convert it to COMPASS line images.

3.2.2 THERE IS AN ENTRY IN THIS NAMELIST STATEMENT OTHER THAN A SLASH, A COMMA, OR A VARIABLE.

3.2.3 ENTRY FOLLOWING A SLASH IS EITHER NOT A VARIABLE OR A VARIABLE THAT HAS BEEN PREVIOUSLY USED.

3.2.4 NAMELIST GROUP NAME IS NOT SURROUNDED BY SLASHES.

3.2.5 IN ONE OF THE NAMELIST GROUPS IN THIS STATEMENT, THERE IS AN IMPROPER ENTRY IN PLACE OF A VARIABLE.

3.2.6 'Variable name' - NAME OF A VARIABLE THAT HAS A MODE OTHER THAN LOGICAL, INTEGER, REAL, DOUBLE, OR COMPLEX.

3.2.7 'Variable name' - THIS VARIABLE HAS VARIABLE DIMENSIONS. THIS IS NOT ALLOWED IN CONJUNCTION WITH NAMELIST.

3.3 Informative

None

3.4 Non-USASI

All NAMELIST statements receive the NON-USASI diagnostic.
'THIS STATEMENT IS NON-USASI.'

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 17.5
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

4.0 Environment

4.1 The following cells are referenced and expected to be set accordingly:

4.1.1 SELIST - address of the next entry in E-LIST.

4.1.2 ELAST - address of the end of E-LIST.

4.1.3 DOLAST - address of the end of the D0 tables.

4.1.4 DIM1 - address of the start of the dimension list.

4.1.5 SYM1 - start of the symbol table.

4.2 The cell SUBREF {RA+16B} is set non-zero if NAMELIST issued a SUB macro reference.

5.0 Structure. The processing of NAMELIST statements is accomplished in one pass for each NAMELIST group. A NAMELIST group must begin with a series of three entries: a slash, followed by a variable, followed by a slash. A NAMELIST variable list must then follow, where the variables are separated by commas.

5.1 The variables associated with the NAMELIST name are entered into the symbol table if not already there. If previously defined the variable is checked for acceptable mode. If not, the variable is typed, using natural typing, BSS storage is issued, and the common and defined bits are set for the symbol table entry. The common bit is set to inform ENDPROC not to issue storage, since the NAMELIST processor has already issued storage.

5.2 If the variable is equivalenced, the base and bias are formed for use in the macro reference. If the variable is a formal parameter, this information is saved for the macro reference.

5.3 The macro reference is formed in the following manner:

NAME N,T,BASE,BIAS,FP,D1,D2,D3

where N = variable name

T = type, i.e. 1=logical, 2=integer, 4=real, 5=double, 6=complex

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 17.6
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

BASE = base name of equivalence class

BIAS = pointer, relative to base, for this
equivalence class element

FP = 1 implies this variable is a formal parameter

D1 = dimension 1 of an array

D2 = dimension 1 times dimension 2 of an array

D3 = the number of elements of a 3 dimension array

Only the necessary parameters are passed, i.e. for a
simple variable the macro reference would be:

NAME N,T

For a one dimension array the macro reference would be:

NAME N,T,,,,D1

The structure of the NAMELIST group information passed to
the COMPS file would be as follows:

A Any BSS storage allocation for this group

B NAMELIST group name BSS 0

C VFD 18/0,42/0L °NAMELIST group name°

D All macro references for the group

E DATA 0B

6.0 FORMATS

6.1 The following internal cells are used during NAMELIST
processing:

6.1.1 PNEXT - next position available in working storage.

6.1.2 DNEXT - next position available in NAMELIST for BSS
storage.

6.1.3 VARNAME - contains variable name - eight characters,
right adjusted, with blank fill.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 17.7
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- 6.1.4 MACROIN - flag used to determine whether the NAME macro definition has been written onto the COMPS file.
- 6.1.5 DATASKEL - two word skeleton in which each BSS storage issue is formed.
- 6.1.6 DATABSS - forty word area in which BSS storage issued is temporarily accumulated for the COMPS file.
- 6.1.7 MACSKEL - seven word area in which macro calls are built.
- 6.1.8 AUXSTORE - first available position in temporary storage - this is where the macro calls begin.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 181
PRODUCT NAME FORTAN Extended
PRODUCT MODEL NO. 4P116 MACHINE SERIES 6400/6500/6600

1.0 GENERAL

1.1 PRINT process the I/O type statements READ, WRITE, PRINT, PUNCH, ENCODE, DECODE, BUFFER IN, BUFFER OUT, END FILE, BACKSPACE and REWIND. It calls the D0 processor to process all implied D0 loops, and call ARITH to generate the addresses for all variables in I/O lists. The processor is located in Phase 2 of the compiler.

2.0 Entry point names.

2.1 PRINT

This entry point is entered when a statement of the form PRINT f or PRINT f, k is encountered.

2.2 READ

This entry point is entered when a statement of the form READ {u} k or READ {u, f} k or READ {u, f} is encountered.

2.3 PUNCH

This entry point is entered when a statement of the form PUNCH f or PUNCH f, k is encountered

2.4 WRITE

This entry point is entered when a statement of the form WRITE {u} or WRITE {u, f} or WRITE {u, f} k is encountered.

2.5 BUFIN

This entry point is entered when a statement of the form BUFFER IN {u, p} {A, B} is encountered.

2.6 BUFOUT

This entry point is entered when a statement of the form BUFFER out {u, p} {A, B} is encountered.

2.7 ENDFILE

This entry point is entered when an statement of the form ENDFILE u is encountered.

2.8 REW

This entry point is entered when a statement of the form REWIND u is encountered.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 18.2
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 6400/6500/6600

2.9 DEC

This entry point is entered when a statement of the form
DECODE {c, n, v} k is encountered.

2.10 ENC

This entry point is entered when a statement of the form
ENCODE {c, n, v} k is encountered.

2.11 BKSP

This entry point is entered when a statement of the form
Backspace u is encountered.

2.12 D0ITX

This entry point is used by the D0 processor when returning
to the PRINT processor after being called to set up an
implied D0 loop.

2.13 D0G00FX

This entry point is used by the D0 processor when returning
to the PRINT processor after being called when an error
has occurred.

2.14 D0NEX

This entry point is used by the D0 processor when returning
to the PRINT processor after being called to generate the
final coding for an implied D0 loop.

2.15 EXTERNAL SYMBOLS

2.15.1 IXFN

This routine is called to generate the address for all
variables in I/O lists. SELIST contains the variable that
the address is to be generated for. Upon return X2 contains
the ordinal of the identifier in the SYMBOL table.

2.15.2 LABEL

This routine is called to enter labels into the SYMBOL
table.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 18.3
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 6400/6500/6600

2.15.3 WRWDS

This routine is called to transfer R-List into the R-List buffer.

2.15.4 ERPR0

This routine is called to enter diagnostics into the error table.

2.15.5 D0CALL

This routine is called to inform the D0 processor that a jump to an external procedure will be made.

2.15.6 D0IT

This routine is called when it is necessary to set up an implied D0 loop. The address of the equal sign for the implied D4 being processed is contained in B1.

2.15.7 D0G00F

This routine is called when an error situation occurs. The D0 processor will close out any D0 loops which may be in effect.

2.15.18 DONE

This routine is called to generate the terminal code for the implied D0 loop. SELIST points to the D0 control variable.

2.15.9 STR

This routine is used send one word to the R-List buffer.

2.15.10 ASAER

This entry point is called whenever a non-USASI I/O statement appears.

2.15.11 PH2RETN

This entry point is called after a fatal error is diagnosed.

2.15.12 SYMBOL

This entry point is used to enter and find symbols in the symbol table.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 18.4
PRODUCT NAME F0RTRAM Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 6400/6500/6600

2.15.13 D0DEF

This routine is called whenever an integer variable definition occurs.

3.0 Diagnostics produced

3.1 Fatal to compilation. None.

3.2 Fatal to execution

3.2.1 ILLEGAL SEQUENCE IN I/O LIST. The parameter associated with this diagnostic indicates the element which caused the processor to stop processing the statement.

3.2.2 UNIT NUMBER OR PARITY INDICATOR MUST BE AN INTEGER CONSTANT OF VARIABLE.

3.2.3 NUMBER OF CHARACTERS IN AN ENCODE/DECODE STATEMENT MUST BE AN INTEGER CONSTANT OR VARIABLE.

3.2.4 FORMAT REFERENCE MUST BE AN INTEGER CONSTANT OR AN ARRAY REFERENCE.

3.2.5 THE VALUE OF THE PARITY INDICATOR IN A BUFFER I/O STATEMENT MUST BE 0 OR 1.

3.3 INFORMATIVE. None.

3.4 Non-USASI

THIS FORM OF AN I/O STATEMENT DOES NOT CONFORM TO USASI SPECIFICATIONS

4.0 STRUCTURE

4.1 PROCOM

This routine is used to control the processing of both the PRINT and PUNCH statement since, except for the object time routine being called, the processing required is identical.

4.2 FINIS

This routine is entered to set up the terminating call to execution I/O routines for all I/O lists.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 18.5
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. _____ 4P616 _____ MACHINE SERIES 64/65/6600

4.3 GETORD

Enter with name in X1. This routine calls SYMBOL to enter the two entry points used for each I/O routine, and sets the EXT bit. It then generates a RJ {with error trace} to the appropriate entry point.

4.4 IOLIST

Performs a partial syntax check on all I/O lists. When a left parenthesis is encountered {not including subscripted references} a syntax check is made until the corresponding right parenthesis is encountered. If no errors are found when the zero level right parenthesis is encountered, a call to D0IT is made to set up an implied D0 loop.

4.5 IDENT

This routine is called by IOLIST when it {IOLIST} has determined there is a variable to process. In the case of

{ {A{K}, B{J}, K=1, 10}, J=1, 10}

five calls would be made to IDENT from IOLIST the E-list pointer the five cases are pointing a A, B, K, J. In the case of K and J, IDENT will look ahead and determine the next operator is an equal sign and call DONE to generate the terminal D0 loop code. IDENT determines whether the identifier is a variable, array element, array name, and the type. The address of the identifier is obtained by making a call to IXFN. IDENT then generates the appropriate R list.

4.6 PROUF

This routine determines the mode of the I/O statement and calls a routine {FMTN0} to process the format number if it is necessary.

4.7 PROU

Process unit. This routine is used for two functions. {1} process unit number for all statements and {2} process- ing the parity indicator for BUFFER IN/OUT statements.

4.8 TAPEN

If the unit number is constant, TAPEN is entered to gener- ated R list code for SB2 TAPEN.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 1A.6 _____
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. _____ 4P616 _____ MACHINE SERIES 64/65/6600 _____

4.9 TESETI

This routine is called when the macro TESET is referenced. Its purpose is to generate a type III R-list entry followed by a register stace. The parameter in the macro reference is the S0 field of the R-list entry.

4.10 REGCOMP

This routine is called when the macro REGCOM is referenced. Its purpose is to generate R-list instructions to complement a specific {SB2-B2}. The parameter in the macro reference is the S field of the R-list entry.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 19.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

ASSIGN

- 1.0 General
 - 1.1 Assign processes the ASSIGN statement, and is a part of PHASE 2 of Pass 2.
- 2.0 ASSIGN
 - 2.0.1 Processes the "ASSIGN label TO variable" statement.
 - 2.0.2 ASSIGN is entered via a return jump, and exits through its entry point.
 - 2.0.3 The statement is Syntax checked. If correct, R-LIST entries are made to accomplish the following:
 - SXi statement label's address
 - SAi variable - store the address
- 3.0 Diagnostics
 - 3.1 No fatal to compilation diagnostics.
 - 3.2 Fatal to Execution
 - 3.2.1 THIS ASSIGN STATEMENT HAS IMPROPER FORMAT (ONLY ALLOWABLE FORM IS ASSIGN LABEL TO VARIABLE).
 - 3.3 No informative diagnostics
 - 3.4 ASA
 - 3.4.1 THE VARIABLE IN THIS ASSIGN STATEMENT IS NOT OF TYPE INTEGER.
- 4.0 The following cells are referenced and expected to be set accordingly.
 - 4.0.1 SELIST address of the first entry in ELIST.
 - 4.0.2 NRLN value to use for the next Register number.
 - 4.0.3 DIM1 contains first word address of the dimension table.
 - 4.0.4 CLABEL contains the current statement label
 - 4.1 The following routines are called:

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 19.2
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

- 4.1.1 WRWDS used to transfer words to the R-LIST file.
- 4.1.2 SYMBOL used to find/or enter entries into the symbol table.
- 4.1.3 DOLABA used to enter the label into the symbol table and get its ordinal.
- 4.1.4 ERPRO used to issue fatal to execution diagnostics.
- 4.1.5 ASAER used to issue non-USASI diagnostics.

- 5.0 STRUCTURE The processing is straight forward.
- 5.1.1 The current label is saved and DOLABA is called, if the E-LIST entry is a constant, to enter the label in the symbol table and get its ordinal.
- 5.1.2 The statement is then checked for syntactical errors.
- 5.1.3 The instructions for setting a register to the address of the label are formed and saved.
- 5.1.4 The portion of the variable is then checked to see that a variable is there and SYMBOL is called to enter it in the symbol table.
- 5.1.5 If it is equivalenced, that is handled at this time, by referencing the base plus a bias (if any).
- 5.1.6 The store instruction is then formed and WRWDS is called to place these entries in RLIST.
- 5.1.7 The non-USASI diagnostic is issued if the variable is not of type integer.

- 6.0 Formats - no flags are used.
- 6.1 FIRST and SECOND contain the packed op code for the set and store instructions.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 19.3
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- 7.0 **Modification**
- 7.1 All cells referenced are set by EQU's.
- 8.0 **Method - straight forward.**
- 9.0 **No cautions to modifier.**

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 20.1
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

ENTRY

- 1.0 General
 - 1.1 ENTRY is in Phase 2 of Pass 1 and performs the following operations.
 - 1.1.1 Processes the ENTRY statement.
 - 1.1.2 Places the entry macro ENTR. in the COMPASS file if it is not already there.
 - 1.1.3 Places the reference to the entry macro in the R-list file.
 - 2.0 ENTRY has one entry point
 - 2.1 ENTRY
 - 2.1.1 ENTRY processes the ENTRY statement.
 - 2.1.2 ENTRY is entered with a return jump and exits through its entry point. It processes one entry statement.
 - 2.1.3 An 'entry' entry {0C=55} is placed in the R-list by calling STR. If the entry macro has not yet been placed in the COMPASS file, it is via WRWDS.
 - 3.0 Diagnostics Produced
 - 3.1 No fatal to compilation diagnostics are produced.
 - 3.2 Fatal to Execution
 - 3.2.1 ENTRY STATEMENT IS NOT ALLOWED TO APPEAR IN A PROGRAM, ONLY IN A SUBROUTINE OR FUNCTION.
 - 3.2.2 IMPROPER FORM OF ENTRY STATEMENT. ONLY ALLOWABLE FORM IS {ENTRY NAME}.
 - 3.2.3 ENTRY POINT NAMES MUST BE UNIQUE - THIS ONE HAS BEEN PREVIOUSLY USED IN THIS SUBPROGRAM.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 20.2
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- 3.2.4 STATEMENT NUMBER IS NOT ALLOWED ON AN ENTRY STATEMENT.
- 3.4 No informative diagnostics.
- 3.5 USASI diagnostics
 - 3.5.1 The following diagnostic is always given: THE ENTRY STATEMENT IS A NON-USASI STATEMENT.
- 4.0 The following cells are used and expected to be set accordingly.
 - 4.1.1 SELIST Address of next entry in ELIST
 - 4.1.2 ELAST Address of last entry in ELIST
 - 4.1.3 CLABEL Label of the current statement
 - 4.1.4 PGM Program/subprogram flag word
 - 4.2 The following routines are called.
 - 4.2.1 WRWDS To transfer words to the compass file
 - 4.2.2 STR To transfer one word to the R-list file
 - 4.2.3 D0ENT To inform D0 of an entry statement
 - 4.2.4 SYMBOL To handle the symbol table
 - 4.2.5 ERPRO To handle any fatal to execution errors detected in the statement
 - 4.2.6 ASAER To give the non-USASI diagnostic
 - 4.2.7 SCHBET To determine if the ENTRY is the same as any intrinsic or basic external name, and make unique by suffixing a \$ if so.
- 5.0 The CLABEL and PGM words are checked to make sure the ENTRY statement does not have a label and that it is not appearing in a program.
- 5.1 If this is the first ENTRY statement, one of the following ENTRY macros will be transferred via WRWDS to the COMPASS file.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 20.3
PRODUCT NAME FORTTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.1.1 Macro for a subprogram with no arguments

```
ENTR. MACRO NAME
      LOCAL T
      EQ T
NAME BSS 1
      ENTRY NAME
      SA1 NAME
      BX6 X1
      SA6 ENTRY
T BSS 0
      ENDM
```

5.1.2 Macro for a subprogram with arguments {includes more than a macro}

```
      USE ENTRY.
FTNNOP. DATA 46000460004600046000B
      USE DATA.
NOPPS. DATA 46000460004600046000B
ENTR. MACRO NAME
      LOCAL X, Z, T
      EQ T
NAME BSS 1
      ENTRY NAME
      SA2 X
      BX6 X2
      SA6 FTNNOP.
      EQ ENTRY.+1
X      EQ Z
Z      SA1 NOPPS.
      SA2 NAME
      BX6 X1
      LX7 X2
      SA6 FTNNOP.
      SA7 ENTRY.
T      BSS 0
      ENDM
```

5.2 The statement is then checked further for proper format.

5.3 The entry name is entered into the symbol table by calling SYMBOL; if it was there already, a diagnostic is given.

5.4 The mode of the name is set to ENTRY statement {12B} and the ENT bit {53} is set.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 20.4
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- 5.5 An entry reference {0C=55B} is placed in the R-list file by calling STR.
- 5.6 The non-ASA diagnostic is given.
- 5.7 For Version 2, if the larger macro is to be referenced, one word each is added to the DATA Δ count and the ENTRY Δ count to account for the no-ops created.

- 6.0 FORMATS:
 - 6.1 ENTRYF - initially zero, set non-zero when the entry macro is transferred to the COMPASS file.

- 7.0 Modification:
 - 7.1 All diagnostic numbers are set by EQU's, as are all low core cells referenced,

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 21.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

GO TO

1.0 General

1.1 The GO TO processor processes all GO TO-type transfer statements except unconditional GO TO's following logical IF'S and generates appropriate R-list code. It is called by PH2CTL and it will call GTARTH, DOLABR, WRWDS and STR for R-list output.

2.0 Usages

2.1 Entry Point Name: GOTO

2.1.1 GOTO processes the GO TO statements.

2.1.2 GOTO is entered via a return jump. If the statement is in correct form, the processor exits through its entry point after the statement has been processed.

2.2 GOTO is to produce the following code:

2.2.1 Unconditional GO TO

EQ "LABEL"

2.2.2 Assigned GO TO

$R_1 \leftarrow$ Variable

$B_1 = R_1$

JP B_1 (B1)

2.2.3 Two branch transfer

a) If neither branches are the same as next statement

$R_1 \leftarrow$ Variable

$R_2 = R_1 - 2$

NG R_2 , 1st branch

PL R_2 , 2nd branch

b) If 1st branch is same as next statement

$R_1 \leftarrow$ Variable

$R_2 = R_1 - 2$

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 21.2
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

2.2.4 Three branch transfer

- a) No branches same as next branch

R_1 ← Variable

$R_2 = R_1 - 2$

NG R_2 , 1st branch

ZR R_2 , 2nd branch

LT BO, R_2 , 3rd branch

- b) If either the second or third branches are the same as the next statement, the appropriate test jump will be deleted.

- c) If the 1st branch is the same, it will be deleted on 6400's but not on 6600's.

2.2.5 More than 3 branches

R_1 ← Variable

$R_2 = R_1 - \text{CONSTANT}_1$

PL R_2 , LAST BRANCH

$B_1 = R_1 - \text{CONSTANT}_2$

LT B_1, BO , FIRST BRANCH

JP $GLn., B_1$

$GLn.$ BSS. 0

EQ 2nd BRANCH

EQ 3rd BRANCH

...

EQ NEXT TO LAST BRANCH

CONSTANT (1) is equal to the number of branches minus the number of times the last branch appeared before a different branch was encountered plus 1.

CONSTANT (2) the number of times the first branch appeared before a different branch was encountered plus 1.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 21.3
PRODUCT NAME FORTAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

EXAMPLE:

GO TO (1, 1, 1, 1, 2, 3, 4, 5, 5, 5, 5) I

would compile into

R₁ ← I

R₂ = R₁ - 10B

PL R₂, .5

B₁ = R₂ - 4

GE B₀, B₁ .1

JP GL44., B₁

GL44.BSS 0

EQ .2

EQ .3

EQ .4

3.0 Diagnostics

3.1 No fatal to compilation diagnostics are produced.

3.2 Fatal to Execution

3.2.1 EITHER THE EXPECTED LIST OF TRANSFER LABELS IS NON-EXISTENT, EMPTY, OR NOT ENCLOSED IN PARENTHESES.

3.2.2 THIS IS NOT A RECOGNIZABLE FORM OF THE GO TO STATEMENT.

3.2.3 THERE IS A NON-NUMERIC ENTRY IN THIS LIST OF TRANSFER LABELS WHEN ONE IS REQUIRED.

3.2.4 IN THIS GO TO WHICH WAS ASSUMED TO BE AN UNCONDITIONAL GO, THERE IS AN ENTRY FOLLOWING THE STATEMENT LABEL.

3.3 Informative Diagnostics

3.3.1 THERE IS AN ENTRY FOLLOWING THE RIGHT PARENTHESIS OF THIS ASSIGNED GO TO LIST.

3.4 Non-USASI

3.4.1 THIS GO TO STATEMENT CONTAINS NON USASI USAGES

CONTROL DATA CORPORATION . COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 21.4
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

4.0 Environment:

The following cells are expected to be properly set.

- 4.0.1 SELIST Address of next entry in E-List.
- 4.0.2 NLABEL Label, if any, on the next statement to be processed.
- 4.0.3 NGLN Number to use for the next generated label.
- 4.0.4 ELAST Address of the end of statement.
- 4.0.5 NRLN Next R number available to use.
- 4.0.6 DOLAST Contains first word address of working storage during PASS 1.
- 4.0.7 LELIST Non-zero if GOTO is object of logical IF.
- 4.0.8 GOTOCEL Set non-zero if GO TO is not object of a logical IF.
- 4.1 The following routines are called.
 - 4.1.1 DOLABR Informs DO of a transfer label. The symbol table ordinal for the label is returned in B₁.
 - 4.1.2 SYMBOL Symbol table entry and search.
 - 4.1.3 STR Enter one word in R List.
 - 4.1.4 WRWDS Enter many words to a specified file.
 - 4.1.5 ASAER Called to issue non USASI diagnostics.
 - 4.1.6 ERPRO Called to issue fatal to execution diagnostics.
 - 4.1.7 GTARTH Is called to handle the i field of the assigned or computed TO TO. It handles expressions, mode changes and equivalence changes. UPON return NRLN contains the R plus 1 that contains the value of the index. (If the i field was not a simple integer variable an USASI diagnostic is issued).

5.0 STRUCTURE: The statement is examined to determine which type of GO TO it is.

5.1 UNCONDITIONAL GO TO: The entry after the statement label is checked and a diagnostic flag is set if it is not end of statement. An EQ LABEL R-list entry is formed in X6 and STR is called to transfer it to the RLIST file. GOTO then exits, giving a fatal error diagnostic, if necessary.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 21.5
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

5.2 ASSIGNED GO TO

5.2.1 The statement is checked for proper format and at the same time DOLABR is informed of the transfer labels.

5.2.2 The variable is placed in a cell within GOTO that is preceded by an end of statement. SELIST is set to this cell and GTARTH is called to compile instructions for the reading of the variable. This was done so the GO TO processor would not have to worry about equivalenced variables.

5.2.3 A reference is placed in the RLIST file, using STR, to the following macro.

```
GO MACRO1      RMACRO          0,1      ASSIGNED GO TO
              SA      I1,0,P1      ONE PARAMETRIC R
              RS      I1,0,B1.     SET I1 TO PARAMETRIC R
              JIN     0,I1,0,0,0    I1 IS TO BE B1
              ENDR                INDEXED JUMP
```

GO TO then exits via its entry.

5.3 Computed GO TO

5.3.1 The list is scanned for correct syntax and to determine the following:

5.3.1.1 The number of branches

5.3.1.2 The number of times the first branch was repeated before another branch intervened.

5.3.1.3 The number of times the last branch was repeated before another branch intervened. The computed GO TO is then broken down into two cases.

5.3.2 TWO or THREE branch GO TO's are special cased in that each branch is compared to the label of the next statement and if they are the same the test instruction is not compiled.

The only special case is under 2.2.4

5.3.3 GTARTH is called to compile instructions to bring the value of the index to a register. The R+1 that contains the value will be in NRLN upon return.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 21.6
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

5.3.4 A macro reference to the following macro is placed in a temporary RLIST buffer.

GOMACRO2 RMACRO 3,1,2 COMPUTED GO TO

THREE IH FIELDS ARE:

1. FIRST BRANCH
2. LAST BRANCH
3. VALUE FOR GENERATED LABEL

PARAMETRIC R IS R THAT HAS INDEX VALUE IN IT. (COMPUTED BY GTARTH)

TWO CONSTANTS ARE:

1. REPEAT COUNT FOR LAST BRANCH
2. REPEAT COUNT FOR FIRST BRANCH

STT	I1,P1,K1	$R_i = R_{(P1)}^{-K1}$
JPOS	I1,0,0,2	PL R_i , LAST BRANCH
STT	I2,P1,K2,0	$SB1 = R_{(P1)}^{-K2}$
RS	I2,0,B1.	
JLT	I2,0,0,1	GE $B0, B1$, FIRST BRANCH
JIN	0,12,0,3	JP $B1 + GLn.$
LAB	0,3	$GLn.$ BSS 0
ENDR		

5.3.5 Unconditional jumps are then placed in a temporary RLIST buffer for all inside transfer labels.

If the next to the last non-repeated label is the same as the next statement, no jump is compiled.

6.0 Formats of internal cells:

6.1 ASA - Set non-zero when non ASA usage detected.

6.2 VARIABLE -Contain first entry in expression of computed GO TO.

6.3 ANLBRNCH - Contain address of next to last branch in computed GO TO.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 217
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- | | | |
|-----|--------------------|--|
| 6.4 | INDEX | Used to save 3 index registers. |
| 6.5 | FBRANCH
LBRANCH | First and last branch of computed GO TO list. |
| 6.6 | EOBL
SOBL | End and start of branch list after it is moved. |
| 6.7 | SAVEA7 | Used to save A7. |
| 6.8 | NLBRANCH | Next to last branch in computed GO TO. |
| 7.0 | Modification | All cells referenced as well as diagnostic numbers are set with EQU's. |

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 22.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6000

ASFPRO

1.0 General Information

1.1 ASFPRO consists of two independent subroutines ASFDEF and ASFREF. ASFDEF processes all ASF definitions by saving the text in a table; ASFREF processes all references to ASF's by expanding the ELIST by inserting the ASF definition.

2.0 Usages

2.1 ASFDEF

2.1.1 Function of ASFDEF. Calling ASFDEF will process the entire E-list. The modified E-LIST is moved to ASFTAB and the CONLIST and DOL pointers ^(ASF LAST) are incremented to compensate for the growth in ASFTAB.

2.1.2 ASFDEF is called with a return jump. It is only necessary that SELIST (RA + 32B) point to the E-LIST entry containing the ASF name. Return is to PH2RETN either directly or through ERPRO.

2.1.3 Processing flow description.

1. The parameter list is checked for proper format.
2. All references on the right of the equal sign to parameters are replaced with parameter ordinal indicators. Any entires in CONSTOR are moved to the ACF TAB area.
3. The ELIST after the equal sign is moved to the ASF TAB area.
4. The ASF text is linked to any previous ASF texts.

2.2 ASF REF

2.2.1 Function of ASFREF. ASF ref is called whenever a reference to an ASF is encountered by ARITH. ASFREF replaces the reference to the ASF with the ASF text resulting in an expanded E-LIST statement.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 22.2
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

2.2.2 ASFREF is called by a return jump with SELIST pointing to the ASF name within **E-LIST**.

2.2.3 Processing flow description.

1. The ASF name is checked to see if it has been properly defined.
2. The remainder of ELIST is moved next to DOL.
3. The parameters to the ASF are bracketed and checked for correspondence in number with the definition.
4. The text is expanded and appended to ELIST.
5. The part of the statement following the ASF parameter list is ELIST.

3.0 Diagnostics produced.

1. Dummy parameter is an arithmetic statement function definition occurred twice.
2. Arithmetic statement function has caused a table overflow while being processed.
3. Arithmetic statement function has more dummy parameters than allowed.
4. Arithmetic statement function has an improperly formed parameter list of no = following the list.
5. A reference to this arithmetic statement function was not followed by an open paranthesis.
6. Insufficient memory was available for the evaluation of this arithmetic statement function reference, possibly a recursively defined ASF.
7. A reference to an improperly specified arithmetic function has

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 22.3
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

been encountered.

8. A reference to this arithmetic statement function has unbalanced parenthesis within the parameter list.
 9. The number of parameters used in referencing this arithmetic function does not correspond to the number in its definition.
- 3.1 Fatal to compilation diagnostics.
 - 3.2 Fatal to execution diagnostics.
 - 3.3 Information diagnostics.
 - 3.4 Non-ASA diagnostics.
- 4.0 Environment. The following is a list of the required characteristics of variables external to ASFPRO.
 1. RA is zero.
 2. $ASF1 = RA + 25B$ is the first available cell for ASFPRO tables.
 3. $DOL = RA + 30B$ is $ASFLAST + 100$.
 4. $DOLAST = RA + 31B$ is $ASFLAST + 100$.
 5. SELIST = RA + 32B contains machine address of ASF name.
 6. $ASFLAST = RA + 26B$ is next available cell for ASF tables, initially equal to ASF1. *(CAN 1) = beg. of conflict.*
 7. SYMEND = RA + 13B is end of symbol table.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 23.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

PH2CTL

1.0 General Information

1.1 PH2CTL Phase 2 control serves as the interface routine between SCANNER and the appropriate processors for all non-specification type statements.

2.0 Usages

2.1 Entry point names

2.1.1 PH2CTL This is the initial entry point to Phase 2. All Phase 2 core initialization is done at this time (see paragraph 4.0).

2.1.2 PH2RETN. This is the entry point which can be used by any processor when an error situation arises and it is desired to resume normal processing. This routine checks to see if a statement label exists and if so, jumps to DOLAB to do the post-processing label handling. An End-of-Statement is written to the R-list file and processing then resumes for the next statement.

2.1.3 STR (Store R-list) is a routine which can be used to store one word of R-list in the R-list buffer. The R-list entry should be in X6. The routine saves all B registers and X0 through X3.

2.2 Processing flow description

A return jump to SCANNER is made for each statement (one exception: logical IFs). A check is made to determine if the statement is executable, if not, a diagnostic is issued. If the statement is labeled a call to DOLABCN is made, unless the statement is NAMELIST, DATA or FORMAT. If the statement is unlabeled, it is determined if a path to the statement exists, if not an informative diagnostic is issued. A cell in common called STSORD (statement temporaries) is compared against STMAX (maximum number of cells needed for statement temporaries) and STMAX is updated if STSORD > STMAX. NRLN (next R-list name, location 64) is compared to a maximum value if NRLN exceeds the maximum value, NRLN is reset to two. A jump indexed by the statement type is made to VTABL (statement vector table) and control is given to the appropriate processor.

Upon return from the processor, DOLAB is called if a label exists. An end-of-statement code is written to the R-list file and process begins again with a return jump to SCANNER.

3.0 Diagnostic produced.

Number 109, Fatal to execution

DECLARATIVE STATEMENT OUT OF SEQUENCE

Number 178, Informative

DOCUMENT CLASS IMS. PAGE NO. 23.2
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

THERE IS NO PATH TO THIS STATEMENT

4.0 Formats

4.1 Memory pointers and flags

4.1.1 DIM1 is set up by DPCLOSE, fwa and length of auxiliary table.

ASF1 last word address +1 of the auxiliary table.

ASFLAST last word address of the ASF table, initially set to ASF1.

D01 first word address of D0 table, initially ASFLAST+100.

DOLAST last word address of D0 table, initially set to D01.

GOTOCEL flag which indicates the last statement processed was a GO TO,
initially zero.

CONLAST contains last word address used for CONLIST, the program constants,
initially set to CON1.

FSTEX first executable flag, initially zero, set to non zero when the
first executable statement is processed.

NRLN next R-list name, initial value is 2.

STMAX number of cells needed for statement temporaries at execution time,
initial value is 1.

STSORD number of statement temporaries needed during the processing of
any one statement.

NGLN contains the value of the next generated label.

DOFLG Set non-zero when a D0 loop is in effect.

LSFLAG last statement flag, value is 1 if last statement was a RETURN,
GO TO or arithmetic IF, and is used by the END processor.

5.0 Remarks

The vector table is built using a macro which sets several flags depending
on what type of statement is being processed, the name of the macro is JMP.

JMP	MACRO	PROG, LSFLAG, FSTX
	IFEQ	LSFLAG, 0, 1
	SA6	B6
	IFEQ	LSFLAG, 1, 1

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 23.3
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

SA7	B6
IFC	EQ, **FSTX** , 1
SA7	A0
RJ	PROG
EQ	PH2RETN
ENDM	

DOCUMENT CLASS TMS PAGE NO. 24.1
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

DECPRO

Declarative Statement Processors

1.0 General Information

The declarative processing program has entry points entered to process: PROGRAM, SUBROUTINE, FUNCTION, BLOCK DATA, COMMON, DIMENSION, EQUIVALENCE, EXTERNAL and Type statements, and resides in Overlay 1.00.

2.0 Usages

2.1 Entry point: DPR0G.

2.1.1 DPR0G syntax checks PROGRAM statements and produces COMPASS code to set up buffer areas for file names mentioned in the statement.

2.1.2 DPR0G is entered by an unconditional jump to DPR0G and control is returned to PR0GRTN in PH1CTL.

2.1.3 The program name is saved in the RA area and entered into the symbol table. The file names are entered in the symbol table. IDENT and ENTRY line images are sent to the COMPASS file followed by the code to set up the file name buffer areas.

2.2 Entry point: DPSUB.

2.2.1 DPSUB syntax checks SUBROUTINE statements and produces COMPASS code for formal parameter initialization and address substitution code.

2.2.2 DPSUB is entered by an unconditional jump to DPSUB and returns control to PH1SCAN in PH1CTL.

2.2.3 The number of formal parameters is saved in the RA area. The subroutine name and formal parameter names are entered into the symbol table. IDENT and ENTRY line images are sent to the COMPASS file followed by formal parameter initialization code if necessary.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.2
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

2.3 Entry point: DPFUN.

2.3.1 DPFUN syntax checks FUNCTION statements and produces COMPASS code for formal parameter initialization.

2.3.2 DPFUN is entered by an unconditional jump to DPFUN and returns control to PH1SCAN in PH1CTL.

2.3.3 The number of formal parameters is saved in the RA area. The function name and parameter names are entered into the symbol table. The COMPASS code produced is the same as for a subroutine statement, followed by storage allocation to hold the function result.

2.4 Entry point: DPBDA.

2.4.1 DPBDA syntax checks BLOCK DATA statements and produces the IDENT COMPASS line image.

2.4.2 DPBDA is entered by an unconditional jump to DPBDA and returns control to PH1SCAN in PH1CTL.

2.4.3 The block data name or an assumed name is entered into the symbol table. The IDENT 'Name' line image is sent to the COMPASS file.

2.5 Entry point: DPEXT

2.5.1 DPEXT syntax checks EXTERNAL statements and makes and/or modifies symbol table entries for the names that appear in the statement.

2.5.2 DPEXT is entered by an unconditional jump to DPEXT and returns control to PH1SCAN in PS1CTL.

2.5.3 Each name appearing in the statement is entered into the symbol table if not already there.

2.6 Entry point: DPDIM.

2.6.1 DPDIM syntax checks DIMENSION statements, makes symbol entries for the names that appeared if necessary and makes entries in the auxiliary table to hold the dimensional information.

2.6.2 DPDIM is entered by an unconditional jump to DPDIM and returns control to PH1SCAN in PS1CTL.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.3
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- 2.6.3 The dimensional information is held in a two word entry in the auxiliary table area. Each new entry is linked to its successor by table address. The index of the dimension list entry is placed in the symbol table entry for the name. The index is increased by two preparing for the next entry.
- 2.7 Entry point: DPCOM.
- 2.7.1 DPCOM syntax checks COMMON statements, makes symbol table entries for the names that appeared if necessary and makes entries in the auxiliary table area to hold the common block information.
- 2.7.2 DPCOM is entered by an unconditional jump to DPCOM and returns control to PH1\$SCAN in PS1CTL.
- 2.7.3 The first appearance of a block name generates a two word entry in the common list; one to hold the block name and linkage to the table area for the next block name, and one to hold equivalencing information if necessary and linkage to the table area for the next appearance of the same block name. Each block member name generates a one word entry to hold linkage to the next block member and member name identification.
- 2.8 Entry point: DPTYP.
- 2.8.1 DPTYP syntax checks Type-statements and makes and/or modifies symbol table entries for the names that appear.
- 2.8.2 DPTYP is entered by an unconditional jump to DPTYP and returns control to PH1\$SCAN in PS1CTL.
- 2.8.3 Symbol table entries are made for the names if not already made with the explicit type set. If dimensional information appears this will generate the auxiliary table entries as described in 2.6.
- 2.9 Entry point: DPEQU.
- 2.9.1 DPEQU syntax checks EQUIVALENCE statements and generates equivalence list entries in the auxiliary table for each name that appears.
- 2.9.2 DPEQU is entered by an unconditional jump to DPEQU and returns control to PH1\$SCAN in PS1CTL.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.4
PRODUCT NAME FORTTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- 2.9.3 A two word entry is made in the equivalence list for each name in an equivalence group. The first word holds linkage to the next equivalence group and name identification. The second holds the subscripts if any that appeared with the name.
- 2.10 Entry point: DPCL0SE.
- 2.10.1 DPCL0SE is called when the first executable statement is seen, to process the common, dimension and equivalence information, if any, that is being held in the auxiliary table area.
- 2.10.2 DPCL0SE is entered by an unconditional jump and returns control to PH2CTL.
- 2.10.3 EQUIV is called to process any equivalence list information. If COMMON statements appeared, COMPASS instructions are generated to produce appropriate storage allocation. Further storage allocation instructions are generated for local arrays if necessary and the dimension list is delinked and moved to consecutive storage locations.
- 2.11 Entry point: EQUIV.
- 2.11.1 EQUIV forms equivalence classes and makes necessary adjustments to the symbol table and the common list when necessary.
- 2.11.2 EQUIV is entered by an unconditional jump to EQUIV with EQ1{RA+55B} in X1. Control is returned to COM in DPCL0SE unless a fatal to compilation error occurs, in which case control is passed to FATALER.
- 2.11.3 The equivalence classes are established using "An Improved Equivalence Algorithm", Communications of the ACM, Volume 7/Number 5/May, 1964. The table required by the algorithm is formed in the auxiliary table area and uses two words per class name. Details in section 5.11. Each equivalence class will have a base member chosen as the name that will be assigned the lowest address in memory for the class. In the case where more than one member is assigned the lowest location, the name that was mentioned first in an EQUIVALENCE statement will be the base. Once a base is established, all other class members will be referred to as the base name plus the number of memory locations {bias} between the base origin and the member origin.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.5
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

One member in each class can be declared as being in common storage. In this case the origin of the common block will be the base of the class and all members of the block will be considered as class members.

3.0 Diagnostics Produced

3.1 Fatal to Compilation

3.1.1 AUXILIARY TABLE OVERFLOW. MORE MEMORY REQUIRED.

3.1.2 TABLE OVERFLOW DURING EQUIVALENCE PROCESSING. MORE MEMORY REQUIRED.

3.2 Fatal to Execution

3.2.1 PROGRAM CARD DELIMETER MISSING.

3.2.2 FILE NAME GREATER THAN 6 CHARACTERS.

3.2.3 FILE NAME PREVIOUSLY DEFINED.

3.2.4 EQUATED FILENAME NOT PREVIOUSLY DEFINED.

3.2.5 MORE THAN 50 FILES ON PROGRAM CARD OR 63 PARAMETERS ON SUBROUTINE/FUNCTION CARD.

3.2.6 RETURNS LIST ERROR.

3.2.7 DOUBLY DEFINED FORMAL PARAMETER.

3.2.8 NO LEGAL LIST TERMINATOR.

3.2.9 ILLEGAL SEPARATOR BETWEEN VARIABLES.

3.2.10 VARIABLE HAS MORE THAN 3 SUBSCRIPTS.

3.2.11 VARIABLE WITH ILLEGAL SUBSCRIPT.

3.2.12 VARIABLE DIMENSION NOT FORMAL PARAMETER.

3.2.13 COMMON VARIABLE WITH ADJUSTABLE SUBSCRIPT OR VARIABLE FORMAL PARAMETER.

3.2.14 COMMON BLOCK NAME NOT ENCLOSED IN SLASHES.

3.2.15 COMMON VARIABLE IS FORMAL PARAMETER OR PREVIOUSLY DECLARED IN COMMON.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.6
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- 3.2.16 ILLEGAL BLOCK NAME.
- 3.2.17 ILLEGAL SEPARATOR IN EXTERNAL STATEMENT.
- 3.2.18 ALL ELEMENTS IN AN ECS COMMON BLOCK MUST BE OF TYPE ECS.
- 3.2.19 A PREVIOUSLY MENTIONED ADJUSTABLE SUBSCRIPT IS NOT TYPE INTEGER.
- 3.2.20 ALL ECS VARIABLES MUST APPEAR IN A ECS COMMON BLOCK.
- 3.2.21 FORMAL PARAMETERS OR ECS VARIABLES CANNOT APPEAR IN EQUIVALENCE STATEMENTS.
- 3.2.22 SUBSCRIPTS NOT INTEGER CONSTANTS IN EQUIVALENCE STATEMENT. EQUIVALENCING ABANDONED.
- 3.2.23 ONLY ONE SYMBOLIC NAME IN EQUIVALENCE GROUP.
- 3.2.24 SYNTAX ERROR IN EQUIVALENCE STATEMENT. GROUP IGNORED.
- 3.2.25 VARIABLE SUBSCRIPTED BUT NO DIMENSIONS. GROUP IGNORED.
- 3.2.26 COMMON-EQUIVALENCE ERROR. EQUIVALENCING ABANDONED.
- 3.2.27 COMMON BLOCK ORIGIN EXTENDED. EXTENSION NOT ALLOWED.
- 3.2.28 XXXXXXX SUBSCRIPT VALUE OUT OF RANGE AS DETERMINED BY DIMENSIONS. EQUIVALENCING ABANDONED.
- 3.2.29 XXXXXXX SYMBOL WAS INVOLVED IN CONTRADICTIONARY EQUIVALENCING. EQUIVALENCING ABANDONED.
- 3.3 Informative and Non-USASI.
 - 3.3.1 DOUBLY DIMENSIONED VARIABLE.
 - 3.3.2 PREVIOUSLY TYPED VARIABLE.
 - 3.3.3 ARRAY NAME NOT SUBSCRIPTED. FIRST ELEMENT WILL BE USED.

4.0 Environment

The RA area is used to hold the beginning and end of the E-list, namely, SELIST at RA+32B and ELAST at RA+14B. The explicit type of a FUNCTION or Type-statement is expected at ATYPE or RA+51B. The beginning address of storage for the auxiliary table is initialized and maintained at AUXEND or RA+15B.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.7
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.0 Structure

5.1 DPROG

- 5.1.1 The program name is packed in location PROGRAM [RA+56B], left-adjusted to bit 53 and with a bias of zero, i.e., in VFD 12/2000B, 48/8H^oname^o. The name is entered into the symbol table at ordinal one with the type set to 11B and the ENT bit set to 1. As the file names are encountered, the special character, declared in MICRO form in the options file and named ^oC^o, is suffixed to the file name and entered into the symbol table, the U bit set to 1, and the following macro is sent to the COMPS file:

```
FILE MACRO LN,NAME  
  ENTRY NAME#C# BSSZ 1  
  IN# SET NAME#C#  
  LG# SET LN+1  
  VFD 16/1,26/12,18/IN#+1?  
  VFD 60/IN#+1?,60/IN#+1?  
  VFD 60/IN#+1?+LG#  
  BSSZ 12  
  BSS LG#  
  ENDM
```

Then the following image is sent to the COMPS file:

```
IDENT oProgram Nameo
```

The, for each new equivalenced file encountered, the following macro call is formed and saved in a temporary list for the COMPS file:

```
FILE Length,Name
```

After all the macro references,

```
FILES. BSS 0B
```

is sent to the COMPS file, and for each file

```
VFD 42/0L ofile nameo, 18/ofile nameo#C#
```

is also sent to the COMPS file.

The FILES. information is terminated by a DATA 0B.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.8
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

For each of the equivalenced files, for example,

TAPE1=INPUT

the following lines are sent to the COMPS file:

```
VFD 42/0L TAPE1,18/INPUT#C#  
TAPE1#C# EQU INPUT#C#  
ENTRY TAPE1#C#
```

After all file code is issued:

```
TRACE. VFD 60/7L Program Name  
USE CODE.  
ENTRY Program Name  
SA1 FILES.  
RJ Q8NTRY  
USE DATA.
```

is issued, and Q8NTRY is placed in the symbol table and marked as an external. DPROG expects SELIST {RA+32B} and ELAST {RA+14B}, the pointers to the start and end of the E-list to be set.

Each file name is also defined in the symbol table, i.e. the relocation base {RB} and reference address {RA} relative to zero of the relocation base. Bits 12-18 of symbol table word 2 hold 0 the RB for START., and bits 19-36 hold the RA.

For Version 2 the START. and CODE. lengths allocated in DPROG are added to the STARTA and CODEA totals.

5.2 DPSUB

- 5.2.1 The number of formal parameters is packed with a bias of 1 into RA+56B, i.e., VFD 12/2001B,48/n. The subroutine name is entered in the symbol table at ordinal one with the type set to 11B and the U, FP, ASF, and ENT bits set to 1. The formal parameter names are entered in the symbol table, implicitly typed. The FP bit is set to one and the order of appearance, starting at zero, placed in the RB bits of the word B entry. When a RETURNS list appears, the RETURNS parameter names are entered in the symbol table with a type of 7, and the U, FP, ASF, and ENT bits are set to 1. The order of appearance of RETURNS parameters, starting at 0, is placed in bits 34 to 39 of the word B entry.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.9
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

If no parameters appeared, the following line images are sent to the COMPASS file:

```

    IDENT ▽subroutine name▽
    USE START.
  TRACE. VFD 60/7L ▽subroutine name▽
  ENTRY. BSS 0B
    ENTRY ▽subroutine name▽
  ▽subroutine name▽ BSS 1B
    USE CODE.
    USE DATA.
  
```

Formal parameters require the FORPAR and SUB macros to be issued as well as code to save AD, as follows:

```

    IDENT ▽subroutine name▽
    USE START.
  TRACE. VFD 42/7L ▽subroutine name▽, 18/▽subroutine name▽
  TEMPA0. BSS 1
  FORPAR MACRO X
    USE X
  X      BSS 0B
    ENDM
  SUB    MACRO FP,CON
  .POS  SET 59-#
  .ORG  SET #-#/59
    USE FP
    VFD 3/2,9/.POS,30/CON,18/.ORG
    USE #
  ENTRY. BSS 0B
    ENTRY ▽subroutine name▽
  ▽subroutine name▽ BSS 1B
    SX6 AD
    SA0 A1
    SA6 TEMPA0.
    USE VARDIM.
    USE ENTRY.
    USE CODE.
    USE DATA.
    USE DATA..
    USE HOL.
    FORPAR ▽formal parameter▽
    :
    :
    :
  
```

DPSUB expects SELIST {RA+32B} and ELAST {RA+14B} to be set to the start and end of E-list respectively.

For Version 2 the START. and DATA. lengths allocated in DPSUB are added to STARTA and DATAA.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.10
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.3 DPFUN

- 5.3.1 The number of formal parameters is packed with a bias of 2 in location RA+56B, i.e. VFD 12/2002B,48/°n°. The function name is entered in the symbol table at ordinal one with type set to 11B and the F, U, FP, ASF and ENT bits set to 1. The function is implicitly typed unless the explicit type appears in ATYPE {RA+51B}. The order of appearance of the parameters, starting at 0, is placed in the RB bits. Identical code is generated as for a subroutine with parameters {see 5.2}.

In addition, the line image:

```
USE DATA.  
VALUE. BSS 1B
```

for single word functions or:

```
VALUE. BSS 2B
```

for two word functions is issued. DPFUN expects SELIST {RA+32B} and ELAST {RA+14B} to point to the E-list for the statement. For Version 2 the START. length allocated in DPFUN is added to START.

5.4 DPBDA.

- 5.4.1 When a name appears on the block data card {otherwise BLKDAT. is used} it is entered in the symbol table at ordinal one with type set to 11B and the ASF and ENT bits set to 0. The following line image is sent to the COMPASS file:

```
IDENT °block data name°
```

DPBDA expects SELIST {RA+32B} and ELAST {RA+14B} to hold the beginning and ending addresses of the E-list for the statement.

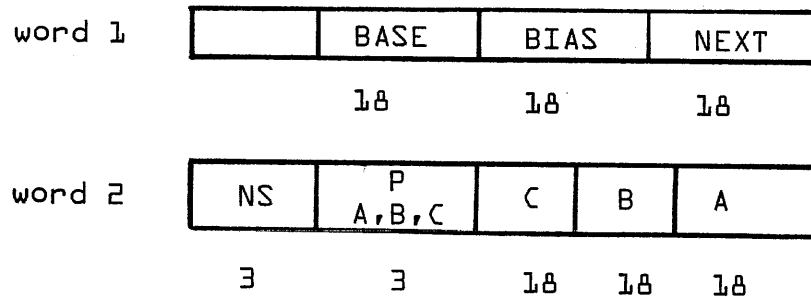
5.5 DPEXT.

- 5.5.1 DPEXT expects the E-list pointers SELIST {RA+32B} and ELAST {RA+14B} to be set. Each name appearing is entered into the symbol table if not already there with the EXT bit set to 1.

DOCUMENT CLASS IMS PAGE NO. 24.11
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PCDB MACHINE SERIES 64/65/6600

5.6 DPDIM

5.6.1 DIM1 {RA+17B} holds the auxiliary table address of the first {bits 30-47} and last {bits 0-17} dimension list entries or will be zero if none. The dimensional information is held in two words:



where:

NEXT = auxiliary table address of next dimension list entry, or zero for the last one. BASE, BIAS are defined in Section 2.11, the EQUIVALENCE processor.

A = the first constant dimension or the symbol table ordinal of the formal parameter that is the adjustable dimension.

B = same as A for the second dimension.

C = the total number of memory locations needed to hold the constant array or possibly a constant third dimension if either A or B were adjustable or the symbol table ordinal of the third dimension if adjustable. PA, B, C set to 0 or 1 corresponding to the A, B, C fields are constant or adjustable respectively.

NS = the number of subscripts, 1, 2, or 3. A symbol table entry is made for the name if not already there.

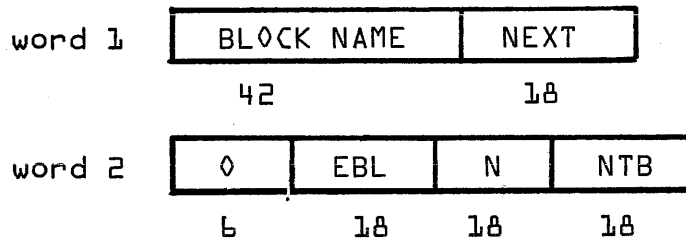
The index into the dimension list starting at zero is entered into the DIMP bits of the symbol table for the dimensioned name. The D bit is set. The index is increased by 2 and placed in DIMORD which is an entry point in DPDIM. DPDIM expects SELIST {RA+32B} and ELAST {RA+14B} to be set upon entry.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.12
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.7 DPCOM.

5.7.1 COM1 {RA+32B} holds the auxiliary table address of the first {bits 30-47} and last {bits 0-17} common list entries or will be zero for no COMMON statements. The first appearance of a block name requires two words in the common list.



where:

BLOCK NAME = left adjusted blank filled display code of the block name.

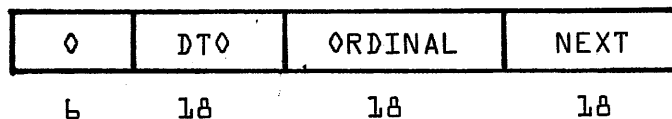
NEXT = auxiliary table address of next block name or zero for last.

0 = block name common list ordinal, starting with 1 with a maximum of 77 octal.

EBL = class range when the block is involved in equivalencing or zero if not.

N = the number of names that appeared with this mention of the block.

NTB = auxiliary table address of the same type entry for the next mention of the block. Each subsequent mention of the block name will require only the word 2 entry. Each block member requires one word and they follow the word 2 entry sequentially.



DT0 = distance from block origin to origin of this name, set only if equivalencing involved.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.13
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

ORDINAL = symbol table ordinal of block name.

NEXT = auxiliary table address of next common list entry.

A symbol table entry is made for the name if not already there. The C bit is set. For blank COMMON the BC bit is set. The block ordinal number is placed in the RB bits.

5.8 DPTYP.

5.8.1 A symbol table entry is made for the names that appear unless already made. The type is set to the explicit type found in ATYPE{RA+51B}. If the statement contains dimensional information, dimension list entries are made as described in 5.6. If the explicit type changes an array from one word to two word elements, the total size field {c} is doubled. SELIST and ELAST, the E-list pointers, are expected to be set upon entry.

5.9 DPEQU.

5.9.1 EQ1 {RA+55B} is maintained to hold two pointers, one the location of the first equivalence group information in the auxiliary table and another the last group. The first is held in bits 30-47, the last in bits 0-17. The next available location in the auxiliary table is maintained in AUXEND {RA+15B}. The first word of each group information holds:

bits 48-49, 200m where m is the number of names in the equivalence group. For all names but the first in the group, the first word for the name only holds the bits 0-17 information.

The second word for all group names holds:

bits 0-17, the first subscript if any.
bits 18-35, the second subscript if any.
bits 36-53, the third subscript if any.
bits 57-59, 0, 1, 2, or 3 the number of subscripts that appeared with the name.

The EQUIVALENCE statement E-list is expected to start at {SELIST} with the first left parenthesis and continue to {ELAST} with the end-of-statement. The syntax errors looked for are:

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24-14
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

1. group must be enclosed in parenthesis.
2. any misplaced information.
3. more than three subscripts.
4. subscripts not integer constants.

When a syntax error occurs the group is ignored.

SYMBOL is called and the name placed in the symbol table if not already there. The name is also implicitly typed if not already in the symbol table.

CONVERT is called to transform the E-list form of the constant subscripts to binary form.

5.10 DPCL0SE.

- 5.10.1 If EQ1 {RA+55B} is non-zero, EQUIV is called for equivalence list processing.

If COM1 {RA+20B} is non-zero, the following COMPASS line images are sent to the COMPASS file:

```
USE/▽block name▽/  
  ▽member name▽ BSS ▽member length▽
```

The ▽member name▽ line is repeated for each name in the block unless equivalencing is involved, in which case only the

```
  ▽origin member name▽ BSS ▽extended block length▽
```

line is sent to the COMPASS file.

For ECS common blocks a USE/▽block name▽/ is sent to the COMPASS file, followed by the ECS declared variable and/or array names prefixed by a decimal point and BSS storage in octal, i.e., . ▽name▽ BSS ▽name length B▽. After storage is allocated, a USE DATA is issued, followed by:

```
  ▽name▽ VFD 60/. ▽name▽
```

for each ECS declared name.

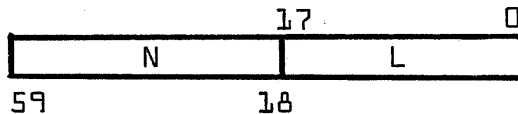
The dimension list is now moved to consecutive locations starting at PH2CTL+RBUFF. BSS storage is issued for arrays if no COMMON or EQUIVALENCING has already forced

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.15
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

storage to be allocated for the array. DIM1 is changed to hold in the length of the dimension list in bits 30-59 and PH2CTL+RBUFF in bits 0-17.

5.10.2 For Version 2 the name of and storage allocated for each common block is saved in ORGTAB in the following form:



where:

N = common block name left justified with blank fill.

L = common block length, in binary.

After the last common block entry in ORGTAB, a word of zeros is stored. The total length of DATA.. created through BSS is preserved in location DATAAA.

5.11 EQUIV.

5.11.1 The table required by the algorithm {called GFTABLE} is formed in the auxiliary table starting at the current location of AUXEND {RA+15B}. Two words are needed for each distinct name in a class. Two passes are required. One to define the classes and the second to determine the base and calculate the bias values.

During pass one the two words hold:

word one, 0-17, the address of the GFTABLE of the next class member or zero for last class member.

18-35, H, the distance needed below the root.

36-53, H', the distance needed above the root.

word two, 0-17, word B address of the symbol table entry for the class name.

18-35, R, the signed distance to the connective mode of the tree.

36-47, S, the GFTABLE ordinal of the connective mode entry.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.16
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

After the classes are established a pass is made using the GFTABLE to assign relative addresses to the class members. After this pass the changes to the two words are:

word two, 18-35, $a\{\bar{E}\}$, the length from zero needed to hold the name.

In addition, for all root entries, word one changes are:

word one, 18-35, the range of the class.

36-53, the GFTABLE address of the next class root entry or zero for the last class.

Each class is now searched for the class base which is the member with the largest value of $a\{\bar{E}\}$ or for the same value of $a\{\bar{E}\}$ the first mentioned member. The bias for all non-base members is now calculated $\{a\{\bar{E}\} \text{ member}\}$ and placed in word 2, 18-35.

When no common is involved, all that remains is to make the base and bias entries in the DIMLIST entries for all non-base members, and also to set the U and E bits in the symbol table for non-base members. The range is stored in bits 36-53 of word 2 of the base DIMLIST entry. If the base is a variable, the S bit is set in the symbol table base entry. When common is involved, the class members are searched to make sure no more than one member has been declared in common. The distance between the origin of the common block and the block members is calculated and stored in bits 36-53 of the COMLIST entry for the block member. A DIMLIST entry is made for the variables in the block. The total range of the class, $\text{max}\{\text{block range, distance}\{\text{common origin}\} + \text{local class range}\}$, is stored in bits 36-53 of the first word 2 of the COMLIST block entry. The base and bias entries are made for the block members, and the local equivalence class members have an additional bias due to common added to their bias and stored in DIMLIST. The DIMLIST areas effected are word 1 bits 18-35, the bias, and bits 36-53, the symbol table ordinal of the class base.

When a common block has already been biased, it is only necessary to calculate the class range and base and bias for the local class members. The process is repeated for all equivalence classes, then control is returned to COM in DPCLOSE.

DOCUMENT CLASS IMS PAGE NO. 24.17
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

If any diagnostic situations occur the class is skipped and the next class is processed.

Fields modified for class base:

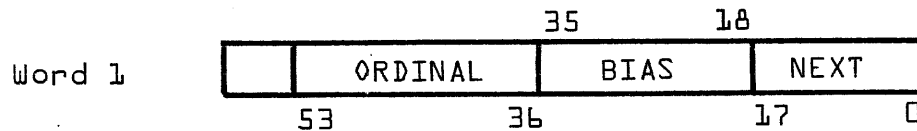
When no common storage is involved, the class range is placed in word 1 of the DIMLIST entry for the base. If the base is a simple variable, the S bit in word B of the symbol table is turned on. When common is involved the class range is held in word 2 of the block information in COMLIST.

Fields modified for non-base class members:

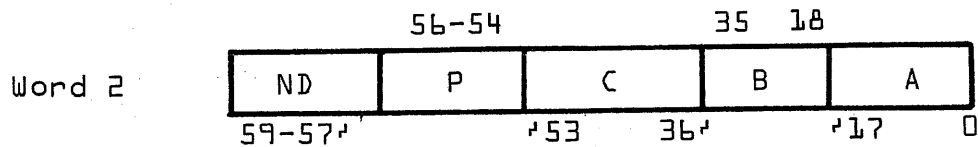
The symbol table ordinal of the class base and the member bias are held in word 1 of the DIMLIST entry for the class member. The E and U bits in word A of the symbol table are turned on.

6.0 Formats

DIMLIST. Used to hold dimensional information as well as base and bias information when the name is a non-base member of an equivalence class. A two word entry is used.



ORDINAL = ordinal of base symbol table entry
 BIAS = equivalence class member.
 NEXT = address of next DIMLIST entry, or zero for last DIMLIST entry.



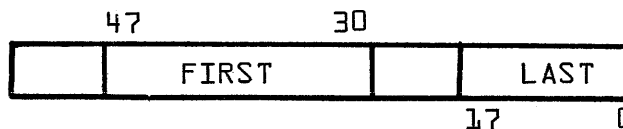
ND = 0, 1, 2, or 3, the number of dimensions.
 P = 0, 1, 2, 3, 4, 5, 6, or 7, corresponding to C, B, A. The bits are set when C, B, A are adjustable dimensions.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.18
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

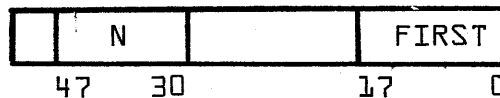
C, B, A = the symbol table ordinal of the dummy arguments that are the adjustable dimensions or: the binary form of the constant for constant dimensions for B, A, C the total number of storage units needed for the array.

DIML Is at RA+17B, and is zero for no dimensions or:



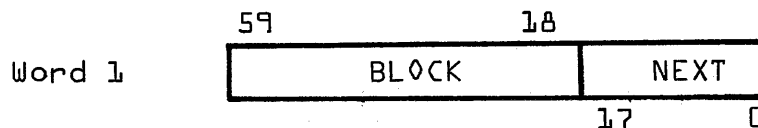
FIRST = address of first DIMLIST entry.
 LAST = address of last DIMLIST entry.

At the completion of Phase 1 the DIMLIST entries are moved to consecutive locations and DIML =



FIRST = address of DIMLIST.
 N = length of DIMLIST.

COMLIST. Used to hold COMMON block information as well as the equivalence class range when a block number is also an equivalence class member. The first mention of the block requires two words:

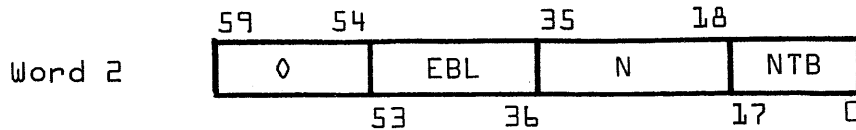


BLOCK = left adjusted, blank filled, display code of the block name.

NEXT = COMLIST address of word 1 of the next block name or zero for last block name.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.19
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600



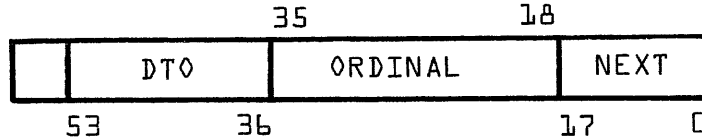
0 = the block name COMLIST ordinal, must be in the range 1 through 77.

EBL = The equivalence class range when one of the block members is also an equivalence class member.

N = the number of names that appeared in the COMMON statement with this mention of the block name.

NTB = address in COMLIST of the word 2 entry for the next mention of the same block name, or zero for last mention.

Each block member requires one word.

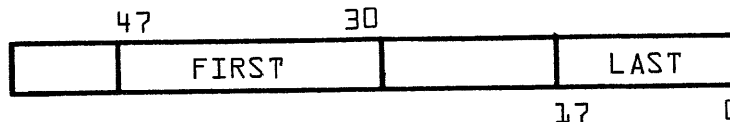


D Δ O = the distance to block ordinal, only set when one block member is also an equivalence class member.

ORDINAL = symbol table ordinal of the block member.

NEXT = COMLIST address of next block member entry or zero for last member entry.

COM1 is at RA+20B and is zero for no COMMON blocks on:



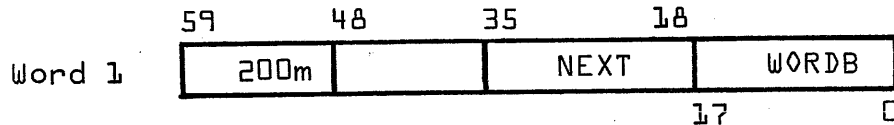
FIRST = address of first word 1 block entry.

LAST = address of last word 1 block entry.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.20
 PRODUCT NAME FORTTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

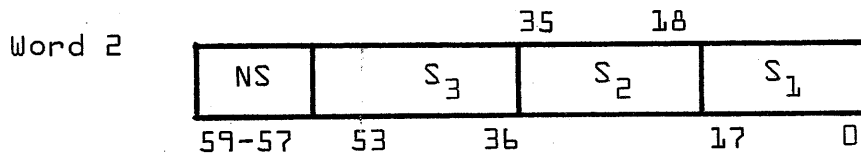
EQLIST. Used to hold equivalence group information for processing as part of the termination of Phase 1. Each name of each group requires two words.



200m = with m the number names in the group, this appears only with the first name of the group.

NEXT = EQLIST address of word 1 for the next group and will be zero for the last group.

WORDB = symbol table address of word B of the group name.



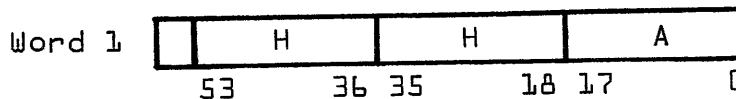
NS = 0, 1, 2, or 3, the number of subscripts that appeared with the group name.

S₁ = the first subscript if any.

S₂ = the second subscript if any.

S₃ = the third subscript if any.

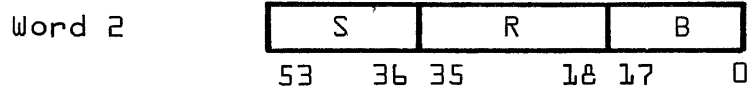
GFTABLE. During equivalence processing two words for each distinct class name are used to hold information necessary to the processing. During class information time the two words hold:



CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

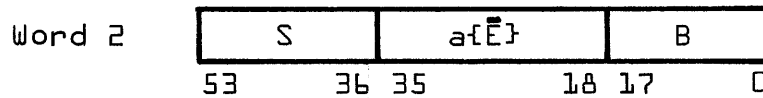
DOCUMENT CLASS IMS PAGE NO. 24.21
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- H = the distance needed below the class root.
- H = the distance needed above the class root.
- A = the GFTABLE address of word 1 of the next class member or zero for last class member.



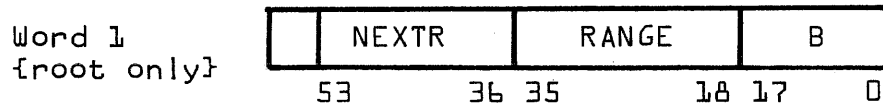
- S = the GFTABLE ordinal of the connective mode entry.
- R = the signed distance to the connective mode.
- B = the address of word B of the symbol table entry for the class name.

After the classes are established, relative addresses are assigned to class members, at which time word 2 holds:



a{E} = the length from zero needed to hold the name.

At this time all root entries word 1 hold:

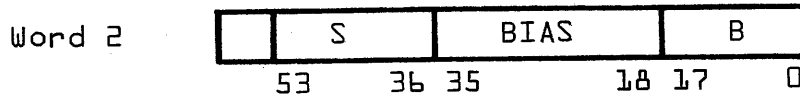


- NEXTR = GFTABLE address of word 1 entry for next class root or zero for last class.
- RANGE = the class range, the number of storage units needed to hold the class.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 24.22
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

The new information is used to find the class base and then the class member bias values which is held in word 2.



BIAS = the number of storage units added to the {1, 1, 1} element of the base to reference the {1, 1, 1} element of the non-base member.

8.0 Method

8.1 General

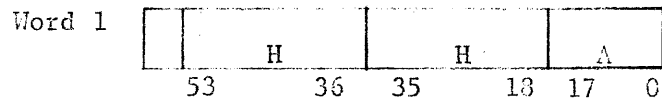
Basic list processing techniques are used to link the several lists. The head of each list {COM1, DIM1, EQ1} is held in the RA area of the compiler and holds pointers to the first and last entry of the list. Each entry of the list holds a pointer to the next member of the list and a 0 pointer signifies the last entry of the list. This process allows each list to grow subject only to the total working area available and not be bounded by any preassigned fixed size per table.

8.2 Equivalencing algorithm. The algorithm used is due to Bernard A. Galler and Michael J. Fisher, "An Improved Equivalence Algorithm", Communications of the ACM, Volume 7/Number 5/May, 1964, pages 301-303 and "In Defense of the Equivalence Algorithm", Communications of the ACM, Volume 7/Number 8/August, 1964, page 506.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 24.23
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

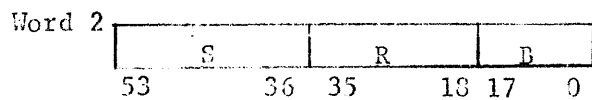
GFTABLE. During equivalence processing two words for each distinct class name are used to hold information necessary to the processing. During class formation time the two words hold:



H = the distance needed below the class root.

H = the distance needed above the class root.

A = the GFTABLE address of word 1 of the next class member or zero for last class member.

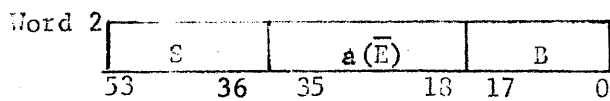


S = the GFTABLE ordinal of the connective mode entry.

R = the signed distance to the connective mode.

B = the address of word B of the symbol table entry for the class name.

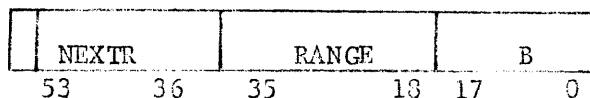
After the classes are established relative addresses are assigned to class members at which time word 2 holds:



$a(\bar{E})$ = the length from zero needed to hold the name.

At this time all root entries word 1 hold:

Word 1
(root only)



NEXTR = GFTABLE address of word 1 entry for next class root or zero for last class.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 24.24
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

RANGE = the class range, the number of storage units needed to hold the class.

The new information is used to find the class base and then the class member bias values which is held in word 2.

	S	BIAS	B
53	36	35	18 17 0

BIAS = the number of storage units added to the (1, 1, 1) element of the base to reference the (1, 1, 1) element of the non-base member.

3.0 METHOD

3.1 General

Basic list processing techniques are used to link the several lists. The head of each list (COM1, DIM1, EQ1) is held in the RA area of the compiler and holds pointers to the first and last entry of the list. Each entry of the list holds a pointer to the next member of the list and a 0 pointer signifies the last entry of the list. This process allows each list to grow subject only to the total working area available and not be bounded by any preassigned fixed size per table.

3.2 Equivalencing algorithm. The algorithm used is due to Bernard A. Galler and Michael J. Fisher, 'An Improved Equivalence Algorithm', Communications of the ACM, Volume 7/Number 5/ May, 1964; pages 301-303 and 'In Defense of the Equivalence Algorithm', Communications of the ACM, Volume 7/ Number 3/ August, 1964, page 506.

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 25.1
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

PHICTL

1.0 General

- 1.1 PHICTL is the controlling routine for all non-executable statements. All Phase I memory pointers are initialized in this routine.

2.0 Usages

2.1 Entry Point names

- 2.1.1 PHICTL. This is the initial entry point to Phase I from the 0.0 overlay. The memory pointers (see paragraph 4.0) are initialized and a call is made to SCANNER to obtain the first source statement.

- 2.1.2 PHISCAN. This entry point is used by all Phase I processors to return control.

- 2.1.3 PROGRN. This entry point is referenced by DPPROG to return to PHICT2 after processing the PROGRAM card.

- 2.2 Calling sequences. All calls to the processors are made with an EQ jump, except FORMAT which is entered via a return jump.

2.3 Processing Flow Description.

Control is entered through PHICTL at which time the necessary points (see paragraph 4.0) are initialized. A call to SCANNER is made to obtain the first source card. If this card is a legitimate header card processing continues normally. If the first card is an end-of-record, a call is made to load COMPASS, if neither a dummy header card is manufactured. The source interpretation of this card is:

PROGRAM START. (INPUT, OUTPUT)

DPROG is then called to process the card and normal processing resumes.

3.0 Diagnostics Produced

Error number 10 which is fatal to execution.

HEADER CARD NOT FIRST CARD

4.0 Environment

- 4.1 SYMEND location 13B contains the field length -1 (first word address of symbol table)

- 4.2 SYM1 location 12B, initially set to field length -1 (last word address of symbol table)

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. _____ 25.2 _____
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. _____ 4P616 _____ MACHINE SERIES _____ 64/65/6600 _____

- 4.3 AUXEND location 15B, initially set to PHICTL, contains the first word address (fwa) of the auxiliary table.
- 4.4 SUBREF location 16B, initially set to zero.
- 4.5 DIM1 location 17B, initially set to zero (pointer to fwa of the dimension table)
- 4.6 EQ1 location 55B, initially set to zero (pointer to the fwa of the equivalence table)
- 4.7 COM1 location 20B, initially set to zero (pointer to the fwa of the common table).
- 4.8 FX location 40B, initially set to zero (contains the number of errors encountered during a compilation).
- 4.9 FC location 41B, initially set to zero (set to non-zero if any fatal to compilation errors are encountered).
- 4.10 PROGRAM location 56B, initially set to zero.
- 5.0 Remarks

Two routines exist for generating I/O files within the compiler; one in PS1CTL and the other in PS2CTL. Each routine is tailored to the specific type of I/O in each pass of the compiler. In PHICTL, a return jump to the Pass 1 I/O routine is stored into the appropriate locations in FTN. These locations are PLUG1, and PLUG2 (see paragraph 2.1.12 in FTNS10 task description).

DOCUMENT CLASS IMS PAGE NO. 26.1
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

PASS 2 - COMPILING TECHNIQUE

The method used for code selection in the FORTRAN Extended compiler can best be explained with an example. Consider the following FORTRAN source statements:

```

IF {ATAG .LT. B*B} CALL COMET
[
PSI={B+ACT {N}} **2+RHO**SIGMA {N}
K=N+K
QTAB {N}=XTAB {I}/RHO+PSI
]
4 TAB2 {2,2*K}=PSI+ATAN {RHO}
    
```

Statements within the bracket constitute a flow block or sequence. Initially, these statements are analyzed and converted to a register free notation called R-list which would appear as:

$R_1 \leftarrow B$	$R_{11} = R_6 + R_{10}$	$R_{19} = R_{17} / R_{18}$
$R_2 \leftarrow N$	$R_{12} = N \{R_{11}\}$	$R_{20} \leftarrow PSI$
$R_3 \leftarrow ACT - 1, R_2$	$R_{12} \rightarrow PSI$	$R_{21} = R_{20} + R_7$
$R_4 = R_1 + R_3$	$R_{13} \leftarrow N$	$R_{22} = N \{R_{21}\}$
$R_5 = N \{R_4\}$	$R_{14} \leftarrow K$	$R_{23} \leftarrow N$
$R_6 = R_5 * R_5$	$R_{15} = R_{13} + R_{14}$	$R_{22} \rightarrow QTAB - 1, R_{23}$
$R_7 \leftarrow N$	$R_{15} \rightarrow K$	
$R_8 \leftarrow SIGMA - 1, R_7$	$R_{16} \leftarrow I$	
$R_9 \leftarrow RHO$	$R_{17} \leftarrow XTAB - 1, R_{16}$	
$R_{10} = R_8 * R_9$	$R_{18} \leftarrow RHO$	

{N{R_i}} indicates the normalization of the result of a floating add or subtract; left arrows are loads and right arrows are stores.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

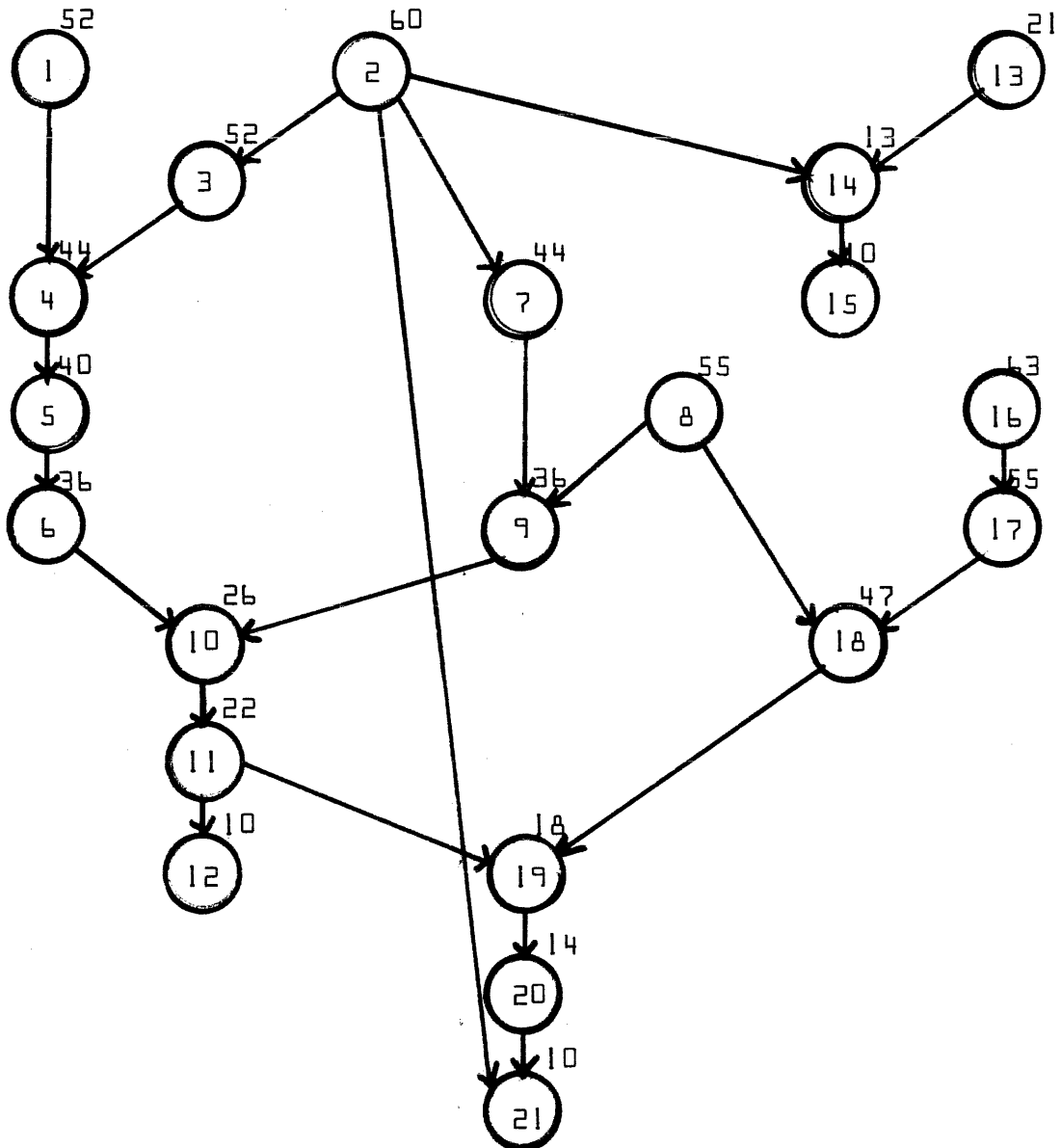
DOCUMENT CLASS IMS PAGE NO. 26.2
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

The generated R-list is then scanned and common suboperations are eliminated resulting in the squeezed R-list.

$R1 \leftarrow B$ $R10 = R8 \times R9$ $R17 \leftarrow XTAB-1, R16$
 $R2 \leftarrow N$ $R11 = R6 + R10$ $R19 = R17 / R9$
 $R3 \leftarrow ACT-1, R2$ $R12 = N\{R11\}$ $R21 = R12 + R19$
 $R4 = R1 + R3$ $R12 \rightarrow PSI$ $R22 = N\{R21\}$
 $R5 = N\{R4\}$ $R14 \leftarrow K$ $R22 \rightarrow QTAB-1, R2$
 $R6 = R5 \times R5$ $R15 = R2 + R14$
 $R8 \leftarrow SIGMA-1, R2$ $R15 \rightarrow K$
 $R9 \leftarrow RH0$ $R16 \leftarrow I$

DOCUMENT CLASS IMS PAGE NO. 26.3
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

From the squeezed list a PERT-like network, the following depend-
 ency tree, is formed showing the precedence of operations.



CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 26.4
PRODUCT NAME FORTTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

The numbers within the circles at the nodes are keyed to the R-list on page 2. The time in machine cycles required for each operation is known. From this information, the latest time at which each operation must begin in order to finish executing the network in the minimum amount of time is calculated. This is done assuming no conflicts of any kind and parallel instruction issue as well as execution. These times called priorities, are the numbers shown next to each circle; in a PERT sense they correspond to negative late start times with the network being completed at time zero. Code is generated beginning with the highest priority entry in the squeezed R-list noting which function unit is used and for how long. For all later instructions it is required that the preceding operations have been issued and there are no function unit or register conflicts; for this purpose, a picture of the status of all registers and function units must be maintained. Using this approach, the code shown on the next page produced resulting in the indicated overlap of operations.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 26.5
 PRODUCT NAME FORTTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

R	Instruction	Function Units											Load/Store							t
		B	B	F	S	M	M	D	L	I	I	X	X	X	X	X	X	X	X	
		L	R	A	H	I	2	V	A	1	2	1	2	3	4	5	6	7		
16	SA1 I																			
2	SA2 N																			
8	SA3 RH0																			
17	SA4 XTAB-1+X1																			
1	SA5 B																			
3	SA1 ACT-1+X2																			
18	FX0 X4/X3																			
4	FX4 X5+X1																			
7	SA1 SIGMA-1+X2																			
5	NX5 B7, X4																			
9	FX4 X1+X3																			
13	SA1 K																			
6	FX3 X5+X5																			
10	FX5 X3+X4																			
14	IX6 X2+X1																			
11	NX7 B7, X5																			
15	SA6 K																			
19	FX1 X7+X0																			
7	NOP																			
12	SA7 PSI																			
20	NX6 B7 X1																			
	NOP																			
21	SA6 QTAB-1+X2																			

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 27.1
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

PRE

1.0 General Information

PRE is the main controlling routine for PASS 2.

1.1 PRE Functions.

- a. Call the macro expanders (MACROE, PRODB, PRODE, PROIXFN).
- b. Expand all R-list into three-word form and define the sequence.
- c. Put out inactive labels, formal parameter names, variable dimension storage and the END line to COMPASS.
- d. Call the optimizers (COPY, SQUEEZE, PURGE, BUILDDT, OPTA).
- e. Accumulate APLIST entries and call APLISTP.
- f. Issue BSS storage for statement, D0, and optimizing temporary storage.

2.0 The only entry point for PRE is PRE.

2.1 A jump to PRE causes COMPASS code generation for all R-list on the file.

2.1.1 Calling Sequence:

JP	PRE
a. APL APLAST	address of end of symbol table (SYMEND)
b. VAR1 VARLAST	SYMEND - 100B
c. MACORG	EQU to lowest significant macro number in MACROX
d. FFLAG	=0 for standard mode
e. PROGRAM	contains transfer address, if main program

2.1.2 Processing Flow

After reading in a fixed number of words at a time from the R-list file, each positive entry with a positive opcode is expanded into 3-word form. The descriptor ST field of each is examined

DOCUMENT CLASS IMS PAGE NO. 27.2
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

to detect one of the following cases:

- a. If ST=7, a macro ordinal switch determines which of the macro expanders to call. A PRODE call terminates the sequence after the 60 end jump.
- b. If ST=0, the entry is considered "normal R-list" and is added to the sequence at PS2CTL by a call to ADDTOSQ.
- c. If ST=1-6, the entry is directed to a jump table, where the following are detected:
 1. Sequence terminators are: DO end jump, entry statement, active label, and unconditional jumps.

They cause generation of an end of sequence entry (100). Normally, PROSEQ is called for code generation. However, in a well-behaved DO, sequences are allowed to accumulate before this call.
 2. APLIST entries are added to the list by ADDTOAP.
 3. The end of R-list entry triggers PASS 2 closing (CLOSE2). If VARDIM is present, it is coded at this time.

3.0 Diagnostics Produced

3.1 Fatal to compilation

INSUFFICIENT MEMORY.
THE STATEMENT AT (or AROUND) THIS LINE IS TOO COMPLEX FOR THIS COMPILER. PLEASE SIMPLIFY IT.

3.2 Fatal to execution: none.

3.3 Informative

MORE MEMORY WOULD HAVE RESULTED IN BETTER OPTIMIZATION.

4.0 Environment

MACWRDS - local to READL. Set by macro expanders to the number of words placed in MACBUF.

FWAWORK - local to PRE. Must contain current lowest unused work storage address.

LWAWORK - EQU VARLAST. Must contain current highest unused work storage address.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 27.3
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 _____ MACHINE SERIES 64/65/6600

WELLBE - Set by DO processor to 1 if a well-behaved DO loop is in process; else zero.

5.0 Structure: PRE Subroutines.

5.1 READRL

5.1.1 READRL reads in one word at a time from MACBUF, or, if empty, from NORL.

- a. A negative opcode indicates a macro reference. The entire reference is read in and placed at MACREF/RLIST.
- b. A positive opcode indicates a single entry. An additional read is made to pick up the second word of all type III entries. The descriptor is obtained from the local descriptor table. The entry is passed back to PRE in the following cells:

TWA1 = R-list descriptor. (This cell is set to -0 when a macro reference is sensed).

TWA2 = First word of the entry.

TWA3 = Second word if type III; else +0.

5.2 ADDTOSQ

5.2.1 ADDTOSQ adds one R-list entry at a time to the accumulated sequence: first word, second word, descriptor. If PRE sets the sequence-terminating flag, an end of sequence and an end of statement are generated after the word which was added.

5.3 PROSEQ

5.3.1 PROSEQ sets up the calls to COPY, SQUEEZE, PURGE and OPTA, indicating exactly which portion of R-list to code. Normally, the entire sequence is taken as a unit. If a sequence consists of only one entry, PROSEQ calls POST directly, bypassing the optimizers. When coding the VARDIM sequence, a call to SQZVARD is inserted after SQUEEZE. This allows elimination of redundant stores.

COPY, BUILDDT or OPTA may return with X6 set negative, indicating failure to perform the usual functions. PROSEQ then cuts the sequence in half and calls all routines as usual. If failure repeats, the cutting down continues down to failure on the smallest divisible portion, one statement (if VARDIM, the code between two stores). This portion is attempted once again, skipping the squeezing process. If this proves unsuccessful,

DOCUMENT CLASS IMS PAGE NO. 27.4
 PRODUCT NAME FORTTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

processing terminates with a fatal diagnostic indicating either memory limitations or statement complexity as the cause.

5.4 ADDTOAP

ADDTOAP adds one entry at a time to the list at AP1, omitting the descriptor and setting HIGHORD equal to the highest ordinal so far encountered. If the 100B locations allocated this area are insufficient, VARDIM is pushed back 100B locations.

5.5 VARCLOS

- 5.5.1 PRE calls VARCLOS if there is VARDIM code present when the end of R-list entry is detected. VARCLOS flips the VARDIM R-list and sets the NOR1 buffer pointers to this area. READRL and ADDTOSQ are called alternately to get this R-list into the usual 3-word format at the usual sequence area. PROSEQ is then called for code generation.

5.6 SQZVARD

- 5.6.1 SQVARD scans the descriptor list at FWADNEW to locate "live" stores (KILL bit = 0), and compares RI fields to find redundancies. The special symbol]VD is appended to the H field of all stores.

```
]VD "ordinal"      BSS      1B
```

is put out to COMPASS for one store of a particular RI field value.

```
]VD "ordinal"      BSS      0B
```

is put out for all other stores of that RI value. Setting the descriptor KILL bit for these stores assures that they will be purged from the R-list.

Each JVD label is defined relative to the CODE. relocation base and this definition is maintained in working storage.

5.7 NFPUNT

- 5.7.1 NFPUNT sets PUNTF LG=1, which will cause CLOSE2 to put out the non-fatal memory diagnostic. (3.1)
- 5.7.2 PUNT sends out the fatal memory diagnostic (3.1) and the END line, and exits to LDCOM.
- 5.7.3 OPTPUNT send out the fatal diagnostic resulting from statement complexity (3.2). The R-list is scanned for the nearest end of statement. If it contains no card number, the scan continues

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 27.5
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

for a substitute. If no number is found, the cell CARDCT contains the last one detected by PRE. The octal card number is converted to display code and incorporated into the message.

5.8 CLOSE2

5.8.1 CLOSE2 takes care of PASS 2 cleanup. It calls APLISTP to process the accumulated APLIST and sends out the informative memory diagnostic, if applicable. The substitution list for each formal parameter name is terminated by a DATA 0B.

The [AP and]VD definition tables are moved to just below the GL table. The ST., DO., and OT. BSS storage is issued for statement, DO, and optimizing temporary storage.

The program or subprogram terminating END line is issued and if the "R" option is selected CLOSE2 exits to LDPH1. Otherwise the FTNX assembler is called. Upon return from the assembler CLOSE2 exits to LDPH1.

6.0 Formats

6.1 The R-list Descriptor

59	58	57	56-53	52-49	48-47	46-42	41	40	39-30		
LD	SR	JP	F1	F2	TY-1	FT	K	IG	USES		
29	28	27	26		8	7	6	5-3	2	1	0
JD	RS	SS	SC	DO	FE	SZ	ST	KL	SQ	CM	

LD - set on long and short loads.

SR - set on long and short stores, including register store.

JP - set on all jumps

F1 - function unit used.

F2 - second possible function unit, = F1 if only one possible.

TY-1- type (I-IV).

FT - 6600 function time, = 10B for loads, 12B for stores.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 27.6
 PRODUCT NAME FORTTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

	<u>Unit</u>	<u>Indicator</u>
	Branch	1
	Broolean	2
	Shift	3
	Add (Integer)	4
	Add (Floating)	5
	Multiply #1	6
	Multiply #2	7
	Divide	10
	Increment #1	11
	Increment #2	12
K	set if hardward instruction includes no k field.	
IG	used by DOPRE.	
USES	set by BUILDDT, used by OPT.	
JK	set if hardware instruction includes jk field.	
RS	subsequent register store bit. Used in OPT.	
DO	this area is used by DOPRE.	
FE	set for jumps and stores	
SZ	set for 30 bit instruction.	
ST	DOST = 1, APLIST = 2, unconditional jump = 3, label = 4, end of R-list = 5, entry = 6.	
KL	initially = 0, set by SQUEEZE and SQZVARD when instruction killed.	
SQ	set for squeezable instructions.	
CM	set if operands commutative.	

6.2 Flag words

6.2.1 In PRE (for PROSEQ)

VARFLAG = 0, set ot 1 at VARDIM time.

6.2.2 In READRL

MACWRDS = 0, set to number or words placed in MACBUF.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 27.7
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

6.2.3 In ADDTOSQ

MACREF (EQU RLIST) LWA+1 of parameter words.

6.2.4 In ADDTOAP (for APLISTP)

APLAST - LWA, APLIST+1.

6.2.5 In PROSEQ

NORLIST - number of entries to process.

FWASEQ - FWA of R-list to process external to PROSEQ.

LASTR - first word of last single entry sequence encountered.

LASLBL - if LASTR = label, LASLBL = LASTR.

6.2.6 In SQZVARD (for PROSEQ)

LENGTH - number of cells in redundant VARDIM store code list
for COMPASS.

STORBUF - FWA of redundant store list.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 28.1
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

APLISTE

- 1.0 APLISTE - Aplist Expander is a part of Pass 2.
- 1.1 Converts the APLIST to card images and places them into the COMPASS file.
- 1.2 Outputs a SUB macro reference for any formal parameters that were not referenced in the subprogram and maintains the length of the relocation base associated with each formal parameter.
- 2.0 APLISTP
 - 2.0.1 APLISTP processes the accumulated APLIST and outputs it to the COMPASS file.
 - 2.0.2 APLISTP is entered by a return jump from CLOSE2 and exits through its entry point. If there is sufficient room to process the APLIST, at least as much working storage as there is APLIST, it exits to PUNT.
 - 2.0.3 The specified APLIST is rebuilt and grouped so that each list is contiguous in memory and in the proper order in which they should appear. These grouped lists are then examined and any that can be eliminated and combined into another are. The list is then converted to COMPASS line images and placed in the COMPASS file.
- 3.0 No diagnostics are produced.
- 4.0 The following cells are referenced and are expected to contain the indicated values.
 - 4.1 APl. The address of the first entry in the APLIST.
 - 4.2 APLAST. The address of the last entry in the APLIST.
 - 4.3 HIGHORD. The APLIST ordinal of the largest APLIST.
 - 4.4 FWAWORK. First word address of the working storage.
 - 4.5 SYM1. Start of the symbol table.
 - 4.6 PGM. The program/subprogram indicator.
 - 4.7 SUBREF. Flag that is non-zero if the ADDSUB code has been put out.
 - 4.8 S.NEC. Flag set if it is necessary to issue at least one word of ST. storage.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 28.2
PRODUCT NAME FORTTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- 4.9 AOSUBO All referenced for the ADDSUB coding.
CMPSPR
CMPSPS
ADDSUBN
CLSPOST
- 4.10 WRWDS2 is called to transfer the converted APLIST to the compass file.
- 5.0 The APLIST processing is divided into three phases. The format of the table built in the processing is under 6.0.
- 5.1 First, the jumbled APLIST is grouped so members of each group appear in contiguous memory cells in the proper order. This is done in the following manner.
- 5.1.1 The present APLIST number that is being grouped is set to one.
- 5.1.2 If the present APLIST number is greater than the highest APLIST number (HIGHORD) the grouping of the APLIST is complete.
- 5.1.3 Otherwise, the entire APLIST as it appears is searched and each entry that has the same number as the present APLIST number is combined into one word and placed in the grouped APLIST. The grouped lists start at FWAWORK and grow toward APLAST.
- 5.1.4 After all entries in a group have been extracted from the jumbled APLIST, an entry is made in the GAPL giving the FWA and LWA of this group. This table starts at AP1 and grows toward APLAST.
- 5.1.5 The present APLIST number is bumped by one and processing continues at 5.1.2.
- 5.2 The grouped APLISTS are then examined to see if it is possible to eliminate any of them. This is done by comparing two lists (a primary and a secondary) at a time. If it is found that we hit the end of either list before finding a non-equal entry, the group that we hit the end of can be eliminated. In this case the following is done:
- 5.2.1 The GAPL entry for the group which is to be eliminated is set to minus 1.
- 5.2.2 In the last examined entry of the non-eliminated group a GN (Group Number) is placed in the upper 12 bits. If it has a GN already, it is used rather than generating a new one.

CONTROL DATA CORPORATION . COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 28.3
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- 5.2.3 The APLIST number of the eliminated group is combined with this GN and added to the OUT table (starts at end of GAPL - grows towards FWAWORK). This table is used when translating the APLIST to COMPASS line images.
- 5.2.4 If the eliminated group was considered the first or prime group, its grouped APLIST is searched for any non-zero GN fields. If any are found, the OUT table is searched for each such GN and this GN in the OUT table is then replaced by the new GN. When translating to COMPASS, a non-zero entry in the GN field of an APLIST entry says, "Before this entry is translated to a COMPASS line image, the APLIST numbers that are linked to this GN in the OUT table must be placed in the temporary COMPASS file first."
- 5.2.5 This is continued until all APLIST groups have been compared.
- 5.3 The list is then translated into COMPASS line images.
- 5.3.1 An entry is picked from the GAPL to find parameters for a list that is to be translated to compass line images. Negative entries are ignored and a zero entry indicates the end of the GAPL.
- 5.3.2 When a usable GAPL entry is found, the number of this list is placed in the temporary COMPASS file as [APn. BSS 0 where n is the APLIST number.
- 5.3.3 The actual list is then output to the temporary COMPASS file. At selected times during this transfer, the available core is checked. If it is running short, the temporary COMPASS lines that have been formed this far are transferred and the pointers are reset. Each member of an APLIST is transformed in the following manner:
- 5.3.3.1 If there is a non-zero GN field, the OUT table is searched. Each entry in the OUT table that has this GN also contains an APLIST number of an eliminated that should be output to the temporary COMPASS file before this APLIST entry is processed.
- 5.3.3.2 Then the I field is examined. It can indicate either a statement temporary, a variable, or it can be an indication of non-standard return. Default goes to b.
- a) Statement temporary, (I=1) the H1 field is converted to display code, suffixed with a period and prefixed with the letters ST and added to the temporary COMPASS file.

CONTROL DATA CORPORATION . COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 28.4
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- b) Non-standard returns, (I=7) places a VFD 60/0 in the temporary COMPASS file. This is done so that substitution of actual parameter addresses at execution time terminates before any non-standard return addresses are substituted.
- c) Symbols, (I=0) the symbol is read from the symbol table. If it is a formal parameter, a flag is set saying it is necessary to output a substitution macro reference. The symbol along with any constant add in (CAfield) is placed in the temporary COMPASS file followed by a SUB macro reference if necessary.

5.4 Example:

Consider a subprogram as follows:

```
SUBROUTINE X (A,B,C)
DIMENSION A (50)
CALL T (A(40), B, C+1, 24)
CALL S (X11, Y, A)
CALL G (Y, A)
END
```

APLIST would be placed in the COMPASS file in the following format:

```
[AP1.BSS 0
VFD 60/A+47B
SUB A,47B
VFD 60/B
SUB B
VFD 60/ST1. (Statement Temporary)
VFD 60/CON.+nB
VFD 60/0
```

DOCUMENT CLASS IMS PAGE NO. 28.5
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

[AP2. BSS 0
 VFD 60/X11

[AP3. BSS 0
 VFD 60/Y
 VFD 60/A
 SUB A
 VFD 60/0

6.0 TABLE FORMATS

6.1 APLIST INPUT

APLAST

	12		18
0	0	I	H1

word 2 of
 APLIST entry

	12	18	14	16
AP1.	2nnn	CA	0	NO

word 1 of
 APLIST entry

nnn is the number of cells back from the present location in the jumbled APLIST that contains the next member of this APLIST (nnn=0 indicates the end of a particular APLIST).

NO, the APLIST group that this entry belongs to.

CA, I, H1, have the usual meaning.

6.2 GROUPED APLIST

FWAWORK	12	18	12	18
	GN	CA	I	H1

(GN initially zero)

↓
 APLAST

CONTROL DATA CORPORATION . COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 28.6
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

6.3 GAPL, Grouped APLIST parameters list.

APLAST

	24	18	18
APL.	0	FWA	LWA

FWA first word address of this group.
LWA last word address of this group.

6.4 OUT Out Table

	24	18	18
	0	ON	GN

GAPL

APL.

GN is a group number
ON is the number (output number) of an APLIST group
that has been eliminated.

6.5 CMPSTR, contains the first word address of the temporary
COMPASS file.

6.6 APNAME, contains name of last symbol if it was a formal
parameter.

6.7 APCA, CA field of last symbol output if it was a formal
parameter.

DOCUMENT CLASS IMS PAGE NO. 29.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

1.0 General Information

1.1 Task Description

The second pass D0 processor examines D0 begin and D0 end macro references, standard index function macro references and all R-list instructions generated within the innermost loop of a D0 nest provided the loop is well behaved [see section 8.2]. R-list instructions are generated to count D0 loops, reference standard index functions, and to materialize the control variable when necessary. The R-list is generated by considering the optimum use of B registers and all code in the D0 loop is altered to take advantage of B register assignments where possible.

2.0 ENTRY POINTS

2.1 PRODB

2.1.1 PRODB is referenced by PRE when a D0 begin R-list macro is encountered in the R-list buffer. PRODB determines the mode by examining FFLAG from the low memory communications region. If the compiler is in the first level of optimization mode, MACROE is called to generate loop counting R-list from the D0 begin pseudo R-list. If the compiler is in standard mode, flags are set for ensuing calls to PROIXFN and PRODE.

2.1.2 The calling sequence to PRODB is a return jump to PRODB. Flags and addresses needed are:

FFLAG - 0 if standard, non-zero if first level optimization
MACREF {RLIST} - address of the D0 begin R-list macro in memory
FWAWORK - address of the working buffer where R-list items may be stored. Overflow is checked against VARLAST.

2.2 PROIXFN

2.2.1 PROIXFN is referenced by PRE when pseudo R-list for a standard index function is encountered in the R-list buffer. PROIXFN breaks the index function into a table of terms and either preserves this table for PRODE by replacing part of the pseudo R-list [standard] or generates the R-list to compute and reference the standard index function [first level optimization].

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 29.2
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600

2.2.2 The calling sequence to PROIXFN is a return jump to PROIXFN. It is expected that the following variables in the communications region will be properly set:

MACREF {RLIST} - address of the D0 begin pseudo R-list macro memory
VARLAST - address of the next available call in the VARDIN buffer
FWAWORK - address of the working buffer where R-list items may be stored. Overflow is checked against VARLAST.

2.3 PRODE

2.3.1 PRODE is referenced by PRE when a D0 end R-list macro is encountered in the R-list buffer. For an ill-behaved loop {no optimizing attempted} MACROE is called to generate the instructions for the bottom of a D0 loop given the D0 end R-list macro. If the compiler is in standard mode and the well-behaved flag is set, then a series of ten scans or phases of optimization is executed. These scans and their functions are:

- I. Scan all R-list between the D0 begin and the D0 end R-list macros selecting candidates for available execution time B registers from among the load, store and set R-list instructions and the pseudo R-list macros for standard index functions. Scan I compiles candidate information in two word entries and places them in the A table {see section 6.0 - FORMATS}. A table items are linked together on the basis of type: constant, address, index function or variable increment {see section 8.0}.
- II. Scan the A table computing the cost in instruction parcels if a B register is not assigned this candidate. Assuming that the lowest number of parcels used generates the most efficient object code, B registers are assigned to the candidates having the largest cost. Scan II computes this cost for all constant, address and increment entries in the A table, marking each as to type and as a candidate. In each index function chain the member which appears in the largest number of instruction sequences is given a cost and marked as a candidate; the differences between the candidate and other group members are filed as candidates.
- III. Scan the A table and assign the B registers by generating a B table of up to seven entries. Registers are

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.3
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- assigned by largest cost until the A table is exhausted or the B table is full. If a B register is assigned to a constant, the constant chain is searched and wherever possible constants are marked to use the sum or difference of presently assigned B registers. If a B register is assigned to an address, the address chain is scanned and the differences between the candidate and other addresses are filed as candidates.
- IV. Scan the B table and decide from register assignments which one of 12 methods should be employed to count the D0 loop. The decision is based primarily upon the contents of the B registers. {See section 8.0 for loop counting methods.}
 - V. Scan the B table and mark the candidates in A with the register {or registers} that have been assigned to them.
 - VI. Scan the B table for A assignments. If the A table entry is a constant, address or difference type candidate set up the R-list instructions to load this register at the top of the D0 loop.
 - VII. Scan the A table for increments and index functions generating the initializing R-list instructions to pre-compute variable increments and to initialize and increment index functions.
 - VIII. Generate the D0 loop counting code selected by Scan IV. R-list instructions are generated to initialize and increment the loop count and, if necessary, the control variable. The 12 methods of counting are shown in section 8.0.
 - IX. Scan all R-list between the D0 begin and D0 end R-list macros creating R-list references to B registers when applicable. The general R-list given by pass I will be tailored to the B register assignments made by previous scans and will replace the general R-list passed on by PRE. The pseudo R-list index function macros will be expanded into the proper sequence of instructions generated for double precision and complex arrays.
 - X. Separates top of the loop R-list from end of the loop R-list and inserts the body of the loop. This is done by moving the TOP-END buffer just below VARDIM, putting the referencing R-list just below this. Top of the loop instructions are then extracted and moved to the beginning of the R-list buffer where the D0 begin

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.4
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

macro was. The body of the loop follows, then the end of the loop instructions. 'Ignore' op codes are squeezed out and any remaining negative op codes cause generation of normal R-list to replace them.

- 2.3.2 The calling sequence to PRODE is a return jump to PRODE. Addresses needed are:

MACREF {R-list} - address of the D0 begin R-list macro
in memory
VARLAST - address of the next available cell in the
VARDIM buffer
FWAWORK - address of the working buffer where R-list
items may be stored. Overflow is checked
against VARLAST.

3.0 Diagnostics

3.1 None

4.0 Environment

4.1 Not applicable

5.0 Structure

5.1 CANON

5.1.1 CANON generates and orders the table of index function terms {T} given the standard index function pseudo R-list macro produced by ARITH in pass one of the compiler. Each index function may have as many as five terms dependent upon the combination of variables and constants used in the subscript. CANON uses TERM and FILET.

5.2 TERM

5.2.1 TERM unpacks a single subscript. If the subscript contains a variable, exit is made to the second word following the return jump to TERM, otherwise the first.

5.3 FILET

5.3.1 FILET makes the actual entry of a term in T, combines terms with like variables, orders the table in decreasing order using all 60 bits as key for the sort and leaves the address of the next available location in the cell TN.

5.4 IXFN

DOCUMENT CLASS _____ IMS _____ PAGE NO. 29.5 _____
PRODUCT NAME FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

5.4.1 IXFN is called whenever a standard index function is encountered in the R-list. IXFN then searches the integer definitions following the D0 end macro to see if the subscript uses any of the variables that are redefined within the loop. If so, and it includes the control variable, the control variable is marked for materialization. If not, and the index function has not appeared before, it is filed as a candidate in the A table.

5.5 LINKA-NSRTA

5.5.1 LINKA files candidates for B registers in the A table, if not already there and links them to other candidates of the same type {address, constant, etc.}. If a candidate has already been filed, then the sequence in which it appears is noted for use in determining the value of having a B register assigned.

5.5.2 NSRTA files a candidate in the A table without linking and without checking for prior entry.

5.6 REF

5.6.1 Given the 0C, CA, S0, RI, H2, RF, I, H1 fields, REF files a three word type three R-list item in the R-list buffer area of memory. This R-list item will generate a reference to an array element at object time. If the array is double precision, a second type three R-list item will be placed in the buffer.

5.7 MRKIXFN

5.7.1 MRKIXFN chains through all index function entries in the A table by group, selecting the most popular member of each group {appearance in most sequences} and marking that member as a candidate for a B register and as a member of a group. A MRKIXFN then calls FORMDIF.

5.8 FORMDIF

5.8.1 FORMDIF inspects address and index function groups and if a difference is found between the inspected item and the head of a group does the following:

- 1} If the difference is constant the difference is filed in the constant chain.
- 2} If the difference is symbolic then an item is filed containing both symbols and the constant difference {if any}.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS TMS PAGE NO. 29.6
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

5.9 EVALCON

5.9.1 EVALCON evaluates a constant entry in the A table placing the number of parcels saved by using a B register in the value field of the A entry. It also marks the entry as a candidate and as a constant.

5.10 MRKCON

5.10.1 MRKCON scans the linked constant candidates in the A table marking each as a candidate and constant and computing the cost in parcels if a B register is not assigned the candidate.

5.11 MRKADR

5.11.1 MRKADR searches the address chain of the A table for the most popular candidate {used in most sequences} and then marks that entry as a candidate having a group and constant. The cost of not assigning a B register is stored in the value field of the candidate.

5.12 MRKINC

5.12.1 MRKINC scans the linked increment entries in the A table marking each as a candidate and computing the cost of not having a B register.

5.13 SRCHC

5.13.1 SRCHC is called to determine if a constant entry in the A table may be formed using constants already assigned to B registers. Given a constant which is the sum or difference of assigned constants, SRCHC chains through the constant chain for a candidate equal to it. If found, the two are related by setting the appropriate bits in the REG, REG2 and NEG fields in the A table entry found.

5.14 TOPB

5.14.1 Given a one or two word R-list item, TOPB adds a descriptor word and files a three word R-list item in the TOPEND buffer after insuring that the buffer won't overflow.

5.15 ENDB

5.15.1 Given a one or two word R-list item, ENDB adds a descriptor word and an end flag and files a three word R-list item in the TOPEND buffer after insuring that overflow won't occur.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.7
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- 5.16 VARBI
- 5.16.1 Given a one word R-list item, VARBI stores it in the VARDIM buffer.
- 5.17 VARB2
- 5.17.1 Given a two word R-list item, VARB2 stores it in the VARDIM buffer.
- 5.18 TDOWN
- 5.18.1 TDOWN displaces the T table lower by one storage address.
- 5.19 MOVETR
- 5.19.1 MOVETR moves the T table {through and including the first entry of all zeroes} to the last six words of the 12-word standard index function pseudo R-list, {see section 6.2 - the second table on the page.}
- 5.20 MOVERT
- 5.20.1 MOVERT moves the last six words of a standard index function pseudo R-list item to the first six words of the T table. {See section 6.2}
- 5.21 GENT
- 5.21.1 GENT generates the R-list to compute and reference a standard index function. This code replaces the standard index function pseudo R-list produced by ARITH in pass one of the compiler. GENT calls GENVAR.
- 5.22 GENVAR
- 5.22.1 GENVAR generates the R-list to compute the variable part of an index function. This code appears at the loop top or point of reference depending on the index function. Non-variable computations appear at the program top. GENVAR calls SUMC.
- 5.23 SUMC
- 5.23.1 SUMC generates the R-list to add up the coefficients of a single written variable and places this code in the VARDIM buffer. SUMC calls COEFF.
- 5.24 COEFF

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.8
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- 5.24.1 COEFF generates the R-list to multiply the factors of a coefficient and stores the generated R-list in the VARDIM buffer.
- 5.25 DOONE
- 5.25.1 DOONE generates a reference to a subscripted quantity based on the contents of the B registers. The generated R-list replaces the standard index function pseudo R-list.
- 5.26 GENALL
- 5.26.1 GENALL moves R-list down in memory and generates the R-list to compute and reference redefined index functions. GENALL calls GENT.
- 5.27 MATC
- 5.27.1 When the control variable must be materialized {updated in memory} during the course of a DO loop, MATC makes entries in the A table to allow the loop parameters to compete for B registers.
- 5.28 MCOST
- 5.28.1 MCOST determines the minimum cost for DO loop counting considering all possible situations and available methods.
- 5.29 CCOST
- 5.29.1 CCOST determines the cost of counting a DO loop by using the upper limit.
- 5.30 LOADX
- 5.30.1 Given an A table item, LOADX generates R-list instructions to load an X register with the item.
- 5.31 LOADY
- 5.31.1 Given a loop parameter, LOADY generates R-list instructions to load a register R with it.
- 5.32 STOREX
- 5.32.1 When counting a DO loop in memory and the count has been updated in an X register, STOREX generates R-list instructions to store the X register in memory.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.9
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

5.33 LOADBMC

5.33.1 Generates R-list to load the first limit and subtract the second limit of the DO statement. LOADBMC calls LOADY.

5.34 COUNTR

5.34.1 Generates the R-list to compute $\{C-B\}/D$ for counting the DO loop. Calls CONCNT.

5.35 COUNTNR

5.35.1 Generates the R-list to compute $\{B-C\}/D$ for counting the DO loop in the negative direction. Call CONCNT.

5.36 CONTOP

5.36.1 Generates R-list instructions to multiply R by an integer constant in the most efficient manner using shift and add combinations. R-list is stored in the TOP-END buffer.

5.37 Generates R-list instructions to multiply R by an integer constant in the most efficient manner using shift and add combinations where R is a variable division or product of variable dimensions. R-list is stored in the VARDIM buffer.

5.38 CONCNT

5.38.1 Determines whether the DO may be counted by a constant or not, i.e., whether $\{B-C\}/D$ is constant.

5.39 DIVCON

5.39.1 Generates R-list to compute $\{B-C\}/D$ optimally by checking constants {if any} and shifting or no division at all where possible.

5.40 LMCHK

5.40.1 LNCHK checks DO loop limits to recognize one trip loops and sets flags for section VIII to bypass the generation of loop counting code.

5.41 REDINC

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.10
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

5.41.1 REDINC checks for index functions with same increment having these conditions:

1. Not involved in counting the loop.
2. Not requiring two B registers to form index function.
3. Both index functions in B registers.
4. One incrementing instruction has already been generated.

When conditions are met, REDING generates code to compute difference of index functions which will replace index function in higher numbered B register. All references to index functions, other than the base, must then be generated as a sum of the base index function and the difference register. At least one parcel per use is saved as the incrementing instruction is eliminated.

Example:

	<u>GIVEN CODE</u>		<u>ALTERED CODE</u>
	SBI A		SBI A
	SB2 B		SB2 B
	SB7 A+5		SB2 B2-B1
}AA	SA1 B1		SB7 A+5
	BX7 X1	}AA	SA1 B1
	SA7 B2		BX7 X1
	SBI B1+1		SA7 B1+B2
	SB2 B2+1		SBI B1+1
	GE B7,B1,}AA		GE B7,B1,}AA

5.42 INIDV

INIDV generates the R-list to set an X register to the initial value to be given the induction variable {lower limit of the loop} and then generates the store instruction to materialize the variable in memory.

6.0 FORMATS

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 29.11 _____
PRODUCT NAME FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

DOPRE is concerned with two levels of optimization:

- 1} First level optimization where PRODB calls upon the MACRO expander {MACROE} to generate R-list for the top of a D0 loop. PROIXFN generates R-list to process the standard index functions in a D0 loop without benefit of B register optimization. PRODE calls upon MACROE to generate R-list for the bottom of the D0 loop.
- 2} Second level optimization results in PRODB and PROIXFN setting flags and addresses for PRODE to generate candidates for B registers at execution time. In addition, instructions generated within the loop during pass I by other processors are changed to take advantage of B register assignment at execution time.

In first level optimization only the table of index function terms {T} is generated. In second level optimization a table of candidates {A} and a table of B register assignments {B} is generated in addition to the table of index function terms {T}. The A table is a variable length table with two words used for each entry. Candidates may come from constants, increments, addresses, address differences, or index functions. The T table is one word per entry and a maximum of six entries in the table. The last entry must always be zero. The B table is a fixed table of seven entries, one word per entry, and each entry contains information related to the assignment of the corresponding B register. Thus, the third word of the B table designates how B₃ was assigned to be used at execution time. Certain fields of both A and B have double usage and these fields will be noted.

In addition to these tables, section 6.0 will be concerned with the standard index function pseudo macro before and after the call to PROIXFN and the TOP-END buffer created by PRODE in the second level of optimization.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
 DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.12
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

The tables and formats follow.

6.1 T TABLE { INDEX FUNCTION TERMS }

59	58	24	23	22	21	18	17	0
L	V1	V	V	V		C		
		2	3	4				

L - Set to 1 if V1 is the loop control variable

V1 - Base-bias of variable involved in the term

V2 - Set to 1 if the first dimension of the array ~~V1~~ is adjustable

V3 - Set to 1 if the second dimension of the array ~~V1~~ is adjustable

V4 - The I part {table number} where C{I} is variable and the H{BIAS} is found in the C field

C - Constant multiplier for the variable part of the term

B TABLE { B REGISTER ASSIGNMENTS }

59	29	0
VALUE/R		ALOC

VALUE - Savings in parcels made by the assignment of this B register

R - After Scan V of PRODE, R is register number defined as that B register

ALOC - Address of the A table entry for the candidate given this register

DOCUMENT CLASS IMS PAGE NO. 29.13
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

6.2 STANDARD INDEX FUNCTION PSEUDO R-LIST MACRO
 BEFORE PROIXFN

59	OC	47	NWF	30	NOT USED	18 16 15	TYPE	RI	0
NOT USED		CA		IH OF THE ARRAY					
NS	P	C	B			A			0
57	A B C	53	35	17					
MC OF FIRST SUBSCRIPT									
		CA		IH OF VARIABLE IN FIRST SUBSCRIPT					
AC OF FIRST SUBSCRIPT									
MC OF SECOND SUBSCRIPT									
		CA		IH OF VARIABLE IN SECOND SUBSCRIPT					
AC OF SECOND SUBSCRIPT									
MC OF THIRD SUBSCRIPT									
		CA		IH OF VARIABLE IN THIRD SUBSCRIPT					
AC OF THIRD SUBSCRIPT									

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.14
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

AFTER PROIXFN

59	47	30	17	15
OC	NWF	P NOT USED	R D I X P N	RI
CA 53	41 NOT USED 35	IH OF THE ARRAY 17		
NOT USED	ADPSUB	ACHAIN	LSUB	
NOT USED	ASUB	AINC	LINC	
FIRST DIMENSION OF THE ARRAY				
SECOND DIMENSION OF THE ARRAY				
T TABLE				
ENTRIES To - Tc				
{SEE T TABLE FORMAT}				

OC, RI, CA, IH are described in R-list literature.

TYPE - 3 bit type field as used in SYMTAB

NS - number of subscripts

PABC - indicates adjustable dimensions for subscript 1, 2 or 3 respectively

C, B, A - constants or IH of variables for dimensions of subscripts 3, 2, or 1 respectively

MC - multiplicative constant in subscript

AC - additive constant in subscript

ADPSUB - if array is double length points to second A table item for this index function

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.15
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- ACHAIN - points to the A table item which heads the chain for this unique index function
- LSUB - points to previous unique index function in R-list
- ASUB - points to A table item for this index function
- AINC - points to A table item for this variable increment
- LINC - points to the previous index function with unique local terms
- NWF - number of words following as part of this macro

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.16
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

**6.3 A TABLE { CANDIDATES FOR B REGISTERS }
 FOR INDEX FUNCTIONS**

	59	56	53	51		47		29	17		
WORD 1	C A N D	G R P	I X A D	REG	REG2	O N E	N E G	D I F F	VALUE/ADIFF	SEQ	LINK
WORD 2	H					CA		XXXXXX		RLOC	

FOR INCREMENTS

C A N D	G R P	I X A D	REG	REG2	O N E	N E G	D I F F	VALUE/ADIFF	SEQ	LINK	
								NAME	XXXXXX		RLOC

FOR CONSTANTS

C A N D	G R P	I X A D	REG	REG2	O N E	N E G	D I F F	VALUE/ADIFF	SEQ	LINK
THE 60 BIT CONSTANT										

FOR ADDRESSES

C A N D	G R P	I X A D	REG	REG2	O N E	N E G	D I F F	VALUE/ADIFF	SEQ	LINK
H					CA		XXXXXX		XXXXXX	

ADDRESS DIFFERENCES

C A N D	G R P	I X A D	REG	REG2	O N E	N E G	D I F F	VALUE/ADIFF	SEQ	LINK
H					CA		XXXXXX		H2	

CONTROL DATA CORPORATION . COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.17
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- H - the ordinal in the symbol table for the variable
- CA - the constant addend or displacement of the variable if any
- SEQ - a field of flags indicating which of 12 possible sequences the candidate is used in. Sequences are indicated from right to left
- LINK - address of next group member
- RLOC - address in the R-list for the head of the linked entries
- CAND - set to 1 if this entry is a candidate for a B register
- GRP - set to 1 if this candidate is a member of a group
- IXAD - set to 1 for address type candidates
- REG - number of the B register assigned {if any}
- REG2 - number of the second B register assigned {if any}
- ONE - set to 1 if increment of index function is constant
- NEG - relates B registers used to form a sum {} or difference {}
- DIFF - set to 1 if A item is difference of addresses and cannot take advantage of SB B+B
- NAME - I, H values of program temporary or loop temporary
- VALUE - cost in parcels if this candidate is not given a B register
- ADIFF - the address of the A table entry created as a difference with this A table entry

7.0 MACROS

- 7.1 ADDRR - {BUF, XRJ} Generates R-list to do an integer add of two X registers {R and XRJ} $IX\{R+1\} = X\{XRJ\} + X\{R\}$. R-list fields are OC, RJ, RK, RI found in IADD, XRJ, B7, B7+1 respectively. R-list is stored in an address determined from the index given as BUF. The macro uses TYPEI, OUTBUF macros and the subroutines TOPB, ENDB, VARB1, VARB2.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS TMS PAGE NO. 29.18
 PRODUCT NAME FORTTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- 7.2 DEFBR - {BUF, XREG} Defines the B register that will receive the information in the register specified by RI SB{XREG}=R. R-list fields are 0C, S0, RI found in DEFINE, LOCKB+XREG, B7 respectively. Type II R-list is stored in an address determined from the index given as BUF. This macro uses macro OUTBUF and the subroutines TOPB, ENDB, VARB1, VARB2.
- 7.3 DESB - DEXBR {BR1, XR2, XR3} Sets the NEG, REG and REG2 fields of the first word of an A table entry. Arguments are BR1, XR2, XR3 which represent a B register holding a one or zero, and X registers holding the two B registers assigned to be used as a sum or difference for generating a needed constant. The A table entry is located at L{DESB} or X{DEXBR}.
- 7.4 IMUL - Packs two integers given in X1, X2 and does a double multiply with the result left in Xb.
- 7.5 INCRR - {BUF} sets up an 18 bit {short} add of two B registers storing the result back into one of the registers {SB{R-1} = B{R-1} + B{R}}. R-list fields used to generate type I R-list are 0C, RJ, RK, RI which are given by SADD, B7, B7-1, B7-1. The R-list is stored in an address determined from BUF. Uses TYPEI, OUTBUF macros, TOPB, ENDB, VARB1, VARB2 subroutines.
- 7.6 LOADADR - {BUF, XIH, BCA} sets up the R-list for the load address instruction SA{R} = {IH+CA} R-list fields are 0C, CA, RI, I-HI found in LOAD, BCA, B7, XIH respectively. The type III R-list is stored in an address determined from the index given as buf. This macro uses the macro OUTBUF and subroutines TOPB, ENDB, VARB1, VARB2.
- 7.7 MULCON-- {BUF} sets up R-list to multiply a register specified by R by an integer constant. Instructions are stored at an address determined from BUF. Uses macros SHIFT, PACK, MULT, OUTBUF and subroutines TOPB, ENDB, VARB1, VARB2, CONTOP, CONVAR.
- 7.8 MULRR - {BUF} sets up the R-list to convert the integers in registers R and R-1, place in registers R+1, R+2, and then does a double floating point multiply with the result ending in R+3.
- | | |
|---------|---------------|
| PX{R+1} | R-1, B0 |
| PX{R+2} | R, B0 |
| DX{R+3} | {R+1} * {R+2} |

R-list fields are RK, RI, 0C, RK2, 0C2, RJ, RK3, RI3, 0C3 given by R-1, R+1, PACK, R, R+2, PACK, RI1, RI2, R+3, DMUL respectively. The generated R-list is stored in an address

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.19
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

determined from the index BUF. Calls macro OUTBUF and uses subroutines TOPB, ENDB, VARB1, VARB2.

7.10 SADDR - {BUF, XRJ} sets up Type I R-list to do a short add of two registers with the result going to a third register. Fields needed are: RJ, RK, RI, OC, given by R, XRJ, R+1 and SADD respectively. R-list storage address is determined from BUF. Calls on macros TYPEI, OUTBUF and subroutines TOPB, ENDB, VARB1, VARB2.

7.11 SETRCON - {BUF, BRI, XCON} generates R-list to set a register to a constant value. Fields needed are: IN, OC, RI given by XCON, SETII, BRI respectively. The type II R-list is stored at an address determined from BUF. SETRCON uses the macro OUTBUF and calls on subroutines TOPB, ENDB, VARB1, VARB2.

7.12 SETADR - {BUF, XIH, BCA, XRI, XRF, XH2} generates R-list to set a register to an address. Fields needed are: CA, RF, IH, RI, H2, OC given by BCA, XRF, XIH, XRI, XH2, SETIII respectively. The type III R-list is stored at an address determined from BUF. Uses the macro OUTBUF which in turn calls upon subroutines TOPB, ENDB, VARB1, VARB2.

7.13 STORADR - {BUF, XIH, BCA} generates type III R-list to set Ab or A7 to an address causing a corresponding store of Xb or X7 respectively. Fields needed are IH, CA, OC given by XIH, BCA, STORE respectively. The generated R-list is stored in a buffer determined from the index BUF. STORADR uses the macro OUTBUF which uses subroutines TOPB, ENDB, VARB1, VARB2.

7.14 SRBMB - {BUF, XB1, XB2} sets up the R-list for the short subtract and stores the result into the R register $\{SB\{R\} = RBTAB+XB1 - RBTAB+XB2\}$. R-list fields used to generate this type one R-list item are OC, RJ, RK, RI given by SSUB {67}, RBTAB+XB2, RBTAB+XB2, R respectively. XB1, XB2 are indices for the B table which contains the designated R that is assigned the B register. The generated R-list is stored in an area determined from the argument BUF. This macro uses the OUTBUF macro and TOPB, ENDB, VARB1, VARB2 subroutines.

7.15 SRBPB - {BUF, XB1, XB2} sets up the R-list for the short add and stores the result into the R register $\{SB\{R\} = RBTAB+XB1 + RBTAB+XB2\}$. R-list fields used to generate this type one R-list item are OC, RJ, RK, RI given by SADD{46}, RBTAB+XB1, RBTAB+XB2, R respectively. XB1, XB2 are indices for the B table entry that contains the R assigned

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.20
PRODUCT NAME FORTTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

to B register XB1 or XB2. The generated R-list is stored in an area determined from the argument BUF. This macro uses the OUTBUF macro and TOPB, ENDB, VARB1, VARB2 subroutines.

- 7.16 BPMB - {BUF, XRJ, XRK} sets up R-list for a short load, store or difference with the result going into the RI register. R-list fields used to generate this type I R-list item are OC, RI, RJ, RK given by B4, OP, XRJ, XRK respectively. The generated R-list is stored in an area determined from BUF. This macro uses the OUTBUF macro and TOPB, ENDB, VARB1, VARB2 subroutines.
- 7.17 SUBRR - {BUF, XRJ} integer subtract of two X registers {R and XRJ} $IX\{R+1\} = X\{XRJ\} - X\{R\}$. R-list fields are OC, RJ, RK, RI found in ISUB, XRJ, B7, B7+1 respectively. R-list is stored in an address determined from the index given as BUF. This macro uses TYPEI and OUTBUF macros and the subroutines TOPB, ENDB, VARB1, VARB2.
- 7.18 SUBXRR - {BUF} same SUBRR except $IX\{R+1\} = X\{R\} - X\{R-1\}$.
- 7.19 DIVRR - {BUF} integer division of two registers $X\{R+1\} = X\{R-1\}/X\{R\}$. R-list fields are OC, RJ, RK, RI given by IDIV, B7-1, B7, B7+1 respectively. R-list is stored in an address determined from the index given as BUF. This macro uses the macros TYPEI and OUTBUF which uses subroutines TOPB, ENDB, VARB1, VARB2.
- 7.20 JUMP - {BUF, BOC, BRI, BRF} generates the loop ending jump as type III R-list given OC, RI, RF as BOC, BRI, BRF respectively and the IH of the label for the top of the D0 is found in the D0-END pseudo R-list generated by pass I. The jump instruction is stored at an address determined from BUF. JUMP uses OUTBUF which uses TOPB, ENDB, VARB1, VARB2.
- 7.21 TYPEI - {BUF, XRJ} generates a type I R-list instruction from the fields OC, RJ, RK, RI given by B4, XRJ, B7 and X3 respectively. The generated R-list is stored at an address determined from BUF and this macro uses the macro OUTBUF which in turn uses subroutines TOPB, ENDB, VARB1, VARB2.
- 7.23 SHIFT - {BUF, XCOU} sets up R-list to do a shift transmit of R to R+1 and a constant shift left the number of places indicated by register XCOU for R+1. The resulting R-list is stored in memory as determined from BUF using the macro OUTBUF and subroutines TOPB, ENDB, VARB1, VARB2.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 20.21
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- 7.24 PACK - {BUF} generates R-list to pack the exponent zero with the fraction R into a register R+1. Uses OUTBUF, TOPB, ENDB, VARB1, VARB2 to put the R-list at an address determined from BUF.
- 7.25 MULT - {BUF} generates R-list to do a double precision floating multiply of R and R-2 leaving results in the register specified by R+1. Places instructions at address determined from BUF by using macro OUTBUF which uses subroutines TOPB, ENDB, VARB1, VARB2.
- 7.26 SADXRR - {BUF, XRJ} sets up type I R-list to do a short add of two registers with the result going to one of the registers {SRI = Ri + XRJ}. Fields used are RJ, RK, RI, OC, given by R, XRJ, R and SADD respectively. R-list storage address is determined from BUF. Calls on MACRO's TYPEI, OUTBUF and subroutines TOPB, ENDB, VARB1, VARB2.
- 7.27 XMIT - {BUF} sets up type I R-list to transmit from an input register to an output register. Fields used are RJ and RI given by R and R+1 respectively. R-list storage address determined from BUF. Calls on the macro OUTBUF and subroutines TOPB, ENDB, VARB1, VARB2.
- 7.28 ADDXRR - {BUF, XRJ} Generates R-list to do an integer add of two X registers {R and XRJ} and put the result back into R. R-list fields are OC, RJ, RK, RI found in IADD, XRJ, B7, B7 respectively. R-list is stored in an address determined from BUF. The macro uses TYPEI, OUTBUF macros and subroutines TOPB, ENDB, VARB1, and VARB2.
- 7.29 SADZRR - {BUF, XRJ} Generates R-list to set an X register to the value in a B register. XRJ is the R number for the B register and R is the X register number. Fields needed are RJ, RI given by XRF and R. R-list storage determined from BUF. Calls on Macro OUTBUF and subroutines TOPB, ENDB, VARB1 and VARB2.
- 7.30 CONLS - {BUF, XCOU, XREG} Generated R-list to do a constant left shift of XREG the number of places specified by register XCOU. Operation code given by KLS and BUF determines R-list storage. Uses the macro OUTBUF and subroutines TOPB, ENDB, VARB1 and VARB2.
- 8.0 DOPRE

DOCUMENT CLASS _____ IMS _____ PAGE NO. 29.22
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600

8.1 Principles

Sample test cases using straightforward compilation techniques indicate that 50% or more of the running time of FORTRAN benchmark programs is spent inside innermost DO loops, although they occupy under 10% of program space. Therefore, reduction of the time spent in loops, particularly inner loops, is of the first importance.

However, since the time required for a short loop is greatly reduced by retaining it in the stack, it is important to reduce the space required within loops as well. To avoid special cases, DOPRE concerns itself with space reduction only, assuming that time reduction is usually a by-product.

8.2 Definition of Well-behaved Loops

Loops containing extended ranges, input-output statements, CALL's, FUNCTION references, or calls or arithmetic statement functions which contain CALL's or FUNCTION references, or implicit subroutine calls, are not considered well-behaved and are not optimized for the following reasons:

- a. The existence of an exit and return to the loop makes retaining results in registers throughout the loop impossible;
- b. The existence of an unconditional jump in the loop makes retaining the entire loop in the stack impossible.
- c. The computation outside the loop, which may be a considerable proportion of the compute-time involved, does not benefit from the loop optimization.
- d. The existence of most of the above situations creates implicit definition points which complicates the analysis.
- e. The existence of an extended range requires the use of program-wide temporaries.

Currently, a loop must meet three other requirements in order to be optimizable:

- f. It must fit in memory.
- g. It must be an innermost loop. Outer loop optimization complicates the analysis considerably, but saves less execution time and is less likely to produce in-stack operation than innermost optimization.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.23
PRODUCT NAME FORTTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

- h. The increment must not be the control variable. This situation is rare but when it exists it makes calculation of the increments of local index functions outside the loop impossible.

8.3 Objectives of DOPRE

DOPRE generates code

- a. to compute and reference all standard index functions
- b. to count D0 loops
- c. to materialize control variables where necessary
- d. to provide optimization features if requested

PRE calls DOPRE when it encounters a D0 begin or D0 end macro, or a standard index function pseudo-macro.

If no optimization is requested, DOPRE expands each macro and returns to PRE. Standard index functions are reordered to minimize the computations required, and computations involving adjustable dimensions are done at the program top.

The code generated for D0 I=B, C, D is:

B Constant

Set R to B
Store R in I

B Variable

Load R with B
Transmit R to R
Store R in I

The code generated for D0 end is:

D Constant

Load R₁ from I
R₁ + D to R₃ or
Store R₃ in I

D Variable

Load R₁ from I
Load R₂ from D
R₁ + R₂ to R₃
Store R₃ in I

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.24
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

<u>C Constant</u>		<u>C Variable</u>
R3 - {C+1} to R4		Load R4 from C
	or	
NG R4 L		R4-R3 to R5
		PL R5 L

If standard optimization is requested, DOPRE allows PRE to form an R-list buffer containing the above macros and pseudo-macros and the expended R-list items for an entire well-behaved loop and the first sequence outside the loop. It analyses this information in detail, modifies it, and returns it in final form to PRE.

The analysis provides the following broad features:

1. Standard index functions which do not change in the loop {global} are computed once outside the loop.
2. Standard index functions which change in the loop only when the control variable changes {local} are initialized outside the loop and incremented within the loop.
3. Variable increments of local index functions are computed outside the loop.
4. Standard index functions containing variables changed in the loop {redefined} are computed at the point of reference.
5. The adjustable parts of all standard index functions, local, global or redefined, in a loop or out, are computed at subprogram top.
6. B registers are initialized at loop top with constants, addresses, index functions, and variable increments chosen to reduce space in the loop.
7. Instructions which reference the sums and differences of B registers are used to reduce space further.

The quantities loaded in B registers are selected to minimize the space required by the loop, in accordance with Section 1 above.

The following quantities are initially candidates for B registers:

1. Index functions none of whose variables are redefined in

DOCUMENT CLASS IMS PAGE NO. 29.25
 PRODUCT NAME FORTTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

the loop.

2. Variable increments required for modifying index functions containing the current control variable.
3. Addresses of variables in load and store instructions.
4. Constants in Type II Set instructions.
5. Loop limits for use in materializing the control variable and/or counting the loop.

B.4 How Index Functions Are Computed

The general form of a standard index function is:

```
DIMENSION A{L,M,N}
...A {aI+d, bJ+e, cK+f}
```

{L, M, and N are each either constant or adjustable; a, b, and c are positive integer constants; d, e, and f are signed integer constants. I, J, and K are variables.}

The above reference necessitates computing the address

$$A + \{aI + d - 1\} + \{bJ + e - 1\} \times L + \{cK + f - 1\} \times L \times M$$

DOPRE reduces the subscript to a canonical form containing the following items:

1. The array name, e.g. A
2. The constant addend CA, e.g., d-1 in the example {assuming L & M are adjustable}.
3. From zero to 5 additive terms, each containing from 1 to 3 variables.

Assuming all dimensions are adjustable and I, J, and K are different, the terms in the example would be aI, bJL, {e-1}L, cKLM, and {f-1} LM.

Terms are sorted so that

1. Index functions which look different, but require the same computations, are recognized, e.g., A{2×I+1} and D{I+1} where D is double length.
2. Non-adjustable variables appearing twice require at most one multiplication, e.g., A{2×I,1} yields A+{L+2}×I+CA, not A+2×I+L×I+CA; {CA=-L-1}.

DOCUMENT CLASS _____ IMS _____ PAGE NO. 29.26
 PRODUCT NAME FORTRAN Extended _____
 PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

DOPRE computes the constant part of the index function. It generates code at subprogram initialization to compute the adjustable part of the index function

Multiplication by some integer constants is simulated using shifts and adds. The special cases identified are:

1. Negative constant. An instruction BD-R to R is issued and positive multiplication proceeds.

2. constant = 1. No generation

3. constant = 3.

R1+R1 to R2. R1+R2 to R3

4. constant = 6

R1+R1 to R2. Then use code for 3.

5. constant = 2ⁿ.

Shift transmit and left shift n

6. constant = 2ⁿ+2^m. {0...010...010...0}

Shift transmit, left shift n-m, add, shift transmit, left shift m

7. constant = 2ⁿ-2^m {0...01...10...0}

subtract instead of add

B.5 Program flow

When fast compilation is requested PRODB, PROIXFN, and PRODE merely generate R-list in line. When standard compilation is requested, most of DOPRE's work is done in 10 sections of code within PRODE. Section I scans the R-list for the entire loop and forms a table of candidates. Section II computes the value of each candidate - the number of bits saved inside the loop by assigning it to a B register. Section III assigns the most promising candidate to the 1st B register, recomputes values if they are altered, assigns the next B register, etc. until there are no more candidates or no more B registers. Section IV determines the most space-saving way of materializing the induction variable {if necessary} and counting the loop, and may alter B? accordingly. Section V records the final assignments.

DOCUMENT CLASS IMS PAGE NO. 29.27
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

Sections VI, VII, and VIII generate code in another table to: load addresses and constants; compute index functions and increments; and materialize and count, respectively. Section IX re-scans the R-list for the loop, changing references to refer to B registers. Section X merges the generated code with the original R-list. Control returns to PRE.

B.6 Evaluating Candidates

As Section I forms the potential candidate list, it notes in which sequences each candidate was referenced, notes uses of B registers already present in the R-list, and groups index function and address candidates.

Where Z is a candidate

SEQ {Z} = a bit pattern containing a bit in the n^{th} position if Z is referenced in the n^{th} sequence.

L = the number of bits in SEQ {Z}

DOPRE cannot tell what the final code will be; it assumes that quantities in X registers are loaded only once per sequence in which used.

Section II evaluates the candidates. The 'value' of a candidate is the difference between the number of bits required for instructions to load and/or reference it within the loop if it is not assigned a B register, and the number required if it is assigned a B register.

a) Addresses in Load A and Store A Instructions

All addresses become candidates. After an address is assigned a B, the differences between it and each other required address become candidates also. That is, if Y and Y + 1 are referenced, the code will be one of the following:

	<u>Number of Registers Assigned</u>				
	0	1	1	2	2
set B ₁ to	-	Y	Y+1	Y	Y+1
set B ₂ to	-	-	-	1	1
reference Y	ref Y	ref B ₁	ref Y	ref B ₁	ref B ₁ -B ₂
reference Y+1	ref Y+1	ref Y+1	ref B ₁	ref B ₁ +B ₂	ref B ₁

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. _____ 29.28
 PRODUCT NAME _____ FORTRAN Extended _____
 PRODUCT MODEL NO. _____ 4P616 _____ MACHINE SERIES 64/65/6600 _____

Differences are used because of the likelihood that constant differences {and possibly symbolic differences} may occur several times. In the above example, the constant 1 may be referenced directly and may occur also as the difference of A and A+1, B and B+1, C+1 and C+2, etc.

As can be seen from the above chart, the value of an address is 15L, and the value of a difference is also 15L. {Symbolic differences are not formed when either symbol is a formal parameter.}

- b} Constants in Set X Instructions, as increments, as differences, as loop limits. All constants are candidates {negative constants are filed in positive form}. Whenever a constant is assigned a B register, Section III scans the other B registers assignments for other constants and determines which, if any, useful constants may be formed as the sum or difference of assigned constants. Suppose the constants 1, 2, 3, 4, 5, 6, 10 and 50 are required. Suppose also that B register assignments of constants are as follows:

B1	4
B3	1
B6	50

References to 5 are generated as B1+B3. References to 3 are generated as B1-B3. References to 2 are generated as B3+B3. Although the assignment of, say, 3, 2, and 50 would have been preferable {because 1, 4, 5, and 6 can be formed} no attempt is made to assign the optimum constants.

The value of a constant is 15L, since 'SX1 B1' is 15 bits shorter than, e.g., 'SX1 4'.

- c} Variable Increments required for modifying local index functions. With the index function in B2 the code required is:

	No B	1B
Instruction:	SAI INC	-
	{SB2 X1+B2 or SX6 X1+B2}	{SB2 B2+B1 or SX6 B2+B1}
Cost:	30	0
Value:	30	

DOCUMENT CLASS IMS PAGE NO. 29.29
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

d} Index Function

All index functions with identical variable parts comprise a 'group', e.g., $X\{I+5\}$, and $Y\{I\}$, are in the same group; except that each array which is a formal parameter has its own group.

Index functions are grouped because costs may be reduced by using the same memory cell or B register for more than one of them; e.g., if $X\{I\}$ is in B1, $Y\{I\}$ may be loaded as follows:

SAI Y-X+B1

and this may minimize space when there aren't enough B registers to go around.

Formal parameter arrays have their own groups because references like $Y-X+B1$ would otherwise have to be computed at run time.

The value {in bits, within the loop} of an index function depends on how it is computed and referenced.

There are six ways to compute/reference $X\{I\}$ where $Y\{I\}$ is also required:

	<u>Outside Loop</u>	<u>Reference</u>
In Memory	X+I to LTEMP	SAI LTEMP ref XI
In B	X+I to B1	ref B1
Share Memory	Y+I to LTEMP	SAI LTEMP ref X-Y+XI
Share B	Y+I to B1	ref X-Y+B1
Difference Memory	Y+I to LTEMP	
	X-Y to B2	SAI to LTEMP ref XI+B2
Difference B	Y+I to B1	
	X-Y to B2	ref B1+B2

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 29.30
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600

The initial value of a group of index functions is the cost of having no B register for the group minus the cost of having one B register for the group. This is the same as the cost of loading the index function in each sequence in which any member of the group is referenced, or $\exists 0 \times \#$ of bits in $\{SE0 v \dots v SE0n\}$, plus incrementation cost {for local index functions} of 45.

The differences between the candidate {the most-loaded group member} and each other member of the group also become candidates with values of 15L.

8.7 Assignment of B registers

Section II performs the initial evaluation of candidates. Section III assigns the most valuable candidate to B1, re-evaluates as necessary and assigns the next highest candidate to B2, etc., until there are no more B registers or no more candidates. The first time an address is assigned the differences with other addresses are filed. Each time a constant is assigned, combinations of it and previously assigned constants are generated.

8.8 Loop Control Code and Materialization

Subroutine MATC following Section I, and Sections IV and VIII select and generate the code to test at the bottom of the loop and to materialize the control variable if required.

MATC files the loop limits as candidates if materialization is required. Section IV chooses the best code for materialization and testing based on the assignments to B registers made in Section III. Section IV may alter assignments to produce better code.

Section VIII generates the materialization and testing code. The code produced is one of the following:

SECOND LEVEL OF OPTIMIZATION MODE LOOP COUNTING

{where the general form of the DO is DO SN I=B,C,D}

1. Count the loop in memory

<u>TOP</u>	<u>END</u>
$\{\frac{c-b}{d}\} \rightarrow LTEMP$	SAI LTEMP
	SXB XI-1
	SAB LTEMP
	PL XB, TOP

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
 DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.31
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

2. Count the index function to zero

<u>TOP</u>	<u>END</u>
{-mc+mb} → Bi	{increment Bi}
	GE B ₀ , Bi, LABEL

3. Count the index function up to mc {multiplier times upper limit}

<u>TOP</u>	<u>END</u>
{index function} → Bi	{increment Bi}
mc → B ₇	GE B ₇ , Bi, LABEL

4. Count in B₇ where D is a B register

<u>TOP</u>	<u>END</u>
{B-C} → B ₇	SB ₇ B ₇ +BD
	GE B ₀ , B ₇ , LABEL

5. Count in memory where D is in a B register

<u>TOP</u>	<u>END</u>
{B-C-I} → LTEMP	SAI LTEMP
	SXB XI+BD
	SAB LTEMP
	NG XB, LABEL

6. Count in B₇ where there is a 1 in a B register

<u>TOP</u>	<u>END</u>
{ $\frac{B-C}{D}$ } → B ₇	SB ₇ B ₇ +B _{one}
	GE B ₀ , B ₇ , LABEL

7. Count in memory where there is a 1 in a B register

<u>TOP</u>	<u>END</u>
{ $\frac{B-C}{D}$ } - 1 → LTEMP	SAI LTEMP
	SXB XI+B _{one}
	SAB LTEMP
	NG XB, LABEL

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
 DIVISION

DOCUMENT CLASS IMS PAGE NO. 29.32
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

8. Count the loop B_7

```

      TOP                                END
      {  $\frac{C-B}{D}$  } B7                    SB7 B7-1
                                           GE B7, B0, LABEL
  
```

9. Test the control variable I with C loaded

```

      TOP                                END
                                           {load C}
                                           IX0 Xc-Xi
                                           PL X0, LABEL
  
```

10. Test the control variable I with C+1 loaded

```

      TOP                                END
                                           {load C+1}
                                           IX0 Xi-Xc-1
                                           NG X0, LABEL
  
```

11. Test the control variable I where count {C} is materialized

```

      TOP                                END
                                           SX0 Xi-C-1
                                           PL NG X0, LABEL
  
```

12. Test the control variable I where the address of count {C} is in a B register

```

      TOP                                END
                                           SAi Bc
                                           IX0 Xc-Xi
                                           PL X0, LABEL
  
```

DOCUMENT CLASS _____ IMS _____ PAGE NO. 29.33 _____
PRODUCT NAME FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

8.9 Generation

Section V marks the final B register assignments in the A table. Section VI generates code at loop top to load addresses, constants, and symbolic differences.

Section VII generates code at loop top to initialize index function and variable increments of local index functions, and code at loop end to increment local index functions. Section IX re-scans the original R-list, changing the references to reflect the B register assignments and generating adjustable computations in redefined index functions. Section X merges the loop top and loop end code with the R-list for the body of the loop, and generates code in-line to compute and reference redefined index functions.

The program then exits to PRE.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 30.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

SQUEEZE

1.0 General Information

SQUEEZE resides in PASS2 as part of PRE. The primary function of SQUEEZE is to purge from a sequence superfluous loads and stores and redundant computation.

2.0 Usage

2.1 SQUEEZE

2.1.1 Calling SQUEEZE will result in the indicated R-list being modified to the extent of indicating operations which are to be eliminated and modifying the remainder to compensate for their removal.

2.1.2 Calling Sequence

SQUEEZE is called by placing the number of R-list entries in NORLIST and the first word address of the sequence in FWARLIST and the first word address of the descriptors in FWADESC, further the R-list sequence must be arranged as is done by COPY. All of these cells are set up by calling COPY.

2.1.3 Processing

The sequence is scanned for the definition of an RI. The subsequent R-list entries are scanned for definitions of a different RI, R', by an identical operation. If such a case is encountered, the operation defining R11 is marked for elimination by setting the KILL bit in the descriptor. The R-list following the definition of RI' is scanned for references to RI' and these are replaced with references to RI.

2.2 COPY

2.2.1 COPY moves the specified sequence into the working storage area. In the process of moving, end of statement and end of sequence markers are eliminated and all of the remaining R-list entries are split apart so that all of the first words of the sequence are followed by all of the second words of sequence are followed by all of the descriptors. The working storage markers are adjusted to protect the sequence.

2.2.2 Calling Sequence

COPY is entered by a return jump with NORLIST containing the number of R-list entries in the sequence and FWARLIST containing the first word address of the sequence. If the operation failed due to lack of working storage X1 will be negative on return, otherwise not.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 30.2
PRODUCT NAME FORTTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

2.2.3 Processing

The original sequence is scanned and the number of executable R-LIST entries is calculated; from this the beginning address for each of the three blocks if working storages is calculated. The move takes place and the working storage limits modified.

2.3 PURGE

2.3.1 PURGE takes a sequence in COPYed format and squeezes it in place removing all entries which have the KILL bit set in their descriptor.

2.3.2 Calling Sequence

PURGE is called by a return jump with NORLIST containing the number of R-list entries in the sequence and FWARLIST containing the address of the first word of the sequence. These will be set up by SQUEEZE.

2.3.3 Processing

The operation is performed in two passes. The first pass squeezes each of the first, second and descriptor word groups in place. The second pass moves the blocks next to one another.

2.4 REPLACE

2.4.1 Calling REPLACE will result in all R fields referring to the RI field of the just killed operation being modified to refer to the master RI.

2.4.2 Calling Sequence

REPLACE is called by a return jump with the following register settings:

A0 - first word address of sequence-1
X3 - first word of the killed entry
X4 - second word of the killed entry
B4 - first word address of descriptor - 1
B5 - master RI

2.4.3 Processing

Each entry in the R-list following the killed entry is scanned to see if it refers to the just killed RI. If such a R-field is encountered it is replaced with the master RI.

3.0 Diagnostics Produced

The COPY routine will set a flag before making a fail exit. On completion of the subprogram this flag should be interrogated and the information diagnostic "MORE MEMORY WOULD PRODUCE BETTER CODE" should be issued.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. 30.3
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

4.0 Environment

FMAWORK and LMAWORK must contain the limits of working storage.
FMAWORK will be increased by COPY and decreased by PURGE. The cells
are located in PRE.

6.0 FORMATS

Everything operated on is in standard R-list format except that it is
reshuffled as described in the COPY description.

7.0 Modification Facilities

In COPY the end of statement and end of sequence operation codes are
set-up with EQU's EOSTMT and EOSEQ respectively.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 31.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

Task Description: OPT

1.0 General Information

1.1 The function of OPT is to determine the order in which instructions are to be issued and to fix register assignments. The basic input to OPT is an R-list sequence and its associated dependency tree; the output is a series of calls to POST, each call giving the next R-list entry and register assignments to be converted to COMPASS line images. OPT is located in Pass 2.

2.0 Usages. OPT has one control entry point, OPTA, and two information entry points, LASLBL and LASTR.

2.1 Entry Point Name: OPTA

2.1.1 Entry Point Function: PROSEQ calls OPT at the entry point OPTA to cause an R-list sequence and associated dependency tree to be converted to COMPASS line images with fixed register assignments and code ordering appropriate to the target computer.

2.1.2 Calling Sequence and Returns: OPTA is entered by a return jump from PROSEQ. On return, if Xb is positive, successful processing of the entire sequence has taken place. If Xb is negative, the attempt has been unsuccessful, no COMPASS output has been produced, and remedial action {currently sequence halving} must be taken.

2.1.3 Processing Flow Description: When OPTA is entered, all working cells are initialized according to the flag word, LASTR, which describes the terminal entry in the last previous sequence, and OPNPOST is called. Control is then transferred to the OPTB section which determines the next R-list entry to be processed and which, if any, registers are to be used. This information is passed on to ISSUE which updates all counts, tables, and clocks as required and calls POST to cause production of the desired COMPASS line image. The cycle, OPTB to ISSUE to POST to OPTB, continues until the last R-list entry in the sequence has been processed or until it is determined that OPT cannot produce code successfully for the complete sequence. If production was successful, ISSUE will call CLSPOST to transfer the generated line images to the COMPASS file {COMPS} and exit to PROSEQ with Xb positive. If the conversion attempt was unsuccessful, control is returned to PROSEQ with Xb negative.

2.2 Entry Point Name: LASLBL

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 31.2
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

2.2.1 Entry Point Function: LASLBL is a flag word containing the R-list word specifying the last label encountered. This is set ordinarily by OPT at CLSPOST time but will also be set by PROSEQ for a sequence consisting solely of a label. LASLBL is used by OPTA during the initialization process.

2.3 Entry Point Name: LASTR

2.3.1 Entry Point Function: LASTR is a flag word containing the first word of the R-list entry terminating the last sequence if it was successfully processed or a zero otherwise. This is set by OPT before return to PROSEQ and by PROSEQ for a single-entry sequence. LASTR is used by OPTA to determine which working cells must be initialized.

3.0 Diagnostics: No diagnostics are produced by OPT.

4.0 Environment: The following cells are required to be set up before OPT is called:

<u>Cell Name</u>	<u>Contents</u>	<u>Producing Processor</u>
FWARLIS	first word address of R-list sequence being processed.	COPY
NORLIST	number of R-list entries in current sequence.	PURGE
FWATREE	first word address of the dependency tree.	BUILDDT
TREELNG	length of the dependency tree.	BUILDDT

In addition, OPT references all the 10 entry points in POST and the entry point NFPUNT in PRE.

5.0 Structure: The three major areas of OPT are OPTA, OPTB, and ISSUE.

5.1 OPTA

OPTA performs two main functions which prepare the environment for OPTB and ISSUE:

- 1) clearing out cells containing unwanted information left over from the last sequence processed.
- 2) processing R-list defines and initializing flag words.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 31.3
PRODUCT NAME FORTTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

5.1.1 The contents of LASTR determine whether OPTA performs the clearing function.

Clearing results when LASTR = zero, return jump, entry, label.

The following cells are cleared:

A-Register Contents
X-Register Contents
Register Availabilities (packed zeros)
Scratch Registers
Function Unit Availabilities
CLOCK, PARCEL
NXTSTOR, NXTLOAD
X6AVAIL, X7AVAIL

In addition the following cells are set:

FCHAR = 45B (this forces upper for the first generated COMPASS instruction)

ADR = 1 (this defines the contents of A0 to be R=1)

5.1.2 OPNPOST is called for every sequence. NORLISTR, FINALR, TREETOP, TPRIME, IPOINT and SIZEFALT are initialized.

Defines are processed by storing the RI field in the corresponding Register Contents Cell, packing the descriptor 'uses' field into the Register Availability word, removing the defined register from the scratch list, and calling PRUNE to remove the RI from the tree of unprocessed R-list entries.

The descriptors for the sequence are scanned to initialize CODSIZE (total parcel count) and NOSTORE (number of store instructions). X6 and X7 are removed from the scratch register list if there are more than two stores, and X7 is removed if there are exactly two.

The interword time, WRDTIME, is set to 1 if the target machine is the 6400 or if in-stack timing is to be used on the 6600. This latter condition exists when CODSIZE 28 and the last entry in the sequence is a conditional branch to the most recently encountered label. Otherwise, WRDTIME=5.

Exit is to OPTB at OPT.1B.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 31.4
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600 _____

5.2 OPTB

OPTB selects the next R-list entry to be processed and determines which registers are to be used. This information is transmitted to NOCODE if no COMPASS code is to be produced or at one of the ISSUE entries. If no issuable entry can be found for the current time and parcel in the object machine, exit is made to ISSUEI with the pointer, IPOINT, negative to request remedial action by ASYNCH.

OPTB starts by scanning the dependency tree, starting at TPRIME, looking for the first entry which will fit at the current parcel and all of whose predecessors have already been issued, i. e., an R-list entry all of whose predecessor fields in the tree have been set to zero. Such an entry is said to be logically issuable (LI).

Each LI entry is then checked to see if it is machine issuable (MI), i.e., to see if there is a function unit and uncommitted destination (result) register available at the current simulated clock time. (For 6400 code selection, functional units and result-registers are always available). The first check is for availability of the function unit. If none is found, the dependency tree is further scanned looking for LI entries. Most of the OPTB code is concerned with location of a destination register. If one can be located, then the entry is MI. If the instruction is a jump or a store, no destination register is involved. If a specific destination can be determined from the SO field in the R-list entry or if the instruction precedes a register store specifying the destination, it is necessary only to check whether the register is otherwise uncommitted and can be used for a result at current clock time. If neither of the above conditions exists, all feasible registers are checked for remaining uses and for availability at current clock time. The uses check includes the possibility of using a source (operand) register as the destination register. If a destination register is located under the above conditions, the instruction is MI.

Next the instruction is checked to see if it is machine executable (MX). If it is a register store or a dummy transmit, exit is made to NOCODE with IPOINT designating the instruction. Otherwise, the entry is MX if its operands are available at the current simulated CLOCK time. If they are, IPOINT is set to indicate the instruction, the function unit and registers are recorded in FUNPRIME, DESTPR, JPRIME, KPRIME, the delay (DELTAT) is set to zero and control is transferred to ISSUEX. If they are not available and IPOINT is not already pointing to another instruction, then IPOINT is set to point to this R-list entry, the function unit and registers are recorded, DELTAT is set to the delay before execution, and the scan of the tree is continued, looking for an MX entry. If the end of the tree is encountered before an MX entry is found, control is transferred to ISSUEI which will process an issuable instruction if IPOINT designates an MI entry or attempt corrective action (ASYNCH) if IPOINT = -1.

DOCUMENT CLASS IMS PAGE NO. 31.5
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

5.3 ISSUE

The primary function of ISSUE is the maintenance of various clocks, tables, and pointers involved in machine simulation and code selection. ISSUE also passes information to POST for the generation of COMPASS line images. ISSUE receives control from OPTB and returns to OPT.1B unless it determines that code selection for the current sequence has been completed or that no further progress can be made in code selection. In either of the latter cases, exit is made through OPTA to PROSEQ with X6 flagging success or failure.

Control is transferred to ISSUE at NOCODE, ISSUEI, and ISSUEX from OPTB and to PRUNE from OPTA.

5.3.1 Tables maintained by ISSUE:

- Register Contents (Sec. 6.1)
- Register Availability (Sec. 6.2)
- A-Register Contents (Sec. 6.3)
- Scratch Register List (Sec. 6.4)
- Function Unit Availability (Sec. 6.5)

Cells maintained by ISSUE:

- CLOCK - current simulated object machine time for this sequence.
- ISUCLOK - minimum simulated clock time for next instruction issue.
- PARCEL - currently available parcel in object machine word.
- ISUPRCL - minimum parcel for next instruction.
- NXTSTOR - minimum clock time for next store instruction.
- NXTLOAD - minimum clock time for next load instruction.
- X6AVAIL - clock time when X6 is available as a destination register.
- X7AVAIL - clock time when X7 is available as a destination register.
- NORLISR - current number of R-list entries left to process.
- THISR - R-list address of entry currently being processed.
- NOSTOR - number of store instructions remaining in the sequence.
- TPRIME - address of the first (highest priority) unissued tree entry.

5.3.2 PRUNE

PRUNE is a closed subroutine within ISSUE which removes all references to a given R-list address from the dependency tree. It is used primarily by ISSUE but also by OPTA during the processing of register defines.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 31.6
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. _____ 4P616 _____ MACHINE SERIES 64/65/6600

5.3.3 NOCODE

Entry is made at NOCODE for processing R-list entries which produce no object code. Register stores and dummy register transmits are handled by NOCODE. Appropriate register contents and availability words are updated and the tree is pruned, but no COMPASS output is generated.

5.3.4 ISSUEI and ISSUEX

If no issuable instruction can be found at the current clock time and parcel number, entry is made to ISSUEI with IPOINT negative and control is transferred immediately to ASYNCH. If a machine executable instruction has been chosen by OPTB, entry is made to ISSUEX; if a machine issuable instruction has been chosen, entry is made to ISSUEI with X5 containing the number of minor cycles until the instruction begins execution. In either case, IPOINT points to the tree entry containing the address of the R-list entry to be processed.

After appropriate initialization, the two paths merge and the following activities are performed:

1. The uses count for each operand entering into the instruction is decremented and X-registers are added to the scratch list when the uses count becomes zero.
2. PARCEL count and CLOCK are updated, taking account of word boundary crossing to reset PARCEL to zero and add WRDTIME into CLOCK.
3. The various tables and cells described in 5.3.1 are made current.
4. If the result register is an X, it is removed from the scratch list.
5. PRUNE is called to remove references to the current R-list entry from the dependency tree.
6. POST is called to generate COMPASS output.
7. Exit is made to OPTB if more work is to be done for the current sequence or to PROSEQ (after calling CLSPOST) if the last R-list entry has just been processed.

5.3.5 ASYNCH

ASYNCH attempts corrective action when OPTB cannot find an R-list entry to issue. If code selection is for the 6400, this corrective action consists of generation of a NO instruction if PARCEL = 3 or of adding X6 or X7 to scratch if possible. If ASYNCH is successful, exit

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 31.7
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

is back to OPTB; if not, exit is through OPTA to PROSEQ with X6 marking failure. If code selection is for the 6600, the processing in ASYNCH is concerned primarily with advancing CLOCK to the next point in time when a function unit or result register becomes available. At each new future time, exit is made back to OPTB for another try at code selection. If PARCEL = 3 and out-of-stack timing is being used, the effect of NO generation is investigated at the various future times. When all else fails, attempt is made to add X6 or X7 to the scratch list. If ASYNCH is unsuccessful, exit is made through OPTA to PROSEQ with X6 marking failure.

6.0 Formats

Five tables are used by OPT in connection with machine simulation.

6.1 Register Contents

Cells XOR through B7R contain, in the rightmost 18 bits, the number of the R currently residing in that register in the simulated machine. If the sign bit is set, the R has been locked into that register.

6.2 Register Availability

Cells XOS through B7S contain, in the rightmost 18 bits, the simulated clock time when the contents of the associated register are available as source input to a function unit. The upper 12 bits of each entry contains (in packed format) the number of remaining unissued instructions requiring the R found in the corresponding register contents cell.

6.3 A-Register Contents

Each pair of cells, AOCON through A7CON, contain the complete RLIST1 and RLIST2 entries used to set the corresponding A-register. These are used in the current implementation to convert 30-bit store instructions to 15-bit instructions when possible.

6.4 Scratch Register List

Cells XOSCR through X7SCR are used to keep track of the availability of the various X registers for use as destination registers. If the sign bit of a cell is set, the corresponding X register is unavailable as a destination. Otherwise the lower 18 bits give the simulated clock time at which the register can be used as a destination.

6.5 Function Unit Availability

The 12 cells starting with BRANCH and ending with INC2 contain, in the rightmost 18 bits, the simulated clock time when the function unit becomes available for issuing an instruction.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 32.1
PRODUCT NAME FORTRAN EXTENDED VERSION 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

POST

- 1.0 General
- 1.1 POST resides in PASS 2 and performs the following operations.
 - 1.1.1 Converts R-list entries into COMPASS line images.
 - 1.1.2 Inserts SUB and DELAY macro references into the COMPASS string when a location field contains a formal parameter.
 - 1.1.3 Forms traceback information for 60 bit return jumps.
 - 1.1.4 Maintains the number of SUB and DELAY references issued for each formal parameter.
 - 1.1.5 Maintains the length of generated code for both the CODE. and VARDIM. relocation blocks.
 - 1.1.6 Defines statement labels (prefixed with a decimal point) and generated labels (prefixed with equivalence symbol \equiv).
- 2.0 Entry points.
 - 2.1 OPNPOST
 - 2.1.1 Initializes the POST routine and is entered via a return jump each time a new sequence is initiated. It sets the COMPASS string buffer limits and initializes to zero the number of SUB's and DELAY's issued for each formal parameter.
 - 2.2 POST
 - 2.2.1 Entered via a return jump to translate one R-list entry into a COMPASS line image. Also uses the instruction size (15 or 30 bits) and the label field to maintain the length of the sequence and total length of code issued.
 - 2.2.2 Prior to entering POST via a return jump the following cells within POST must be set:

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 32.2
PRODUCT NAME FORTTRAN EXTENDED VERSION 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

2.2.2 (continued)

POSTIFO+0 Contains 1st R-list entry. The R-list op code must have been changed to the appropriate machine code if the R-list code has a choice of machine codes (i.e., R-list code 10 can be machine code 10 or 22).

POSTIFO+1 Contains the second R-list word (in case of type III R-list).

POSTIFO+2 Contains the descriptor for the R-list entry.

POSTIFO+3 or I These three words contain a code for the actual register type

POSTIFO+4 or J and number that is to be used. I is the destination register.

POSTIFO+5 or K J, K are the two source registers (if there are two).

The following is a list of the register codes:

X0 = 1	A0 = 9	B0 = 17
X1 = 2	A1 = 10	B1 = 18
X2 = 3	A2 = 11	B2 = 19
X3 = 4	A3 = 12	B3 = 20
X4 = 5	A4 = 13	B4 = 21
X5 = 6	A5 = 14	B5 = 22
X6 = 7	A6 = 15	B6 = 23
X7 = 8	A7 = 16	B7 = 24

2.2.3 Generally each instruction is placed in a string buffer, starting at LWAWORK and working towards FWAWORK. Each word will contain a right justified character, and the string is terminated by a zero word.

After each R-list entry is formed, the information is packed and added

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. 32.3
PRODUCT NAME _____ FORTRAN EXTENDED VERSION 2.0 _____
PRODUCT MODEL NO. 3PC08 _____ MACHINE SERIES 64/65/6600 _____

2.2.3 (continued)

to the list of line images to be transferred to the COMPASS file. Any extra information that needs to be added, such as trace back information or SUB macro references, is added at this time. If it seems as though there is insufficient room (40 words) to POST the next instruction, POST exits with X6 negative. This looking ahead is done so that OPT does not release needed flags if POST happens to fail on the last entry in a sequence.

2.3 CLSPOST

2.3.1 Entered via a return jump to transfer the COMPASS line images for a sequence to the COMPS file via WRWDS2. Also the number of SUB's and DELAY's issued for each formal parameter this sequence is added to the total for the parameter. This sum and subsum are maintained in word 2 of the 2 word symbol table entry for the parameter.

2.4 POSTIFO.

2.4.1 POSTIFO is declared an entry point so that the routine calling POST can preset the information POST needs and is not to be entered as it is a data area.

2.5 FCHAR

2.5.1 FCHAR is declared an entry point in POST so that OPT can force instructions upper as it wishes.

2.5.2 FCHAR is set by OPT and is either a blank or a plus in display code.

2.5.3 POST always sets FCHAR blank before it exits.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 32.4
PRODUCT NAME FORTRAN EXTENDED VERSION 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

2.6 PARCEL

2.6.1 Contains an instruction parcel count 0, 1, 2, 3 or 4. It is an entry point because POST will add or subtract 1 parcel when it adds or deletes an NO instruction from the code.

2.7 OVERS

2.7.1 Contains a parcel count 0, 1, 2 or 3 of the instruction word from the immediately prior sequence. This is necessary to maintain the correct length of issued code.

2.8 COVD

2.8.1 Contains the address of the location that holds the length of code issued and is either the address of CODE for the CODE. block or VARDIM for the VARDIM. block

3.0 POST produces no diagnostics. It will translate the information it is given. If the information is bad, it will go into the COMPASS file improperly and will promptly be diagnosed by the assembler.

4.0 POST is called by OPT. It expects the information described under 2.2.2 to be preset.

5.0 Structure.

5.1 Initialization

5.1.1 POST starts by converting the contents of the I, J, K cells, which OPTB has set to codes for specific registers, to display code. The initial values of these cells I, J, K are given on the preceding page. After they are converted to display code, the register type will be in the lower 30 bits of the word, the register number in the upper 30 bits.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 32.5
PRODUCT NAME FORTTRAN EXTENDED VERSION 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- 5.1.2 FCHAR is transferred to LWAWORK using X7, as the store register. A7 is therefore initialized and it is used as the store register and pointer for the strung out COMPASS line. A blank is added to the string and this will eventually appear in column 2. Thus we start with a '+b' or a "bb".
- 5.1.3 The R-list descriptor is read up to determine the type of instruction to be converted and each type is processed in the following several manners:
- 5.2 TYPE1 only a maximum of three registers. The op code is closely examined to find the proper instruction and the letters for the instruction and the register assignments are added to the string.
- 5.3 TYPE2 only a mask or set instruction needed. Which one is determined and the letters, register and constant are added to the string.
- 5.4 TYPE3 the proper letters are added according to the op code. The IH field is then processed which takes into account the necessity for issuing a SUB macro reference.
- 5.5 TYPE4 only jump can be compiled in this case. The proper letters are added and the symbol is processed.
- 5.6 PACK this is the routine that the above four routines go to when they have finished placing a card image backwards in the string. The string is packed to look like a card that comes through the card reader and thus it is terminated either be a zero word or by the last word having 12 bits of zero in the lower 12 bits.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 32.6
PRODUCT NAME FORTRAN EXTENDED VERSION 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

After the instruction has been added to the temporary COMPASS file, the substitution macro reference or delay substitution macro reference, is placed in the COMPASS file if necessary. The substitution macro is referenced any time a 30 bit instruction has a formal parameter as part of its address field. The delay macro is referenced only when the same formal parameter is used in both the upper and lower parts of the same word. Traceback information is also output at this time if a 60 bit return jump was just processed. POST then exits with X6 positive unless there are not 40 more words of working storage left (the maximum amount of storage POST could need to process one R-list entry.) If not enough room is left, X6 is negative.

6.0 FORMATS

6.1 The code at PACK8 may be changed during execution to facilitate the necessity of putting out a delay substitution macro reference for the following situation:

Code desired:	SA1	FP1
	SA2	FP2

where FP1, FP2 are formal parameters.

Due to the way the address substitution takes place, the following COMPASS output must be produced:

SA1	FP1
SUB	FP1
SA2	FP1+1
DELAY	FP1
SUB	FP1,1

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 32.7
PRODUCT NAME FORTRAN EXTENDED VERSION 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- 6.2 APCA contains the CA field of the instruction presently being sent to COMPASS
- 6.3 APLST contains zero or the name of the formal parameter just formed in a K field.
- 6.4 APLAST contains the name of formal parameter last used in a K field.
- 6.5 CALAST contains the last CA field put into a sub macro reference.
- 6.6 CMPSPR contains the starting address of the line images for the COMPASS file (OPNPOST sets it to FWAWORK).
- 6.7 CMPSPS contains the address in which COMPASS line images stored.
- 6.8 TRACEP set non-zero on a 60 bit return jump. Indicates traceback information should be added. This is done after the RJ instruction has been converted to a COMPASS line image.
- 6.9 COMPASS line string each line for COMPASS is strung out, one character per central memory word, backwards starting at LWAWORK. When the entire instruction has been translated it is packed 10 characters per word and added to the temporary COMPASS file.
- 6.10 Temporary COMPASS file starts at FWAWORK and grows forward toward LWAWORK. Contains COMPASS card images for all instructions translated between a call to OPNPOST and one to CLSPOST.
- 6.11 Sequence SUB and DELAY count is held in bits 41-52 of word 2 of the 2 word symbol table. The total SUB and DELAY count is held in bits 19-36 of the word 2 symbol table entry for the formal parameter.

DOCUMENT CLASS IMS. PAGE NO. 33.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

MACROE

- 1.0 MACROE expands a macro reference into proper R-LIST information.
- 1.1 MACROE is part of Pass 2. It processes one macro reference each time it is called.
- 2.0 MACROE has only one entry point.
- 2.1 MACROE
 - 2.1.1 One macro reference will be expanded into R-LIST.
 - 2.1.2 Calling sequence:
 - SAI "first word address of the macro reference".MACROE is called by PRE. Upon returning to the caller, the macro will have been expanded starting at MACBUF the external location and the number of R-list entries the macro expanded into will be found in MACWRDS. Both MACBUF and MACWRDS are contained in READRL.
 - 2.1.3 The macro parameters are strung out one per central memory word starting at PARAN which is an entry point to MACROX. The macro descriptor is picked up and the macro text is transferred to the macro expansion area. All fields of the text that have to be modified are processed during this transfer. When the complete text has been transferred, MACROE will exit.
- 3.0 No diagnostics are produced.
- 4.0 The following cells are referenced and expected to be set accordingly.
 - 4.0.1 RNAME - (RA+64B) (same as NRLN) contains the next R register to use.
 - 4.0.2 DESCR - external to MACROE start of the descriptors. This address is used as the starting point to index into the descriptors.
 - 4.0.4 PARAN - external to MACROE - is the starting address of storage used for expanding the calling parameters for each macro reference. Starting at

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 33.2
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

PARAN (in MACROX) there is sufficient room to expand the maximum calling sequence.

4.0.5 MACORG - external to MACROE that is set by an EQU in MACROX. It is set to the table bias for the macro numbers.

4.0.6 DOMACK - the bias added to the macro descriptor to find the D0 macros.

4.0.7 MACBUF - is the starting address for storing the expanded macro.

4.1 MACROE will have set the following cells upon exit:

4.1.1 RNAME- will be updated to reflect any register numbers (R's) generated in the macro expansion.

4.1.2 MACWRDS - will contain the number of words in the macro expansion.

4.1.3 MACBUF - external to MACROE. First word address of the area in which the macro is expanded.

4.1.4 MACWRDS - set to the number of words in the macro expansion.

4.1.5 MACNXT - set to MACBUF if there were no RLIST entries to expand.

5.0 Structure - The expansion can be broken into three distinct portions.

5.1 Pick up Macro descriptor

5.1.1 The macro number is adjusted by the table bias (MACORG) and the descriptor is picked by indexing into the macro descriptor table which starts at MACDESC. This descriptor contains information to do the following:

5.1.2 The initial value of the register number (R's) to be used for this macro expansion is saved and then updated to reflect the number of registers to be generated in this expansion.

5.1.3 The macro length is extracted and placed in MACWRDS.

5.1.4 The descriptor for each macro contains the number of registers passed as formal parameters. The number of symbolic fields (IH) and the number

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS. PAGE NO. 33.3
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

of constant fields (CA) passed as formal parameters. These fields are extracted from the descriptor and saved.

- 5.1.5 A read register is initialized to the first word of the actual macro text.
- 5.2 Expand the actual macro parameters.
 - 5.2.1 The number of R's, IH's and CA's have been saved from the macro descriptor. They are now extracted from the macro referenced area and expanded starting at location PARAN. They are unpacked so that each parameter occupies one word.
 - 5.2.2 The addresses of the start of each field are kept in B registers. This allows indexing into the list when an actual parameter substitution is required.
- 5.3 EXPAND the macro text.
 - 5.3.1 One word is read from the text. The OC is extracted and used to index into the descriptors to determine the type of R-list desired.
 - 5.3.2 TYPE1 - The three R fields are extracted, examined to determine what, if anything, should be substituted for them. They are then reformed and added to the expansion area. In all cases R fields may be replaced with a parameter R, a generated R, or they may be left alone (in case of A0 or B0).
 - 5.3.3 TYPE2 - the RI field and CA field are examined to determine if any change is necessary. The lower 13 bits of the 14 bit S0 field are saved. (The upper bit is an indication as to whether or not the CA field is an actual parameter). The R-LIST entry is combined and added to the macro expansion area.
 - 5.3.4 TYPE3 -

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS. _____ PAGE NO. _____ 33.4
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. _____ 4P616 _____ MACHINE SERIES 64/65/6600 _____

- 5.3.4.1 First word has a replaceable RI and CA field. The lower 13 bits of the SO field are saved. The RI and CA fields are replaced if necessary. The entry is reformed and added to the macro expansion.
- 5.3.4.2 The second word is read up. It has a replaceable IH and R field. The H2 field, if any, is destroyed. The fields are processed, the entry recombined, and added to the expansion.
- 5.4 This continues at 5.3 until all the text has been processed.
- 6.0 FORMATS -
- 6.1 No flags or tables are held or built by MACROE.
- 6.2 The expanded macro is in the form of regular R-LIST as described in the R-LIST write-up. The formats of macro descriptor and macro text are given there also.
- 7.0 No modification facilities.
- 8.0 Method:
The macro is expanded one word at a time, replacing actual fields as necessary.
- 9.0 The coding in MACROE has been reordered several times to improve its timing. Any modification should be done with care.

DOCUMENT CLASS IMS PAGE NO. 34.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

BUILDDT

1.0 General Information

1.1 BUILDDT builds a dependency tree from a specified sequence in the R-list. One should be familiar with the information in Chapter 26 before reading this document. The tree will reflect the following:

1.1.1 The requirements of an R-list entry for results produced by another R-list entry.

1.1.2 The requirement that store operations take place within the most immediate surrounding jumps, (stores must take place prior to the following jump and after the preceeding jump).

1.1.3 The requirement that all references to identical IH fields are in the same relative order as originally in the R-list. (This is to make sure A (5) is stored into last in the following example)

DO 10, I = 1, 10

A (I) = expression

10 A (5) = expression

1.1.4 The requirement that the updating of DO loop index functions takes place after all of their uses within the DO loop.

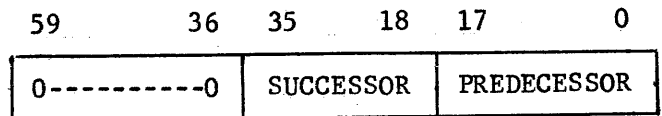
1.2 The following conditions will prevail when BUILDDT exits:

- a) If there was insufficient working storage for the tree to be built X6 will be negative.
- b) Otherwise X6 will be positive and the entry point TREELNG contains the length of the tree, the entry point FWATREE

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 34.2
 PRODUCT NAME FORTRAN Extended
 PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

contains the start of the tree, FFAWORK has been updated to save the tree, the descriptor for each R-list entry include the number of times the entry is used as a predecessor. (See section 27 for a more detailed explanation of an R-list descriptor). Each entry in the tree occupies one computer word in the following format.



The predecessor and successor fields are each 18 bits long and contain the address of the operation within the R-list. The tree is ordered such that the entry with the highest priority appears first.

- 2.0 Usage
- 2.1 Entry Point Name: BUILDDT
 - 2.1.1 BUILDDT is entered with a sequence in the R-list specified for which a tree is to be built. The start of the R-list (FWARLIS), the number of entries in the R-list sequence (NORLIST), and the available core limits FFAWORK, LFAWORK are all accessed.
 - 2.1.2 BUILDDT is entered via a return jump. It exits through its entry point with X6 negative if there is insufficient room; otherwise, X6 is positive and the conditions listed under 1.2 above prevail.
 - 2.1.3 BUILDDT is divided into three parts; the building of the tree, the calculation of priorities, and the sorting of the tree. Initially, a temporary tree is built starting at LFAWORK and extending toward FFAWORK. The tree is then sorted such that all

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 34.3
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

equal predecessors are together and in ascending order. The priorities are then calculated. A priority sort is then performed and the tree is moved to start at FFAWORK and grow toward LFAWORK.

3.0 No diagnostics are produced.

4.0 Environment: The following information must be provided in the following cells, all external to BUILDDT.

4.1.1 FFAWORK, LFAWORK must contain the available working storage limits.

4.1.2 FFAWLIS must contain the first work address of the sequence.

4.1.3 NORLIST must contain the number of R-list entries in the sequence.

4.1.4 It is expected that the sequence will have been transformed into three sections that lie in series. The first section contains the first word of each R-list entry, the second section contains the second word of each R-list entry, (zero for R-list types of one word length) and third section contain the descriptor for each R-list entry. The routine COPY placed the tree in this format.

5.0 Structure

5.1 First Section - Forming the Tree

5.1.1 After initialization, one pass is made through the R-list to link any store instructions to surrounding jump instructions. The first store instructions are linked to the end of the sequence with an instruction time of 10 (time for a store) and a total time of minus one. (This total time and instruction time are used in the priority calculation.) As each store instruction is linked,

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 34.4
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

its address is kept in another list. When/if a jump instruction is found, all of these store instructions are made successors of the jump instruction. After this list of store instructions is depleted in this manner, this search is continued. Store instructions are now made predecessors to this new jump instruction and will be made successors to a preceding jump instruction if there is one. This search terminates upon detecting the start of the sequence. During this pass through the R-list, the uses of B1 are noted and made predecessors to the initialization of B1 for DO loop usage (if there is such an initialization).

5.1.2 Now the tree entries for register dependency and for DO loop functions are made. At the same time we do this, information is saved to enable us to later accomplish the sequential linking of IH fields. Since, in a well behaved DO loop, addresses can be accessed through B registers, short loads and stores are also entered into this list of IH fields. An R-list entry is read and unpacked. If it has a negative exponent, this indicates a DO loop function is being updated and linkage to this functions use is required. The RI field is extracted and compared to all prior RJ, RF, and RK fields in the sequence. Any that are equal to this RI result in a tree entry. This is done in a routine called SUBLINK. The instruction is repacked, with a positive exponent, and stored back into the R-list 1 section. We then determine the type of R-list entry, extract the appropriate registers and search the rest of the R-list for the definition of these registers. If they are not defined in the sequence they

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 34.5
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

are linked to the start of the sequence. If they are defined, the instruction for the definition is extracted from the descriptor and included in the temporary tree. The predecessor count in the descriptor is updated by 1 only when its resultant register is needed in another operation. This continues until the whole R-list has been examined and proper tree entries made for it.

5.1.3 At this time the list containing the IH field information is processed and tree entries made, if necessary. The list is searched for a usable entry. When one is found, its IH field is extracted and the rest of the list is searched to find if there is an entry specifying that a store was made into this IH group. If a store was made, the entries using this IH field are sequentially linked so that these operations take place in the same order in which they were specified in R-list.

5.2 Second Section - Priority Calculation

5.2.1 The tree is sorted so all entries with the same predecessor are together and the list is in ascending order. This is done to reduce the number of memory accesses to accomplish the building of priorities.

5.2.2 The building of priorities starts with entries linked to the end of the sequence. The priorities are built by working backwards in the tree and calculating the maximum amount of time it will take to reach each point in the tree. When a node is found in the tree for which the time to reach has not been determined for all paths to the point, one of these paths is chosen and a

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 34.6
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

forward search is made along this new path until a node in this path is found for which the time has been determined. The priority building starts again, backwards from this point. The priority field, which is in the upper 13 bits of each tree entry is set negative as each path to the point is analyzed and set positive when all paths to that point have been analyzed. When there are no entries other than those linked to the start of the sequence with negative time, the priorities have been calculated.

5.3 Sorting of the tree - two parts.

5.3.1 A simple sort of the tree is performed by comparing two entries and switching them if the second has a higher priority than the first one. Each time a switch is performed, a flag is set and after each pass through the tree, if the flag is set, another pass is made. This method was employed because the tree is fairly well sorted from the start as a result of the order the tree entries are initially made.

5.3.2 Any entries with a priority of zero (dangling code) are eliminated from the tree at this time and the uses count (see description of descriptor, section 27) of its predecessor is decreased by 1.

5.3.3 The tree is then transferred so it starts at FVAWORK and grows toward LVAWORK. During this transfer, all information but the predecessor and successor is extracted from each entry. When entries with equal priorities are encountered, the section of equal priorities is further examined and reordered so that the entries with the greater number of predecessors appear first.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 34.7
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

6.0 Formats

6.1 Tree

TT	IT	0	SUCCESSOR	PREDECESSOR
13	5	6	18	18

TT - total time to get to this successor in tree

IT - time to get from this predecessor to this successor.

DOCUMENT CLASS IMS PAGE NO. 35.1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

PS2CTL

1.0 General

PS2CTL is the first routine entered in Pass 2. It places all Pass 1 diagnostic messages into the output file before calling PRE. The Pass 2 I/O routines are entry points PS2CTL. The display code diagnostic messages are also contained in PS2CTL.

2.0 Entry Point Names

2.1 ERROR2. The first entered entry point of Pass 2. It checks for errors and lists diagnostic messages. If fatal to compilation or execution errors occurred, Pass 2 is not executed but control is returned to LDCOM1 in CLOSE2 to reload and execute Pass 1.

2.2 PS2CTL. PS2CTL is not used as an entry point as such but simply to denote the start (low memory address) of working storage for Pass 2.

2.2 RDWDS. The routine for reading R-list from the disk. The calling sequence is:

SB6 "file number"
SB7 "fwa" of central memory to read in to
SB1 "number of words to transfer"
RJ RDWDS

2.3 WRWDS2. The routine used for placing line images into the COMPASS file. The calling sequence is:

SB6 "file number"
SB7 "fwa" of central memory to transfer from
SB1 "number of words to transfer"
RJ WRWDS 2

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. 35.2
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

3.0 PS2CTL detects no error conditions.

4.0 Environment

PS2CTL looks at or initializes the following named location of the low core communications area: SYMEND, PTEMP, APLAST, VAR1, PROGRAM, VARLAST and NRLN.

ERTAB contains the error code number and the line number of the error as set by ERPRO.

5.0 Structure

5.1 RDWDS first checks to see if the given buffer is empty and if so a call is made to CIO to fill it. MVWRDS is called to actually transfer the given number of words from the buffer to the CM location. The IN and OUT locations and NUMBER of words out of the buffer are maintained.

5.2 MVWRDS moves a given number of words from a buffer area to the requested central memory location.

5.3 WRWDS2 essentially performs the write function, maintaining buffer pointers and calling MVWRDS to actually perform the move from a CM location to an output buffer.

5.4 ERROR2 is the primary entry point; issues diagnostic messages to the list file, then either calls PRE to continue Pass 2 or LDCOM1 to process the next program, if any.

6.0 Formats

The error table format is described in ERPRO, Section 7, as is the format of diagnostic messages.

DOCUMENT CLASS IMS PAGE NO. 36.1
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

FORTRAN EXTENDED ASSEMBLER

1.0 GENERAL INFORMATION

1.1 Task Description

The FORTRAN Extended assembler FTNXAS replaces COMPASS as the assembly pass of FORTRAN Extended Version 2.0. It is a one pass assembler designed specifically to increase compilation speed. It accepts a formatted subset of the COMPASS assembly language and produces binary relocatable subprograms. All information required to facilitate a one pass assembly is gathered during the previous two passes of the compiler.

2.0 USAGE

2.1 Entry Points

2.1.1 FTNXAS

The assembler is entered either by a direct jump if the assembler is a separate overlay or by a return jump from CLOSE2.

3.0 DIAGNOSTICS

3.1 Fatal to Execution

3.1.1 ILL

The word ILL appearing to the left of the source line on the assembly listing means the assembler could not recognize the statement or encountered an ERR pseudo-op. When this occurs assembly is abandoned, the fatal to execution flag is set, the source line is printed regardless of the '00' option, and assembly proceeds with the next statement. The fatal to execution flag causes the message 'FTNX ERRORS' to be written on LG0 in place of the normal relocatable subprogram. Note that since the code produced by the first two passes is assumed to be correct, minimum error checking is designed into the assembler. The usual response to an error in the source string will be this diagnostic. Abnormal termination of the job may occur during compilation because various unused jump vectors in the assembler are used for storage or unrelated code.

3.1.2 SYMBOL ERR

This message appears to the left of the source line if the assembler was unable to find a symbol in the two word symbol table. This error causes the same action as described for ILL.

DOCUMENT CLASS IMS PAGE NO. 36.2
 PRODUCT NAME FORTTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PCDB MACHINE SERIES 64/65/6600

3.1.3 STORAGE OVERFLOW

This message is printed on a separate line when the assembler has expended all available working storage for chained common and external reference information. Assembly will proceed to count the increase in field length required, but no relocatable binary deck will be produced. The message 'INCREASE FIELD LENGTH BY XXXXXX' is printed after the END statement of the subprogram.

3.2 Informative

None.

4.0 ENVIRONMENT

The assembler may reside after CLOSE2 in the {1,2} overlay, as a separate overlay, or with Pass 1 and 2 of the compiler in a non-overlay configuration.

4.1 Input String

The input string consists of COMPASS language card images constructed using the following rules.

Executable Instructions

1. The label field must be blank except the forcing characters + {upper} and - {lower} may be used in column 1.
2. The operation code must begin in column 3.
3. There must be only one space between the operation and address fields.
4. Following is a list of executable instruction mnemonics and address fields that FTNXAS can recognize, followed by the binary produced.

<u>Opcode</u>	<u>Address</u>	<u>Binary in Octal</u>	<u>Other Action</u>
RJ	Symbol	0100000000	External relocation noted; force upper next instruction.
JP	B1, Symbol	0210XXXXXX	Program relocation noted; force upper next instruction.
JP	B1	0210000000	

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 36.3
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

<u>Opcode</u>	<u>Address</u>	<u>Binary in Octal</u>	<u>Other Action</u>
ZR	X_j , Symbol	030 _j XXXXXX	Program relocation noted.
NZ	X_j , Symbol	031 _j XXXXXX	Program relocation noted.
PL	X_j , Symbol	032 _j XXXXXX	Program relocation noted.
NG	X_j , Symbol	033 _j XXXXXX	Program relocation noted.
EQ	Symbol	0400XXXXXX	Program relocation noted; force upper next instruction.
EQ	B_i , B_j , Symbol	04 _i _j XXXXXX	Program relocation noted.
NE	B_i , B_j , Symbol	05 _i _j XXXXXX	Program relocation noted.
GE	B_i , B_j , Symbol	06 _i _j XXXXXX	Program relocation noted.
LT	B_i , B_j , Symbol	07 _i _j XXXXXX	Program relocation noted.
BX _i	X_j	10 _i _j	
BX _i	$X_j + X_k$	11 _i _{jk}	
BX _i	$X_j - X_k$	12 _i _{jk}	
BX _i	$X_j \times X_k$	13 _i _{jk}	
BX _i	$-X_k$	14 _i _{kk}	
BX _i	$-X_k + X_j$	15 _i _{jk}	
BX _i	$-X_k - X_j$	16 _i _{jk}	
BX _i	$-X_k \times X_j$	17 _i _{jk}	
LX _i	jk^B	20 _i _{jk}	
LX _i	k^B	20 _i _{0k}	
AX _i	jk^B	21 _i _{jk}	
AX _i	k^B	21 _i _{0k}	
LX _i	B_j, X_k	22 _i _{jk}	
AX _i	B_j, X_k	23 _i _{jk}	

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 36.4
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

<u>Opcode</u>	<u>Address</u>	<u>Binary in Octal</u>	<u>Other Action</u>
NX _i	B _j X _k	24 _{ijk}	
UX _i	B _j X _k	26 _{ijk}	
PX _i	B _j X _k	27 _{ijk}	
FX _i	X _j +X _k	30 _{ijk}	
FX _i	X _j -X _k	31 _{ijk}	
DX _i	X _i +X _j	32 _{ijk}	
DX _i	X _i -X _j	33 _{ijk}	
IX _i	X _j +X _k	36 _{ijk}	
IX _i	X _j -X _k	37 _{ijk}	
FX _i	X _j *X _k	40 _{ijk}	
DX _i	X _j *X _k	42 _{ijk}	
MX _i	jkB	43 _{ijk}	
MX _i	kB	43 _{i0k}	
FX _i	X _j /X _k	44 _{ijk}	
NO		46000	
S _{ri}	A _j +A.E.	g0 _{ij} XXXXXX	Relocation noted.
S _{ri}	B _j +A.E.	g1 _{ij} XXXXXX	Relocation noted.
S _{ri}	A.E.	g1 _i 0XXXXXX	Relocation noted.
S _{ri}	X _j +A.E.	g2 _{ij} XXXXXX	Relocation noted.
S _{ri}	X _j +B _k	g3 _{ijk}	
S _{ri}	X _j	g3 _{ij} 0	
S _{ri}	A _j +B _k	g4 _{ijk}	
S _{ri}	A _j	g4 _{ij} 0	
S _{ri}	A _j -B _k	g5 _{ijk}	

DOCUMENT CLASS IMS PAGE NO. 36.5
 PRODUCT NAME FORTTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

<u>Opcode</u>	<u>Address</u>	<u>Binary in Octal</u>	<u>Other Action</u>
S_{ri}	$B_j + B_k$	g^b_{ijk}	
S_{ri}	B_j	g^b_{ij0}	
S_{ri}	$B_j - B_k$	g^7_{ijk}	

where:

- 1} If $r=A$ then $g=5$, if $r=B$ then $g=6$, if $r=X$ then $g=7$.
- 2} A.E. is an address expression consisting of octal constants and/or symbols in the same relocation base separated by plus {+} or minus {-} symbols.

Pseudo-Ops

1. Any pseudo-op may be labeled.
2. Pseudo-ops are free field except labels, if present, must begin in column 1.
3. Following is a list of permissible pseudo-ops, their forms, and the results.

<u>Label</u>	<u>Opcode</u>	<u>Address</u>	<u>Action</u>
Required	BSS	Octal constant	a} force upper. b} define the label. c} increment the origin counter by the value of the octal constant.
Optional	DATA	Octal constant	a} Convert the octal constant to binary and write it out. b} Define the label, if present.
Optional	DIS	n, character string	a} Define the label, if present. b} Write n words with blank fill.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 36.6
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

<u>Label</u>	<u>Opcode</u>	<u>Address</u>	<u>Action</u>
Optional	EQU	Anything	a} Ignored.
None	ORG	Address expression	a} The origin counter is reset to the value indicated by the address expression.
Required	SET	*	a} The label is defined to have the current value of the origin counter.
None	REPI	S/Symbol, B/Octal constant, C/octal constant	a} A REPI table is produced.
None	USE	name or /name/	a} Change the origin counter to that of the block indicated.
Optional	VFD	1} n/octal constant. 2} n/address expression, $18 \leq n \leq 60$. 3} {nM6}/{M}{C,H,L or R}, Character string, $1 \leq n \leq 10, 0 \leq M \leq 10$. 4} Any combination of the above separated by commas. 5} The total bits specified must be 15, 30 or 60. 6} Any relocatable quantity must be in the lower 18 bits of the generated field.	a} The specified data is converted to binary and written. Relocation is noted.
Optional	MACRO	Anything	a} Causes the assembler to ignore all lines following until the characters 'ENDM' are encountered beginning in column 3.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 36.7
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

<u>Label</u>	<u>Opcode</u>	<u>Address</u>	<u>Action</u>
NOTE - FTNXAS ignores macro descriptions; the macros that it expands are treated as pseudo-ops.			
None	ENDM	{Cols. 3-6}	a} Terminate skipping of a macro prototype and return to normal assembly mode.
	END	{must start in Cols. 3-8} Symbol or blank	a} Terminate assembly and return.
	IDENT	Anything	a} Causes assembler initialization to take place and assembly to begin.
	LIST	Anything	a} Ignored.

Macro Calls

- Macro calls for SUB, DELAY, FILE, and ENTR must have exactly one space between the call and the parameters. NAME and ADDSUB are free field.
- Below is a list of macro names the assembler will recognize. See Section 5 of this document for the macro prototype description and the generated code.

<u>Label</u>	<u>Macro Name</u>	<u>Parameters</u>	<u>Comments</u>
None	ADDSUB	FP	
None	SUB	FP, CON	
None	DELAY	FP	
None	FILE	LN, NAME	
None	FORPAR	X	FORPAR calls are ignored by FTNXAS.
None	NAME	N, T, BASE, BIAS, FP, D1, D2, D3	
None	ENTR	NAME	

DOCUMENT CLASS IMS PAGE NO. 36.8
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

4.2 Two Word Symbol Table

Active statement labels, entry points, and the labels DO., ST., and OT. are defined by a block relative address and a relocation base indicator. The length of the relocation base associated with each formal parameter is also held in this table. Other entries are marked as undefined and their addresses are determined during assembly.

4.3 Actual Parameter, Variable Dimension and Generated Label Tables

Three tables contain respectively actual parameter, variable dimension, and generated label definitions relative to the CODE. relocation base.

4.4 ORGTAB

A table of one word entries which is used to pass to the assembler the names and lengths of the common relocation bases and the lengths of the local relocation bases, START., ENTRY., VARDIM., CODE., DATA., DATA., and HOL..

4.5 APTAB, VDTAB, GLTAB

Set in ORGTAB to the first word address of the AP, VD, and GL tables.

4.6 Object Listing Option

Bit zero of location RA+33B is set to 1 to indicate the '0' option has been selected.

4.7 SYM1A, SYMENDA

RA+46B and RA+47B point to the first and last entries of the two word symbol table respectively.

4.8 RA+4

Contains in the upper 42 bits the name of the binary output file and in the lower 18 bits a pointer to the FET for the file.

4.9 ILLFLAG

Set to non-zero if a fatal to execution or compilation error has occurred before entry to the assembler.

DOCUMENT CLASS IMS PAGE NO. 36.9
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.0 MAJOR SUBROUTINES AND LOGICAL SECTIONS

5.0.1 Summary

1. PIDENT - initializes the assembler for each program.
2. BUILDOT - creates a 22 word ORGTAB entry.
3. MR.CLEAN - removes blank fill from a symbolic name.
4. INITL{NOBINP0} - moves the next source image to ILINE from the COMPS buffer. Controls reading the COMPS file.
5. START - controls analysis of the label field of each source statement.
6. COLVEC - determines the contents of the label field.
7. BLANK - processes a blank label field.
8. PFC - processes forcing characters + and - in the label field.
9. PLABEL - processes a symbolic name in the label field.
10. OCSKAN - controls opcode field analysis.
11. Opcode transfer vector - decodes the contents of the opcode field.
12. L3.RJ - produces a binary instruction for RJ.
13. L3.JP - produces a binary instruction for JP.
14. L3.XJP - produces binary instructions for ZR, NZ, PL and NG.
15. L3.EQ - produces a binary instruction for EQ.
16. L3.BJP - produces binary instructions for NE, GE, and LT.
17. L3.B00L - produces binary instructions for all BX_i.
18. L3.MX, L3.SH - produces binary instructions for shift unit instructions MX_i, LX_i, and AX_i.
19. L3.PUN - produces binary instructions for NX_i, UX_i, and PX_i.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 36.10
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

20. L3.ARIT - produces binary instructions for FX_i, DX_i, IX_i.
21. L3.SET - produces binary instructions for the increment unit instruction SA_i, SB_i, SX_i.
22. L3.VFD - translates the address field of VFD pseudo-op to binary.
23. PDIS - processes the DIS pseudo-op.
24. PDATA - converts the address field of the DATA pseudo-op to binary.
25. PBSS - processes the BSS pseudo-op.
26. PUSE - processes the USE instruction.
27. USENXT - changes relocation bases.
28. USESTAR - returns to the previous relocation base.
29. PSET - processes the SET pseudo-op.
30. L3.ORG - processes the ORG pseudo-op.
31. PMACR0 - processes the MACR0 pseudo-op.
32. PREPI - processes the REPI pseudo-op.
33. PADDSUB - processes and ADDSUB macro call.
34. PDELAY - processes a DELAY macro call.
35. PSUB - processes a SUB macro call.
36. PFILE - processes a FILE macro call.
37. PNAME - processes a NAME macro call.
38. PENTR. - processes an ENTR macro call.
39. EVAL - evaluates an address expression.
40. REF - obtains the value of a symbolic name.
41. CONVERT - converts display coded octal constants to binary.
42. PACKID - separates a symbolic name from the input string.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 36.11
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- 43. WRSEQ - writes a sequence of words on the LG0 file.
- 44. WRTEXT - maintains ORGC and POSC, creates and writes TEXT tables for the loader.
- 45. FOTEXT - forces out the current text table.
- 46. WRLIST - prints each source line, generated binary instruction and the ORGC.
- 47. L4.CKL,DEF - defines statement labels.
- 48. L4.CKRB - does address relocation bookkeeping.
- 49. ILL,SILL,STOVER - processes error conditions.
- 50. PEND - terminates the assembly process.

5.0.2 Subroutine Interconnection Matrix

See attached pages.

5.1 PIDENT

5.1.1 This routine initializes the assembler. It is entered from the opcode vectors when the IDENT pseudo-op is encountered. The following tasks are performed:

1. The pointers to the first and last entries of the two word symbol table, SYM1 {RA+12B} and SYMEND {RA+13B}, are reset from SYM1A {RA+46B} and SYMENDA {RA+47} respectively.
2. MEMEND, the upper limit of the assembler working storage, is set to the contents of VDTAB-1.
3. Three switches, LSTSW1, LSTSW2 and LSTSW3, which are entry points in LSTPROC, are reset to the values KLS1A, KLS2A and KLS3A, also entry points in LSTPROC. This change is necessary because the names of basic external functions {SIN, AL0G, etc.} are initially entered in the two word symbol table without a trailing special character {., or #}, but they appear in the COMPASS line image with this character suffixed. Resetting the three switches causes LSTPROC to delete the trailing non-blank character for symbols of fewer than seven characters or to exclusive-or the last character with 06 for seven character names and search for the resulting name.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 36.12
PRODUCT NAME FORTAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

4. Allocate an LGO Buffer from MEMSTRT to MEMSTRT+LGOSIZE+1, set FIRST, IN, OUT to MEMSTRT and LIMIT to MEMSTRT+LGOSIZE+1 in the LGO FET, and reset MEMSTRT to MEMSTRT+LGOSIZE+2.
5. Set LCNT, and entry point in LIST, to zero to cause a page eject and set LINE to LINE+4 to blanks.
6. If LCC cards are present place each on the file LGO file with an end of record write.
7. Print the IDENT and LIST cards and reset COMPS OUT to the next card image.
8. If the object listing option '00' has been specified, set the switches at NOBINPO and WRTSW with return jumps to WRLIST.
9. Put a 15 word prefix table with the program name into the LGO buffer.
10. Put the ID word and program name word for the PIDL table in the LGO buffer.
11. Scan the COMMON portion of ORGTAB until a zero word is encountered moving the COMMON block names and lengths to the PIDL table and using the subroutine BUILDOT make a 22 word CORGTAB entry for each. The 22 word CORGTAB is built upwards in memory from MEMSTRT and is terminated by a zero word.
12. LORGTAB entries of 22 words each are built for the seven local relocation bases that are always present using the subroutine BUILDOT. The lengths of the bases are stored in the locations STARTA, VARDIMA, ..., HOLA which are entry points in ORGTAB. The names of the relocation bases are taken from the assembler table LOCNAM.
13. If the current subprogram is a subroutine or function and there are formal parameters there must be a 22 word LORGTAB entry setup and initialized for each. The names and lengths come from the two word symbol table. The formal parameters are contiguous from ordinal two each having the FP bit set to one. The length of the relocation base is placed by Pass 2 in the RA field and is changed to a program relative address at this time while the RB are set beginning at seven in increments of one and RL is set to 1.
14. The LORGTAB is terminated with a zero word and the program length which was accumulated during LORGTAB construction is printed then stored in the PIDL table. MEMSTRT is adjusted to the next location after the terminating zero word.

DOCUMENT CLASS IMS PAGE NO. 36.13
 PRODUCT NAME FORTTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

15. If the program name is an entry point determine its address by the following rules:

Main Program: zero in CODE.
 Subroutine: no formal parameters; one in START.
 Subroutine: with formal parameters; two in START.
 Function: two in START.

and start an ENTRY table after the PIDL in the LGO buffer with the program name as the first item.

16. Scan the two word symbol table beginning below the formal parameters.
- a. Symbols with bit 36 of word B equal to zero are given a program relative address, i.e., the first word address of the relocation base indicated by the RB field is added to the RA field and RL is set to one.
 - b. Symbols with the ENT. bit equal to one are put into the ENTRY table along with their calculated program relative addresses.
 - c. External names are moved to the LINK foundation table being built upwards from MEMSTRT. The contents of bits 41-43 of word B are exclusive-ored to the first trailing blank in the name before it is stored in the LINK table.
17. When the symbol table scan has been completed, the word count for the ENTRY table is placed in word one, the LINK foundation is terminated by a zero word and MEMSTRT is reset to the next available location.
18. The next step is to print the ENTRY and LINK tables and to remove the blank fill from the names. The tables appear below the IDENT and LIST pseudo-ops regardless of the '0' option. Each table is prefaced by a label indicating its type.
19. The actual parameter, variable dimension and generated label definition tables are scanned. The tables lie immediately below the two word symbol table and are terminated by a zero word. Each entry contains in the lower 18 bits definitions of the [AP,]VD and =GL labels relative to the CODE. relocation base. Any or all of these tables may be empty. The base address of CODE. is added to make the definitions program relative. The entries reformatted to look like word B of the two word symbol table.

DOCUMENT CLASS IMS PAGE NO. 36.14
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

20. IOPARM, an entry point in WRWDS2, is set to 16B to cause assembly output to be in binary mode.
21. FREEMEM, the pointer to the next free location of working storage, is set to the contents of MEMSTRT. RBTEMP, used for relocation calculation, is initialized to zero.

5.2 BUILD0T

- 5.2.1 This subroutine is used during assembler initialization to allocate and initialize a 22 word ORGTAB entry corresponding to each relocation base in the program.

The calling sequence is:

B2 = origin counter {base address} for this block. This will be zero for common blocks and the sum of the lengths of previous local blocks for program relocation base.

B3 = Relocation base code. This is the LCT ordinal passed out to LG0 and used by the loader. It starts at three for common blocks and for local blocks is always one.

B6 = First word address of the 22 word entry.

X1 = The relocation base name in bits 18-59 with blank fill. Bits 0-17 must be zero.

X2 = Block length, a running sum of block lengths is maintained in B2 to provide the current origin counter for local blocks and the program length at the end of ORGTAB initialization.

5.3 INITL {NOBINP0}

- 5.3.1 INITL with its alternate entry point NOBINP0 is the first routine in the main assembly loop. Its function is to read the next source image into ILINE from the COMPS file. While doing this INITL also manages the COMPS FET and issues read requests whenever there is room for one PRU in the buffer. The number of words moved to ILINE is stored in location SWC. The alternate entry NOBINP0 is used whenever the last line processed did not produce any binary output. It is one word located at INITL-1 and filled with NOPs unless the '00' option is specified. In this case it is plugged with a call to WRLIST during initialization. INITL exits to START to begin processing the line.

5.4 START

DOCUMENT CLASS IMS PAGE NO. 36.15
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.4.1 This routine sets the following registers for the character pickup macros GCH and CWD:

B6 = 6
 B7 = 54
 A5 = ILINE
 X5 = {ILINE}
 X0 = 54 bit mask; left justified.

START exits by jumping into COLIVEC using the first character of the line as an index.

5.5 BLANK, DCSCAN, PFC, PLABEL, PNL, OPCODE VECTORS

5.5.1 COLIVEC: A jump vector used to determine the contents of the label field. It partitions the first 50 display characters into the following sets and branches to the indicated routines for further processing.

{A, B, C, ..., Z,], ., =, [,]} branch to PLABEL

{+, -} branch to PFC

{blank} branch to BLANK

{Zero Byte, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, !, @, =, comma} branch to ILL

5.5.2 BLANK

FFLAG is set to NFLAG to cause a force upper if the last instruction was an unconditional jump. BLANK exits to DCSCAN.

5.5.3 PFC

FFLAG is set to +1 or -1 for plus or minus in column one respectively. This indicates force upper or force lower to WRTEXT. PFC skips to the opcode field beginning in column three, places the first 2 characters in B2 and B3 and transfers to FLVEC-1+B2 to interpret the opcode.

5.5.4 PLABEL

The label beginning in column 1 is separated from the line using the subroutine PACKID and saved in the location ALABEL for later definition. FFLAG is set to +1 to indicate forcing upper is required and PLABEL exits to DCSCAN.

5.5.5 DCSCAN

Blanks between the label field and the opcode are skipped and the first two characters of the mnemonic are placed in

DOCUMENT CLASS IMS PAGE NO. 36.16
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

B2 and B3. 0CSCAN exits by transferring to FLVEC-1+B2.

5.5.6 OPCODE RECOGNITION TRANSFER VECTORS

Opcode fields are decoded by indexed jumps using succeeding characters of the mnemonic. Most instructions can be identified by examining the first two characters of the opcode field but for those that require further scanning, a vector for each of the third, fourth and fifth letters has been included along with the routine PNL which picks the next character from the input string and jumps back to the vectors. The vector exits that result from an executable instruction mnemonic will set X4 to an instruction prototype that is completed as the address field is processed. Other jumps that result from pseudo-ops and macro calls simply transfer to specific routines, although in some cases the spare 30 bits in the last vector position is used.

5.6 L3.RJ

5.6.1 This routine processes the RJ instruction. The address for the symbol is fetched from the two word symbol table using the REF subroutine and NFLAG is set to one indicating the next instruction is to be forced upper.

5.7 L3.JP

5.7.1 The JP may appear with or without a symbol in the address field but is always indexed by B1. If there is a symbol the subroutine REF is used to obtain its address. NFLAG is set to one and the routine exits to WRTEXT.

5.8 L3.XJP

5.8.1 This routine processes the X register conditional jumps ZR, NZ, PL, and NG. The register number is selected from the input string and REF is called to obtain the jump address before exiting to WRTEXT.

5.9 L3.EQ

5.9.1 The EQ jump may be either conditional or unconditional. If unconditional no B register is specified or Bi will be the same as Bj. In this case REF is called to obtain the jump address, NFLAG is set to one and exit is made to WRTEXT. When the two B registers present are different i and j are selected from the input string, added to the prototype and REF is called for the symbol definition before the exiting to WRTEXT.

5.10 L3.B00L

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 36.17
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

- 5.10.1 For boolean instructions the 15-bit opcode is determined by examination of the address field. This is done by observing the first and second operators to determine the g and h fields and setting i, j and k from the input string. The exit is made to L4.15.
- 5.11 L3.MX, L3.SH
- 5.11.1 This routine processes the mask instruction and the four shift instructions. Here it is required to determine whether the shifts are nominal or constant, reset the opcode in the former case and set the i, j and k fields in both cases. The exit is to L4.15.
- 5.12 L3.PUN
- 5.12.1 The assembler must select and set the i, j and k fields for the pack, normalize and unpack instruction. The exit is to L4.15.
- 5.13 L3.ARIT
- 5.13.1 The instruction for double and single precision floating point operations and the 60 bit integer operations require only the selection of a mask from ARITAB using the operator character code as an index and OR it with the prototype in X4. This routine exits to L3.PUN to set the i, j and k fields.
- 5.14 L3.SET
- 5.14.1 The increment unit instructions have the largest variety of address fields of any class of statements that are encountered by the assembler. This routine must determine the second digit of the opcode and the remainder of the 15 or 30 bit instruction. The analysis begins by transferring into L3.JVEC using the first character of the address field as an index. This will cause a transfer to L3.S1 if the first character is an A, B, or X to L3.S1IM if the first character is a minus sign, to L3.S7 if the first character indicates a symbol or constant follows and to ILL otherwise. At L3.S1 a character-by-character scan of the input string is used to determine if the first item in the address field is a register name or a variable. For variables a transfer is made to L3.S7 to evaluate the address expression while for registers the h field of the opcode is adjusted, j is selected and included in the instruction and the scan continues after setting X7 to remember the sign unless the next character is a zero byte. If the next item in the string is a register name the opcode is further adjusted, k is selected and set and control is transferred to L4.15. Otherwise, a branch is made to L3.S7 to continue the address expression analysis. At L3.S1IM, X7

DOCUMENT CLASS IMS PAGE NO. 36.18
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

is set to indicate the preceding minus sign and control goes to L3.S7 where the subroutine EVAL is called to evaluate the symbolic address field. The routine exits to WRTEXT with a 30 bit instruction.

5.15 L3.VFD

5.15.1 The VFD pseudo-op routine translates data subfields one by one, packing the information into the item being constructed. Numeric and symbolic fields are converted using the subroutine EVAL while character string data is packed and formatted by the VFD routine itself. The data fields that can be successfully assembled are limited as follows:

1. The total of the bits that are specified in any one appearance of a VFD must be 15, 30 or 60,
2. Relocatable fields must appear as the lower 18 bits of the specified field,
3. Character data must be C, H, L or R specification.

5.16 PDIS

5.16.1 The DIS routine moves the number of words specified from the line buffer to the current text table by calling WRTEXT once for each word.

5.17 PDATA

5.17.1 The only type of DATA fields the assembler will encounter are single octal constants. They are converted to binary by the subroutine CONVERT and added to the current text table by WRTEXT.

5.18 PBSS

5.18.1 The BSS routine first forces upper by setting FFLAG to 1 and calling WRTEXT. The argument is converted to binary by the subroutine CONVERT and added to the ORG counter, then FOTEXT is called to write out the current text table and start a new one with the updated origin counter. This routine exits to NOBINP0 to print the line.

5.19 PUSE, USENXT, USESTAR

5.19.1 The USE processor separates the relocation base name from the input string using PACKID and after determining whether it is common or local and selecting the correct portion of the 22 word ORGTAB it calls the subroutine USENXT to change TEXT.ADD to this relocation base entry. USENXT makes a linear search

DOCUMENT CLASS IMS PAGE NO. 36.19
 PRODUCT NAME FORTTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

of either the CORGTAB or the LORGTAB until it finds the relocation base name. It then saves TEXT.ADD in the location USEBB and resets it to the address of the new ORGTAB entry and exchanges the contents of NFLAG with the contents of the second word in the table entry. USENXT is also called by some of the pseudo macro routines. USESTAR is called to change back to the previous block by resetting TEXT.ADD from USEBB and exchanging NFLAGS.

5.20 PSET

5.20.1 The SET pseudo-op is used by FORTRAN to establish an address of the source data for a REPI pseudo-op to follow. Hence it appears as S_Δ SET M^P only. The subroutine DEF is called to redefine S_Δ to the current origin counter after forcing upper.

5.21 L3.ORG

5.21.1 The ORG instruction requires, after evaluation of the address field by EVAL, the relocation base to be changed and the origin counter in the new base to be reset. The subroutine FOTEXT is used to force out the old text table and start a new one with the correct origin counter.

5.22 PMACRO

5.22.1 The assembler skips over macro prototypes by changing the word at START to an EQ PMAC2^P and returning to NOBINP0 when a macro pseudo-op is encountered. This causes all future statements to transfer control to PMAC2 which checks for the ENDM command and restores the contents of START for normal processing. PMAC2 always exits to NOBINP0 to print the line.

5.23 PADDSUB

5.23.1 PADDSUB processes the address substitution macro, ADDSUB. This macro will be called by the program being assembled only once, in the event that it is a subprogram having parameters. After calling USENXT to change to the VARDIM. relocation block, its prototype at ADDSC, except for the last word, is written on the LGO file by WRSEQ. The last word will be 30 bits if MACHINE \neq 6600B, and 60 bits if MACHINE=6600B; it is written on the LGO file by WRTEXT. The correct relocation byte is added, and USESTAR is called to change back to the previous relocation block. Exit is to INITL.

The prototype is set to execute at a location specified by the LOC pseudo-op which just precedes it; its address field should be set to the address of the first word of the VARDIM. relocation block, which will be the same for all subprograms.

DOCUMENT CLASS IMS PAGE NO. 36,20
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.23.2 The macro prototype is:

```

ADDSUB MACRO FP
  USE VARDIM,
  SB4 I
  SA3 FP-1
  MX0 42
  SB6 60
  NO
B} SA3 B4+A3
  SB7 X1
  SA1 A1+B4
  SA2 X3
A} UX4 X3,B2
  SB3 A2
  LX4 42
  SB5 B6-B2
  SA3 A3+B4
  LX2 B2,X2
  SX5 X4+B7
  BX4 X0X2
  SA2 X3
  BX6 -X0X5
  IX4 X6+X4
  LX6 B5,X4
  SA6 B3
  NZ X3,A}
  NO
  NZ X1,B}
  JP ++1
  USE *
  ENDM

```

5.24 PSUB

5.24.1 PSUB processes the SUB macro. This macro is called by the program being assembled after it encounters a reference to a formal parameter. The form of the call is

SUB FP,K

where the comma and K may be missing and the form of the resulting entry is

59	57	56	48	47		18	17	0
2	POSC		K			ORGC		

Entry is made into the FP relocation block.

DOCUMENT CLASS IMS PAGE NO. 36.21
 PRODUCT NAME FORTTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.24.2 PACKID is called to strip the block name FP, which is left in position for the subsequent call to USENXT. The position counter {POSC} and the origin counter {ORGC} are extracted from the current block and saved in B registers. USENXT is called to change to the FP block. Returning, FFLAG is set to 1 to force upper and the position counter is converted from FTNXAS form to COMPASS form. Now the sublist entry will be formed, starting with the origin counter. BI contains the delimiter left by PACKID, and if non-zero, the constant is converted by CONVERT, shifted, and ORed into the entry. Lastly, the position counter is packed into the entry and WRTEXT is called to write the entry on the LGO file. The correct relocation byte is added, and USESTAR is called to change back to the previous block. Exit is to INITL.

5.24.3 The macro prototype is:

```
SUB MACRO FP,CON
  .POS SET 59-5
  .ORG SET x-5/59
  USE FP
  VFD 3/2,9/.POS,30/CON,18/.ORG
  USE x
  ENDM
```

5.25 PDELAY

5.25.1 PDELAY processes the DELAY macro. This macro is called by the program being assembled if it is necessary to call SUB twice in the same word for the same formal parameter. The form of the call is

DELAY FP

and the form of the resulting entry is

2	30	ST.
---	----	-----

where ST. is the address of the start of the sublist table. Entry is made into the FP block.

5.25.2 PACKID is called to strip the block name FP, and USENXT is called to change to the FP block. SYMBOL is then called to obtain the address of ST., which is shifted, marked off, and packed with the 30 field into X4. WRTEXT is called to write the entry on the LGO file. The correct relocation byte is added, and USESTAR is called to change back to the previous block. Exit is to INITL.

DOCUMENT CLASS IMS PAGE NO. 36.22
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.25.3 The macro prototype is:

```

DELAY MACRO FP
  USE FP
  VFD 3/2,9/30,30/0,18/ST.
  USE *
ENDM
    
```

5.26 PFILE

5.26.1 PFILE processes the FILE macro. This macro is called when a main program being assembled wishes to set up a FET and buffer for a file. It uses two prototypes, which are formatted as follows:

1) FET prototype:

	36	18	
FILEC	1	12	FIRST
			IN
			OUT
			LIMIT
		0	

2) REPI prototype:

59	54	53	27	26	0
43		2			1
					FWA OF ZEROS
	13				
59	42	41	18	17	0

5.26.2 CONVERT is called to convert the buffer length to binary. The origin counter is obtained, placed into the FIRST, IN, and OUT fields of the FET prototype at FILEC, and the buffer length + origin counter is placed into the LIMIT field. Since the zero word which is the sixth word of the FET must be repeated 13B times, its address is placed into the second word of the REPI prototype. The new origin counter is saved in PFILEC, and WRSEQ is called to write the prototype on the LG0 file. FOTEXT is then called to force out the current text table so that the REPI table can duplicate the zero word. Then the origin counter is reset to the value saved in PFILEC, and WRWDS2 is called to write the REPI table on the LG0 file. Exit is to INITL.

DOCUMENT CLASS IMS PAGE NO. 36.23
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.26.3 The macro prototype is:

```
FILE MACRO LN,NAME
  ENTRY NAME → .
  NAME → . BSSZ 1B
  IN# SET NAME → .
  LG# SET LN+1
  VFD 16/1,26/12,18/IN#+17
  VFD 60/IN#+17/60/IN#+17
  VFD 60/IN#+17+LG#
  BSSZ 14B
  BSS LG#
  ENDM
```

5.27 PNAME

5.27.1 PNAME processes the NAME macro. This macro is called by the program being assembled when a NAMELIST string is to be defined. Processing takes place in two phases: 1] stripping, partially processing, and saving the actual parameters, and 2] forming the required binary output and writing it onto the LG0 file. The format of the call is:

```
NAME N,T,BASE,BIAS,FP,D1,D2,D3
```

where N and T are always present, and the rest of the string may be entirely missing. In addition, BASE and BIAS may be concurrently missing, and the following combinations of the D fields may be missing: D1, D2, D3; D2, D3; D3.

5.27.2 Phase 1 begins by setting the locations from NNAME to Z3N, which will contain information derived from Phase 1 processing of the actual parameters, to zero. REF is called to obtain the NAMELIST name {N}, which is saved in NNAME, and its address, which is stored in VNAME. CONVERT is called, which converts the NAMELIST type {T}, and this is stored in TNAME. At this point a test is made to see if the next character is a zero byte. If it is, then there are no more parameters and transfer is made to phase 2 at PNAME4. Otherwise, a check is made to see if the BASE field is present, and, if it is, then the base is stripped by REF and stored in BASN and the bias is stripped by CONVERT and stored in BIASN. Then CONVERT strips the FP field, if present, and it is stored in FPN. At this point CONVERT is called to strip D1, D2, and D3 until one is found missing or all three are stripped, and they are saved in Z1N, Z2N, and Z3N, respectively. If all three fields are missing, Z3N is left at its initial value of zero, otherwise it is set to one. Phase 2 at PNAME4 begins by getting the NAMELIST name from NNAME, changing the trailing blanks to zero characters, and calling WRTEXT to write onto the LG0 file a word in the following format:

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 36.24
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

59	42	0
0	NAME	

If the BASE and BIAS fields were present, they are added together to form the address for the next word; otherwise the NAMELIST address from VNAME is used. This address and the NAMELIST type from TNAME are written on the LG0 file in the following format:

59	54	53	30	29	0
1	ADDRESS		TYPE		

Now if FPN is not zero, it indicates that a sublist entry must be made for the address just written: USENXT is called to change to the formal parameter block specified by NNAME and a sublist entry in the following format is written onto the LG0 file by WRTEXT:

59	57	56	48	47	0
2	59	VNAME			

Then USENXT is called to change back to the DATA block. The last word is written onto the LG0 file by WRTEXT in one of the following four formats, depending on the presence of D1, D2, and D3:

1) D1, D2, and D3 present:

59	54	53	36	35	18	17	0
0	D1	D2	D3				

2) D1 and D2 present, D3 missing:

0	0	D1	D2				
---	---	----	----	--	--	--	--

3) D1 present, D2 and D3 missing:

0	0	0	D1				
---	---	---	----	--	--	--	--

4) D1, D2, and D3 missing:

1	0	0	1				
---	---	---	---	--	--	--	--

Exit is to INITL.

DOCUMENT CLASS IMS PAGE NO. 36.25
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.27.3 The macro prototype is:

```

NAME MACRO N,T,BASE,BIAS,FP,D1,D2,D3
  LOCAL Z,Z1,Z2,Z3
  VFD 18/0,42/0L->N
  VFD 6/1,6/0
  IFC NE,#BASE##,1
  VFD 18/BASE+BIAS
  IFC EQ,#BASE##,1
  VFD 18/N
  IFC NE,#FP##,1
  SUB N
  VFD 30/T
  IFC EQ,#D1##,1
  Z SET 0
  IFC NE,#D1##,1
  Z SET 1
  IFC NE,#D3##,4
  Z1 SET D1
  Z2 SET D2
  Z3 SET D3
  IFEQ 0,1
  IFC NE,#D2##,4
  Z1 SET 0
  Z2 SET D1
  Z3 SET D2
  IFEQ 0,1
  IFC NE,#D1##,4
  Z1 SET 0
  Z2 SET 0
  Z3 SET D1
  IFEQ 0,1
  Z1 SET 0
  Z2 SET 0
  Z3 SET 1
  ENDIF
  VFD 6/Z,18/Z1,18/Z2,18/Z3
  ENDM

```

5.28 PENTR

5.28.1 PENTR processes the ENTR. macro. This macro will be called by the subprogram being assembled when an entry point other than the main entry point is to be provided. Although there are two types of ENTR. macro expansions possible, depending upon whether a subprogram has formal parameters or not, the expansion throughout a particular subprogram will be consistent, and a jump to the particular processor required will be stored over a word of NOP's located at the beginning of the initialization phase, at PENTR1. Thus, the first call to PENTR. will fall through PENTR1 and execute the initialization phase; subsequent

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 36.26
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

calls will be directed to the appropriate processor at PENTR1.
 The format of the call is

ENTR. NAME

ENTR. macro prototype with arguments:

59	44	29	17	0	Word:
0					1
SA2	{M+2}	BX6 X2	NO		2
SA6	{FTNNOP.}	EQ	{ENTRY.+1}		3
EQ	{M+1}	NO	NO		4
SA1	{NOPS.}	SA2	{NAME}		5
BX6 X1	LX7 X2	SA6	{FTNNOP.}		6
SA7	{ENTR.}	NO	NO		7

ENTR. macro prototype without arguments:

59	0	Word:	
0		1	
SA1	{NAME}	BX6 X1 NO	2
SA6	{ENTR.}	NO NO	3

Parentheses indicate values to be substituted.

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 36.27
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

5.28.2 REF is called to get the address of NAME and it is stored in ENAME. If this is not the first time through, PENTR1 transfers to PENTR6 if the subprogram has arguments, or to PENTR8 if it does not. Otherwise, initialization begins by testing word A of the second symbol table entry to see if the FP bit is set; if it is not, then there are no formal parameters and after setting a jump to PENTR8 into PENTR1, control is transferred to PENTR5 to complete the initialization for this case. Otherwise there are parameters, and the jump at PENTR1 is set to a jump to PENTR6. SYMBOL is called to get the address of NOPS., which is saved in ANOPS; FTNNOP., which is saved in AFTNN; and ENTRY., which is saved in AENTR. Control is then transferred to PENTR1 to jump to the proper processor. At PENTR5, initialization for the no parameters case continues as above with the calling of SYMBOL to save the address of ENTRY. in AENTR. PENTR6 is the start of the processing for a call to ENTR. in a subroutine with arguments. The origin counter is obtained to form a base for the self-relative substitutions in the macro prototype, and a jump around the entire macro expansion is formed and left in X4. Then the following substitutions are made:

1. $\diamond RGC+4 \rightarrow$ upper address of word 2
2. $\diamond RGC+5 \rightarrow$ upper address of word 4
3. FTNNOP. \rightarrow lower address of word 6
4. FTNNOP. \rightarrow upper address of word 3
5. ENTR.+1 \rightarrow lower address of word 3
6. NOPS. \rightarrow upper address of word 5
7. NAME \rightarrow lower address of word 5
8. ENTR. \rightarrow upper address of word 7

Now the jump in X4 is sent to the LG0 file by WRTEXT, and WRSEQ is called to write out the entire prototype onto the LG0 file. Exit is to INITL. PENTR8 is the start of the processing for a call to ENTR. in a subroutine with no arguments. The origin counter is obtained and a jump around the entire macro expansion is formed and left in X4. Then the following substitutions are made:

1. NAME \rightarrow upper address of word 2
2. ENTR. \rightarrow upper address of word 3

Now the jump in X4 is sent to the LG0 file by WRTEXT, and WRSEQ

DOCUMENT CLASS IMS PAGE NO. 36.28
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

is called to write out the entire macro expansion on the LGO file. Exit is to INITL.

5.28.3 The macro prototypes are:

```

  ENTR, MACRO NAME
    LOCAL X,Z,T
    EQ T
  NAME BSS 1
    ENTRY NAME
    SA2 X
    BX6 X2
    SA6 FTNNOP.
    EQ ENTRY.+1
  X EQ Z
  Z SA1 NOPS.
    SA2 NAME
    BX6 X1
    LX7 B0,X2
    SA6 FTNNOP.
    SA7 ENTRY.
  T BSS 0
  ENDM

```

when formal parameters appear on:

```

  ENTR, MACRO NAME
    LOCAL T
    EQ T
  NAME BSS 1
    ENTRY NAME
    SA1 NAME
    BX6 X1
    SA6 ENTRY.
  T BSS 0
  ENDM

```

when formal parameters do not appear.

5.29 EVAL

5.29.1 This subroutine is used to evaluate address expressions consisting of octal constants, symbols and the operators + and -. It must be entered by a return jump to EVAL with:

B3 = X1 = the first character of the expression,

X6 = plus or minus zero indicating a preceding minus
 {or implied plus},

DOCUMENT CLASS IMS PAGE NO. 36.29
PRODUCT NAME FORTTRAN Extended Version 2.0
PRODUCT MODEL NO. 3PC08 MACHINE SERIES 64/65/6600

X7 = any previous address sum,

B2 = bit count of the character in X1.

EVAL returns with the expression value in X7 and it has been added to X4.

5.30 REF

5.30.1 REF is used to obtain the equivalent address of any symbol or label encountered during the assembly process. This routine is entered with a return jump to REF with the symbol name in X1, the first character in B3 and the bit count of X1 in B2. If the name begins with either of the special characters $\{, \}$ or $\{, \}$ then its equivalent address is obtained from the generated label, actual parameter or variable dimension label definition table respectively by converting the coded index in the label to binary and indexing into the proper table with it. Otherwise, the subroutine PACKID is used to separate the label from the input string and format it left justified with blank fill in the lower 48 bits of X1 in preparation for calling SYMBOL or LABEL. If the symbol is an external name, X3 is set to zero {relative address} and the exit is made. If not external, word B of the symbol table entry is stored in RBTEMP unless it is already non-zero, in which case it is cleared to zero, the RA field is separated to X3 and the exit is made.

5.31 CONVERT

5.31.1 A display coded octal number which may be preceded by a minus sign is converted to binary. The first character of the constant must be placed in the lower 6 bits of X1 before the return jump entry. On return the converted value is in X1.

5.32 PACKID

5.32.1 This subroutine is used to separate identifier names from the input string. It will pack up to eight characters until a zero byte, +, -, /, blank or comma is encountered. The character string is left justified and blank filled in the lower 48 bits of X1. The character that served as the delimiter is preserved in B1 upon exit.

5.33 WRSEQ

5.33.1 WRSEQ writes up to 15 words of data with relocation information into the current text table. The calling sequence is:

DOCUMENT CLASS IMS PAGE NO. 36.30
 PRODUCT NAME FORTAN Extended Version 2.0
 PRODUCT MODEL NO. 3P C08 MACHINE SERIES 64/65/6600

A4 = first word address of text block,
 X4 = first word of text block,
 X7 = left justified 4 bit relocation bytes,
 RJ WRSEQ.

5.34 L4.15, WRTEXT

5.34.1 L3.15 is an alternate entry to WRTEXT used when a 15 bit quantity is to be written. WRTEXT builds and writes text tables for the loader from the instructions and data the assembler produces. The calling sequence is:

X4 = data to be written,
 B1 = bit count of data,
 B6 = return address,
 EQ WRTEXT.

WRTEXT forces upper when it is required by data size or the FFLAG is greater than zero. It also maintains the position counter and the origin counter for each relocation base and adjusts the relocation byte word in the text table. If the switch at WRTSW has been set WRLIST will be called to print the line.

5.35 F0TEXT

5.35.1 This subroutine is entered by a return jump. It terminates the current text table by forcing upper, installing the word count in word 1 & left justifying the data bytes in word 2. The text table is then written on the binary output file. Before returning a new text table is initialized with the origin counter in word 1.

5.36 WRLIST

5.36.1 This subroutine is responsible for producing the assembly listing. It is given the binary to print, if any, and the current source statement in ILINE. If the position counter is 60 the origin counter and the current relocation base name are also printed. If RBTEMP is non-zero the relocation base name of the address field will also be printed.

5.37 L4. CKL, DEF

5.37.1 After a statement has been decoded and the binary added to the text table these two subroutines are used to define the label that appeared on the statement if any. The pseudo-ops DATA and VFD result in transfer of control to L4.CKL but BSS calls DEF itself. At L4.CKL DEF is called if the contents of ALABEL are non-zero and control is passed to L4.CKRB. DEF looks up the

DOCUMENT CLASS IMS PAGE NO. 36.31
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3P C08 MACHINE SERIES 64/65/6600

symbol whose name is in X1 on entry in the two word symbol table using LSTPROC. For the label it then sets the RA field to the origin counter, the RL field to 1 if the current block is local, for common the RL field is set to 2 and the RB field to the ORGTAB ordinal for the block the symbol is defined in. Before the return location ALABEL is cleared to zero. DEF is disabled by the USE processor which stores an 'EQ DEF' in DEF.1 after the second 'USE CODE.'

5.38 L4.CKRB

- 5.38.1 Any 30 or 60 bit quantity that has been generated may have a relocatable address in the lower 18 bits. In this case RBTEMP will reflect this by having been set to word B of the two word symbol table entry, otherwise, it will be zero. If RBTEMP is non-zero the RL field is examined to determine the type of relocation necessary. Program relocation requires the relocation byte word in the current text table have a 2 added to it and RBTEMP be set to zero.
- A common or external reference requires a data byte for the loader be created and linked into the reference chain corresponding to the variable. For common the starting address of this chain is located in the lower 18 bits of the first CORGTAB word for the entry that corresponds to the common block in which the variable is defined. This word is located at {CORGTAB} + RB*22. The external reference chain begins in the lower 18 bits of the LINKTAB entry that contains the variable name. The RA field of RBTEMP contains this address. The new one word link is taken at the address contained in FREEMEM, which is incremented by one and compared to the contents of MEMEND and replaced. If FREEMEM is greater than MEMEND working storage has been exhausted and the error routine STOVER is called. Otherwise RBTEMP is cleared and L4.CKL exits to INITL to prepare for the next line.

5.39 ILL, SILL, STOVER

- 5.39.1 ILL picks up the message 'ILL' transfers it to location ILL.1 which stores the message in LINE+3, sets ILLFLAG to non-zero, calls WRLIST to print the message and the source image then exits to INITL to prepare for the next statement.
- 5.39.2 SILL, where control is transferred in the event LSTPROC cannot find a symbol name in the two word symbol table, picks up the message 'SYMBOL ERR' and transfers to ILL.1.
- 5.39.3 STOVER is called when all available working storage has been used. The first time this occurs the message 'STORAGE OVERFLOW, NO OBJECT PROGRAM WILL BE PRODUCED' is printed. The size of the overflow is added to the contents of location STOVSIZE and FREEMEM is reset to the contents of MEMSTART. The exit is to

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 36.32
PRODUCT NAME FORTRAN Extended Version 2.0
PRODUCT MODEL NO. 3P COB MACHINE SERIES 64/65/6600

INITL.

5.40 PEND

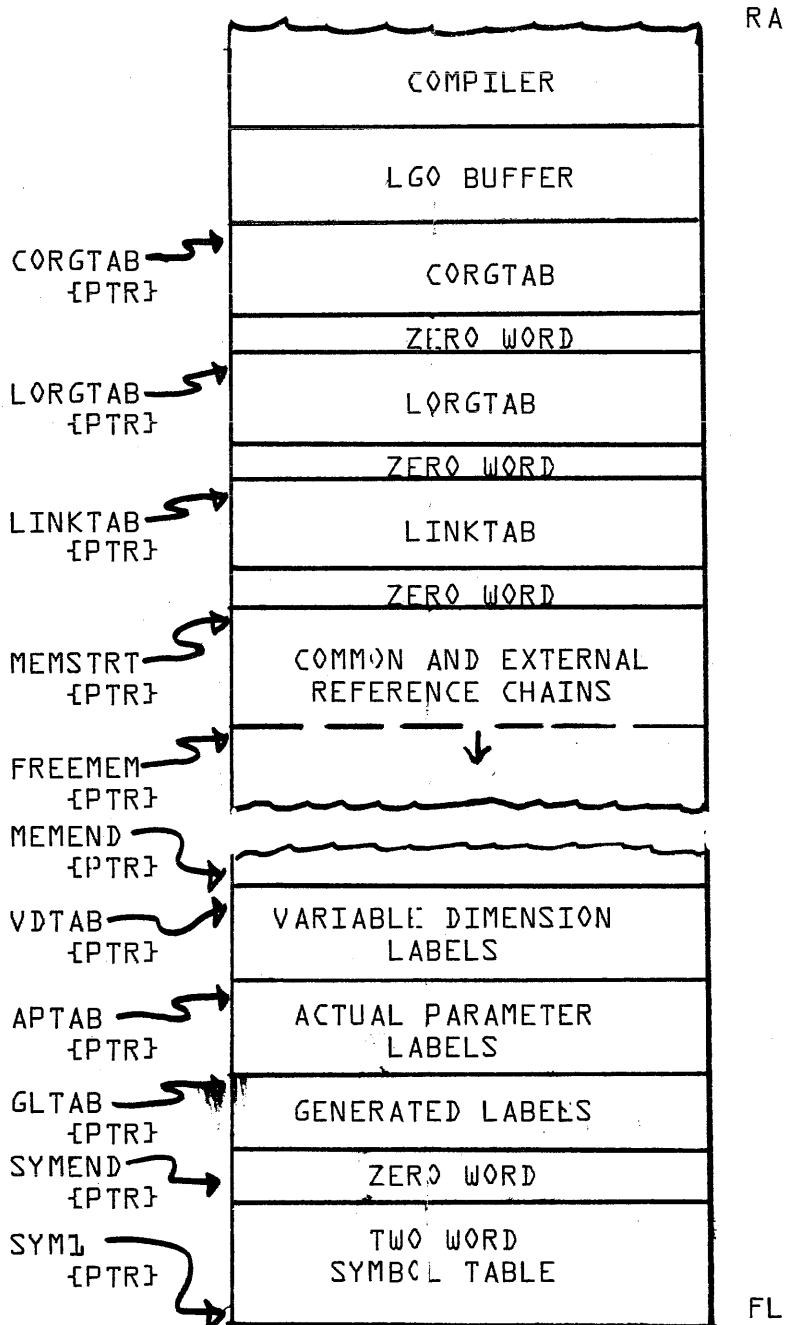
5.40.1 Control is transferred to PEND when the END pseudo-op is encountered to clean up the assembly process and terminate. The following tasks are performed to accomplish this:

1. If the contents of STOVSIZE is not equal to zero the size of the overflow is calculated and printed with the message " INCREASE FIELD LENGTH BY XXXXXX" and exit is made as if ILLFLAG was non-zero.
2. If the contents of ILLFLAG are non-zero the following steps are taken before exiting:
 - a. An end-of-record is written on LG0,
 - b. LG0 is backspaced one logical record,
 - c. A prefix table with the program name is written on LG0,
 - d. Control is transferred to EX.90 to do another end-of-record write on LG0, print the END pseudo-op and return.
3. All partially filled text tables are forced out on LG0 from CORGTAB and LORG TAB by calling F0TEXT once for each relocation base.
4. The accumulated common and external reference information is formed into FILL and LINK tables respectively and written on LG0. This is accomplished by following the chain that begins at the CORGTAB and the LINKTAB entries for common block and external symbols and packing the data bytes which are the upper 30 bits in each link into the correct table formats. The COMPS buffer is used as scratch memory for this purpose.
5. An XFER table containing the date and the transfer symbol, if present, is written on LG0.
6. The LG0 buffer is cleared out and the relocatable deck terminated by doing an end-of-record write on LG0.
7. The last statement of the program {the END Pseudo-op} is printed by calling WRLIST.
8. IOPARM is reset to 14B to cause output to be in BCD mode, ILLFLAG is cleared.
9. The assembler returns thru its entry point to DPCL0SE in Pass 2 for the 4 overlay and the non-overlay systems or to FAXRET in DPCL0SE in Pass 2 for the 5 overlay system.

DOCUMENT CLASS IMS PAGE NO. 36.33
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3P C08 MACHINE SERIES 64/65/6600

6.0 Data Formats

Working storage during the assembly process.



DOCUMENT CLASS TMS PAGE NO. 36.34
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3P C08 MACHINE SERIES 64/65/6600

6.1 22 Word ORGTAB {LORGTAB and CORGTAB}

6.1.1 These two tables have the same format. They contain a 22 word entry for each relocation base that will be encountered by the assembler. COMMON relocation bases are entered in the CORGTAB and local relocation bases are entered in LORGTAB. Each 22 word entry is formatted as follows:

	59	17	0
WORD 1	BLOCK NAME		PTR
2	NFLAG		
3		LCT	ORGC
4			POSC
5			TABC
6	TEXT TABLE ID WORD		
7	TEXT TABLE RELOCATION BYTE WORD		
8	FIRST TEXT WORD		
9	TEXT {UNINITIALIZED}		
22			

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. 36-35
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3P COB MACHINE SERIES 64/65/6600

- WORD1: Bits 18-59/Relocation base name, left adjusted with blank fill,
 Bits 0-17/Pointer to Fill chain for COMMON blocks; unused for locals. Initialized to zero in both cases.
- WORD2: Bits 0-59/NFLAG {force upper, next instruction}, initialized to zero.
- WORD3: Bits 24-59/Unused,
 Bits 18-23/Relocation base code. This is 1 for all local blocks and may be 3 to 77B for common blocks. It serves as the LCT ordinal for the loader.
 Bits 0-17/Origin counter, initialized to the first word address of the block.
- WORD4: Bits 18-59/Unused,
 Bits 0-17/Position counter, initialized to 60.
- WORD5: Bits 18-59/Unused,
 Bits 0-17/Test table ordinal indicates the current word being filled, initialized to 2.
- WORD6: Bits 54-59/Text table code number, 40B,
 Bits 36-47/Word count, initialized to zero,
 Bits 18-23/Relocation code,
 Bits 0-17/Load address, initialized to ORGC,
- WORD7: Bits 0-59/4 bit relocation fields, initialized to zero.
- WORD8: Bits 0-59/Initialized to zero.
- WORD9-22: Bits 0-59/Uninitialized.

6.2 LINKTAB

- 6.2.1 The LINKTAB consists of one one-word entry for each external symbol used in the program being assembled. The last entry is a zero word. Each word is formatted as below,

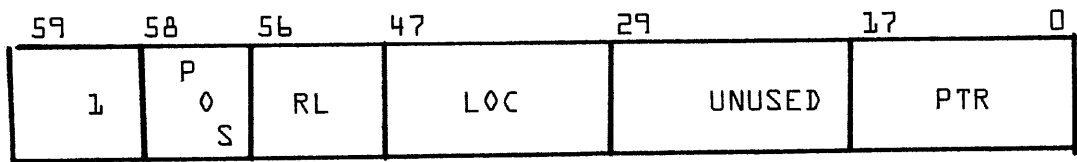
DOCUMENT CLASS IMS PAGE NO. 36.36
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3P COB MACHINE SERIES 64/65/6600



where NAME is the external symbol name, left adjusted with zero fill and PTR is the start of the reference chain that describes the usage of the symbol. PTR is initialized to zero.

6.3 LINK and FILL chains {external and common reference information}.

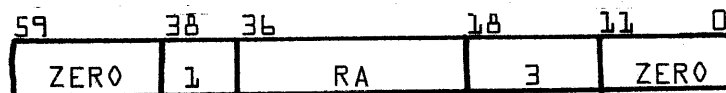
6.3.1 One chain exists for each common relocation base and external symbol. They are linear linked lists of one word elements containing in upper 30 bits a data byte for the loader and in the lower 18 bits a pointer to the next link. Each list is terminated by a zero pointer. The elements of these lists are taken from working storage beginning above the LINKTAB. FREEMEM always points to the next available word in this area. Following is the format of each link.



where LOC specified the relative address of the reference, RL specifies the relocation of this address, POS the position in the word and PTR points to the next link in the chain.

6.4 Actual Parameter, Variable Dimension, and Generated Label Definition Tables.

6.4.1 Upon entry these tables contain in the lower 18 bits the address relative to the origin of the Code. relocation base [AP,] VD or ≡GL labels. They are reformatted during initialization and take the following form during assembly.

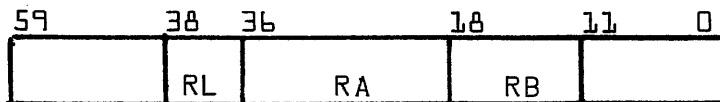


where RA is the program relative address of the label.

6.5 TWO WORD SYMBOL TABLE

6.5.1 During initialization of the assembler and during assembly before the second 'USE CODE.', word B of the two word symbol table entry for each symbol is reformatted as follows:

DOCUMENT CLASS IMS PAGE NO. 36.37
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3P COB MACHINE SERIES 64/65/6600



where RA is a program relative or common block relative address or a pointer to the LINKTAB entry that corresponds to the external symbol, RL is the type of relocation {0}=program, 02=COMMON, 03=EXTERNAL} and RB is an ordinal to either the CORGTAB or LORGTAB entry which corresponds to the relocation base in which the symbol was defined.

7.0 Modification Facilities

7.1 Assembly Options

7.1.1 Options File

1. OVERLAY. When set to a non-zero value this option causes a non-overlay compiler to be generated. This has two effects on the assembler: {1} MEMORY, the first word address of working storage must be set to the end of the compiler rather than the end of the assembler, and {2} several flags and switches in FTNXAS and in LSTPROC must be reset to their initial value upon exit since the same copy of the assembler will possibly be used again.
2. MACHINE. This option, which is set to either 6400B or 6600B, causes the ADDSUB Macro to be expanded with a `JP *+1` in the last two parcels if set to 6600B.
3. LGOSIZE. The assembler will allocate LGOSIZE+1 words for the binary output buffer.
4. ASSEMBLE. The assembler will produce a LCC overlay {FTNX, 1, 3} card and a transfer address of FTNXAS# and will return to FAXRET in close 2 via a direct jump.

7.1.2 DEBUG ETC.

This option and several assembly flags which are normally equated to it, control debugging aids that have been left in the assembler. The normal value of DEBUG is zero, however setting it to 1 and reassembling will cause {1} SNAP calls to be inserted at strategic points, {2} ORG and MOVE macros not to be expanded, and {3} the last few words of L3.JVEC to be assembled in. The SNAP routines must be available when DEBUG is non-zero.

7.2 Opcode Vectors

DOCUMENT CLASS IMS PAGE NO. 36.3A
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3P COB MACHINE SERIES 64/65/6600

7.2.1 Opcode fields are decoded by the assembler by placing the first two characters of the field in index registers B2 and B3 and jumping to B2+FLVEC-1. If the opcode can be uniquely recognized from the first two letters, an exit is made to the correct routine for processing the address field. Otherwise, the address of a further vector is placed in B2 and the subroutine PNL is called to separate the next character from the input string, add it to B2 and jump to the contents of B2. New entries and new vectors can be placed anywhere in the existing general scheme, but care must be taken to insure that a valid vector entry is not within the range of an ORGSTART, ORGEND pair. Also, each unused word in new vectors should contain an 'EQ ILL' and each block of two or more unused words should be surrounded by ORGSTART, ORGEND macro calls.

7.3 MIC, ORG, MOVE Macros

7.3.1 MIC creates a micro with the name equal to the second parameter. It will be a character string representing the value of the first parameter. Thus, after

```
N SET 423
MIC N,Z
```

the catenation A B C ≠ Z ≠ would equal ABC423

The purpose of the move macros are to allow the definition of unused areas in the assembler, of areas of code which can be moved, and the moving of the code into the unused areas at assembly time.

The unused areas are bracketed by ORGSTART and ORGEND macro calls, which build tables of lengths and first word addresses of unused areas. After the last ORGEND macro call, the areas of code which can be moved are bracketed by MOVSTART and MOVEND macro calls. Each pair of calls result in the relocation of the associated code into the smallest area in which it will fit. The tables are adjusted to reflect the usage. If all unused areas are too small, no action is taken. Changes in the size of a code block bracketed by MOVSTART, MOVEND must be reflected in the MOVSTART call parameter or an ERR pseudo-op will be produced.

DOCUMENT CLASS IMS PAGE NO. 37.1
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. 3P COB MACHINE SERIES 64/65/6600

ORGTAB

1.0 General Information

ORGTAB resides in the 1.0 overlay which also contains LSTPROC the symbol table handler. ORGTAB contains entry points to hold the local and common relocation block lengths. It also contains a copy of WRWDS the output and circular I/O buffer control program.

2.0 Usage

The length of all common relocation bases and all local relocation bases {except CODE. and VARDIM. which are determined during Pass 2 by POST and CLOSE V2} is determined during Pass 1 {the 1,1 overlay} and retained in ORGTAB for use by the assembler at the end of Pass 2.

2.1 Entry Points.

- | | | |
|--------|---------|--|
| 2.1.1 | ORGTAB | ,75B locations to hold lengths of all possible common relocation blocks. |
| 2.1.2 | START^ | ,holds length of START. relocation block. |
| 2.1.3 | VARDIM^ | ,holds length of VARDIM. relocation block. |
| 2.1.4 | ENTRY^ | ,holds length of ENTRY. relocation block. |
| 2.1.5 | CODE^ | ,holds length of CODE. relocation block. |
| 2.1.6 | DATA ^ | ,holds length of DATA. relocation block. |
| 2.1.7 | DATA^^ | ,holds length of DATA. relocation block |
| 2.1.8 | HOL^ | ,holds length of HOL. relocation block. |
| 2.1.9 | GLTAB | ,holds address of ≡GL label definitions. |
| 2.1.10 | APTAB | ,holds address of [AP label definitions. |
| 2.1.11 | VDTAB | ,holds address of]VD label definitions. |
| 2.1.12 | OT.SIZE | ,holds the number of words needed for optimizing temporary storage. |
| 2.1.13 | NDOTEMP | ,holds the number of words needed for DO loop temporary storage. |

In addition to facilitate the non-overlay configuration the following entry points have been moved to the 1,0

CONTROL DATA CORPORATION

DIVISION

DOCUMENT CLASS IMS PAGE NO. _____
 PRODUCT NAME FORTRAN Extended Version 2.0
 PRODUCT MODEL NO. *P COB MACHINE SERIES 64/65/6600

overlay. CARDCT, ENDLIST, ENDSEQ, FWAWORK, LWAWORK,
 HIGHORD and FWASEQ.

3.0 Diagnostics

Not applicable.

4.0 Environment

ORGTAB resides in the 1,0 overlay. The only executable
 code is the output and buffer control program WRWDS which
 is self contained.

5.0 Structure

The tables are ordered consecutively so they may be
 initialized easily between compiling subprograms.

6.0 Formats

The common block information is held starting at ORGTAB
 in the following manner:

Common Block Name	Word Length
42	18

The last common block name is followed by a zero word.

The local block lengths are held in the lowest 18 bits.
 All lengths are in binary.

DOCUMENT CLASS IMS PAGE NO. A-1
 PRODUCT NAME FORTRAN Extended
 PRODUCT NO. 4P516 VERSION _____ MACHINE SERIES 64/65/6600

R-List Language Description

R-list is an intermediate language which is intended to be both easy to generate and convenient to produce good assembly code from. The language is register independent as much as possible; there shall be a provision for specifying specific registers. A limited macro facility shall be provided to facilitate the generation of frequently occurring code and reduce the size of the intermediate R-list. Generally the operation code in R-list will correspond to the operation code of the associated machine instruction.

1. General Description

There are four formats for R-list instructions to permit maximum description of the operation requested in the minimum amount of space. The formats are shown in figure one. In all cases the following definitions will hold.

First word will have the same format:

Bit 59 = 0

Bit 58 = 1

Bit 47-57 = Operation Code (OC)

It is formed by placing the operation code in a B register and appending it to the rest of the information in the first word of the R-list instruction by using a PACK instruction. Any negative words encountered in the R-list will be ignored.

1.2 R Fields

All R fields will be sixteen bits long. All R fields generated during pass one will have the high order bit equal to zero; intermediate R's resulting from macro processing will have that bit set.

1.2.1 RI Fields

RI indicates the result of the operation indicated by OC or an

DOCUMENT CLASS IMS PAGE NO A-2
 PRODUCT NAME FORTRAN Extended
 PRODUCT NO. 4P616 VERSION _____ MACHINE SERIES 64/65/6600

operand of a branch instruction. This field will always be rightmost in the first word whenever it occurs.

1.2.2 RJ Fields

These fields indicate the first operand of a triple address instruction.

1.2.3 RK Fields

These fields indicate the second operand of a triple address instruction.

1.2.4 RF Fields

These indicate the index function to be added to a base address or may be used to indicate an operand of a branch.

1.3 CA Fields

CA fields are eighteen bits long and usually contain the constant addend to be added to the reference to an array element. They may also contain some other constant.

1.4 I and H Fields

The I field is twelve bits long and identifies the table for which H is an ordinal. The H1 field is eighteen bits long. The defined values for I are as follows:

0 = Symbol Table	5 = APLIST ordinal
1 = Statement Temporary	6 = 1 Branch if label
2 = DO Range temporary	
3 = Subprogram temporary	7 = Requests a zero entry in a plot
4 = Conlist ordinal	

If the H1 field is negative or zero both I and H will be ignored. If the H2 field is non zero, it is interpreted as an ordinal in the symbol table and the final symbol generated will be H1-H2. All H1 are interpreted as ordinals in the list specified by I.

1.5 IN Field

IN fields are eighteen bits long and contain constants.

1.6 SO Fields

SO fields are fourteen bits long and modify the operation code.

DOCUMENT CLASS IMS PAGE NO. A-3
 PRODUCT NAME FORTRAN Extended
 PRODUCT NO. 4P616 VERSION _____ MACHINE SERIES 64/65/6600

The high order bit of the SO Field is used only in macro descriptions. The next two bits specify respectively, a sequence terminating and unlocking instruction, a locking instruction. A locking operating reserves a register for the RI of the operation, this reservation holds until an instruction with the next to high order bit in the SO field is encountered. The next bit is set to indicate a register specification is not to be held until the end of a sequence. The low order two octal digits are used to specify a class of registers or a particular register. The right hand two bits specify a register class as follows:

- 1 - X
- 2 - B
- 3 - A

If the next bit is set the other octal digit specifies the particular register of the class. This specification stipulates which register the RI of the instruction is to be placed in, i.e., the destination register of the operation. Currently only X's and B's may be specified as destination and the particular one must be stated.

2. Type I R-List

2.1 Type I Format

All type I instructions occupy one word and consist of operation code, RJ, RK, and RI fields in that order.

2.2 Type I Operations

<u>Description</u>	<u>R-List Code</u>	<u>Machine Code</u>
Transmit	10	10,22
AND	11	11
OR	12	12
Exclusive or	13	13
Transmit complement	14	14
Stroke	15	15
Implication	16	16
Equivalence	17	17
Floating add	30	30
Short add (after all other uses of RI)	2	53
Short difference load	3	57
Short difference store	7	57

DOCUMENT CLASS IMS PAGE NO A-4
 PRODUCT NAME FORTTRAN Extended
 PRODUCT NO. 4P616 VERSION _____ MACHINE SERIES 64/65/6600

2.2 Continued

<u>Description</u>	<u>R-List Code</u>	<u>Machine Code</u>
Floating subtract	31	31
Double Floating add	32	32
Double Floating subtract	33	33
Integer add	36	36
Integer subtract	37	37
Floating Multiply	40	40
Double floating multiply	42	42
Floating divide	44	44
Short add	46	6N, 7N n=3,4,6
Short subtract	67	67,77
Short Load	41	57
Short Store	45	56,57
Index right shift	23	23
Index left shift	22	22
Normalize	24	24
Round and normalize	25	25
Unpack	26	26
Pack	27	27
Shift transmit	62	10,22
Unpack into B0	34	26

3. Type II R-list

3.1 Type II R-List Format

All Type II instructions occupy one word and consist of operation code IN field, SO field, RI field, in that order.

3.2 Type II Operations

<u>Description</u>	<u>R-List Code</u>	<u>Machine Code(s)</u>
Form Mask	43	43
Set	61	61,71
Define	53	NA
Register store	52	NA

DOCUMENT CLASS IMS PAGE NO. A-5
PRODUCT NAME FORTTRAN Extended
PRODUCT NO. 4P616 VERSION _____ MACHINE SERIES 64/65/6600

The Type II set operation may only be used for placing an 18 bit constant into a B or X register.

The Define operation is used to set an initial condition for the sequence of which it is a member. A Define states that RI is in the register specified by the SO field. The intended use of the Define is to specify the whereabouts of results of function references on returning.

The register store is used to stipulate the destination register of the previous operation of which RI was the result. In effect this is used to permit the appending of an SO field to a Type I instruction. Once an R is placed in a register by an SO or register store specification the register is no longer available until the next sequence.

Neither Define nor Register Stores result in the production of any executable code.

Define and Register Stores differ from locks in that locked definitions hold until the end of DO loop jump is encountered which may be several sequences. Define and register store specifications have meaning only within their own sequences.

If an SO field specifies a reserved register the reservation will be overridden. Moral: garbage in, garbage out.

4. Type III R-list

4.1 Type III R-List Format

All type III instructions occupy two words. The first word consists of the operation code, CA field, SO field, and an RI field in that order. The second word contains a RF field, I field, and an H field right justified.

DOCUMENT CLASS IMS PAGE NO A-6
 PRODUCT NAME FORTRAN Extended
 PRODUCT NO. 4P616 VERSION _____ MACHINE SERIES 64/65/6600

4.2	Type III Operations	R-List	Machine
	<u>Description</u>	<u>Code</u>	<u>Code(s)</u>
	Load	50	50 - 57
	Store	51	50 - 57
	Jump if RI=RF	70	04
	Jump if RI .NE. RF	71	05
	Jump if RI .GE. RF	72	06
	Jump if RI .LT. RF	73	07
	Zero X Jump	74	030
	Nonzero X Jump	75	031
	Positive X Jump	76	032
	Negative X Jump	77	033
	Indexed Jump	102	02
	Set	104	60 - 77
	APLIST	56	NA
	Left Shift (CA)	20	20
	Right Shift (CA)	21	21

APLIST entries will specify the address of the actual parameter in the I, H1, and CA fields. The RI field will contain an ordinal which will say which parameter list this entry is a member of. Although entries for a given argument list (identical RI fields) need not be consecutive APLIST entries then must be in order.

Any R which is operated on by either a 20 or 21 operation must be defined by shift transmit immediately prior to the shift operation.

5. Type IV R-List

5.1 Type IV R-List Format

All type IV instructions occupy one word, consisting of an operation code, CA, I and H fields.

	<u>Description</u>	<u>R-List Code</u>	<u>Machine Code(s)</u>
	Unconditional Jump	54	0400
	Return Jump (60 bit)	101	01
	Entry	55	NA

DOCUMENT CLASS IMS PAGE NO A-7
 PRODUCT NAME FORTTRAN Extended
 PRODUCT NO. 4P616 VERSION _____ MACHINE SERIES 64/65/6600

5.1	Continued	R-List	Machine
	<u>Description</u>	<u>Code</u>	<u>Code(s)</u>
	End of Statement	57	NA
	End of Sequence	100	NA
	Label	60	NA
	End of RLIST	103	
	Return Jump (30 bit)	105	01

6. MACROS

6.1 Macro References

Macros are referenced using a Type II format. The OC field contains the macro ordinal and is formed by placing the complement of the macro ordinal (MO) in a B-register and concatenating it to the rest of the information by use of a PACK instruction. The IN field contains the number of words which follow which contain parameters.

6.2 Actual Parameters

The actual parameters immediately follow the macro reference in the following order:

1. symbols, if any
2. R's, if any
3. constants, if any

6.2.1 Actual symbolic parameters are specified two to a word, the first in the lower half of the word, the second in the upper half and so on. Each half word consisting of a right justified I field and H field.

6.2.2 Actual R parameters are three to a word, as is Type I format, with the first being rightmost.

6.2.3 Actual constant parameters are specified three to a word in 18 bit fields right justified.

DOCUMENT CLASS IMS PAGE NO A-8
PRODUCT NAME FORTRAN Extended
PRODUCT NO. 4P616 VERSION _____ MACHINE SERIES 64/65/6600

6.3 Macro Descriptors

The descriptor locates the macro text and specifies its length and the number of parameters of each type. It occupies one word and has the following format:

Bits 0 - 17	MA	beginning address of the macro
Bits 21 - 35	ML	length of the macro in words
Bits 36 - 41	NI	number of generated intermediates
Bits 42 - 47	NK	number of integer constant parameters
Bits 48 - 53	NR	number of R parameters
Bits 54 - 59	NS	number of symbolic parameters

6.4 Macro Text

The macro text is written in normal R notation with the following modifications:

1. If the high order bit of an RI, RJ, or RK field is not set and is greater than 1 it is interpreted as a formal reference to an R; the remainder of the field specifies the ordinal of the actual parameter list.
2. If the H field is zero the IH pair is interpreted as a formal reference with the I field containing the ordinal of the actual parameter in the parameter list.
3. IN and CA fields are interpreted as containing ordinals of actual parameters if the high order bit of the SO field is set. The use of type IV format instruction requiring a CA field as a formal parameter is not allowed.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. A-9
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

7.0 Four types of RLIST entries

TYPE I	2	0C 10	RJ 16	RK 16	RI 16
TYPE II	2	0C 10	IN 18	S0 14	RI 16
TYPE III	2	10 0C H2 14	18 CA RF 16	14 S0 I 12	16 RI H1 18
TYPE IV	2	0C 10	CA 18	I 12	H1 18

8.0 Defining R-List Macros

All macros are contained in the routine MACROX which is part of Pass 2. A set of COMPASS macros has been constructed to facilitate the definition of R-list macros. The following describes their use.

Structure: Each macro definition must begin with an RMACRO macro reference followed by the text of the R-list macro followed by an ENDR macro reference.

RMACRO. The RMACRO reference has the following form:

0C - RMACRO
Address field - NOSY, NOR, NOK

where

NOR - the number of R's in the macro reference.
NOSY - the number of IH fields in the macro reference.
NOK - the number of constants in the macro reference.

ENDR. The ENDR reference has the following form:

0C - ENDR

Text. Text is written with R-list mnemonic operation codes in the 0C field; a list of them is attached. The variable field will contain the arguments to the operation in the following orders:

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. A-10
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

TYPE I RI, RJ, RK
TYPE II RI, IN, SO
TYPE III RI, RF, CA, SY, SO
TYPE IV CA, SY

Omitted fields are interpreted as zero.

R-fields. All R-fields {RI, RJ, RK, and RF} must either begin with an I or P or must consist of 1 or zero. If it begins with an I or P, the rest of the field must be a decimal quantity giving the ordinal of the R in the associated list.

I - the list of R's generated by the macro processor. These must be numbered greater than 0.

P - the list of R's in the parameter list referencing the macro.

0 and 1 are the constant R's located in B0 and A0 respectively.

IN and CA fields. These fields must either be empty or contain an X or a B followed by a digit in the range 0-7 followed by a period. If an X or B is specified, that specification may optionally be preceded by a T to indicate a temporary register assignment.

SY fields. These must be either empty or a constant in which case it is interpreted as being an ordinal in the list of IH parameters.

Example:

```
ABSF      RMACRO      0,2,0  
          SXT         I1, P1  
          KRS         I2, I1, 59  
          XOR         P2, I2, P1  
          ENDR
```


CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS TMS PAGE NO. B-1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

FORTRAN Extended I/O Calling Sequences

READ, WRITE, PRINT, PUNCH

First entry B2 = address of FET or complimented address of variable tape number. S2 → P2

B3 = address of format statement or complemented address of variable format number {for coded only, B3 is not set for binary}. S3 → P3

RJ INPUTBI./INPUTCI./OUTPUTBI./OUTPUTCI. SI

Intermediate entries

B1 = address of data item or beginning address of array. PI

(B2 = array length {# of words} or 1.)

RJ INPUTB./INPUTC./OUTPUTB./OUTPUTC.

Final entry

B1 = -1

RJ INPUTB./INPUTC./OUTPUTB./OUTPUTC.

ENCODE, DECODE

First entry

B1 = address of packed data.

B3 = address of format statement or complemented address of variable format.

B4 = character length or complemented address of variable character length.

RJ DECODI./ENCODI.

Intermediate entries

B1 = address of data item or beginning address of array.

(B2 = array length {# of words} or 1.)

RJ DECODE./ENCODE.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. B-2
PRODUCT NAME F0RTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

Final entry B1 = -1
 RJ DECODE./ENCODE.

BUFFER IN, BUFFER OUT

{single entry} B1 = mode, or complemented address if variable mode.
 B2 = address of FET or complemented address of variable tape number.
 B3 = Fwa of data block.
 B4 = lwa of data block, or complemented lwa of data block if type double or complex.
 RJ BUFEI./BUFE0.

NAMELIST

{single entry} B1 = address of NAMELIST information.
 B2 = address of FET or complemented address of variable tape number.
 RJ INPUTN./OUTPTN.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. C-1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 _____ MACHINE SERIES 64/65/6600

SUBROUTINE LINKAGE AND FORMAL REFERENCING

All references to formal parameter in the following circumstances will be direct, i.e. address substitution will be performed at subroutine initialization time -

1. Array element reference
2. Entry in an actual parameter list
3. Reference within a DD
4. Reference to a formally defined subprogram

All remaining references will be via indirect references. The address of the actual parameter list will be maintained in AD throughout the execution of the text of the execution of the text of the subprogram. The previous AD will be saved in local memory.

SUBROUTINE LINKAGE

Calling Sequence:

SAI APLIST
RJ SUBR

Where APLIST is the address of the first element of the actual parameter list. The list has the addresses of the actual parameters stored in order one per word in the low order position with zero fill. The list is terminated by a full word of zero. SUBR is the subroutine name.

Substitution List Format:

Each non-zero entry has the following format:

	2	S	G	CA	G	IA
	<u>8</u>					
59	57	48	37	20	17	0

Where

IA is the address of the instruction to be modified
CA is the constant addend

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. C-2
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

S is the number of bit positions to be shifted over to place the address field of interest in the low order 18 bits, i.e., 0, 30 or 45

G is unused

The list is ordered in sequence of the associated formal parameters. Following all entries referencing a given formal parameter is a zero word followed by the entries associated with the next formal parameter.

Restriction:

Consecutive entries in the substitution list may not reference the same instruction word.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. D-1
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

SIO Description

1.0 The function of SIO is to perform all communication with the files used during compilation. The files are assigned by logical unit number and are referenced by this fixed unit number.

1.1 The logical unit assignment is as follows:

<u>Number</u>	<u>File</u>
1	INPUT
2	OUTPUT
3	LGO
4	RLIST
5	COMPS

SIO is located in the 0.0 overlay which is called FTNSIO.

2.0 Entry Point Names

2.1 OPEN

2.1.1 This entry point sets up the FET.

2.1.2 Calling Sequences:

SB1	Buffer Size
SB6	Unit Number
SB7	fwa
RJ	OPEN

The buffer size must be at least a PRU+10 in size (see ERS SCOPE 64/6600 Version 3.0 for definition of PRU). The FET is located at fwa thru fwa+10 in the buffer.

2.2 CLOSE

2.2.1 This entry is used to dump a write file. The specified buffer (file) is emptied with an end-of-record write request.

2.2.2 Calling Sequence:

SB6	logical unit number
RJ	CLOSE

2.3 REWIND

2.3.1 This entry is used to rewind a write file. The specified buffer (file) is emptied with an end-of-file write and rewind.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. D-2
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

2.3.2 Calling Sequence:

SB6 logical unit number
RJ REWIND

2.4 LIST

2.4.1 This entry point is used for writing on the OUTPUT file. Page ejects and titles are inserted when necessary. If a page eject is wanted, it is only necessary to set LCNT (line count) to zero.

2.4.2 Calling Sequence:

SB1 number of words
SB6 logical unit number
SB7 fwa of the array
RJ LIST

The words to be put out must be in display code and the last word must contain at least 12 bits of zero in the low order position.

2.5 RDWDS

2.5.1 This entry is used to read a file which was created by the corresponding write request.

2.5.2 Calling Sequence:

SB1 number of words
SB6 logical unit number
SB7 fwa of array
RJ RDWDS

Upon return the number of words read will be B1. X4 will contain the status. If X4 .GT. 0 this indicates the last read encountered on end-of-file. If X4 .LT. 0, this indicates the buffer is static and contains useful information. If X4 .EQ. +0, this indicates an end-of-record was encountered during the last read.

2.6 RDCARD

2.6.1 This routine is used to read card images from the disk. The characters are separated into one per word right justified zero fill. The requestors array must be 40 words in length. The remaining words in the array are zero filled with a display code blank in the low order position.

2.6.2 Calling Sequence:

SB6 logical unit number
SB7 fwa of array
RJ RDCARD

The return parameters are defined in 2.5.2.

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS _____ IMS _____ PAGE NO. _____ D-3
PRODUCT NAME _____ FORTRAN Extended _____
PRODUCT MODEL NO. _____ 4P616 _____ MACHINE SERIES _____ 64/65/6600 _____

2.7 WRWDS

2.7.1 This entry point is used to write words in a continuous record on a file.

2.7.2 Calling Sequence:

SB1	number of words
SB6	logical unit number
SB7	fwa of array
RJ	WRWDS

A CLOSE or REWIND must be issued to insure that the file contains all valid information.

2.8 STATUS

2.8.1 This entry is used to determine the STATUS of a file.

2.8.2 Calling Sequence:

SB6	logical unit number.
RJ	STATUS

The status parameters in X4 which are described in 2.5.2 are returned.

2.9 TITLE1

2.9.1 This location is the fwa of a 100 character title. The title is initially blank except for the page count.

2.10 LCNT

2.10.1 This is the location of the line count, it is initially zero which will cause a page eject upon the first reference to LIST.

2.11 PAGE

2.11.1 This location contains the current page count in display code in the low order 18 bits. The initial contents of this location are "GE,bNO.bb0". To reset the page count, it is necessary to store a word of binary zeros into PAGE.

3.0 No diagnostics are produced by SIO.

4.0 Structure

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. D-4
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

5.0 Major Subroutines

5.1 RDLIMIT

This routine returns the contiguous buffer space available for use in B2. If X0 is ≠0, the total amount of unused buffer is returned. The address of OUT is returned in X2. X1 contains the address of the FET upon entry. If the buffer is empty and the last operation did not return an end-of-record or end-of-file status, a read will be initiated and a RECALL request will be made. If the buffer is empty and the end-of-record status is on, the status is returned to the user and cleared in the FET.

5.2 MVEXIT

This routine checks if the buffer is half empty and request another read if it is. This routine is entered after each read request.

5.3 WRLIMIT

Similar to RDLIMIT except it is used for finding the buffer limits for write requests.

5.4 MVWDS

This routine is used to move words to and from the I/O buffer. The destination address is in B7, the origin address is X2 and the number of words in B2.

5.5 FRMCAL

This routine is used to format File Manager requests (CIO). The FET address is in X1, the parameter is X4 and the return address is in B6.

5.6 RCL1

This routine makes all requests for RECALL. The return address is in B6.

5.7 CKSTAT

This routine is called to determine the status of a buffer. The reply is in X4.

X4	LT	-0	buffer busy
X4	EQ	-0	buffer static
X4	GT	+0	EOF encountered
X4	EQ	+0	EOR encountered

CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
DIVISION

DOCUMENT CLASS IMS PAGE NO. D-5
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

6.0 Formats

6.1 FET Table

Word 1	bits 0-6 bits 18-60	buffer status file name, zero fill
Word 2	bits 0-18	address of FIRST
Word 3	bits 0-18	address of IN
Word 4	bits 0-18	address of OUT
Word 5	bits 0-18	address of LIMIT
Word 6	bits 0-18	half buffer size
Word 7	bit 59	first reference flag
Word 8	not used	
Word 9	not used	
Word 10	not used	
Word 11	not used	

6.2 ODDFLG

Used by MVWDS to signify an odd number of words to be moved.

6.3 FWA1, LWA1

Location of card image in INPUT buffer, needed by FRDCARD in SCANNER.

6.4 SVWDCNT

Saved word count when requests are made via LIST.

6.5 SVADDR

Saved fwa when requests are made via LIST.

7.0 Modification Facilities

7.1 IMAX

May be changed by an EQU to modify the number of lines per page on the listing.

DOCUMENT CLASS IMS PAGE NO. D-6
PRODUCT NAME FORTRAN Extended
PRODUCT MODEL NO. 4P616 MACHINE SERIES 64/65/6600

7.2 I/O Files

The name of the I/O files (left justified zero filled) must be placed in RA+2 thru RA+n before an OPEN request is made. The logical unit numbers are related to the name in the RA area as the name in RA+n is the file used for logical unit n-1. If the name OUTPUT appears in RA+3 any reference to file number 2 will result in a reference to the OUTPUT file.