HONEYWELL EDP

APPLICATION HANDBOOK

SERIES 200 USER'S LIBRARY

EXTENDED CONTROL AND SIMULATION LANGUAGE COMPILER

GENERAL SYSTEM: SERIES 200/APPLICATIONS SYSTEM

SUBJECT: Extended C.S.L. — A Language and Compiler System Used to Describe and Formulate Problems Involving Complex Logic for Simulation on a Series 200 Compiler.

SPECIAL INSTRUCTIONS: Extended C.S.L. is included in Honeywell's Series 200 Users' Library. Details of the operation of the library and the procedure for requesting programs can be found in the General Bulletin titled Submittal and Request Procedures for Users' Library, Order No. 217.

DATE: September 15, 1966

FILE NO.: 134.8405.0000.0-524*

9257
31066
Printed in U.S.A.

*Underscoring denotes Order Number.

CONTENTS

INTRODUCTION

Extended C.S.L. is intended to assist in the formulation and description of problems which consist of complicated logic. It will be of great use in the simulation of industrial and commercial systems on a computer.

The language is based on the use of groups of entities, which are the elements of the system, and in particular sets of entities which have some common property. The most powerful feature of this language is its ability to perform set arithmetic rapidly i.e. testing and operating on the members of a set.

The system also incorporates methods of sampling from standard and empirical distributions, and for the statistical presentation of results in the form of histograms. Also incorporated is a built-in time mechanism to assist simulation.

The extended C.S.L. compiler is designed for the Honeywell Series 200 computer and translates the simulation program directly into machine code.

A sample program including an explanation is present in Appendix A.

N.B: Words underlined in this text represent Extended C.S.L. structural words.

SECTION 1

Definition of Terms in Extended C.S.L.

Name: An item in C.S.L. (eg. cell, array) is identified by a sequence of any number of letters. The compiler, however only recognises the first six letters; so care must be taken to avoid ambiguity (eg. TRANSPORT, TRANSPOSE). Any sequence of letters (numeric and special characters are not allowed) can be used for a name except those reserved for the structural words in the Extended C.S.L. language.

Cell: A single storage location which is addressed by a name. The contents of a cell are assumed to be integer values unless otherwise specified.

Array: A multi-dimensional array of cells may be used for the storage of data. Each array can be identified by a name eg. GROUP and each cell in the array can be referenced by the name followed by a list of subscripts appropriate to the dimension of the array, in parenthesis. Eg. GROUP (9,A) is the 9th by Ath cell (element) of a 2-dimensional array.

Distribution: A distribution is a special form of array which is used for random sampling and/or the accumulation of results. This array is in the form of a frequency distribution.

Entity: An entity is an object

Class: A class is a group of entities which is identified by a name eg. SHIPS.

Entity Name: An entity is referenced by a name, which consists of the class name followed by a class index.

eg. SHIPS I
SHIPS Y
SHIPS 1

where Y & I have integer values.

The above all reference particular ships within the class SHIPS

Index: An index is an integer, a cell or an expression in parenthesis, and must always have a positive integer value.

Set: A set is a device for holding the names of entities of a specific class. It may be addressed by a name and optionally an index.

eg. 1. ATSEA could be a set holding entities from a class SHIPS which are at present at sea.

2. WORKERS 2 could be a set holding entities from a class STAFF which are at present employed on a certain job.

The entities recorded in a set must all be from one specific class, but the members of a class may be recorded in several sets. The set holds a list of entity names. This list is ordered and may therefore be used to represent a queue. Eg. if UNLDD is a set holding the members of class TRUCK which are waiting to be loaded, then the first entity in UNLDD represents the truck waiting longest etc. Advanced list processing methods are used in Extended C.S.L. to manipulate sets.

T-Cell: T-Cells are used to hold time values for simulation purposes. A T-Cell may be associated with each entity of a class and addressed as T.THING 6, for example. In addition, T-Cells not associated with entities may be used and given names preceded by T or T..

eg. T LATE T. LOST

Expression: This term is used to denote a collection of variables, brackets and operation symbols which form a meaningful single valued arithmetical expression (see Section 3).

Variable: A single location addressed as a cell, T-Cell, or a member of an array. Unless the name of the variable is mentioned in a FLOAT, BOOLEAN or STRING statement (See Section 2) it will have an integer value. The capacity of integer and floating point cells is as defined under Constant, below.

Label: A number between 1 and 99999, which is used to identify a statement, is known as a label. This number can be written anywhere in the first five character positions of an extended C.S.L. statement.

Constant: A constant is a string of digits, possibly including a decimal point. It will be assumed to be an integer unless a decimal point is included. The decimal may not be the first character of the string. In this manual n_1, n_2 etc. are used to indicate obligatory constants. The range of values for integer constants is from -9999 to +9999, although integer variables can range from -4194304 to +4194303. For floating point constants the argument is held to an accuracy of 7 decimal digits and the exponent can have any value between - 32 and + 32 (base 10)

Destination Clause: Many statements in Extended C.S.L. involve the carrying out of an explicit or implied test. Depending on the result of the test, control may be passed to another part of the same sector of program. These test statements may be terminated by a destination clause of the following form:-

@ L - where L is the label of a statement in the program sector;

A successful test will result in a transfer of control to the next statement in the program. If the test fails control is transferred to the statement labelled L.

The destination clause may start in the first position following the statement proper, or it may be separated from it by one or more blanks.

N.B. 1. If no destination clause is given with a test statement, then control passes to the first statement of the next sector (i.e. to the next BEGIN statement) if the test results in failure.

2. In all description of statements in this manual the @ L form will be used.

Subscript: Up to three subscripts may be used for addressing array elements. eg.

```
LENGTHS  (5)
LIST     (I,J)
ELEMENTS (A, B, C,)
```

A subscript may itself be an expression, and so may itself include a subscripted variable.

eg. LIST (ALLOC(I) *2, 3*X(J) + Y(J))

SECTION 2

Definition Statements

CLASS

The CLASS statement defines a class of entities together with any associated T-cells and sets and must precede all mention of the entities and sets. The general form of this statement is as follows:-

CLASS TIME $name_1$,n_1, SET $name_2$, $name_3$, n_2, $name_4$ etc., where $name_1$, - - -, - - -, $name_4$ are Extended C.S.L. names and n_1, n_2, are unsigned integer constants.

n_1 must be less than or equal to 2047

n_2 must be less than or equal to 4095

The above statement defines a class of n_1 entities which is identified by the name '$name_1$'. Each entity has a T-cell (only if the word TIME appears) and the entity names may be used in any statement associated with sets.

If the word SET appears followed by a list of names, then sets known by these names are defined. The general form of specifying a set in the list is:- $name_k$, n_1.

Here '$name_k$' is the name of the set. n_1, if it appears, indicates that n_1 sets with the names '$name_k1$', '$name_k2$', upto '$name_kn_1$' are to be defined.

Dots and commas in the statement are optional and initially all sets are empty.

Examples:-

```
1  CLASS WAREHSS 12

2  CLASS TIME SHIPS. 15 SET ATSEA, BERTH2

3  CLASS WORKERS 50 SET ONSTRIKE, SHIFTS 3, ABSENT
```

The second of the above examples defines a class of 15 entities SHIPS 1, SHIPS 2 up to SHIPS 15. Associated with each of these entities is a T-cell T.SHIP 1,T.SHIP 2 up to T.SHIP 15. The entities of the class can be placed in sets ATSEA, BERTH 1 and BERTH 2.

ARRAY

The ARRAY statement is used to define one or more arrays of numbers. This definition must precede all mention of the cells of the arrays. The general form of the ARRAY statement is:-

ARRAY $name_1$ (n_1, n_2), $name_2$ (n_3, n_4, n_5), $name_3$ (n_6) - -

where $name_1$, $name_2$, $name_3$ are Extended C.S.L. names and n_1, n_2, n_3, n_4, n_5, and n_6 are unsigned integer constants, less than or equal to 1023, or names of classes.

The above statement defines 3 arrays '$name_1$' with 2 dimensions and n_1 x n_2 cells, '$name_2$' with 3 dimensions and n_3 x n_4 x n_5 cells and '$name_5$' with n_6 cells. n_1, n_2, etc. specify the maximum value of the array subscript and if a class name is used then the maximum value of the subscript is the number of entities in that class. The total number of cells in an array must be less than or equal to 1023. All commas in the statement are optional.

Examples:-

1 CLASS DRIVERS 20

2 CLASS TIME TRUCKS 30

3 ARRAY CHESSBOARD (8,8), RESISTANCE (20)

4 ARRAY GARAGE (TRUCKS), MILEAGE (DRIVERS, TRUCKS)

Example 3 defines an 8 by 8 cell array CHESSBOARD and a 2Ø cell array RESISTANCE.

Example 4 defines a 30 cell array GARAGE and a 20 by 30 cell array MILEAGE.

N.B. Single cell and isolated T-cells are defined by their appearance in a statement.

FLOAT

The general form of this statement is:-

FLOAT $name_1$, $name_2$, $name_3$, - - - -, - - - -, - - - -,

where $name_1$, $name_2$, $name_3$, are cell or function names.

The cells mentioned in this statement will hold floating point numbers and can be used for floating point calculations. The functions mentioned will all present their results in floating point forms. This statement must precede all mention of the cell or function names.

FLOAT ARRAY

The general form of this statement is :-

FLOAT ARRAY $name_1$ (n_1, n_2) $name_2$ (n_3) $name_3$ (n_4, n_5, n_6,), - - - - - -. Here the arrays are defined as for the ARRAY statement but each cell will hold a floating point number and can be used in floating point calculations.

BOOLEAN

The general form of this statement is :-

BOOLEAN $name_1$ n_1, $name_2$ n_2, $name_3$ n_3 - - - -

where $name_1$, $name_2$ and $name_3$ are Extended C.S.L. names and n_1, n_2, n_3, if present are the number of elements in the vector, n_1, n_2, and n_3 must be less than 2048 and if they are not used the number of elements in the vector will be assumed to be 1.

STRING

The general form of this statement is :-

STRING $name_1$, n_1, $name_2$, n_2, $name_3$, n_3, - - - - -

where $name_1$, $name_2$ and $name_3$ are Extended C.S.L. names and n_1, n_2, and n_3, if present are the number of characters in the string. n_1, n_2, and n_3, must be less than 4096 and if they are not present the number of characters in the string will be assumed to be 1.

DIST

The DIST statement is used to define frequency distributions which may be used to sample and/or to accumulate data compiled during the execution of program (See Section 8). The general form of this statement is :-

DIST $name_1$ n_1 (n_2, n_3, n_4), $name_2$ n_5 (n_6, n_7, n_8), - - - - -

where $name_1$ and $name_2$ are Extended C.S.L. names.

n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8 are unsigned integer constants.

n_1, n_2 and n_5, n_6 must be less than 1024.

n_3, n_4 and n_7, n_8 are to all intents unlimited (i.e. 9999)

This statement defines groups of distributions $name_1$ 1, $name_1$ 2 upto $name_1$ n_1 and $name_2$ 1, $name_2$ 2 upto $name_2$ n_2. If n_1 or n_2 are omitted then a single distribution $name_1$ or $name_2$ is defined. If the first group of distributions is considered then each of these distributions has n_2 cells, n_3 is value associated with the first cell and n_4 is the incremental value between cells.

Distributions are cleared on definition.

HIST

This is a synonym for DIST

Examples :-

```
1  DIST ARRIVALTIMES (30,1,1), SHIFTPROD 3 (12,1,1)
2  HIST ACCIDENTS (150,5,5)
```

Example 1 defines 4 distributions:- ARRIVALTIMES, SHIFTPROD 1, SHIFTPROD 2 and SHIFTPROD 3 and example 2 defines one distribution ACCIDENTS.
ACCIDENTS has 150 cells whose associated values will be 5,10,--,-- --, 750.

IS

The IS statement enables two names to reference the same item such as an array, cell etc. The general form is:-

$name_1$ IS $name_2$, $name_3$, and $name_4$, are Extended C.S.L. names. This statement causes $name_2$ to be regarded as synonymous with $name_1$ and $name_4$ to be regarded as synonmous with $name_3$. $name_1$, $name_3$, etc. may be constants, in which case the apprearance of names, i.e. $name_2$, $name_4$ etc. in the program are taken to be the relevant constant, not a variable.

CLASS

This is the second form of the CLASS statement and its general form is:-

CLASS $name_1$ ($name_2$ n_2, $name_3$ n_3, - - -, - - -)
where $name_1$, $name_2$, and $name_3$ are Extended C.S.L. names and n_2 and n_3 are unsigned integer constants.

This statement defines a main class '$name_1$' which is broken into subclasses $name_2$, $name_3$, - - -. The first entity of the first subclass $name_2$ 1 is the same as the first entity of the main class i.e. $name_1$ 1 and the first entity of the second subclass i.e. $name_3$ 1 is the same as the $(n_2 + 1)^{th.}$ entity of the main class i.e. $name_1$ $(n_2 + 1)$. The name of a subclass may be omitted, in which case the unsigned constant simply displaces the next subclass by that number of entities. The number of entities in the main class must not exceed 2047.

The word TIME may be used in this statement as for the normal CLASS statement but sets must not be defined.

Example:-

1 CLASS MACHINES (7, TRDIG 3, 10, TRHARROW 2, TRGART 8, 20).

This statement defines a class MACHINES with 50 entities, 3 of which belong to a subclass TRDIG, 2 to a subclass TRHARROW and 8 to a subclass TRCART. 37 entities in class MACHINES have no other name.

CLASS	SUBCLASS
MACHINES 1 - MACHINES 7	none
MACHINES 8 - MACHINES 10	TRDIG 1 - TRDIG 3
MACHINES 11- MACHINES 20	none
MACHINES 21- MACHINES 22	TRHARROW 1 - TRHARROW 2
MACHINES 23- MACHINES 30	TRCART 1 - TRCART 8
MACHINES 31- MACHINES 50	none

SET

The general form of this statement is:-

$name_1$ SET $name_2$ n_1, $name_3$ n_3

where $name_1$ is a class or subclass name

$name_2$, $name_3$ are Extended C.S.L. names

n_1, n_3 are unsigned integers.

The above statement is used as for the SET section of the CLASS statements. Two groups of sets $name_2$ and $name_3$ are defined, each is large enough to hold all members of class/subclass $name_1$.

Example:-

1. TRCART SET REPAIR, INFIELDS 8

This example defines nine sets REPAIR, INFIELDS 1 - - - up to INFIELDS 8. If considered in conjunction with example 1 of the CLASS statement above, then each set can hold 8 entities from TRCART.1 up to TRCART.8 i.e. 8 entities from MACHINES.23 up to MACHINES.30.

SECTION 3

ARITHMETIC EXPRESSIONS AND STATEMENTS.

Expressions

An expression is any meaningful combination of constants and/or variables, separated by operation symbols, which is used to define the procedure for calculating a value. The permissable operation symbols are :-

+ - addition

- - subtraction

* - multiplication

/ - division

** - exponentiation (N.B. The combination of two asterisks is treated as one symbol).

Example :-

A ** 3 + 3ØØ * EL(I,J) /ABC

A valid expression will be ensured if the following points are respected :-

1. Any fixed point or floating point constants, variable, entity cell, T-cell or array element is itself considered to be an expression.

Examples :-

1. 17
2. X
3. TRUCK. 9
4. T. SHIP 5
5. MATRIX (I,J)

2. If any two quantities, separated by an operational symbol, are of different mode then the resulting value will be of floating point mode.

3. A function (see Section 15) is an expression

Example:- SQRT (TIMA)

4. Parenthesis are used to avoid ambiguity as in normal mathematical practice.

Example:- Y*(T.XYZ + 5)

The expressions in parenthesis are always evaluated first. To avoid further ambiguity the following rules should be respected:-

1. In the absence of parenthesis the order of performance is :-

a) Exponentiation.

b) Multiplication/Division

c) Addition/Subtraction.

Example:- A + C/D-E**3 is equivalent to:- A+(C/D)-(E**3)

2. When an expression consists entirely of operation symbols on the same hierarchical level then evaluation will procede from left to right.

Example:-

7/Y*I is equivalent to

$$\frac{7}{Y} * I$$

Arithmetic Statements.

Assignment Statement.

The general form of this statement is :-

Variable = Expression

and it is used to give the named variable the evaluated result of the expression. The left hand side may be a cell, T-cell or an array element. The expression must obey the rules given in the first part of this section. The result obtained from the expression is converted to the mode of the variable and then overwrites the previous contents of the variable.

Examples:-

1. AVTIME = (T. CELLA - T. CELLB) /2
2. J = 1
3. A(I,J) = G* (A+C/D - E**3)
4. L = (J = 1) * (K - 7)
5. N = Z + SQRT (A(I,J) + L)

Incremental Statement.

The general form of this statement is :-

a) variable + expression

b) variable - expression.

The previous definition of variable and expression apply to the above statement. In the above statements the expression is evaluated, the result is converted to the mode of the variable and added to or subtracted from the current value of the variable.

Examples :-

1. J + 1
2. T. SHIP. 5 + 7 * STIME
3. X + ((Z - 3) + EL (12) **2)

N.B. In cases where a floating point number is converted to fixed point, e.g. when the result of a mixed mode expression is to be placed in a fixed point variable or when a floating subscript is used, the value taken will be the largest integer less than or equal to the floating point value.

FOR Statement.

The execution of a group of statements may be repeated under the control of the FOR statement. The group of statements to be repeated must have an equal indentation, this level of indentation being further to the right than the beginning of the FOR statement.

This group of statements to be repeated under the control of the FOR statement is known as the FOR loop.

There are two general forms of the FOR statement :-

a) FOR $name_1 = m_1, m_2, m_3$

xxxx

xxxx

b) FOR $name_1 = name_2$

xxxx

xxxx

xxxx

Where $name_1$ is a cell name (it may be a member of an array), known as an index, and m_1 and m_2 are either unsigned integers, cell names or expressions, and $name_2$ is either a class name or a set name. All arguments must have integral values.

m_3 can be a cell name or expression or a signed integer, i.e.

+n is for an incrementing loop and

-n is for a decrementing loop.

(N.B. - For a decrementing loop the value of m_2 must never be zero or less).

a) For this version of the FOR statement the first loop is executed with the index '$name_1$' = m_1, thereafter m_1 is increased by m_3 (if m_3 is not specified it is taken as 1) and the loop repeated with '$name_1$' = the new value of m_1. This process is discontinued when the next addition of m_3 to m_1 would result in a value greater than m_2. When this condition arises control passes to the statement after the FOR loop.

The range of a FOR loop is defined as the group of statements with indentation further to the right than the FOR statement.
The index is available throughout the range of the FOR loop and can be used as a variable in either a normal or a subscript expression. If it is not used it will act simply as a control counter for the FOR loop.

Examples:-

```
1.  FOR I = 1,20
        T. SHIP I = 0
```

This example will zeroize the T-cells associated with T.SHIP 1 up to T. SHIP 20.

```
2.  ASVM=ø
    FOR J = 1,K,2
        GRID (L,J) = 3 * POT (J)
        ASVM + GRID (L,J)
```

Here all the odd elements of the Lth row of GRID (up to the nearest odd number less than or equal to K) are set equal to three times the corresponding odd elements of the vector POT. A sum of the values of the elements of GRID involved in this loop is formed in the variable ASVM.

b) In the second form of the FOR statement, if 'name$_2$' is a class name then repetition over the indented range is performed with index, 'name$_1$' = 1, thereafter being increased by 1 until its value reaches the number of entities in the class. The = sign is optional.

Example:-

```
FOR I = SHIPS
 T. SHIPS.I = ø
```

Here the time cells associated with all the ships in class SHIPS are zeroized (cf. Example 1 of a)

If in the second form, 'name$_2$' is a set name then repetition takes place over the indented range as many times (including zero) as there are members in the set 'name$_2$'. The index 'name$_1$' takes in turn the value of the class index of each entity in the set. Again the use of the = sign is optional.

Example:-

```
1.  FOR W = QUEUE
        T. MAN W = 0
```

Here the time cells of any MAN at present in the QUEUE are zeroized.

The following points should be noted in relation to FOR loops

1. FOR loops may occur within FOR loops.

```
e.g.  FOR I = 1,50
          FOR J = 1,50
              L (I,J) = 0
          SUM + N (I)
```

This process is known as nesting and nesting can occur to a depth of 6 loops. In the above example the outer loop is executed 50 times and the inner loop is executed 50 times i.e. 2500 loops are performed. At each repetition of the outer loop, all the elements of the Ith row of matrix are zeroized (this is performed by the inner loop) and a sum is kept in SUM of the Ith elements of the one dimensional array N.

2. There must be no transfer of control into the range of a FOR loop from outside the loop and the last statement of a FOR loop must not itself be a transfer of control statement (GOTO). To avoid the latter condition three special statements are available which do nothing but enable the programmer to avoid ending the loop in a transfer of control.

They are DUMMY

synonymous to REPEAT

synonymous to CONTINUE

3. No statement in the range of the FOR loop may alter the values of its index or controlling parameters.

Test Statements.

Tests between quantities can be performed according to the following six relational-operators:-

1. LT less than
2. LE less than or equal to
3. GT greater than
4. GE greater than or equal to
5. EQ equal to
6. NE not equal to

The general form of this statement is :-

'name$_1$', ro name$_2$' @ L

where name$_1$ and name$_2$ can be a cell or T-Cell, a constant or an expression & ro is one of the six above relational operators.

If the statement is true then control passes to the next statement, otherwise the test fails and control passes to the statement labelled L in that sector (or to the first statement of the next sector, if @ L is omitted).

Examples:-

1. T.SHIP. 5 EQ Ø

Control will pass to the next program sector if the time-cell for SHIP. 5 is not zero.

2. A GT B

3. T. MAN . 1 + 67 LT 12Ø

4. X NE Y + 5Ø

Transfer Statement.

The following allows unconditional transfer of control and its general form is :-

GOTO L

where L is a statement label within the same sector.

Example

1. GOTO 777

this will cause a transfer of control to statement 777

SECTION 4

SET ARITHMETIC

The following statements are provided to facilitate the manipulations of sets. In all the general forms of statements in this section "setname" should be taken to mean an Extended C.S.L. name which is defined as a set. This name may or may not be qualified by a suffix. 'Entityname' is an Extended C.S.L. entity name.

GAINS

The general form of this statement is:-

"setname$_1$" GAINS 'setname$_2$'

This statement causes the members of 'setname$_2$' which are not already in 'setname$_1$' to be added to the end of 'setname$_1$'.

Examples:-

1. FLEET GAINS REPAIRED

Here any member of REPAIRED not already in FLEET is added to FLEET.

2. QUEUE 1 GAINS ARRIVALS

LOSES

The general form of the statement is:-

'setname$_1$' LOSES 'setname$_2$'

This statement causes the members of 'setname$_1$' which are also in 'setname$_2$' to be removed from 'setname$_1$'.

Examples:-

1. QUEUE 2 LOSES SERVICED

Here the vehicles which have been SERVICED and are still registered in QUEUE 2 are removed from QUEUE 2.

2. STAFF LOSES STRIKERS

LOAD

The general form of this statement is :-

'name$_1$n' LOAD 'setname$_1$', 'setname$_2$', ...

where 'name$_1$' is a class name (i.e. the class for which the sets are defined) and 'n' is an integer, variable name or expression in parenthesis.

This statement causes 'setname$_1$' and 'setname$_2$' to be emptied and then loaded with entities 'name$_1$.1','name$_1$.2' up to 'name$_1$. n' in that order. (N.B. No set may have a capacity less than 'n').

Examples:-

1. VEHICLES. 8 LOAD QUEUE 1, SERVICE

This example causes VEHICLES. 1, VEHICLES. 2, up to VEHICLES. 8 to be loaded, in that order, into sets QUEUE 1 and SERVICE.

2. SHIPS . 15 LOAD ATSEA

ZERO

The general form of this statement is :-

ZERO 'setname$_1$', 'setname$_2$', ...

This causes the named sets to be emptied.

Example:-

1. ZERO RECRUITS, STAFF 1, STAFF 2, APPLICATIONS

The following set statements allow tests to be performed on sets.

IN

The general form of this statement is :-

'entity name' IN 'setname' @ L

The above statement tests if 'entity name' is in the set 'setname'. If it is, the test succeeds and control passes to the next statement, otherwise control passes to the statement labelled L in the same sector. (If @ L is omitted and the test fails then control passes to the first statement of the next sector).

Examples:-

1. MAN. I IN OVERTIME @ 55

This statement causes control to pass to the next statement if and only if MAN. I is in the set OVERTIME, otherwise control passes to statement labelled 55.

2. ITEM. 5 IN SECTION 3

3. ELEMENT (I*J) IN VECTOR 6 @ 77

NOTIN

The general form of this statement is :-

'entity name' NOTIN 'setname' @ L

This is the reverse of the IN test, and the test succeeds if the entity is not present in the set.

Example:-

1. MAN. 6 NOTIN SHIFT 1 @ 232

Here, control passes to the next statement if MAN. 6 is not present in the set SHIFT1, otherwise control passes to statement 232.

2. SPARE. 2 NOTIN STORE

Here, if SPARE.2 is present in set STORE, then control will pass to the first statement of the next sector.

3. DEFINITION. 7 NOTIN APPENDAB

EQUALS

The general form of this statement is :-

'setname$_1$', EQUALS 'setname$_2$' @ L

The above test succeeds only if the membership of both sets is identical, otherwise control passes to the statement labelled L in the sector (only if @ L is present, otherwise as before). No significance is given to the ordering of members within the sets.

Examples:-

1. TOTAL EQUALS CHECK

Control will pass the next sector if the set TOTAL does not have an identical membership to the set CHECK.

2. CHOCBOX 3 EQUALS VARIETIES @ 696

WITHIN

The general form of this statement is :-

'setname$_1$' WITHIN 'setname$_2$' @ L

This test succeeds only if all entities within 'setname$_1$' are also present in 'setname$_2$' otherwise the test fails. No significance is given to ordering of entities within the sets.

Examples:-

1. LABFORCE WITHIN EMPLOYEES @ L

Control will pass to the next statement if all the entities within set LABFORCE are also recorded in set EMPLOYEES, otherwise control passes to statement labelled L

2. TUGS WITHIN FLEET

DISJOINT

This is logical opposite of WITHIN, i.e. success results if named sets have no common members.

EMPTY

The general form of this statement is :-

'setname$_1$', 'setname$_2$', ... EMPTY @ L

This test results in success if all the sets listed before EMPTY are empty. If at least one entity name is present in any one of the sets then the test fails and control passes to statement labelled L (if @ L is present).

Examples:-

1. LIST EMPTY

If there are any entities in the set LIST then control passes to the next sector of the program, otherwise the next statement in sequence is processed.

2. (QUEUE K) K=4, 8 EMPTY @ 12

This example shows the use of the implied list i.e. only if QUEUE 4, QUEUE 5, up to QUEUE 8 are all empty will the test succeed.

3. GROUP, SHIFT, STAFF EMPTY @ 44

The following statements combine a set operation statement and set test statement to form a compound statement.

HEAD

The general form of this statement is :-

'entityname' HEAD 'setname' @ L

In this statement an implied test is carried out to determine if the entity 'entityname' is a member of set 'setname'. If it is then the test fails and control passes to statement labelled L (or to next sector if @ L is omitted). Otherwise the entity is added as the first member of the set 'setname' and control passes to the next statement.

Examples:-

1. SHIP. I HEAD QUEUE 3 @ 200

If SHIP. I is not at present in set QUEUE 3 then it is placed at the head of QUEUE and control passes to the next statement. Otherwise control is transferred to statement labelled 200.

2. CAR. 3 HEAD FERRY Q

3. ITEM (NUMBER **2) HEAD ELEMENTS @ 70

INTO

TAIL

TAIL and INTO are synonymous. The general form of the statement is:-

'entityname' TAIL 'setname' @ L

If the entity 'entityname' is in the set 'setname' the test fails and control passes to the statement labelled L (or to the next sector if @ L is omitted); otherwise the entity 'entityname' is added to the end of the members in set 'setname' and control passes to the next statement.

Examples:-

1. MAN. L INTO QUEUE @ 4

If MAN. L is not in the set QUEUE then he is added to the end of the set and control passes to the next instruction. Otherwise set QUEUE is left unaltered and control passes to the statement numbered 4.

2. LATEARRIV 6 TAIL SERVICE

FROM

The general form of this statement is:-

"entityname" FROM "setname" @ L

In this statement the implied test fails if the entity "entityname" is not present in the set "setname" and control is transferred to the statement labelled L (or to the next sector if @ L is omitted). Otherwise the test will succeed and the entity "entityname" is removed from the set "setname" and control passes to the next statement.

Examples:-

1. TRUCK K FROM SERVICED @ 767

If TRUCK K is at present in SERVICED and control is transferred to statement labelled 767.

2. PARA. 3 FROM PAGE 7

Compounding & Listing

1. EMPTY, WITHIN, LOSES & GAINS may be used together to form a compound statement. So also may IN, NOTIN, HEAD, INTO and FROM

2. Set names occurring after the keyword may be listed.

Examples of 1. CARS FROM GARAGE INTO USED @ 222

This test succeeds if CAR. 8 is in set GARAGE but not in set USED. Under this condition CAR, 8 will be removed from GARAGE and placed in USED and control will be transferred to next instruction. Otherwise the test will fail the sets will be unaltered and control will pass to statement labelled 222.

2. SHIP.2 NOTIN ATLANTIC, PACIFIC, INDIAN @ 8

The above test will succeed if SHIP. 2. is in none of the sets ATLANTIC, PACIFIC and INDIAN and control will pass to the next statement. If however SHIP. 2 is a member of at least one of the above sets then the test fails and control is transferred to statement labelled 8 within that sector.

3. This example is show in Fig. 4.1 to illustrate the layout of statements on the coding sheet.

a) Columns 1 to 5 are used to contain the statement label

b) The statement proper is written in columns 7 to 72

c) Columns 73-80 are used for card identification and sequencing.

d) Column 6 is used to denote continuation lines of a statement too large for one line (never split variable names over two lines)
e.g. if a statement needs 3 lines of the coding sheet, then column 6 for the first line could contain $\emptyset$ (or be left blank) and the succeeding column 6's should contain some digit say 1 and 2.

e) Column 1, if it contains a 'C' denotes that any characters on this line are to be ignored by the compiler but will be produced in any listings of the program i.e. comments.

```
C     THIS EXAMPLE CONTAINS A RANDOM SELECTION OF STATEMENTS

     ØCLASS TIME EMPLOYEE 2Ø SET QUEUING SERVED ONBREAK DELIVERING
     1ARRIVED
      FLOAT ARRAY WAGES (2Ø)
      FOR I=1,2Ø
         WAGE(I)=Ø
      GOTO 77
      - -  - -
   77 FOR X=EMPLOYEE
         EMPLOYEE.X TAIL QUEUING
      - -  - -
      SUM=Ø
      FOR X=1,2Ø
       EMPLOYEE.X FROM QUEUING INTO SERVED
       SUM+SAMPLETIME
       T.MAN X+SUM
       WAGE(X)+(1.4*SUM)
       QUEUING EMPTY @ 6
       CONTINUE
```

FIG 4.1

Boolean Algebra

Boolean variables can be created using the BOOLEAN statement given in Section 2. They can be given values TRUE or FALSE by having values read into them or they can be given values by the assignment statement which follows.

Assignment Statement

The general form of this statement is:-

$name_1$ = $name_2$

where $name_1$ is a BOOLEAN variable

and $name_2$ is either a BOOLEAN constant (TRUE or FALSE) or another BOOLEAN variable (or expression, see later)

The BOOLEAN variable '$name_2$' is given the value TRUE or FALSE depending on the value of the BOOLEAN constant or variable '$name_2$'

Example:- TABLE (6) = TRUE

Here the BOOLEAN element TABLE (6) is assigned the value TRUE

BOOLEAN expressions

There are two BOOLEAN operators
1. + = this performs a BOOLEAN OR
2. * = this performs a BOOLEAN AND
These can be used to form a BOOLEAN expression.

Example:-

1. A = B + C

There, if A,B and C are BOOLEAN variables, then if either B or C is TRUE A will be given the value TRUE. Otherwise A will be FALSE.

2. A = B * C

If B is TRUE and C is TRUE, then A will be given the value TRUE. Any other combination of B and C will result in A being FALSE.

Parenthesis can also be used in BOOLEAN expressions to determine priority.

3. A = (B * C) + D

If A, B, C & D are BOOLEAN variables then B * C will be evaluated (TRUE if both B & C TRUE, otherwise FALSE) and if either its value or that of D is TRUE then A will be given a value TRUE.

BOOLEAN 'IF'

The general form of this statement is:-

IF bexp @ L

where 'bexp' is a BOOLEAN expression and L is a statement label.

If the BOOLEAN expression is TRUE then control passes to the next statement otherwise control passes to statement labelled L (or to the next sector if @ L is omitted).

Example:-

IF STORY @ 99

If STORY is FALSE control passes to statement labelled 99.

BOOLEAN 'UNLESS'

The general form of this statement is:-

UNLESS bexp @ L

This is the reverse IF statement. It signifies that UNLESS the BOOLEAN expression is FALSE transfer control to statement labelled L.

Switch Statement

A special instruction is provided for testing the sense switches. Its general form is:-

a) SWITCH n OFF @ L

b) SWITCH n ON @ L

where n is a digit between 1 & 4 which represents a particular sense switch

and L is a statement label

a) This first version causes control to pass to statement labelled 'L' (or to the beginning of the next sector, if @ L is omitted) if sense switch 'n' is OFF

b) This version causes transfer of control if sense switch 'n' is ON

SECTION 5.

TEST CHAINS

A test chain is a group of test statements, which produce a single result of success or failure.

They are used to qualify the action and/or transfers which result from the complex test statements to be defined in Section 6. These statements will in general, be represented by 'Keyword' and would normally be followed by "@L" The test chain must contain at least one test and must be indented to the right of the 'Keyword'.

Simple Test Chain

The general form of a simple chain is:-

'Keyword' @L

test 1

test 2

test 3

— — — —

— — — —

— — — —

test n

where $test_1$ to $test_n$ are unlabelled tests of the following type

1. Arithmetic test (Page 3.5)
2. Set Test (Page 4.3 to 4.5)
3. Compound Set Test (Page 4.6 to 4.7)
4. Complex test (another test using a test chain)
5. Compound Find Test (see Later)

Success results in a test chain if all tests succeed but, if there is failure in any of the tests, then the test chain results in failure.

Examples:

1. 'Keyword' @ L

   A LT B

   B LT C

   C EQ D

   D GT (A + B)

Success results if A is less than B is less than C and if C and D are equal and greater than the sum of A & B.

2. 'Keyword' @ L

   T. SHIP X EQ ∅

   SHIP X IN ATSEA

Success results if SHIP.X is in set ATSEA and the T-cell associated with SHIP.X is equal to zero.

Complex Test Chain

The general form of this chain is:-

'Keyword' @ L

$test_1$

$test_2$

- - - -

- - - -

$test_n$

OR $test_{n+1}$

$test_{n+2}$

- - - -

- - - -

$test_m$

OR $test_{m+1}$

$test_{m+2}$

- - - -

- - - -

$test_n$

where $test_1$ up to $test_p$ are tests as for the simple test chain.

Here the test chain is composed of subchains or disjoints separated by the word OR.

i.e.

$Test_1$
....
....
$test_n$) 1st. disjoint

$test_{n+1}$
....
....
$test_m$) 2nd disjoint

$test_{m+1}$
....
....
$test_p$) 3rd disjoint

The complex test chain has again only one result, success or failure. Success is accomplished if any one of the disjoints produce success as defined for the simple test chain. If none of the disjoints succeed then failure results.

Examples:-

1.
```
'Keyword' @ L
        C GE 7ØØ
        B LT 5Ø
        A NE Ø
        OR D GT (B * B - 4 * A * A)
        OR A NE B
        B NE C
        C NE A
```

Success results if either a) C is greater than 7ØØ, B is less than 5Ø and A is not equal to zero.

OR b) D is greater than $B^2 - 4AC$

OR c) No two of A, B and C are the same.

a. 'Keyword' @ L

SHIP 6 IN UNLOADED, PORT 1

SHIP 6 NOTIN REPAIRS

OR SHIP 6 IN SEAAREA 1

T. SHIP. 6 LT 7

LOADC(6) LT 1125

Success results if either a) SHIP. 6 is unloaded in port but not being repaired.

or b) SHIP. 6 is in area 1, is less than 7 days from port and has a load less than 1125 tons.

Indentation

The range of any test chain is determined by the number of statements of equal indentation after the Keyword. (N.B. For a disjoint the word OR must start at the common level.)

Act Statements within chains.

Unlabelled acts (i.e. statements other than tests, transfers or compiler control statements) may be interspersed with tests in a test chain only if preceding tests succeed.

Example:-

1. 'Keyword' @ L

SHIP. 6 NOTIN REPAIRS

A = 1

OR SHIP. 6 NOTIN ATSEA

B = 1

If the above example resulted in success then by examining the value of A & B the test or tests, giving success could be determined.

FOR loops in chains

FOR loops (over unlabelled acts only) are also permissable in test chains

Example:-

```
1.  'Keyword' @ L
        A GT B
        FOR X = 1,10
            Y (X) NE ø
            J (X) EQ M (11,X)
        OR A LT B
        B NE C
```

N.B. The indented block in a for loop is, by implication, itself a test chain.

Nested Test Chains

Test Chains can be nested to a depth of 6 provided each inner chain has its own common level further to the right than the common level of the chain around it.

Example:-

```
'Keyword1' @ L
      A LT B
      OR A EQ D
      'Keyword2'
            C GT A * B
            C LT 1øøø
            OR 'Keyword3'
               E EQ 5ø
               OR F LE 9ø
             X NE ø
      X NE 1
```

Assuming that success results if the conditions demanded by the 'Keyword' are satisfied then the further success conditions are:-

a) i) (E equals 5Ø OR F less than 9Ø) and X does not equal Ø

or ii) C is less than 1ØØØ but greater than the product of A & B

AND b) A is less than B or (A equals D and X does not equal 1)

SECTION 6

COMPLEX STATEMENTS

COMPLEX TEST STATEMENT

The complex test statement is normally qualified by a test chain. If '@ L' is present in the statement then transfer of control is to statement labelled 'L' within that statement. Transfer of control takes place if the complex test results in failure.

CHAIN

The general form of this statement is:

CHAIN @ L

$test_1$

$test_2$

$test_3$

.

.

.

$test_n$

where $test_1$ up to $test_n$ are tests within a test chain

and L is a statement label (which may be omitted).

The result of the CHAIN statement is simply the result of the test chain qualifying it.

If the test chain fails then control passes to the statement labelled L (or to the first statement of the next sector if '@ L' is omitted). Success causes control to pass to the next statement following the test chain.

EXAMPLES:

1.

```
CHAIN @ 77
  A EQ 0
  I = 1
  OR NEG LT (B*B-4*A*C)
  II = 1
```

If either A is zero or if NEG is less than B^2 - 4AC then success results and control passes to the next statement. Otherwise control passes to the statement labelled 77.

2.

```
CHAIN @ 50
  (QUEUE I) I = 2, 8 EMPTY
  CHAIN
    (T.TRUCK. X) X = 1, 8 GT 5
    OR (TRUCK. X) K = 1, 8 NOTIN AREAA
```

If QUEUE 2, QUEUE 3, QUEUE 4, up to QUEUE 8 are empty and if either:

a) the T-cells associated with TRUCK 1 up to TRUCK 8 are greater than 5

or b) TRUCK. 1 up to TRUCK. 8 are not in set AREAA

then the test succeeds and control passes to the following statement. Under any other condition failure results and control passes to the statement labelled 50

Dummy Index

The following complex test statements use a cell (i.e: an unsubscripted variable). This variable acts as a dummy index, and takes in turn the value of the class index of each member of the set on which the statement operates.

If the complex test statement is qualified by a test chain, then the chain is performed once for each member of the set involved in the test and a record of the results is maintained. There should be at least one test in the test chain involving the dummy index, so that for each member of the set there is a corresponding test chain result.

ALL

The general form of the statement is:

ALL $name_1$ $name_2$ @ L

$test_1$

$test_2$

.

.

$test_n$

where $test_1$ up to $test_n$ is a test chain,

$name_1$ is a fixed point cell (dummy index),

$name_2$ is a setname,

and L is a statement label within same sector (may be omitted).

Success results if, for all members of set '$name_2$', the test chain results in success.

Example:

1. ALL X ATSEA @ 2Ø

SHIP X NOTIN AREA 1

OR T. SHIP. X GT 23

Control passes to the following statement if all ships in set ATSEA are not in set AREA 1 or if the T-cells associated with the ships in ATSEA are all less than 23. Otherwise control is transferred to the statement labelled 2Ø (Remember if '@ L' is omitted control would pass (if tests result in failure) to the first statement of the next program section).

2. ALL K STOCK @ 999

BOOK K NOTIN ONLOAN

BOOK K NOTIN DAMAGES

BOOK K NOTIN AVAILABLE

BOOK K TAIL MISSING

EXISTS

The general form of this statement is:

EXISTS (expression) $name_1$ $name_2$ @ L

$test_1$
.
.
.
$test_n$

where 'expression' has an integral result,

$name_1$ is the dummy index,

$name_2$ is a set name,

L is a statement label (which may be omitted)

and $test_1$ up to $test_n$ form the test chain.

Success results if the test chain is satisfied for at least as many members of set '$name_2$' as the value of the expression. If the number of members of set '$name_2$' satisfying the chain is less than the value of the expression failure results.

If the expression is omitted a value of 1 is assumed and if the test chain is omitted, the test is simply on the number of members at present in the set.

The expression must not involve the dummy index '$name_1$'

Examples:

1. EXISTS (TOTAL/2) K STAFF @ 9Ø

MAN. K IN ABSENT

If at least half the TOTAL membership of STAFF are IN ABSENT then the test 'succeeds' and control passes to the statement following the test, otherwise control passes to the statement labelled 9Ø.

2. EXISTS K LISTING @ 2

MEMBER K IN AREA 2

AGE (K) LT 45
CHAIN
MILEAGE (K) GT 5ØØØØ
OR OFFENCE (K) LT 2

Control passes to the statement following the test if at least one MEMBER from the set LISTING lives in AREA 2, is younger than 45 and has either: a) driven more than 5ØØØØ miles since joining the company or,

b) no more than one offence recorded against him since joining.

UNIQUE

This statement is identical to EXISTS but success only results if the number of members from the set 'name$_2$' is equal to the current value of the expression.

Examples:

1. UNIQUE (7) X QUEUE @ 3

MEMBER X IN GROUP 9

T. MEMBER X GT 5Ø

Control passes to the statement following the complex test if there are exactly 7 MEMBERS from QUEUE who also belong to GROUP 9 and whose T-cells contain a value greater than 5Ø. Otherwise control passes to statement labelled 3.

2. UNIQUE (2Ø) K ATSEA @ 2Ø

Control is transferred to statement labelled 2Ø if there are not exactly twenty members at present recorded in group ATSEA.

3. UNIQUE L TRANSPORT

TRUCK. L IN AVAILABLE

CAR. L IN AVAILABLE

OR TASK. 6 IN COMPLETED

Control is transferred to the first statement of the next sector if there is not exactly one TRUCK and one CAR IN set AVAILABLE and if TASK.6 is not IN set COMPLETED.

Incremental Indexing

Instead of the test chain being controlled by the members of the set mentioned in the complex test statement it can be controlled by any of the indexing forms available with the FOR statement. (See pg. 3.3)

Examples:

1. ALL I = 3, MAX @ 4Ø

MAN. I NOTIN ABSENT

ABTIME (I) LT 1Ø

Success results if for ALL I from 3 to the value of MAX (in steps of 1), MAN.I is NOTIN set ABSENT and ABTIME (I) is less than 1Ø. Otherwise control passes to statement labelled 4Ø

2. EXISTS (6* (L-J)) K = SETA @ 12

T. SHIP. K EQ Ø

The test is repeated with K taking successive values of the class indices of members of SETA. Success results if exactly 6 (L-J) members of SETA satisfy the given conditions that their T-cells are zero.

3. UNIQUE J = TRANSPORT

TRUCK. J IN STANDBY

The chain is repeated with J taking value from 1 up to (in steps of 1) the size of class TRANSPORT. If exactly one TRUCK is IN set STANDBY control passes to the statement following the test chain. Otherwise control is transferred to the first statement of the next sector.

COMPLEX ACT STATEMENTS

The following statements perform some action on a set and, in general are qualified by a test chain. A dummy index is used and is as defined in the complex test statement (see page 6.2)

COUNT

The general form of this statement is:

COUNT $name_1$ $name_2$
$test_1$
.
.
.
.
$test_n$

where $name_1$ is the dummy index

$name_2$ is the set

& $test_1$ up to $test_n$ form the test chain.

If the test chain is present a count is made of the members of set '$name_2$' which satisfy the test chain and a record of this number is available for use, in cell '$name_1$', after the act is performed.

If the test chain is omitted a count is produced in cell '$name_1$' of the membership of set '$name_2$'.

Examples:

1. COUNT K ATSEA

SHIP K. IN AREA 1

SHIP K. IN UNLOADED

CAPACITY (K) GE 5ØØØ

A count is produced in K of all SHIPS in ATSEA which are at present in AREA 1, are UNLOADED and have a CAPACITY greater or equal to 5ØØØ tons.

2. COUNT ISUMS QUEUING

Place the number of members at present in set QUEUING into cell ISUMS

SUM

The general form of this statement is:

SUM (Expression) incremental indexing

$test_1$

$test_2$

.
.
.
.

$test_n$

where 'Expression' has integer values

$test_1$ up to $test_n$ forms the test chain.

and 'incremental indexing' is of exactly the same form as for the FOR statement i.e. a) $name_1 = m_1, m_2, m_3$

b) $name_1 = name_2$ (see page 3.3)

The value of the expression is summed for each value of '$name_1$' which satisfied the test chain. The sum is left in I.

N.B.:

1. The expression must involve the index '$name_1$'
2. The index '$name_1$' should occur in the test chain, if present, so that for each value of '$name_1$' there is a corresponding result to the test chain.
3. If the test chain is omitted the expression is summed for all values of '$name_1$'.
4. If every value of '$name_1$' causes failure in the qualifying test chain then zero is left in '$name_1$'.

Examples:

1.
```
SUM  (6 * A (I)) I = 1,17
     A (I) NE  B(I)
     C (I) LT  5Ø
     T.CELL NE Ø
```

The expression is summed for every value of I (from 1 to 17 in steps of one) which satisfied the test chain. The sum is left in I.

2.
```
CLASS TIME MAN 15 SET ABSENT
SUM  (T. MAN. K)  K = MAN
     MAN K. NOTIN ABSENT
```

The values in the time cells associated with each entity in class MAN are summed provided that entity is not in set ABSENT. The sum is left in K.

3. SUM (CAPACITY (J)) J = DEPOT

TRUCK J NOTIN QSERVICE

The capacity of all lorries in the set DEPOT but not in set QSERVICE is summed and the result placed in J.

SECTION 7

FIND Statement.

The FIND statement involves an incremental statement which can be any of the forms available with the FOR statement (see page 3.3)

i.e. a) $name_1 = m_1, m_2, m_3$

b) $name_1 = name_2$

where $name_1$ = indexing variable (fixed point)

$name_2$ = set or class name

m_1, m_2, m_3 = integer constants or cells containing integer values.

The general form of the FIND Statement is:-

FIND (indexing as for FOR statement) Criterion @ L

$test_1$

$test_2$

$test_n$

where 'Criterion' is some condition to be satisfied (see below), $test_1$ up to $test_n$ form the test chain and L is the label of some statement in same sector. (@ L may be omitted)

This statement causes a subset of all values of '$name_1$' which result in success in the test chain to be formed. Then one of that subset is selected according to the criterion given and that value (i.e. class index of the entit from the subset is left in '$name_1$'.

If no subset can be formed (i.e. no value of '$name_1$' causes success in the test chain or if the set '$name_2$', (see (b) above is empty) then control is transferred to statement labelled L (or to the beginning of the next sector if @ L is omitted).

The criteria which can be used in the FIND statement are listed below (in each case assume the subset of values of 'name$_1$' which satisfy the test chain has been formed):-

1. ANY (stream)

Where 'stream' is a cell holding an initial value to be used for a random number generation (see later)

This criterion causes each member of the subset to have a random number generated and associated with it. The member associated with the highest random number is chosen to give in effect a random member of the subset. The value of this random member is left in 'name$_1$'.

2. FIRST

The first member of the subset is selected and its value placed in 'name$_1$'

3. LAST

The last member of the subset is selected and its value placed in 'name$_1$'.

4. MAX (expression)

The member of the subset which gives the expression its minimum value is chosen.

5. MIN (expression)

The member of the subset which gives the expression its minimum value is chosen.

N.B. In 4 & 5 the 'expression' must involve the index variable 'name$_1$'.

Example:-

```
1.      FIND J = 1, VAL, 2 MAX (A(J) **2) @ 2Ø
            A (J) GE Ø
```

All odd values of J from 1 up to but not greater than the value in VAL are selected.

A subset of the values which do not give a negative A(J) is formed. The value on that subset which gives the largest value to $A(J)^2$ is chosen and placed in J, and is available for use as a subscript or entity index after the FIND statement and qualifying chain. Note that the contents of J are unspecified if no member satisfies the test chain and therefore J has no meaning if used in statement 2Ø.

2. FIND I NOTWORKING ANY (START) @ 555

MAN.I NOTIN ABSENT

MAN.I NOTIN ONSTRIKE

A subset of all values of I (i.e. class indices of all members in set NOTWORKING), which satisfy the condition than MAN.I is neither in set ABSENT nor ONSTRIKE, is formed.

A value from this subset is chosen at random and placed in I and control passes to the statement following the test chain. If the set NOTWORKING is empty or if no member of that set satisfies the test chain, control passes to statement labelled 555 and the value of I is unspecified.

3. FIND K SHIPS FIRST

SHIPS K IN QUEUE

T. SHIPS K GT 1Ø

A subset of all values of K (indices of entities in class SHIPS) which satisfy the condition that SHIPS. K is in set QUEUE and the time cell associated with SHIPS. K is greater than 1Ø, is formed. The value of the first member of the subset (the first class index) is then chosen and placed in index variable K. If no member of class SHIPS satisfy the test chain, control passes to the next sector and K contains an unpredictable value.

SECTION 8.

HISTOGRAMS AND STATISTICAL DISTRIBUTIONS.

User Specified Distributions.

A distribution is a device which serves one of the following two functions:-

a) It may be used by the program to obtain random samples from an empirical distribution supplied as data.

b) It may be used as a device for accumulating data developed by the program.

DIST synonymous with HIST

The general form of this statement is:-

DIST $name_1 n_1 (n_2, n_3, n_4)$, $name_2 n_5 (n_6, n_7, n_8)$------

Where $name_1$ $name_2$ are extended C.S.L. names.

n_1, n_2 n_5; n_6 are unsigned integer constants less than 1023

n_3 n_4, n_7, n_8 are unsigned integer constants

This statement defines groups of distributions $name_1 1$ $name_1 2$ up to $name_1$ n_1 and $name_2 1$, $name_2 2$ up to $name_2$ n_5. If n_1 or n_5 were omitted single distributions would be defined i.e. $name_1$ and $name_2$.

Consider only the '$name_1$' distributions. Each of these would have n_2 cells, n_3 would be the value associated with the first cell and n_4 the incremental value between cells. This statement must precede all reference to the distributions.

Example:-

1. DIST OBSERVATIONS (60,1,1) FREQU 3 (20,0 5)

Here four distributions are defined - OBSERVATIONS, FREQU1, FREQU2, FREQU3.

The last three each have 20 cells which have associated values 0,5,10, - - - up to 95. These values must not be confused with the contents of the cells, i.e. they can be considered as the sampling scale of the distribution/histogram.

2. HIST WAITTIME (8,Ø,1Ø)

Here one distribution, WAITTIME, is defined having 8 cells, associated values Ø, 1Ø, 2Ø, 3Ø, 4Ø, 5Ø, 6Ø, 7Ø.

SAMPLE

The function SAMPLE may be included in an arithmetic statement.

Its general form is:-

SAMPLE (dist, stream)

Where 'dist' is a distribution name including a suffix (if required) and 'stream' is the name of a fixed point cell which contains some initial value for the random number generator.

This causes a random value to be selected from the distribution and the value selected is taken as the value of the expression.

Example:-

1. T. SHIP. 5 = SAMPLE (WAITTIME, STR) + T.TIDEIN

Here the time cell associated with SHIP 5 is given the value of T.TIDEIN plus a sample waiting time taken from the distribution defined under DIST i.e. WAITTIME.

2. TIMEAWAY (I) = 2 * MILES (I)/ AVSPEED + SAMPLE (TURNROUND, STARTNO)

ADD

This statement is used to enter data into a histogram. Its generalform is:-

ADD expression, dist

where 'expression' is any integer expression
and 'dist' is a user defined distribution.

One is added to the cell of the distribution which is nearest to the current value of the expression.

Examples:-

1. ADD A / 6 FREQU 2

Assuming 'A' had the value 18 then one would be added to the cell of the distribution, FREQU2, whose associated value was nearest to 3.

2. FIND K = QUEUING FIRST

SHIPCAP (K) GE 5ØØØ

ADD - T.SHIP.K,QUEUETM

CLEAR synonymous with ERASE

The general form of this statement is:-

CLEAR $dist_1$,$dist_2$, - - -, - - -

where $dist_1$, $dist_2$ are defined distribution names.

All the distributions mentioned in this statement have the values in their cells set to zero. CLEAR/ERASE can be used on any valid member of the I/O List.

(See Section 9)

EXAMPLES:-

1. ERASE FREQU 1

2. CLEAR WAITTIME, QTIMES

Theoretical Distributions.

The following distribution functions are available to the user and may form a part or the whole of an expression, subject to the rules for functions in Section 10.

RANDOM

The general form of this statement is:-

RANDOM (range, stream)

where 'range' is an expression
and 'stream' is an unsuffixed cell name containing an initial value for random number generation.

This statement causes an integer to be selected at random in the range Ø to R-1 where R is the current value of the expression 'range'

Examples:-

1. QUTIME = RANDOM (11, STRM)

Here QUTIME is given a random value between Ø and 1Ø

2. TRY (1) = X + RANDOM (B ** 3, NUMB)

NORMAL

The general form of this statement is:-

NORMAL (mean, dev, stream)

where mean' is an expression whose value represents a mean value. 'dev' is an expression whose value represents a standard deviation and 'stream' is an unsuffixed cell name which contains an initial value for random number generators.

This statement enables the user to take a sample normal deviate from the normal distribution whose mean is the current value of the mean expression and whose standard deviation is the current value of the deviation expression.

Examples:-

1. IQ = NORMAL (1ØØ, 9, VAL)

Here a sample is taken from the normal distribution whose mean is 100 and standard deviation 9.

2. T. SHIP K = NORMAL (Ø, 3, STRM) * 3 + T/2

A sample is taken from a normal distribution whose mean is the value T/2 and whose standard deviation is 9. i.e. the above is equivalent to

T. SHIP K. = NORMAL (T/2, 9, STRM)

NEGEXP

The general form of this statement is:-

NEGEXP (mean, stream)

Where stream'is an unsuffixed cell which contains an initial value for the random number generator.

This statement causes a random sample to be taken from the negative exponential distribution whose mean is the current value of the 'mean' expression.

Examples:-

1. WAITTIME = T. LASTLOADED + INT + NEGEXP (1, STRNG)

This statement calculates the sum of the time the last truck was loaded plus the interval plus a sample from a negative exponential whose mean is 1 and places the result in cell, WAITTIME.

2. ELEM (K) = NEGEXP (8, VAL)

N.B. In all statements involving random sampling an initial value, the stream parameter, is specified in an unsuffixed cell. This value is used to generate random numbers using the formula:-

$$X_{n+1} = 125.X_n \bmod 2^{19}$$

Where X_{n+1} is the random number to be generated

X_n is the last random number (in the first case the value in the unsuffixed cell).

i.e. the next random number is obtained from the remainder when 125 x X_n is divided by 2^{19}

SECTION 9

INPUT/OUTPUT STATEMENTS

Input/Output statements are in the form of a keyword followed by a list of information to be input or output. These keywords fall into three groups:

a) Input from Cards

- READ — synonymous with
- INPUT — synonymous with
- LIST

b) Output to Cards

- PUNCH

c) Output to the Printer

- PRINT — synonymous with
- OUTPUT
- CHECK

and TYPE (output to console)

d) Magnetic Tape Operations

- READTAPE
- WRITETAPE
- REWIND
- BACKSPACE

and ENDFILE

List elements

The list may be composed of any of the following elements seperated by commas:

a) Cells (I/O)

A cell name may appear anywhere in a list and may be suffixed or unsuffixed.

Example:- A, GRID (1,K), T. ATIME, ARRIVALTIME, X, Y, Z

If used following an input keyword then values will be read into each of the cells listed. If used with an output keyword then the current values of the listed cells will be output.

b) Arrays (I/O)

Array names (no subscripts) may be also be included in the input output list, if an entire array is to be transmitted in or out. The elements of an array to be filled or output are always dealt with in such a way that the first subscript varies most rapidly, then, the second etc.

Example:- X, Y, COEFFS, W (where X, Y & W are single cells and COEFFS is a 3 by 3 array).

If the list follows an input keyword then cells X, Y COEFFS (1,1) COEFFS (2,1), COEFFS (3,1) COEDDS (1, 2) - - - - - - -COEFFS (3,3) and W are filled with input data. If the list follows an output keyword then the current contents of the list are output, in the order specified above.

c) HISTOGRAMS AND DISTRIBUTIONS (I/O)

Histogram and distribution names may also be included in an input/ output list. On input, data will be read in to fill all cells of the named distribution/histogram. On output the total number of observations in the distribution/histogaram is given along with the number of observations recorded in each cell.

Example:- FREQ 1, AMPLETIMES where both names represent distributions.
If used in conjuntion with an input keyword then these distributions/ histograms are filled with input data. (N.B. the first value to be input must be the total number of observations being input).
If used in conjunction with an output keyword then consider the following:-
Example: FREQ1 has 8 cells whose values are:-
0 7 5 18 9 6 2 1
then output will be:
48 0 7 5 18 9 6 2 1 where 48 is the total number of observations recorded.

Implied Loops (I/O)

Elements in an input/output list may be qualified by an implied loop which can assume any of the forms available with the FOR statement (see page 3.3)

i.e. a) $name_1 = m_1, m_2$

N.B. m_3 must be omitted since only unit incrementing is available.

b) $name_1 = name_2$

The elements to be controlled by the implied loop must be enclosed in brackets followed by one of the forms of the implied loop given above.

Examples:-

X, (A (1,K), T. MAN. K) K = QUEUEA, MATRIX

For K equal to all the class indices of entities on set QUEUEA the value of attribute i.e. A(1,K), and of the time cell associated with each entity, T.MAN will be input/output.
The implied loop is also useful for inputing or outputing sections of arrays.

Example:-

(K (6,J))J = 3,6
Outputs/inputs K (6,3), K(6,4), K(6,5) and K(6,6)

Loops of the above type may be nested to a depth of four.

Example:-

(K (I,J) I = 1,3)J = 2,4

Inputs/Outputs K(1,2), K(2,2) K(3,2) K(1,3), K(2,3) - - - - - - - - - - - - - - - - , K(3,4) in that order.

Literal Strings (O)

Strings of characters may appear in the list associated with an output keyword. The string must be enclosed within inverted commas and causes the string of characters specified to be output e.g. To print headings.

Example:-

"ANSWER =", K, "TONS"

If K had the value 175 then the following would be output:-

ANSWER = 175 TONS

The '/' separator (I/O)

The character '/' may be used in an input/output list in place of a comma.

Input:- When the '/' is encountered this causes data to be taken from the next record to fill the following list elements (e.g. next card)

Output:- When the '/' is encountered this causes data to be output to a new record from the following list elements (e.g. new line on printer)

Example:- "PAGE NO.", I/VECTOR

Assuming I contains 1 then PAGE NO. 1 is printed on one line and the elements of the array VECTOR are printed on succeeding lines.

The *n specification (O)

For printed output or output to tape each cell is allowed 10 character positions to hold its value unless *n precedes these cells. The first of the 1Ø characters is the sign, the last is a space, enabling input and output to be compatible. 'n' is an unsigned integer constant which represents the number of consecutive character positions to be used to contain following cell values. The number of characters specified by 'n' must exclude the initial sign character and the final blank characters. 'n' must be less than 16.

Example:- X,Y, * 5 Z, W

X and Y will each be allowed 1Ø characters, Z and W will each be allowed 7, i.e. each field includes a sign character (blank or -) and a terminal blank character for I/O compatibility.

The ** specification (O)

This specification is designed for output to the printer and when encountered in an output list it causes the printer to throw the paper to the head of the next page.

Example:- ** "PAGE NO. ", I/A, B,C

This will cause PAGE NO. n, where n is the value in cell I, to be printed at the head of the next page and on a new line the values of A,B and C.

READ synonomous with

INPUT synonomous with LIST

These key words are used to input data from cards and take the following form:-

INPUT List

where list contains any permissable element as defined previously for input

Example:- 1. READ (K(1,J)) J = 1,10,/X,Y,Z

2. LIST FREQUI, AVTIME, MATRIX

PRINT synonomous with OUTPUT

These two keywords are used to output information to the printer and take the form:-

PRINT list

Where 'list' is a list of permissable elements in an output list.
Each line of print can contain 132 characters. If the PRINT list is greater than 132 then the first 132 are printed on one line and the next 132 on the following line etc. If the configuration has a 120 print position printer the compiler can be altered to suit this.

Example:-

PRINT "OBSERVATIONS" // * 5 (SMPLETIMES I/) I = 1,3

CHECK

This statement is only executed if SENSE SWITCH 2 is on. The general form is:-

CHECK list

Where 'list' is as previously specified for output keywords.
When SENSE SWITCH 2 is on the first six letters of all variables in the list are printed along with their current values. If SENSE SWITCH 2 is off this statement is equivalent to DUMMY

Example:-

CHECK " TRACING "/T.AB I / MATRIX / OBSERVATIONS

where MATRIX is a 2 by 2 array.

T.AB. I is T-call associated with entity I
OBSERVATIONS is a distribution of 6 cells
& TRACING is a fixed-heading

this statement would result in the following printed output (if SENSE SWITCH 2 was on)

TRACING

AB I 5

MATRIX 17 - 9 12 5

OBSERV 2Ø Ø 1 3 14 2 Ø Ø

NB. CHECK must not use implied loops in its output list.

CHECK ON

CHECK ON

This statement overrides the use of the sense switch and all CHECK statements after this become executable unconditionally.

CHECK OFF

CHECK OFF

This statement when encountered, returns the CHECK statement to the control of the sense switch.

TYPE

If a console typewriter is present in the configuration then this statement may be used to output messages to the operator. In no console is present this statement is illegal. This statement should only be used to communicate with the operator.

The general form of this statement is:-

TYPE list

where 'list' is as defined for the output list.

Example:- TYPE 'PUT SENSE SWITCH 2 ON AND PRESS RUN'

READTAPE

The general form of this statement is:-

READTAPE n list

where'list' is as specified for the input list
and 'n' is one of 2,3,4,5,6, & 7, specifying the logical drive on the magnetic tape unit. The statement cause a record/records from the specified tape to be read into the specified list. A record on tape is 120 charcaters long (print image). Enough records are read to fill all members of the list.

Example:-

READTAPE 2 A, B, C /MATRIX / DISTRIB

WRITEPATE

The general form of this statement is:-

WRITEPATE n list

Where 'list' is as specified previously for the output list
and 'n' is one of 2,3,4,5,6 or 7, specifying the logical drive to be used on the magnetic tape unit.

This statement will write the current values of the list to tape.

Example:-

WRITETAPE 2 (VECTOR(I))I = 1,100, MED, A, B, C,/(Z(J))J = 1,10

REWIND

The general form of this statement is:-

REWIND n

where 'n' is one of 2,3,4,5,6, or 7, specifying a logical drive on the magnetic tape unit. This statement rewinds tape 'n'

BACKSPACE

The general form of this statement is:-

BACKSPACE n

where 'n' is one of 2,3,4,5,6 or 7, specifying a logical drive on the magnetic tape unit. This statement backspaces the specified tape 'n' one record.

ENDFILE

The general form of this statement is:-

ENDFILE n

where 'n' is one of 2,3,4,5,6 or 7, specifying a logical drive on the magnetic tape unit.
This statement writes the end of file record to tape 'n'. If this record is encountered during a READTAPE statement the program will stop and 'SY$TEM CARD ENCOUNTERED' will be printed on the printer. If start is pressed the system will proceed with next job. See Section 16.

SECTION 10

PROCEDURES AND FUNCTIONS

It may often occur that a section of program, i.e. routine, is a common requirement of several programs or even to different parts of the same program. Such routines may be written as entities and are known as subprograms.

There are two types of subprograms in Extended C.S.L.

a) PROCEDURE - routines written in Extended C.S.L.

b) FUNCTION - routines written in EASYCODER assembly language.

PROCEDURE

A PROCEDURE must be defined before it is first encountered in an executable statement. Any variables defined within a PROCEDURE are defined within that PROCEDURE only. The general form of a PROCEDURE is

PROCEDURE name(d_1, d_2, d_3, - - - - - d_n)

- - - - - -
- - - - - -
- - - - - -

where 'name' is a permissable Extended C.S.L. name used to identify the PROCEDURE

and d_1, d_2 - - - - - - up to d_n are dummy parameters used in statements within the PROCEDURE and can be replaced by any Extended C.S.L. entity except a distribution/histogram name.

The body of the PROCEDURE is written in Extended C.S.L. following the PROCEDURE statement.

If in a PROCEDURE a test fails then control returns to the main program. Control will also return to the main program if the PROCEDURE is completely executed.

The PROCEDURE is used form the main program by means of the following statement, i.e. by including the PROCEDURE name in an expression, as for FUNCTION (see below):-

var = name (P_1, P_2, ----------p_n)

where 'name' is the name of the PROCEDURE to be executed and P_1, P_2, up to p_n are the arguments to correspond to the dummy variables specified and used in the PROCEDURE.

The PROCEDURE name appearing by itself with parameters as a statement is compiled as a call to the PROCEDURE,

The PROCEDURE name may appear in the body of the PROCEDURE since it can be used to transmit the result. It will be a fixed point cell but if a floating point result is required the PROCEDURE statement must be as follows:-

FLOAT PROCEDURE name (d_1, d_2, ----------d_n)

FUNCTION

The general form of this statement is:-

FUNCTION name$_1$, name$_2$, ----------

where name$_1$, name$_2$, etc. are taken to be FUNCTION names. This statement tells the compiler to include these functions from the library in this program, since they occur in the body of the program. The EASYCODER routines corresponding to these names must have been added to the function library. (All functions, excepting SAMPLE, NORMAL, NEGEXP and RANDOM, must be dealt with in this way).

These functions are then used by writing their name followed by a list of parameters in an Extended C.S.L. statement.

function name (expression$_1$, expression$_2$----------)

The expressions are then evaluated and their values taken as the parameters required by the FUNCTION. The FUNCTION will produce a fixed point result unless its name has appeared in a FLOAT statement or it is defined as a floating point FUNCTION. The number of parameters should correspond to the number expected by the FUNCTION.

Example:-

1. X1 = (B-SQRT (B*B-4*A*C))/(2*A)

Here SQRT must be a routine on the function library for finding the square root of one parameter which in this csse is the value of the expression $B^2 - 4AC$

2. Y = MAXOF (H,C,J(3), D)

The Extended C.S.L. function library will contain a standard group of functions which may be used in Extended C.S.L. programs. These are:

a) FIXED POINT

MAXOF (e_1, e_2, - - - - - - e_n) - where e_1, e_2, - - - - - e_n are arithmetical expressions having integer values. The result is the value of the maximum expression.

MINOF (e_1, e_2, - - - - - e_n) - as for MAXOF but the result is the value of the minimum expression.

SQROOT (e) - This finds the square root of a single expression which has a positive integer value.

GENERAL (c_1, c_2, c_3, c_4, c_5, stream)

This function selects a random sample from a general distribution using the formula $AX^2\log_2(1-X) + B(1-X)^2\log_2 X + C + DX + EX^2$

where A,B,C,D, & E are the values in cells c_1, c_2, c_3, c_4, & c_5

Stream gives an initial value to the random number generator.

MOD (e) - this finds the absolute value of a single expression

b) FLOATING POINT

FLOAT - this function does not concern the codes as it is handled automatically. However it must be present in the function library.

SQRT (e) - this finds the square root of a positive floating point expression.

LOGE (e) - this finds natural logarithm of a positive floating expression.

LOGTEN (e) - this finds the log to the base 10 of a positive floating point expression.

EXP (e) - this finds the exponential of a positive floating point expression.

EXPTEN (e) - this finds the antilog to the base 10 of a single floating point expression.

ABS (e) - this finds the absolute value of a floating point expression.

The function library may be added to at any time by writing new functions in BASIC EASYCODER with the following modifications.

1. The first card of a FUNCTION must have the following format:-

Col. 6 - 13 - these columns must contain the word FUNCTION

Col. 14 - if blank this designates a fixed point function if '.' this designates a floating point function.

Cols. 15 - 20 - these columns contain the name of the function and must be an Extended C.S.L. name of up to 6 characters.

Cols. 21 - 72 - parameters which must be legal EASYCODER tags separated by commas. These tags can be preceeded by an '*' or '.'

* - this specifies address of variable is parameter, not value.

. - this specifies variable is a floating point quantity otherwise fixed point value of tag is considered.

N.B. If function name appears as parameter, then its value is placed in the result location.

2. Instructions must be in admode 3. So the following instructions are illegal:-

1) CAM
2) ADMODE

Also illegal is 3) DC

3. The following extra instructions are available:-

1) RETURN - returns control to main program

2) COM A,B - causes a numeric comparison between values in addresses A & B

3) M A,B - multiplies the contents of A by the contents of B and leaves result in B

4) D A,B - contents of B are divided by contents of A - result in B.

5) P A,B - contents of B are raised to the power of the contents of A and the result is placed in B.

The above are all fixed point arithmetic, the following are floating point:

6) FC A,B - as for 2) but floating point numbers.

7) FA A,B - add the floating point quantity in A to the floating point quantity in B. Result in B.

8) FS A,B - subtract the floating point quantity in A from the floating point quantity in B. Result in B.

9) FM A,B - as for 3) but floating point numbers

10) FD A,B - as for 4) but floating point numbers

11) FP A,B - as for 5) but floating point numbers

The following are two conversion instructions:-

12) FIX A,B - the quantity in A is converted to a fixed point number in B.

13) FLOAT A,B - the quantity in A is converted to a floating point number in B.

The above rules, if observed, will produce valid functions. To maintain a function library the following rules must be observed:

1. A system card containing an asterisk in column 1 and the word LIBRARY in columns 7 - 14 must preceed the pack of cards to be used to update the library.

2. The library is held in alphabetical order and so the input must be in alphabetical order. i.e. according to function names.

3. If the name of an input function is the same as a function on tape then the new function overwrites the old one.

4. If the word DELETE appears from column 6 onwards in place of FUNCTION and the function name, only, appears in column 15 onwards, then that function is deleted from the library.

5. A card with END punched in columns 6 - 8 specifies the end of a library update.

6. The system tape should be on logical drive Ø and the tape being created should be on logical drive 1. The END card causes a halt and if a program execution is to follow, the new tape should be placed on logical drive Ø and a new work tape placed on logical drive 1.

SECTION 11

SIMULATION

A simulation model may be formulated using Extended C.S.L. If the structure of the model is defined by the Simulation Control Statements, given later in this section, the computer will supply a built in mechanism for time advancement and iteration.

T-cells can be defined (e.g. in CLASS statement) and are considered as holding times relative to the present time, zero. The value in the T-cell, associated with an entity, could for example specify when this entity needs consideration. The units in the T-cell can be considered as anything, e.g. days, hours, minutes, etc., but all T-cells must be assumed to hold values in the same units. A negative time would specify that consideration of the associated entity is overdue.

Example:-

> T.SHIP 5 might represent the time that must elapse before SHIP 5 can enter harbour.

A simulation operates in a two phase manner, based on a consideration of the time cells.

PHASE 1

This first phase scans the T-cells associated with entities to find which event will occur first, i.e. T-cells with smallest positive value. All T-cells are then incremented by that amount to simulate time advancement.

CLOCK - This is a special T-cell which holds the total elapsed time since the beginning of simulation. It may be used and/or reset by the programmer. CLOCK is also incremented on any time advancement.

PHASE 2

This is the iteration phase where a scan of all program sectors (equivalent to activities) is made and, if possible, some action to advance the state of the model is made. When no further action can be performed then a return to Phase 1 is made. For example if all SHIPs were outside the harbour and their associated T-cells all held positive values (e.g., waiting for tide to come in) and no other action could be performed then a return to Phase 1 and a time advancement would enable progress.

The cycle of operations between Phase 1 and 2 is continued until the value in CLOCK exceeds a given value specified in the ACTIVITIES statement. At this point control is passed to the FINALIZATION section (see later in this section).

SIMULATION CONTROL STATEMENTS.

ACTIVITIES

This has the form:-

ACTIVITIES cell name

This statement defines the start of the list of program sectors (activities) in the program. This causes the built in mechanism for time advancement and iteration to be incorporated, i.e. sectors will be treated as in Phase 1 & Phase 2. 'cellname' is either an unsigned integer constant or a cell name containing an integer, which is used to specify the duration of simulation. Simulation will cease when the value in CLOCK exceeds this value, i.e. control will pass to the FINALIZATION sector. The statement ACTIVITIES, as well as signifying the start of program sectors, can be considered as terminating the first section of program in which definition and initialization statements occur.
NB. The duration of simulation cannot be changed by altering CLOCK in program execution, since a twin cell, which cannot be accessed by program, is incremented with CLOCK to control the simulation duration!

BEGIN

The general form of this statement is:-

BEGIN identification

This statement is used to specify the start of activities (program sectors). It may be omitted from the first activity whose start is defined by the ACTIVITIES statement. An identification of up to 28 characters will be printed on the standard output.

RECYCLE

The general form of this statement is:-

RECYCLE identification.

If this statement is encountered during the execution of an activity then a further complete cycle of all activities, once the present cycle is completed is performed before time advancement is considered. Only one further cycle is performed regardless of how many RECYCLE statements there are in the program. Characters for identification may follow the structural word RECYCLE but these characters will be ignored at each execution of the statement.

Normally time advancement occurs after a pass through all activities is made. However, these activities may interact in a complex manner, or the action in one sector (e.g. the releasing of a machine) may enable further actions to occur at that instant in time. The encounter of at least one RECYCLE statement in the program will ensure another pass of all activities to be performed before time advancement occurs.

FINALIZATION

The general form of this statement is:-

FINALIZATION identification.

When the CLOCK value exceeds the value specified in the ACTIVITIES statement then activity cycling ceases and control passes to the section of program following the FINALIZATION statement. Identification can follow the FINALIZATION key word but this will be ignored and not output. The FINALIZATION card can also be considered as terminating the activities section.

Either both ACTIVITIES and FINALIZATION statements must occur or neither.

SECTION 12

PROGRAM TERMINATION

The various methods of bringing the program to a halt are discussed below:-

EXIT

The form of this statement is:-

EXIT

This causes termination of program execution and control is returned to the Extended C.S.L. Monitor System on the H200. The words 'NORMAL EXIT' appear on the printer.

EXIT need not appear in the program to terminate the execution since it is automatically inserted, by the compiler, prior to termination by END (see below).

Program execution is terminated, and control returned to the Extended C.S.L. Monitor System, if an input statement is encountered for which insufficient data (or no data) is available and a system card is read; or if an End of File record is read from tape. The words 'Normal Exit' do not occur (see page 9.9)

END

The form of this statement is:-

END

This must be the last physical statement (last card in deck) in the program.

FINISH

The form of this statement is:-

FINISH

This statement causes control to pass to FINALIZATION section, at the end of the cycle, regardless of CLOCK value.

RESTART

The form of this statement is:-

RESTART

This statement causes constant data to be re-entered and transfers control back to the beginning of the initialization section. RESTART does not re-zero locations other than those mentioned in DATA (see later) or in the initialization section.

ADVANCE

The form of this statement is:-

ADVANCE

This causes the cycle to be completed without considering the succeeding activities.

SECTION 13

DATA STATEMENT

The form of this statement is:-

DATA

This card must preceed the cards which contain the values to be initially set as data, which in turn must preceed the END card. The following rules for creating the DATA statements must be adhered to:-

a) No individual numbers may contain characters other than digits (they may however be preceeded by a minus sign.) Numeric data is written in columns 7 to 72 of the card and any characters other than above will be ignored.

b) If there is too much data in any item of the statement for one card, a second card may be used for continuation, columns 1 - 6 being left blank.

Examples:-

```
          DATA
1.  RUNTIME          55
2.  PARAMETER        2ØØ
3.  MATRIX            3  -7  1  5 19
                      2   4 -5  Ø  6
                      8   Ø  Ø  7  3
                     12  11  9 13  Ø
4.  FREQU 1          24   1  2  3  7  8  3  Ø
5.  INQUEUE       1  3   4  5  6  7
          END
```

The numbering on the left hand side is only present for reference in the explanations below.

1. This sets the value 55 in cell RUNTIME

2. This sets the value 2ØØ in cell PARAMETER

3. If MATRIX is a 5 by 4 array then the 4 cards specified fill that array.

4. In this case FREQU 1 is a distribution the value 24 represents the total number of observations and is followed by the observations to be recorded in each of the 7 cells.

5. a) If INQUEUE is a set then entities with indices, 1,3,4,5,6, & 7 are recorded in that set, in that order.

   b) If INQUEUE 1 is defined as a set then entities 3,4,5,6, & 7 are recorded in that set, in that order.

SECTION 14

OPERATING AN EXTENDED C.S.L. PROGRAM

Operation of the computer is under the control of the Extended C.S.L. Operating System. This system minimizes operator intervention and is controlled by using System Control Cards. These cards all have an asterisk in column 1 and are detailed below:-

DATE card

Columns:	1	7	15
	*	DATE	DD/MM/YY

Normally this will be the first card on the input deck and is used to supply the date (col. 15 - 22) to the computer for use in program printed output.

LIBRARY card

Columns:-	1	7
	*	LIBRARY

This card must preceed any function (written in Basic Easycoder) to be used to modify and update the function library. (Pgs. 10.2 to 10.5) All cards following, up to but not including the next card with an asterisk in column 1, are taken to be function cards.

COMPILE card

Columns:-	1	7	17
	*	COMPILE	PRNAME

This card indicates that the named program, PRNAME, (Up to 6 characters) which follows it is to be compiled up to the END card. The compiler leaves a binary version of the program on tape. 1. An 'R' in col. 71 denotes floating point calculations are involved and column 72 should contain (one less than the number of 4K banks; e.g., column 72 should contain 7 if one wishes to use 32K). An 'N' in column 70 will prevent the compilation of statement numbers to be printed out under control of sense switch 4.

EXECUTE card

Columns	1	7
	*	EXECUTE

When encountered after an END card this causes the program which is currently on tape 1. (placed there by * COMPILE) to be executed.

END RUN card

Columns:-	1	7
	*	END RUN

This card is used to designate the end of the deck of programs being run and must be followed by at least 2 blank cards.

REMARK card

Columns:-	1	7	17 - - - - - - - - - - 80
	*	REMARK	

This card can occur anywhere in the deck except between source statement cards or data cards. When encountered the message in columns 17 - 80 is typed on the console and a few seconds elapse when the operator can put on sense switches. (This card must not be used if there is no console in the configuration).

TAPES card

Columns:-	1	7
	*	TAPES

When encountered the card causes a dump of information on all tapes to be given.

GET card

Columns:-	1	7	17
a)	*	GET C	PRNAME
b)	*	GET Tn	PRNAME

Where n is the tape drive number and PRNAME is the program name.

Form a) of this card causes the binary program card deck to be read into memory.

Form b) causes the binary program on drive 'n' to be read into memory.

KEEP card

Columns:-	1	7
a)	*	KEEPC
b)	*	KEEPTn

Form a) causes a binary program card deck to be punched and

Form b) causes a binary program dump to be given on tape drive 'n'.

POSTMORTEM card

Columns:-	1	7
	*	PMDUMP

Provided the last*card was an EXECUTE card and provided a post mortem dump has not just been given then the contents of all variables are dumped and an exit from the program occurs.

WAIT card

Column:-	1	7
	*	WAIT

This causes program to halt at XXXXX and awaits operator action. To continue press run.

OPERATING PROCEDURES

The System tape must be on logical drive Ø, and three work tapes on logical drives 1,2,3.

The operating procedure is:-

Bootstrap- tape Ø twice to location Ø,
Press RUN three times.
The call card has the format:
CSL2ØØA1Ø $\overset{18}{*}$

From here on the system is controlled by the System Control cards just described. The Sense Switches have the following significance, if ON:-

Sense Switch 1 - Clock changes printed.

Sense Switch 2 - 'CHECK' lists are output.

Sense Switch 3 - Gives dumps of all variables every time control passes to the Extended C.S.L. Monitor.

Sense Switch 4 - Prints out sector Headings. 'The statement numbers of the latest statement executed in proceeding sector will be printed provided there was no 'N' in col. 7Ø of COMPILE card'.

Messages Output from Extended C.S.L. compiler are:

PASS 1

Definition Errors

INVALID DEFINITION OF XXXXXX

DIMENSION TOO LARGE

FUNCTION XXXXXX NOT IN LIBRARY

LIBRARY NOT ON TAPE Ø

UNDEFINED WORD

PREVIOUSLY DEFINED WORD AFTER IS

* XXXXXX IS MISSPELT

Syntax Errors

EXCESS OPEN BRACKETS

EXCESS CLOSE BRACKETS

FIRST WORD SHOULD BE A SET NAME

FIRST WORD SHOULD BE A CLASS NAME

KEY WORD MISSING

SET USED WITH WRONG CLASS

SET AFTER KEYWORD MISSING

HISTOGRAM MISSING

REAL VARIABLE IN ADD

UNSUBSCRIPTED ARRAY NAME

Logical Errors

* UNBALANCED LABELS

DUPLICATED LABEL

INDENTATION ERROR

NO LABEL

SET CONTROLLING LOOP ALTERED

JUMP INTO LOOP

REAL VARIABLE!

ALL T.CELLS ARE ZERO

Other Errors

SYMBOL TABLE OVERFLOW

LOOPS NESTED TOO DEEP

STATEMENT TOO LONG

ERROR IN COLUMN nnn (OCTAL)

TOO MUCH DATA

N.B.

Those marked with * can be non fatal errors.

PASS 2

Syntax Errors

INVALID ARITHMETICAL EXPRESSION

INVALID CHARACTER IN LIST

EXPRESSION MISSING

NOT ENOUGH PARAMETERS FOR FUNCTION

NON-ARITHMETIC WORD IN EXPRESSION

SUBSCRIPT ERROR

INVALID USE OF FUNCTION

n AFTER L.H.S.

NON BOOLEAN RHS

INVALID WORD IN EMPTY LIST

PROGRAM TOO BIG

NO INDEX CAN BE FOUND

KEYWORD FOUND INSTEAD OF VARIABLE

CONSTANT IN I/O LIST

Logical Error

LOOP INDEX ALTERED

Execution

TAPE ERROR L

SYSTEM CARD

ENTITY ZERO L

DIVSR ZERO L

STREAM ZERO L

DISTN ZERO L

NB. L is statement number which will be output provided there was no 'N' in column 7Ø of COMPILE card.

Binary Program Loading

WRONG BINARY DECK

INVALID TAPE NUMBER

PROGRAM NOT IN LIBRARY

Binary Program Creation

INVALID TAPE NUMBER

PROGRAM - TEST

```
C    SIMULATION OF PRIVATE TELEPHONE LINES                                    0
       CLASS TIME PERSON 50 SET WAITING UNUSED CONNECTED                      001
       CLASS TIME LINE 6 SET FREE                                             002
       ARRAY USER(LINE)                                                       003
       HIST DELAY(30,0,1)POISSON(7,0,1) INCOMING(20,0,1)                      004
       READ MEAN LENGTH NOLI STREMA,STREMB,STREMC,STREMD,INCOMING             005
       T.ARRIVAL=SAMPLE(INCOMING,STREMC)+1                                    006
       INPUT POISSON                                                          007
       T CALL =2                                                              008
       LINE NOLI LOAD FREE                                                    009
       PERSON 50 LOAD UNUSED                                                  010


       ACTIVITIES LENGTH                                                      011


       BEGIN PHONING                                                          012
   100 T.CALL EQ 0                                                            013
       T.CALL = SAMPLE (POISSON,STREMB)                                       014
       FIND X UNUSED FIRST                                                    015
       PERSON X FROM UNUSED INTO WAITING                                      016
       CHECK X,T.CALL                                                         017
       T.PERSON X = 0                                                         018
       GOTO 100                                                               019


       BEGIN RINGOFF                                                          020
       FOR Y = 1,NOLI                                                         021
            T.LINE.Y EQ 0        :200                                         022
            LINE.Y INTO FREE                                                  023
            X=USER(Y)                                                         024
            PERSON X FROM CONNECTED INTO UNUSED                               025
            CHECK X , Y                                                       026
   200      DUMMY                                                             027


       BEGIN INCOMING                                                         028
     1 T.ARRIVAL EQ 0                                                         029
       T.ARRIVAL = SAMPLE(INCOMING,STREMC)                                    030
       MADE+1                                                                 031
       CHECK MADE                                                             032
       FIND Y FREE   FIRST : 1                                                033
       RECEIVED +1                                                            034
       CHECK Y                                                                035
       Z=RANDOM(50,STREMD)+1                                                  036
       PERSON Z INTO CONNECTED : 1                                            037
       THRU +1                                                                038
       USER(Y)=Z                                                              039
```

FIG A.1.

PROGRAM - TEST

```
     CHAIN                                                    100
          PERSON Z FROM UNUSED                                101
          OR PERSON Z FROM WAITING                            102
     T.LINE Y = NEGEXP(MEAN,STREMA) + 1                       103
     LINE Y FROM FREE                                         104
     CHECK Z                                                  105
     GO TO 1                                                  106


     BEGIN SPEECH                                             107
300  FIND X WAITING FIRST                                     108
     FIND Y FREE FIRST                                        109
     T.LINE.Y = NEGEXP (MEAN,STREMA)+1                        110
     LINE Y FROM FREE                                         111
     ADD -T.PERSON.X,DELAY                                    112
     USER(Y)=X                                                113
     PERSON X FROM WAITING INTO CONNECTED                     114
     CHECK X,Y,T.LINE Y                                       115
     GOTO 300                                                 116


     FINALIZATION                                             117
     PRINT  "MADE     RECEIVED  THRU"                         118
     PRINT MADE RECEIVED THRU                                 119
     PRINT "WAITING TIME HISTOGRAM"                           120
     OUTPUT DELAY                                             121


     END                                                      122


*    EXECUTE
```

FIG A.1

APPENDIX A

Figure A.1. shows the listing for an Extended C.S.L. program TEST, and a detailed description follows.

Initialization Section

1. A comment line (denoted by a 'C' in column 1) is used to expand the title TEST and explain the object of the program.

2. A class PERSON with 50 members is defined with associated T-cells. All or any of the members of PERSON can be held in sets WAITING, UNUSED, CONNECTED. i.e. to hold PERSON waiting to make a call, not thinking of making a call and at present calling respectively.

3. A class LINE with 6 members and associated T-cells is defined to represent say 6 lines into an office building. All or any of the members of class LINE can be held in set FREE. i.e. if a line is not in FREE it is assumed to be in use.

4. Three histograms are defined:-

    a) DELAY which has 30 cells & associated values $\emptyset$ - 29 to represent the waiting time in minutes for a FREE line under the given conditions, i.e. 6 lines & a staff of 50. This will be built up in the program.
    b) POISSON which has 7 cells & associated values from $\emptyset$ - 6 will be filled with 7 observed totals to sample lengths of telephone call.
    c) INCOMING which has 20 cells with associated values from $\emptyset$ - 19 which are to be filled with sample frequencies of elapse times between incoming calls.

5. The next statement reads in data from cards into:-

    a) MEAN - the value read into this cell will be taken as the mean value of a negative experimental distribution.
    b) LENGTH - this cell is used to hold the value which represents the duration of the run. i.e. 'N' minutes.
    c) NOLI - this is the cell whose value will be taken as the number of lines to be used for the random number generator.
    d) STREMA - the value read into this cell will be used as a starting value for the random number generator.
    e) STREMB, STREMC AND STREMD as for d) but different start values.
    f) INCOMING - the frequencies recorded are read into cells of the histogram.

6. A sample time for the next incoming call is placed in T.ARRIVAL.

7. This statement reads in the seven observations into the distribution POISSON. (N.B.) DELAY is initially cleared so no CLEAR/ERASE statement is needed.

8. The T cell associated with the length of a call; T CALL is initially set to 2, i.e. it will be 2 minutes till this call is finished.

9. This statement, where NOLI has a value between 1 and 6, is used to load that number of lines into set FREE.

10. Set UNUSED is loaded with PERSON. 1 up to PERSON. 50.

Activities Section

The statement ACTIVITIES LENGTH, where LENGTH contains the length of simulation parameter, defines the end of initialization and the beginning of activities.

PHONING

This activity is introduced by the BEGIN PHONING statement, If T. CALL is not equal to zero then control passes to the next activity (RINGOFF). Otherwise T.CALL is set to a sample value from POISSON. The next statement places in X the class index of the first member of set UNUSED and the value of X is printed by the following statement. PERSON.X i.e. first member of set UNUSED, is removed from that set and placed in set WAITING.

The CHECK statement (if Sense Switch 2 is on) prints out current values of X and T.CALL. The time cell for PERSON. X is then set equal to ø i.e. PERSON.X is now available to start phoning. Control is then passed back to statement 1øø. T.CALL may now not be zero, so control would pass to the next sector.

Summary

If T.CALL is zero, (i.e. a phone call has ended) then a new sample time is loaded into it and the first person of set UNUSED is placed at the end of set WAITING (to make a call).

If T.CALL is not zero control passes to the SPEECH activity in an attempt to advance its state.

RING OFF

This sector involves a FOR loop where Y takes values from 1 up to the number of lines used in the simulation (NOLI) in steps of 1. If the T-cell associated with LINE Y is not equal to zero no action is performed and control is transferred to the beginning of the next loop, (@200). If it is zero, i.e. the duration the call on that line has terminated, the corresponding line (LINE Y) is placed into and at the end of set FREE and PERSON is removed from the set CONNECTED into the end of set UNUSED, where X has the value given to USER (Y) activity SPEECH or INCOMING.

Summary

As soon as a line becomes free it is placed into set FREE and the associated PERSON X is taken from CONNECTED and placed in UNUSED. (N.B. First time through this activity no action is taken).

INCOMING

If no incoming call is due i.e. T.ARRIVAL is not zero then control passes to the next activity (SPEECH). Otherwise T.ARRIVAL is put equal to a sample time from the INCOMING frequency distribution and MADE is incremented by 1 and CHECKED. If there is a free LINE then received is incremented by 1 and Y checked. Otherwise return to beginning of activity to see if another incoming call is due. A random person Z is chosen and if he is not in CONNECTED he is placed there and THRU is incremented by 1 and the person is linked to the line,(i.e. to beginning of activity). PERSON.Z is then removed from either set UNUSED or WAITING, a sample time is given to the use of the line and LINE Y is removed from set FREE. This activity is repeated until no incoming call is due at that instant in time.

Summary

This activity deals with incoming calls by searching for a free line and then for the person to whom the call is being made, MADE represents the number of attempted incoming calls, RECEIVED represents the number of incoming calls accepted by exchange and THRU represents the number of incoming calls which were successfully connected with the required person.

SPEECH

The class index of the first PERSON in set WAITING is loaded into X. The class index of the first LINE in set FREE is loaded into Y. The T-cell associated with LINE Y is now loaded with a random value from a negative exponential distribution whose mean is the value in cell MEAN. LINE Y is then removed from set FREE. The negative value of T.PERSON X (i.e. if there were no members in either or both of sets WAITING & FREE, then control would have been transferred to the PHONING sector and on the next cycle. (i.e. CLOCK advanced) T.PERSON. X will have a negative value. The person and the line are connected by USER(Y)=X.

Person X is now removed from WAITING and placed in set CONNECTED. Control is transferred to the beginning of the sector, causing the sector to be repeated until either sets WAITING or FREE are emptied.

The above activities are recycled, a time advancement occuring after each cycle, until the value in CLOCK exceeds the parameter in LENGTH. At this point the FINALIZATION sector is reached and the required output is printed out.

INDEX