SIMSCRIPT II.5
INSTANT MANUAL

CDC® OPERATING SYSTEMS
NOS
NOS/BE
SCOPE 2
<table>
<thead>
<tr>
<th>REVISION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Initial printing</td>
</tr>
<tr>
<td>(12-1-77)</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Revised to add the EXTENDED ENTITIES ARRAYS INCLUDE language statement, the MASK.F function mnemonic, and the MOVE.F routine. Other minor changes and corrections made.</td>
</tr>
<tr>
<td>(9-25-76)</td>
<td></td>
</tr>
</tbody>
</table>

Publication No. 84000150

REVISION LETTERS I, O, Q AND X ARE NOT USED

© 1977, 1978 jointly by control data corporation and CACI, Inc.-Federal
SIMSCRIPT II.5 is a trademark of CACI, Inc.-Federal
Reprinted with the permission of CACI, Inc.-Federal
Printed in the United States of America

Address comments concerning this manual to:
CONTROL DATA CORPORATION
Publications and Graphics Division
Data Services Publications
P. O. Box 0 HQC62C
Minneapolis, MN 55440
PREFACE

This instant outlines the SIMSCRIPT II.5 programming language for the CONTROL DATA computer systems. More detailed information on the SIMSCRIPT II.5 language can be obtained from the following manuals:

P.J. Kiviat, R. Villanueva, H.M. Markowitz
SIMSCRIPT II.5 Programming Language
(CACI)

SIMSCRIPT II.5 Reference Handbook
(CACI)

SIMSCRIPT II.5 User's Manual—CONTROL DATA computer systems (CDC)
(CACI)
SIMSCRIPT II.5 ELEMENTS

NAMES

Combination of letters, digits and periods arranged as follows:
- contains at least one letter or
- contains digits and at least two non-terminal periods.
- cannot be broken across card boundaries.

Examples
X
WAITING.LINE
1B
1.2.3
1.5E15

CONSTANTS

Integer Combination of up to 18 digits

Examples
2
347
314159265

Real Combination of up to 15 digits containing at least one period

Examples
3.14
1.
.759

String Sequence of characters enclosed within " ".

Examples
“ABCD”
“...”

Text Sequence of characters enclosed within ".

VARIABLES

Integer Range \(-2^{59} -1\) to \(2^{59} -1\). As a result of multiplication or division or conversion from real to integer, or integer to real, maximum value is \(2^{48} -1\).

Integer packed . Bit packed \((n-m)\)
1 \(\leq n \leq m \leq 60\)
. Field packed \((n/m)\)
m = 2, 3, 4, 5, 6, 10, 12, 15
n = 1 to m
. Intra-packed \((*/m)\)
m = 2, 3, 4, 5, 6, 10, 12, 15

Real Range \(10^{-293}\) to \(10^{+322}\)
(approximately 15 significant digits)

Alpha Up to 10 characters for a replacement statement.
SOURCE LANGUAGE

NOTATION

The notation employed in describing SIMSCRIPT II.5 is a combination of conventions used in several computer programming language descriptions. In the following pages:

1. Words in capital letters are statement keywords.

2. Primitives shown in italics are basic language constructs. They are:
   - name
   - integer $i$
   - number $n$
   - string $s$
   - text $t$

3. A metavariable denotes an occurrence of an element of the type represented by the metavariable symbol shown in italics.

4. A statement is a combination of keywords, primitives, and metavariables that follows a certain pattern, called the syntax of the statement.

5. Brackets [ ] and braces { } denote choices. When brackets appear, a choice may be made from the options indicated. When braces appear, a choice must be made. The items available for selection appear in a vertical list within the brackets or braces. When a choice can be repeated, a symbol (or symbols) that must separate the items in the list of choices is written at the upper right-hand corner of the brackets or braces. For example, if a choice appears as $\{A^2\}$, the sequence $A,A,B,A,\ldots,B$ might be selected. The choice represented by $\{A\}$ is logically equivalent to $A[.]A[.]A\ldots[.]A$.

6. The null character $\&$ is used to indicate that no symbol need separate the items in a list of choices. An example of $\{A\}^\&$ might be $AABABB\ldotsA$. The choice represented by $\{A\}^\&$ is logically equivalent to $A[A][A]\ldots[A]$.

7. A list separator symbol can itself be complex, involving choices and repetitions, as in $\{A[\text{AND}]B[\text{OR}]\}$ an instance of which might be $A$ AND $B$ OR $A$ OR $B$.

8. Plural keywords ending in $s$ such as VARIABLES and LINES, can be written in singular form as VARIABLE or LINE when called for by the grammar of a statement.
SYMBOLS

character =
  \{ letter
  \}
  \{ special character
  \}
  \{ digit
  \}
  \{ blank
  \}
  \{ period
  \}

letter =
special character =
digit =
blank =
period =

\{ A \}
\{ B \}
\{ C \}
\{ D \}
\{ E \}
\{ F \}
\{ G \}
\{ H \}
\{ I \}
\{ J \}
\{ K \}
\{ L \}
\{ M \}
\{ N \}
\{ O \}
\{ P \}
\{ Q \}
\{ R \}
\{ S \}
\{ T \}
\{ U \}
\{ V \}
\{ W \}
\{ X \}
\{ Y \}
\{ Z \}

\{ * \}
\{ ** \}
\{ , \}
\{ . \}
\{ / \}
\{ < \}
\{ > \}
\{ \| \}
\{ = \}
\{ \} \}
\{ : \}
\{ ; \}
\{ ? \}
\{ % \}
\{ \& \}
\{ \$ \}
\{ @ \}

empty space
METAVARIABLES

arithmetic expression $e = \left\{ a, n, r, s, l, [S] \right\} \left\{ \begin{array}{c} + \\ - \\ * \\ ** \\ / \end{array} \right\}

comma $c = \left\{ , \text{AND} \right\}

AND

for phrase $for = \left\{ \begin{array}{c} \text{name}_{i} \backslash \text{BACK \ FROM \ i \ \text{TO} \ v \ [BY \ e]} \\ \text{EACH} \\ \text{EVERY} \\ \text{ALL} \\ \text{EACH \ name \ IN \ THE \ DICTIONARY} \end{array} \right\} \left\{ \begin{array}{c} \text{name}_{i} \ [\text{CALLED \ name}_{i}] \\ \text{OF} \\ \text{IN} \\ \text{ON} \\ \text{AT} \\ \text{IN \ REVERSE \ ORDER} \end{array} \right\}

format $f_1 = \left\{ \begin{array}{c} B e \\ S e \\ l \end{array} \right\}

format $f_2 = \left\{ \begin{array}{c} f_1 \\ i \ \text{Ce} \\ i \ l \ e \\ i \ A \ e \\ i \ D \ (e,e) \\ i \ E \ (e,e) \\ i \ T \ e \end{array} \right\}

label $l = \left\{ \begin{array}{c} \text{name} \\ n \end{array} \right\}
logical expression $\phi =$

$\{ (\phi) \\ e \ (\text{IS} \ \Omega e) \ ^{\text{(POSITIVE)}} \\\n e \ (\text{IS} \ \text{NOT}) \ ^{\text{(NEGATIVE)}} \\\n e \ (\text{IS} \ \text{ZERO}) \ ^{\text{(ZERO)}} \\\n \text{MODE} \ (\text{IS} \ \text{NOT}) \ ^{\text{(INTEGER)}} \\\n \text{ALPHA} \ ^{\text{(REAL)}} \\\n \text{DATA} \ (\text{IS} \ \text{NOT}) \ \text{ENDED} \\\n \text{CARD} \ (\text{IS} \ \text{NOT}) \ \text{NEW} \\\n \text{PAGE} \ (\text{IS} \ \text{NOT}) \ \text{FIRST} \\\n \text{THE} \ ^{\text{THIS}} \ \text{\&}\ (\text{IS} \ \text{NOT}) \ \text{EMPTY} \\\n \text{THE} \ ^{\text{THIS}} \ e \ (\text{IS} \ \text{NOT}) \ \text{IN} \ ^{\text{AN}} \ ^{\text{THE}} \ ^{\text{NAME}} \ name \\\n \text{EVENT} \ (\text{IS} \ \text{NOT}) \ ^{\text{INTERNAL}} \ ^{\text{EXTERNAL}} \ ^{\text{ENDOGENOUS}} \ ^{\text{EXOGENOUS}} \}$

relational operator $\Omega =$

$\{ \text{EQUAL TO} \ ^{\text{EQ}} = \\\n \text{NOT EQUAL TO} \ ^{\text{NE}} \neq \\\n \text{LESS THAN} \ ^{\text{LT}} < \\\n \text{GREATER THAN} \ ^{\text{GR}} > \\\n \text{NOT GREATER THAN} \ ^{\text{NLGT}} \leq \\\n \text{NOT LESS THAN} \ ^{\text{NLSS}} \geq \}$

routine $r = \text{\textquoteleft}name\textquoteright$

selection clause $sc = \{ \text{WITH} \ ^{\text{WHEN}} \ ^{\text{EXCEPT}} \ ^{\text{UNLESS}} \} \phi \ ; \ }
termination clause tc = \{ \textbf{WHILE} \}
\{ \textbf{UNTIL} \}
\{ \phi \} \{ , \}

variable v = name
\left[ \{ \{ e \} , \{ , \} , \{ \{ \} , \} \{ \textbf{\{ \} , \} \} \} \right]

word w = \begin{cases} t \\ name \\ n \\ special \ character \\ s \end{cases}

Words must be separated from each other by one or more blanks unless they are special characters.

Periods ( . ) are ignored between words and at the end of statements.

Comments can be inserted between any two words in a program by enclosing them in quote marks (" ") formed by two consecutive apostrophes. The right-hand set of quotes is not necessary if the comment is the last item on a card.
LANGUAGE STATEMENTS

\[
\nu \left\langle \text{AS} \right\rangle \left\langle \text{TH} \right\rangle \text{[name]} \quad \begin{cases} \text{ACCUMULATE} \\
\text{TALLY} \end{cases}
\]

\[\text{name} (n \text{ TO } n \text{ BY } n) \left\langle \text{AS} \right\rangle \left\langle \text{TH} \right\rangle \text{[name]} \text{ HISTOGRAM} \]

Specifies automatic data collection and analysis for named variables.

\[
\begin{cases}
\text{SCHEDULE} \\
\text{ACTIVATE} \\
\text{CAUSE} \\
\text{REACTIVATE} \\
\text{RESCHEDULE}
\end{cases}
\begin{cases}
\text{[A]} \\
\text{[AN]} \\
\text{[THE [ABOVE]]} \\
\text{[THIS]} \\
\text{[GIVEN]} \\
\text{[GIVING]} \\
\{e\}^c
\end{cases}
\begin{cases}
\text{name} \left\langle \text{CALLED } \nu \right\rangle \\
\text{AT } e \\
\text{IN } \{e\} \\
\text{IN } \{\text{NOW}\} \\
\text{IN } \{e\}^c \\
\text{IN } \{\text{DAYS}\} \\
\text{IN } \{\text{UNITS}\} \\
\text{IN } \{\text{HOURS}\} \\
\text{IN } \{\text{MINUTES}\}
\end{cases}
\]

Files an event notice in the events set according to its time.

\[\text{ADD } e \text{ TO } \nu\]

Adds the value of \(e\), the value of the variable \(\nu\).

\[
\begin{cases}
\text{ADVANCE} \\
\text{BACKSPACE} \\
\text{INPUT} \\
\text{OUTPUT} \\
\text{FILES} \\
\text{RECORDS} \\
\text{USING} \\
\text{TAPE} \\
\text{UNIT}
\end{cases}
\begin{cases}
e\end{cases}
\]

84000450 A
Specifies a call to a named routine whenever the indicated statement is executed. Inputs to the routines are:

<table>
<thead>
<tr>
<th></th>
<th>BEFORE</th>
<th>AFTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE</td>
<td>Not allowed</td>
<td>Entity identifier</td>
</tr>
<tr>
<td>DESTROY</td>
<td>Entity identifier</td>
<td>Not allowed</td>
</tr>
<tr>
<td>CAUSE</td>
<td>Entity identifier, time</td>
<td>Entity identifier, time</td>
</tr>
<tr>
<td>CANCEL</td>
<td>Entity identifier</td>
<td>Entity identifier</td>
</tr>
<tr>
<td>FILE</td>
<td>Entity identifier, subscripts</td>
<td>Entity identifier, subscripts</td>
</tr>
<tr>
<td>REMOVE</td>
<td>Entity identifier, subscripts</td>
<td>Entity identifier, subscripts</td>
</tr>
</tbody>
</table>

Logical phrases control the execution of statements that follow them. When more than one statement is to be controlled, the word DO precedes them. Multiple control phrases terminating control on the same LOOP statement are preceded by the word ALSO.

\[
\{ \text{for} \} \times \begin{bmatrix} \text{for} \\ tc \\ sc \end{bmatrix} \begin{bmatrix} \text{DO} \\ \text{THIS} \\ \text{THE FOLLOWING} \end{bmatrix}
\]

Synonyms indicating the transfer point of the false condition of a preceding IF statement.

BACKSPACE — See ADVANCE

BEFORE — See AFTER

BEGIN HEADING
Marks the beginning of a heading block within a report section.
BEGIN REPORT [ON A NEW PAGE] [PRINTING for IN GROUPS OF i [PER PAGE]]

Marks the beginning of a report section.

BREAK name TIES \( \{ \text{HIGH} \} \) name \( \{ \text{LOW} \} \) c THEN

Establishes a priority order within an event class.

\[
\left\{ \begin{align*}
\text{PERFORM} \\
\text{CALL} \\
\text{NOW}
\end{align*} \right\} \text{name}
\left[ \begin{align*}
\text{THE} \\
\text{THIS} \\
\text{GIVEN} \\
\text{GIVING}
\end{align*} \right] \left\{ e \right\}_c \left[ \begin{align*}
\text{YIELDING} \{ v \}_c
\end{align*} \right]
\]

Calls a routine used as a procedure. Both input GIVEN and output YIELDING argument lists are optional.

CANCEL \[ \text{THE [ABOVE]} \] \[ \text{THIS} \] \[ \text{name} \] \[ \text{CALLED v} \]

Removes a scheduled event notice from the event set.

CAUSE — See ACTIVATE

\[
\left\{ \begin{align*}
\text{CLOSE} [\text{TAPe}]_e \\
\text{ENDFILE} [\text{UNIT}]
\end{align*} \right\}
\]

Writes an end-of-file mark on an output device.
COMPUTE \( v \begin{cases} \text{AS} \end{cases} \text{THE} \) \( \begin{cases} \text{AVERAGE} \\ \text{AVG} \\ \text{MEAN} \\ \text{SUM} \\ \text{NUMBER} \\ \text{NUM} \\ \text{VARIANCE} \\ \text{VAR} \\ \text{STD.DEV} \\ \text{STD} \\ \text{SUM.OF.SQUARES} \\ \text{SSQ} \\ \text{MEAN.SQUARE} \\ \text{MSQ} \\ \text{MINIMUM} (e) \\ \text{MIN} (e) \\ \text{MAXIMUM} (e) \\ \text{MAX} (e) \\ \text{MIN} \\ \text{MAXIMUM} \\ \text{MAX} \end{cases} \) \( \text{OF} \ e \)

Must be controlled by a logical control phrase. Computes the indicated statistics of the expression \( e \) after the LOOP statement if the control is over a DO..LOOP block.

\[
\begin{cases}
\text{DESTROY} \begin{cases} \text{AN} \end{cases} \text{name [CALLED v]} \\
\text{CREATE} \begin{cases} \text{EACH} \end{cases} \begin{cases} \text{ALL} \end{cases} \begin{cases} \text{EVERY} \end{cases} \text{name [(e)]} \text{c} \\
\text{CYCLE} \\
\text{NEXT} \\
\end{cases}
\]

A control statement.
DEFINE \{name\} \(c\) AS [A]
    \(\begin{cases} \text{FIFO} \\ \text{LIFO} \end{cases}\)
SETS \[\begin{cases} \text{HIGH} \\ \text{LOW} \end{cases}\] RANKED \(c\) THEN
\[\begin{cases} \text{FF} \\ \text{FL} \\ \text{FB} \\ \text{FA} \\ \text{RF} \\ \text{RL} \\ \text{RS} \end{cases}\]
WITHOUT \(\begin{cases} \text{F} \\ \text{L} \\ \text{P} \\ \text{S} \\ \text{N} \end{cases}\) ATTRIBUTES \(i, j\) WITHOUT \(\begin{cases} \text{F} \\ \text{L} \\ \text{R} \end{cases}\) Routines.

Defines set ranking, owner and member attributes and generated set processing routines.

\[
\begin{align*}
&\text{DEFINE } \{name\} \(c\) \text{ AS } [A] \\
&\quad \{\text{INTEGER} \\ \text{ALPHA} \\ \text{REAL} \\ \text{TEXT} \} \\
&\quad \{\text{RELEASABLE} \\ \text{FORTRAN} \} \\
&\quad \{\text{GIVING} \\ \text{GIVEN} \} \quad \{\text{VALUES} \} \\
&\quad \{\text{WITH} \} \quad \{\text{ARGUMENTS} \} \\
&\quad \text{YIELDING} \quad \{\text{VALUES} \} \\
&\quad \{\text{ARGUMENTS} \} \\
&\text{DEFINE } \{name\} \(c\) \text{ AS } [A] \\
&\quad \{\text{INTEGER} \} \quad \{\text{VARIABLES} \} \\
&\quad \{\text{STREAM} i \} \quad \{\text{ARRAYS} \} \\
&\text{DEFINE } \{name\} \(c\) \text{ AS } [A] \\
&\quad \{\text{SIGNED INTEGER} \\ \text{INTEGER} \\ \text{ALPHA} \\ \text{REAL} \\ \text{TEXT} \} \\
&\quad \{\text{DIMENSIONAL} \} \\
&\quad \{\text{DIM} \} \\
&\quad \{\text{DUMMY} \} \\
&\quad \{\text{SUBPROGRAM} \} \\
&\quad \{\text{FORTRAN} \} \\
&\quad \{\text{EXTENDED} \} \\
&\quad \{\text{SAVED} \} \\
&\quad \{\text{RECURSIVE} \} \\
&\quad \{\text{MONITORED ON THE} \} \\
&\quad \{\text{LEFT} \} \quad \{\text{RIGHT} \} \quad \{c\} \\
\end{align*}
\]

Define properties of global and local variables, and routines.
DEFINE w TO MEAN \{w\} \cong

Instructs the compiler to substitute the words (up to the end of the card on which the statement appears) following the keyword MEAN for the indicated word in all subsequent statements, before they are compiled.

DESTROY — See CREATE

DO — See ALSO

\{ELSE \}
\{OTHERWISE\}

Synonyms indicating the transfer point of the false condition of a preceding IF statement.

END

Marks the end of a program preamble, routine, report section, and heading block of a report section.

ENTER WITH \nu

Used to transfer a "right-hand" value to a left-handed function.

\{UPON \} \text{\name} \left(\begin{array}{c}
\{\text{THE} \} \\
\{\text{THIS} \} \\
\{\text{GIVEN} \} \\
\{\text{GIVING} \}
\end{array}\right) \{\text{name} \} \text{\_c} \left[\text{SAVING THE EVENT NOTICE}\right]

Event declaration. Unless SAVED, an event notice is destroyed before an event routine is executed.

EVENT NOTICES \left[\text{INCLUDE} \{\text{name} \} \text{\_c}\right]

\{EVERY \} \text{\name} \text{\_c} \left[\begin{array}{c}
\{\text{MAY} \} \\
\{\text{HAVE} \}
\end{array}\right] \left[\text{AN} \right] \left[\text{THE} \right] \left[\text{SOME} \right] \text{\name} \text{\_c}

RANDOM \left[\begin{array}{c}
\{\text{STEP} \} \\
\{\text{LINEAR} \}
\end{array}\right] \left[\begin{array}{c}
\{\text{VARIABLES} \} \\
\{\text{ARRAYS} \}
\end{array}\right] \left[\begin{array}{c}
\{\text{IN ARRAY} \} \text{\_c} \\
\{\text{IN WORD} \} \text{\_c}
\end{array}\right]
Entity-attribute-set structure declaration. Specifies attribute packing, equivalence, word assignment and function options.

```
EXTENDED \{ ENTITIES \} INCLUDE \{ name \}^c
```

Defines extended memory data structures.

```
\{ EXTERNAL \} \{ EVENT \} \{ EXOGENOUS \} \{ PROCESS \} UNITS ARE \{ name \}_i^c
```

Names units from which external event data will be read.

```
\{ EXTERNAL \} \{ EVENTS \} \{ EXOGENOUS \} \{ PROCESSES \} ARE \{ name \}^c
```

Declares the names of events that can be triggered externally.

```
FILE [THE THIS]_e \begin{bmatrix}
FIRST \\
LAST
\end{bmatrix}
IN [THE THIS]_v
```

Files an entity in a set.

```
FIND \{ \{ v = [THE] [FIRST] e \} \}^c \begin{bmatrix}
\IF \{ FOUND \} \\
\ELSE \{ NONE \}
\end{bmatrix}
```

Must be controlled by a logical control phrase, but cannot be within a DO...LOOP block. The optional IF phrase directs control after the control phrase has been completed, depending upon the "success" of the FIND.

FOR — See ALSO and for metavariable
Subprogram declaration. Routines used as functions only have GIVEN arguments. If LEFT or RIGHT are not stated, RIGHT is implied.

\{GENERATE\} \{INHIBIT\} LIST ROUTINES

Controls the generation of LIST ATTRIBUTES routines for entities. By default, no LIST routines are generated.

\texttt{GO [TO] \{l [(e)]\}'}
\texttt{\{l [(e)]\}}

Transfers control to the indicated label.

\texttt{GO [TO] \{l \{OR\} \{n\} PER e}

Transfers control to the \texttt{n}th label in the label list according to the integer value of the transfer expression \texttt{e}.

\texttt{HERE} — A control statement

\texttt{IF \phi[...]} — See \texttt{\phi} metavariable

\texttt{INTERRUPT name [CALLED \nu]}

\texttt{JUMP \{AHEAD\}}
\texttt{\{BACK\}}

A control statement.

\texttt{l [(i)]}'

A statement label identifies a transfer point.
LAST COLUMN \( i \)

Characters beyond column \( i \) are ignored on subsequent cards.

LEAVE — A control statement.

\[ \text{LET } v = e \]

Assigns the value of \( e \) to the variable \( v \).
If \( v \) is integer and \( e \) is real, the result is rounded before storing.

\[ \text{LIST } \{ \{ e \} \text{ ATTRAIBUTES OF } \{ \text{name [CALLED e] } \} \{ \text{EACH name [OF v]} \} \} \]

A free-form output statement that labels and displays values of expressions and 1- and 2-dimensional arrays.

\{ \text{LOOP } \}
\{ \text{REPEAT } \}

Used with DO to delimit a group of statements controlled by one or more logical control phrases.

MAIN

Marks the beginning of a program's main routine. Execution commences at the first executable statement after MAIN.

\text{MOVE } \{ \text{FROM } e \} \{ \text{TO } v \} \]

Used within a routine defined for a monitored variable to access or set the value of the variable.

NEXT — See CYCLE

OTHERWISE — See ELSE

84000450 A
Establishes background conditions for properties of variables and functions that are effective unless overridden by subsequent DEFINE declarations or, in the case of local arrays, first use.

NOW — See CALL

PERFORM — See CALL

[VERY OLD]
[OLD]
[NEW]
PREAMBLE

Marks the beginning of the program preamble.

PRINT i [DOUBLE] LINES
WITH {e}
A GROUP OF {e}
{c}
FIELDS
SUPPRESSING FROM COLUMN i
LIKE THIS
THUS
AS follows

The i lines following the PRINT statement are format lines containing text and pictorial formats for the display of indicated expression values. The phrases
A GROUP OF {e} {c} FIELDS and SUPPRESSING FROM COLUMN i

can only be used within report sections that have column repetition.

PRIORITY ORDER IS {name} {c}

Assigns a priority order to different classes of events.
PROCESS name
{ THE
  THIS
  GIVEN
  GIVING
  (name)^c
}

PROCESSES INCLUDE (name)^c

REACTIVATE — See ACTIVATE

READ
{ v }^c
AS
{ BINARY
  [(e)]
  (f_2)^c
}
USING
{ THE BUFFER
  TAPE UNIT }^c

Used without an AS clause indicates a free-form data input.

RECORD
{ MEMORY
  STATUS
}

REGARDLESS — See ALWAYS

RELEASE { v }^c

Releases blocks of core pointed to by v; v's are assumed to be pointer variable

RELINQUISH e [UNITS OF] name [{ e }]

REMOVE
{ THE
  FIRST }
{ LAST }
{ THIS }
{ ABOVE }
FROM
{ THE
  THIS
  e
}

Removes an entity from a set.
REPEAT — See LOOP

REQUEST e [UNITS OF] name [(e)] [, WITH PRIORITY e]

RESCHEDULE — See ACTIVATE

RESERVE \{ \{v\}^c \text{ AS } \{e\} \text{ BY } \{\text{BY}^*\}\}^c

Allocates blocks of core of specified size to the pointer variables \( v \). Words assigned are data if no \text{BY}* phrase appears, and are pointers otherwise.

RESET \{\text{THE}} \{name\} \{\text{TOTALS OF} \} \{v\}^c

Initializes ACCUMULATE or TALLY counters associated with \( v \). If TOTALS is not qualified by a word, all counters of \( v \) are initialized.

RESOURCES \{ INCLUDE \{name\}^c \}

RESUME name [CALLED v]

\{RESUME \{SUPPRESS\}\} SUBSTITUTION

Used to override a currently defined substitution. This statement must not be placed on program cards with other statements.

RETURN \{ \{(e)\} \{WITH e\} \}

Used as a procedure, a routine returns control to its calling program with the statement RETURN; used as a function, a routine returns control and a value to its calling program by either of the statements \( \text{RETURN}(e) \) or \( \text{RETURN} \{e\} \) or \( \text{RETURN} \{WITH e\} \).
RE WIND \[ TAPE \]
\[ UNIT \] \( e \)
Rewinds an input/output device.

ROUTINE — See FUNCTION

SCHEDULE — See ACTIVATE

\[ \text{SKIP} \] \( e \) \begin{bmatrix} \text{FIELDS} \\ \text{INPUT} \\ \text{OUTPUT} \end{bmatrix} \begin{bmatrix} \text{CARDS} \\ \text{LINES} \\ \text{RECORDS} \end{bmatrix} \)
Applies to the current input or current output unit. SKIP \( e \) FIELDS applies to the current input unit only when it is used for freeform data input. CARDS, LINES, and RECORDS are synonyms. If neither UNPUT nor OUTPUT is specified, INPUT is implied.

\[ \text{START NEW} \] \begin{bmatrix} \text{PAGE} \\ \text{INPUT} \\ \text{OUTPUT} \end{bmatrix} \begin{bmatrix} \text{LINE} \\ \text{CARD} \\ \text{RECORD} \end{bmatrix} \)
Applies to the current input or current output unit. LINE, CARD, and RECORD are synonyms. If neither INPUT nor OUTPUT is specified, INPUT is implied.

START SIMULATION
Starts simulation by removing the first event from the events set and executing it.

STOP
Halts program execution.

\[ \text{STORE} \] \( e \) \( \text{IN} \) \( v \)
Assigns a value to a variable without mode conversion.
SUBROUTINE — See FUNCTION

SUBSTITUTE \begin{align*}
\{ \text{THIS} \\
\text{THESSE} \end{align*} \}_{i} \text{ LINES FOR } w

Similar to DEFINE TO MEAN but allows more than one card of words to be substituted.

SUBTRACT $e$ FROM $v$

Subtracts the value of $e$ from the value of the variable $v$.

SUPPRESS — See RESUME

SUSPEND [PROCESS]

The suspend statement may only appear in a process routine. The process is placed in the passive state.

[THE] SYSTEM

See EVERY

TALLY — See ACCUMULATE

\begin{align*}
\{ \text{TEMPORARY ENTITIES} \\
\text{PERMANENT ENTITIES} \end{align*} \{ \text{INCLUDE} \{ name \}^c \}

Declares the type of following EVERY statements to be permanent or temporary, as the case may be.

THE SYSTEM — See EVERY

[THEN] IF $\phi$ \{,\}

See IF

TRACE \begin{align*}
\text{USING} \\
\text{TAPE} \\
\text{UNIT}^e \end{align*}

Produces a backtrack of current subprogram calls. When the SIMSCRIPT II operating system uses TRACE the standard output device (printer) is used.
UNLESS — See tc metavariable (termination clause)

UNTIL — See sc metavariable (selection clause)

UPON — See EVENT

USE
  \{ THE BUFFER \}
  \{ TAPE \}^e
  \{ UNIT \}
FOR
  \{ INPUT \}
  \{ OUTPUT \}

Sets the indicated input/output device as the current input or output unit. All subsequent input/output statements that do not specify their own devices in USING phrases use these current units. THE BUFFER causes reading and writing in an internal file.

\{ WORK \}^e
\{ UNIT \}
\{ DAYS \}
\{ HOURS \}
\{ MINUTES \}

WHEN — See sc metavariable (selection clause)

WHILE — See tc metavariable (termination clause)

WITH — See sc metavariable (selection clause)

WORK — See WAIT

WRITE
\{ e \}^c
\{ \text{AS} \}
\{ \text{BINARY} \}
\{ [(e)] \}^c
\{ f_2 \}
\{ s \}^c
\{ \text{USING} \}
\{ \text{THE BUFFER} \}
\{ TAPE \}^e
\{ UNIT \}

\{ f_1 \}
\{ s \}
\{ \text{AS} \}
\{ \text{c} \}

84000450 A
### Attributes and Routines Required for Set Statements

<table>
<thead>
<tr>
<th>SET STATEMENT</th>
<th>ATTRIBUTES REQUIRED</th>
<th>ROUTINES REQUIRED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>L</td>
</tr>
<tr>
<td>FILE FIFO</td>
<td>F</td>
<td>L</td>
</tr>
<tr>
<td>FILE LIFO</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>FILE ranked</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>FILE FIRST</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>FILE LAST</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>FILE BEFORE</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>FILE AFTER</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>REMOVE FIFO</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>REMOVE LIFO</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>REMOVE ranked</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>REMOVE FIRST</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>REMOVE LAST</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>REMOVE specific</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>set IS EMPTY</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>IS IN set</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>FOR EACH...OF set</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>...REVERSE</td>
<td>L</td>
<td>P</td>
</tr>
<tr>
<td>FOR EACH...FROM...OF set</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>...REVERSE</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>FOR EACH...AFTER...OF set</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>...REVERSE</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>Automatic checking</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>
### Statistical Keywords for ACCUMULATE Statement

<table>
<thead>
<tr>
<th>STATISTICAL KEYWORD</th>
<th>SYNONYM</th>
<th>COMPUTATION</th>
<th>REQUIRED COUNTERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER</td>
<td>NUM</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SUM</td>
<td></td>
<td>$\sum X^*(TIME.V-T_L)$</td>
<td>SUM, $T_L$</td>
</tr>
<tr>
<td>MEAN</td>
<td>AVG</td>
<td>SUM/(TIME.V-T_0)</td>
<td>SUM, $T_L$, $T_0$</td>
</tr>
<tr>
<td></td>
<td>AVERAGE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUM.OF.SQUARES</td>
<td>SSQ</td>
<td>$\sum X^2*(TIME.V-T_L)$</td>
<td>SSQ, $T_L$</td>
</tr>
<tr>
<td>MEAN.SQUARE</td>
<td>MSQ</td>
<td>SSQ/(TIME.V-T_0)</td>
<td>SSQ, $T_L$, $T_0$</td>
</tr>
<tr>
<td>VARIANCE</td>
<td>VAR</td>
<td>MSQ - MEAN^2</td>
<td>SSQ, SUM, $T_L$, $T_0$</td>
</tr>
<tr>
<td>STD.DEV</td>
<td>STD</td>
<td>SQRT.F(VAR)</td>
<td>SSQ, SUM, $T_L$, $T_0$</td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>MAX</td>
<td>$M = \text{maximum (X) for all X}$</td>
<td>$M$, $N$</td>
</tr>
<tr>
<td>MINIMUM</td>
<td>MIN</td>
<td>$m = \text{minimum (X) for all X}$</td>
<td>$m$, $N$</td>
</tr>
</tbody>
</table>

**NOTES:**

- **TIME.V**: current simulated time
- **$T_L$**: simulated time at which variable was set to its current value
- **$T_0$**: simulated time at which accumulation started
- **X**: sample value of accumulation variable (before it changes to a new value)
<table>
<thead>
<tr>
<th>STATISTICAL KEYWORD</th>
<th>SYNONYM</th>
<th>COMPUTATION</th>
<th>REQUIRED COUNTERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER</td>
<td>NUM</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SUM</td>
<td></td>
<td>Σ X</td>
<td>SUM</td>
</tr>
<tr>
<td>MEAN</td>
<td>AVG</td>
<td>SUM/N</td>
<td>SUM, N</td>
</tr>
<tr>
<td></td>
<td>AVERAGE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUM.OF.SQUARES</td>
<td>SSQ</td>
<td>Σ X^2</td>
<td>SSQ</td>
</tr>
<tr>
<td>MEAN.SQUARE</td>
<td>MSQ</td>
<td>SSQ/N</td>
<td>SSQ, N</td>
</tr>
<tr>
<td>VARIANCE</td>
<td>VAR</td>
<td>MSQ - MEAN^2</td>
<td>SSQ, SUM, N</td>
</tr>
<tr>
<td>STD.DEV</td>
<td>STD</td>
<td>SQRT.F(VAR)</td>
<td>SSQ, SUM, N</td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>MAX</td>
<td>M = maximum (X) for all X</td>
<td>M, N</td>
</tr>
<tr>
<td>MINIMUM</td>
<td>MIN</td>
<td>m = minimum (X) for all X</td>
<td>m, N</td>
</tr>
</tbody>
</table>

**NOTE:** X is the sample value of tallied variable (before it changes to a new value).
<table>
<thead>
<tr>
<th>Statement Type</th>
<th>Statement</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>NORMALLY</td>
<td>Can appear anywhere in preamble.</td>
</tr>
<tr>
<td>1b</td>
<td>DEFINE TO MEAN</td>
<td></td>
</tr>
<tr>
<td>1c</td>
<td>SUBSTITUTE</td>
<td></td>
</tr>
<tr>
<td>1d</td>
<td>SUPPRESS SUBST.</td>
<td></td>
</tr>
<tr>
<td>1e</td>
<td>RESUME SUBST.</td>
<td></td>
</tr>
<tr>
<td>1f</td>
<td>GENERATE INHIBIT</td>
<td></td>
</tr>
<tr>
<td>2a</td>
<td>TEMPORARY ENTITIES</td>
<td>A preamble may contain many Type 2a, 2b, and 2c statements.</td>
</tr>
<tr>
<td>2b</td>
<td>PERMANENT ENTITIES</td>
<td>Each may be followed by a group of Type 3a, 4, and 5 statements.</td>
</tr>
<tr>
<td>2c</td>
<td>EVENT NOTICES</td>
<td></td>
</tr>
<tr>
<td>3a</td>
<td>EVERY</td>
<td>Many can follow a Type 2 statement. An entity or event notice name can appear in more than one EVERY statement.</td>
</tr>
<tr>
<td>3b</td>
<td>THE SYSTEM</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>DEFINE VARIABLE</td>
<td>No precedence relation if it defines a global variable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Must follow all Type 3a statements if it defines an attribute named in them.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A variable, attribute, or function name can appear in only one DEFINE statement.</td>
</tr>
<tr>
<td>5</td>
<td>DEFINE SET</td>
<td>Must follow Type 4 statements in a Type 2 statement group if it qualifies a set named in them.</td>
</tr>
<tr>
<td>6a</td>
<td>BREAK TIES</td>
<td>One statement allowed for each event notice.</td>
</tr>
<tr>
<td>6b</td>
<td>EXTERNAL EVENTS</td>
<td></td>
</tr>
<tr>
<td>6c</td>
<td>EXTERNAL UNITS</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>PRIORITY</td>
<td>Must follow all Type 2c and 6b statements.</td>
</tr>
<tr>
<td>8a</td>
<td>BEFORE</td>
<td>Allowed for each temporary entity, set, and event notice.</td>
</tr>
<tr>
<td>8b</td>
<td>AFTER</td>
<td></td>
</tr>
<tr>
<td>9a</td>
<td>TALLY</td>
<td>Either TALLY or ACCUMULATE statistics but not both, may be specified for unsubscripted global variables, attributes of permanent or temporary entities, event notices, processes or resources.</td>
</tr>
<tr>
<td>9b</td>
<td>ACCUMULATE</td>
<td></td>
</tr>
</tbody>
</table>

Of these statements, only Types 1 and 4 can be used in routines to declare local background conditions, variables, and substitutions.
## SYSTEM DEFINED NAMES

<table>
<thead>
<tr>
<th>Variable</th>
<th>Mode</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>BETWEEN.V</td>
<td>Subprogram</td>
<td>Subprogram variable called before each event is executed.</td>
<td>0</td>
</tr>
<tr>
<td>BUFFER.V</td>
<td>Integer</td>
<td>The length of the internal buffer.</td>
<td>132</td>
</tr>
<tr>
<td>EOF.V</td>
<td>Integer</td>
<td>End-of-file code; zero denotes notes that an end-of-file marker is an error; one indicates return control with EOF.V set to 2 when end-of-file is encountered; one for each input unit.††</td>
<td>0</td>
</tr>
<tr>
<td>EVENT.V</td>
<td>Integer</td>
<td>Code representing the event class to occur next.</td>
<td>0</td>
</tr>
<tr>
<td>EVENTS.V</td>
<td>Integer</td>
<td>The number of event classes.</td>
<td>0</td>
</tr>
<tr>
<td>F.EV.S(*)</td>
<td>Integer</td>
<td>Array containing the first-in-set pointers for the event set, EV.S</td>
<td>0</td>
</tr>
<tr>
<td>HEADING.V</td>
<td>Integer</td>
<td>A subprogram variable tested by the system for each new page.††</td>
<td>0</td>
</tr>
<tr>
<td>HOURS.V</td>
<td>Real</td>
<td>Number of hours per simulated day.</td>
<td>24</td>
</tr>
<tr>
<td>LINE.V</td>
<td>Integer</td>
<td>Number of the current output line.††</td>
<td>1</td>
</tr>
<tr>
<td>LINES.V</td>
<td>Integer</td>
<td>Number of lines per page.††</td>
<td>55</td>
</tr>
<tr>
<td>L.EV.S (*)</td>
<td>Integer</td>
<td>Array containing the last-in-set pointers for the event set EV.S</td>
<td>0</td>
</tr>
<tr>
<td>MARK.V</td>
<td>Alpha</td>
<td>Termination character required on external event cards and on the input for random variables.</td>
<td>*</td>
</tr>
<tr>
<td>MINUTES.V</td>
<td>Real</td>
<td>Number of minutes per simulated hour.</td>
<td>60</td>
</tr>
</tbody>
</table>

†† A separate value is maintained for each unit; only the currently used value is accessible to the program.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Mode</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>N.EVS (n)</td>
<td>Integer</td>
<td>Function returning the number of pending event notices of class n.</td>
<td></td>
</tr>
<tr>
<td>PAGE.V</td>
<td>Integer</td>
<td>Number of the current page.</td>
<td>1</td>
</tr>
<tr>
<td>PAGECOL.V</td>
<td>Integer</td>
<td>If ≠ 0, column number in which the word PAGE and the value of PAGE.V is to be printed on the output listing.</td>
<td>0</td>
</tr>
<tr>
<td>PARM.V (*)</td>
<td>Alpha</td>
<td>Array containing the user parameters passed to the program.</td>
<td></td>
</tr>
<tr>
<td>PRMB.V (*)</td>
<td>Integer</td>
<td>Array containing the values of the attributes of THE SYSTEM with IN WORD clauses.</td>
<td></td>
</tr>
<tr>
<td>RCOLUMN.V</td>
<td>Integer</td>
<td>Pointer to the last column read in the input buffer.</td>
<td>0</td>
</tr>
<tr>
<td>READ.V</td>
<td>Integer</td>
<td>Number of the current input unit.</td>
<td>5</td>
</tr>
<tr>
<td>RECORD.V(n)</td>
<td>Integer</td>
<td>The number of records read from or written on the indicated unit.</td>
<td>0</td>
</tr>
<tr>
<td>RRECORD.V</td>
<td>Integer</td>
<td>The number of records read from the current input unit.</td>
<td>0</td>
</tr>
<tr>
<td>SEED.V</td>
<td>Integer</td>
<td>Array containing initial random numbers.</td>
<td>Yes</td>
</tr>
<tr>
<td>TIME.V</td>
<td>Real</td>
<td>Current simulated time.</td>
<td>0</td>
</tr>
<tr>
<td>WCOLUMN.V</td>
<td>Integer</td>
<td>Pointer to the column last written in the output buffer.</td>
<td>0</td>
</tr>
<tr>
<td>WRECORD.V</td>
<td>Integer</td>
<td>The number of records written on the current output unit.</td>
<td>0</td>
</tr>
<tr>
<td>WRITE.V</td>
<td>Integer</td>
<td>Number of the current output unit.</td>
<td>6</td>
</tr>
</tbody>
</table>

†† A separate value is maintained for each unit; only the currently used value is accessible to the program.
<table>
<thead>
<tr>
<th>Constant</th>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXP.C</td>
<td>Real</td>
<td>2.7182818284590452</td>
</tr>
<tr>
<td>INF.C</td>
<td>Integer</td>
<td>Largest INTEGER value that can be stored.</td>
</tr>
<tr>
<td>PI.C</td>
<td>Real</td>
<td>3.1415926535897932</td>
</tr>
<tr>
<td>RADIUS.C</td>
<td>Real</td>
<td>57.295779513082321 degrees/ radian</td>
</tr>
<tr>
<td>RINF.C</td>
<td>Real</td>
<td>Largest REAL value that can be stored.</td>
</tr>
<tr>
<td>Function Mnemonic</td>
<td>Arguments†</td>
<td>Function Mode</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>---------------</td>
</tr>
<tr>
<td>ABS.F</td>
<td>e</td>
<td>Mode of e</td>
</tr>
</tbody>
</table>
| ALPHA.F           | a,b        | Integer       | Compares two alphanumeric words a and b and returns: 
|                   |            |               | $-1$ if $a < b$ 
|                   |            |               | $0$ if $a = b$ 
|                   |            |               | $1$ if $a > b$ |
| AND.F             | a,b        | Integer       | Logical product of a and b. |
| ARCCOS.F          | e          | Real          | Computes the arc cosine of a real expression; $-1 \geq e \geq 1$ |
| ARCTAN.F          | $e_1,e_2$  | Real          | Computes the arc tangent of $e_1/e_2$; $(e_1,e_2) \neq (0,0)$ |
| BETA.F            | $e_1,e_2,e_3$ | Real | Returns a random sample from a beta distribution 
|                   |            |               | $e_1 = \text{power of } x, \text{ real}$; $e_1 \geq 0$ 
|                   |            |               | $e_2 = \text{power of } (1-x), \text{ real}; e_2 \geq 0$ 
|                   |            |               | $e_3 = \text{random number stream, integer}$ |
| BINOMIAL.F        | $e_1,e_2,e_3$ | Integer | Returns a random sample from a binomial distribution 
|                   |            |               | $e_1 = \text{number of trials, integer}$ 
|                   |            |               | $e_2 = \text{probability of success, real}$ 
|                   |            |               | $e_3 = \text{random number stream, integer}$ |
| COS.F             | e          | Real          | Computes the cosine of a real expression given in radians. |

† $e =$ expression that can be of any complexity, including functions
v = variable
<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE.F</td>
<td>$e_1, e_2, e_3$</td>
<td>Real</td>
<td>Converts a calendar date to cumulative simulation time, based on values given to ORIGIN.R. $e_1 =$ month, integer $e_2 =$ day, integer $e_3 =$ year, integer</td>
</tr>
<tr>
<td>DAY.F</td>
<td>$e$</td>
<td>Integer</td>
<td>Converts simulation time to the day portion based on values given to ORIGIN.R. $e =$ cumulative simulation time, real</td>
</tr>
<tr>
<td>DIM.F</td>
<td>$v(*)$</td>
<td>Integer</td>
<td>Returns the number of elements pointed to by the pointer variable $v$, in the dimension of the array $v$.</td>
</tr>
<tr>
<td>DIV.F</td>
<td>$e_1, e_2$</td>
<td>Integer</td>
<td>Returns the truncated value of $(e_1/e_2)$ $e_1 =$ dividend, integer $e_2 =$ divisor, integer; $e_2 \neq 0$</td>
</tr>
<tr>
<td>EFIELD.F</td>
<td>none</td>
<td>Integer</td>
<td>Returns the ending column of the next data field to be read by a READ Free Form statement.</td>
</tr>
<tr>
<td>ERLANG.F</td>
<td>$e_1, e_2, e_3$</td>
<td>Real</td>
<td>Returns a sample value from an Erlang distribution $e_1 =$ mean, real $e_2 =$ $k$, integer $e_3 =$ random number stream, integer</td>
</tr>
<tr>
<td>Function Mnemonic</td>
<td>Arguments</td>
<td>Function Mode</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>EXP.F</td>
<td>e</td>
<td>Real</td>
<td>Computes EXP.C to the e\textsuperscript{th} power; e must be real.</td>
</tr>
<tr>
<td>EXPONENTIAL.F</td>
<td>e&lt;sub&gt;1&lt;/sub&gt;, e&lt;sub&gt;2&lt;/sub&gt;</td>
<td>Real</td>
<td>Returns a random sample from an exponential distribution</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>e&lt;sub&gt;1&lt;/sub&gt; = mean, real</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>e&lt;sub&gt;2&lt;/sub&gt; = random number stream, integer</td>
</tr>
<tr>
<td>FRAC.F</td>
<td>e</td>
<td>Real</td>
<td>Returns the fractional portion of a real expression.</td>
</tr>
<tr>
<td>GAMMA.F</td>
<td>e&lt;sub&gt;1&lt;/sub&gt;, e&lt;sub&gt;2&lt;/sub&gt;, e&lt;sub&gt;3&lt;/sub&gt;</td>
<td>Real</td>
<td>Returns a random sample from a gamma distribution</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>e&lt;sub&gt;1&lt;/sub&gt; = mean, real</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>e&lt;sub&gt;2&lt;/sub&gt; = k, real</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>e&lt;sub&gt;3&lt;/sub&gt; = random number stream, integer</td>
</tr>
<tr>
<td>GAMMAJ.F</td>
<td>e&lt;sub&gt;1&lt;/sub&gt;, e&lt;sub&gt;2&lt;/sub&gt;, e&lt;sub&gt;3&lt;/sub&gt;</td>
<td>Real</td>
<td>Same as GAMMA.F</td>
</tr>
<tr>
<td>HOUR.F</td>
<td>e</td>
<td>Integer</td>
<td>Converts event time to the hour portion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>e = cumulative event time, real</td>
</tr>
<tr>
<td>INT.F</td>
<td>v, e</td>
<td>Integer</td>
<td>Returns the rounded integer portion of a real expression.</td>
</tr>
<tr>
<td>ISTEP.F</td>
<td>v, e</td>
<td>Integer</td>
<td>Returns a random sample from a look-up table without interpolation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>v = variable that points to the look-up table</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>e = random number stream, integer</td>
</tr>
<tr>
<td>Function Mnemonic</td>
<td>Arguments</td>
<td>Function Mode</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>ITOA.F</td>
<td>e</td>
<td>Alpha</td>
<td>Converts an integer expression to an alpha-numeric value, left adjusted in a blank field.</td>
</tr>
<tr>
<td>LIN.F</td>
<td>v, e</td>
<td>Real</td>
<td>Returns a random sample from a look-up table, using linear interpolation</td>
</tr>
<tr>
<td>LOG.E.F</td>
<td>e</td>
<td>Real</td>
<td>Computes the natural logarithm of a real expression $e &gt; 0$.</td>
</tr>
<tr>
<td>LOG.NORMAL.F</td>
<td>$e_1, e_2, e_3$</td>
<td>Real</td>
<td>Returns a random sample from a lognormal distribution</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$e_1 = \text{mean, real}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$e_2 = \text{standard deviation}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$e_3 = \text{random number stream, integer}$</td>
</tr>
<tr>
<td>LOG.10.F</td>
<td>e</td>
<td>Real</td>
<td>Computes $\log_{10}$ of a real expression; $e &gt; 0$.</td>
</tr>
<tr>
<td>MASK.F</td>
<td>e</td>
<td>Integer</td>
<td>Builds a left justified mask of e bits.</td>
</tr>
<tr>
<td>Function Mnemonic</td>
<td>Arguments</td>
<td>Function Mode</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>MAX.F</td>
<td>$e_1, e_2, \ldots, e_n$</td>
<td>Real if any $e_i$ real; if none, integer</td>
<td>Returns the value of the largest $e_i$.</td>
</tr>
<tr>
<td>MIN.F</td>
<td>$e_1, e_2, \ldots, e_n$</td>
<td>Real if any $e_i$ real; if none, integer</td>
<td>Returns the value of the smallest $e_i$.</td>
</tr>
<tr>
<td>MINUTE.F</td>
<td>$e$</td>
<td>Integer</td>
<td>Converts event time to the minute portion. $e = \text{cumulative event time, real}$</td>
</tr>
<tr>
<td>MOD.F</td>
<td>$e_1, e_2$</td>
<td>Real if either $e_i$ real; if none, integer</td>
<td>Computes a remainder as $e_1 \cdot \text{TRUNC.F}(e_1/e_2) \cdot e_2$; $e_2 \neq 0$</td>
</tr>
<tr>
<td>MONTH.F</td>
<td>$e$</td>
<td>Integer</td>
<td>Converts simulation time to month portion based on values given to ORIGIN.R $e = \text{cumulative simulation time, real}$</td>
</tr>
<tr>
<td>NDAY.F</td>
<td>$e$</td>
<td>Integer</td>
<td>Converts event time to the day portion $e = \text{cumulative event time, real}$</td>
</tr>
<tr>
<td>NORMAL.F</td>
<td>$e_1, e_2, e_3$</td>
<td>Real</td>
<td>Returns a random sample from a normal distribution $e_1 = \text{mean, real}$ $e_2 = \text{standard deviation, real}$ $e_3 = \text{random number stream, integer}$</td>
</tr>
<tr>
<td>OR.F</td>
<td>$a, b$</td>
<td>Integer</td>
<td>Logical sum of $a$ and $b$.</td>
</tr>
<tr>
<td>Function Mnemonic</td>
<td>Arguments</td>
<td>Function Mode</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>OUT.F</td>
<td>e</td>
<td>Alpha</td>
<td>Sets or returns the alphabetic value of the \text{e}th character in the current output buffer; \text{e} must yield an integer value; \text{e} \geq 0; both right- and left-handed function.</td>
</tr>
<tr>
<td>POISSON.F</td>
<td>\text{e}_1, \text{e}_2</td>
<td>Integer</td>
<td>Returns a random sample from a Poisson distribution ( \text{e}_1 = ) mean, real ( \text{e}_2 = ) random number stream, integer</td>
</tr>
<tr>
<td>RANDI.F</td>
<td>\text{e}_1, \text{e}_2, \text{e}_3</td>
<td>Integer</td>
<td>Returns a random sample uniformly distributed between a range of values ( \text{e}_1 = ) beginning value, integer ( \text{e}_2 = ) ending value, integer ( \text{e}_3 = ) random number stream, integer</td>
</tr>
<tr>
<td>RANDOM.F</td>
<td>e</td>
<td>Real</td>
<td>Returns a pseudo-random number between zero and one ( e = ) random number stream, integer</td>
</tr>
<tr>
<td>REAL.F</td>
<td>e</td>
<td>Real</td>
<td>Converts an integer expression to a real value.</td>
</tr>
<tr>
<td>Function Mnemonic</td>
<td>Argument</td>
<td>Function Mode</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>RSTEP.F</td>
<td>v, e</td>
<td>Real</td>
<td>Returns a random sample from a look-up table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>v = variable that points to the look-up table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>e = random number stream, integer.</td>
</tr>
<tr>
<td>SFIELD.F</td>
<td>none</td>
<td>Integer</td>
<td>Returns the starting column of the next data field to be read by a READ Free Form statement.</td>
</tr>
<tr>
<td>SHL.F</td>
<td>w, n</td>
<td>Integer</td>
<td>Shift w left n positions, end around.</td>
</tr>
<tr>
<td>SHR.F</td>
<td>w, n</td>
<td>Integer</td>
<td>Shift w right n positions, arithmetic.</td>
</tr>
<tr>
<td>SIGN.F</td>
<td>e</td>
<td>Integer</td>
<td>Indicates the sign of a real expression.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 if e &gt; 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 if e = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-1 if e &lt; 0</td>
</tr>
<tr>
<td>SIN.F</td>
<td>e</td>
<td>Real</td>
<td>Computes the sine of a real expression given in radians.</td>
</tr>
<tr>
<td>SQRT.F</td>
<td>e</td>
<td>Real</td>
<td>Computes the square root of a real expression; e ≥ 0.</td>
</tr>
<tr>
<td>TAN.F</td>
<td>e</td>
<td>Real</td>
<td>Computes the tangent of a real expression given in radians.</td>
</tr>
<tr>
<td>TRUNC.F</td>
<td>e</td>
<td>Integer</td>
<td>Returns the truncated integer value of a real expression.</td>
</tr>
<tr>
<td>Function Mnemonic</td>
<td>Arguments</td>
<td>Function Mode</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>UNIFORM.F</td>
<td>$e_1, e_2, e_3$</td>
<td>Real</td>
<td>Returns a uniformly distributed random sample between a range of values. $e_1 =$ beginning value, real $e_2 =$ ending value, real $e_3 =$ random number stream, integer</td>
</tr>
<tr>
<td>WEEKDAY.F</td>
<td>$e$</td>
<td>Integer</td>
<td>Converts event time to the weekday portion. $e =$ cumulative event time, real</td>
</tr>
<tr>
<td>WEIBULL.F</td>
<td>$e_1, e_2, e_3$</td>
<td>Real</td>
<td>Returns a sample value from a Weibull distribution. $e_1 =$ scale parameter, real $e_2 =$ shape parameter, real $e_3 =$ random number stream, integer</td>
</tr>
<tr>
<td>XOR.F</td>
<td>$a, b$</td>
<td>Integer</td>
<td>Logical difference of $a$ and $b$.</td>
</tr>
<tr>
<td>YEAR.F</td>
<td>$e$</td>
<td>Integer</td>
<td>Converts simulation time to the year portion based on values given to ORIGIN.R $e =$ cumulative simulation time, real</td>
</tr>
<tr>
<td>ZTIME.F</td>
<td>none</td>
<td>Real</td>
<td>Returns the elapsed execution time in seconds.</td>
</tr>
<tr>
<td>Routine</td>
<td>Arguments</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| ORIGIN.R  | $e_1, e_2, e_3$ | Establishes an origin time when the calendar format is used  
  $e_1 = \text{month, integer}$  
  $e_2 = \text{day, integer}$  
  $e_3 = \text{year, integer}$ |
| TIME.R    | none      | Controls simulation timing and selects events.                              |
| SNAP.R    | none      | User supplied routine called by SIMSCRIPT II.5 when an execution error is detected. |
| DATE.R    | $y_1, y_2$ | Yields the current date and current time of day from the operating system. 
  $y_1 = \text{date, alpha}$  
  $y_2 = \text{time, alpha}$ |
| EOF.R     | $e$       | Writes an end of file on logical unit $e$.                                 |
| EXIT.R    | $e$       | Stops the simulation after printing EXIT $e$ and issuing a traceback.     |
| MOVE.F    | $v_1, v_2, n$ | Move $n$ words from one storage area to another. 
  $v_1 = \text{Base pointer of source array}$  
  $v_2 = \text{Base pointer of target array}$  
  $n = \text{Number of words}$ |
<table>
<thead>
<tr>
<th>Generated for</th>
<th>Generated Elements</th>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accumulated and tallied variables</td>
<td>Routine</td>
<td>Q.variable</td>
<td>A left-hand monitoring routine that accumulates or tallies data</td>
</tr>
<tr>
<td>Entities (5 unique characters)</td>
<td>Variables</td>
<td>G.entity*</td>
<td>Global variable having the entity class name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N.entity</td>
<td>No. of entities of the entity class (permanent entities only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>K.entity</td>
<td>Size of entity in computer words (temporary entities only)</td>
</tr>
<tr>
<td>Routines</td>
<td>C.entity</td>
<td></td>
<td>To reserve storage for permanent entities (i.e., to create them)</td>
</tr>
<tr>
<td></td>
<td>D.entity</td>
<td></td>
<td>Called when destroying a temporary entity to check for set membership error</td>
</tr>
<tr>
<td></td>
<td>A.entity</td>
<td></td>
<td>Called to list the values of entity attributes</td>
</tr>
<tr>
<td>Event Notices (5 unique characters)</td>
<td>Variables</td>
<td>G.event*</td>
<td>Global variable having the event notice name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I.event</td>
<td>Global variable holding the subscript for this event class in the event set</td>
</tr>
</tbody>
</table>

*The G prefix entry name is generated only when a word or array number is not specified for the indicated name. If this entry is not generated, and associated routine and variable names are not present, there is no restriction on the length of the name.*
<table>
<thead>
<tr>
<th>Generated for</th>
<th>Generated Elements</th>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Routines</td>
<td>C.event</td>
<td>Files events, whose priorities are declared in BREAK TIES statements, in the proper event set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D.event</td>
<td>Called when destroying event notice to check for set membership error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A.event</td>
<td>Called to list values of event notice attributes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event notice Records</th>
<th>Attributes</th>
<th>TIME.A</th>
<th>Time event is to occur</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attributes</td>
<td>EUNIT.A</td>
<td>Equals 0 for an endogenous event; equals input unit number (≠ 0) for an exogenous event</td>
</tr>
<tr>
<td></td>
<td>Attributes</td>
<td>P.EV.S</td>
<td>Pointer to predecessor event in the event set</td>
</tr>
<tr>
<td></td>
<td>Attributes</td>
<td>S.EV.S</td>
<td>Pointer to successor event in the event set</td>
</tr>
<tr>
<td></td>
<td>Attributes</td>
<td>M.EV.S</td>
<td>Set to 1 if the event is in the set; set to 0 if the event is not in the set</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Random Variables of RANDOM.E</th>
<th>Attributes</th>
<th>PROB.A</th>
<th>Probability value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attributes</td>
<td>IVALUE.A</td>
<td>Sample value: IVALUE.A contains an integer value;</td>
</tr>
<tr>
<td></td>
<td>Attributes</td>
<td>RVALUE.A</td>
<td>RVALUE.A contains a real value</td>
</tr>
<tr>
<td></td>
<td>Attributes</td>
<td>S.variable</td>
<td>Pointer to successor</td>
</tr>
</tbody>
</table>

84000450 A
<table>
<thead>
<tr>
<th>Generated for</th>
<th>Generated Elements</th>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets (5 unique characters)</td>
<td>Attributes of owner entities</td>
<td>F.set</td>
<td>Pointer to first entity in set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L.set</td>
<td>Pointer to last entity in set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N.set</td>
<td>No. of entities currently in the set</td>
</tr>
<tr>
<td></td>
<td>Attributes of member entities</td>
<td>P.set</td>
<td>Pointer to predecessor entity in set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>S.set</td>
<td>Pointer to successor in set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M.set</td>
<td>Equals 1 if the entity is in the set; equals 0 if the entity is not in the set</td>
</tr>
<tr>
<td>Routines</td>
<td>T.set</td>
<td>Files entity first or ranked in set</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U.set</td>
<td>Files entity last in set</td>
<td></td>
</tr>
<tr>
<td></td>
<td>V.set</td>
<td>Files entity before specified entity in set</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W.set</td>
<td>Files entity after specified entity in set</td>
<td></td>
</tr>
<tr>
<td></td>
<td>X.set</td>
<td>Removes first entity from set</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y.set</td>
<td>Removes last entity from set</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Z.set</td>
<td>Removes specific entity from set</td>
<td></td>
</tr>
</tbody>
</table>
SIMI15 COMPILATION CONTROL CARD

\{ SIMI15, P1, P2, P3, P4, P5, P6, P7, P8 \}

All the \( p_1 \) parameters are order independent.

- \( p_1 \) . Source input . absent
  \hspace{1cm} \text{INPUT assumed}
  \hspace{1cm} \text{Source input on file 1fn}
  \hspace{1cm} I = 1fn

- \( p_2 \) . Binary output
  . absent
  \hspace{1cm} \text{LGO assumed}
  \hspace{1cm} \text{Relocatable binary on file 1fn}
  \hspace{1cm} B = 1fn
  \hspace{1cm} No binary output
  \hspace{1cm} B = 0

- \( p_3 \) . Source listing
  . absent
  \hspace{1cm} \text{Source listing on file OUTPUT}
  \hspace{1cm} L = OUTPUT
  \hspace{1cm} L = 0
  \hspace{1cm} Suppresses listing output except for errors
  \hspace{1cm} Source listing on file 1fn
  \hspace{1cm} L = 1fn

- \( p_4 \) . Star, allstar and object listing
  . absent
  \hspace{1cm} \text{No listing}
  \hspace{1cm} S = 1fn
  \hspace{1cm} \text{Star listing on file 1fn}
  \hspace{1cm} O = 1fn
  \hspace{1cm} \text{Object listing on file 1fn}
  \hspace{1cm} A = 1fn
  \hspace{1cm} \text{Allstar listing on file 1 fn}
  \hspace{1cm} OS = 1fn
  \hspace{1cm} \text{Star and object listing on file 1 fn}
  \hspace{1cm} AO = 1 fn
  \hspace{1cm} \text{Allstar and object listing on file 1 fn}

- \( p_5 \) . Compilation options
  . OPT = f_1 f_2 \ldots f_n
  \hspace{1cm} \text{Generate line numbers for traceback}
  \hspace{1cm} fi = T
  \hspace{1cm} \text{Continue compilation even if errors are detected in the}
  \hspace{1cm} \text{PREAMBLE}
  \hspace{1cm} fi = P
  \hspace{1cm} \text{Do not abort the job if compilation errors are detected}
  \hspace{1cm} fi = E
  \hspace{1cm} \text{Print output at 8 lines/inch}
  \hspace{1cm} fi = 8
  \hspace{1cm} \text{Print output at 6 lines/inch}
  \hspace{1cm} fi = 6
  \hspace{1cm} \text{Suppress listing of the PREAMBLE}
  \hspace{1cm} fi = N
  \hspace{1cm} \text{Compress source listing}
  \hspace{1cm} fi = C
  \hspace{1cm} \text{Use small buffers for compilation}
  \hspace{1cm} fi = V

- \( p_6 \) . Last column
  . LC = n
  \hspace{1cm} \text{Last card column to be scanned by the compiler}
  \hspace{1cm} R = 1

- \( p_7 \) . Symbolic reference map
  . absent
  \hspace{1cm} \text{Local map, no line numbers}
  \hspace{1cm} R = 1
  \hspace{1cm} \text{Global map, no line numbers}
  \hspace{1cm} R = 2
  \hspace{1cm} \text{Local and global map, no line numbers}
  \hspace{1cm} R = 3
  \hspace{1cm} \text{Global map with line numbers}
  \hspace{1cm} R = 4
  \hspace{1cm} \text{Local map, no line numbers}
  \hspace{1cm} R = 5
  \hspace{1cm} \text{Global map with line numbers}
. R = 6   Local map with line numbers
. R = 7   Local map with line numbers
. R = 8   Global map, no line numbers
          Local and global map with
          line numbers
- p8    Structured       IF = NEW (i.e. new IF-ELSE-
    absent     ALWAYS construct)
          Unstructured IF
          IF = OLD

Examples:
SIMIIS.
- Source input on INPUT      — Abort job if errors in the
- Source listing on OUTPUT   — PREAMBLE or compilation
- Relocatable binary on LGO  — errors
- No star or assembly listings — Last column = 80
- No line numbers            — IF = NEW and R = 1
SIMIIS, I = COMPILE, LS = OUTPUT, OPT = PTE8, LC = 72.
- Source input on file COMPILE
- Source listing and star listing on file OUTPUT (Note: If all the list-
  ings are directed to the same file, the options can be merged)
- Continue compilation even if errors are detected in the PREAMBLE
- Include line numbers for traceback
- Execute the program even if compilation errors occurred
- Print the listings at 8 lines/inch
- Last column = 72.
SIMII5 EXECUTION CONTROL CARD

\{LGO.
\{LGO, f_1, f_2, \ldots f_n, p_1, p_2, p_3.\}\}

- \( f_1, f_2, \ldots f_n \) External file name substitution
  SIMUn = 1fn
  \( n = \) internal logical unit number \( 0 \leq n \leq 9 \)
  - even numbered units are carriage control files
  - odd numbered units are non-slew BCD files
  - any unit number can be used for binary files. (except 5 and 6)
  1fn = external file name
  FORTRAN file name definition
  TAPEn = 1fn
  FTN routines must be compiled with SYSEDIT = FILES
  All files used by FORTRAN must be declared.
  FORTRAN cannot share SIMSCRIPT files.

- \( p_1 \) - Print limit
  PL = n \( n \) is the maximum number of decimal lines to print on the OUTPUT file (default: 10000)

- \( p_2 \) - Dynamic storage
  DS = n \( n \) is the octal or decimal number of words reserved initially for dynamic storage. Assumed octal.
  (Ex: 4000 or 4000B, 10000D)

- \( p_3 \) - User parameters
  PARM = \( g_1, g_2, \ldots, g_n \);
  \( g_i \) represents either a single value or a group of values separated by an =, a + or a — sign.

Examples:
LGO.
- SIMU5 = INPUT
- SIMU6 = OUTPUT
- No FORTRAN files
- Print limit on output = 10000
LGO, SIMU5 = INPT, SIMU8 = REPORT, PL = 15000, DS = 20000B.
- Input on file INPT
- SIMU6 = OUTPUT
- Additional file SIMU8 = REPORT
- Print limit on OUTPUT = 15000 lines
- Initial dynamic storage = 20000B
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>:1</td>
<td>:</td>
<td>001</td>
<td>8 2</td>
<td>00</td>
<td>8 2</td>
<td>3A</td>
<td>6</td>
<td>6</td>
<td>41</td>
<td>6</td>
<td>06</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>01</td>
<td>12 1</td>
<td>61</td>
<td>12 1</td>
<td>41</td>
<td>7</td>
<td>7</td>
<td>42</td>
<td>7</td>
<td>07</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>02</td>
<td>12 2</td>
<td>62</td>
<td>12 2</td>
<td>42</td>
<td>8</td>
<td>8</td>
<td>43</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>03</td>
<td>12 3</td>
<td>63</td>
<td>12 3</td>
<td>43</td>
<td>9</td>
<td>9</td>
<td>44</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>04</td>
<td>12 4</td>
<td>64</td>
<td>12 4</td>
<td>44</td>
<td>*</td>
<td>*</td>
<td>45</td>
<td>12</td>
<td>60</td>
</tr>
<tr>
<td>E</td>
<td>E</td>
<td>05</td>
<td>12 5</td>
<td>65</td>
<td>12 5</td>
<td>45</td>
<td>*</td>
<td>*</td>
<td>46</td>
<td>11</td>
<td>40</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>06</td>
<td>12 6</td>
<td>66</td>
<td>12 6</td>
<td>46</td>
<td>*</td>
<td>*</td>
<td>47</td>
<td>11 8</td>
<td>54</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
<td>07</td>
<td>12 7</td>
<td>67</td>
<td>12 7</td>
<td>47</td>
<td>*</td>
<td>*</td>
<td>50</td>
<td>0 1</td>
<td>21</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
<td>08</td>
<td>12 8</td>
<td>68</td>
<td>12 8</td>
<td>48</td>
<td>*</td>
<td>*</td>
<td>51</td>
<td>8 0</td>
<td>34</td>
</tr>
<tr>
<td>I</td>
<td>I</td>
<td>09</td>
<td>12 9</td>
<td>69</td>
<td>12 9</td>
<td>49</td>
<td>*</td>
<td>*</td>
<td>52</td>
<td>12 8</td>
<td>74</td>
</tr>
<tr>
<td>J</td>
<td>J</td>
<td>10</td>
<td>11 0</td>
<td>70</td>
<td>11 0</td>
<td>50</td>
<td>*</td>
<td>*</td>
<td>53</td>
<td>11 8</td>
<td>53</td>
</tr>
<tr>
<td>K</td>
<td>K</td>
<td>11</td>
<td>11 1</td>
<td>71</td>
<td>11 1</td>
<td>51</td>
<td>*</td>
<td>*</td>
<td>54</td>
<td>8 3</td>
<td>13</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>12</td>
<td>11 2</td>
<td>72</td>
<td>11 2</td>
<td>52</td>
<td>*</td>
<td>*</td>
<td>55</td>
<td>no punch</td>
<td>20</td>
</tr>
<tr>
<td>M</td>
<td>M</td>
<td>13</td>
<td>11 3</td>
<td>73</td>
<td>11 3</td>
<td>53</td>
<td>*</td>
<td>*</td>
<td>56</td>
<td>0 8</td>
<td>32</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>14</td>
<td>11 4</td>
<td>74</td>
<td>11 4</td>
<td>54</td>
<td>*</td>
<td>*</td>
<td>57</td>
<td>12 8</td>
<td>73</td>
</tr>
<tr>
<td>O</td>
<td>O</td>
<td>15</td>
<td>11 5</td>
<td>75</td>
<td>11 5</td>
<td>55</td>
<td>*</td>
<td>*</td>
<td>58</td>
<td>8 0</td>
<td>36</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>16</td>
<td>11 6</td>
<td>76</td>
<td>11 6</td>
<td>56</td>
<td>*</td>
<td>*</td>
<td>59</td>
<td>0 8</td>
<td>17</td>
</tr>
<tr>
<td>Q</td>
<td>Q</td>
<td>17</td>
<td>11 7</td>
<td>77</td>
<td>11 7</td>
<td>57</td>
<td>*</td>
<td>*</td>
<td>60</td>
<td>11 8</td>
<td>55</td>
</tr>
<tr>
<td>R</td>
<td>R</td>
<td>18</td>
<td>11 8</td>
<td>78</td>
<td>11 8</td>
<td>58</td>
<td>*</td>
<td>*</td>
<td>61</td>
<td>8 0</td>
<td>37</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td>19</td>
<td>11 9</td>
<td>79</td>
<td>11 9</td>
<td>59</td>
<td>*</td>
<td>*</td>
<td>62</td>
<td>8 0</td>
<td>32</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>20</td>
<td>12 0</td>
<td>80</td>
<td>12 0</td>
<td>60</td>
<td>*</td>
<td>*</td>
<td>63</td>
<td>11 8</td>
<td>56</td>
</tr>
<tr>
<td>U</td>
<td>U</td>
<td>21</td>
<td>12 1</td>
<td>81</td>
<td>12 1</td>
<td>61</td>
<td>*</td>
<td>*</td>
<td>64</td>
<td>8 0</td>
<td>26</td>
</tr>
<tr>
<td>V</td>
<td>V</td>
<td>22</td>
<td>12 2</td>
<td>82</td>
<td>12 2</td>
<td>62</td>
<td>*</td>
<td>*</td>
<td>65</td>
<td>11 0 or 52</td>
<td>12</td>
</tr>
<tr>
<td>W</td>
<td>W</td>
<td>23</td>
<td>12 3</td>
<td>83</td>
<td>12 3</td>
<td>63</td>
<td>*</td>
<td>*</td>
<td>66</td>
<td>11 8-2111</td>
<td>11-0111</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>24</td>
<td>12 4</td>
<td>84</td>
<td>12 4</td>
<td>64</td>
<td>*</td>
<td>*</td>
<td>67</td>
<td>0 8</td>
<td>37</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>25</td>
<td>12 5</td>
<td>85</td>
<td>12 5</td>
<td>65</td>
<td>*</td>
<td>*</td>
<td>68</td>
<td>11 8-55</td>
<td>55</td>
</tr>
<tr>
<td>Z</td>
<td>Z</td>
<td>26</td>
<td>12 6</td>
<td>86</td>
<td>12 6</td>
<td>66</td>
<td>*</td>
<td>*</td>
<td>69</td>
<td>8 0</td>
<td>37</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>27</td>
<td>12 7</td>
<td>87</td>
<td>12 7</td>
<td>67</td>
<td>*</td>
<td>*</td>
<td>70</td>
<td>11 8-56</td>
<td>56</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>28</td>
<td>12 8</td>
<td>88</td>
<td>12 8</td>
<td>68</td>
<td>*</td>
<td>*</td>
<td>71</td>
<td>12 0 or 72</td>
<td>32</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>29</td>
<td>12 9</td>
<td>89</td>
<td>12 9</td>
<td>69</td>
<td>*</td>
<td>*</td>
<td>72</td>
<td>12 8-4</td>
<td>31</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>30</td>
<td>13 0</td>
<td>90</td>
<td>13 0</td>
<td>70</td>
<td>*</td>
<td>*</td>
<td>73</td>
<td>11 8-7</td>
<td>57</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>31</td>
<td>13 1</td>
<td>91</td>
<td>13 1</td>
<td>71</td>
<td>*</td>
<td>*</td>
<td>74</td>
<td>8 5</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>32</td>
<td>13 2</td>
<td>92</td>
<td>13 2</td>
<td>72</td>
<td>*</td>
<td>*</td>
<td>75</td>
<td>12 8-5</td>
<td>75</td>
</tr>
</tbody>
</table>

1 Twelve or more zero bits at the end of a 60-bit word are interpreted as end-of-line mark rather than two colons. End-of-line mark is converted to external BCD 1632.

2 In installations using the CDC 63 graphic set, display code 00 has no associated graphic or Hollerith code; display code 63 is the colon (8 2 punch).
The % graphic does not exist and translations from ASCII/EBCDIC % yield a blank (080).

3 The alternate Hollerith (02B) and ASCII (02B) punches are accepted for input only.