DIAGNOSTIC LANGUAGE — DL-1
SOFTWARE DESCRIPTION
1A PROCESSOR

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

The DL-1 statements described in this section are current with the CPR7 generic issue of the 1A Processor diagnostic programs.

1. GENERAL

A. Introduction

1.01 This section describes the Diagnostic Language (DL-1) and provides the following:

(a) Description of the basic structure of DL-1

(b) Description of statement format and definition of terms

(c) Detailed explanation of each DL-1 statement.

1.02 This section is reissued to:

(a) Add a description of the AP3BMSG statement

(b) Add a description of the EQUIPCHK statement

(c) Add a description of the I2MEMR statement

(d) Add a description of the I2MEMW statement

(e) Add a description of the I2PCLOOP statement

(f) Add a description of the LCKCODE statement

(g) Add a description of the LDSAR statement

(h) Add a description of the MPXHEAD statement

(i) Add a description of the RDXHEAD statement

(j) Add a description of the RDZREG statement

(k) Add a description of the SAPADDR statement

(l) Add a description of the TBLDELY statement

(m) Add a description of the TESTDMA statement

(n) Make minor changes as required for the CPR7 generic program. Revision arrows are used to emphasize the more significant changes.

1.03 Part 8 provides a list of abbreviations and acronyms with applicable terms used in this section.

B. Purpose of Diagnostic Language—DL-1

1.04 The DL-1 is a macro language that consists of many individual statements. When these DL-1 statements are assembled, the results are data table-driven diagnostic programs that direct diagnostic tests to be run on 1A Processor equipment.

2. BASIC STRUCTURE OF DL-1

2.01 The diagnostic programs run on the 1A Processor are, in general, based on repetitive execution of simple tests involving:

(a) Setting a location to a known value

(b) Reading the value of a location

(c) Comparing the read results with an expected value.

2.02 Most programs repeat the same type of test hundreds of times in diagnosis of a particular unit. The program instructions required to perform each differ only in the location address and the data to be read or written. Instead of repeating these instructions for each and every test, the unique portion of each, ie, addresses, data and expected results, are compiled in tables referred to as data tables. Only one set of instructions, called a task routine, is then provided in the program to execute all these types of tests.

2.03 The DL-1 macro language is used to generate these data tables. A DL-1 macro is a high-level statement which is expanded by the assembly program into a predefined data table format. In general, each DL-1 statement has an associated test routine. An example of a DL-1 statement and the data table which it produces is given in Fig. 1.

2.04 Figure 1 illustrates the IOREAD macro statement and its data table expansion as it appears in a program listing. In this example of the macro statement,

IOREAD OPER(LOOPDATA),DATA(O(26565113),MASK(1DG_ONES),EXPECT(O(26565113)),MTCPU

LOOPDATA represents a Datapool-defined operation code which will write the data (O(265655113)) and then read back the expected data (O(26565113)) through a mask of all ones (1DG_ONES). A complete description of the IOREAD statement is given in this section and is not repeated here. An explanation of each line of data is given in Fig. 1.

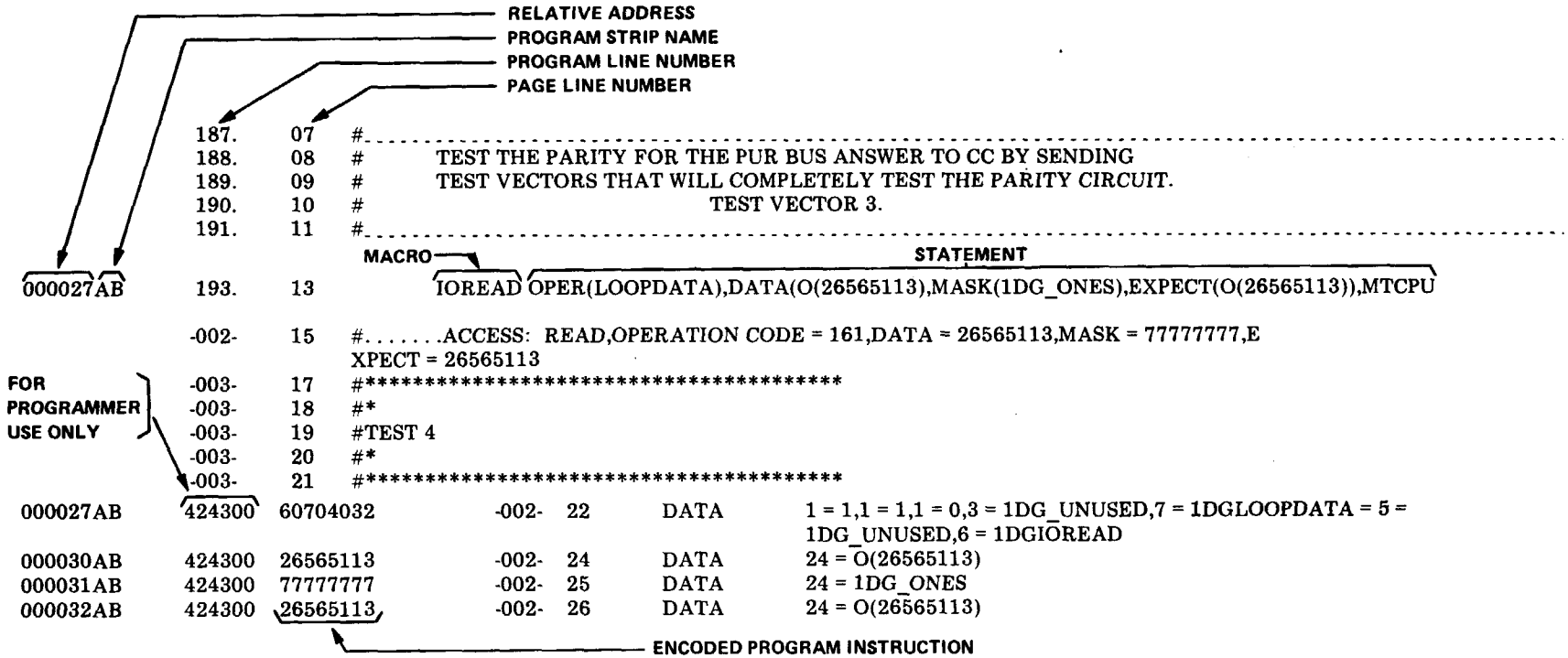
2.05 In the example of Fig. 1, the first word in the data table contains an index (octal 32) in the rightmost six bits (binary). This index is used by a control program to locate the IOREAD task routine responsible for executing the tests as given in the data table. This procedure is illustrated and explained in Fig. 2.

2.06 A description of diagnostic programs is given in Section 254-280-220, Diagnostic Programs—Description, 1A Processor.

2.07 There are two major classes of DL-1 statements:

(a) General purpose statements are independent of the unit being tested and include the following classes of statements:

(1) Internal data manipulation




```
                    187.    07  #-----------------------------------------------------------------------------
                    188.    08  #       TEST THE PARITY FOR THE PUR BUS ANSWER TO CC BY SENDING
                    189.    09  #       TEST VECTORS THAT WILL COMPLETELY TEST THE PARITY CIRCUIT.
                    190.    10  #                               TEST VECTOR 3.
                    191.    11  #-----------------------------------------------------------------------------
                                MACRO                   STATEMENT
000027AB            193.    13          IOREAD OPER(LOOPDATA),DATA(O(26565113),MASK(1DG_ONES),EXPECT(O(26565113)),MTCPU

                    -002-   15  #.......ACCESS:  READ,OPERATION CODE = 161,DATA = 26565113,MASK = 77777777,E
                                XPECT = 26565113
FOR                 -003-   17  #****************************************
PROGRAMMER          -003-   18  #*
USE ONLY            -003-   19  #TEST 4
                    -003-   20  #*
                    -003-   21  #****************************************
000027AB    424300  60704032        -002-   22      DATA    1 = 1,1 = 1,1 = 0,3 = 1DG_UNUSED,7 = 1DGLOOPDATA = 5 =
                                                            1DG_UNUSED,6 = 1DGIOREAD
000030AB    424300  26565113        -002-   24      DATA    24 = O(26565113)
000031AB    424300  77777777        -002-   25      DATA    24 = 1DG_ONES
000032AB    424300  26565113        -002-   26      DATA    24 = O(26565113)
```

Notes:

1. Page line numbers 07 through 11 are comments that describe the following program statement.
2. Page line number 13 is the statement which produces the data table. The actual data table expansion is shown on page line numbers 22 through 26. (These lines begin with the word DATA.)
3. Page line numbers 15 through 21 are comments showing the assigned test number.
4. Page line number 22 is the first word of the data table. It is the index word. The first bit (Bit 23) is a 1. The second bit (Bit 22) is a 1. The third bit (Bit 21) is a 0. The next 3 bits (Bits 20—18) are unused. The next 7 bits (Bits 17—11) are the operation (1DGLOOPDATA). The next 5 bits (Bits 10—6) are unused. The last 6 bits (Bits 5—0) are the index (1DGIOREAD).
5. Page line number 24 is the second word of the data table. For this example, it is 24 bits (Bits 23—0) of data.
6. Page line number 25 is the third word of the data table. For this example, it is the mask. It is 24 bits (Bit 23—0) of all 1s.
7. Page line number 26 is the fourth and last word of the data table. For this example, it is the expected result. It is 24 bits (Bits 23—0) of expected results.

Fig. 1—Example Data Table Expansion as it appears on a Program Listing

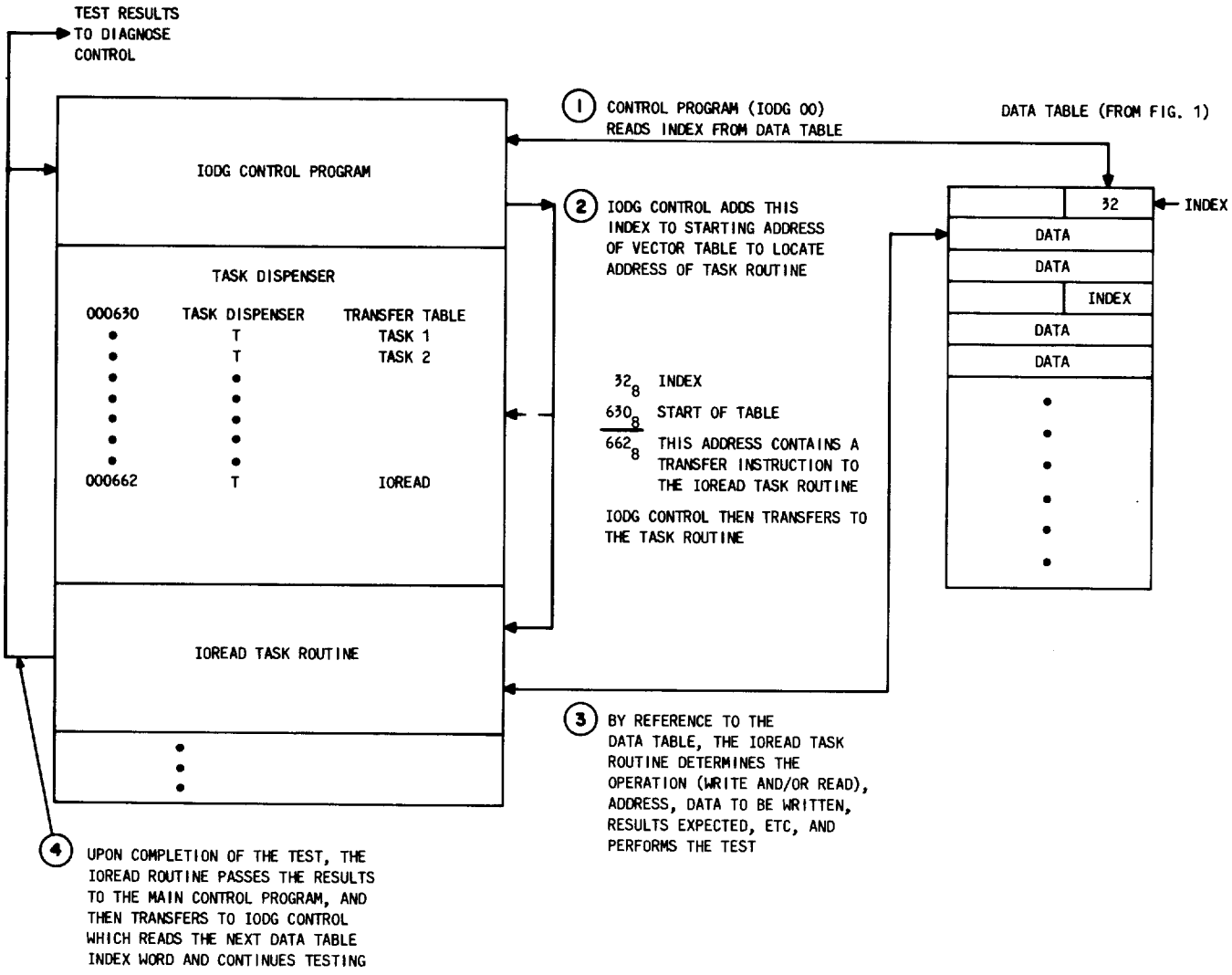


Fig. 2—Table-Driven Diagnostic Test Structure

(2) Control and decision

(3) Defining and calling subroutines.

(b) Testing statements specify the diagnostic test sequence to be applied to 1A Processor units.

3. STATEMENT FORMAT

A. Major Subfields of DL-1 Statement

3.01 A typical DL-1 statement contains four major subfields:

- Location field
- Operation field
- Variable (or operand) field
- Comment field.

3.02 The location field may contain a symbol to be referenced by other DL-1 statements; the symbol in the location field is also called a label. The operation field contains the name of the statement to be executed. The variable (or operand) field normally contains parameters and variables related to the operation. The comment field contains diagnostic design notes for the program listing user. The following are example statements as they may appear on program listings:

LOCATION	OPERATION	VARIABLE	COMMENT
JDEST	DTDEST		#EXAMPLE 1
	SUBCALL	SUBROUTINE (AUCCINIT_CC)	#EXAMPLE 2
	IOCONFIG	PUBR	#EXAMPLE 3
	ADSPULSE	ITEM (DS1CPMFF1),EXPECT(0)	#EXAMPLE 4

B. Variable Subfield Content

3.03 The variable subfield contains parameters and variables. Variables are data or addresses to be used in the statement. Some parameters have no associated variables. Variables associated with a parameter normally appear in parentheses following the parameter; in the statement format, variables are represented by a number preceded by a dollar mark ($). In the statement format:

ADSPULSE ITEM ($1),EXPECT($2)

The $1 and $2 represent variables for the ITEM and EXPECT parameters, respectively.

3.04 Choices of parameters appear in a vertical row in the appropriate position in the statement. An overscore over a parameter or variables indicates that the parameter or variables are optional.

Unless otherwise indicated, all numbers in this section are decimal.

3.05 Diagnostic programs make extensive use of symbolic names of Datapool-defined tables, words, and items in place of absolute numbers. Datapool-defined state names are also used. Definitions of these symbols are shown in the 1A Processor Datapool listing (PK-5A001).

C. Register Usage

3.06 Special use of general purpose registers are made by task routines invoked by DL-1 statements as follows:

F—The F register contains the base address of the scratch memory allocated to the diagnostic by the Maintenance Control Program (MACP).

Y—The Y register is the pointer to the next data table word.

G—The G register contains, upon entry to a task routine, the index word for the current data table entry.

D. Formatting Macros

3.07 Throughout this section many DL-1 statements will be provided which consist of more than one line. The example statements which require more than one line show the statement name immediately followed by an underscore, the last line beginning with ME (macro end), and the lines between the first and last line (if any) beginning with MC (macro continue). The underscore and the formatting macros MC and ME are not required for statements consisting of more than one line; they are used solely for formatting purposes to make the DL-1 statement easier to read. For more information, see Section 254-280-010, Datapool Documents — Description.

4. DETAILED EXPLANATION OF EACH DL-1 STATEMENT

A. Introduction

Classes of DL-1 Statements

4.01 Listed below are the classes of DL-1 statements that are described in the following paragraphs:

(a) Statement for Internal Manipulation of Data

(b) Statements for Control and Decision

(c) Statements to Define and Call Subroutines

(d) Statements for Testing.

Format for the Explanation of Each DL-1 Statement

4.02 The detailed explanation of each DL-1 statement is given alphabetically within the class of statements to which the DL-1 statement belongs. The description of each statement includes:

(a) The function of the statement

(b) The format of the statement

(c) The characteristics of the parameters in the statement

(d) Examples as they may appear on a diagnostic phase program listing with a brief explanation.

4.03 Some statements have more than one format shown. This is because the parameters in one format do not apply to another.

B. Statements for Internal Manipulation of Data

ARITH

4.04 The description of the ARITH statement includes:

Function:

The ARITH statement is used for data manipulation of two arguments (X and Y) by a function; the result is stored at a specified destination.

Format:

ARITH DESTINATION,ARGX,FUNCTION,ARGY

Characteristics of Parameters:

DESTINATION — Specifies a destination. The DESTINATION is one of the following:

WORD($1)
WORD(IND($1))
WORD($2)
WORD($1,IND($1))
SWORD($1)
SWORD(IND($1))
SWORD($1,IND($1))
DTWORD($1)
DTWORD($2)
SITEM($3)
SITEM($4($5))

WORD — Specifies an absolute location for a destination. The $1 is a label; $2 is a list of labels. The IND specifies indirect access via call store scratch location. The base address (in register F) is added to $1 value. A maximum of four labels and/or IND (label)s may appear.

SWORD — Specifies a destination in a call store scratch location. $1 is a label. The base address (in register F) is added to $1 value. The IND specifies indirect access via call store scratch location. The address equals the contents of the location at "$1 plus the base address (in register F)" added to the base address (in register F). A maximum of four labels and/or IND(labels)s may appear.

DTWORD — Specifies a destination in a data table location. The $1 is a label and $2 is a list of labels (4 maximum).

SITEM — Specifies a destination in a call store scratch location defined by Datapool item plus the base address (in register F). Insertion masking is per item. The $3 is an item name. The $4 is a list of a maximum of four item names followed by ($5), the number 1, 2, 3, or 4 in parentheses. If $4 is only one item ($5) does not appear. The value is left-adjusted to the item's displacement.

ARGX — Specifies the X argument. Argument X is one of the following:

DTWORD($1)
WORD($1)
WORD(IND($1))
SWORD($1)
SWORD(IND($1))
ITEM($3)
SITEM($3)
ITEMS($4)
SITEMS($4)
LIT($5)

DTWORD — Specifies an argument in a data table location. The $1 is a label.

WORD — Specifies an absolute location containing the argument. The $1 is a label. The IND specifies indirect access via a call store scratch location. The base address (in register F) is added to the $1 value to determine the location of the address of the argument.

SWORD — Specifies an argument in call store scratch location. The $1 is a label. The base address (in register F) is added to $1 value. The IND specifies indirect access via a call store scratch location. The address equals the contents of the location at "$1 plus the base address (in register F)" added to the base address (in register F).

ITEM — Specifies an argument in an absolute location specified by a Datapool item. The $3 is an item name. The value is right-adjusted before being used.

SITEM — Specifies an argument in a call store scratch location defined by Datapool item plus the base address (in register F). The $3 is an item name. The value is right-adjusted before being used.

ITEMS — Specifies an argument in an absolute location defined by Datapool items. The $4 is a list of items all in the same word. ITEMS is used only with the XOR, OR, and AND functions.

SITEMS — Specifies an argument in a call store scratch location defined by Datapool items. The base address (in register F) is added. The $4 is a list of items all in the same word. The SITEMS is used only with the XOR, OR and AND functions.

LIT — Specifies a fixed quantity to be used as an argument. The $5 is any arithmetic expression to be used as the argument.

FUNCTION — Specifies the function to be performed on ARGX and ARGY, the resultant of which is stored in the specified DESTINATION. Function is one of the following:

ADD — ARGX plus ARGY

SUB — ARGX minus ARGY

SHIFT — Shift ARGX left ARGY bit positions (the value of ARGY is 0 through 23)

ROT — Rotate ARGX per ARGY

XOR — Exclusive-OR ARGX with ARGY

OR — Union ARGX with ARGY

AND — Product ARGX with ARGY

SROT — 16-bit rotate ARGX left ARGY bit positions

MOVE — Move ARGX to destination.(If function is not specified, it is MOVE).

ARGY — Specifies the Y argument. The Y argument may be any one of the choices listed under ARGX. (ARGY does not appear on MOVE function.)

Examples:

(a) ARITH SITEM(DG1FSDFEQB),WORD(DG1AUAXA1),SHIFT,LIT(-H(XL1FSGQSEMO))

This statement causes the contents of location DG1AUAXA1 to be shifted right by the displacement of XL1FSGQSEMO and insertion masked into call store scratch item DG1FSDFEQB.

(b) ARITH SWORD(DG1AUSCR2),SWORD(DG1AUSCR1),ADD,SWORD(DF1AUSCR2)

This statement causes the contents of the call store scratch location DG1AUSCR1 to be used as the contents of the call store scratch location DG1AUSCR2 and placed in the call store scratch location DG1AUSCR2.

(c) ARITH SWORD(DG1COMTC,IND(DG1KODE)),SWORD(IND(DG1AUSCR1)),SUB,LIT(7)

This statement causes the contents of the call store scratch location at the address calculated by "the contents of DG1AUSCR1 plus the base address (in regular F)" to have subtracted from it the constant seven and placed in two call store scratch locations: one call store scratch location is at address DG1COMTC and the address of the other call store scratch location is given by "the contents of DG1KODE plus the base address (in register F)."

(d) ARITH SITEM(DG1FSDFEQB),SWORD(IND(DG1AUAXA1)),SHIFT,
LIT(-H(XL1FSGQSMEO))

This statement causes the contents of the call store scratch location at the address calculated by "the contents of DG1AUAXA1 plus the base address (in register F)" to be shifted right by the displacement of XL1FSGQSEMO and insertion masked into the call store scratch item DG1FSDFEQB.

(e) ARITH SWORD(DG1AUSCR2),SWORD(DG1AUSCR1),ROT,LIT(3)

This statement causes the contents of call store scratch location DG1AUSCR1 to be rotated left 3 positions and inserted in the call store scratch item DG1AUSCR2.

(f) ARITH DTWORD(ARDT),SITEM(DG1AUKCODE),XOR,LIT(A(FS10AR)(AU1START))

This statement causes the contents of the call store scratch location designated by Datapool item DG1AUKCODE to be exclusive-ORed with "the address of FS10AR ORed with the value of AU1START" and then to be placed in the data table word at address ARDT.

(g) ARITH SWORD(DG1KODE),SWORD(DG1KODE),OR,LIT(—M (IO1ECTR))

This statement causes the contents of the call store scratch location at address DG1KODE to be ORed with the mask of the IO1ECTR item and placed in the call store scratch location DG1KODE.

(h) ARITH SWORD(DG1KODE),SWORD(DG1KODE),AND,LIT(—MIO1ECTR))

This statement causes the contents of the call store scratch location at address DG1KODE to be ANDed with the complement of the mask of the IO1ECTR item and placed in the same call store scratch location DG1KODE.

(i) ARITH SITEM(DG1FSDMFSM0(1),DG1FSDFSM1(2)),ITEM(XL1FSDFSQD),SROT,
LIT(H(FS1FSQO))

This statement causes rotate-16 the displacement of the FS1FSQO item bit positions action on the XL1FSDFSQD item. The result is then insertion masked into the call store scratch items DG1FSDMFSM0 and DG1FSDFSM1.

(j) ARITH SITEM(DG1ITEMA(1),DG1ITEMB(2),DG1ITEMC(3)),ITEM(DG1ABITEMD),MOVE

This statement causes item DG1ABITEMD to be insertion masked into the call store scratch items DG1ITEMA, DG1ITEMB, and DG1ITEMC. The same function would be performed if the MOVE were omitted. Any ARITH statement which has no function and no ARGY is a MOVE.

DL1COLLAPSE

4.05 The description of the DL1COLLAPSE statement includes:

Function:

The DL1COLLAPSE statement is used to collapse raw data results from the specified tests into one data word.

Format:

```
                BEGIN    LAMP
DL1COLLAPSE     END  ,   PHYSICAL
                         BOTH
```

Characteristics of Parameters:

BEGIN — Indicates the beginning point of the tests to be collapsed.

END — Indicates the termination point of the tests to be collapsed.

LAMP — The tests are to be collapsed for LAMP fault data processing only.

PHYSICAL — The tests are to be collapsed for PHYSICAL fault data processing insertion only.

BOTH — The tests are to be collapsed for both LAMP and PHYSICAL. If none is specified, it will default to BOTH.

Example:

```
DL1COLLAPSE  BEGIN, LAMP
```

DL1DELETE

4.06 The description of the DL1DELETE statement includes:

Function:

The DL1DELETE statement is used to exclude raw data entries from being included in the trouble location process.

Format:

```
                BEGIN    LAMP
DL1DELETE       END  ,   PHYSICAL
                         BOTH
```

Characteristics of Parameters:

BEGIN — Indicates the beginning point of the deletion.

END — Indicates the end point of the deletion.

LAMP — The tests are to be deleted from the processing LAMP fault data only.

PHYSICAL — The tests are to be deleted from the processing of the PHYSICAL fault data.

BOTH — The tests are to be deleted for both LAMP and PHYSICAL. If nothing is specified, it will default to BOTH.

Example:

DL1DELETE BEGIN, PHYSICAL

C. Statements for Control and Decision

DELAY10μ

4.07 The description of the DELAY10μ statement includes:

Function:

The DELAY10μ statement is used to obtain units of delay in 10-microsecond increments. A delay in this form should not exceed 2.5 milliseconds.

Format:

DELAY10μ $1

Characteristics of Parameters:

$1 — Specifies a delay in 10-microsecond increments. Should not exceed 250 due to the possibility of an interrupt.

Example:

DELAY10μ 150

This statement will result in delaying 1500 microseconds or 1.5 milliseconds.

DL1ETERM

4.08 The description of the DL1ETERM statement includes:

Function:

The DL1ETERM statement allows the diagnostic to terminate early as the result of certain test failures. This statement can be used in conjunction with the DL1TFZAP macro.

Format:

$$\text{DL1ETERM} \quad \left\{\begin{matrix}\text{TESTFAIL}\\ \text{ANYFAIL}\\ \text{FAILURE}\end{matrix}\right\} \quad \left\{\begin{matrix}\text{DIAGEND}\\ \text{LABEL (LOC)}\end{matrix}\right\} \quad [,\text{YESTERM}]$$

Characteristics of Parameters:

TESTFAIL — Used when it is desired to terminate if a failure occurs after a DL1TFZAP was issued.

ANYFAIL — Terminate if any failure occurs in the diagnostic.

FAILURE — Terminate if a failure occurs during this phase.

DIAGEND — If terminating due to a failure, the equivalent of a SEGEND will be executed followed by a jump to the end of the phase, and then the diagnostic is terminated.

LABEL (LOC) — If terminating, a jump to this location is made. The code at this location must be code to restore the unit to a safe state followed by SEGEND and PHASEEND macros.

YESTERM — If specified, this parameter ensures that a termination due to DL1ETERM will occur if a failure has been detected. This will override the "UCL" on a DGN message.

Example:

```
DL1ETERM     FAILURE,DIAGEND
```

This statement will terminate the diagnostic if a failure has occurred up to the point in the phase that this statement exists.

DL1MTSKIP

4.09 The description of the DL1MTSKIP statement includes:

Function:

The DL1MTSKIP statement is used to circumvent program code in accordance with the Laboratories Design Information (LDI) value of the frame. Thus, one diagnostic program may be used on various frame LDI issues with only the appropriate code being executed.

Format:

```
DL1MTSKIP     LABEL,EQMTVAL ($1)
                    NEMTVAL ($1)
```

Characteristics of Parameters:

LABEL — The label to which the jump is made.

EQMTVAL — Jump to LABEL if LDI value is equal to any of the $1 values.

NEMTVAL — Jump to LABEL if LDI value is not equal to any of the $1 values.

$1 — May be one or a string of LDI symbols.

Example:

```
DL1MTSKIP     LABEL = END,EQMTVAL (1XLMTDUS1A)
```

This statement will jump around the code for DUS LDI Issue 1A only, making the diagnostic forward compatible.

DL1SKPTST

4.10 The description of the DL1SKPTST statement includes:

Function:

The DL1SKPTST statement is used to skip tests in a diagnostic but maintain the proper total test count, for transmission level point purposes, as though all tests were executed.

Format:

```
DL1SKPTST     SKIP = $1
```

Characteristics of Parameters:

SKIP = $1 — This parameter indicates the number ($1) of tests to be skipped, and causes a cumulative test counter to be incremented by this number.

Example:

```
DL1SKPTST     SKIP = 4
```

This statement causes the next four tests to be skipped and increments the test counter accordingly.

DL1TFZAP

4.11 The description of the DL1TFZAP statement includes:

Function:

The DL1TFZAP statement clears the failure indicator examined when DL1ETERM TESTFAIL is specified. This provides control over the period during which a failure is to be observed; ie, between the DL1TFZAP statement and the DL1ETERM TESTFAIL statement.

Format:

DL1TFZAP

Example:

DL1TFZAP

This statement will clear the TESTFAIL indicator (DIGITSFAIL) which is marked by the Diagnostic Control Program (DCON) when a failure occurs.

DTDEST

4.12 The description of the DTDEST statement includes:

Function:

The DTDEST statement specifies a location as the forward destination of a DTJUMP statement.

Format:

LAB DTDEST

Characteristics of Parameters:

LAB — Specifies location field symbolic name as DTJUMP destination.

Example:

NODF01 DTDEST

This statement provides a destination for a previous DTJUMP statement.

DTJUMP

4.13 The description of the DTJUMP statement includes:

Function:

The DTJUMP statement is used to conditionally or unconditionally skip around a strip of test (data table words) to another location (forward) in the data table.

Format:

$$\text{DTJUMP LABEL(\$1),IF(\$3(F),}\overline{\text{\$2}}\text{,MASK(\$4),\$5,\$6)}$$

Characteristics of Parameters:

LABEL — Specifies the destination of the jump to a symbolic labeled data table location $1. The $1 appears in the location field of a subsequent DTDEST statement.

IF — Specifies location contents to be compared to $6 per logical condition $5. The jump is executed if the tested condition is true; otherwise, the next data table task block is executed. The $2 is an absolute location or an EXTERN vector name. The $3 is a call store scratch word or item. The $3 is followed by (F) to indicate that $3 will be indexed by the base address.

MASK — Optionally specifies a mask for the location. The $4 is any arithmetic expression which expresses the mask.

$5 — Specifies one of the logical conditions of the control flip-flops: AZ, AU, GT, GE, LT, LE as the IF condition to be true for execution of the jump.

$6 — Specifies an arithmetic expression to be compared to the contents of the location per $5.

Example:

DTJUMP LABEL(NODF01),IF(DG1FSEQSME0(F),AZ,0)

This statement causes a jump to the label NODF01 (which is on a subsequent DTDEST statement) if the contents of DG1FSEQSME0 indexed by the base address (in register F) compared to 0 is arithmetic zero.

PHASEEND

4.14 The description of the PHASEEND statement includes:

Function:

The PHASEEND statement clears the auxiliary unit bus lock, checks that all of the tests in the phase were either executed or properly skipped via DL1SKPTST (if not, the diagnostic is aborted), and terminates the phase.

Format:

LABEL PHASEEND PHASE ($1)

Characteristics of Parameters:

LABEL — Pident name followed by "E."

PHASE ($1) — The number ($1) of the phase being executed.

Example:

CCDG01E PHASEEND PHASE (01)

This statement ends phase 1 of the central control diagnostic.

PHASEINIT

4.15 The description of the PHASEINIT statement includes:

Function:

The PHASEINIT statement clears the PEST (interrupt inhibit) control words, clears the auxiliary unit bus lock flag, loads the stack address used by DTJUMP macros, turns off auto-segmenting, and turns on manual segmenting. This statement also performs standard initialization on a unit basis.

Format:

LABEL PHASEINIT PHASE($1), NOLAMP,NOPHYS

Characteristics of Parameters:

LABEL — Pident name.

PHASE($1) — Specifies the phase of the diagnostic in which this statement is. The $1 is the phase number.

NOLAMP — When specified, indicates transmission level point will not search a LAMP fault data file for this phase.

NOPHYS — When specified, indicates transmission level point will not search a PHYSICAL fault data file for this phase.

Example:

CCDG01 PHASEINIT PHASE(01)

This statement will perform basic initialization for phase 1 of the central control diagnostic.

SEGEND

4.16 The description of the SEGEND statement includes:

Function:

The SEGEND statement checks that the next statement is either SEGINIT or PHASEEND, and if not, it aborts the diagnostic. Otherwise, a segment break is taken and requests delays, if specified.

Format:

```
          6SEC($1)
          SEC($2)
SEGEND    MSEC($3)
```

Characteristics of Parameters:

6SEC($1) — Requests $1 6-second delays.

SEC($2) — Requests $2 1-second delays.

MSEC($3) — Requests $3 100-millisecond delays.

Example:

SEGEND MSEC(1)

This statement takes a segment break and requests that a new segment is not started for at least 100 milliseconds.

SEGINIT

4.17 The description of the SEGINIT statement includes:

Function:

This statement requests special services, such as auxiliary unit soft stops, auxiliary unit hard stops, inhibit of various interrupts, for the next segment. It also performs common initialization on a per unit basis and certain audits. An example of the latter is the checking of routing flip-flops in the Central Control Diagnostic Control Program (CCDG) to ensure proper state.

Format:

SEGINIT INHAU = $1, STOPAU = $2, INHINT = $3,

LPBND = $4, AUBUS (REQUEST)

Characteristic of Parameters:

INHAU = $1 — If $1 is Y, this will request an inhibit on the auxiliary unit bus request (auxiliary unit soft stop).

STOPAU = $2 — If $2 is Y, this will request a stop of all active auxiliary units, and also stops the auxiliary unit sequencer in the central control (hard stop). ⧫It also inhibits K-code match and auxiliary unit bus drivers (trouble PIF).⧫

INHINT = $3 — If $3 is Y, this will set the normal PESTS (interrupt inhibit) for this unit. Default inhibits only G, H, and J level interrupts.

LPBND = $4 — If $4 is Y, this indicates that this is a valid loop boundary when using the EX mode, and a corresponding entry will be made in the valid loop boundary table at the end of the phase.

AUBUS (REQUEST) — If this parameter is used, the first three parameters will automatically be set to "Y," and an auxiliary unit lock request is made.

Example:

SEGINIT INHINT = Y, LPBND = N

This statement will set the PESTS for this unit type and also indicates that this is not a valid loop boundary.

D. Statements to Define and Call Subroutines

DL1SUB

4.18 The description of the DL1SUB statement includes:

Function:

The DL1SUB statement starts a definition of a data table subroutine. The statement is followed by the statements which define the operation of the subroutine. These statements are followed by a SUBTN statement which signals the end of the subroutine definition. The DL1SUB statement automatically generates a DTJUMP statement to jump around the subroutine to isolate it from straight line execution.

Format:

NAME DL1SUB

Characteristics of Parameters:

NAME — Specifies a location field to be referred to by a SUBCALL statement used to call the subroutine.

Example:

AUINIT_CC DL1SUB

This statement defines the beginning of a subroutine named AUINIT_CC.

SUBCALL

4.19 The description of the SUBCALL statement includes:

Function:

The SUBCALL statement conditionally or unconditionally calls a defined data table subroutine. The subroutine is defined with the DL1SUB header statement followed by a DL-1 statement and terminated by the SUBRTN statement. Following execution, subroutine control is returned to the next statement following the SUBCALL statement.

Format:

$$\text{SUBCALL SUBROUTINE(\$1),IF(\$3(F),}\overbrace{\text{MASK(\$4),\$5,\$6)}}^{\$2}$$

Characteristics of Parameters:

SUBROUTINE — Specifies the label of a data table subroutine, that is a DTLABEL to be jumped to. The $1 is the name of the subroutine.

IF — Specifies location contents to be compared to $6 per logical condition $5. The jump is executed if the tested condition is true; otherwise, the next data table task block is executed. The $2 is an absolute location or an EXTERN vector name. The $3 is a call store scratch item or word. The $3 is followed by (F) to indicate that $3 will be indexed by the base address.

MASK — Optionally specifies a mask for the location. The $4 is any arithmetic expression which expresses the mask.

$5 — Specifies one of the logical conditions of the control flip-flops: AZ, AU, GT, GE, LT, LE as the IF condition to be true for execution of the jump.

$6 — Specifies an arithmetic expression to be compared to the contents of the location per $5.

Example:

SUBCALL SUBROUTINE(AUCCINIT_CC)

This statement causes an unconditional transfer to the subroutine AUCCINIT_CC.

SUBRTN

4.20 The description of the SUBRTN statement includes:

Function:

The SUBRTN statement specifies the termination of a subroutine and generates a return to the main flow.

Format:

SUBRTN

Characteristics of Parameters:

This statement has no parameters.

Example:

SUBRTN

This statement specifies the termination of a subroutine.

E. Statements for Testing

ADSPULSE

4.21 The description of the ADSPULSE statement includes:

Function:

The ADSPULSE statement control pulses the auxiliary data system (ADS) under test.

Format:

ADSPULSE NOSTORE

ADSPULSE ITEM($1) ITEMS($2),EXPECT($3),KCODE($4)

Characteristics of Parameters:

NOSTORE — If specified, the ADS is control pulsed, and the results sent back on the auxiliary unit reply bus are ignored.

ITEM — Generates a mask for the results pulsed back from the ADS. The $1 is an item name.

ITEMS — Generates a mask for the results pulsed back from the ADS. The $2 is a list of item names all in the same word.

EXPECT — Specifies the result the ADS should send on the auxiliary unit reply bus (ie, data of concern after masking.) The $3 is any arithmetic expression which expresses the expected result.

KCODE — Specifies which data unit selector (DUS) is being written into and supplies bits 11 through 15 of the auxiliary unit address. If not specified, the KCODE supplied by translations is inserted. The $4 is the KCODE.

Example:

ADSPULSE ITEM(DS1CPMFF1),EXPECT(0)

This statement control pulses the ADS under test. The DS1CPMFF1 item is the mask for the results pulsed back and the expected result is zero.

ADSREAD

4.22 The description of the ADSREAD statement includes:

Function:

The ADSREAD statement is a general purpose read for the ADS.

Format:

```
          ITEM($1) ,NOSTORE
ADSREAD ITEMS($2),EXPECT($4),OPTIONS($5),KCODE($6),ACODE($7)
          WORD($3)
```

Characteristics of Parameters:

ITEM — Specifies the location being read. This parameter supplies bits 0 through 10 of the auxiliary unit address bus if ACODE is not specified, else bits 0 through 5. The mask for the expected results is also generated from this parameter. The $1 is the item name.

ITEMS — Specifies the location being read. This parameter supplies bits 0 through 10 of the auxiliary unit address bus if ACODE is not specified, else bits 0 through 5. The mask for the expected results is generated from this parameter. The $2 is list of items all in the same word.

WORD — Specifies the location being read. This parameter supplies bits 0 through 10 for the auxiliary unit address bus, if ACODE is not specified, else bits 0 through 5. The mask for the expected result is also generated from this parameter. The $3 is the address of the word.

NOSTORE — If specified, nothing is done with the data read from the ADS.

EXPECT — If specified, this is the result the ADS should send on the auxiliary unit reply bus (ie, data of concern after masking). The $4 is any arithmetic expression.

OPTIONS — Specifies the options associated with the maintenance load (ML) assembly language instruction. This parameter supplies bits 16 through 18 of the auxiliary unit address bus. The ML instruction is described in Section 254-280-020, 1A Processor Assembly Language — Description, 1A Processor. The $5 is the list of options. Table A is a list of options for the ML instructions.

KCODE — Specifies which DUS is being read and supplies bits 11 through 15 of the auxiliary unit address bus. If not specified, the KCODE supplied by translations is inserted. The $6 is the KCODE.

ACODE — Specifies which data unit controller (DUC) is being read. This parameter supplies bits 6 through 9 of the auxiliary unit address bus and sets bit 10 = 1. The $7 is the ACODE. If $7 is UUT, the ACODE supplied by translations is used.

Example:

```
ADSREAD_ITEMS(TU1CCEQZ,TU1LRCFL),EXPECT(M(TU1CCEQZ)),KCODE(O(15)),
ME          ACODE(0(12))
```

This statement causes a read of the ADS. The items TU1CCEQZ and TU1LRCFL specify bits 0 through 5 of the auxiliary unit address bus (the location to be read). The KCODE supplies bits 11 through 15 of the auxiliary unit address bus (DUS). The ACODE supplies bits 6 through 9 of the auxiliary unit address bus (DUC) and sets bit 10 to 1. The EXPECT parameter specifies the result expected on the auxiliary unit reply bus after the read.

TABLE A

OPTIONS FOR THE ML AND MS INSTRUCTIONS

MS/ML OPTION	EXPLANATION
C	Control Mode
M	Maintenance Mode
R	Read
W	Write
IPKA	Invert Address parity
IST(IT3,3T5E, 3T5L,4T6)	Inhibit store timing, only one pulse at a time
NGCP	Do not execute two GCPs before instruction (STRDGCP, STWRGCP, and STWRSTAT statements only)
DELAY	Introduce a 10-microsecond delay between MS/ML instruction and generate control pulse (GCP) (STRDGCP, STWRGCP, and STWRSTAT statements only)
MS ONLY OPTION	**EXPLANATION**
IP1	Invert data parity bit P1
IP2	Invert data parity bit P2
IWE	Inhibit write enable
ML ONLY OPTION	**EXPLANATION**
READ2	Execute a second ML instruction (STAREAD statement only)

ADSWRITE

4.23 The description of the ADSWRITE statement includes:

Function:

The ADSWRITE statement is a general-purpose write for the ADS.

Format:

ADSWRITE WORD($1),DATA($2),OPTIONS($3),KCODE($4),ACODE($5)

Characteristics of Parameters:

WORD — The location being written into. If ACODE is not specified, this parameter supplies 0 through 10 for the auxiliary unit address bus. If ACODE is specified, it supplies bits 0 through 5 for the auxiliary unit address bus. The $1 is the address.

DATA — Up to 24 bits of data to be written into the location specified by WORD. This parameter supplies bits 0 through 23 of the auxiliary unit write bus. The $2 is any arithmetic expression which expresses the data.

OPTIONS — These are the options associated with the maintenance store (MS) 1A Processor Assembly language instruction. This parameter supplies bits 16 through 18 of the auxiliary unit address bus. The MS instruction is described in Section 254-280-020 (Assembly Language—Description 1A Processor). The $3 is a list of options. Table A is a list of options for the MS instruction.

KCODE — Specifies which DUS is being written into and supplies bits 11 through 15 of the auxiliary unit address bus. If not specified, the KCODE supplied by translations is inserted. The $4 is the KCODE.

ACODE — Specifies which DUC is being written into. This parameter supplies bits 6 through 9 of the auxiliary unit address bus and sets bit 10 = 1. The $5 is the ACODE. If $5 is UUT, the ACODE supplied by translations is used.

Example:

ADSWRITE WORD(TU1DR),DATA(1DGBIT15),OPTIONS(C,W),KCODE(O(21)),ACODE(UUT)

This statement causes a write to the ADS. The WORD parameter supplies bits 0 through 5 of the auxiliary unit address bus (the location being written into). The value of 1DGBIT15 will be written into TU1DR. The options parameter supplies bits 16 through 18 of the auxiliary unit address bus. The KCODE supplies bits 11 through 15 of the AU address bus (DUS). The ACODE specifies that bits 6 through 9 of the auxiliary unit address (DUC) be supplied from translation and sets bit 10 to 1.

AP3BMSG

4.24 The description of the AP3BMSG statement includes:

Function:

The AP3BMSG statement will request the 3B executed attached processor interface (API) diagnostics and retrieve the results of the diagnostic. This is done by sending a message to the 3B via the active API. A 3B process will be started in response to the message. This process will request the running of the 3B diagnostics. At the completion of the 3B diagnostics, the process will return the results to the 1A Processor. The results are passed in one word with a specified bit set for the first failing phase.

Format:

```
          TSTMSG
AP3BMSG   RESULTS
```

Characteristics of Parameters:

TSTMSG — Specifies that a message is to be sent to the 3B processor which starts the running of the 3B executed API diagnostics.

RESULTS — Specifies that the result word of the 3B executed API diagnostics is to be read. This word has a specified bit set for the first failing phase. Bit 1 set = Phase 1 failed, Bit 2 set = Phase 2 failed, ..., Bit 12 set = Phase 12 failed.

Example:

AP3BMSG TSTMSG

This statement will send a message to the 3B processor to begin execution of the 3B executed API diagnostics.

AP3BMSG RESULTS

This statement reads the results word sent to the 1A from the 3B. The word shows the results of the 3B executed API diagnostics.

AUBRQ

4.25 The description of the AUBRQ statement includes:

Function:

This AUBRQ statement assures an auxiliary unit diagnostic that the auxiliary unit community will not be softstopped or hardstopped while the diagnostic is taking a segment break. When the auxiliary units are softstopped, the bus requests are inhibited in central control denying auxiliary units the use of the auxiliary unit bus for autonomous data transfers. When the auxiliary units are hardstopped, the auxiliary unit sequencer in central control is stopped, also denying auxiliary units the use of the auxiliary unit bus.

Format:

AUBRQ

Characteristics of Parameters:

This statement has no parameters.

Example:

AUBRQ

This statement permits an auxiliary unit diagnostic to take a segment break and not have the auxiliary unit bus inhibited during that time.

AUGCPCLR

4.26 The description of the AUGCPCLR statement includes:

Function:

The AUGCPCLR statement control pulses an auxiliary unit initializing various circuits and sending a 1-bit reply to central control on bit 20. The results may be passed to the DCON program or ignored.

Format:

AUGCPCLR EXPECT(1DGBIT20)

Characteristics of Parameters:

EXPECT — Specifies the expected result.

Examples:

(a) AUGCPCLR

This statement control pulses the auxiliary unit under diagnosis initializing the frame. The frame response is not of interest.

(b) AUGCPCLR EXPECT(1DGBIT20)

This statement control pulses the auxiliary unit under diagnosis initializing the frame and expects a response on bit 20.

AUKCRDCC

4.27 The description of the AUKCRDCC statement includes:

Function:

The AUKCRDCC statement reads one of the 16-bit auxiliary unit registers in the active or standby central control. The DG1AUPOS indexed by the base address (in register F) contains the bit position corresponding to the auxiliary unit under test. Optionally, the auxiliary unit maintenance request bit 0/1 (AUMRQ0-1) or auxiliary unit operational interject bit 0/1 (AUOI0-1) may be read. The read bus is calculated from DG1AUPOS: Even positions read on bus 0, odd positions on bus 1. Any failing result is passed to the DCON in bit position 0.

Format:

AUKCRDCC ITEM($1),EXPECT($2)

Characteristics of Parameters:

ITEM — Specifies the auxiliary unit request register to be read. The $1 is the symbolic name of any auxiliary unit request register in the active or standby central control.

EXPECT — Specifies the expected result. The $1 is 0 or 1.

Examples:

AUKCRDCC ITEM(ST1EVG),EXPECT(0)

This statement reads the auxiliary unit enable verify expect group register (ST1EVG) in the standby central control. The read bus is calculated from DG1AUPOS. The result is compared to zero.

AUKCRDCC ITEM(1N1AUOI),EXPECT(1)

This statement reads the OI. The result is compared to 1.

AUKCWRCC

4.28 The description of the AUKCWRCC statement includes:

Function:

The AUKCWRCC statement writes (insertion masks) a 16-bit auxiliary unit register in the active or standby central control. The DG1AUPOS indexed by the base address (in register F) contains the bit position corresponding to the auxiliary unit under test. The register may be set or reset. All the other bits in the word remain unchanged.

Format:

AUKCWRCC WORD($1),DATA($2)

Characteristics of Parameters:

WORD — Specifies the register to be written. The $1 is the symbolic name of any auxiliary unit request register in the active or standby central control.

DATA — Specifies the data to be written. The $2 is 1 for set, 0 for reset.

Example:

AUKCWRCC WORD(IN1RIG),DATA(0)

This statement resets the auxiliary unit request inhibit group (IN1RIG) flip-flop corresponding to DG1AUPDS in both central controls. All other bits are unchanged.

AUMREAD

4.29 The description of the AUMREAD statement includes:

Function:

The AUMREAD statement generates an ML instruction. The ML instruction is described in Section 254-280-020 (Assembly Language—Description, 1A Processor). The read (ML) instruction may be directed to the auxiliary unit under test or to another auxiliary unit designed by KCODE. It also sets and resets 1PP_FUAC.

Format:

```
          ITEM($1)  NOSTORE
AUMREAD ITEMS($2),EXPECT($4),OPTIONS($6),KCODE($7)
          WORD($3) TO($5)
```

Characteristics of Parameters:

ITEM — Specifies the location to be read. The $1 is the item name.

ITEMS — Specifies the location to be read. The $2 is a list of items all in the same word.

WORD — Specifies the location to be read. The $3 is the address.

NOSTORE — Specifies that the results are to be ignored.

EXPECT — Specifies the expected results. The $4 is any arithmetic expression which expresses the expected results.

TO — Specifies the call store location containing the expected result. The $5 is the call store address which will be indexed by the base address (in register F).

OPTIONS — Specifies the option for the ML instruction. The ML instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor. The $6 is a list of options. If not specified, the options are C and R. Table A shows the options for the MS or ML instruction.

KCODE — Specifies the KCODE of the auxiliary unit to be read. The $7 is the KCODE. If not specified, the auxiliary unit under test is read and its KCODE is in DG1AUKCODE.

Example:

AUMREAD ITEM(FS1SAMLFWK),EXPECT(M(FS1SAMLFWK)),OPTIONS(R),KCODE(28)

This statement reads with the R option the FS1SAMLFWK item in the auxiliary unit whose KCODE is 28 and compares the result to the mask of the FS1SAMLFWK item.

AUMWRITE

4.30 The description of the AUMWRITE statement includes:

Function:

The AUMWRITE statement generates an MS instruction. The MS instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor. The write (MS) instruction may be directed to the auxiliary unit under test or to another auxiliary unit designated by KCODE. It also sets and clears 1PP-FUA.

Format:

AUMWRITE WORD($1),FROM($3),DATA($2)
OPTIONS($4),KCODE($5)

Characteristics of Parameters:

WORD — Specifies the location to be written. The $1 is the address.

DATA — Specifies the data to be written. The $2 is any arithmetic expression which expresses the data.

FROM — Specifies the call store location containing the data to be written. The $3 is the address of the call store location which is indexed by the base address (in register F).

OPTIONS — Specifies the options for the MS instruction. The MS instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor. The $4 is a list of options. If not specified, the options are C and W. Table A shows the options for the MS or ML instruction.

KCODE — Specifies the KCODE of the auxiliary unit to be written. The $5 is the KCODE. If not specified, the auxiliary unit under test is written and its KCODE is in DG1AUKCODE.

Example:

AUMWRITE WORD(FS1SAM),FROM(DG1SCR1)

This statement writes the data contained in call store location DG1SCR1, indexed by the F register, into the auxiliary unit location FS1SAM in the auxiliary unit under test.

AUPULSE

4.31 The description of the AUPULSE statement includes:

Function:

The AUPULSE statement control pulses an auxiliary unit location initializing various circuits and sending a 24-bit reply to central control. The results may be passed to the DCON program or ignored.

Format:

```
AUPULSE NOSTORE

        ITEM($1),NOSTORE
AUPULSE ITEMS($2),EXPECT($4)
        WORD($3)
```

Characteristics of Parameters:

NOSTORE — Specifies that the results of the read are to be ignored.

ITEM — Specifies the mask to be used with the reply. The $1 is the item name.

ITEMS — Specifies the mask to be used with the reply. The $2 is a list of items all in the same word.

WORD — Specifies the mask to be used with the reply. The $3 is the address.

EXPECT — Specifies the expected result. The $4 is any arithmetic expression which expresses the expected result.

Example:

```
AUPULSE WORD(FS1SAM),EXPECT(1AUDAT6)
```

This statement control pulses the auxiliary unit under test and forms a 24-bit mask for the restarting reply and compares the result to the value of 1AU1DAT6.

AURPLY

4.32 The description of the AURPLY statement includes:

Function:

The AURPLY statement initializes the auxiliary unit buffer register (AU1SWR) with the desired data pattern and clears all error sources.

Format:

```
AURPLY PATTERN($1)
```

Characteristics of Parameters:

PATTERN — Specifies the data pattern. The $1 is any 24-bit value to be written into the AU1SWR in the auxiliary unit.

Example:

```
AURPLY PATTERN(O(0))
```

This statement writes octal 0 into AU1SWR in the auxiliary unit under diagnosis and clears all error sources.

AUSTADD

4.33 The description of the AUSTADD statement includes:

Function:

The AUSTADD statement is used to specify the value of the 22 bits of the auxiliary unit store address bus and to specify the value of the read bit of the store address group (bit 22) as either 1 or 0. This initializes the store address register in the auxiliary unit and causes the auxiliary unit to transmit the desired store address to the central control.

Format:

AUSTADD PATTERN($1),READBIT($2)

Characteristics of Parameters:

PATTERN — Specifies the data pattern. The $1 is any 22-bit value used to specify the value of the 22 store address bits of the auxiliary unit bus. If 24 bits are specified, bits 22 and 23 are zeroed.

READBIT — Specifies the value of the read bit of the store address group (bit 22) as either 1 or 0; the write bit of the store address group (bit 23) is the opposite value of the read bit. The $2 is 0 or 1.

Example:

AUSTADD PATTERN (O(17777775)),READBIT(0)

This statement initializes the store address register in the auxiliary unit and causes the auxiliary unit to transmit octal 17777775 in bits 21 through 0; 0 in bit 22 and 1 in bit 23 to the central control.

AU_XOVER

4.34 The description of the AU_XOVER statement includes:

Function:

The AU_XOVER statement specifies which auxiliary unit is to be used to test for cross-over of enable and store access permitted (SAP) leads between central control and the auxiliary units.

Format:

AU_XOVER $1,$2,$\overline{\text{SAP}}$

Characteristics of Parameters:

$1 — Specifies auxiliary unit type by mnemonic symbol such as 1XLUFS for file store or 1XLUDUS for DUS.

$2 — Specifies auxiliary unit member number by decimal number.

SAP — If specified, a SAP cross-over test is made. If not specified, an enable test is made.

Example:

AU_XOVER 1XLUFS,1,SAP

This statement specifies that FS1 is to be used in the SAP cross-over test.

BUSACT

4.35 The description of the BUSACT statement includes:

Function:

The BUSACT statement tries to configure the specified call store or program store bus active. If the specified bus is the active bus, no action is taken. If the specified bus is out of service, a message is printed on the input/output terminal.

Format:

BUSACT ABUS($1),COMM($2)

Characteristics of Parameters:

ABUS — Specifies the bus desired active. The $1 is 0 or 1.

COMM — Specifies the community. The $2 is program store or call store.

Example:

BUSACT ABUS(0),COMM(PS)

This statement tries to configure the program store bus 0 active. If program store bus 0 is active, no action is taken. If program store bus 0 is out of service, a message is printed on the input/output terminal.

CCAAS_ST

4.36 The description of the CCAAS_ST statement includes:

Function:

The CCAAS_ST statement initializes all standby central control address source registers. The standby central control auxiliary unit send address register (AAS) contains the specified call store or program store address. All other standby central control address source registers contain the complement of the specified address. This statement also sets the auxiliary unit sequencer step flag, puts the auxiliary unit sequencer into state 1, and writes the value of the address parity bit (PKA) which is calculated by the statement, and sets the AASR and/or the AASW bit in the auxiliary unit verify receive group (VRG) register, and the MTCPS or MTCCS bit in the central control error summary register (CES) if M is specified in the OPTIONS parameter. Certain standby central control registers are initialized so that the standby central control is ready to address the program store or central control from the AAS.

Format:

CCAAS_ST ADD($1),OPTIONS($2),INHCLK($3)

Characteristics of Parameters:

ADD — Specifies the call store or program store address to be written. The $1 is any arithmetic expression which expresses the complete call store or program store address.

OPTIONS — Specifies the mode, parity, and store timing during the read (ML) instruction. The ML instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor. The $2 is a list of options. Table A is a list of options for the MS and ML instructions.

INHCLK — If specified, specifies a clock phase to be inhibited during the test. The $3 is 45S65, 7T9, 8T10, or 12T0.

Example:

CCAAS_ST ADD(1DGCS | 1DGK3 | 1DGA0),OPTIONS(M,R),INHCLK(45S65)

This statement initializes the standby central control to write into the address resulting from the ORing of the values of 1DGCS, 1DGK3, and 1DGA0, using the maintenance read with the 45S65 clock inhibited.

CCARR_ST

4.37 The description of the CCARR_ST statement includes:

Function:

The CCARR_ST statement initializes the following standby central control registers:

(a) Memory address decoder flip-flops (MSFs) with IDLEPS or IDLECS set

(b) Instruction fetch register (IFR) to indicate a full stack

(c) Auxiliary unit and central control block counters (ABK) with BLKAUOV and BLKAU set

(d) Auxiliary unit miscellaneous group B (AMB) with auxiliary unit sequencer in second state and AUQS set

(e) Auxiliary unit store address register (AAS) contains program store or call store address

(f) Auxiliary unit verify received group (VRG) with write (W) and address parity bit (PKA) set

(g) Auxiliary unit reply register (ARR) with desired data

(h) Data buffer register (BR) with complement of desired data

(i) Auxiliary unit enable verify expect group (EVG) with correct data parities

(j) Clock error group register (CLE) to inhibit the specified clock phase.

Format:

CCARR_ST ADD($1),REG_INIT($2),INHCLK($3)

Characteristics of Parameters:

ADD — Specifies the call store or program store address. The $1 is any arithmetic expression which expresses the complete address.

REG_INIT — Specifies the data. The $2 is any arithmetic expression which expresses the data.

INHCLK — If specified, specifies a clock phase to be inhibited during the test. The $3 is 45S65, 7T9, 8T10, or 12T0.

Example:

CCARR_ST ADD(1DGCS | 1DGK0 | 1DGA0)),REG_INIT(1DGD77)

This statement initializes the standby central control registers. The AAS will contain the address resulting from ORing the values of 1DGCS, 1DGK0, and 1DGA0. The ARR will contain the value of 1DGD77. The BR will contain the complement of the value of 1DGD77.

CCATOTST

4.38 The description of the CCATOTST statement includes:

Function:

The CCATOTST statement starts the central control analog timer, waits a specified amount of time, then tests to verify that the analog timer timeout did or did not occur.

Format:

CCATOTST DELAY=$1,TIMEOUT=$2

Characteristics of Parameters:

DELAY — Specifies the delay time before the test. The $1 is the number of milliseconds to delay.

TIMEOUT — Specifies the expected result of the test. The $2 is YES or NO.

Example:

CCATOTST DELAY = 125, TIMEOUT = YES

This statement starts the analog timer, waits 125 ms, then tests that an analog timer timeout did occur.

CCAUBRQ

4.39 The description of the CCAUBRQ statement includes:

Function:

The CCAUBRQ statement starts an auxiliary unit sending bus requests to the central control. This statement is used by the central control diagnostic auxiliary unit bus testing phases. The auxiliary unit to be used is specified through the auxiliary unit status words stored in the diagnostic call store scratch area. The following call store words contain the following information:

DG1GCPADDR — Unit generate control pulse (GCP) address

DG1AUKCODE — Unit KCODE

DG1AUUTMN — Unit type and member number.

Format:

CCAUBRQ

Characteristics of Parameters:

This statement has no parameters.

Example:

CCAUBRQ

This statement starts the auxiliary unit, specified in the auxiliary unit status words in the diagnostic call store scratch area, sending bus requests to the central control.

CCAUINIT

4.40 The description of the CCAUINIT statement includes:

Function:

The CCAUINIT statement initializes the auxiliary unit bus system for a test segment of the central control diagnostic auxiliary unit bus tests.

Format:

CCAUINIT $\overline{\text{BUSINH}}$

Characteristics of Parameters:

BUSINH — If specified, the contents of scratch word DG1DTSCR4 is written into the standby central control buffer pulse source (BPS) register, thus establishing the bus inhibit condition desired during the test segment.

Example:

CCAUINIT

This statement initializes the auxiliary unit bus system for a test segment of central central diagnostic auxiliary unit bus tests.

CCAURSTR

4.41 The description of the CCAURSTR statement includes:

Function:

The CCAURSTR statement restores the auxiliary unit system after a test segment of the central control diagnostic auxiliary unit bus tests.

Format:

CCAURSTR

Characteristics of Parameters:

This statement has no parameters.

Example:

CCAURSTR

This statement restores the auxiliary unit system after a test segment of the central control diagnostic auxiliary unit bus tests.

CCAUSTAT

4.42 The description of the CCAUSTAT statement:

Function:

The CCAUSTAT statement gets an auxiliary unit status for the central control diagnostic auxiliary unit bus testing phases. Status is derived by an auxiliary unit fault recovery (AUFR) subroutine. Given a unit type number and a member number, the following call store words will be updated to reflect the status of the corresponding units:

DG1KCODE — Contains the KCODE of the unit.
DG1AUSTAT — Contains the AUFR status of the unit.
DG1AUAXA — Contains the auxiliary block table address of the unit.
DG1GCPADDR — Contains the GCP address of the unit.
DG1AUPOS — Contains the mask of the unit's central control register position.
DG1AUUTMN — Contains unit type and member numbers.
DG1AUCTRL — Contains odd KCODE flag.

Format:

CCAUSTAT UTYN(SCR)

CCAUSTAT UTYN($1),MEMN($2)

Characteristics of Parameters:

UTYN — Specifies the unit type number. If SCR is specified, the call store scratch word DG1AUUTMN contains the unit type and member number. The $1 is the unit type, usually specified with symbols such as 1XLUFS for file store or 1XLUDUS for DUS.

MEMN — Specifies the member number. The $2 is 0, 1, or 3.

Example:

CCAUSTAT UTYN(1XLUDUS),MEMN(1)

This statement gets the status of the DUS 1 for the central control diagnostic auxiliary unit bus testing phases.

CCBITEST

4.43 The description of the CCBITEST statement includes:

Function:

The CCBITEST statement causes the active central control to do a series of ten write/read operations into the standby central control register specified. The data for each write operation is one of a group of five general-purpose bit patterns and their complements which are masked with the mask of the read/write items of the register being tested. Following each of the ten writes, a read operation passes the contents of the standby central control register, the mask of its read/write items, the expected results, and the data originally written to the DCON program for the processing of raw results.

The ten general-purpose bit patterns are as follows:

SYMBOL	PATTERN	PATTERN(—)
1DG_BPAT1	25252525	52525252
1DG_BPAT2	31463146	46314631
1DG_BPAT3	36074170	41703607
1DG_BPAT3	40077600	37700177
1DG_BPAT5	77700000	00077777

Format:

CCBITEST $1

Characteristics of Parameters:

$1 — Specifies a register in the standby central control without the prefix ST1. The 1DG_RW_REG is defined in Datapool.

Example:

CCBITEST BC0

This statement causes the active central control to write/read ten bit patterns into the bit control register 0 (BC0) in the standby central control.

CCBR_ST

4.44 The description of the CCBR_ST statement includes:

Function:

The CCBR_ST statement initializes the standby central control to send data to the desired program store or call store. The address is loaded in the standby central control X register (XR); the instruction is loaded in the buffer order word register left-half (BOL) register, and the data is loaded in the G register (GR) and data buffer register (BR). Also, this statement writes the complement of the specified data into the auxiliary unit reply register (ARR).

Format:

CCBR_ST ADD($1),REG_INIT($2),OPTIONS($3),INHCLK($4)

Characteristics of Parameters:

ADD — Specifies the call store or program store address. The $1 is any arithmetic expression which expresses the address.

REG_INIT — Specifies the data to be written. The $2 is any arithmetic expression which expresses the data.

OPTIONS — Specifies the mode, parity, and store timing during the write (MS) instruction. The MS instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor. The $3 is a list of options. Table A is a list of options for the MS and ML instructions.

INHCLK — If specified, specifies a clock phase to be inhibited during the test. The $4 is 45S65, 7T9, 8T10 or 12T0.

Example:

CCBR_ST ADD(1DGCS | 1DGK0 | 1DGA0),REG_INIT(1DGD77),OPTIONS(W,IP1,IP2), INHCLK(8T10)

This statement loads the address resulting from ORing the values of 1DGCS, 1DGK0, and 1DGA0 into the standby call store XR; the send instruction is loaded into the standby central control BOL, the value of 1DGD77 is loaded in the standby central control GR and BR; and the complement of the value of 1DGD77 is loaded into the standby central control ARR. The write is performed with the W, IP1, and IP2 options. Clock phase 8T10 is inhibited.

CCCLR

4.45 The description of the CCCLR statement includes:

Function:

The CCCLR statement clears most registers of the standby central control or any of the four areas (INAD, UBMB, BB, AU) specified.

Format:

CCCLR AREA($1)

Characteristics of Parameters:

AREA — Specifies the area in the standby central control to be cleared. The $1 is either ALL or any combination of the following: INAD, UBMB, BB, AU.

INAD — The registers in the standby central control instruction address area are cleared in the following order:

(1) Order word register (OW)
(2) Buffer order word register right-half (BOR)
(3) Half word register (HWR)
(4) Auxiliary buffer word register left-half (ABL)
(5) Save data register (SDA)
(6) Index adder augend register (IA)
(7) Index adder addend register (ID)
(8) Add-one register (AOR)
(9) Auxiliary current register (ACR)
(10) Save current address register (SCA_B)
(11) Save program address register (SPA_B).

UBMB — The registers in the standby central control masked and unmasked bus area are cleared in the following order:

(1) L register (LR)
(2) F register (FR)
(3) G register (GR)
(4) K register (KR)
(5) X register (XR)
(6) Y register (YR)
(7) Z register (ZR)
(8) J register (JR)
(9) Peripheral unit enable register (ER)
(10) P register's most significant bits (PRM_B)
(11) P register's least significant bits (PRL)
(12) Central pulse distributor (CDP) reply register (RR)
(13) Stack register (SR)
(14) Argument register (AG)
(15) Sign and homogeneity flip-flops (CF)
(16) Size displacement register (SDR).

BB — The registers in the standby central control buffer bus area are cleared in the following order:

(1) Central control error summary register (CES)
(2) Interrupt level activity flip-flop (ILA)
(3) Interrupt level request register (ILR)
(4) Interrupt inhibit register (INH)
(5) Interrupt request register (INJ)
(6) Interrupt sequence register (INR)
(7) Interrupt source register (INS)
(8) Internal timer register (ITR)
(9) Millisecond clock (MCL)
(10) Processor configuration sanity timers (SAT)
(11) Peripheral unit error summary (PES)
(12) Processor configuration control register (PCR)
(13) Clock error group (CLE) register
(14) Central control status flip-flop (CSC)
(15) Peripheral unit sequencer control (PSC)
(16) Instruction fetch register (IFR)
(17) Instruction execution register (IER)
(18) Memory address decoder flip-flop (MDF)
(19) CDP diagnostic and echo register (DE)
(20) Stack counter (SC_B)
(21) Peripheral matcher 0 (PM0)
(22) Peripheral matcher 1 (PM1)
(23) Lower protected area bound register (LPA)
(24) Upper protected area bound register (UPA)
(25) Activity flip-flop group (ACT)
(26) Delay and limited run register (DLR)
(27) Stop-start register (SSR)
(28) Data buffer register (BR)
(29) Buffer shadow register (BRS)
(30) Inhibit buffer register (IBR).

AU — The registers in the standby central control auxiliary unit area are cleared in the following order:

(1) Auxiliary unit and central control block counters (ABK)
(2) Auxiliary unit store access verify expect group (SVG)
(3) Auxiliary unit write slow register (AWS_B)
(4) Auxiliary unit enable buffer register (EBG)
(5) Auxiliary unit request inhibit group (RIG)
(6) Auxiliary unit reply register (ARR_B)
(7) Auxiliary unit write fast register (AWF_B)
(8) Auxiliary unit enable verify expect group (EVG)
(9) Auxiliary unit miscellaneous group A (AMA)
(10) Auxiliary unit miscellaneous group B (AMB)
(11) Auxiliary unit miscellaneous group C (AMC)
(12) Bit control register 0 (BC0)
(13) External match register 0 (ME0)
(14) Bit control register 1 (BC1)
(15) Internal match register 1 (M11)
(16) External match register 1 (ME1)
(17) Match mode control register (MMR)
(18) Match summary register (MSR)

(19) Matcher 0 control register (M0R)
(20) Matcher 1 control register (M10)
(21) Internal match register 0 (MI0)
(22) Combined mask 0 (CM0)
(23) Combined mask 1 (CM1)
(24) Auxiliary unit store address register (AAS_B)
(25) Auxiliary unit request group (RQG)
(26) Auxiliary unit verify receive group (VRG).

ALL — All standby central control registers are cleared from the areas (INAD, UBMB, BB, AU).

Example:

CCCLR AREA(INAD,UBMB)

This statement clears the registers in the standby central control in the INAD and UBMB areas.

CCDAR_ST

4.46 The description of the CCDAR_ST statement includes:

Function:

The CCDAR_ST statement initializes all the standby central control address source registers. The standby central control data address register (DAR) contains the specified program store or call store address. All other standby central control address source registers contain the complement of the specified address. This statement also initializes all necessary registers in the standby central control to be ready to address the call store or program store from the DAR. The statement also initializes any one of the central control-to-store clock syncs in the standby central control and/or certain central control clock phases.

Format:

CCDAR_ST ADD($1),OPTIONS($2),INHSYNC($3),INHCLK($4)

Characteristics of Parameters:

ADD — Specifies the program store or call store address to be written. The $1 is any arithmetic expression which expresses the complete program store or call store address.

OPTIONS — Specifies the mode, parity, and store timing during the MS or ML instructions. The ML and MS instructions are described in Section 254-280-020, Assembly Language—Description, 1A Processor. The $2 is a list of options. Table A is a list of options for the MS and ML instructions.

INHSYNC — If specified, specifies one of the central control-to-store clock syncs in the standby central control.The $3 is any one of three central control-to-store clock syncs 1T3, 3T5, or 5T7.

INHCLK — If specified, specifies a clock phase to be inhibited during the test. The $4 is 45S65, 7T9, 8T10, or 12T0.

Example:

CCDAR_ST ADD(1DGCS | 1DGK3 | 1DGA0),OPTIONS(IST),INHSYNC(1T3)

This statement initializes the standby central control address source registers with the address resulting from the ORing of the values of 1DGCS, 1DGK3, and 1DGA0. The IST option is used on the write instruction. The 1T3 central control-to-store clock sync is inhibited.

CCGATE

4.47 The description of the CCGATE statement includes:

Function:

The CCGATE statement tests the standby central control. It writes the instruction fetch register (IFR), buffer order word right and left-half (BOR and BOL) registers, program address register (PAR), memory address decoder flip-flop (MDF), and any other registers specified in the statement. Then it runs the standby central control the number of cycles specified (one if not specified), then reads whatever registers are specified and compares it to the expected results. The statement also specifies whether the test is active or inactive. Finally, it writes the standby central control pests and resets ST1FREZ in the buffer pulse source (BPS) register.

Format:

CCGATE $1

Characteristics of Parameters:

$1 — Specifies any combination of registers, data, cycle time, and active or inactive tests.

Example:

```
CCGATE(CYC1__SET=(GOCOND1)),A,WIFR(1DG__IFQ__00 M(ST1BOLV,ST1BORV)),
ME          RIFR(1DG__IFQ__01  M(ST1CYC1))
```

This statement performs an active test. First it writes the value of 1DG_IFQ_00 ORed with the mask of the ST1BOLV and ST1BORV items into the IFR in the standby central control. It also writes data into the BOL, BOR, PAR, and MDF. Then it runs the standby central control 1 cycle. It then reads the IFR and compares the result with the value of 1DG_IFQ_01 ORed with the mask of ST1CYC1. It writes the standby central control pests and resets ST1FREZ.

CCGCPTST

4.48 The description of the CCGCPTST statement includes:

Function:

The CCGCPTST statement is a special purpose statement used by the pulse source tests in the central control diagnostic program. The CCGCPTST statement executes six tests of pulse source control of the specified flip-flop. The statement results in three main strips of code: control code, relocatable code, and the store subroutine. The tests are actually done in the relocatable code strip which can be optionally executed from call store if specified in the statement. All GCPs to be generated by the standby central control are done by the GCP_EXEC subroutine which is part of the relocatable code strip. The relocatable code returns to the control code after every two tests with the L register (LR) and L register shadow (LRS) containing the raw results of the two reads. All test results are stored by the GCP_STOR subroutine. The actual order in which the tests are executed is test 2, test 1, test 4, test 3, test 6, and test 5.

Format:

CCGCPTST FF($1),$2,POINTS($3,$4),$\overline{\text{CSPGM}}$

Characteristics of Parameters:

FF — Specifies the flip-flop. The $1 is the hardware name of the flip-flop which is controlled by GCP pulses from either central control.

$2 — Symbol specifying the 1-out-of-10 field information (board) for the GCP address, ie, BD15.

POINTS — Specifies the pulse point. The $3 is a symbol specifying the information for the two 1-out-of-6 fields (point) to set the specified flip-flop. The $4 is a symbol specifying the two 1-out-of-6 fields (point) to clear the specified flip-flop.

CSPGM — If specified, the relocatable code strip is copied to and executed from call store. This is specified when the flip-flop being tested could interfere with program store communications in the active central control.

Example:

CCGCPTST FF(TCC),BD21,POINTS(33,32)

This statement tests the trouble in central control (TCC) flip-flop by pulsing it with GCP pulses from the board address BD21, points (33, 32).

CCINT_ST

4.49 The description of the CCINT_ST statement includes:

Function:

The CCINT_ST statement initializes the standby central control to send data to the desired program store or call store during a D- or E-level interrupt. The interrupt sequencer is placed in state 4, and the specified data is placed in the current address register (CAR) and the control flip-flops. The buffer order word registers left and right halves (BOL and BOR) are loaded with an instruction to set the ILRD or ILRE in the interrupt level request (ILR) register.

Format:

CCINT_ST ADD($1),REG_INIT($2)

Characteristics of Parameters:

ADD — Specifies the call store or program store address. The $1 is any arithmetic expression which expresses the complete call store or program store address.

REG_INIT — Specifies the data to be written. The $2 is any arithmetic expression which expresses the data.

Example:

CCINT_ST ADD(1DGCS) | 1DGK0 | 0(4011)),REG_INIT(1DG25)

This statement initializes the standby central control to write the value of 1DG25 into the address resulting from ORing the values of 1DGCS, 1DGK0, and octal 4011 during an E-level interrupt.

CCISOL

4.50 The description of the CCISOL statement includes:

Function:

The CCISOL statement performs central control isolation tests concerning program store or call store buses. This test makes sure there is no crosstalk between bus 0 and bus 1 of program store or call store buses. This statement dynamically creates a small program and executes it from the opposite community specified by bit 21 of ADD data; ie, call store if bit 21 is set. After executing this program, test results are processed according to the TEST parameter. If the test is EN, the expected value is RDATA; if the test is INH, the expected value is the complement of RDATA. The small program created by this statement configures program store bus selection flip-flops (PBO and PBT) or call store bus selection flip-flops (CBO and CBT), changes the original K-code of a test store to a desired K-code specified in ADD data, and sets the maintenance flip-flop in a test store whose community is specified by bit 21 of ADD data. The store location specified by the ADD data is intialized with data specified by RDATA. This program also runs or stops the standby central control according to the STBCC parameter. It executes an MS/ML instruction with options specified. The last thing the program does is to restore system configuration and original store data. The CCISOL must be immediately preceded by a central control-to-store initialization statement.

Format:

CCISOL TEST($1),STORE(ADD$2),$\overline{\text{RDATA}(\$3)}$,ACTCC(OPTIONS($4),$\overline{\text{WDATA}(\$5)}$),STBCC(6),$\overline{\text{OPTIONS}(\$7)}$

Characteristics of Parameters:

TEST — Specifies the manner in which the test results are to be processed. The $1 is INH or EN. If $1 is EN, the expected value is RDATA or WDATA. If $1 is INH, the expected value is the complement of RDATA or WDATA.

STORE — Specifies the address of the store and the expected results.

ADD — Specifies the address of the store. The $2 is any arithmetic expression which expresses the complete store address.

RDATA — Specifies the expected results for an address isolation test. The $3 is any arithmetic expression which expresses the expected result.

ACTCC — Specifies the options and data for the MS/ML instruction executed by the active central control. The MS and ML instructions are described in Section 254-280-020, Assembly Language—Description, 1A Processor.

OPTIONS — Specifies the options for the MS/ML instruction. The $4 is any combination of options for the MS/ML instruction. Table A is a list of options for the MS and ML instructions.

WDATA — Specifies the data for the MS instruction for a data isolation test. The $5 is any arithmetic expression which expresses the data.

STBCC — Specifies whether the standby central control is to run and the options for the MS/ML instruction. The MS and ML instructions are described in Section 254-280-020, Assembly Language—Description, 1A Processor.

$6 — Specifies whether the standby central control is to run. The $6 is RUN or STOPPED.

OPTIONS — Specifies the options for the MS/ML instruction. The $7 is any combination of options for the MS/ML instruction. Table A is a list of options for the MS and ML instructions.

Example:

```
CCISOL_TEST(EN),STORE(ADD(1DGCS | 1DGK3 | 1DGA0),RDATA(1DGD25)),ACTCC
ME        (OPTIONS(R,M)),STBCC(RUN,OPTIONS(C,W,R))
```

This statement performs a central control address bus isolation test to make sure there is no crosstalk between bus 0 and bus 1 of the call store buses. A small program is created and executed in the program store community. The store address is the result of ORing the values of 1DGCS, 1DGK3, and 1DGA0. After the program has been executed, the result is compared to the value of 1DGD25. The small program sets maintenance flip-flops in a call store test store. The store location at the address resulting from ORing the values of 1DGCS, 1DGK3, and 1DGA0 is initialized with the value of 1DGD25. This program also runs the standby central control. The active central control executes an MS/ML instruction with R and M options, the standby central control executes an MS/ML instruction with C, W, and R options.

CCMCP3P

4.51 The description of the CCMCP3P statement includes:

Function:

The CCMCP3P statement tests the standby central control millisecond clock (MCL) circuitry. The statement starts the delay and limited run (DLR) sequencer in the standby central control. The operational clock error detector in the standby central control is then initialized and the operational clock error detector indicator, analog timeout indicator and analog timer toggle inhibit flip-flop are cleared. An 8-cycle (5.6 ms) delay is taken. During the delay, circuit functions of the standby central control MCL may or may not occur, depending on the test being performed. No tests are generated by this statement.

Format:

CCMCP3P

Characteristics of Parameters:

This statement has no parameters.

Example:

CCMCP3P

This statement starts the DLR sequencer in the standby central control, then initializes the operational clock error detector and clears the operational clock error detector indicator, the analog timeout indicator, and the analog timer toggle inhibit flip-flop. Then an 8-cycle delay is taken.

CCMUTIME

4.52 The description of the CCMUTIME statement includes:

Function:

The CCMUTIME statement searches for a nonbase test store in the community specified in the COMM parameter. In the case of REC test and call store community or RECB test and call store community, the statement searches for a test call store on the branch of call store reply bus specified by the TEST parameter (ie, REC indicates call store reply bus branch A and RECB indicates call store reply bus branch B). The method of search is in two fields. First, a search by K-code for duplicated stores is performed until maximum K-code is reached. If successful, the search terminates. Otherwise, a search by member number is performed until maximum allowable member number for the office is reached. Any member number which is out of service or contains base K-code is rejected. After a test store is found, this statement calculates delay constants depending on the type of test and community. If no store is available, the delay constant is zero.

Format:

CCMUTIME TEST($1),COMM($2)

Characteristics of Parameters:

TEST — Specifies the type of test. $1 is one of the following:

REC: Receive test for program store and receive test on call store reply bus branch A
RECB: Receive test on call store reply bus branch B
TRAN: Transmission test
ISOL: Isolation test.

COMM 1 — Specifies the store community. The $2 is program store or call store.

Example:

CCMUTIME TEST(RECB),COMM(CS)

This statement searches for a test call store on branch B of the call store reply bus.

CCPAR_ST

4.53 The description of the CCPAR_ST statement includes:

Function:

The CCPAR_ST statement initializes all the standby central control address source registers. The standby central control program address register (PAR) contains the specified call store or program store address. All other standby central control address source registers contain the complement of the specified program store or call store address. This statement also initializes all necessary registers in the standby central control to be ready to address call store or program store from the PAR.

Format:

CCPAR_ST ADD($1),INHCLK($2)

Characteristics of Parameters:

ADD — Specifies the call store or program store address to be written. The $1 is any arithmetic expression which expresses the complete call store or program store address.

INHCLK — If specified, specifies a clock phase to be inhibited during the test. The $2 is 45S65,7T9, 8T10, or 12T0.

Example:

CCPAR_ST ADD(1DGCS) | 1DGK34 | 1DGA77)

This statement initializes the standby central control to write at the address resulting from the ORing of the values of 1DGCS, 1DGK34, and 1DGA77.

CCPCCNFG

4.54 The description of the CCPCCNFG statement includes:

Function:

The CCPCCNFG statement can perform three functions:

(1) The CCPCCNFG statement verifies that the processor configuration sequencer can be safely fired from a specified processor configuration state. If base K-code is not duplicated, an entire set of tests is aborted. A check is made that the program store which will be selected as base by the processor configuration sequencer for the specified processor configuration state contains the base K-code. If not, the next test is skipped.

(2) The CCPCCNFG statement verifies that the standby central control is either all tests pass (ATP) or conditional all tests pass (CATP) and that the demand processor configuration tests have been entered via the proper input message. If the input message was incorrect, a message is printed on the input/output terminal.

(3) The CCPCCNFG statement will either set or reset the TCC flip-flop in the active central control or restore it to the state it exhibited during the last PHASEINIT statement.

Format:

CCPCCNFG PCSTATE ($1), SKIPLAB($2), $\overline{\text{ABORTLAB}(\$3)}$

$\overline{\text{DGNRUN}}$ ($4), $\overline{\text{ABORTLAB}(\$3)}$, $\overline{\text{TCC}}$($5)

Characteristics of Parameters:

PCSTATE — Specifies the processor configuration counter state in which the next trigger is to occur. The $1 is any symbolic processor configuration counter state.

SKIPLAB — Specifies the location to which a DTJUMP is made if the selected program store does not contain base K-code. The $2 is a label defined on a DTDEST statement immediately after the next test.

ABORTLAB (used with PCSTATE) — If specified, specifies a location to which a DTJUMP is made to abort the entire sequence of tests which will fire the processor configuration sequencer in a configuration state. The $3 is a label defined in a DTDEST statement.

DGNRUN — Specifies that it is to be verified, that the standby central control is ATP or CATP, and that the proper input message was entered for running the demand processor configuration tests. The $4 is ATP or CATP.

ABORTLAB (used with DGNRUN) — If specified, specifies a location to which a DTJUMP is made to terminate the diagnostic. The $3 is a label defined in a DTDEST statement.

TCC — If specified, specifies what action is to be performed on the TCC flip-flop in the active central control. The $5 is one of the following:

SET — TCC flip-flop is set
RESET — TCC flip-flop is reset
RESTORE — TCC flip-flop is restored to the state it exhibited during the last PHASEINIT statement.

Examples:

(a) CCPCCNFG PCSTATE(02), SKIPLAB(CCDG91_SKIP1),ABORTLAB(CCDG91_SKIP3)

This statement checks to see that K-code 20 is duplicated in the program store community. If not, a DTJUMP to label CCDG91_SKIP3 will be made. If so, PS0 is checked to see if it contains K-code 20. If not, a DTJUMP to label CCDG91_SKIP3 will be made.

(b) CCPCCNFG DGNRUN(CATP),ABORTLAB(CCDG91A),TCC(SET)

This statement checks to see that all previous phases have run CATP (or ATP) and that the proper input message for running the demand processor configuration tests are entered. If not, a DTJUMP to label CCDG91A will be made and a message will be made and a message will be printed on the input-output terminal. The TCC flip-flop in the active central control is set.

CCPCINIT

4.55 The description of the CCPCINIT statement includes:

Function:

The CCPCINIT statement initializes the processor configuration circuit in the standby central control. The following registers are cleared: PC register (PCR), sanity timers (SATs), activity flip-flop group (ACT), interrupt source register (INS), and interrupt level activity flip-flops (ILA). Also, the cable driven inhibits may be set and/or the analog timer flip-flop toggled.

Format:

CCPCINIT [CDINH,ATMRT]

Characteristics of Parameters:

CDINH — If specified, sets the cable driven inhibits.

ATMRT — If specified, toggles the analog timer flip-flop.

Example:

CCPCINIT CDINH,ATMRT

This statement initializes the processor configuration circuit in the standby central control, clears the PCR, SAT, ACT, INS, and ILA registers, sets the cable driven inhibits, and toggles the analog timer flip-flop.

CCPCNOTR

4.56 The description of the CCPCNOTR statement includes:

Function:

The CCPCNOTR statement tests to see that no processor configuration source indicator flip-flops are set in the absence of any processor configuration sources.

Format:

CCPCNOTR

Characteristics of Parameters:

This statement has no parameters.

Example:

CCPCNOTR

This statement tests to see that no processor configuration source indicator flip-flops are set in the absence of any processor configuration sources.

CCPCTRIG

4.57 The description of the CCPCTRIG statement includes:

Function:

The CCPCTRIG statement triggers the processor configuration circuitry in either the active or standby central control and performs a cleanup operation. It is used when the diagnostic fires the analog clock in either central control to ensure system sanity is not lost. If the processor configuration state counter is in a configuration state, the statement which performs the processor configuration trigger is executed from call store. Also, in this case, a status table is built describing which central control is active and the state of the base program store and the two rovers both before and after the processor configuration trigger. If the processor configuration trigger is executed from program store, only the central control status is placed in the status table. After the before status is compiled and prior to the program store trigger, the base and two rover program stores may be reconfigured for test purposes. This statement generates 0, 1, or 2 tests.

Format:

CCPCTRIG UNIT($1),SOURCE($2,BLOCK),STATE($3),VERIFY($4,NOT),SBYCLOCK($5)

,ROSET($6),COMPKC($6)

Characteristics of Parameters:

UNIT — Specifies the unit in which the processor configuration circuit is to be activated. The $1 is ACTIVE or STANDBY.

SOURCE — Specifies the source of the processor configuration trigger. The $2 is DELIBERATE, MATCH, PST, PCST, or NOTRIGGER. DELIBERATE causes a deliberate processor configuration trigger. MATCH causes an activity match trigger. The PST causes a program sanity timer trigger. The PCST causes a processor configuration sanity timer trigger. NOTRIGGER causes the trigger action to be omitted. If specified, BLOCK causes the selected source to be blocked from triggering the analog clock. BLOCK is specified only if $1 is standby, $2 is DELIBERATE, PST, or PCST, and $3 is not a configuration state.

STATE — Specifies the processor configuration counter state in which the trigger is to occur. The $3 is any symbolic processor configuration counter state.

VERIFY — If specified, that the processor configuration counter was incremented in the unit specified is verified. If NOT is specified, that the processor configuration counter was not incremented is verified. The $4 is ACTIVE or STANDBY.

SBYCLOCK — If specified, that the operational clock in the standby central control is in the specified state after the processor configuration sequence is verified. The $5 is RUNNING or STOPPED. Also, may be used to start the operational clock in the standby central control prior to the processor configuration sequence (no test is performed in this case). The $5 is START.

ROSET — If specified, specifies the stores in which the RO flip-flop is to be set prior to the processor configuration trigger. Any stores not listed will have the flip-flop cleared. The $6 is PS0 RS0, RS1, or any combination. No preconfiguration is performed if ROSET is not specified.

COMPLC — If specified, specifies the stores in which the K-code is to be complemented prior to the processor configuration trigger. The $6 is PS0, RS0, RS1, or RS1, or any combination. If COMPKC is specified, ROSET must also be specified.

Examples:

(a) CCPCTRIG_UNIT(STANDBY),SOURCE(PCST,BLOCK),STATE(00),VERIFY(STANDBY,
ME NOT),SBYCLOCK(RUNNING)

This statement triggers the standby central control processor configuration circuitry in processor configuration counter state 00 from the processor configuration sanity timeout. The processor configuration sanity timeout is blocked from triggering the analog clock. That the processor configuration counter state is not incremented and that the operational clock is running in the standby central control is verified.

(b) CCPCTRIG UNIT (ACTIVE),SOURCE(DELIBRATE),STATE(15),ROSET(RS1),
COMPKC(PS0)

This statement deliberately triggers the active central control processor configuration circuitry in processor configuration counter state 15 after the K-code in PS0 is complemented. The RO flip-flop is set in rover store 1 and cleared in rover store 0 and PS0.

CCPCTST1

4.58 The description of the CCPCTST1 statement includes:

Function:

The CCPCTST1 statement tests the operational clock stop-start actions of the processor configuration sequencer. The operational clock should be stopped at PCP1 and restarted at PCP3 in nonswitch states. A deliberate processor configuration trigger is generated in the standby central control and two tests are performed to verify that the operational clock is first stopped and then started.

Format:

CCPCTST1

Characteristics of Parameters:

This statement has no parameters.

Example:

CCPCTST1

This statement tests the operational clock stop-start actions of the processor configuration sequencer. A deliberate processor configuration trigger is generated in the standby central control and two tests are performed to verify that the operational clock is first stopped and then started.

CCPHAPHB

4.59 The description of the CCPHAPHB statement includes:

Function:

The CCPHAPHB statement tests the active central control matcher hardware. Matchers 0 and 1 control registers (MC0 and MC1) are initialized by prior DL-1 statements. This statement sets certain active central control registers, starts the match sequencers, and then executes an instruction. The instruction, during its execution, causes specific data to be present in the stack counter (SC_B) and data address registers (DARs) during match phases A and B of matchers 0 and 1, respectively. This data is matched automatically by the matchers and then the match sequencers are stopped. No test results are generated; they are obtained by succeeding DL-1 statements.

Format:

CCPHAPHB

Characteristics of Parameters:

This statement has no parameters.

Example:

CCPHAPHB

This statement causes specific data to be present in the SC_B and DAR registers during match phases A and B of central control matchers 0 and 1, respectively.

CCPHBPHA

4.60 The description of the CCPHBPHA statement includes:

Function:

The CCPHBPHA statement tests the active central control matcher hardware. Matchers 0 and 1 control registers (MC0 and MC1) are initialized by prior DL-1 statements. This statement sets certain active central control registers, starts the match sequencers, and then executes an instruction. The instruction, during its execution, causes specific data to be present on the masked bus (MB) and unmasked bus (UB) during match phases B and A of matchers 0 and 1, respectively. This data is matched automatically by the matchers and then the match sequencers are stopped. No test results are generated; they are obtained by succeeding DL-1 statements.

Format:

CCPHBPHA

Characteristics of Parameters:

This statement has no parameters.

Example:

CCPHBPHA

This statement causes specific data to be present on the MB and UB during match phases B and A of central control matchers 0 and 1, respectively.

CCPHBPHB

4.61 The description of the CCPHBPHB statement includes:

Function:

The CCPHBPHB statement tests the active central control matcher hardware. Matchers 0 and 1 control registers (MC0 and MC1) are initialized by prior DL-1 statements. This statement sets certain active central control registers, starts the match sequencers, and then executes an instruction. The instruction, during its execution, causes specific data to be present on the MB and in the data address register (DAR) during match phases B and B of matchers 0 and 1, respectively. This data is matched automatically by the matchers and then the match sequencers are stopped. No test results are generated; they are obtained by succeeding DL-1 statements.

Format:

CCPHBPHB

Characteristics of Parameters:

This statement has no parameters.

Example:

CCPHBPHB

This statement causes specific data to be present on the MB and in the DAR during match phases B and B of central control matchers 0 and 1, respectively.

CCPHBPHC

4.62 The description of the CCPHBPHC statement includes:

Function:

The CCPHBPHC statement tests the active central control matcher hardware. Matchers 0 and 1 control registers (MC0 and MC1) are initialized by prior DL-1 statements. This statement sets certain active central control registers, starts the match sequencers, and then executes an instruction. The instruction, during its execution, causes specific data to be present on the MBs and UBs during match phases B and C of matchers 0 and 1, respectively. This data is matched automatically by the matchers and then the match sequencers are stopped. No test results are generated. They are obtained by succeeding DL-1 statements.

Format:

CCPHBPHC

Characteristics of Parameters:

This statement has no parameters.

Example:

CCPHBPHC

This statement causes specific data to be present on the MB and UB during match phases B and C of central control matchers 0 and 1, respectively.

CCPHCPHB

4.63 The description of the CCPHCPHB statements includes:

Function:

The CCPHCPHB statement tests the active central control matcher hardware. Matchers 0 and 1 control registers (MC0 and MC1) are initialized by prior DL-1 statements. This statement sets certain active central control registers, starts the match sequencers, and then executes an instruction. The instruction, during its execution, causes specific data to be present in the stack counters (SC_B) and data address (DAR) registers during match phases of C and B of matchers 0 and 1, respectively. This data is matched automatically by the matchers and then the match sequencers are stopped. No test results are generated; they are obtained by succeeding DL-1 statements.

Format:

CCPHCPHB

Characteristics of Parameters:

This statement has no parameters.

Example:

CCPHCPHB

This statement causes specific data to be present on the SC_B and DAR during match phases C and B of central control matchers 0 and 1, respectively.

CCPULSE

4.64 The description of the CCPULSE statement includes:

Function:

With the GCP instruction, the CCPULSE statement causes the active central control to pulse the point specified in the point parameter. The second and third parameters produce a pulse/read operation. If the pulse/read option is specified by the second and third parameters, the active central control performs a read of the active central control immediately following the pulse instruction. The GCP instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor.

Format:

```
                    ITEMS($2) NOSTORE
CCPULSE POINT($1),ITEMS($3),EXPECT($5)
                    WORD($4)
```

Characteristics of Parameters:

POINT — Specifies the symbolic reference to any of the 360 possible pulse points. The $1 is the symbolic reference.

ITEM — Specifies the symbolic location of an item in the active central control. The $2 is an item.

ITEMS — Specifies the symbolic locations of items in the active central control. The $3 is a list of items all in the same word.

WORD — Specifies the symbolic location of a word in the active central control. The $4 is the address.

EXPECT — Specifies the expected results. The $5 is any arithmetic expression that expresses the expected results.

NOSTORE — No raw data will be stored.

Example:

```
CCPULSE POINT(1PP_CLSBY)
```

This statement causes the active central control to pulse the point 1PP_CLSBY.

CCRDZ

4.65 The description of the CCRDZ statement includes:

Function:

The CCRDZ statement causes the active central control to perform a series of read operations of certain areas in the standby central control. The results from the read are passed along with the mask of the read/write items to the DCON where nonzero results are recorded as failures. This statement is generally used with the CCCLR statement.

Format:

```
CCRDZ AREA($1)
```

Characteristics of Parameters:

AREA — Specifies the area of this standby central control. The $1 is either ALL or any combination of INAD, UBMB, BB, and AU.

INAD — The read of the read/write items of the registers of the instruction address area in the standby central control. The registers and the order in which they are tested follow:

(1) Order word register (OW)
(2) Buffer order word register left-half (BOL)
(3) Buffer order word register right-half (BOR)
(4) Half word register (HWR)
(5) Auxiliary buffer word register right-half (ABR)
(6) Auxiliary buffer word register left-half (ABL)
(7) Data address register (DAR)
(8) Save data address register (SDA)
(9) Index adder augend register (IA)
(10) Add-one register (AOR)
(11) Auxiliary current address register (ACR_B)
(12) Current address register (CAR_B)
(13) Save current address register (SCA_B)
(14) Save program address register (SPA_B)
(15) Program address register (PAR_B).

UBMB — The read of the read/write items of the registers of the mask and unmasked bus area in the standby central control. The registers and the order in which they are tested follow:

(1) L register (LR)
(2) F register (FR)
(3) G register (GR)
(4) K register (KR)
(5) X register (XR)
(6) Y register (YR)
(7) Z register (ZR)
(8) J register (JR)
(9) Peripheral unit enable register (ER)
(10) P register's most significant bits (PRM_B)
(11) P register's least significant bits (PRL)
(12) Central pulse distributor (CPD) reply register (RR)
(13) Stack register (SR)
(14) Argument register (AG)
(15) Sign and homogeneity flip-flips (CF).

BB — The read of the read/write items of the registers of the buffer bus area in the standby central control. The registers and the order in which they are tested follow:

(1) Central control error summary register (CES)
(2) Peripheral matcher 0 (PM0)
(3) Peripheral matcher 1 (PM1)
(4) Interrupt level activity flip-flops (ILA)
(5) Interrupt level request register (ILR)
(6) Interrupt inhibit register (INH)
(7) Interrupt request register (INJ)
(8) Interrupt sequence register (INR)
(9) Interrupt source register (INS)
(10) Millisecond clock (MCL)
(11) Processor configuration sanity timers (SAT)

(12) Peripheral unit error summary (PES)
(13) Processor configuration control register (PCR)
(14) Central control status flip-flop (CSC)
(15) Peripheral unit sequencer control (PSC)
(16) Instruction fetch register (IFR)
(17) Instruction execution register (IER)
(18) Memory address decoder flip-flop (MDF)
(19) CPD diagnostic and echo register (DE)
(20) Stack counter (SC_B)
(21) Lower protected area bound register (LPA)
(22) Upper protected area bound register (UPA)
(23) Delay and limited run register (DLR)
(24) Buffer register shadow (BRS).

AU — The read of the read/write items of the registers of the auxiliary unit area in the standby central control. The registers and the order in which they are tested follow:

(1) Auxiliary unit and central control block counters (ABK)
(2) Auxiliary unit store access verify expect group (SVG)
(3) Auxiliary unit write slow register (AWS_B)
(4) Auxiliary unit enable buffer register (EBG)
(5) Auxiliary unit request inhibit group (RIG)
(6) Auxiliary unit reply register (ARR_B)
(7) Auxiliary unit write fast register (AWF_B)
(8) Auxiliary unit enable verify expect group (EVG)
(9) Auxiliary unit miscellaneous group A (AMA)
(10) Auxiliary unit miscellaneous group B (AMB)
(11) Auxiliary unit miscellaneous group C (AMC)
(12) Bit control register 0 (BC0)
(13) External match register 0 (ME0)
(14) Bit control register 1 (BC1)
(15) Internal match register 1 (MI1)
(16) External match register 1 (ME1)
(17) Match mode control register (MMR)
(18) Match summary register (MSR)
(19) Matcher 0 control register (M0R)
(20) Matcher 1 control register (M1R)
(21) Combined mask 0 (CM0)
(22) Combined mask 1 (CM1)
(23) Internal match register 0 (MI0)
(24) Auxiliary unit store address register (AAS_B)
(25) Auxiliary unit request group (RQG)
(26) Auxiliary unit verify receive group (VRG).

ALL — The read of read/write items of the registers from the areas (INAD, UBMB, BB, AU) in the standby central control.

Example:

CCRDZ AREA(INAD,UBMB)

This statement causes the active central control to perform a series of read/write operations of the INAD and UBMB areas in the standby central control.

CCREAD

4.66 The description of the CCREAD statement includes:

Function:

The CCREAD statement causes the active central control to read an item, a group of items (of the same address), or a word in the standby central control. If EXPECT is specified, the reply, the mask, and the expected results are passed to the DCON program for processing raw results. If NOSTORE is specified, only the read is performed.

Format:

```
        ITEM($1), NOSTORE
CCREAD  ITEMS($2),EXPECT($4)
        WORD($3)
```

Characteristics of Parameters:

ITEM — Specifies the symbolic location of an item in the standby central control. The $1 is the item name.

ITEMS — Specifies the symbolic location of a list of items in the standby central control. The $2 is a list of items all in the same word.

WORD — Specifies the symbolic location of a word in the standby central control. The $3 is the address.

EXPECT — Specifies the expected results of the read. The $2 is any arithmetic expression that expresses the expected results.

NOSTORE — No raw data will be stored.

Example:

```
CCREAD WORD(ST1XR),EXPECT(-0)
```

This statement causes the active central control to read the X register (XR) in the standby central control. The expected results are all 1s (-0).

CCREC

4.67 The description of the CCREC statement includes:

Function:

The CCREC statement dynamically creates a small program and executes it from the opposite community specified by bit 21 of ADD data (ie, call store if bit 21 is set). Before the small program is generated, the program store or call store bus selection flip-flop PBO or CBO is reset. If the test is INH, the program store or call store bus selection flip-flop PBT or CBT is set; if the test is EN, the PBT or CBT is reset. If the office is a 2-wire office, TOLL in the upper protected area bound register (UPA) is set. After the program is executed, TOLL, PBO or CBO, and PBT or CBT are reset. Then test results are processed according to the TEST parameter. If the test is EN, the expected value is RDATA; if the test is INH or CLK, the expected value is the complement of RDATA. The small program created by this statement configures PBO and PBT or CBO and CBT, changes the original K-code to a desired K-code specified in ADD data, and sets the maintenance flip-flop in a test store whose community is specified by bit 21 of ADD data. Store location specified by the ADD data is initialized with data specified by RDATA data. Then the program starts the standby central control operational clock by writing into its delay and limited run register (DLR). The standby central control executes the second cycle of a load (L) instruction. The last thing the program does is restore system configuration and original store data. The CCREC must be immediately preceded by a store-to-central control initialization statement.

Format:

CCREC TEST($1,$2),STORE(ADD($3),RDATA($4)),ACTCC(ML,OPTIONS($5)),STBCC(DEST($6))

Characteristics of Parameters:

TEST — Specifies the manner in which the test results are to be processed. The $1 is INH, CLK, or EN. If $1 is EN, the expected value is RDATA. If $1 is INH or CLK, the expected value is the complement of RDATA. The CLK should be used if a central control operational clock phase has been initiated. If specified, $2 is ITS, and causes the parity bits P1 and P2 to be zero independent of data.

STORE — Specifies the address of the store and the expected results.

ADD — Specifies the address of store. The $3 is any arithmetic expression which expresses the complete store address.

RDATA — Specifies the expected result. The $4 is any arithmetic expression which expresses the expected result.

ACTCC — Specifies the options for the ML instruction executed by the active central control. The ML instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor.

OPTIONS — Specifies the options for the ML instruction. The $5 is any combination of options for the ML instruction. Table A is a list of options for the MS and ML instructions.

STBCC — Specifies the destination register for the second half of the load instruction executed by the standby central control.

DEST — Specifies the destination register. The $6 is any of the following:

(a) Auxiliary buffer order word register, right-half (ABR)
(b) Auxiliary buffer order word register, left-half (ABL)
(c) Data buffer register (BR)
(d) Auxiliary unit write register(s) (AUW).

Example:

```
CCREC_TEST(EN),STORE(ADD(1DGCS | 1DGK3 | 1DGA1),RDATA(1DGD52)),
ME      ACTCC(ML,OPTIONS(R,M,IPKA)),STBCC(DEST(BR))
```

This statement creates and executes a small program in the program store community. The store address is the result of ORing the values of 1DGCS, 1DGK3, and 1DGA1. After the program has been executed, the result is compared to the value of 1DG52. The small program configures CBO and CBT and sets maintenance flip-flops in a call store test store. The store location at the address resulting from ORing the values of 1DGCS, 1DGK3, and 1DGA1 is initialized with the value of 1DGD52. This program also starts the standby central control. The active central control executes an ML instruction with R, M, and IPKA options; the standby central control executes the second half of an L instruction, putting the value of 1DGD52 in the standby central control BR.

CCRISTEP

4.68 The description of the CCRISTEP statement includes:

Function:

The CCRISTEP statement puts the standby central control in step with the active central control to execute the code specified in the statement. The code is executed three times. For the first two times, directed matching is set up. For the third time routine matching is set up. The instructions to be matched are specified by the parameters, as well as what is matched during directed matching. For the first two passes through the code, the match summary register (MSR) of the standby central control is passed to the DCON program for outputting with the expected results = M(ST1W) and a mask = M(ST1M1I, ST1MOI, ST1Y, ST1X, ST1W, ST1S, ST1L, ST1T). For the third pass, the standby MSR is passed to DCON for outputting with the expected results = M(ST1W) and a mask = M(ST1M1E, ST1M1I, ST1MOE, ST1MOI, ST1Y, ST1X, ST1W, ST1S, ST1L, ST1T). The MSR of the active central control is also passed to DCON with expected result = 0 and a mask = M(AC1M1E, AC1M1I, AC1MOE, AC1MOI).

Format:

CCRISTEP $1,$2,INSTR($3($4))

Characteristics of Parameters:

$1 — Specifies a quoted test string which is a descriptive title of the particular test.

$2 — Specifies any one of the valid instruction types: Y, X, W, S, L, or T. This is used to specify the directed matching MATCH points. Refer to Section 254-201-030 (Central Control—Description) and Section 254-201-031 (Central Control—Theory) for more information.

INSTR — Specifies the 1A Processor assembly language pseudo-operation and instruction and specifies if matching is to be performed. The 1A Processor assembly language pseudo-operations and instructions are described in Section 254-280-020, Assembly Language—Description, 1A Processor. The $3 is one of the following:

MLONG — Matched instruction, FLONG on
RLONG — RELLOAD on, FLONG on
MRLONG — Matched instruction, RELLOAD on, FLONG on
MSHORT — Matched instruction, FSHORT on
RSHORT — RELLOAD on, FSHORT on
MRSHORT — Matched instruction, RELLOAD on FSHORT on
LONG — FLONG on
SHORT — FSHORT on.

The $4 is the 1A Processor assembly language instruction to be executed. The INSTR parameter may be repeated any number of times.

Example:

```
CCRISTEP_"TEST OF LOAD (LO2A)",L,INSTR(  LW X,MEM3)),
ME          INSTR(MLONG(  L GJ, 0(X)))
```

This statement puts the standby central control in step with the active central control to execute two instructions. The second instruction will be matched using the L-type match points.

```
LW X,MEM3
L  GJ,0(X)
```

Both instructions are described in Section 254-280-020, Assembly Language—Description, 1A Processor. Both instructions are long; the LW instructions uses relative addressing and the L instruction is a matched instruction. These instructions are executed twice and the L-type INSTR is matched by direct matching. Then the instructions are executed and the L-type INSTR is matched by routine matching.

CCRUN

4.69 The description of the CCRUN statement includes:

Function:

The CCRUN statement causes the active central control to set ST1LIMIT in the delay and limited run register (DLR) of the standby central control to the number of cycles specified in the CYCLE parameter. The standby central control is then started by a write to the buffer pulse source (BPS) setting ST1IDLRCA. If STAMA is specified, start match (1PP_STAMA) is pulsed with a GCP instruction before ST1IDLRCA is set. If LR is specified, the data specified as the LR parameter is loaded into the active central control L register (LR) before ST1IDLRCA is set. The GCP instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor.

Format:

```
                  STAMA
CCRUN CYCLE($1),LR($2)
```

Characteristics of Parameters:

CYCLE — Specifies the number of cycles the standby central control is to run. The $1 is a number 1 through 15.

STAMA — Specifies the 1PP-STAMA be pulsed before ST1IDLRCA is set.

LR — Specifies that the value of $2 is to be loaded in the active central control LR before ST1IDLRCA is set. The $2 is any arithmetic expression.

Examples:

(a) CCRUN CYCLE(1)

This statement causes the active central control to set ST1LIMIT in the DLR of the standby central control to 1. The standby central control is then started by a write to the BPS setting ST1IDLRCA and runs 1 cycle.

(b) CCRUN CYCLE(1),STAMA

This statement causes the active central control to set ST1LIMIT in the DLR of the standby central control to 1. The 1PP_STAMA is pulsed by a GCP instruction. Then the standby central control is started by a write to the BPS setting ST1IDLRCA and runs 1 cycle.

(c) CCRUN CYCLE(1),LR(0)

This statement causes the active central control to set ST1LIMIT in the DLR of the standby central control to 1. A 0 is loaded into the LR of the active central control. Then the standby is started by a write to the BPS setting ST1ILDRCA and runs 1 cycle.

CCRWBR

4.70 The description of the CCRWBR statement includes:

Function:

The CCRWBR statement runs the standby central control with a desired data pattern in its data buffer register (ST1BR). This is accomplished by writing the ST1DELAY and the ST1LIMIT of the delay and limited run register (DLR) of the standby central control to the desired number of cycles. The standby central control is then started by setting the ST1IDLRCA in its buffer pulse source register (BPS). While the standby central control delays, the active central control writes the desired data pattern in the data buffer register of the standby central control (ST1BR).

Format:

CCRWBR DATA($1),CYCLE($2),$\overline{\text{DELAY}}$($3)

Characteristics of Parameters:

DATA — Specifies the data pattern. The $1 is any arithmetic expression.

CYCLE — Specifies the number of cycles the standby central control is to run. The $2 is a number 1 through 15.

DELAY — Specifies the number of cycles the standby central control will delay before running. If not specified, there is a 3-cycle delay. The $3 is a number 1 through 15.

Examples:

(a) CCRWBR DATA (-0),CYCLE(1)

This statement causes the active central control to set ST1LIMIT to 2 and ST1DELAY to 3 in the DLR of the standby central control. During the 3-cycle delay of the standby central control, the active central control writes all 1s (-1) into ST1BR of the standby central control. The standby central control is then started by setting the ST1IDLRC in its BPS. The standby central control then runs for 1 cycle.

(b) CCRWBR DATA(-0),CYCLE(1),DELAY(2)

This statement causes the active central control to set ST1LIMIT to 1 and ST1DELAY to 2 in the LDR of the standby central control. During the 2-cycle delay of the standby central control, the active central control writes all 1s (-1) into ST1BR of standby central control. The standby central control is then started by setting the ST1IDLRCA in its BPS. The standby central control then runs for 1 cycle.

CCSC_ST

4.71 The description of the CCSC_ST statement includes:

Function:

The CCSC_ST statement writes 1DLECS into the standby central control memory address decoder flip-flop (MDF), sets the standby central control instruction fetch register (IFR) to indicate a full stack, writes a POP instruction into the standby central control buffer order word register left-half (BOL), and initializes the standby central control stack register (SR) with test call store stack address. This statement also initializes all other standby central control address source registers with the complement of the address.

Format:

CCSC_ST ADD($1),OPTIONS($2),INHCLK($3)

Characteristics of Parameters:

ADD — Specifies the stack address. The $1 is any arithmetic expression that expresses a complete stack address to be written.

OPTIONS — Specifies the mode, parity, and store timing during the write (MS) instruction. The MS instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor. The $2 is a list of options. Table A is a list of options for the MS and ML instructions.

INHCLK — If specified, specifies a clock phase to be inhibited during the test. The $3 is 45S65, 7T9, 8T10, or 12T0.

Example:

CCSC_ST ADD(1DGCS | 1DGK0 | O(4400))

This statement writes the stack address (the address resulting from ORing the values of 1DGCS, 1DGK01, and octal 4400) in the standby central control SR.

CCSTANTI

4.72 The description of the CCSTANTI statement includes:

Function:

The CCSTANTI statement tests to the read/write access of the analog timer inhibit flip-flop (ANTI). The ANTI is cleared and a test is performed to verify that ANTI is 0. The ANTI is then set and it is verified that ANTI is 1.

Format:

CCSTANTI

Characteristics of Parameters:

This statement has no parameters.

Example:

CCSTANTI

This statement clears ANTI and tests to see that ANTI is 0; sets ANTI and tests to see that ANTI is 1; then clears ANTI and generates a deliberate processor configuration trigger and tests to see that ANTI is 1.

CCST_ABL

4.73 The description of the CCST_ABL statement includes:

Function:

The CCST_ABL statement initializes the standby central control to be ready to receive data as follows:

(1) From call store reply bus (CSR) to auxiliary buffer order word register left-half (ABL)
(2) From CSR to auxiliary buffer order word register right-half (ABR)
(3) From program store reply bus left-half (PXL) and program store reply bus right-half (PSR) to ABL and ABR.

The following standby central control registers are initialized as:

(1) Addressing state in the instruction fetch register (IFR)
(2) Stall instruction in the buffer order word register left-half (BOL), buffer order word register right-half (BOR), and half word register (HWR)
(3) Specified data in the ABL, ABR, data buffer register (BR), auxiliary unit write slow register (AWS), and auxiliary unit write fast register (AWF)
(4) Clock error group register (CLE) is initiated to inhibit any specified clock phase.

Format:

CCST_ABL ADD($1),REG_INIT($2),ASW($3),INHCLK($4)

Characteristics of Parameters:

ADD — Specifies the call store or program store address. The $1 is any arithmetic expression which expresses the complete call store or program store address.

REG_INIT — Specifies the data to be written. The $2 is any arithmetic expression which expresses the data.

ASW — If specified, specifies sense of all-seems-well (ASW) test. The $3 is PASS or FAIL. If not specified, it is PASS.

INHCLK — If specified, specifies a central control clock phase to be inhibited during the test. The $4 is 7T13.

Example:

CCST_ABL ADD(1DGCS | 1DGK22 | 1DGA0),REG_INIT(O(70707070)),INHCLK(7T13)

This statement initializes the standby central control to receive data from the address resulting from the ORing of the values of 1DGCS, 1DGK22, and 1DGA0. Octal 70707070 is written into the standby central control's ABL, ABR, BR, ASW, and AWF. Central control long clock phase 7T13 is inhibited.

CCST_AUW

4.74 The description of the CCST_AUW statement includes:

Function:

The CCST_AUW statement initializes the standby central control to be ready to receive data from a call store or program store into its auxiliary unit write slow register (AWS) or auxiliary unit write fast register (AWF). In addition, the following occurs:

(a) The standby central control auxiliary unit request register (ARR) is set to 1.
(b) Auxiliary unit inhibit request register (RIG) is cleared.
(c) Auxiliary unit sequence is set to state 1.
(d) The auxiliary unit inhibit verify mismatch flip-flop (FF) is set.
(e) The auxiliary unit store address register (AAS) contains the desired address.
(f) The standby central control runs 1 cycle to initialize the right-half program store (RPS) and right-half program store reply (RPSR) FFs and advance the auxiliary unit sequencer.
(g) The instruction fetch register (IFR) indicates a full stack.
(h) The standby central control gives up the next cycle and the data buffer register (BR), auxiliary buffer order word register left-half (ABL), auxiliary buffer order word register right-half (ABR), AWS, and AWF are initialized with the specified data.
(i) The clock error group register (CLE) is initialized to inhibit any specified clock phase.

Format:

CCST_AUW ADD($1), REG_INIT($2),$\overline{\text{ASW}}$($3),$\overline{\text{INHCLK}}$($4)

Characteristics of Parameters:

ADD — Specifies the call store or program store address. The $1 is any arithmetic expression which expresses the complete call store or program store address.

REG_INIT — Specifies data to be written. The $2 is any arithmetic expression which expresses the data.

ASW — If specified, specifies sense of ASW test. The $3 is PASS or FAIL. If not specified, it is PASS.

INHCLK — If specified, specifies a central control clock phase to be inhibited during the test. The $4 is 7T13.

Example:

CCST_AUW ADD(1DGCS | 1DGK03 | 1DGA25),REG_INIT(O(07070707))

This statement initializes the standby central control to receive data from the address resulting from the ORing of the values of 1DGCS, 1DGK03, and 1DGA25. This address is placed in the standby central control AAS. Octal 07070707 is written into the standby central control BR, ABL, ABR, AWS, and AWF.

CCST_BR

4.75 The description of the CCST_BR statement includes:

Function:

The CCST_BR statement initializes the standby central control to be ready to receive data from a call store or program store into its data buffer register (BR). This statement also initializes the standby central control BR, auxiliary buffer order word register left-half (ABL), auxiliary buffer order word register right-half (ABR), auxiliary unit write slow register (AWS), and auxiliary unit write fast register (AWF) with the specified data. The standby central control data address register (DAR) is initialized with the specified address of the data to be read. The standby central control instruction fetch and execution sequencers are initialized to execute the second half of the read instruction (L L,0). The clock error group register (CLE) is initiated to inhibit any specified clock phase.

Format:

CCST_BR ADD($1),REG_INIT($2),$\overline{\text{ASW}}$($3),INHCLK($4)

Characteristics of Parameters:

ADD — Specifies the call store or program store address. The $1 is any arithmetic expression which expresses the complete call store or program store address.

REG_INIT — Specifies the data. The $2 is any arithmetic expression which expresses the data.

ASW — If specified, specifies sense of ASW test. The $3 is PASS or FAIL. If not specified, it is PASS.

INHCLK — If specified, specifies a central control clock phase to be inhibited during the test. The $4 is 7T13.

Example:

CCST_BR ADD(1DGPS | 1DGK34 | 1DGA1),REG_INIT(O(25252525))

This statement places the address resulting from the ORing of the values of 1DGPS, 1DGK34, and 1DGA1 (this is the address from which data is to be received) in the standby central control DAR and places octal 25252525 in the standby central control BR, ABL, ABR, AWS, and AWF.

CCSWCC

4.76 The description of the CCSWCC statement includes:

Function:

The CCSWCC statement switches central control activity by pulse source. Immediately after the switch, a test is performed to verify that a switch actually occurred. If the test fails, one of two actions may occur:

(1) A DTJUMP to a specified label is performed.

(2) If no label is specified, an additional attempt is made to switch central control activity using the stop-start sequencer in the standby central control. This is called a determined switch.

Format:

CCSWCC $\overline{\text{FAIL}}$($1)

Characteristics of Parameters:

FAIL — Specifies the location to which a DTJUMP is made if the central control switch fails. The $1 is a label defined on a DTDEST statement.

Example:

CCSWCC FAIL (CCDG89ET)

This statement switches central control activity by pulse source and performs a test to verify that the switch actually occurred. If the test fails, a DTJUMP is made to the location with the DTDEST label CCDG89ET.

CCTRAN

4.77 The description of the CCTRAN statement includes:

Function:

The CCTRAN statement dynamically creates a small program and executes it from the opposite community specified by bit 21 of ADD data; ie, call store if bit 21 is set. Before the small program is generated, the program store or call store bus selection flip-flop PBO or CBO is reset. If the office is a 2-wire office, TOLL in the upper protected area bound register (UPA) is set. After this program is executed, TOLL, PBO or CBO and PBT or CBT are reset.Then test results are processed according to the test parameter. If the test is EN, the expected value is RDATA or WDATA; if the test is INH, the expected value is the complement of RDATA or WDATA. The small program created by this statement configures PBO and PBT or CBO and CBT, changes the original K-code to a desired K-code specified in ADD data, and sets the maintenance flip-flop in a test store whose community is specified by bit 21 of ADD data. Store location specified by the ADD data is initialized with data specified by RDATA or WDATA data. The ACTPS or ACTCS in the activity flip-flop group (ACT) is set in both central controls. Then the program starts the standby central control operational clock by writing into its delay limited run register (DLR). The standby central control executes the MS/ML instruction with the specified options. The last thing the program performs is to restore system configuration and original data. The CCTRAN must be immediately preceded by a central control-to-store initialization statement.

Format:

CCTRAN TEST($1),STORE(ADD($2),RDATA($3),ACTCC(OPTIONS($4)),

STBCC(SOURCE($5),WDATA($6),OPTIONS($7))

Characteristics of Parameters:

TEST — Specifies the manner in which the test results are to be processed. If $1 is INH or EN, $1 is EN; the expected value is RDATA or WDATA. If $1 is INH, the expected value is the complement of RDATA or WDATA.

STORE — Specifies the address of the store and the expected results.

ADD — Specifies the address of the store. The $2 is any arithmetic expression which expresses the complete store address.

RDATA — Specifies the expected results for an address transmission test. The $3 is any arithmetic expression which expresses the expected result.

ACTCC — Specifies the options for the MS/ML instruction executed by the active central control. The MS and ML instructions are described in Section 254-280-020, Assembly Language—Description, 1A Processor.

OPTIONS — Specifies the options for the MS/ML instruction. The $4 is any combination of options for the MS/ML instruction. Table A is a list of options for the MS and ML instructions.

STBCC — Specifies the source register and the options and data for the MS/ML instruction for the standby central control. The MS and ML instructions are described in Section 254-280-020, Assembly Language—Description, 1A Processor.

SOURCE — Specifies the source register. The $5 is any of the following:

(a) Data Buffer Register (BR)
(b) Program Address Register (PAR)
(c) Data Address Register (DAR)
(d) Auxiliary Unit Store Address Register (AAS)
(e) Auxiliary Unit Reply Register (ARR)
(f) Stack Counter (SC).

WDATA — Specifies the data for the MS/ML instruction. The $6 is any arithmetic expression which expresses the data.

OPTIONS — Specifies the options for the MS/ML instruction and the expected result from a data transmission test. The $7 is any combination of options for the MS/ML instruction. Table A is a list of options for the MS and ML instructions.

Example:

```
CCTRAN_TEST(EN),STORE(ADD)1DGCS | 1DGK3 | 1DGA0),ACTCC(OPTIONS(C,M,W)),
ME        STBCC(SOURCE(BR),WDATA(1DGD70),OPTIONS(M,W,C,IP1,IP2))
```

This statement creates and executes a small program in the program store community. The store address is the result of ORing the value of 1DGCS, 1DGK3, and 1DGA0. After the program has been executed, the result is compared to 1DG070. The small program sets the maintenance flip-flops in a call store test store. The store location at the address resulting from ORing the value of 1DGCS, 1DGK3, and 1DGA0 is initialized with zeros. The ACTCS is set in both central controls. This program also starts the standby central control. The active central control executes an MS/ML instruction with C, M, and W options; the standby central control executes an MS/ML instruction with M, W, C, IP1, and IP2 options. The source register for the standby central control is BR and the data to be written is the value of 1DGD70.

CCWALK

4.78 The description of the CCWALK statement includes:

Function:

The CCWALK statement causes the active central control to perform a series of write/read operations into the standby central control. The write operation in the first part of the couplet performs a write of a 0 right-adjusted in a field of 1s, leaving all other bits outside the field zero. The read operation, the second part of the couplet, consists of reading the field and expecting a pattern to be written. This sequence of operations is repeated, one bit position shifting 0 left one bit each time until the 0 has been shifted throughout the field. A 1 may be walked through a field of 0s in the same manner.

Format:

```
CCWALK FIELD($1),WALK($2)
```

Characteristics of Parameters:

FIELD — Specifies the symbolic location in the standby central control. The $1 is the location name.

WALK — Specifies walk a 1 through a field of 0s or walk a 0 through a field of 1s. The $2 is 1THRU0 or 0THRU1.

Example:

CCWALK FIELD(ST1TA),WALK(1THRU0)

This statement performs a series of write/read operations in the ST1TA location in the standby central control. For the first write, the data is a 1 right-adjusted (bit position 0) in a field of 0s. The ST1TA is then read and compared to the data. For the second write, the data is a one in bit position 1 in a field of zeros. The ST1TA is then read and compared to the data. The write/read and shifting the 1 left on bit position continues until the last read, when the data written is a 1 left-adjusted (bit position 23). The ST1TA is then read and compared to the data for the final time.

CCWRITE

4.79 The description of the CCWRITE statement includes:

Function:

The CCWRITE statement causes the active central control to store a word of data in the standby central control at the location specified. The data is stored with a store secure instruction in the standby central control without product or insertion masking. The store secure instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor.

Format:

CCWRITE WORD($1),DATA($2)

Characteristics of Parameters:

WORD — Specifies the symbolic word location in the standby central control. The $1 is the symbolic location.

DATA — Specifies the data to be stored. The $2 is any arithmetic expression that expresses the data to be written.

Example:

CCWRITE WORD(ST1XR),DATA(-0)

This statement causes the active central control to write all 1s (—0) in the X register in the standby central control (ST1XR). There is no product or insertion masking.

CCXAUSYC

4.80 The description of the CCXAUSYC statement includes:

Function:

The CCXAUSYC statement is used in the central control diagnostic auxiliary unit bus testing phases to test the transmission of sync signals to the auxiliary units. The buffer order word register left-half (BOL) in the standby central control contains the test instruction to be executed by the standby central control when the CCXAUSYC statement is executed. The results of a GCP read of bit 3 (AU1ACTPS) is stored in scratch word DG1DTSCR3 for later interrogation.

Format:

CCXAUSYC

Characteristics of Parameters:

This statement has no parameters.

Example:

CCXAUSYC

This statement tests the transmission of sync signals to the auxiliary units. The results of a GCP read of bit 3 (AU1ACTPS) is stored in DG1DTSCR3.

CCXGCP

4.81 The description of the CCXGCP statement includes:

Function:

The CCXGCP statement executes a GCP instruction in sync with the standby central control execution of the same instruction. The pulse source activity flip-flops (FFs) are reversed so that only the standby central control actually generates a pulse. When this statement is executed during testing pulse source isolation, the standby central control is blocked from actually pulsing a unit, but a reply is looked for in the active central control. If a reply is received, the affected unit is restored immediately. The pulse reply, if present, is stored in scratch word DG1DTSCR1 for later interrogation. This statement is executed with the standby central control not isolated to test transmitting control pulses to the auxiliary units. In this case, bit 19 of DG1DTSCR1 indicates that a nonzero reply was received.

Format:

CCXGCP POINT (AUSCRATCH)

Characteristics of Parameters:

POINT — Specifies the GCP address of the point under test. The $1 is any arithmetic expression which expresses the address of the pulse to be generated. The AUSCRATCH specifies that the address is to be taken from scratch word DG1GCPADDR.

Example:

CCXGCP POINT (1PP_PAU01)

This statement causes the standby central control to generate a control pulse from pulse point 1PP_PAU01. The active central control looks for a reply and if a reply is received, it is stored in scratch word DG1DTSCR1.

CCXNSYNC

4.82 The description of the CCXNSYNC statement includes:

Function:

The CCXNSYNC statement starts the standby central control using the delay-limited run maintenance features. Delay and limit amounts are specified in the statement. After starting the standby central control, the active central control executes the long 1A Processor assembly language instruction specified in the statement. Prior to the execution of this instruction, the K register (KR) is loaded from DG1DTSCR1 indexed by the base address (in register F) and the G register (GR) is loaded from DG1DTSCR2 indexed by the base address (in register F); these can be used for the address and data in the active central control instruction. All stores are forced to appear slow to the active central control. Optionally, the active central control will set (by a GCP) the force auxiliary unit address (AUA) bus (FAUA) flip-flop. The GCP and other long 1A Processor assembly language instructions are described in Section 254-280-020, Assembly Language—Description, 1A Processor.

Format:

CCXNSYNC INSTR($1),DELAY($2),RUN($3),FAUA,NOSYNC

Characteristics of Parameters:

INSTR — Specifies the 1A Processor assembly language instruction to be executed in the active central control. $1 is any long instruction.

DELAY — Specifies the number of cycles the standby central control should delay. The $2 is a number 1 through 15.

RUN — Specifies the number of cycles the standby central control should run. The $3 is a number 1 through 15.

FAUA — Specifies that the FAUA flip-flop is to be set.

NOSYNC — Specifies that no SYNC pulse is to be sent.

Examples:

CCXNSYNC INSTR(SS G,0)(K)),DELAY(8),RUN(2)

This statement causes the active central control to execute the following 1A Processor assembly language instructions: SS G,0(K). The standby central control waits 8 cycles, then runs for 2 cycles.

CHGICC

4.83 The description of the CHGICC statement includes:

Function:

The CHGICC statement reads an internal central control register, saves the present state of an item in that register, then changes that item to a new state. A subsequent RSTICC statement is used to restore the saved item state.

Format:

```
          ITEM(1)        FF(4)
CHGICC ITEMS(2),ST(3),DATA(5)
```

Characteristics of Parameters:

ITEM — Specifies an internal central control register address and bit or contiguous bits to be saved and changed. The $1 is the item name.

ITEMS — Specifies an internal central control register address and noncontiguous bits to be restored from call store scratch. The $2 is a list of item names all in the same word.

ST — Specifies one of six call store scratch addresses for saving the present item state. The $3 is TEMP0, TEMP1, TEMP2, TEMP3, TEMP4, or TEMP5.

DATA — Specifies the new state for the item being changed. The $5 is any arithmetic expression which expresses the data.

FF — Specifies the item being changed is a central control buffer bus flip-flop; $4 specifies the new SET or RESET state via convention of rightmost character "S" or "R." The $4 is the item name with "S" or "R" as the rightmost character.

Example:

```
CHGICC ITEMS(INCUZ,IN1CUG1,IN1CUGS),ST(TEMP2),DATA(0(40))
```

This statement causes the INCUZ, IN1CUG1, and IN1CUGS items in the internal central control register to be saved in the third call store scratch address (TEMP2) for saving the present item state. Then octal 40 is written into the items.

CHKSRDUC

4.84 The description of the CHKSRDUC statement includes:

Function:

The CHKSRDUC statement calls a Data Unit Administration Program (DUAD) subroutine to determine if any DUC is in the system reinitialization (SR) mode. A diagnostic scratch word will be set according to the response from DUAD.

Format:

```
CHKSRDUC
```

Characteristics of Parameters:

This statement has no parameters.

Example:

```
CHKSRDUC
```

This statement calls a DUAD subroutine to determine if any DUC is in the SR mode. A diagnostic scratch word is set according to the response from DUAD.

CKEANTI

4.85 The description of the CKEANTI statement includes:

Function:

The CKEANTI statement tests the clock error (CKE) and analog timer inhibit (ANTI) flip-flops in the processor configuration circuit. Write/read tests are performed on these flip-flops in the processor configuration register (PCR) in the standby central control. This statement, due to hardware restriction, starts the standby central control operational clock before performing the tests and stops the clock afterward. All data generated from the reads of the two flip-flops are stored in diagnostic scratch call store for interrogation by other DL-1 statements.

Format:

CKEANTI

Characteristics of Parameters:

This statement has no parameters.

Example:

CKEANTI

This statement tests the CKE and ANTI flip-flops in the processor configuration circuit by performing write/read tests on the PCR in the standby central control. All data generated from the reads are stored in diagnostic scratch call store.

CLKINH

4.86 The description of the CLKINH statement includes:

Function:

The CLKINH statement tests the operational clock phase inhibit circuitry in the standby central control. The standby central control operational clock is started, the operational clock error detector is initialized, and the clock error detector is read and saved in diagnostic scratch call store. An operational clock phase, specified by $1, is inhibited by control writing a data pattern generated by $1 into the clock error group register (CLE). The clock error indicator is then read and saved in diagnostic scratch call store. The clock error detector is again initialized (synchronized) and the clock error indicator is read and saved in diagnostic scratch call store. The standby central control is then clear stopped, stopping the operational clock and clearing the clock phase inhibit circuitry. The diagnostic scratch call store data is interrogated by other DL-1 statements.

Format:

CLKINH $1

Characteristics of Parameters:

$1 — Specifies an operational clock phase in the form XXTYY, where XX is the leading edge and YY is the trailing edge times. All clock phases specified are of 100 nanoseconds duration. The $1 generates a symbol of the form 1DGIXXTYY which is Datapool defined.

Examples:

(a) CLKINH 00T02

This statement starts the standby operational clock, initializes the operational clock error detector, reads the clock error detector, and saves it in diagnostic scratch call store. The operational clock phase 00T02 is inhibited by control writing the data pattern which is the value of 1DGI00T02 into the CLE. Then the clock error indicator is read and saved in diagnostic scratch call store. The clock error detector is again initialized and the clock error indicator is read and saved in diagnostic scratch call store. The standby central control is then clear stopped, stopping the operational clock and clearing the clock phase inhibit circuitry.

(b) CLKINH 05T07

This statement causes the same actions as the previous statement, except the operational clock phase is 05T07 and the data pattern written is the value of 1DGI05T07.

CLRTUCTF

4.87 The description of the CLRTUCTF statement includes:

Function:

The CLRTUCTF statement calls a Data Unit Fault Recovery Program (DUFR) subroutine to request that DUFR reset the trouble flip-flop in all in-service tape unit controllers (TUCs). The subroutine is called after an SR test.

Format:

CLRTUCTF

Characteristics of Parameters:

This statement has no parameters.

Example:

CLRTUCTF

This statement performs an SR test, then calls a DUFR subroutine to reset the trouble flip-flop in all in-service TUCs.

DCREAD

4.88 The description of the DCREAD statement includes:

Function:

The DCREAD statement reads a file store register with a control read instruction. Data read is ignored or compared to specified expected results and stored as raw data.

Format:

```
        ITEM($1) NOSTORE
DCREAD  ITEM($2),EXPECT($4)
        WORD($3)
```

Characteristics of Parameters:

ITEM — Specifies a file store register address and bit or contiguous bits of register to be read. The $1 is an item name.

ITEMS — Specifies a file store register and bits within that register to be read. The $2 is a list of items all in the same word.

WORD — Specifies a file store register address to be read as a full 24-bit word. The $3 is the address.

EXPECT — Specifies data to be used for expected results. The $4 is any arithmetic expression that expresses the expected results.

NOSTORE — No raw data will be stored.

Example:

```
DCREAD ITEM(FS1SCP),EXPECT(O(777777))
```

This statement causes the FS1SCP item in the file store register to be read and compared to octal 777777.

DCWRITE

4.89 The description of the DCWRITE statement includes:

Function:

The DCWRITE statement writes a full 24-bit word into a file store register with a control write instruction.

Format:

```
DCWRITE WORD($1),DATA($2)
```

Characteristics of Parameters:

WORD — Specifies a file store register address to be written. The $1 is the address.

DATA — Specifies 24-bit word to be written. The $2 is any arithmetic expression which expresses the data.

Example:

```
DCWRITE WORD(FS1ILLA),DATA(62)
```

This statement causes decimal 62 to be written in the FS1ILLA file store register.

DGSCRTP

4.90 The description of the DGSCRTP statement includes:

Function:

The DGSCRPT statement calls a DUAD subroutine to determine if a TUC is assigned to the DGN (diagnostic) function. A diagnostic scratch word will be set according to the response from DUAD.

Format:

DGSCRTP

Characteristics of Parameters:

This statement has no parameters.

Example:

DGSCRTP

This statement calls a DUAD subroutine to determine if a TUC is assigned to the DGN function and sets a diagnostic scratch word according to the response.

DKCODE

4.91 The description of the DKCODE statement includes:

Function:

The DKCODE statement verifies the file store K-code specified for the unit being diagnosed. Proper response for all possible K-codes, both common and unique, for both communities of file stores is verified. Supplementary data words are generated by the DKCODE statement as follows:

Word 0 — Unique K-code of file store being diagnosed.
1 — Common K-code of file store being diagnosed.
2 — Opposite common K-code.
3 — Mate file store unique K-code.
4 — Other community unique K-code.
5 — Other community, other unique K-code.
6 — Other community, common K-code.
7 — Other community, other common K-code.

Each supplementary data word corresponds to a test. The file store under test should recognize K-codes in tests 0 through 2 and not recognize K-codes in tests 3 through 7.

Format:

DKCODE

Characteristics of Parameters:

This statement has no parameters.

Example:

DKCODE

This statement verifies the response of the unit under test to all possible K-codes assigned to file stores to ensure that the unit responds only to the proper K-code. Supplementary data words are generated if required for the results of the tests.

DLRCLSBY

4.92 The description of the DLRCLSBY statement includes:

Function:

The DLRCLSBY statement tests the delay and limited run (DLR) sequencer circuitry in the standby central control. The statement starts the DL sequencer and then stops the operational clock in the standby central control. An operational clock status indicator is then reset and the standby central control is pulse cleared. Operational clock control and status indicators are then read and stored in diagnostic scratch call store. The operational clock is then stopped by control pulse to ensure it is stopped. No tests are generated by this statement. Results stored in diagnostic scratch call store are interrogated by other DL-1 statements to verify proper circuit reactions (functions) and generate test results.

Format:

DLRCLSBY

Characteristics of Parameters:

This statement has no parameters.

Example:

DLRCLSBY

This statement starts the DLR sequencer and then stops the operational clock in the standby. An operational clock status indicator is then reset and the standby central control is pulse cleared. Operational clock control and status indicators are then read and stored in diagnostic scratch call store. The operational clock is then stopped by control pulse.

DLRRUN

4.93 The description of the DLRRUN statement includes:

Function:

The DLRRUN statement tests the delay and limited run (DLR) sequencer in the standby central control. The DELAY and LIMIT counters in the DLR circuit are initialized by the DELAY and LIMIT parameters and then the DLR sequencer is started by buffer pulse source (BPS) control write. The DELAY and LIMIT counters are then read and their values are stored in diagnostic scratch call store. The DELAY and LIMIT counters are read seven times. The operational clock stop-start register (SSR) is then control read and the results stored in diagnostic scratch call store. The standby central control operational clock is then stopped. The results stored in diagnostic scratch call store are interrogated by other DL-1 statements.

Format:

DLRRUN LIMIT($1),DELAY ($2)

Characteristics of Parameters:

LIMIT — Specifies the value to be control written in the DLR LIMIT counter. The $1 is a decimal value of 0 to 15.

DELAY — Specifies the value to be control written in the DLR DELAY counter. The $2 is a decimal value of 0 to 15.

Examples:

(a) DLRRUN LIMIT(2),DELAY(0)

This statement control writes the decimal value 2 into the LIMIT counter and the decimal value 0 into the DELAY counter in the DL circuit, then starts the DLR sequencers by a BPS control write. The DELAY and LIMIT counters are then read seven times and their values are stored in diagnostic scratch call store. The operational clock SSR is then control read and the results stored in diagnostic scratch call store. The standby central control operational clock is then stopped.

(b) DLRRUN LIMIT(15),DELAY(3)

This statement causes the same actions as the previous statement, except the decimal value 15 is control written into the LIMIT counter and the decimal value 3 is control written into the DELAY counter.

DLRSTAT

4.94 The description of the DLRSTAT statement includes:

Function:

The DLRSTAT statement tests the delay and limited run (DLR) sequencer and associated circuitry in the standby central control. The LIMIT and DELAY counters of the DLR circuitry are control written to the values specified by the LIMIT and DELAY parameters and then the DLR sequencer is started by control writing the DLR by buffer pulse source (BPS) access. The stop-start register (SSR) (operational clock stop-start-status) is then control read three times and the results stored in diagnostic scratch call store call store. The operational clock is then stopped. The diagnostic results are interrogated by other DL-1 statements.

Format:

DLRSTAT LIMIT($1),DELAY($2)

Characteristics of Parameters:

LIMIT — Specifies a value to be control written into the DLR LIMIT counter. The $1 is a decimal value of 0 to 15.

DELAY — Specifies a value to be control written into the DLR DELAY counter. The $2 is a decimal value of 0 to 15.

Examples:

(a) DLRSTAT LIMIT(10),DELAY(5)

This statement control writes the decimal value 10 into the LIMIT counter and the decimal value 5 into the DELAY counter of the DLR circuitry, then starts the DLR sequencers by control writing the DLR by a BPS access. The SSR is control read three times and the results stored in diagnostic scratch call store. The operational clock is then stopped.

(b) DLRSTAT LIMIT(2),DELAY(0)

This statement causes the same action as the previous statement, except that the decimal value 2 is control written into the LIMIT counter and the decimal value 0 is control written into the DELAY counter.

DL1PWRMON

4.95 The description of the DL1PWRMON statement includes:

Function:

The DL1PWRMON statement executes the power monitor test sequence and examines the test results for the specified frame. The DL1PWRMON acquires frame information from the diagnostic buffer table.

Format:

DL1PWRMON SMEMN,FRSPI,TYPE

Characteristics of Parameters:

SMEMN — Submember number is used only when required for frames with subunits.

FRSPI — Frame scan point index is used only when required for frames with subunits.

TYPE — Specifies unit type. Presently, only need to specify if called by program store, call store, or disk file.

Example:

DL1PWRMON

This statement calls the power monitor test sequence for the frame being diagnosed.

DMSECR

4.96 The description of the DMSECR statement includes:

Function:

The DMSECR statement reads a 32-word data block from a specified maintenance sector of a disk file in the file store under test. The following supplementary data words are generated for corresponding failing results.

WORD	TEST	EXPECTED RESULT
0	Pulse read interject FF	Expect 0
1	File store request register 0 (DRR0) idle bit	Expect 1
2	Error register 0	Expect all 0s
3	Error register 1	Expect all 0s
4	Error register 2	Expect all 0s
5	Error register 3	Status errors all 0s
6	Read file store read buffer 3 (FSRB3) status word	Start and abort bits
7	All 32 words read correctly. Expect 0s (FAILING READ omits word number 7, PASSING READ omits word number 100 through 137)	
100	Octal word numbers for failing read match corresponding to data block words 0 through 32	
137	Block matching stops on first word mismatching	

Format:

DMSECR DISK($1),FACE($2),TRACK($3),EXPECT($4)

Characteristics of Parameters:

DISK — Specifies disk file to be read of the file store being diagnosed. The $1 is 0, 1, 2, or 3.

FACE — Specifies the face of that disk file. The $2 is 0 or 1.

TRACK — Specifies the track of that disk file. The $3 is a number 0 through 99.

EXPECT — Specifies the data to be expected in each of the 32 words of that data block read from the disk file. The $1 is any arithmetic expression which expresses the data.

Example:

DMSECR DISK(0),FACE(1),TRACK(80),EXPECT(O(77777777))

This statement causes a 32-word block to be read from disk file 0, face 1, track 80 and compared to the expected results of all 1s (octal 77777777). Supplementary data words are generated if required by a failing result.

DMSECW

4.97 The description of the DMSECW statement includes:

Function:

The DMSECW statement writes a 32-word data block in a specified maintenance sector of a disk file in the file store under test. Supplementary data words are generated for corresponding failing results.

WORD	TEST	EXPECTED RESULTS
0	Pulse read interject FF	Expect 0
1	File store request register 0 (DRR0) idle bit	Expect 1
2	Error register 0	Expect all 0s
3	Error register 1	Expect all 0s
4	Error register 2	Expect all 0s
5	Error register 3	Expect all 0s
6	Read file store read buffer 3 (FSRB3) status word	Start and abort bits

Format:

DMSECW DISK ($1),FACE($2),TRACK($3),EXPECT($4)

Characteristics of Parameters:

DISK — Specifies disk file to be written in the file store under test. The $1 is 0, 1, 2, or 3.

FACE — Specifies of that disk file. The $2 is 0 or 1.

TRACK — Specifies of that disk file. The $3 is a number 0 through 99.

DATA — Specifies 24-bit data word to be written in each of the 32 words of the block to be written. The $4 is any arithmetic expression which expresses the data.

Example:

DMSECW DISK(1),FACE(0),TRACK(1),DATA(O(25252525))

This statement causes a 32-word block containing octal 25252525 to be written on 0F 1, face 0, track 1. Supplementary data words are generated if required by a failing result.

DNREAD

4.98 The description of the DNREAD statement includes:

Function:

The DNREAD statement reads a file store register with a normal read instruction. Data read is ignored or compared to specified expected results and stored as raw data.

Format:

```
        ITEM(1) NOSTORE
DNREAD  ITEM($2),EXPECT($4)
        WORD($3)
```

Characteristics of Parameters:

ITEM — Specifies a file store register address and bit or contiguous bits of item to be read. The $1 is an item name.

ITEMS — Specifies a file store register and bits within that register to be read. The $2 is a list of items all in the same word.

WORD — Specifies a file store address to be read as a full 24-bit word. The $3 is the address.

NOSTORE — Indicates the data read is to be ignored.

EXPECT — Specifies data to be used for expected results. The $4 is any arithmetic expression which expresses the expected results.

Example:

DNREAD ITEMS(FS1TT,FS1LL,FS1X),EXPECT(M(FS1LL,FS1X))

This statement causes the items FS1TT, FS1LL, and FS1X in the file store register to be read and compared to the mask of items FS1LL and FS1X.

DNWRITE

4.99 The description of the DNWRITE statement includes:

Function:

The DNWRITE statement writes a full 24-bit word into a file store register with a normal write instruction.

Format:

DNWRITE WORD($1),DATA($2)

Characteristics of Parameters:

WORD — Specifies address of the file store register to be written. The $1 is the file store register address.

DATA — Specifies 24-bit data word to be written. The $2 is any arithmetic expression which expresses the data.

Example:

DNWRITE WORD(FS1ILAA),DATA(62)

This statement causes decimal 62 to be written in the FS1ILAA file store register.

DREU

4.100 The description of the DREU statement includes:

Function:

The DREU statement reads a file store register with a control read instruction. The expected result is any nonzero data, within the mask of the parameter specified.

Format:

```
      ITEM($1)
DREU  ITEMS($2)
      WORD($3)
```

Characteristics of Parameters:

ITEM — Specifies a file store register address and bit or contiguous bits of register to be read. The $1 is an item name.

ITEMS — Specifies a file store register and noncontiguous bits within that register to be read. The $2 is a list of items all in the same word.

WORD — Specifies a file store register address to be read as a full 24-bit word. The $3 is the address.

Example:

```
DREU WORD(FS1IARR)
```

This statement causes the FS1IARR register in the file store to be read and expects it to be nonzero.

DSKCLKCK

4.101 The description of the DSKCLKCK statement includes:

Function:

The DSKCLKCK checks the disk clocks, has no parameters, and generates one test. The DSKCLKCK task routine checks the frame equipage to determine how many disk files are equipped. For each equipped disk file the following applies:

(1) Reads the sector portion of the disk clock counter twice and stores one of these in scratch memory if the two readings agree. If they differ, a third reading is made and this one is stored.

(2) Takes a nominal one sector delay (approximately 340 microseconds).

(3) Reads the sector counter again after the delay using the procedure in Step 1.

(4) Determines whether the reading made after the delay is either +1 or +2 sectors greater than the first reading that was saved. If this is not the case, bit 0, bit 1, bit 2, and bit 3 in the test result are used to indicate the test failed for disk files 0, 1, 2 and 3, respectively.

Format:

```
DSKCLKCK
```

Characteristics of Parameters:

This statement has no parameters.

DTOGGLE

4.102 The description of the DTOGGLE statement includes:

Function:

The DTOGGLE statement writes a single bit into a file store register one or more times with a control write instruction.

Format:

DTOGGLE N($1),BIT($2)

Characteristics of Parameters:

N — Specifies the number of times the control write is repeated. The $1 is a decimal number. ♦Maximum number that can be specified is 99.♦

BIT — Specifies a single bit in a file store register to be written. The $2 is the name of a 1-bit item.

Example:

DTOGGLE N(20),BIT(FS1XX)

This statement causes the FS1XX bit in the file store register to be written 20 times.

DUADRDFL

4.103 The description of the DUADRDFL statement includes:

Function:

The DUADRDFL statement calls a DUAD subroutine to determine if a TUC has failed a read function for a client. A diagnostic scratch word is set according to the response from DUAD.

Format:

DUADRDFL

Characteristics of Parameters:

This statement has no parameters.

Example:

DUADRDFL

This statement calls a DUAD subroutine to determine if a TUC has failed a read function for a client. It sets a diagnostic scratch according to the response from DUAD.

DUCCHK

4.104 The description of the DUCCHK statement includes:

Function:

The DUCCHK statement calls a DUFR subroutine to determine if an equipped DUC is out of service. A diagnostic scratch word is set according to the response from DUFR.

Format:

DUCCHK ACODE($1)

Characteristics of Parameters:

ACODE — Specifies the ACODE of the DUC to be checked for out of service. The $1 is a decimal number between 0 and 15.

Example:

DUCCHK ACODE(12)

This statement calls a DUFR subroutine to determine if the DUC whose ACODE is 12 is out of service.

DUSMREAD

4.105 The description of the DUSMREAD statement includes:

Function:

The DUSMREAD statement is a maintenance-read instruction for the DUS.

Format:

```
          ITEM($1)  NOSTORE
DUSMREAD  ITEMS($2),EXPECT ($4)
          WORD($3)
```

Characteristics of Parameters:

ITEM — Specifies the internal DUS location to be read. This supplies bits 0 through 6 of the auxiliary unit address bus. K-code is supplied by the task routine. The mask for the results is also generated from the attributes of this parameter. The $1 is the item name.

ITEMS — Specifies internal DUS location to be read. This supplies bits 0 through 6 of the auxiliary unit address bus. K-code is supplied by the task routine. The mask for the results is also generated from the attributes of this parameter. The $2 is a list of items all in the same word.

WORD — Specifies the internal DUS location to be read. This supplies bits 0 through 6 of the auxiliary unit address bus. K-code is supplied by the task routine. The $3 is the address.

NOSTORE — If specified, nothing is done with the reply from the DUS.

EXPECT — If specified, this is used to match with the reply from the DUS. The $4 is any arithmetic expression which expresses the expected results.

Example:

DUSMREAD ITEM(DS1ASEQER),EXPECT(M(DS1ASEQER))

This statement causes a read of the DS1ASEQER item in the DUS. The DS1ASEQER supplies bits 0 through 6 of the auxiliary unit address bus and the K-code is supplied by the task routine. This result is compared with the mask of the item.

DUSMWRITE

4.106 The description of the DUSMWRITE statement includes:

Function:

The DUSMWRITE statement is a maintenance-write instruction for the DUS.

Format:

DUSMWRITE WORD($1),DATA($2)

Characteristics of Parameters:

WORD — Specifies the internal DUS location to be written. This supplies bits 0 through 6 for the auxiliary unit address bus. K-code is supplied by the task routine. The $1 is the address.

DATA — Specifies 24 bits of data to be written into the DUS location specified by WORD. This supplies bits 0 through 23 of the auxiliary unit write bus. The $2 is any arithmetic expression which expresses the data.

Example:

DUSMWRITE WORD(DS1SUR),DATA(O(25252525))

This statement causes octal 25252525 to be written into DS1SUR in the DUS. The DS1SUR supplies bits 0 through 6 of the auxiliary unit address bus and the K-code is supplied by the task routine.

DUSOARIN

4.107 The description of the DUSOARIN statement includes:

Function:

The DUSOARIN statement adds the relative address of the start of scratch memory, used for autonomous block transfers of the ADS, to the address of scratch memory (in register F) and puts the sum into the DUS — output address register (OAR).

Format:

DUSORIN $1,$2,$\overline{3}$

Characteristics of Parameters:

$1 — Specifies the Datapool name given to the start of scratch memory for ADS autonomous block transfers.

$2 — Specifies the Datapool name for either the read or write bits in the OAR.

$3 — Specifies the Datapool name for either the read or write bits in the OAR.

Example:

DUSOARIN DG1ADSLK,DS1OARW,DS1OARR

This statement adds the address of DG1ADSLK (the start of scratch memory for ADS autonomous block transfers) to the base address (in register F) and puts the sum plus the DUS read and write bits (DS1OARR and DS1OARW) in the OAR register.

DUSOAROT

4.108 The description of the DUSOAROT statement includes:

Function:

The DUSOAROT statement reads the contents of the DUS output address register (OAR). The contents is compared to the parameter(s) supplied with the statement after adding the address of scratch memory (in register F) to the parameters supplied with the statement. If the DUS OAR content is not equal to the parameters plus address of scratch memory, DUSOART will:

(1) Force the mismatch data to all one (O(77777777)

(2) Record the actual mismatching bits as a supplementary information (SI) word

(3) Force a diagnostic termination to prevent an unknown memory location from being written or read.

Format:

DUSOAROT $1, $2,$\overline{3}$

Characteristic of Parameters:

$1 — Specifies the expected relative.

$2 — Specifies the Datapool name for either the read or write bits in the OAR.

$3 — Same as $2.

Example:

DUSOAROT DG1ADSBLK,DS10ARW

This statement reads the DUS-OAR and compares its contents to the value obtained by adding the contents of register F to the logical OR of the parameters DG1ADSBLK and DS10ARW.

DWNAME

4.109 The description of the DWNAME statement includes:

Function:

The DWNAME statement writes a new name in the K-code register of a file store, with a control-write instruction. Optional parameters MASK and EXPECT may be used to verify the K-code written, and to obtain raw data according to expected results.

Format:

DWNAME NAME($1),KCODE($2),MASK($3),EXPECT($4)

Characteristics of Parameters:

NAME — Specifies the new name to be written. The $1 is any of 32 valid file store K-codes defined in Datapool. If $1 is XREG, it specifies the new name in the K-code of the file store being diagnosed.

KCODE — Specifies the file store frame which is to have its K-code register changed. The $2 is the same as $1 described for NAME.

MASK — Specified with EXPECT and specifies that verification of the new name or that the name did not change is accomplished by a GCP read. The $3 is the same as $1 described for NAME. The GCP instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor.

EXPECT — Specified with MASK and specifies that verification of the new name or that the name did not change is accomplished by a GCP read. The $4 is the same as $1 described for NAME. The GCP instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor.

Example:

DWNAME NAME(XREG),KCODE(XREG),MASK(M(AU1KCDPS)),EXPECT(0)

This statement causes the K-code of the file store under diagnosis to be written in itself. The new name is verified by a GCP read.

EDINIT

4.110 The description of the EDINIT statement includes:

Function:

The EDINIT statement tests the standby central control operational clock error detector initializing circuitry. The standby central control operational clock and its error-detecting circuitry are started. The error detector is then repetitively initialized. Each time, the error detector indicator is read to verify that no error was generated due to an initializing failure. If no failure occurs before the end of the test loop, the last read of the error indicator is stored in diagnostic scratch call store. If a failure occurs during the test loop, the failing result is stored in diagnostic scratch call store. The operational clock is stopped. The diagnostic scratch call store result is interrogated by other DL-1 statements.

Format:

EDINIT

Characteristics of Parameters:

This statement has no parameters.

Example:

EDINIT

This statement starts the standby central control operational clock and its error detecting circuitry, then repeatedly initializes and reads the error detector indicator. If a failure due to an initializing failure occurs before the end of the test loop, the failing result of the error indicator read is stored in diagnostic scratch call store; otherwise, the last read of the error indicator is stored in diagnostic scratch call store. The operational clock is stopped.

▶EQUIPCHK

4.111 The description of the EQUIPCHK statement includes:

Function:

The EQUIPCHK task routine is used to determine if unit 1XLUSD29 is either in the growth or operational state and whether the unit 1XLUSD29 is for a 2-wire or 4-wire office application. Compool symbol XL1MCDMTHG is compared with the value of parameter three (1XLMCSD29). If a comparison is found it indicates that the master control console panel SD-5A029 is for a 2-wire office application. A noncomparison indicates that the master control console panel SD-5A029 is for a 4-wire office application and bit 3 designated (DG1UNEQUIP) is set in scratch word DG1STAT.

Format:

EQUIPCHK UTYN($1),MEMN($2),HGVAL($3)

Characteristics of Parameters:

UTYN — Specifies the unit type. The $1 is 1XLUSD29.

MEMN — Specifies the member number. The $2 is 0.

HGVAL — Specifies the hardware group value. The $3 is 1XLMCSD29.

Examples:

EQUIPCHK UTYN(1XLUSD29),MEMN(0),HGVAL(1XLMCSD29)

This statement will determine the state of unit type (1XLUSD29) and member number (0) and if it is either growth or operational a comparison of the hardware group value (1XLMCSD29) and the Compool symbol XL1MCDMTHG is made to determine if the application is for a 2-wire or 4-wire office.◀

EXECUTE

4.112 The description of the EXECUTE statement includes:

Function:

The EXECUTE statement causes the active central control to write specified 1A Processor language instructions into the instruction stack in the standby central control. Execution of these instructions takes place after the instruction execution register (IER) is cleared, the instruction fetch register (IFR) is initialized to octal 300036, and the delay and limited run register (DLR) is set to the number of cycles the standby central control is to run. The 1A Processor assembly language instructions and pseudo-operations are described in Section 254-280-020, Assembly Language—Description, 1A Processor.

Format:

EXECUTE $1($2($3)),CYCLE($4)

Characteristics of Parameters:

$1 — Specifies any of the five instruction stack registers:

(a) Buffer order word register left-half (BOL)

(b) Buffer order word register right-half (BOR)

(c) Half word register (HWR)

(d) Auxiliary buffer order register left-half (ABL)

(e) Auxiliary buffer order word register right-half (ABR).

$2 — Specifies the pseudo-operation for assembling the instruction. The $2 may be:

RLONG — RELLOAD on, FLONG on

RSHORT — RELLOAD on, FSHORT on

SHORT — FSHORT on

LONG — FLONG on.

$3 — Specifies the 1A Processor assembly language instruction.

CYCLE — Specifies the number of cycles the standby central control is to run. The $4 is a decimal number 1 through 15.

Example:

EXECUTE BOL(SHORT(LW X, 0)),CYCLE(1)

This statement causes the active central control to write the 1A Processor assembly language short instruction LW X, 0 into the BOL instruction stack register. Then the IER is cleared, the IFR is initialized to octal 300036, and the DLR in the standby central control is set to 1. The instruction is then executed.

INREAD

4.113 The description of the INREAD statement includes:

Function:

The INREAD statement causes the active central control to read an item, a group of items of the same address, or a word internal to the active central control. If EXPECT is specified, the reply, the mask, and expected results are passed to the DCON program for processing raw results. If NOSTORE is specified, only the read is performed.

Format:

```
        ITEM($1) NOSTORE
INREAD  ITEMS($2),EXPECT($4)
        WORD($3)
```

Characteristics of Parameters:

ITEM — Specifies the symbolic location of an item in the internal central control. The $1 is an item name.

ITEMS — Specifies the symbolic location of items in the internal central control. The $2 is a list of items all in the same word.

WORD — Specifies the symbolic location of a word in the internal central control. The $3 is the address.

EXPECT — Specifies the expected results. The $4 is any arithmetic expression that expresses the expected results.

NOSTORE — No raw data is stored.

Example:

```
INREAD WORD(IN1XR),EXPECT(-0)
```

This statement causes the active central control to read its X register (IN1XR) and compares it to all 1s (-0).

INWRITE

4.114 The description of the INWRITE statement includes:

Function:

The INWRITE statement causes the active central control to store a word of data internal to the active central control at the location specified. The data is stored with a store secure instruction into the active central control without product or insertion masking.

Format:

```
INWRITE WORD($1),DATA($2)
```

Characteristics of Parameters:

WORD — Specifies the symbolic word location in the internal central control. The $1 is the symbolic location.

DATA — Specifies the data to be stored. The $2 is any arithmetic expression that expresses the data to be written.

Example:

INWRITE WORD(IN1XR),DATA(-0)

This statement causes the active central control to write all 1s (-0) into its X register (IN1XR).

IOCONFIG

4.115 The description of the IOCONFIG statement includes:

Function:

The IOCONFIG statement configures the active central control to proper peripheral unit bus or buses for the input/output diagnostic.

Format:

IOCONFIG $\overline{\$1}$

Characteristics of Parameters:

$1 — The $1 is either PUBR or PUBA to indicate the PUBR or PUBA flip-flops in the central control buffer bus register (central control status flip-flops) (CSC). The PUBR and PUBA are initially set to zero. When PUBR or PUBA is specified, its associated flip-flop is set to one. The following PB bus configurations are used by input-output diagnostic:

PU ACCESS & WRITE BUS	PU REPLY BUS	PARAMETER
0	0	
1	1	PUBA
0+1	0+1	PUBR

Example:

IOCONFIG PUBR

This statement causes both peripheral unit buses to be configured as active.

IOCONIOUS

4.116 The description of the IOCONIOUS statement includes:

Function:

The IOCONIOUS statement calls a peripheral unit fault recovery subroutine to configure or restore one IOUS or the whole IOUS community to the correct bus configuration.

Format:

RESTORE
IOCONIOUS CONFIG(R0($1),S0($2),S1($3)),ALL

Characteristics of Parameters:

CONFIG — Specifies to what state the receive-on and send-on flip-flops should be set. The $1, $2, and $3 are 0 or 1.

RESTORE — Specifies that all in-service IOUSs should have their configuration restored according to status.

ALL — If specified, specifies that all in-service IOUSs should have their configuration set according to the CONFIG parameter.

Example:

IOCONIOUS CONFIG(R0(0),S0(1),S1(0))

This statement causes the receive-on flip-flop to be set to 0, the send-on 0 flip-flop to be set to 1, and the send-on 1 flip-flop to be set to 0.

IOMACON

4.117 The description of the IOMACON statement includes:

Function:

The IOMACON statement calls a peripheral unit fault recovery subroutine to set or clear the MA flip-flop for all equipped IOUCs in an IOUS.

Format:

IOMACON MA($1)

Characteristics of Parameters:

MA — Specifies that the MA flip-flop of the IOUS should be set or restored. The $1 is SET or RESTORE.

Example:

IOMACON MA(SET)

This statement causes all the MA flip-flops for all equipped IOUCs in the IOUS under diagnosis to be set.

IOPOLL

4.118 The description of the IOPOLL statement includes:

Function:

The IOPOLL statement performs an input/output polling pulse-source operation.

Format:

```
IOPOLL EXPECT($1)
       NOSTORE
```

Characteristics of Parameters:

NOSTORE — Specifies that the reply from the polling pulse be ignored.

EXPECT — Specifies the 1-bit expected result from the IOUS under diagnosis. The $1 is 0 or 1.

Example:

```
IOPOLL EXPECT(1)
```

This statement causes an input-output polling pulse source operation. The 1-bit result from the IOUS under diagnosis is compared to 1.

IOPUCON

4.119 The description of the IOPUCON statement includes:

Function:

The IOPUCON statement calls a peripheral unit fault recovery subroutine to set the state of the peripheral unit control bus clock or poll selector flip-flops.

Format:

```
IOPUCON POLL($1)
        CLOCK($1)
```

Characteristics of Parameters:

POLL — Specifies the state of the peripheral unit control bus poll selector flip-flop. The $1 is SET or CLEAR.

CLOCK — Specifies the state of the peripheral unit control bus clock selector flip-flop. The $1 is SET or CLEAR.

Example:

```
IOPUCON POLL(SET)
```

This statement causes the peripheral unit control bus poll selector flip-flop to be set.

IOPULSE

4.120 The description of the IOPULSE statement includes:

Function:

The IOPULSE statement performs a GCP peripheral unit pulse-source operation. It toggles the RO flip-flop, sets the IOUS maintenance flip-flop (MAS) and gates back on the peripheral unit reply bus the status register and error indicators from the IOUS under test.

Format:

IOPULSE NOSTORE

```
          ITEM($1)
IOPULSE ITEMS($2),EXPECT($3),REPULSE
```

Characteristics of Parameters:

NOSTORE — Specifies that the reply from the pulse source be ignored.

ITEM — Specifies the product mask for one Datapool-defined item. The $1 is a Datapool-defined item which is contained in the input/output pulse-source register.

ITEMS — Specifies the product mask of a number of Datapool-defined items. The $2 is a list of Datapool-defined items which are contained in the input-output pulse-source register.

EXPECT — Specifies the expected result of the read. It is exclusive-ORed with the result of the mask operation. The $3 is any arithmetic expression which expresses the expected result.

REPULSE — Specifies that another pulse-source operation be executed to toggle RO back to its original state.

Example:

IOPULSE ITEM(IO1PPFE),EXPECT(M(IO1PPFE)),REPULSE

This statement toggles the RO flip-flop, sets MAS, and gates back the status register and error indicators from the IOUS under test. The reply is compared to IO1PPFE (the expected result). Then another pulse-source operation is executed to toggle the RO back to its original state.

IOREAD

4.121 The description of the IOREAD statement includes:

Function:

The IOREAD statement specifies a peripheral unit operation which returns 24 bits of data on the peripheral unit reply bus from the input/output circuit for processing raw test results.

Format:

IOREAD OPER($1),$\overline{\text{DATA}}$($2),MASK($3),EXPECT($4),$\overline{\text{MTCPU}}$

Characteristics of Parameters:

OPER — Specifies the operation to be performed by the input-output circuit; it is sent over peripheral unit write bus bits 35 through 29. The $1 is a Datapool-defined operation code for the IO circuit.

DATA — Specifies the 24-bit data to be sent to the input-output circuit on peripheral unit write bus bits 23 through 0. If not specified, it is 0. The $2 is any arithmetic expression which expresses the data.

MASK — Specifies the product mask of the response on the peripheral unit reply bus before it is exclusive-ORed and stored into memory. The $3 is any arithmetic expression which expresses the mask.

EXPECT — Specifies the expected result of the read. It is exclusive-ORed with the result of the mask operation. The $4 is any arithmetic expression which expresses the expected result.

MTCPU — Specifies that the maintenance bit be sent. This is bit 12 of the peripheral unit enable address bus. If not specified, it is 0.

Example:

IOREAD OPER(CLEARBUF),DATA(45),MSK(34343),EXPECT(5655),MTCPU

This statement causes the Datapool-defined operation CLEARBUF to be sent out on bits 35 through 29 of the peripheral unit write bus, 45 to be sent out on bits 23 through 0 of the peripheral unit write bus, and 1 to be sent out on bit 12 of the peripheral unit enable address bus. The result is masked through 34343 and compared to 5655.

IOREQRD

4.122 The description of the IOREQRD statement includes:

Function:

The IOREQRD statement performs a maintenance read of the poll request register of the IOUS under diagnosis.

Format:

IOREQRD EXPECT(SR($1),MR($2))

Characteristics of Parameters:

EXPECT — Specifies the expected result of the read of the service request and maintenance request bits in the poll request register for the IOUC under diagnosis. The $1 and $2 are 0 or 1.

Example:

IOREQRD EXPECT(SR(0),MR(1))

This statement performs a maintenance read of the poll request register of the IOUS under diagnosis. The expected results are 0 for the service request bit and 1 for the maintenance request bit.

IOWRITE

4.123 The description of the IOWRITE statement includes:

Function:

The IOWRITE statement specifies a peripheral unit operation which sends data to the input/output circuit.

Format:

```
          OPER($1)          INVK($4)
IOWRITE OPAD($2),DATA($3),IOUCCODE($5),IPARE,IPARO,MTCPU,NSYNC
                          KCODE($6)
```

Characteristics of Parameters:

OPER — Specifies the operation to be performed by the input-output circuit; it is sent over peripheral unit write bus bits 35 through 29. The $1 is a Datapool-defined operation code for the input-output circuit. This parameter will not appear with OPAD.

OPAD — Specifies peripheral unit write bus bits 25 through 24. The $2 is any arithmetic expression which expresses the operation code and address. This parameter will not appear with OPER.

DATA — Specifies the 24-bit data to be sent to the input-output circuit on peripheral unit write bus bits 23 through 0. The $3 is any arithmetic expression which expresses the data.

IOUCCODE — If specified, it causes $5 to be insertion-masked into the three least significant bits of the input-output KCODE. It is 0 when not specified. The $5 is any number from 0 to 7.

INVK — If specified, the bit of the input-output KCODE corresponding to the absolute value of $4 is inverted. The $4 is any number from 3 to 11.

KCODE — If specified, it replaces the KCODE that would normally be used for the diagnosis. The $6 is any arithmetic expression which expresses the K-code.

IPARE — If specified, the central control inverts the parity bit it computes over the even bits before sending it out on peripheral unit write bus bit 39.

IPARO — If specified, the central control inverts the parity bit it computes over the odd bits before sending it out on peripheral unit write bus bit 38.

MTCPU — Specifies that the maintenance bit be sent. It is bit 12 of the peripheral unit enable address bus. If not specified, it is 0.

NSYNC — If specified, no address or data sync is to be transmitted to the input-output circuit.

Example:

```
IOWRITE OPER(CLEARBUF),DATA(77777),IPARE,NSYNC
```

This statement causes the Datapool-defined operation CLEARBUF to be sent out on bits 35 through 29 of the peripheral unit write bits; all 1s (77777) to be sent out on bits 23 through 0 of the peripheral unit write bus with inverted parity sent out over the even bits of the peripheral unit bus and no sync.

I2MAPTST

4.124 The description of the I2MAPTST statement includes:

Function:

The I2MAPTST statement is used for testing the ROM sequencer map for all normal mode operation codes with valid and invalid modes.

Format:

I2MAPTST

Characteristics of Parameters:

This statement has no parameters.

Example:

I2MAPTST

This statement tests the ROM sequencer map.

▶I2MEMR

4.125 The description of the I2MEMR statement includes:

Function:

The I2MEMR statement is used for reading consecutive micromemory locations beginning at the address specified. The data read must be a multiple of 3 bytes.

Format:

```
                    EXPPAT($2)
I2MEMR STADD($1),EXPECT($5),COUNT($3),MASK($7),MP($4)
                    DTBEGIN($6)
```

Characteristics of Parameters:

STADD — Specifies the start address in micromemory where the test code is loaded. The $1 is any valid micromemory address. The $1 must be specified with the EXPECT and EXPPAT parameters; it is optional with the DTBEGIN parameter.

EXPPAT — Specifies what is read from micromemory; $2 is any 8-bit arithmetic expression which expresses the pattern expected. The $2 is replicated three times to fill the 24-bit field. Count must be specified with the EXPPAT parameter.

COUNT — Specifies the number of times the EXPPAT is duplicated at consecutive memory locations. Three consecutive memory locations are read for each increment of count. The $3 specifies the number of locations read.

EXPECT — Specifies the data expected from micromemory reads. The $5 is a multiple number of octal words separated by commas.

DTBEGIN — Specifies the label of the block of data that was previously loaded in a diagnostic phase. The $6 is the label of the starting address where the block of data is contained.

MP — If specified, it causes bit 3 of the K-code to be modified according to $4. The $4 is zero, one, or NUT. If NUT is specified, bit 3 is inverted. If zero or one is specified, it is insertion-masked into bit 3 of the input-output K-code.

MASK — If specified, all reads for this macro are masked through $7. The $7 is any arithmetic expression which expresses the mask.

Example:

I2MEMR STADD(1003),EXPECT(0(77777777,00000000,77777777)),MP(NUT)

This statement causes the MP memory, which is not under test, to be read starting at address 1003. Each data byte is exclusive-ORed with the EXPECT parameters.

I2MEMW

4.126 The description of the I2MEMW statement includes:

Function:

The I2MEMW statement is used for writing consecutive micromemory locations beginning at the address specified. The data written must be a multiple of 3 bytes.

Format:

```
                    PATTERN($2)
I2MEMW STADD($1),DATA($5),COUNT($3),MP($4),
                    DTBEGIN($6)
```

Characteristics of Parameters:

STADD — Specifies the start address in micromemory where the test code strip should be loaded. The $1 is any valid micromemory address. The $1 must be specified with the data and pattern parameters. It is optional with the DTBEGIN parameter.

PATTERN — Specifies what is written on peripheral unit write bus bits 23 through 0. The $2 is any 8-bit arithmetic expression which expresses the data. The $2 is replicated three times to fill the 24-bit field. Count must be specified with the pattern parameter.

COUNT — Specifies the number of times the pattern is duplicated at consecutive memory locations. Three consecutive memory locations are written for each increment of count. The $3 specifies the number of locations written.

DATA — Specifies the data written over peripheral unit write bus bits 23 through 0. The $5 may be a multiple number of octal words separated by commas.

DTBEGIN — Specifies the label of the block of data that was previously loaded in a diagnostic phase.

MP — If specified, it causes bit 3 of the K-code to be modified according to $4. The $4 is zero, one, or NUT. If NUT is specified, bit 3 is inverted. If zero or one is specified, it is insertion-masked into bit 3 of the input-output K-code.

Example:

I2MEMW STADD(1003),DATA(0(77777777,00000000,00000000)),MP(NUT)

This statement causes the MP memory, which is not under test, to be loaded starting at address 1003 with the contents of the data parameter.

I2PCLOOP

4.127 The description of the I2PCLOOP statement includes:

Function:

The I2PCLOOP statement is used to put a TN82 peripheral controller into a loop data mode. Known data is transmitted by the TN82 peripheral controller, then looped back and tested for errors. The craftperson must establish the loop on the data link.

Format:

I2PCLOOP

Characteristics of Parameters:

This statement has no parameters.

Example:

I2PCLOOP

This statement puts a TN82 peripheral controller into a loop data mode.

I2MPTST

4.128 The description of the I2MPTST statement includes:

Function:

The I2MPTST statement is used to read and report the results of the MP power alarm monitor tests.

Format:

I2MPTST RESULT

Characteristics of Parameters:

RESULT — Specifies that the saved results of the MP power alarm monitor test should be reported to central control.

If no parameter is specified, then the state of the MP out-of-service bits are read and saved.

Example:

I2MPTST

This statement reads the states of both MP out-of-service bits and saves them in a call store location. The results of this read will be reported by the next call of I2MPTST.

I2PCPMP

4.129 The description of the I2PCPMP statement includes:

Function:

The I2PCPMP task routine uses a routine in the Peripheral Fault Recognition (PFLR) program to pump the IOP peripheral controller with its on-board diagnostic. The data that is pumped into the processor configuration resides on the 1A Processor disk file.

Format:

I2PCPMP

Characteristics of Parameters:

This statement has no parameters.

Example:

I2PCPMP

This statement loads the peripheral controller with its on-board diagnostic.

I2POLL

4.130 The description of the I2POLL statement includes:

Function:

The I2POLL statement performs an input-output polling GCP source operation.

Format:

```
        NOSTORE   CLEAR
I2POLL  EXPECT($1),SET
```

Characteristics of Parameters:

NOSTORE — Specifies that the reply from the polling pulse be ignored.

EXPECT — Specifies the 1-bit expected result from the MP under diagnosis is 0 or 1.

CLEAR — Specifies the IN1PUCBF flip-flop be cleared.

SET — Specifies the IN1PUCBF flip-flop be set.

Example:

I2POLL EXPECT(1)

This statement causes an input-output polling GCP source operation. The 1-bit result from the MP under diagnosis is compared with 1 and the results are stored as raw data.

I2PULSE

4.131 The description of the I2PULSE statement includes:

Function:

The I2PULSE statement performs either a maintenance or control GCP source operation.

Format:

```
        CONTROL ITEMS($2) NOSTORE
I2PULSE STATUS,ITEM($1),EXPECT($3),REPULSE
```

Characteristics of Parameters:

CONTROL — Specifies that the GCP to unlock the DMAC bus inhibit is executed. No other parameter should be used with control.

STATUS — Specifies that the maintenance GCP to toggle RO and return status be executed.

NOSTORE — Specifies that the reply from the maintenance GCP source be ignored.

ITEM — Specifies the product mask for one Datapool-defined item which is contained in the DMAC status register.

ITEMS — Specifies the product mask of a number of Datapool-defined items. The $2 is a list of Datapool-defined items which are contained in the DMAC status register.

EXPECT — Specifies the expected result of the read. It is exclusive-ORed with the result of the mask operation. The $3 is any arithmetic expression which expresses the expected result.

REPULSE — Specifies that another maintenance GCP source operation be executed to toggle RO back to its original state.

Example:

I2PULSE ITEM(IO1I2PMAS),EXPECT(M(IO1I2PMAS)),REPULSE

This statement toggles the RO flip-flop, sets MAS, clears S0 and S1, and gates back the status register and error indicators from the DMAC under test. The reply is product masked and exclusive-ORed with IO1I2PMAS. Another maintenance GCP source operation is executed to toggle the RO back to its original state.

I2RDADJ

4.132 The description of the I2RDADJ statement includes:

Function:

The I2RDADJ task routine is used to read the IOP peripheral controller service request or error request. The service request (SR) or error request (ER) from the processor configuration under test is right-adjusted according to the processor configuration member number in order to normalize the raw data.

Format:

```
          SR($1)
I2RDADJ   ER($1)
```

Characteristics of Parameters:

SR — Specifies the service request will be read. The $1 is the expected value (0 or 1).

ER — Specifies the error request will be read. The $1 is the expected value (0 or 1).

Example:

```
I2RDADJ SR(0)
```

This statement reads the IOP peripheral controller service request. The expected value is 0.

I2READ

4.133 The description of the I2READ statement includes:

Function:

The I2READ statement specifies a peripheral unit operation which returns 24 bits of data on the peripheral unit reply bus from the DMAC for processing raw test results.

Format:

```
                  MEMADM($7)MEMDA($8)
I2READ OPER($1),DATA(2),MASK($3),EXPECT($4), LIU($5),MP($6),MTCPU
                  MEMAD($9)
                  MPOP($10)MPDA($11)
```

Characteristics of Parameters:

OPER — Specifies the operation performed by the DMAC circuit; it is sent over peripheral unit write bus bits 34 through 29. The $1 is a Datapool-defined operation code for the DMAC.

DATA — Specifies the 24-bit data sent to the DMAC circuit on peripheral unit write bus bits 23 through 0. The $2 is any arithmetic expression which expresses the data.

▶MPOP — Specifies peripheral unit write bus bits 23 through 16, which is the MP operation code. The $10 is any arithmetic expression which expresses the MP operation code. It must not be specified with the DATA, MEMAD, or MEMADM codes. The MPDA parameter must be qualified.◀

▶MPDA — Specifies peripheral unit write bus bits 15 through 0, which is the 2 MP data bytes. The $11 may be any arithmetic expression which expresses the 2 bytes of data.◀

MEMAD — Specifies peripheral unit write bus bits 23 through 8, which expresses the MP memory address. The $9 may be any arithmetic expression which expresses the memory address. It must not be specified with the DATA, MPOP, or MEMADM parameters. The MEMDA parameter may be defaulted.

▶MEMADM — Specifies the peripheral unit write bus bits 23 through 8, which expresses the memory address. That portion of the address which signifies the LIU to be addressed is modified according to the LIU under test. The $7 is any arithmetic expression which expresses the memory address.◀

MEMDA — Specifies peripheral unit write bus bits 7 through 0, which expresses the data written to a memory address. It may be defaulted or specified with the MEMAD or MEMADM parameters. The $8 is any arithmetic expression which specifies the data.

LIU — If specified, it causes $5 to be insertion-masked into the three least-significant bits of the input-output K-code. The $5 is any number from zero through seven.

▶MP — If specified, it causes bit 3 of the K-code to be modified according to $6. The $6 is zero, one, or NUT. If NUT is specified, bit 3 is inverted. If zero or one is specified, it is insertion-masked into bit 3 of the input-output K-code.◀

MTCPU — Specifies that the maintenance bit is sent. It is bit 12 of the peripheral unit enable address bus.

Example:

I2READ OPER(LOOPDA),DATA(1DG_ONES),MASK(1DG_ONES), EXPECT (DG_ONES),MTCPU

This statement sends the Datapool-defined operation LOOPDA out on peripheral unit write bus bits 34 through 29, all ones sent on bits 23 through 0. The maintenance bit (which is bit 12 of the peripheral unit enable address bus) is sent. The reply from peripheral unit reply bus bits 23 through 0 is masked with all ones and exclusive-ORed with all ones to give a pass or fail test result.

I2TESTMP

4.134 The description of the I2TESTMP statement includes:

Function:

The I2TESTMP statement is used to load and execute the onboard MP diagnostic and to report the test results to central control.

Format:

```
            PSTORE
I2TESTMP    DSTORE
```

Characteristics of Parameters:

PSTORE — Specifies that the program store random access memory area of the MP should be loaded.

DSTORE — Specifies that the data store random access memory area of the MP should be loaded.

If no parameter is specified, then test results are reported to central control.

Example:

I2TESTMP

Since no parameter is specified, the test results from the previous call of I2TESTMP are read from either program or data store and reported to central control.

I2WRITE

4.135 The description of the I2WRITE statement includes:

Function:

The I2WRITE statement specifies a peripheral unit operation which sends data to the DMAC.

Format:

```
          OPER($1) DATA($3)                    INVK($4)
I2WRITE OPAD($2),MEMADM($8),MEMDA($9),LIU($5),IPARE,IPARO,MTCPU,SCHSPD
                  MPOP($10)   MPDA($11)  KCODE($6)
                  MEMAD($12) MEMDA($9)  MP($7)
```

Characteristics of Parameters:

OPER — Specifies the operation performed by the DMAC circuit; it is sent over the peripheral unit write bus bits 34 through 29. The $1 is a Datapool-defined operation code for the DMAC circuit. This parameter must not appear with OPAD.

OPAD — Specifies peripheral unit write bus bits 35 through 24. The $2 is any arithmetic expression. This parameter must not appear with OPER.

DATA — Specifies the 24-bit data sent to the input-output circuit on peripheral unit write bus bits 23 through 0. The $3 is any arithmetic expression which expresses data.

MPOP — Specifies peripheral unit write bus bits 23 through 16, which specifies the MP operation code. The $10 is any arithmetic expression which expresses the MP operation code. This parameter must not be specified with the DATA, MEMAD, or MEMADM parameters. The MPDA parameter must be specified.

▶MPDA — Specifies peripheral unit write bus bits 15 through 0, which specify the 2 MP data bytes. The $11 is any arithmetic expression which expresses the 2 bytes of data.◀

MEMAD — Specifies peripheral unit write bus bits 23 through 8, which express the MP memory address. The $12 is any arithmetic expression which expresses the memory address. This parameter must not be specified with the DATA, MPOP, or MEMADM parameters. The MEMDA parameter may be defaulted.

▶MEMADM — Specifies the peripheral unit write bus bits 23 through 8, which express the memory address. That portion of the address which signifies the LIU addressed is modified according to the LIU under test. The $8 is any arithmetic expression which expresses the MP memory address. The MEMDA parameter may be defaulted.◀

MEMDA — Specifies peripheral unit write bus bits 7 through 0, which express the data written to a memory address. It may be defaulted or specified with the MEMAD or MEMADM parameters. The $9 is any expression which specifies the data.

LIU — If specified, it causes $5 to be insertion-masked into the least three significant bits of the input-output K-code. The $5 is any number from 0 through 7. This parameter must not be specified with the KCODE or INVK parameters.

INVK — If specified, the bit of the input-output K-code corresponding to the absolute value of $4 is inverted. The $4 is any number from 4 through 11 or 24 through 26. This parameter must not be specified with the LIU, KCODE, or MP parameters.

KCODE — If specified, it replaces the K-code that would normally be used with the diagnostic. The $6 is any arithmetic expression which expresses the K-code. This parameter must not be specified with LIU, INVK, or MP parameters.

▶MP — If specified, it causes bit 3 of the K-code to be modified according to $7, The $7 is zero, one, or NUT (not under test). If NUT is specified, bit 3 is inverted. If zero or one is specified, it is insertion-masked into bit 3 of the input-output K-code.◀

IPARE — If specified, the central control inverts the parity bit it computes over the even bits of the peripheral unit write and enable address bus before sending it out on peripheral unit write bus bit 39.

IPARO — If specified, the central control inverts the parity bit it computes over the odd bits of the peripheral unit write and enable address bus before sending it out on peripheral unit write bus bit 38.

MTCPU — Specifies that the maintenance bit is sent. It is bit 12 of the peripheral unit enable address bus.

NSYNC — Specifies that the peripheral unit sync pulse is not sent. It is bit 13 of the peripheral unit enable address bus.

▶SCHSPD — If specified, send channel speed to peripheral controller.◀

Example:

I2WRITE OPER(LOOPDA),DATA(O(7777777)),IPARE,NSYNC

This statement sends the Datapool-defined operation LOOPDA out on bits 34 through 29 of the peripheral unit write bus; all ones are out on peripheral unit data bits 23 through 0 with the even parity bit inverted with no peripheral unit sync.

▶LCKCODE

4.136 The description of the LCKCODE statement includes:

Function:

The LCKCODE statement writes the K-code register of an auxiliary unit to the value contained in DG1AUKCODE. This is done by first control pulsing (GCP) the test unit to obtain the present state of the auxiliary unit K-code register. Then, using the K-code obtained from the GCP reply to address the test unit, a write to the auxiliary unit K-code register is performed.

Format:

LCKCODE

Characteristics of Parameters:

This statement has no parameters.

Example:

LCKCODE

This statement will GCP the test unit (using the GCP address contained in DG1GCPADDR) to obtain the present state of the test unit's K-code register. Using the K-code obtained from the GCP reply to address the test unit, a write of the test unit's K-code register is performed to change its state to the value contained in DG1AUKCODE.

LDSAR

4.137 The description of the LDSAR statement includes:

Function:

The LDSAR statement will take the specified argument (ARGX), add the contents of the F register to it, and store the results in the output address register (OAR) of the attached processor interface (API). The LDSAR will also set the read or write bit in the OAR.

Format:

```
                READBIT
LDSAR ARGX($1),WRITEBIT
```

Characteristics of Parameters:

ARGX — Specifies that the argument is to be added to the contents of the F register. The $1 is any arithmetic expression which expresses an address pattern.

READBIT — Specifies that bit 22 of the OAR is to be set. (***Note:*** Bit 23, the write bit, is defaulted to 0 if this parameter is specified.)

WRITEBIT — Specifies that bit 23 of the OAR is to be set. (***Note:*** Bit 22, the read bit, is defaulted to 0 if this parameter is specified.)

Example:

LDSAR ARGX(O(4271)),READBIT

This statement will load the API's OAR with F+O(4271). It will also set the read bit (bit 22).◀

MBREGTST

4.138 The description of the MBREGTST statement includes:

Function:

The MBREGTST statement causes the active central control to do a series of write/read operations into the standby central control register specified. The following procedure is used:

(1) The register is cleared and then the read/write lists in the register (specified by 1DG_RW_ register name) are read and expected to be 0.

(2) An alternating 1 and 0 pattern which is masked with the read/write bits.

(3) The ST1UBA (unmasked bus read through the add one register), ST1M (masked bus), and ST1UBM (unmasked bus read through the mask and complement circuit) are read and expected to be 0.

(4) An alternating 0 and 1 pattern is written into and read out of the read/write bits in the register.

(5) The ST1UBA, ST1MB, and ST1UBM are read and expected to be 0.

(6) The register is cleared and the read/write bits are read and expected to be 0.

Format:

MBREGTST $1

Characteristics of Parameters:

$1 — Specifies a register name in the standby central control without the prefix ST1.

Example:

MBREGTST XR

This statement causes the active central control to do a series of read/write operations into the X register (XR) in the standby central control.

MCCABLEV

4.139 The description of the MCCABLEV statement includes:

Function:

The MCCABLEV routine is used to generate the GCP to produce the A or B level for testing the manual recovery circuitry. The associated task routine does all the necessary administration to keep the system sane and recover smoothly from the A- or B-level interrupt.

Format:

MCCABLEV GCP($1),COMM($2),RO($3),CONFIG($4)

Characteristics of Parameters:

GCP — Specifies the number of GCPs to be produced by the task routine. Also will specify whether an automatic processor configuration is to be generated. The $1 is one of the following.

NONE — Specifies no GCP.

ONE — Specifies 1 GCP. This is the default case if no GCP parameter is specified.

TWO — Specifies 2 GCPs.

OVREN — Specifies that 1 GCP should be produced if the override enable is on.

AUTOPC — Specifies that an automatic processor configuration will be produced. No GCP is done.

COMM — Specifies the community (program store or call store) which the routine will be executed from. The $2 is only specified as call store. The program store community is the default case.

RO — Specifies the bus which the stores will receive on. The $3 is SET for bus 1. The default case is RESET.

CONFIG — Specifies whether a configuration of the standby central control is to be performed. The $4 is STOPCC if the standby central control is to be stopped and left out of service.

Example:

MCCABLEV COMM(CS),RO(SET),GCP(TWO)

This statement will cause the code to be copied to the call store for execution. The store's active bus will be set to bus 1. Two GCPs will be executed.

MCCBARTST

4.140 The description of the MCCBARTST statement includes:

Function:

The MCCBARTST statement tests the system activity bar graph display for the application panel on the master control console. The MCCBARTST routine has the capability of either sequentially lighting a bar specified to the maximum value or extinguishing the light-emitting diodes (LEDs) of the bar specified.

Format:

MCCBARTST $1,$2

Characteristics of Parameters:

$1 — Represents 1 of 15 bars that can be specified.

$2 — Specifies whether the bar under test is to be sequentially lighted or extinguished. The $2 is ON or OFF.

Examples:

(a) MCCBARTST BAR1,ON

This statement tests bar 1 of the status panel. The lights are sequentially lighted.

(b) MCCBARTST BAR15,OFF

This statement tests bar 15 of the status panel. The lights are sequentially extinguished.

MCCBITOG

4.141 The description of the MCCBITOG statement includes:

Function:

The MCCBITOG statement writes a toggle row on the master control console to the desired pattern. This is done by reading the toggle row, comparing the contents of the row to the desired pattern, then toggling the flip-flops that mismatch. The MCCBITOG statement enables testing of rows with toggle flip-flops by walking either a one through a field of zeros, or a zero through a field of ones.

Format:

MCCBITOG ROW($1),WALK($2),BIT($3),MASK($4)

Characteristics of Parameters:

ROW — Specifies the row address. The $1 is a 7-bit pattern sent out over peripheral unit write bus bits 30 through 24.

WALK — Indicates the type of bit pattern sent on peripheral unit write bus bits 23 through 0. The $2 is either 1T0 (representing a 1 in a field of 0s) or 0T1 (representing a 0 in a field of 1s).

BIT — Specifies which data bit is complemented from the rest. The $3 is a decimal number.

MASK — Specifies 24-bit pattern which specifies the significant bits of the reply on the peripheral unit reply bus. There are four cases of masks that are encoded into the data word depending on which row is under test. The $4 is any arithmetic expression which expresses the mask.

Example:

MCCBITOG ROW (1MCROW29),WALK(1T0),BIT(18),MASK(0(17777777))

This statement reads the row at row address 1MCROW29 and compares the result to a word of data, containing a 1 in bit position 18, all other bits 0. The row read is masked through octal 17777777. The flip-flops which mismatch after the compare are toggled.

MCCBITWK

4.142 The description of the MCCBITWK statement includes:

Function:

The MCCBITWK statement matches a reply to an expected result through a mask to test a matrix row in the master control console with set-reset type flip-flops. One data word is obtained by encoding the bit displacement and pattern (1 in field of 0s; 0 in a field of 1s).

Format:

MCCBITWK ROW($1),WALK($2),BIT($3),MASK($4)

Characteristics of Parameters:

ROW — Specifies the row address. The $1 is a 7-bit pattern sent out over the peripheral unit write bus bits 30 through 24.

WALK — Indicates the type of bit pattern sent on peripheral unit bus bits 23 through 0. The $2 is either 1T0 (representing a 1 in a field of 0s) or 0T1 (representing a 0 in a field of 1s).

BIT — Specifies which data bit is complemented from the rest. The $3 is a decimal number.

MASK — Specifies a 24-bit pattern which specifies the significant bits of the reply on the peripheral unit reply bus. There are four cases of masks that are encoded into the data word depending on which row is under test. The $4 is any arithmetic expression which expresses the mask.

Example:

MCCBITWK ROW(1MCROW3),WALK(1T0),BIT(3),MASK(1MCALL1S)

This statement sends one word of data, containing a 1 in bit position 3, all other bits 0, to the master control console row at row address 1MCROW3. The reply on the peripheral unit reply bus is masked through the value of 1MCALL1S.

MCCINTCON

4.143 The description of the MCCINTCON statement includes:

Function:

The MCCINTCON statement sets up the bus configuration flip-flop in the master control console.

Format:

MCCINTCON RECEIVEON($1),MAINTMODE($2),REPLYON($3)

Characteristics of Parameters:

RECEIVEON — Specifies on which peripheral unit bus (0, 1, or active) the master control console will receive information from the central control. The $1 is BUS 0, BUS 1, or ABUS.

MAINTMODE — Specifies whether the master control console is in maintenance mode. The $2 is OFF or ON. (OFF is normal condition). Does not need to be specified if $1 is ABUS; will default to MAINTMODE ON.

REPLYON — Specifies on which peripheral unit bus the master control console is addressing the central control. If not specified, the master control console sends both. The $3 is BUS 0 or BUS 1. Does not need to be specified if $1 is ABUS.

Example:

MCCINTCON RECEIVEON(BUS 0),MAINTMODE(OFF),REPLYON(BUS 1)

This statement sets the bus configuration flip-flop in the master control console. The master control console will receive on BUS 0, the maintenance mode is off, and the master control console will reply on BUS 1.

MCCKEYSET

4.144 The description of the MCCKEYSET statement includes:

Function:

The MCCKEYSET macro is used to provide the operating personnel with a way of selecting various keys needed for a particular test. It also can be used to check the status of a key by input from the operating personnel. The associated task routine flashes the diagnostic in progress (DIP) lamp and displays the test number being performed on the numeric display of the master control console. The operating personnel then has approximately 1 minute to select the keys specified in the diagnostic listing. When the operating personnel has selected the keys, the direct data insert key 1 is pressed. The routine then reads the keys to verify the correct selection. If the keys are incorrect, the DIP lamp will continue to blink until 1 minute has elapsed or until the correct keys are selected. If 1 minute elapses, a failure will be recorded and the diagnostic will proceed to the next test.

The MCCKEYSET routine also has the capability of skipping tests without recording a failure. If the DIP lamp is flashing and direct data insert key 0 is pressed, the test will be skipped with no failures. This is useful when trying to loop on a particular test. To get to the test wanted, other tests may have to be run which are not cared about. By using key 0, the test can be gotten to much quicker.

The other use of MCCKEYSET is to provide the operating personnel a way of telling the diagnostic the state of a lamp which cannot be read directly. The diagnostic will tell the operating personnel which condition to look for. If the condition is as described, direct data insert key 1 should be pressed. If the condition is incorrect, the operating personnel should respond by pressing direct data insert key 0.

Format:

```
            KEYCHECK($3,$4)
MCCKEYSET KEYS($1),LABEL($2)
```

Characteristics of Parameters:

KEYS — Specifies the keys which the operating personnel should set and reset. These keys will be checked for correctness when the operating personnel responds with direct data insert key 1. The $1 is a list of keys to set from the following list:

UPDATE_FS0	PR_INH_INT
UPDATE_FS1	PR_RST_MTCE_IO
OC_BLK_PS0	PR_MOD_REC_ACT
OC_VAR_PS0	PR_FULL_CFG_EMER_MODE
OC_VAR_PS1	PR_MIN_CFG_EMER_MODE
OC_AUB0	PR_PH6
OC_AUB1	PR_PH5
OC_PSB0	PR_PH4
OC_PSB1	PR_PH3
OC_ACT_CC	PR_PH2
OC_STB_CC	PR_PH1
SR_ENABLE	MC3_DIS
PC_DISABLE	RESET_DIS
PR_CLR_UTIL_FUNC	

LABEL — Specifies the label which will be jumped to if the routine times out (exceeds 1 minute) or if direct data insert key 0 is pressed. The $2 is the label name. A corresponding DTDEST macro must be used at the label destination.

KEYCHECK — Specifies that the test is a check of a key state by the operating personnel. The $3 is any English expression which describes the key which is to be checked by the operating personnel. The $4 is either SET or RESET.

Examples:

(1) MCCKEYSET KEYS(OC_VAR_PS0,OC_AUB1,OC_PSB1,OC_ACT_CC),
LABEL(NEXT_TEST)

(2) MCCKEYSET KEYCHECK(PC_BIT_00,SET)

Example 1 will tell the operating personnel to set variable PS0, auxiliary unit bus 1, program store bus 1, and the active central control keys in the override control section of the master control console. Once the keys are set, the operating personnel should respond by pressing direct data insert key 1. The selected keys will then be read to verify that the correct keys were selected.

Example 2 will tell the operating personnel that processor configuration state counter bit 0 should be checked to see if it is set. If it is set, direct data insert key 1 should be pressed. If not set, direct data insert key 0 should be pressed.

MCCKEYTEST

4.145 The description of the MCCKEYTEST statement includes:

Function:

The MCCKEYTEST statement enables a check of the specified key on the master control console or central control with an expected result. If the flip-flop associated with the specified key does not match the expected value within 20 seconds, a failure is stored at this test.

Format:

MCCKEYTEST KEY($1),EXPECT($2),PASSGOTO($3)

Characteristics of Parameters:

KEY — Specifies an addressable key name defined on Datapool. The address of the key can be in either the central control or the master control console. The $1 is the address of the key.

EXPECT — Specifies the expected value of the key under test. The $2 is 0 or 1.

PASTGOTO — Specifies a data table location where the program will continue execution.

Example:

MCCKEYTEST KEY(MC1DIKO),EXPECT(1),PASSGOTO(DILOC1)

This statement causes the key at address MC1DIKO to be compared to 1 (the expected value), and if the comparison is true, the check will continue at data table with label DILOC1.

MCCONFIG

4.146 The description of the MCCONFIG statement includes:

Function:

The MCCONFIG statement sets the states of the peripheral unit bus configuration flip-flops of the central control buffer register CSC (central control status flip-flops).

Format:

MCCONFIG PUBA,PUBT,PUBR,PUBO,SCBC,SCBB,SCBA,PBMB,PBMA,CDMA,CDMB,CPDB,E14

Characteristics of Parameters:

PUBA — Coded enable peripheral unit bus control. When set, peripheral unit bus 1 active; otherwise, peripheral unit bus 0 is active.

PUBT — Coded enable peripheral unit bus control. Denotes that the standby bus is in trouble.

PUBR — Coded enable peripheral unit bus control. Active central control receives on both peripheral unit buses.

PUBO — Coded enable peripheral unit bus control. Active central control sends on both peripheral unit buses.

SCBC — Peripheral unit reply bus control flip-flop C.

SCBB — Peripheral unit reply bus control flip-flop B.

SCBA — Peripheral unit reply bus control flip-flop A.

PBMB — Peripheral unit write bus control. Controls standby send bus.

PBMA — Peripheral unit write bus control. Controls active send bus.

CDMA — Peripheral unit enable bus control. Controls active send bus.

CDMB — Peripheral unit enable bus control. Controls standby send bus.

CPDB — Peripheral unit enable and CPD reply bus control flip-flop.

E14 — Sends out a one on bit 14 of E register in the central control when specified.

Example:

MCCONFIG PUBA,PBMA,SCBA,PUBT

This statement sets the coded enable bus 1 active, marks the standby coded enable bus as in trouble, sets the standby send bus, and sets the peripheral unit reply bus control flip-flop A.

MCCPULSE

4.147 The description of the MCCPULSE statement includes:

Function:

The MCCPULSE statement is used for pulse-source reads of the status register in the master control console. When a pulse source is activated, it automatically gates out the contents of the status register on both peripheral unit reply buses.

Format:

NOSTORE _____
MCCPULSE POINT($1),MASK($2),EXPECT($3),REPULSE

Characteristics of Parameters:

POINT — Specifies 24-bit address of the pulse point. The $1 is the address.

NOSTORE — Either this parameter or the MASK, EXPECT pair will appear. This indicates that the reply from the pulse source should be ignored.

MASK — Specifies 2-bit mask pattern. The $2 is any arithmetic expression which expresses the mask.

EXPECT — Specifies 24-bit expected results. The $3 is any arithmetic expression which expresses the expected results.

REPULSE — If specified, the pulse point is hit twice. The first pulse toggles the RO (peripheral unit write bus select) flip-flop, and the second pulse source toggles the RO flip-flop back to its initial state.

Example:

MCCPULSE POINT(OPPU000),MASK(1MCALL1S),EXPECT(M(MC1ROFF,MC1MAFF))

This statement causes a read of the pulse source at address OPPU000 in the status register in the master control console. After being masked through the value of 1MCALL1S, the result is compared to the expected result (the mask of the MC1ROFF and MC1MAFF items).

MCCREAD

4.148 The description of the MCCREAD statement includes:

Function:

The MCCREAD statement executes a master control console instruction and matches the reply to an expected result through a mask.

Format:

MCCREAD OPBITS($1),ROW($2),DATA($3),MASK($4), EXPECT($5),SPARE,IPARO,IPARE
,MTCPU,NSYNC

Characteristics of Parameters:

OPBITS — Specifies the four OP code bits on the peripheral unit write bus. The $1 is any combination in any of CM, S, R, T, or NOOP.

CM = PU write bus bit 35 (PUW35)=1

S = PUW33=1

R = PUW32=1

T = PUW31=1

NOOP = PUW35,PUW33,PUW32,PUW31=0.

ROW — Specifies an invalid or valid row address. The $2 is a 7-bit pattern sent out over bus bits PUW30 through 24.

DATA — Specifies the 24-bit pattern on PUW23 through PUW00. If not specified, all 0s are sent. The $3 is any arithmetic expression which expresses the data.

MASK — Specifies a 24-bit pattern which specifies the significant bits of the reply on the peripheral unit reply bus. The $4 is any arithmetic expression which expresses the mask.

EXPECT — Specifies a 24-bit pattern which specifies the expected result of the masked reply. The $5 is any arithmetic expression which expresses the expected result.

SPARE — If not specified, PUW34 = 0. If specified, PUW34 = 1.

IPARO — If specified, the central control inverts the parity bit it computes over the odd bits before sending it out on PUW38.

IPARE — If specified, the central control inverts the parity bit it computes over the even bits before sending it out on PUW39.

MTCPU — If specified, the M-bit (enable address bus bit 12) = 0, else =1.

NSYNC — If specified, the sync is not sent.

Example:

```
MCCREAD OPBITS(CM),ROW(1MCSTATR),MASK(M(MC1AZCO,MC1KM,MC1ROFF,
ME          MC1MAFF)),EXPECT(M(MC1KM,MC1ROFF))
```

This statement sets PUW35 = 1, from OPBITS(CM), sends the row address 1MCSTATR over bus bits PUW30 through 24, makes the reply with the mask of the (MC1AZCO,MC1KM,MC1ROF,MC1MAFF) items and compares the result with the expected result (the mask of the MC1KM and MC1ROFF items).

MCCTOG

4.149 The description of the MCCTOG statement includes:

Function:

The MCCTOG statement writes a toggle row on the master control console to the 24-bit pattern specified by the PATTERN parameter. The specified row is read and compared to the desired pattern; then the bits that mismatch are toggled. The results of the toggle operation may either be matched or ignored.

Format:

```
                                NOSTORE
MCCTOG ROW($1),PATTERN($2),MASK($3),EXPECT($4)
```

Characteristics of Parameters:

ROW — Specifies 7-bit row address. The $1 is the address.

PATTERN — Specifies 24-bit pattern that is written into the toggle row. A read is required of the row and compares the contents of the row to the desired pattern and then toggles the bits that mismatch. The EXPECT and MASK applies to the read after the toggle operation. The $2 is any arithmetic expression which expresses the data.

NOSTORE — Either this parameter or the MASK-EXPECT pair will be present. This indicates that the reply from the master control console is ignored.

MASK — Specifies 24-bit mask. The $3 is any arithmetic expression which expresses the mask.

EXPECT — Specifies expected results. The $4 is any arithmetic expression which expresses the expected result.

Example:

```
MCCTOG ROW(1MCROW24),PATTERN(M(MC1FAK)),MASK(1MCALLIS),EXPECT(M(MC1FAK))
```

This statement reads master control console row 1MCRW24, compares it to the mask of the MC1FAK item, then toggles the bits that do not match. The result is masked through the value of 1MCALLIS and compared to the mask of the MC1FAK item.

MCCTUCSR

4.150 The description of the MCCTUSR statement includes:

Function:

The MCCTUCSR macro is used to set the TUC, specified as a helper unit on the DGN request, to the SR mode. This will cause the READY lamp on the master control console panel to be lighted.

Format:

MCCTUCSR

Characteristics of Parameters:

There are no parameters for this macro. The TUC to be set to the SR mode is obtained from the diagnostic buffer table (DBT). The DBT gets set up from the diagnostic input message.

Example:

MCCTUCSR

This statement will set the TUC specified on the DGN message to the SR mode and will cause the READY lamp on the master control console to be lighted.

MCCWRITE

4.151 The description of the MCCWRITE statement includes:

The MCCWRITE statement executes the master control console instruction with specified values on the peripheral unit write bus and the enable address bus. Where a valid master control console instruction is sent out on the bus, the master control console reply is on the peripheral unit reply bus. The statement always ignores this response.

Format:

MCCWRITE OPBITS($1),ROW($2),DATA($3),KCODE($4),SPARE,IPARO,IPARE,MTCPU,NSYNC

Characteristics of Parameters:

OPBITS — Specifies the four OP-code bits on the peripheral unit write bus. The $1 can be any combination in any order of CM, S, R, T, or NOOP.

CM = Peripheral unit write bus bit 35 (PUW35)=1 (control mode)

S = PUW33 = 1 (set on 1)

R = PUW32 = 1 (reset on 0)

T = PUW31 = 1 (toggle on 1)

NOOP = PUW35 = PUW33 = PUW32 = PUW31 = 0.

ROW — Specifies an invalid or valid row address to be sent on peripheral unit write bits PUW30 through PUW24. The $2 is the row address.

DATA — Specifies the 24 data bits to be sent PUW23 through PUW0. The $3 is any arithmetic expression which expresses the data.

KCODE — If not specified, the true master control console KCODE is sent on the enable address bus bits (ENA) 11 through 0. An invalid K-code can be specified. The $4 is a 12-bit pattern.

SPARE — If not specified, PUW34 = 0, else PUW34 = 1.

IPARO — If specified, the central control inverts the parity bit it computes over the odd bits before sending it out on PUW38.

IPARE — If specified, the central control inverts the parity bit it computes over the even bits before sending it out on PUW39.

MTCPU — If specified, the M-bit (ENA 12) equals 0; if not specified, the M-bit equals 1.

NSYNC — If specified, the sync is not to be sent.

Example:

MCCWRITE OPBITS(S,R),ROW(1MCROW7),DATA(1DG_ALTO),IPARE,NSYNC

This statement sets PUW33 = 1 and PUW32 = 1 from OPBITS(S,R), sends the row address 1MCROW7 over bits PUW33 through 24, sends the value of 1DG_ALTO over PUW23 through PUW0, with inverted parity (even) on PUW39, and no sync is sent.

MCSDPTC

4.152 The description of the MCSDPTC statement includes:

Function:

The MCSDPTC statement clears the signal distributor (SD) point of the SD matrix in three steps so as not to start a power monitor alarm test on the unit whose SD points are in the specified row. The first step is to set all points to a one (extinguish "ACK" and "out-of-service" lamps). Next, clear all the even-numbered points in the row specified. After a 1-ms delay, the remaining odd-numbered points of the row specified are cleared.

Format:

MCSDPTC SSD($1)

Characteristics of Parameters:

SSD — Specifies the row address in the SD matrix that is being tested. The $1 is the row address.

Example:

MCSDPTC SSD(ROW39)

This statement sets all points in ROW39 of the SD matrix to a 1, then clears all even-numbered points. After a 1-ms delay, the odd-numbered points are cleared.

MEMCHECK

4.153 The description of the MEMCHECK statement includes:

Function:

The MEMCHECK statement reads a block of scratch memory and compares each word with a specified data constant. The two parity bits of each word are also checked.

Format:

MEMCHECK ADDR($1),BLOCKSIZE($2),DATA1($3),DATA2($4), . .DATA32($34)

Characteristics of Parameters:

ADDR — Specifies the Datapool name given to the start of scratch memory for auxiliary unit autonomous block transfer. The $1 is the Datapool name.

BLOCKSIZE — Specifies the size of the block to be read. The $2 is any number between 1 and 1024.

DATA1—DATA32 — Specifies a set of 24-bit data patterns to be read from scratch memory. The $3 through $34 are any arithmetic expressions which express the data.

Example:

MEMCHECK ADDR (DG1ADSBLK),BLOCKSIZE(90),DATA1(O(25252525)),DATA2(O(52525252))

This statement reads 90 blocks of memory at address DG1ADSBLK and compares them in turn to octal 25252525 and octal 52525252.

MEMLOAD

4.154 The description of the MEMLOAD statement includes:

Function:

The MEMLOAD statement writes a block of scratch memory with some constant data pattern.

Format:

MEMLOAD ADDR($1),BLOCKSIZE($2),OPTIONS($3),DATA1($4),DATA2($5),. .DATA32($35)

Characteristics of Parameters:

ADDR — Specifies the Datapool name given to the start of scratch memory for auxiliary unit autonomous block transfers. The $1 is the Datapool name.

BLOCKSIZE — Specifies the size of the block to be written. The $2 is any number between 1 and 1024.

OPTIONS — Specifies the mode, parity, and store timing during the write (MS) instruction. The MS instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor. The $3 is a list of options. Table A is a list of options for the MS and ML instructions. This permits inverting parity. If not specified, $3 is W.

DATA1—DATA32 — Specifies a set of 24-bit data patterns to be written into scratch memory. The $4 through $35 are any arithmetic expressions which express the data.

Example:

MEMLOAD ADDR (DG1ADSBLK),BLOCKSIZE(90),DATA1(O(77777777)),DATA2(O(00000000))

This statement writes 90 words of memory starting at address DG1ADSBLK alternately with octal 77777777 (all 1s) and octal 00000000.

MPRDXRUN

4.155 The description of the MPRDXRUN statement includes:

Function:

The MPRDXRUN statement acts upon data generated during one or more previous tests. The MPRDXRUN statement follows an MP, MP7, MP8, RD, or RD6 D1-1 statement which begins a series of tests. It returns the test results to the DCON program.

Format:

MPRDXRUN

Characteristics of Parameters:

This statement has no parameters.

Example:

MPRDXRUN

This statement acts upon data collected from the previous MP, MP7, MP8, RD or RD6 statement.

MP, MP7, and MP8

4.156 The description of the MP, MP7, and MP8 statements includes:

Function:

The MP, MP7, or MP8 statement looks at 1, 7, or 8 match points. One of these statements begins a series of tests to be ended by the MPRDXRUN statement. It stores in call store the mask, the matcher address ST1M1R and ST1M0R, the match phase (A, B, or C) and the match word. It also stores the size and displacement of the mask, the number of cycles the central control is to run, and the expected results of the match.

Format:

MP
MP7 $1,CYCLE($2),EXPECT($3),MPHASE($)
MP8

Characteristics of Parameters:

$1 — Specifies the match point(s). The $1 is the symbolic name of the match point(s).

CYCLE — Specifies the number of cycles the standby central control is to run. The $2 is a decimal number.

EXPECT — Specifies the expected result of the match. For the MP statement, $3 is 1 bit with a value of 0 or 1. For the MP7 and MP8 statements, $3 is any arithmetic expression which expresses the expected result.

MPHASE — Specifies match phase. The $4 is A, B, or C.

Example:

MP DRMBO,CYCLE(1),EXPECT(1)

This statement stores in call store the mask, the matcher address, and the match word (all available from the match point name DRMBO). It also stores the number of cycles the standby central control is to run at the end of the test (1), the match phase (A), and the expected results of the test (1).

▶MPXHEAD

4.157 The description of the MPXHEAD statement includes:

Function:

The MPXHEAD statement saves in call store the mask, address, and number of cycles that the standby central control is to run for the MPXRUN routine which follows immediately after the MPXHEAD routine.

Format:

MPXHEAD

Characteristics of Parameters:

This statement has no parameters.

Example:

MPXHEAD

This statement acts upon data collected from the previous MP, MP7, or MP8 statement.◀

PCCWRITE

4.158 The description of the PCCWRITE statement includes:

Function:

The PCCWRITE routine is used to write the processor configuration state counter to some value and maintain any other bits in the processor configuration register (PCR).

Format:

PCCWRITE DATA($1)

Characteristics of Parameters:

DATA — Specifies the data to be written into the PCR register. The $1 is any valid arithmetic expression which represents the data.

Example:

PCCWRITE DATA(M(IN1PCC3))

This macro call will cause the processor configuration state counter to be set to the value of M(IN1PCC3). All other bits of the PCR will remain unchanged.

PDQWRITE

4.159 The description of the PDQWRITE statement includes:

Function:

The PDQWRITE statement performs two writes into a DUC and can also follow these with a no-store read of a DUC. The number of cycles of delay between the first and second write and read operation is specified. The statement is used for critical timing problems where DUC sequencers are being tested "on the fly" instead of by "stepping" them from state to state.

Format:

PDQWRITE WRITE1(ADDR($1),DATA($2)),WRITE2(DELTA($3),ADDR($4),DATA($5))

,READ(DELTA($6),ADDR($7))

Characteristics of Parameters:

WRITE1 — Keyword denoting the parameter of the first write operation.

ADDR — Specifies the Datapool-defined name for any location in a DUC. The $1 is the Datapool name.

DATA — Specifies any 24-bit data pattern. The $2 is any arithmetic expression which expresses the data pattern.

DELTA — Specifies the number of cycles to delay before the second write operation. The $3 is a number from 8 to 63.

WRITE2 — Keyword denoting the parameters of the second write operation.

ADDR — Specifies the Datapool-defined name for any location in a DUC. The $4 is the Datapool name.

DATA — Specifies any 24-bit data pattern. The $5 is any arithmetic expression which expresses the data pattern.

READ — Keyword denoting the parameter of the read operation.

DELTA — Specifies the number of cycles to delay between the second write and read operation. The $6 is a number from 16 to 63.

ADDR — Specifies the Datapool-defined name for any location in a DUC. The $7 is the Datapool name.

Example:

```
PDQWRITE_ WRITE1(ADDR(TU1MR),DATA(M(TU1STAPEMOT)),
MC        WRITE2(DELTA(63),ADDR(TU1CIBG),DATA(0)),
ME        READ(DELTA(25),ADDR(TU1CIBG))
```

This statement writes the data symbolized by TU1STAPEMOT into DUC address TU1MR, waits 63 cycles, and writes zeros into DUC address TU1CIBG. After a 25-cycle wait, the DUC address TI1CIBG is read.

PPCSTRT

4.160 The description of the PPCSTRT statement includes:

Function:

The PPCSTRT statement tests the pulse point start of the standby central control operational clock. Various circuit functions associated with starting and stopping of the operational clock are also tested. The operational clock is started by pulse point from the active central control. Many clock status and control indicators are read and stored in diagnostic scratch call store. Various clock control functions are then generated and monitor points are read and stored in diagnostic scratch call store. The operational clock is then stopped. No tests are generated by this statement. The results stored in diagnostic scratch call store are interrogated by other DL-1 statements to determine the test results.

Format:

PPCSTRT

Characteristics of Parameters:

This statement has no parameters.

Example:

PPCSTRT

This statement starts the standby central control operational clock by pulse point. Then many clock status and control indicators are read and stored in diagnostic scratch call store. Various clock control functions are then generated and monitor points are read and stored in diagnostic scratch call store. The operational clock is then stopped.

PPIMAP0

4.161 The description of the PPIMAP0 statement includes:

Function:

The PPIMAP0 statement activates "mapping 0" in the processor peripheral interface (PPI) loop-around circuit, sends data patterns out on the peripheral unit write bus bit positions 0 through 23, and tests the information that was looped around and sent back to the central control over the peripheral unit reply bus bits 0 through 23.

Format:

PPIMAP0 PUWB2300($1),NOSTORE,NOSELECT

Characteristics of Parameters:

PUWB2300 — Specifies the data pattern that is to be written out on the bus. The $1 is any arithmetic expression which expresses the data pattern.

NOSTORE — Specifies that the reply is ignored.

NOSELECT — Specifies that there is no data selected.

Example:

PPIMAP0 PUWB2300(M(PPIOPUW02)),NOSTORE

This statement sends the mask of the PPIOPUW02 item out on bit positions 0 through 23 of the peripheral unit write bus. Nothing is done with the reply.

PPIMAP1

4.162 The description of the PPIMAP1 statement includes:

Function:

The PPIMAP1 statement activates that portion of the PPI loop-around circuit that receives from the central control the following bus bits: enable address bus bits 35 through 0 and peripheral unit write bus bits 35 through 24. Upon reception of the data pattern, the PPI loops the bus bits to the appropriate reply bus to be sent back to the central control. The PPI loops enable address bus bits 35 through 12 back to the central control over CPD reply bus bits 23 through 0. Bits 11 through 0 of the enable address bus are looped back to the central control by the PPI over peripheral unit reply bus bits 23 through 12. Peripheral unit write bus bits 35 through 24 are looped back to the central control by the PPI over peripheral unit reply bus bits 11 through 0. Along with the above bits looped-around, the PPI also sends the CPD ASW bit back to the central control over CPD reply bus bit 24.

Format:

PRIMAP1 EA3512($1),EA1100($2),PUWB3524($3),NOSTORE,NOSELECT

Characteristics of Parameters:

EA3512 — Specifies data for enable address bits 35 through 12. The $1 is any arithmetic expression which expresses the data.

EA1100 — Specifies data for enable address bits 11 through 0. The $2 is any arithmetic expression which expresses the data.

PUWB3524 — Specifies data for peripheral unit write bus bits 35 through 24. The $3 is any arithmetic expression which expresses the data.

NOSTORE — Specifies the reply is ignored.

NOSELECT — Specifies there is no data selected.

Example:

```
PRIMAP_EA3512(M(PPI1EA34,PPIEA27)),EA1100(1PPIALL0S),
ME          PUWB3524(M(PPI1PUWB27))
```

This statement sends the mask of items PPI1EA34 and PPIEA27 on enable address bits 35 through 12; the value of 1PPIALL0S on enable address bits 11 through 0, and the mask of the PPI1PUWB27 on peripheral unit write bus bits 35 through 24.

PPIMAP2

4.163 The description of the PPIMAP2 statement includes:

The PPIMAP2 statement activates "mapping 2" in the PPI loop-around circuit and tests the miscellaneous peripheral unit bus leads PDG, PSZ, PLP, PHP, PIO, PUSYC, MBIT, GI, MI, PUPEV, PUPOD, FCG, RESET, APUB, APUT, PURP, and PUASW.

Format:

```
PPIMAP2 PUPEV,PUPOD,PUSYC,MBIT,PSZ,PLP,PHP,PDG,PIO,GI,MI,FCG,RESET,NOSELECT
        ,NOSTORE
```

Characteristics of Parameters:

PDG — Sets bit 23; central control pulse sources are looped-around.

PSZ — Sets bit 22; central control pulse sources are looped-around.

PLP — Sets bit 21; central control pulse sources are looped-around.

PHP — Sets bit 20; central control pulse sources are looped-around.

PIO — Sets bit 19; central control pulse sources are looped-around.

PUSYC — Sets bit 13.

MBIT — Sets bit 12.

GI — Sets bit 10; central control pulse sources are looped-around.

MI — Sets bit 9; central control pulse sources are looped-around.

PUPEV — Sets bit 8 parity.

PUPOD — Sets bit 7 parity.

FCG — Sets bit 6; central control pulse sources are looped-around.

RESET — Sets bit 5; central control pulse sources are looped-around.

NOSTORE — Specifies the reply is to be ignored.

NOSELECT — Specifies that there is no data selected.

Example:

PPIMAP2 FCG,PIO,PLP,NOSTORE

This statement sets bits 6, 19, and 21 central control pulse sources that are looped-around. The reply is ignored.

PSWITCHC

4.164 The description of the PSWITCHC statement includes:

Function:

The PSWITCHC statement tests the start-stop control sequencer (SSCS) circuit in the standby central control. Specific clock control flip-flops are written in the start-stop-register (SSR) and the contents of the SSR and clock error group (CLE) registers are read and saved in diagnostic scratch call store. The contents of these scratch words are interrogated by later DL-1 statements.

Format:

PSWITCHC

Characteristics of Parameters:

This statement has no parameters.

Example:

PSWITCHC

This statement writes specific clock control flip-flops in the SSR and then reads the contents of the SSR and CLE registers and stores them in diagnostic scratches.

PUBCNFIG

4.165 The description of the PUBCNFIG statement includes:

Function:

The PUBCNFIG statement sets up the peripheral unit buses. It pulses the routing flip-flops of the peripheral unit buses and restores them before a segment break. Optionally, it configures all IOUSs onto the standby bus.

Format:

PUBCNFIG RESTORE

PUBCNFIG PUBAS,PUBOS,PUBTS,PUBRS

PUBCNFIG CNFIGIO,RESTORE
BUS=($1)

Characteristics of Parameters:

RESTORE — Specifies that any routing flip-flop pulsed should be restored before a segment break is taken.

PUBAS — If specified, the PUBA flip-flop is pulsed (set).

PUBOS — If specified, the PUBO flip-flop is pulsed (set).

PUBTS — If specified, the PUBT flip-flop is pulsed (set).

PUBRS — If specified, the PUBR flip-flop is pulsed (set).

CNFIGIO — Specifies that all IOUSs be configured on the specified bus.

BUS — Specifies the bus on which all IOUSs be configured. The $1 is 0 or 1.

Example:

This statement pulses the routing flip-flops PUBA and PUBT.

P1P2TEST

4.166 The description of the P1P2TEST statement includes:

Function:

The P1P2TEST statement allows reading the DUS-output buffer register (OBR) and then checking the P1, P2 parity bits sent back to the central control error summary register (CES) in adjacent instructions. This prevents a store operation from being executed after the ML instruction and before checking P1, P2. The ML instruction is described in Section 254-280-020, 1A Processor Assembly Language—Description.

Format:

P1P2TEST EXPECT($1),OPTIONS($2)

Characteristics of Parameters:

EXPECT — The expected state of the P1 and P2 bits in the central control CES register after the ML instruction is executed. The $1 is any arithmetic expression which expresses the expected results.

OPTIONS — If not specified, the C and R options are inserted; otherwise, any of the ML options are specified. The $2 is on list of ML options. The ML instruction is described in Section 254-280-020, 1A Processor Assembly Language—Description. Table A is a list of options for the ML and MS instructions.

Example:

P1P2TEST EXPECT(M(IN1PC1)),OPTIONS(C,R,IPKA)

This statement reads the DUS-OBR register and compares the P1, P2 parity bits to the mask of the IN1PC1 item. The C, R, and IPKA are options for the ML (read) instruction.

RDACTDSK

4.167 The description of the RDACTDSK statement includes:

Function:

The RDACTDSK statement is used in testing the file store servo system. This statement tests the mate file store to determine if the mate disk file is near the ORIGIN pulse (between SECTOR 100 and SECTOR 0). If the mate disk file is in SECTOR 100 and SECTOR 0, a 630-μs delay loop is entered; if not, the delay is omitted.

Format:

RDACTDSK DF($1)

Characteristics of Parameters:

DF — Specifies which disk file on the mate file store is to be tested. The $1 is 0, 1, 2, or 3.

Examples:

(a) RDACTDSK DF(0)

This statement reads active DF 0.

(b) RDACTDSK DF(3)

This statement reads active DF 3.

RD and RD6

4.168 The description of the RD and RD6 statements includes:

Function:

The RD and RD6 statements set up call store for tests that follow. The RD statement is concerned with sequencer tests, the RD6 with decoder tests. They store in call store the number of cycles the central control is to run, the expected results, and the address of the part of central control to be tested.

Format:

RD IEQUA,CYCLES($1),EXPECT($2) RD6

Characteristics of Parameters:

IEQUA — Specifies that the sequencer states are to be read.

CYCLES — Specifies the number of cycles the standby central control is to run. The $1 is a decimal number.

EXPECT — Specifies the expected results. The $2 is any arithmetic expression which expresses the expected results.

Example:

RD IEQUA,CYCLES(1),EXPECT(0)

This statement stores in call store the number of cycles the standby central control is to run (1), the expected results (0), and the address of the sequencer states to be read.

♦RDXHEAD

4.169 The description of the RDXHEAD statement includes:

Function:

The RDXHEAD statement saves in call store the mask, the address, and number of cycles that the standby central control is to run for the MPRDXRUN routine which follows immediately after the RDXHEAD routine.

Format:

RDXHEAD

Characteristics of Parameters:

This statement has no parameters.

Example:

RDXHEAD

This statement acts upon data collected from the previous XCR statement.

RDZREG

4.170 The description of the RDZREG statement includes:

Function:

The RDZREG statement causes the active central control to read a register or group of registers in the standby central control. The RDZREG will read the items that have both read and write access in the specified register(s) and will expect the result(s) to be zero.

Format:

RDZREG PARAMS

Characteristics of Parameters:

PARAMS — Register 1, Register 2, ..., Register 10
Up to 10 registers may be specified, separated by commas.

Example:

RDZREG ILR

This statement causes the active central control to read the interrupt level activity flip-flops in the standby central control and expect all 0s.

RDPPI

4.171 The description of the RDPPI statement includes:

Function:

The RDPPI statement reads a 24-bit row of data from either the master control console or SSD matrix in the processor peripheral interface unit. The controls of the row read are saved in the memory word DG1PPISAVRW.

Format:

RDPPI ROW($1),MASK($2),EXPT($3),RDPPI ROW($1),NOTEST

Characteristics of Parameters:

ROW — The $1 specifies one of the 64 rows to be read with a decimal value. The 0-63 is the range of valid rows.

MSK — The $2 specifies a 24-bit pattern of significant bits.

EXPT — The $3 specifies a 24-bit pattern of expected results.

NOTEST — If specified, the row is read without a call to DCONSTOR.

Examples:

(1) RDPPI ROW(5),MSK(1MCALL1S),EXPT(1MCALL)

(2) RDPPI ROW(24),NOTEST

REGTEST

4.172 The description of the REGTEST statement includes:

Function:

The REGTEST statement causes the active central control to do a series of write/read operations into a standby central control register. The data for the write operations is 0, alternate 1s and 0s, and alternate 0s and 1s ANDed with the mask of the read/write items of the register being tested. The read operation passes to the DCON program the contents of the register, the mask of the read/write items, and the expected results for processing raw results.

Format:

REGTEST $1

Characteristics of Parameters:

$1 — Specifies a register name in the standby central control without the prefix ST1.

Example:

REGTEST DAR

This statement causes the active central control to do a series of read/write operations into the data address register (DAR) in the standby central control.

RESMTST

4.173 The description of the RESMTST statement includes:

Function:

The RESMTST statement tests the circuitry involved with the active and standby central control pulse point reset of the millisecond clock (MCL) in both central controls. The statement guarantees that the active central control MCL will not be affected if it is reset. The statement ensures that no output clock functions are lost and that all internal counters are restored to their proper value if the MCL is reset. The central control MCL internal counters are read and compared with constants in a loop until the counters are within a small range of values. At this time, a new value of the MCL internal counters is computed which is eight central control clock cycles hence. The MCL will not output any clock pulses for at least eight cycles. The present value of the MCL is saved in diagnostic scratch call store. Then, either the active or standby central control specified by $1, generates a control pulse (GCP) to reset the MCL. The value of the active central control MCL is saved in diagnostic scratch call store and then the computed value of the counters is written into the active central control MCL. Writing the computed value of the counter into the active central control MCL restores the counters to their proper value if they were reset and if they were not reset, the computed value is the same value already in the counters. All 1s are written into diagnostic scratch call store location if the active central control MCL was not reset or all 0s are written if the MCL was reset. Also, stored in diagnostic scratch call store is the active central control MCL (1) computed counter value, (2) counter value before the reset pulse is generated, and (3) counter value after the reset pulse. The values and reset/not reset indicator stored in diagnostic scratch call store are interrogated by other DL-1 statements. No tests are generated by this statement.

Format:

RESMTST $1

Characteristics of Parameters:

$1 — Specifies which central control generates the reset pulse. The $1 is ACT or STBY.

Examples:

(a) RESMTST ACT

This statement reads the active central control MCL and compares it with constants in a loop until the counters are within a small range of values. Then eight cycles later, a new value for the MCL internal counters is computed. The present value of the MCL is saved in diagnostic scratch call store. Then the active central control generates a control pulse to reset the MCL. The computed value of the counters is written into the active central control MCL.

(b) RESMTST STBY

This statement performs identical tests as the previous statement except that the standby central control generates the control pulse to reset the MCL.

RSTICC

4.174 The description of the RSTICC statement includes:

Function:

The RSTICC statement restores an item of an internal central control register to a state saved by a previous CHGICC statement.

Format:

```
        ITEM($1)
RSTICC  ITEMS($2),ST($4)
        FF($3)
```

Characteristics of Parameters:

ITEM — Specifies an internal central control register address and bit or contiguous bits to be restored from call store scratch. The $1 is a Datapool-defined item.

ITEMS — Specifies an internal central control register address and noncontiguous bits to be restored from call store scratch. The $2 is a list of Datapool-defined items all in the same word.

FF — Specifies a central control buffer bus flip-flop to be restored from call store scratch. The $3 is a central control buffer bus flip-flop.

ST — Specifies one of six call store scratch locations used by a previous CHGICC statement to save a previous state of item or items. The $4 is TEMP0, TEMP1, TEMP2, TEMP3, TEMP4, or TEMP5.

Example:

RSTICC ITEMS(IN1CUG,IN1CUGL,IN1CUGS),ST(TEMP2)

This statement restores the internal central control registers IN1CUG, IN1CUGL, and IN1CUGS to the state previously saved in call store scratch location TEMP2.

▶SAPADDR

4.175 The description of the SAPADDR statement includes:

Function:

The SAPADDR statement is used to send the API all addresses in a specified range via the auxiliary unit address bus. The SAPADDR, after sending an address, GCPs the test unit to read the illegal address detected flip-flop (AU1ORAEOPS) within the API. The SAPADDR expects AU1ORAEOPS to be set.

Format:

SAPADDR START($1),END($2),[IPKA]

Characteristics of Parameters:

START — Specifies the starting address, inclusive. The $1 is any arithmetic expression which expresses an address.

END — Specifies the ending address, inclusive. The $2 is any arithmetic expression which expresses an address.

IPKA — Specifies that address parity (PKA) is to be inverted. Optional parameter.

Example:

(a) SAPADDR START(O(66)),END(O(77))

This statement attempts a control write of the API using all addresses in the range of O(66) to O(77), inclusive. After sending each address, the illegal address detector is read. Expect AU1ORAEOPS to be set.

(b) SAPADDR START(O(66)),END(O(77)),IPKA

This statement performs the same steps as in the previous example with the exception that address parity is inverted.◀

SBYPULSE

4.176 The description of the SBYPULSE statement includes:

Function:

The SBYPULSE statement is used to control pulse and reset critical pulse points in the standby central control. The flip-flop corresponding to the pulse point is restored to its previous state. The pulse points which are pulsed by this statement are defined by SBYPTEXT in Datapool. If ONLY is specified in the text, this statement is the only one which can pulse these points.

Format:

```
          SETS($1)
SBYPULSE RESET($1)
```

Characteristics of Parameters:

SET — Specifies that the pulse points are to be set. The $1 is the name of the pulse point.

RESET — Specifies that the pulse points are to be reset. The $1 is the name of the pulse point.

Example:

SBYPULSE SET(ST1PUBR)

This statement reads the flip-flop corresponding to pulse point ST1PUBR, sets the pulse point, and restores it to the previous state as indicated by the flip-flop.

SCANMCCROW24

4.177 The description of the SCANMCCROW24 statement includes:

Function:

The SCANMCCROW24 statement checks data insert keys 0 and 1 for a response as to whether a specific visual display appeared on the master control console control and display panel. If the maintenance personnel toggles data insert key 1 to a one, it indicates that the correct display appeared on the panel. If data insert key 0 is toggled to a one, it indicates the test failed. However, when no response is given, neither data insert key 0 or 1 is toggled; within 20 seconds the SCANMCCROW24 routine assumes the test has failed and proceeds to the next instruction.

Format:

SCANMCCROW24

Characteristics of Parameters:

This statement has no parameters.

Example:

SCANMCCROW24

This statement checks data insert keys 0 and 1 for a response as to whether a specific visual display appeared on the master control console control and display panel.

SRTAPTST

4.178 The description of the SRTAPTST statement includes.

Function:

The SRTAPTST statement calls a DUAD program subroutine to determine if the tape on the TUC under test is an SR tape. A diagnostic scratch word is set according to the response from DUAD.

Format:

SRTAPTST

Characteristics of Parameters:

This statement has no parameters.

Example:

SRTAPTST

This statement calls a DUAD subroutine to determine if the tape on the TUC under test is an SR tape.

STAREAD

4.179 The description of the STAREAD statement includes:

Function:

The STAREAD abnormal read statement provides the basic structure for testing the read ability of the store. The maintenance load (ML) instruction is used in the associated macro routine which gives the user complete control over the mode, parity, and store timing pulses. The macro is, briefly, expanded as follows:

Control writer (MS) to reset the CRI flip-flop

ML instruction

GCP

GCP.

Format:

```
          ITEM($1) EXPECT($4)
STAREAD ITEMS($2),EXP_P2P1($5),OPTIONS($7),STATE($8),KCODE($9),MASK($10)
          WORD($3),EXP_ASW($6)
          ,BUS($11),ADDR2($12),OPT2($13),MSK2($14),EXP2($15)
```

Characteristics of Parameters:

WORD — Specifies the address of the read (bits 0 through 15). The $3 is the address.

ITEM — Specifies the address (bits 0 through 15) of the read which is associated with a Datapool layout. The $1 is an item name.

ITEMS — Specifies the address (bits 0 through 15) of the read which is associated with a Datapool layout. The $2 is a list of items all in the same word.

EXPECT — Specifies the expected results in a 24-bit word. The $4 is any arithmetic expression which expresses the expected results.

EXP_P2P1 — Expected results data parity bits, P2 and P1, data bits 24 and 25 are rotated into bits 1 and 0 of the 24-bit word. The module is determined by address bit 15 specified in the word, item or items parameters. The $5 is any arithmetic expression which expresses the expected results data parity bits.

EXP_ASW — The ASW and ASWF leads (in central control) are rotated into bits 0 and 1 of the 24-bit word. The $6 is any arithmetic expression which expresses the ASW and ASWF expected results.

NOSTORE — Ignore results of the read, keeping the CRI flip-flop set.

OPTIONS — Specifies the mode, parity, and store timing during the read (ML) instruction. The $7 is a list of options. Figure 3 shows a list of the options used.

STATE — Specifies the states of the communications reply inhibit (CRI) and maintenance (MTCE) flip-flops during the ML instruction. The $8 is two of the following:

CRIR — Reset CRIR

CRIS — Keep CRI set

MTCER — Reset MTCE

MTCES — Keep MTCE set.

If not specified, $7 is (CRIR, MTCES).

MASK — Specifies the mask to be used. If not specified with WORD, a mask of all ones is used. If not specified with ITEM(S), the mask of the item(s) is used. The $10 is any arithmetic expression which expresses the mask.

KCODE — Specifies the K-code to be sent to the store during the ML instruction. The $9 is the K-code. If not specified the current store K-code is used.

BUS — Specifies the bus the store is put on during the ML instruction which is executed on the active bus. The $11 is ACT (active bus) or STB (standby bus). If not specified, the active bus is used.

ADDR2 — Specifies the address of the second read (ML) instruction if READ2 is specified in OPTIONS. The $12 is the address. If not specified, the address of the first read is used.

OPT2 — Specifies the mode, parity, and store timing during the second read (ML) instruction if READ2 is specified in OPTIONS. The $13 is a list of options. Figure 3 shows a list of the options used. If not specified, $13 is $7.

MSK2 — Specifies the mask for the second read (ML) instruction if READ2 is specified in OPTIONS. The $14 is any arithmetic expression which expresses the mask. If not specified, $14 is $10.

EXP2 — Specifies the expected results of the second read (ML) instruction if READ2 is specified in OPTIONS. The $15 is any arithmetic expression which expresses the expected results. If not specified, the expected results of the first read is used ($4 or $5).

Example:

```
STAREAD WORD(MU1DRW11),EXPECT(M)MU1DCR15)),OPTIONS(R,C,M),
        MASK(M(MU1DCR15))
```

This statement does a maintenance store (MS) to reset the CRI flip-flop, then sets the ML (read) options to R, C, and M; then reads the word at location MU1DRW11, masks the result through the mask of the MU1DCR14 item, and compares the result to the mask of the MU1DCR15 item. After the ML instruction is executed, two GCPs are performed.

MS and ML instructions

C — Control mode
M — Maintenance mode
R — Read
W — Write
IPKA — Invert address parity
ST ($1) — Inhibit store timing pulse. $1 is one of the following:
1T3
3T5E
3T5L
5T7

MS instruction only

IP1 — Invert data parity bit 1
IP2 — Invert data parity bit 2
IWE — Inhibit write enable pulse

Fig. 3—Options Used by the 256K Semiconductor Store

STAREAD3

4.180 The description of the STAREAD3 statement includes:

Function:

The store abnormal read is the basic macro used to test the read capability of the store. The associated macro task routine uses the maintenance load (ML) instruction to interrogate the store. The ML instruction provides control over the mode, parity, and store timing pulses. A general expansion of the macro would appear as follows:

Control write (MS instruction) to reset the CRI flip-flop

ML instruction

GCP

GCP.

Format:

```
          ITEM($1) EXPECT($5)
STAREAD3 ITEMS($2),EXP_P2P1($5),OPTIONS($6),STATE($7),
          WORD($3) EXP_ASW($5)
                   NOSTORE
KCODE($8),MASK($9),BUS($10),ADDR2($11),OPT2($12),MSK2($13),EXP2($14),NOROT
```

Characteristics of Parameters:

ITEM — Specifies the address (bits 0 through 17) of the read which is associated with a Datapool layout. The $1 is an item name.

ITEMS — Specifies the address (bits 0 through 17) of the read which is associated with a Datapool layout. The $2 is a list of items residing in the same word.

WORD — Specifies the address of the read (bits 0 through 17). The $3 is the Datapool-defined name of a word or some value.

EXPECT — Specifies the expected results in a 24-bit word. The $4 is any arithmetic expression which expresses the expected results.

EXP_P2P1 — Specifies the expected results of the data parity bits, P2 and P1, data bits 24 and 25 are rotated into bits 1 and 0 of the 24-bit word. The store word which is read is determined by address bit 15 of the word, item or items parameter. The $5 is any arithmetic expression which expresses the expected results of the data parity bits.

EXP_ASW — The all seems well (ASW), all seems well failure (ASWF) and data parity write enable failure (DPWEF) leads in the central control are rotated respectively into its 0, 1 and 2 of the 24-bit word. The $5 is any arithmetic expression which expresses the ASW, ASWF, and DPWEF expected results.

NOSTORE — Specifies to ignore the results of the read, keeping the CRI flip-flop set.

OPTIONS — Specifies the mode, parity, and store timing during the read (ML instruction). The $6 is a list of options. Figure 3 shows a list of valid options. READ2 may also be specified as an option if a second read (ML) is to be performed.

STATE — Specifies the state of the communications reply inhibit (CRI) and maintenance (MTCE) flip-flops during the ML instruction. The $7 can be specified as two items, one for CRI and one for MTCE from the following list:

CRIR — Reset CRI flip-flop
CRIS — Keep CRI set
MTCER — Reset MTCE flip-flop
MTCES — Keep MTCE set.

If not specified, $7 is (CRIR and MTCES).

KCODE — Specifies the K-code to be sent to the store during the ML instruction. The $8 is the K-code. If not specified, the current store K-code is used.

MASK — Specifies the mask used for the read. If not specified with WORD parameter, a mask of all ones is used. If not specified with ITEM(S) parameter, the mask of the item(s) is used. The $9 is any arithmetic expression which expresses the mask.

BUS — Specifies the bus the test store is put on during the ML instruction (the ML instruction is executed on the active bus). The $10 is ACT (active bus) or STBY (standby bus). If not specified, the active bus is used.

ADDR2 — Specifies the address of the second read (ML instruction) if READ2 is specified in OPTIONS. The $11 is the address. The address of the first read is used if ADDR2 is not specified.

OPT — Specifies the mode, parity, and store timing during the second read if READ2 is specified in OPTIONS. The $12 is a list of options. Figure 3 shows a list of valid options. If OPT2 is not specified, $12 is $6.

MSK2 — Specifies the mask used for the second read if READ2 is specified in OPTIONS. The $13 is an arithmetic expression which expresses the mask. If not specified, $13 is $9.

EXP2 — Specifies the expected results of the second read instruction if READ2 is specified in OPTIONS. The $14 is an arithmetic expression which expresses the expected results. If not specified, the expected results of the first read is used ($4 or $5).

NOROT -- Specifies that the address of the read instruction will not be rotated by the control program. If not specified, the address will be rotated as normal.

Example:

```
STAREAD3 WORD(MS1DRW10),EXPECT(O(0)),OPTIONS(R,C,M),
         MASK(M(MS1DTRFDPF1))
```

This statement does a maintenance store to reset the CRI flip-flop, then sets the read (ML) options to R, C, and M. The word at location MS1DRW10 is read; the results are masked through the mask of MS1DTRFDPF1 and then compared with the expected results O(0). After the read is completed, two GCPs are executed.

STAWRITE

4.181 The description of the STAWRITE statement includes:

Function:

The STAWRITE abnormal write statement provides the basic write facility for testing the store. The associated macro routine uses the maintenance store instruction which gives the user complete control over the mode, parity, and store timing pulses. The STAWRITE does not generate test results as it serves only in test initialization.

Format:

```
STAWRITE WORD($1),DATA($2),OPTIONS($3),STATE($4),KCODE($5),BUS($6)
```

Characteristics of Parameters:

WORD -- Specifies the address (bits 0 through 15) to be written into. The $1 is the address.

DATA — Specifies the data to be written. The $2 is any arithmetic expression which expresses the data.

OPTIONS — Specifies the mode, parity, and store timing during the write (MS) instruction. The $3 is a list of options. Figure 3 shows the list of options used. A delay is taken after the MS if DELAY is specified.

STATE — Specifies the state of the maintenance (MTCE) flip-flop during the MS instruction. The $4 is one of the following.

MTCER — Reset MTCE

MTCES — Keep MTCE set.

If not specified, $4 is MTCES.

If MTCER is specified, one MS instruction is executed to reset the MTCE FF; then the regular MS instruction is executed followed by two GCP instructions. The MS is followed by one GCP instruction if STB is specified in the BUS parameter.

KCODE — Specifies the K-code to be sent to the tested store during the MS instruction. The $5 is the K-code.

BUS — Specifies the bus the store is put on during the MS instruction which is executed on the active bus. The $6 is ACT (active bus) or STB (standby bus). If not specified, the active bus is used.

Example:

STAWRITE WORD(MU1DM200),DATA(1MUDADD15),OPTIONS(W,C,M)

This statement sets the MS (write) options to W, C, and M; then writes the value of 1MUDADD15 into location MU1DM200.

STAWRIT3

4.182 The description of the STAWRIT3 statement includes:

Function:

The store abnormal write is the basic macro used to test the write capability of the store. The associated macro task routine uses the maintenance store (MS) instruction to access the store. This instruction gives control over the mode, parity, and store timing pulses. The STAWRIT3 macro does not generate test results. It is used solely for initialization purposes. A general expansion of the macro would appear as follows:

Control write (MS instruction) to reset MTCE flip-flop if specified

MS instruction

GCP.

GCP — Performed if STB is not specified in the BUS parameter.

Format:

STAWRIT3 WORD($1),DATA($2),OPTIONS($3),STATE($4),KCODE($5),BUS($6),NOROT,NOCW01

Characteristics of Parameters:

WORD — Specifies the address (bits 0 through 17) to be written into. The $1 is the address.

DATA — Specifies the data to be written. The $2 is any arithmetic expression which expresses the data.

OPTION — Specifies the mode, parity, and store timing during the write instruction. The $3 is a list of options from the list in Fig. 3. If DELAY is specified, a delay of approximately 25 microseconds is taken after the MS is executed.

STATE — Specifies the state of the maintenance (MTCE) flip-flop during the MS instruction. The $4 can be one of the following:

MTCER — Reset MTCE

MTCES — Keep MTCE set.

If not specified, the MTCE flip-flop will remain set.

KCODE — Specifies the K-code to be sent to the store during the MS instruction. The $5 is an arithmetic expression representing the K-code. If no K-code is specified, the current K-code is used.

BUS — Specifies the bus the store should be on when the MS instruction is executed on the active bus. The $6 is ACT for active bus or STB for standby bus. If not specified, the active bus is used.

NOROT — Specifies that the address of the MS instruction will not be rotated by the control program. If not specified, the address will be rotated as normal.

NOCW01 — Specifies that the first control write 01 will not be performed.

Example:

```
STAWRIT3 WORD(MS1DRW00),DATA(0(1MSDDATA1S)),
         OPTIONS(W,C,M)
```

This statement sets the write options to W, C, and M, then writes the value of 1MSDDATA1S into location MS1DRW00. After the write operation, two GCPs are executed.

ST3ERRAN

4.183 The description of the ST3ERRAN statement includes:

Function:

The store error analysis is used to locate errors in store memory and place the error information in a table for possible use by the pattern analysis routine to print a histogram of the failing memory location. The routine works in the following manner. The store is initialized to a specified data pattern prior to calling error analysis. Error analysis looks at the diagnostic buffer table (DBT) to see if fault recognition has trapped a failing address and data pattern. If a failure is found, it will be processed as described below. If no failure is found, the store trap register is read to see if an address has been trapped. If so, the address is processed as described below. If no address is found, a delay of 100 ms is done while the store is allowed to refresh. The trap register is read and checked to see if an address was trapped during the 100-ms delay. If an address is found it is processed. If no address is found, another 100-ms delay occurs. The 100-ms delays are performed until four consecutive 100-ms delays occur without an address being trapped or if the maximum number of failures (20) has been reached.

If an address is trapped, the failure is processed as follows. The failure is recorded and the first of four possible data patterns (O(0), O(77777777), O(25252525), O(52525252)) is selected. The selected pattern is written into the failing address and the complement data is written into the complement address. All 18 addresses of 1 hamming distance away from the failing address are written with the test pattern. All 18 complement addresses of 1 hamming distance are written with complement test data. A 1.4-ms delay is taken and the test address is read. If the parity or data fails, the failure is recorded and the next of the four test patterns is tried. If the parity and data pass, the test pattern which cleared the failure is recorded. Once the failure is cleared, the routine goes back and starts delaying for 100-ms breaks to try and find another failing address. If the failure cannot be cleared using one of the four test patterns, all data is recorded and the routine goes on to possibly look for another failing address.

Format:

ST3ERRAN

Characteristics of Parameters:

This macro has no parameters.

Example:

ST3ERRAN

This macro will search for data parity failures using the procedure outlined above.

STBUSACT

4.184 The description of the STBUSACT statement includes:

Function:

The STBUSACT statement specifies which bus the diagnostic phase wants active. If the active specified bus cannot be configured, the phase is no tests run (NTR).

Format:

STBUSACT ABUS($1)

Characteristics of Parameters:

ABUS — Specifies desired bus. The $1 is 0 — BUS 0, 1 — BUS 1, E — either bus — the system's active bus at the start of the diagnostics.

Example:

STBUSACT ABUS(1)

This statement sets bus 1 to be active.

STBUSACT3

4.185 The description of the STBUSACT3 statement includes:

Function:

The store bus active is the macro used to specify which bus is to be made active. If the specified bus cannot be configured, the phase is NTR (no tests run).

Format:

STBUSACT3 ABUS($1)

Characteristics of Parameters:

ABUS — Specifies which bus is to be the active bus. The $1 is 0 for BUS 0, 1 for BUS 1 or E for either bus (the system's active bus will be the active bus).

Example:

STBUSACT3 ABUS(0)

This macro call will set bus 0 as the active bus.

STCREAD

4.186 The description of the STCREAD statement includes:

Function:

The STCREAD control read statement reduces to a STAREAD macro with the R, M, and C bits set. The R, M, and C bits are options which specify the mode of the ML instruction used by this macro. Refer to Table A.

Format:

```
          ITEM($1)   EXPECT($4)_________
STCREAD ITEMS($2), NOSTORE ,MASK($9)
          WORD($3)
```

Characteristics of Parameters:

WORD — Specifies the address of the read. The $3 is the address.

ITEM — Specifies the address of the read which is associated with a Datapool layout. The $1 is an item name.

ITEMS — Specifies the address of the read which is associated with a Datapool layout. The $2 is a list of items all in the same word.

EXPECT — Specifies the expected results in a 24-bit word. The $4 is any arithmetic expression which expresses the expected results.

NOSTORE — Ignore results of the read, keeping the CRI flip-flop set.

MASK — Specifies the mask to be used. If not specified with WORD, a mask of all 1s is used. If not specified with ITEM(S), the mask of the item(s) is used. The $9 is any arithmetic expression which expresses the mask.

Example:

STCREAD ITEM(MU1DRKREG),EXPECT(I(MU1DRKREG)*O(32))

Reduces to

STAREAD ITEM(MU1DRKREG),EXPECT(I(MU1DRKREG)*O(32)),OPTIONS(R,M,C)

STCREAD3

4.187 The description of the STCREAD3 statement includes:

Function:

The store control read macro expands to a STAREAD3 macro with the R, M, and C bits set.

Format:

```
          ITEM($1)
STCREAD ITEMS($2),EXPECT($4),MASK($9)
          WORD($3) NOSTORE
```

Characteristics of Parameters:

ITEM — Specifies the address (bit 0 through 17) of the read which is associated with a Datapool layout. The $1 is an item name.

ITEMS — Specifies the address (bits 0 through 17) of the read which is associated with a Datapool layout. The $2 is a list of items residing in the same word.

WORD — Specifies the address of the read (bits 0 through 17). The $3 is the Datapool-defined name of a word or some value.

EXPECT — Specifies the expected results in a 24-bit word. The $4 is any arithmetic expression which expresses the expected results.

NOSTORE — Specifies to ignore the results of the read, keeping the CRI flip-flop set.

MASK — Specifies the mask used for the read. If not specified with WORD parameter, a mask of all ones is used. If not specified with ITEM(S) parameter, the mask of the item(s) is used. The $9 is any arithmetic expression which expresses the mask.

Example:

STCREAD3 WORD(MS1DRW10),EXPECT(O(0))

Reduces to

STAREAD3 WORD(MS1DRW10),EXPECT(O(0)),OPTIONS(R,C,M)

STCTRTSTO

4.188 The description of the STCTRTSTO statement includes:

Function:

The STCTRTSTO statement will stop the tested semiconductor store's refresh by setting the FRH flip-flop. The flip-flop is then reset after a specified number of 1.4-ms cycles. The macro is, briefly, expanded as follows:

Control write (MS) to set the FRH flip-flop

Delay specified by number of 1.4-ms cycles

Control write (MS) to reset the FRH flip-flop.

Format:

STCTRTSTO CYCLE($1),SET($2)

Characteristics of Parameters:

CYCLE — Specifies the number of 1.4-ms cycles to stop refresh. The $1 is the number of cycles.

SET — Specifies the state of the TWF flip-flop during the execution of the macro and upon exit. The $2 is one of the following:

TWFS — Set TWF, allow refresh parity check

TWFR — Reset TWF, inhibit refresh parity check.

Example:

STCTRTSTO CYCLE(1),SET(TWFS)

This statement executes a control write (MS) to set the FRH and TWF flip-flops. A delay of 1.4 ms is then taken. This is followed by a control write to reset the FRH and set TWF flip-flops. The FRH and TWF flip-flops are set/reset by the same control write.

STCTSTO3

4.189 The description of the STCTSTO3 statement includes:

Function:

The store counter stop macro is used to stop the store's refresh circuitry by setting the FRH flip-flop. The flip-flop is then reset after a specified number of 1.4-ms cycles. A general expansion of the macro would appear as follows:

Control write to set the FRH flip-flop
Delay specified by number of 1.4-ms cycles
Control write to reset the FRH flip-flop.

Format:

STCTSTO3 CYCLE($1),SET($2)

Characteristics of Parameters:

CYCLE — Specifies the number of 1.4-ms cycles to stop refresh. The $1 is the number of cycles.

SET — Specifies the state of the refresh data parity check flip-flop (RDPC) during the execution of the macro and upon exit from the macro. The $2 is either:

RDPCS — Set RDPC to allow refresh data parity check

RDPCR — Reset RDPC to inhibit refresh data parity check.

Example:

STCTSTO3 CYCLE(3),SET(RDPCS)

This statement will execute a control write to set the FRH and RDPC flip-flops. A delay of 3 times 1.4 ms or 4.2 ms is then taken. This is followed by a control write to reset the FRH and set RDPC flip-flops.

STCWRITE

4.190 The description of the STCWRITE statement includes:

Function:

The STCWRITE control write statement reduces to an STAWRITE with W, M, and C bits set. The W, M, and C bits are options which specify the mode of the MS instruction used by the macro. Refer to Table A.

Format:

STCWRITE WORD($1),DATA($2)

Characteristics of Parameters:

WORD — Specifies the address to be written into. The $1 is the address.

DATA — Specifies the data to be written. The $2 is any arithmetic expression which expresses the data.

Example:

STCWRITE WORD(MU1DRW10),DATA(1MUDTRP08)

Reduces to

STAWRITE WORD(MU1DRW10),DATA(1MUDTRP08),OPTIONS(W,M,C)

STDRTEST

4.191 The description of the STDRTEST statement includes:

Function:

The STDRTEST statement tests the store data register (DR). The associated macro routine uses the maintenance store (MS) instruction to write the DR followed by the maintenance load (ML) instruction to read the DR. The user has complete control over the mode, parity, and store timing pulses during the MS and ML instructions. Additional parameters allow to perform tests when a write and a read must be adjacent. The macro is, briefly, expanded as follows:

Control write (MS) to reset the CRI flip-flop

MS instruction

ML instruction

GCP

GCP.

Format:

```
                                                          EXPECT($5)
STDRTEST WORD($1),DATA_EXPECT($2),WROPTS($3),RDADD($4),EXP_ASW($5)
                                                          EXP_P2P1($5)
                                                          NOSTORE

,RDOPTS($6),MASK($7),STATE($8),WRASW($9),ODWINDOW
```

Characteristics of Parameters:

WORD — Specifies the address (bits 0 through 15) at which the data register is control written. The $1 is the address.

DATA_EXPECT — Specifies the data to be written and the expected results of the ML instruction unless the EXPECT, EXP_ASW or EXP_P2P1 parameter is specified. The $2 is any arithmetic expression which expresses the data.

WROPTS — Specifies the mode, parity, and store timing during the write (MS) instruction. The $3 is a list of options. If not specified, $3 is M, C, and W. Figure 1 shows the list of options for the MS instruction.

RDADD — Specifies the address to be read during the ML instruction. If not specified, it is the write address. The $4 is the address.

RDOPTS — Specifies the mode, parity, and store timing during the read (ML) instruction. The $6 is a list of options. If not specified, $6 is M, C, and R. Figure 3 shows the list of options for the ML instruction.

EXPECT — Specifies the expected results in a 24-bit word. The $5 is any arithmetic expression which expresses the expected results.

EXP_P2P1 — Expected results data parity bits, P2 and P1, data bits 24 and 25 are rotated into bits 1 and 0 of the 24-bit word. The $5 is any arithmetic expression which expresses the expected results data parity bits. The module is determined by address bit 15 specified in the address.

EXP_ASW — The ASW and ASWF leads (in central control) are rotated into bits 0 and 1 of the 24-bit word. The $5 is any arithmetic expression which expresses the ASW and ASWF expected results.

NOSTORE — Specifies that the results of the read are to be ignored.

MASK — Specifies the mask to be used. If not specified with WORD, a mask of all ones is used. If not specified with ITEM(S), the mask of the item(s) is used. The $7 is any arithmetic expression which expresses the mask.

STATE — Specifies the states of the communications reply inhibit (CRI) and maintenance (MTCE) flip-flops during the MS and ML instructions. The $8 is one of the following:

CRIR — Reset CRI

CRIS — Keep CRI set

MTCER — Keep MTCE set

MTCES — Keep MTCE set.

If not specified, $8 is (CRIR, MTCES).

WRASW — Specifies the expected result of the ASW and ASWF leads after the MS instruction. The ASW and ASWF leads (in central control) are rotated into bits 0 and 1 of the 24-bit word. The $9 is any arithmetic expression which expresses the ASW and ASWF expected results.

ODWINDOW — Specifies three consecutive read (ML) instructions so that output data window faults can be detected. The first two read results are ignored.

Example:

```
STDRTEST WORD(MU1DM200),DATA_EXPECT(1MUDADD1S),WROPTS
         (W,C,M),EXPECT(0),RDOPTS(R,C,M),STATE(CRIS)
```

This statement sets the MS (write) options to W, C, M, and sets the ML (read) options to R, C, and M. Then the value of 1MUDADD1S is written into location MU1DM200. The location MU1DM200 is then read and compared to zeros. The CRI flip-flop is set. If the STATE (CRIS) was not specified, a maintenance store (MS) instruction would first be executed to reset the CRI flip-flop; then the last operation would be two GCP instructions (after the ML).

STDRTST3

4.192 The description of the STDRTST3 statement includes:

Function:

The Store Data Register Test macro is used to test the store data register (DR). The DR is written using the MS instruction to read back the contents of the DR. The general macro expansion is as follows:

Control write (MS) to reset the CRI flip-flop

MS instruction

ML instruction

GCP

GCP.

Format:

```
                                                                    EXPECT($5)
STDRTST3 WORD($1),DATA_EXPECT($2),WROPTS($3), RDADD($4),EXP_ASW($5),RDOPTS($6),
                                                                    EXP_P2P1($5)
                                                                    NOSTORE

        MASK($7),STATE($8),WRASW($9),ODWINDOW
```

Characteristics of Parameters:

WORD — Specifies the address (bits 0 through 17) to which the DR is written. The $1 is the address.

DATA_EXPECT — Specifies the data to be written and the expected result of the read unless the EXPECT, EXP_ASW, EXP_P2P1 or NOSTORE parameter is specified. The $2 is any arithmetic expression representing the data.

WROPTS — Specifies the mode, parity, and store timing during the write operation. The $3 is a list of options from the list in Fig. 3. If $3 is not specified, the options are W, C, and M.

RDADD — Specifies the address to be read during the ML instruction. If not specified, the write address is used. The $4 is the address.

EXPECT — Specifies the expected results of the read operation in a 24-bit word. The $5 is any arithmetic expression representing the expected results.

EXP_ASW — Specifies the expect for the ASW, ASWF, and DPWEF leads in the central control. These three bits are rotated into the bits 0, 1, and 2, respectively, of the expect word. The $5 is any arithmetic expression representing the ASW, ASWF and DPWEF expected results.

EXP_P2P1 — Specifies the expect for the two parity bits P2 and P1, data bits 24 and 25 are rotated into bits 1 and 0 of the expect word. The $5 is any arithmetic expression which represents the expect value for P2 and P1.

NOSTORE — Specifies that the results of the read instruction are to be ignored.

RDOPTS — Specifies the mode, parity, and store timing during the read. The $6 is a list of options from Fig. 3. If $6 is not specified, R, C, and M are assumed.

MASK — Specifies the mask to be used for the read operation. The $7 is any arithmetic expression which represents the mask value. If not specified, an all ones mask is used.

STATE — Specifies the state of the CRI and MTCE flip-flops during the MS and ML instructions. The $8 is one of the following:

CRIR — Reset CRI flip-flop

CRIS — Keep CRI set

MTCER — Reset MTCE flip-flop

MTCES — Keep MTCE set.

If not specified, $8 is (CRIR, MTCES).

WRASW — Specifies the expected results of the ASW, ASWF, and DPWEF leads after the MS instruction. The leads are rotated into bits 0, 1, and 2, respectively, of the 24-bit word. The $9 is any arithmetic expression which represents the ASW, ASWF, and DPWEF expect value.

ODWINDOW — Specifies that three consecutive reads will be performed so output data window faults can be detected. The first two read results are ignored.

Example:

```
STDRTST3 WORD(MS1DRW00),DATA_EXPECT(1DG_BIT01),
         WROPTS(W,C,M),EXPECT(0),RDOPTS(R,C,M), STATE(CRIS)
```

This statement sets the MS options to W, C, and M and the ML options to R, C, and M. The value of 1DG_BIT01 is written into address MS1DRW00. The CRI flip-flop is set and the MS1DRW00 address is read and expected to be 0. After the read, two GCP instructions are executed.

STEXER

4.193 The description of the STEXER statement includes:

Function:

The STEXER statement exercises the store memory at a high repetition rate—performs memory reads every store cycle.

Format:

STEXER STARTADD($1),BLOCKS($2),$\overline{\text{BITS}}$($3)

Characteristics of Parameters:

STARTADD — Specifies the start address to be exercised. The $1 is the address.

BLOCKS — Specifies the number of blocks of addresses to be exercised. A block consists of 512 consecutive reads. The $2 is the number of blocks.

BITS — Specifies the address bits to be exercised. The $3 is

BRAIL — B RAIL

BCUR — B CURRENT

AVERT — A VERTICAL

AHOR — A HORIZONTAL

ARAIL — A RAIL.

Example:

STEXER STARTADD(O(0)),BLOCKS(1)

This statement performs 512 consecutive memory reads starting at address octal zero.

STEXER2

4.194 The description of the STEXER2 statement includes:

Function:

The STEXER2 statement is used to perform a series of consecutive memory reads which are executed every 1400 ns to operate the tested store at system speed. Segment breaks occur automatically within the associated macro routine after a predetermined number of addresses have been exercised.

Format:

STEXER2 STARTADD ($1), BLOCKS ($2),BITS ($3)

Characteristics of Parameters:

STARTADD — Specifies the start address to be exercised. The $1 is the address (bits 0 through 15).

BLOCKS — Specifies the number of blocks of addresses to be read. A block consists of 512 consecutive reads. The $2 is the number of blocks. If $2 is one, then 512 consecutive memory reads will be executed. A value of zero will result in 512 memory reads of the start address $1.

BITS — Specifies the address bits to be held constant during the memory reads. The $3 is:

BRAIL — Address bits 0, 1, 2

BCUR — Address bits 3, 4

AVERT — Address bits 6, 7, 8

AHOR — Address bits 9, 10, 11

ARAIL — Address bits 12, 13, 14.

Example:

STEXER1 STARTADD(0),BLOCKS(1)

This statement performs 512 consecutive memory reads starting at address 0 and the last address read being O(777). The module read is mod 0 because address bit 15 = 0.

STEXER3

4.195 The description of the STEXER3 statement includes.

Function:

The store exercise macro is used to perform consecutive memory reads at operational speed. Memory is read in blocks of 512 addresses. Segment breaks occur automatically after a predetermined number of addresses have been exercised.

Format:

STEXER3 STARTADD($1),BLOCKS($2),MASK($3),KEEP($4)

Characteristics of Parameters:

STARTADR — Specifies the address to start with for the exercise routine. The $1 is the address (bits 0 through 17).

BLOCKS — Specifies the number of blocks of 512 consecutive addresses which are to be exercised. The $2 can be any number of blocks up to and including 8191.

MASK — Specifies the mask of the address bits which are to remain constant. The $3 is any arithmetic expression which represents the mask value.

KEEP — Specifies the value which the masked address bits are to be kept at. The $4 is either 0 or 1.

Example:

STEXER3 STARTADR(O(0)),BLOCKS(1),MASK(O(676)),KEEP(0)

This statement will exercise one block of 512 addresses starting with address 0. All address bits masked by the value O(676) will be kept at a constant value of 0.

STLKBUS2

4.196 The description of the STLKBUS2 statement includes:

Function:

The STLKBUS2 statement is used to detect leakage from the standby bus on MS instructions and leakage to the active bus, while the store is configured to the standby bus on ML instructions. The tests that are executed use both central controls and both buses. The statement executes a control write to reset CRI, followed by an MS instruction which is executed by both central controls and then an ML instruction is executed again by both central controls. Only the active central control looks at the results. This is then followed by one or two GCPs; one if the store is configured to the standby bus, two if the store is configured to the active bus. The macro is, briefly, expanded as follows:

Control write (MS) to reset CRI flip-flop

MS instruction

ML instruction

GCP

GCP.

Format:

```
                AITEM($1)                              SITEM($5)
STLKBUS2 AWORD($2),ADATA($3),AKCODE($4),SWORD($6),SDATA($7),SKCODE($8),

                MASK($9),

                EXPECT($10)
                EXP_P2P1($10),LPARND,WOPTIONS($11),ROPTIONS($12),STATE($13),BUS($14)
                EXP_ASW($10)
```

Characteristics of Parameters:

AITEM — Specifies the address (bits 0 through 15) to be used by the active central control for the MS and ML instructions. The $1 is a Datapool-defined item.

AWORD — Specifies the address (bits 0 through 15) to be used by the active central control for the MS and ML instructions. The $2 is the address (bits 0 through 15).

ADATA — Specifies data to be used by the active central control for the MS instruction. The $3 is any arithmetic expression which expresses the data.

AKCODE — Specifies the K-code to be used by the active central control for the MS and ML instructions. The $4 is the K-code.

SITEM — Specifies the address (bits 0 through 15) to be used by the standby central control for the MS and ML instructions. The $5 is a Datapool-defined item.

SWORD — Specifies the address (bits 0 through 15) to be used by the standby central control for the MS and ML instructions. The $6 is the address.

SDATA — Specifies the data to be used by the standby central control for the MS instruction. The $7 is any arithmetic expression which expresses the data.

SKCODE — Specifies the K-code to be used by the standby central control for the MS and ML instructions. The $8 is the K-code.

MASK — Specifies the mask. The $9 is any arithmetic expression which expresses the mask.

EXPECT — Specifies the expected results in a 24-bit word. The $10 is any arithmetic expression which expresses the data.

EXP_P2P1 — Expected results data parity bits, P2 and P1, data bits 24 and 25 are rotated into bits 1 and 0 of the 24-bit words. The $10 is any arithmetic expression which expresses the expected results data parity bits. The module tested is specified by address bit 15 of the address used by the active central control for the MS and ML instructions.

EXP_ASW — The ASW and ASWF leads (in central control) are rotated into bits 0 and 1 of the 24-bit word. The $10 is any arithmetic expression which expresses the ASW and ASWF expected results.

LPARND — If specified, the ML instruction performed is an address loop-around.

WOPTIONS — Specifies the mode and parity during the MS instruction. The $11 is a list of options. If not specified, $11 is M, C, and W. Figure 1 shows the list of options for the MS instruction.

ROPTIONS — Specifies the mode, parity, and store timing during the ML instruction. The $12 is a list of options. If not specified, $12 is M, C, and R. Figure 1 shows the list of options for the ML instruction.

STATE — Specifies the state of the maintenance (MTCE) flip-flop during the MS and ML instructions. The $13 is one of the following:

MTCER — Reset MTCE

MTCES — Keep MTCE set.

If not specified, $13 is MTCES.

BUS — Specifies the bus the store is put on during the MS/ML instructions. The $14 is ACT (active bus) or STB (standby bus). If not specified, $14 is ACT.

Example:

```
STLKBUS2 AWORD(DG1SCR1),ADATA(O(177)),AKODE(DG1KCODE),
         SWORD(DG1SCR2),SDATA(O(711)),SKCODE(DG1KCODE),
         MASK(O(711),EXPECT(O(711)
```

This statement executes a control write to reset the CRI flip-flop. Then both central controls execute an MS instruction with M, C, and W options. The data for the active central control is octal 177, the K-code for the active central control is specified by the value of DG1KCODE, and the address for the active central control operation is contained in DG1SCR1. The data for the standby central control is octal 711, the K-code for the standby central control is specified by the value of DG1KCODE, and the address for the standby central control operation is contained in DG1SCR2. Then both central controls execute ML instructions with the M, C, and R options. The address, K-code, and data is the same as for the MS instruction. The mask is O(711). The active central control looks for the expected result of O(711). Then two GCPs are generated.

STLKBUS3

4.197 The description of the STLKBUS3 statement includes:

Function:

The Store Leaky Bus is used to detect leakage from the standby bus (MS instructions) to the active bus (ML instructions) while the store is configured to the standby bus. The tests use both central controls and both buses. A control write to reset CRI is executed followed by an MS instruction which is executed by both central controls. An ML instruction is then executed from both central controls with only the active central controls looking at the results. After the ML, one GCP is executed if the store is configured to the standby bus or two GCPs are executed if the store is on the active bus. The general expansion of the macro would appear as follows:

Control write to reset the CRI flip-flop

MS instruction

ML instruction

GCP

GCP.

Format:

```
            AITEM($1)                                SITEM($5)
STLKBUS3 AWORD($2),ADATA($3),AKCODE($4),SWORD($6),

                                  EXPECT($10)
         SDATA($7),SKCODE($8),MASK($9),EXP_P2P1($10),
                                  EXP_ASW($10)

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

         LPARND,WOPTIONS($11),ROPTIONS($12),STATE($13),BUS($14)
```

Characteristics of Parameters:

AITEM — Specifies the address (bits 0 through 17) to be used by the active central control for the MS and ML instructions. The $1 is any valid Datapool-defined item.

AWORD — Specifies the address (bits 0 through 17) to be used by the active central control for the MS and ML instructions. The $2 is any valid arithmetic expression representing the address.

ADATA — Specifies the data to be used by the active central control for the MS instruction. The $3 is any arithmetic expression which represents the data.

AKCODE — Specifies the K-code to be used by the active central control for the MS and ML instructions. The $4 is the K-code.

SITEM — Specifies the address (bits 0 through 17) to be used by the standby central control for the MS and ML instructions. The $5 is any valid Datapool-defined item.

SWORD — Specifies the address (bits 0 through 17) to be used by the standby central control for the MS and ML instruction. The $6 is any valid arithmetic expression representing the address.

SDATA — Specifies the data to be used by the standby central control for the MS instruction. The $7 is any valid arithmetic expression which represents the data.

SKCODE — Specifies the K-code to be used by the standby central control for the MS and ML instructions. The $8 is the K-code.

MASK — Specifies the mask value. The $9 is any valid arithmetic expression which represents the mask value.

EXPECT — Specifies the expected results of the test in a 24-bit word. The $10 is any valid arithmetic expression which represents the expect data.

EXP_P2P1 — Specifies the expected results of the parity bits, P2 and P1, data bits 24 and 25 are rotated into bits 1 and 0 of the 24-bit word. The $10 is any valid arithmetic expression which represents the expected result of the parity bits.

EXP_ASW — Specifies the expected results for the ASW, ASWF, and DPWEF. The leads in the central control are rotated into bits 0, 1, and 2, respectively, of the 24-bit word. The $10 is any valid arithmetic expression which represents the expected results for ASW, ASWF, and DPWEF.

LPARND — Specifies the ML instruction being executed is an address loop-around; ie, R=W.

WOPTIONS — Specifies the mode and parity during the MS instruction. The $11 is a list of options from Fig. 3. If $11 is not specified, the options are W, C, and M.

ROPTIONS — Specifies the mode, parity, and store timing during the ML instruction. The $12 is a list of options from Fig. 3. If $12 is not specified, the options are R, C, and M.

STATE — Specifies the state of the MTCE flip-flop during the MS and ML instructions. The $13 is one of the following:

MTCER — Reset MTCE flip-flop

MTCES — Keep MTCE set.

If not specified, $13 is MTCES.

BUS — Specifies the bus which the store is put on during the MS/ML instructions. The $14 is ACT for active bus or STB for standby bus. If not specified, $14 is ACT.

Example:

```
STLKBUS3 AWORD(MS1DRW00),ADATA(1MSDDATA0S),
         AKCODE(0(32)),SWORD(MS1DRW01),SDATA(1MSDDATA1S),
         SKCODE(0(32)),MASK(1MSDDATA1S),EXPECT(1MSDDATA0S),
         WOPTIONS(W,M,C),ROPTIONS(R,M,C)
```

This statement first executes a control write to reset the CRI flip-flop. Both central controls then execute an MS instruction with W, M, and C options. The active central controls data is 1MSDDATA0S; the K-code is 32 and the address for the active central control is MS1DRW00. The data for the standby central control is 1MSDDATA1S. The K-code is 32 and the address for the standby central control is MS1DRW01. Both central controls then execute an ML instruction with R, M, and C options. The address, K-code, and data are the same as for the MS instruction. The mask is 1MSDDATA1S and the expected result for the active central control is 1MSDDATA1S. Following this, two GCPs are executed.

STLKYBUS

4.198 The description of the STLKYBUS statement includes:

Function:

The STLKYBUS statement is used to detect leakage from the standby bus on MS instructions and leakage to the active bus, while the store is configured to the standby bus on ML instructions. The tests that are executed use both central controls and both buses. This statement executes a control write to reset CRI, followed by an MS instruction which is executed by both central controls and then an ML instruction is executed again by both central controls. Only the active central control looks at the results. This is then followed by one or two GCPs, one if the store is configured to the standby bus, two if the store is configured to the active bus. The MS, ML, and GCP instructions are described in Section 254-280-020, Assembly Language—Description, 1A Processor.

Format:

```
            AITEM($1)              SITEM($5)
STLKYBUS AWORD($2),ADATA($3),AKCODE($4),SWORD($6),SDATA($7),SKCODE($8),MASK($9),
         EXPECT($10) _____________________________________________________
         EXP_P2P1($10),LPARND,WOPTIONS($11),ROPTIONS($14),STATE($13),BUS($14)
         EXP_ASW($10)
```

Characteristics of Parameters:

AITEM — Specifies the address to be used by the active central control for the MS and ML instructions. The $1 is a Datapool-defined item.

AWORD — Specifies the address to be used by the active central control for the MS and ML instructions. The $2 is the address.

ADATA — Specifies data to be used by the active central control for the MS instruction. The $3 is any arithmetic expression which expresses the data.

AKCODE — Specifies the K-code to be used by the active central control for the MS and ML instructions. The $4 is the K-code.

SITEM — Specifies the address to be used by the standby central control for the MS and ML instructions. The $5 is a Datapool-defined item.

SWORD — Specifies the address to be used by the standby central control for the MS and ML instructions. The $6 is the address.

SDATA — Specifies the data to be used by the standby central control for the MS instruction. The $7 is any arithmetic expression which expresses the data.

SKCODE — Specifies the K-code to be used by the standby central control for the MS and ML instructions. The $8 is the K-code.

MASK — Specifies the mask. The $9 is any arithmetic expression which expresses the mask.

EXPECT — Specifies the expected results in a 24-bit word. The $10 is any arithmetic expression which expresses the data.

EXP_P2P1 — Expected results data parity bits, P2 and P1, data bits 24 and 25 are rotated into bits 0 and 1 of the 24-bit word. The $10 is any arithmetic expression which expresses the expected results data parity bits.

EXP_ASW — The ASW and ASWF leads (in central control) are rotated into bit 0 and 1 of the 24-bit word. The $10 is any arithmetic expression which expresses the ASW and ASWF expected results.

LPARND — If specified, the ML instruction performed is an address loop-around.

WOPTIONS — Specifies the mode and parity during the MS instruction. The MS instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor. The $11 is a list of option. If not specified, $11 is M, C, and W. Table A is a list of options for the MS and ML instructions.

ROPTIONS — Specifies the mode, parity, and store timing during the ML instruction. The ML instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor. The $14 is a list of options. If not specified, $14 is M, C, and R. Table A is a list of options for the MS and ML instructions.

STATE — Specifies the state of the maintenance (MTCE) flip-flop. The $13 is one of the following:

MTCER — reset MTCE

MTCES — set MTCE.

If not specified, $13 is MTCES.

BUS — Specifies the bus the store is configured to during the MS/ML instructions. The $14 is ACT (active bus) or STB (standby bus). If not specified, the active bus is used.

Example:

```
STLKYBUS_AWORD(DG1SCR1),ADATA(O(177),AKCODE(DG1KCODE),SWORD(DG1SCR2),
        SDATA(O(711),SKCODE(DG1KCODE),MASK(O(77777777)),EXPECT(O(77777777))
```

This statement executes a control write to reset the CRI flip-flop. Then both central controls execute an MS instruction with M, C, and W options. The data for the active central control is octal 177; the K-code for the active central control is contained in DG1KCODE, and the address for the active central control operation is contained in DG1SCR1. The data for the standby central control is octal 711, the K-code for the standby central control is contained in DG1KCODE, and the address for the standby central control operation is contained in DG1SCR2. Then both central controls execute ML instructions with the M, C, and R options. The address, K-code, and data is the same as for the MS instruction. The mask is all 1s (O(77777777). The active central control looks for the expected result of all 1s (O(77777777)). Then two GCPs are generated.

STLREAD

4.199 The description of the STLREAD statement includes:

Function:

The STLREAD loop-around read statement reduces to a STAREAD with the R, W, M, and C bits set. The R, W, M, and C bits are options which specify the mode of the ML instruction used by this macro. Refer to Table A.

Format:

```
          ITEM($1) EXPECT($4)
STLREAD ITEMS($2),NOSTORE,MASK($9)
          WORD($3)
```

Characteristics of Parameters:

WORD — Specifies the address of the read. The $3 is the address.

ITEM — Specifies the address of the read which is associated with a Datapool layout. The $1 is an item name.

ITEMS — Specifies the address of the read which is associated with a Datapool layout. The $2 is a list of items all in the same word.

EXPECT — Specifies the expected results in a 24-bit word. The $4 is any arithmetic expression which expresses the expected results.

MASK — Specifies the mask to be used. If not specified with WORD, a mask of all 1s is used. If not specified with ITEM(S), the mask of the item(s) is used. The $9 is any arithmetic expression which expresses the mask.

NOSTORE — Ignore results of the read, keeping the CRI flip-flop set.

Example:

```
STLREAD WORD(1MUDADD040),EXPECT(1MUDADD040),
        MASK(1MUDLOOPMSK)
```

Reduces to

```
STAREAD WORD(1MUDADD040),EXPECT(1MUDADD040),
        OPTIONS(R,W,M,C),MASK(1MUDLOOPMSK)
```

STLREAD3

4.200 The description for the STLREAD3 statement includes:

Function:

The store loop-around read macro expands to a STAREAD3 macro with R, W, M, and C bits set. The macro is used for address loop-around tests.

Format:

```
          ITEM($1) EXPECT($4)_________
STLREAD3 ITEMS($2), NOSTORE, MASK($9)
          WORD($3)
```

Characteristics of Parameters:

ITEM — Specifies the address (bits 0 through 17) of the read which is associated with a Datapool layout. The $1 is an item name.

ITEMS — Specifies the address (bits 0 through 17) of the read which is associated with a Datapool layout. The $2 is a list of items residing in the same word.

WORD — Specifies the address of the read (bits 0 through 17). The $3 is the Datapool-defined name of a word or some value.

EXPECT — Specifies the expected results in a 25-bit word. The $4 is any arithmetic expression which expresses the expected results.

NOSTORE — Specifies to ignore the results of the read, keeping the CRI flip-flop set.

MASK — Specifies the mask used for the read. If not specified with WORD parameter, a mask of all ones is used. If not specified with ITEM(S) parameter, the mask of the item(s) is used. The $9 is any arithmetic expression which expresses the mask.

Example:

```
STLREAD WORD(1MSDADD090),EXPECT(1MSDADD090),
        MASK(1MSDLOOPMSK)
```

Reduces to:

```
STAREAD3 WORD(1MSDADD090),EXPECT(1MSDADD090),
         OPTIONS(R,W,M,C),MASK(1MSDLOOPMSK)
```

STMARCH3

4.201 The description of the STMARCH3 statement includes:

Function:

The store memory march is used to march a data pattern through a complement data pattern. After the memory has been initialized to some known data pattern by the STWRMEM3 routine, the march routine starts at address 0 or 777777 and reads that address. The complement data is then written back into that address and the address is incremented or decremented by one. The entire memory is read in this manner. Segment breaks are automatically taken during the march routine.

Format:

STMARCH3 MARCH($1),THRU($2),DIRCTN($3)

Characteristics of Parameters:

MARCH — Specifies the data pattern to be written into each address after the read. The $1 is any valid arithmetic expression.

THRU — Specifies the data which is expected on the read operation. The $2 is any valid arithmetic expression.

DIRCTN — Specifies the direction of the march. The $3 can be UP for starting at address 0 and incrementing upward or DOWN for starting at address 777777 and decrementing downward. If not specified, direction is assumed UP.

Example:

STMARCH3 MARCH(1DGALT01),THRU(1DGALT10)

This statement will read starting at address 0 and expect 1DGALT10. Following the read, a write will be performed to address 0 with data of 1DGALT01. The address will be incremented by one and the above procedure repeated until the address reaches 777777. Any failures will be stored away for processing by the pattern analysis routine.

STMCCRD

4.202 The description of the STMCCRD statement includes:

Function:

The STMCCRD statement is used to read the store status from the master control console. The status is indicated by the states of the following flip-flops: maintenance (MTCE), answer on bus 0 (ANS0), answer on bus 1 (ANS1), bus receive on (RO), and the K-code of the store under test.

Format:

STMCCRD MASK($1),EXPECT($2)

Characteristics of Parameters:

MASK — Specifies the mask to be used. The $1 is any arithmetic expression which expresses the mask.

EXPECT — Specifies the expected results of the read. The $2 is any arithmetic expression which expresses the expected results.

Example:

STMCCRD MASK (O(07600000)),EXPECT(0)

This statement causes a master control console read to determine the store status as indicated by the MTCE, RO, ANS1, and ANS0 flip-flops, and the K-code of the store under test.

STMCCRD2

4.203 The description of the STMCCRD2 statement includes:

Function:

The STMCCRD2 statement is used to read the store status from the master control console. The status is indicated by the states of the following flip-flops: maintenance (MTCE), answer on bus 0 (ANS0), answer on bus 1 (ANS1), bus receive on (RO), and the K-code of the store under test.

Format:

STMCCRD2 MASK($1),EXPECT($2)

Characteristics of Parameters:

MASK — Specifies the mask to be used. The $1 is any arithmetic expression which expresses the mask.

EXPECT — Specifies the expected results of the read. The $2 is any arithmetic expression which expresses the expected results.

Example:

STMCCRD2 MASK(O(07600000)),EXPECT(O(07600000))

This statement causes a master control console read to determine the store status as indicated by the MTCE, RO, ANS1, and ANS0 flip-flops, and the K-code of the store under test.

The following shows the status bits as they are read from the PPI master control console.

ANS0 — Bit 12
ANS1 — Bit 13
RO — Bit 14
MTCE — Bit 15
KCODE — Bits 16 through 20.

STMCCRD3

4.204 The description of the STMCCRD3 statement includes.

Function:

The store master control console read macro is used to read the store status from the master control console. The status is indicated by the status of the MTCE, answer on bus 0 (ANS0), answer on bus 1 (ANS1) and receive on (RO) flip-flops along with the K-code.

Format:

STMCCRD3 MASK($1),EXPECT($2)

Characteristics of Parameters:

MASK — Specifies the mask used for the read. The $1 is any valid arithmetic expression which represents the mask value.

EXPECT — Specifies the expected result of the read. The $2 is any valid arithmetic expression which represents the expect value.

Example:

STMCCRD3 MASK(M(MC2PSRO)),EXPECT(M)MC1PSRO))

This statement will read the status of the RO flip-flop and expect it to be set. The following shows the status bits which can be read from the PPI master control console:

ANS0 — Bit 12
ANS1 — Bit 13
R0 — Bit 14
MTCE — Bit 15
KCODE — Bits 16 through 20.

STMHWPM3

4.205 The description of the STMHWPM3 statement includes:

Function:

The store march write protect memory macro is used to exercise the write protect memory. The store is first initialized using the STWRWPM3 routine. The march routine then starts at address 0 or 1777 and reads the address. The complement data is then written back into the address. The address is then incremented or decremented and the procedure is repeated. Segment breaks are automatically generated by the routine. Failures are stored away to be processed by the pattern analysis routine.

Format:

STMHWPM3 EXPECT($1),DIRECTION($2)

Characteristics of Parameters:

EXPECT — Specifies the expected value for the read. The $1 is either 1 of 0.

DIRECTION — Specifies the direction of the march. The $2 is UP which starts at address 0 or DOWN which starts at address 1777.

Example:

STMHWPM3 EXPECT(1),DIRECTION(DOWN)

This macro call will start at address 1777 and read the address expecting a 1. The address will be written with the complement data (0) and the address will be decremented. This process will continue until all 1K of the write protect memory has been exercised.

STMREAD

4.206 The description of the STMREAD statement includes:

Function:

The STMREAD maintenance read statement reduces to a STAREAD macro with the M and R bits set. The M and R bits are options which specify the mode of the ML instruction used by this macro. Refer to Table A.

Format:

```
          ITEM($1)  EXPECT($4)
STMREAD ITEMS($2),NOSTORE, MASK($9)
          WORD($3)
```

Characteristics of Parameters:

WORD — Specifies the address of the read. The $3 is the address.

ITEM — Specifies the address of the read which is associated with a Datapool layout. The $1 is an item name.

ITEMS — Specifies the address of the read which is associated with a Datapool layout. The $2 is a list of items all in the same word.

EXPECT — Specifies the expected results in a 24-bit word. The $4 is any arithmetic expression which expresses the expected results.

MASK — Specifies the mask to be used. If not specified with WORD, a mask of all ones is used. If not specified with ITEM(S), the mask of the item(s) is used. The $9 is any arithmetic expression which expresses the mask.

NOSTORE — Ignore results of the read, keeping the CRI flip-flop set.

Example:

STMREAD WORD(MU1DRW00),NOSTORE

Reduces to:

STAREAD WORD(MU1DRW00),NOSTORE,OPTIONS(M,R)

STMWALK

4.207 The description of the STMWALK statement includes:

Function:

The STMWALK statement is the second half of a memory test. Prior to the execution of this statement all memory locations are written with data; ie, all ones. The associated macro routine uses the maintenance load (ML) instruction to read an address expecting the data written (all ones). This is then followed by a maintenance store (MS) instruction to write new data; ie, all zeros. All memory locations are read then written either starting at address 0 and running through address octal 177777 (UP) or starts at address octal 177777 and running through address 0 (DOWN). All data bit failures are shown by the store histogram printout. Segment breaks occur automatically within the routine after a predetermined number of addresses are read/written.

Format:

STMWALK WALK($1),THRU($2),DIRCTN($3)

Characteristics of Parameters:

WALK — Specifies that data to be written. The $1 is any arithmetic expression which expresses the data.

THRU — Specifies the expected result in a 24-bit word. The $2 is any arithmetic expression which expresses the expected result.

DIRCTN — Specifies the direction (up or down) of the read/write. The $3 is either UP (default) or DOWN.

Example:

STMWALK WALK (1DG_ONES),THRU(1DG_ZEROS)

Prior to this statement another macro statement, ie, STWRMEMS, would write all memory locations with data 1DG_ZEROS. The STMWALK routine will read an address expecting data 1DG_ZEROS, then write this address with 1DG_ONES starting at address 0 and running through octal 177777.

STMWRITE

4.208 The description of the STMWRITE statement includes:

Function:

The STMWRITE maintenance write statement reduces to a STAWRITE with the W and M bits set. The W and M bits are options which specify the mode of the MS instruction used by this macro. Refer to Table A.

Format:

STMWRITE WORD($1),DATA($2)

Characteristics of Parameters:

WORD — Specifies the address to be written into. The $1 is the address.

DATA — Specifies the data to be written. The $2 is any arithmetic expression which expresses the data.

Example:

STMWRITE WORD(1MUDADD1S),DATA(1MUDADD0S)

Reduces to

STAWRITE WORD(1MUDADD1S),DATA(1MUDADD0S),
OPTIONS(W,M)

STPCLKA

4.209 The description of the STPCLKA statement includes:

Function:

The STPCLKA statement tests the operational clock control circuitry associated with stopping the operational clock in the standby central control. Clock control flip-flops are written to start and stop the operational clock. Clock control and clock status indicators are read and stored in diagnostic scratch call store. Each clock control flip-flop written causes a particular circuit function to occur and these functions are monitored by reading and storing the control and status indicators. Other DL-1 statements interrogate the values stored in diagnostic scratch call store. No tests are generated by this statement.

Format:

STPCLKA

Characteristics of Parameters:

This statement has no parameters.

Example:

STPCLKA

This statement writes clock control flip-flops to start and stop the operational clock. Clock control and clock status indicators are read and stored in diagnostic scratch call store.

STRCLKA

4.210 The description of the STRCLKA statement includes:

Function:

The STRCLKA statement tests the operational clock control circuitry associated with starting the operational clock in the standby central control. Clock control flip-flops are written which start and stop the operational clock. Clock control and status indicators are read and stored in diagnostic scratch call store. The writing of the control flip-flops causes certain clock circuit functions to occur and these functions are monitored by reading the control and status indicators. No tests are generated by this statement. The results stored in diagnostic scratch call store are interrogated by other DL-1 statements.

Format:

STRCLKA

Characteristics of Parameters:

This statement has no parameters.

Example:

STRCLKA

This statement writes clock control flip-flops which start and stop the operational clock. Clock control and status indicators are then read and stored in diagnostic scratch call store.

STRCLKB

4.211 The description of the STRCLKB statement includes:

Function:

The STRCLKB statement tests a function of the start-stop control sequence (SSCS) start circuitry. The statement tests the processor configuration circuit input to the operational clock start circuitry of the SSCS. The processor configuration input is activated by a control write to the start-stop register (SSR) in the standby central control. After a brief delay, the SSR and the clock error group (CLE) register are control read and their contents are stored in diagnostic scratch call store. The results of the control reads are interrogated by other DL-1 statements.

Format:

STRCLKB

Characteristics of Parameters:

This statement has no parameters.

Example:

STRCLKB

This statement activates the processor configuration input by a control write to the SSR in the standby central control. After a brief delay, the SSR and SLE are control read and their contents are stored in diagnostic scratch call store.

STRCSYNC

4.212 The description of the STRCSYNC statement includes:

Function:

The STRCSYNC statement tests the standby central control ring counter synchronization circuitry. The standby central control ring counter is stopped and started twice. Before and after each stopping and starting of the ring counter, various control and monitor points are initialized and reread. These points are used to inhibit and to verify certain functions of the synchronization circuitry. All results which are control read are stored in diagnostic scratch call store and are interrogated by other DL-1 statements. The standby central control ring counter is left running and in synchronism with the active central control ring counter.

Format:

STRCSYNC

Characteristics of Parameters:

This statement has no parameters.

Example:

STRCSYNC

The statement starts and stops the standby central control ring counter twice, initializing and reading various monitor and control points before and after each stopping and starting. All results of the control reads are stored in diagnostic scratch call store. The standby central control ring counter is left running and in synchronism with the active central control ring counter.

STRDGCP

4.213 The description of the STRDGCP statement includes:

Function:

The STRDGCP statement executes an abnormal read to the store (ML instruction), ignoring the results followed by a GCP to check the effects of the read. The GCP and ML instruction are described in Section 254-280-020, Assembly Language—Description, 1A Processor.

Format:

```
        ITEM($1)
STRDGCP ITEMS($2),EXPECT($4),OPTIONS($5),KCODE($6),MASK($7),BUS($8)
        WORD($3)
```

Characteristics of Parameters:

ITEM — Specifies the address to be read, associated with the address of its layout. The $1 is an item name.

ITEMS — Specifies the address to be read, associated with the address of its layout. The $2 is a list of items all in the same word.

WORD — Specifies the address to be read. The $3 is the address.

EXPECT — Specifies the expected results in a 24-bit word. The $4 is any arithmetic expression which expresses the expected result.

OPTIONS — Specifies the mode, parity, and store timing during the read (ML) instruction. The ML instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor. The $5 is a list of options. Table A is a list of options for the MS and ML instructions.

KCODE — Specifies the K-code to be used during the ML instruction. It does not have to be the current-test K-code. The $6 is the K-code.

MASK — Specifies the mask to be used. If not specified with WORD, a mask of all 1s is used. If not specified with ITEM(S), the mask of the item(s) is used. The $7 is any arithmetic expression which expresses the mask.

BUS — Specifies what GCP results to look at: $8 is ACT — Active bus, first GCP or STB — Standby bus, second GCP.

Example:

```
STRDGCP_ WORD(MU1DRW00),EXPECT(0*I(MU1DWVNCLK1)),OPTIONS(R,M,C),
ME       MASK(M(MU1DWVNCCK1))
```

This statement executes two GCPs and sets the ML (read) options to R, M, and C, then reads location MU1DRW00, ignoring the results. This is then followed by another GCP. A GCP is executed and the reply is masked through the MU1DWVNCLK1 item and compared to 0*I(MU1DWVNCLK1).

STRDGCP2

4.214 The description of the STRDGCP2 statement includes:

Function:

The STRDGCP2 statement executes an abnormal read to the store ignoring the results followed by a GCP to check the effects of the read. The associated macro routine uses the maintenance (ML) load instruction which gives the user complete control over the mode, parity, and store timing pulses. The macro is, briefly, expanded as follows:

GCP

GCP

ML instruction

GCP

GCP.

Format:

```
          ITEM($1)
STRDGCP2  ITEMS($2),EXPECT($4),OPTIONS($5),KCODE($6),MASK($7),BUS($8),STATE($9)
          WORD($3)
```

Characteristics of Parameters:

ITEM — Specifies the address (bits 0 through 15) to be read (bits 0 through 15), associated with the address (bits 0 through 15) of its layout. The $1 is an item name.

ITEMS — Specifies the address (bits 0 through 15) to be read (bits 0 through 15), associated with the address (bits 0 through 15) of its layout. The $2 is a list of items all in the same word.

WORD — Specifies the address (bits 0 through 15) to be read (bits 0 through 15). The $3 is the address.

EXPECT — Specifies the expected results in a 24-bit word. The $4 is any arithmetic expression which expresses the expected result.

OPTIONS — Specifies the mode, parity, and store timing during the read (ML) instruction. A delay is taken after the ML instruction if keyword DELAY is specified. The two GCP instructions prior to the ML instruction are not executed if keyword NGCP is specified. Figure 1 shows the list of options for the ML instruction.

KCODE — Specifies the K-code to be used during the ML instruction. It does not have to be the current-test K-code. The $6 is the K-code.

MASK — Specifies the mask to be used. If not specified with WORD, a mask of all ones is used. If not specified with ITEM(S), the mask of the item(s) is used. The $7 is any arithmetic expression which expresses the mask.

BUS — Specifies what GCP results to look at: $8 is ACT—active bus, first GCP or STB—standby bus, second GCP.

STATE — Specifies the state of the maintenance (MTCE) flip-flop during the ML instruction. The $9 is one of the following:

MTCER — Reset MTCE

MTCES — Keep MTCE set.

If not specified, $9 is MTCES. If MTCER is specified, an MS instruction is executed prior to the ML instruction to reset the MTCE flip-flop.

Example:

```
STRDGCP2 WORD(MU1DRW00),EXPECT(0),OPTIONS(R,M,C),
         MASK(M(MU1DWVNCLK1))
```

This statement executes two GCP instructions and sets the ML (read) options to R, M, and C, then reads location MU1DRW00, ignoring the results. A GCP is then executed and the reply is masked through the MU1DWVNCLK1 item and compared to 0. This is then followed by another GCP instruction.

STRDGCP3

4.215 The description of the STRDGCP3 statement includes:

Function:

The store read and GCP macro executes an abnormal read ignoring the results followed by a GCP to check the results of the read. The general description of the macro would appear as follows:

GCP

GCP

ML instruction

GCP

GCP.

Format:

```
          ITEM($1)
STRDCP3   ITEMS($2),EXPECT($4),OPTIONS($5),KCODE($6),MASK($7),BUS($8),STATE($9)
          WORD($3)
```

Characteristics of Parameters:

ITEM — Specifies the address (bits 0 through 17) to be read associated with the address of the item layout. The $1 is any Datapool-defined item name.

ITEMS — Specifies the address (bit 0 through 17) to be read associated with the address of the items layout. The $2 is any Datapool-defined item names in the same word.

WORD — Specifies the address (bits 0 through 17) to be read. The $3 is any arithmetic expression which represents the address.

EXPECT — Specifies the expected result of the GCP in a 24-bit word. The $4 is any valid arithmetic expression which represents the expect value.

OPTIONS — Specifies the mode, parity, and store timing during the read (ML instruction). If DELAY is specified, a delay of 25 microseconds is taken after the ML instruction. If NGCP is specified, the two GCPs before the ML are not executed. Figure 3 gives a list of options.

KCODE — Specifies the K-code to be used during the ML instruction. If not specified, the current K-code will be used. The $6 is the K-code.

MASK — Specifies the mask to be used for the GCP read. If not specified with the WORD parameter, a mask of all ones is used. If not specified with the ITEM or ITEMS parameters, the mask of the ITEM or ITEMS is used. The $7 is any valid arithmetic expression which represents the mask value.

STATE — Specifies the state of the MTCE flip-flop during the ML instruction. The $9 is one of the following:

MTCER — Reset MTCE flip-flop

MTCES — Keep MTCE set.

If not specified, MTCES is assumed.

If MTCER is specified, an MS instruction is executed prior to the ML to reset the MTCE flip-flop.

Example:

```
STRDGCP3 WORD(MS1DRW00),EXPECT(0*I(MS1GCPSASWF1)),
          OPTIONS(R,C,M),MASK(M(MS1GCPSASWF1))
```

This statement will first execute two GCP instructions. The ML options are set to R, C, and M. Location MS1DRW00 is read and the results are ignored. A GCP instruction is executed and the reply is masked by MS1GCPSASWF1 and compared against the expect, 0*I(MS1GCPSASWF1). A second GCP is then executed.

STRDUPDN

4.216 The description of the STRDUPDN statement includes:

Function:

The STRDUPDN statement is the second half of a memory test in which all addresses are verified. The data in each location is equal to its address or its address complement. The verification either starts at address octal 0 and runs through address octal 177777 (up direction) or starts at address octal 177777 and runs through address octal 0 (down direction).

Format:

STRDUPDN DIRCTN($1),OPTN($2)

Characteristics of Parameters:

DIRCTN — Specifies the direction (up or down) of the read. The $1 is up or down.

OPTN — If used, specified data equals the complement of address. The $2 is C. If not specified, the data equals the address.

Example:

STRDUPDN DIRCTN(UP),OPTN(C)

This statement verifies the complement of the address starting at octal 0 and running through octal 177777.

STRD512

4.217 The description of the STRD512 statement includes:

Function:

The STRD512 statement is the second half of a short version of the regular checkerboard memory tests. The statement verifies the contents of 512 locations using a special algorithm where the combination of addresses used check most of the current drivers, switches, bridge rails, and diode matrices of the store.

Format:

STRD512 START($1),ADD1($2),ADD2($3),ADD3($4),COUNT($5),EXPECT($6)

Characteristics of Parameters:

START — Specifies the starting address to be used by the algorithm. The $1 is the address.

ADD1,ADD2,ADD3 — Specifies the address increments. The $2, $3, $4 are increments used by the algorithm.

COUNT — Specifies the total number of addresses to be read. The $5 is any arithmetic expression which expresses the total number of addresses.

EXPECT — Specifies the expected results in a 24-bit word. The $6 is any arithmetic expression which expresses the expected results. The $6 may also be PWR to indicate power-up K-code.

Example:

```
STRD512_START(1MUDACC1STT),ADD1(1MUDACCADD1),ADD2(1MUDACCADD2),
ME      ADD3(1MUDACCADD3),COUNT(1MUDACCNT),EXPECT(1DGACT01)
```

This statement checks most of the current drivers, switches, bridge rails, and diode matrices of the store. It starts at address 1MUDACC1STT and reads a total number of addresses equal to the value of 1MUDACCNT. The addresses to be read are calculated by the algorithm using 1MUDACCADD1, 1MUDACCADD2, and 1MUDACCADD3 as increments. The results are compared to the value of 1DGACT01.

STREAD

4.218 The description of the STREAD statement includes:

Function:

The STREAD normal read statement reduces to a STAREAD macro with the R bit set. The R bit is an option which specifies the mode of the ML instruction used by this macro. Refer to Table A.

Format:

```
        ITEM($1) EXPECT($4)
STREAD  ITEM($2),NOSTORE,MASK($9)
        WORD($3)
```

Characteristics of Parameters:

WORD — Specifies the address of the read. The $3 is the address.

ITEM — Specifies the address of the read which is associated with a Datapool layout. The $1 is an item name.

ITEMS — Specifies the address of the read which is associated with a Datapool layout. The $2 is a list of items all in the same word.

EXPECT — Specifies the expected results in a 24-bit word. The $4 is any arithmetic expression which expresses the expected results.

NOSTORE — Ignore results of the read, keeping the CRI flip-flop set.

MASK — Specifies the mask to be used. If not specified with WORD, a mask of all 1s is used. If not specified with ITEM(S), the mask of the item(s) is used. The $9 is any arithmetic expression which expresses the mask.

Example:

```
STREAD WORD(MU1DRW11),EXPECT(1MUDDATA1S)
```

Reduces to:

```
STREAD WORD(MU1DRW11),EXPECT(1MUDDATA1S),
       OPTIONS(R)
```

STRUPDN2

4.219 The description of the STRUPDN2 statement includes:

Function:

The STRUPDN2 statement is the second half of a memory test in which all addresses are verified. The data in each location is equal to its address or its address complement. The verification either starts at address 0 and runs through address octal 177777 (up direction) 0 or starts at address octal 177777 and runs through address 0 (down direction). The data parity bits are also checked for correctness. All data failures are shown by the store histogram printout. Segment breaks occur automatically within the associated macro routine after a predetermined number of addresses have been verified.

Format:

STRUPDN2 DIRCTN($1),OPTN($2)

Characteristics of Parameters:

DIRCTN — Specifies the direction (up or down) of the read. The $1 is up or down.

OPTN — If specified, specifies data equals the complement of address. The $2 is C. If not specified, the data equals the address.

Example:

STRUPDN2 DIRCTN(UP),OPTN(C)

This statement verifies the complement of the address starting at octal 0 and running thorough octal 177777.

STSLAVE

4.220 The description of the STSLAVE statement includes:

Function:

The STSLAVE statement causes a faulty store to be used as a duplicate (SLAVE) of some in-service store. This statement uses the parameters of the diagnosed DGN command.

Format:

DGN:PS $1:PH 94;MARG $2,SLTIM $3,KCODE $4!

Characteristics of Parameters:

PS — Specifies program store to be slaved, which must not be duplicated. This may also be call store. The $1 is the member number of the store.

PH — Specifies phase 94 to be run, which contains the STSLAVE statement.

MARG — Specifies the threshold of the store to be set during the slave time. The $2 is HIGH or LOW; the default is normal threshold.

SLTIM — Specifies the slave time. The $3 is the time in minutes.

KCODE — Specifies the K-code the store is to be slave to. The $4 is the K-code in octal.

Example:

DGN:PS 2:PH 94,SLTIM 2,KCODE 20!

This statement will specify phase 94 to be run on program store 2. The STSLAVE statement being called in this phase will thereupon set PS 2 to K-code octal 20. The MTCE flip-flop of the store is then reset while CRI is set. The RO flip-flop is set to the active bus. The contents of the in-service store having chosen K-code are next copied into PS 2. The threshold of PS 2 is then set to the level requested, in this case, NORMAL. The PS 2 is left in this special mode to be exercised by the active system for duration of 2 minutes. At the end of this waiting period, the MTCE flip-flop of the test store is set while CRI is reset. The contents of PS 2 and the good store it is slaving are then compared to look for discrepancies. The results are interpreted by the pattern analyzer program, and are also displayed in raw data format and summarized format (Histogram).

STSLAVE2

4.221 The description of the STSLAVE2 statement includes:

Function:

The STSLAVE2 statement causes a faulty store to be used as a duplicate (SLAVE) of some in-service store. The statement has no parameters, it uses parameter information from the diagnose (DGN) input message.

Format:

DGN:PS $1:PH 94,SLTIM $2,KCODE $3!

Characteristics of Parameters:

PS — Specifies program store $1 to be slaved. This may also be call store. The $1 being the member number of the store.

PH — Specifies phase 94 to be run, which contains the STSLAVE2 statement.

SLTIM — Specifies the slave time. The $2 is the time in minutes.

KCODE — Specifies the K-code the store is to be slaved as. The $3 is the K-code in octal. If the K-code specified is duplicated or an on-line store does not have the specified K-code, the test store will not be slaved.

Example:

DGN:CS 2:PH 94,SLTIM 2,KCODE 1!

This input message will specify phase 94 to be run on call store 2. The STSLAVE2 statement being called in phase 94 will set CS 2 to K-code 1. The MTCE flip-flop of the store is then reset while CRI is set. The RO flip-flop is set to the active bus. The contents of the in-service store having the chosen K-code are next copied into CS 2. The TWF flip-flop is then set to allow refresh data parity check. The CS 2 is then left in this special mode to be exercised by the active system for a duration of 2 minutes. At the end of this waiting period, the MTCE flip-flop of the test store is set while CRI is reset. The contents of CS 2 and the good store it is slaving are then compared to look for discrepancies. All data failures are shown by the store histogram printout.

STSLAVE3

4.222 The description of the STSLAVE3 statement includes:

Function:

The store slave macro is used to duplicate or slave a faulty store to an in-service store. This macro can be used to slave to a store with the K-code specified manually through the input message or automatically by the parameters on the macro call.

Format:

STSLAVE3 AUTO,KCODE

Characteristics of Parameters:

AUTO — Specifies that the slave test be run automatically. The K-code information will come from the data table.

KCODE — Specifies that the routine should locate a nonduplicated store to slave to. If not specified, the K-code is assumed to be in the DBT.

Example:

(1) STSLAVE3 KCODE, AUTO

This macro call will search for a nonduplicated store to slave to. If a store is found, the store under test will be set to the valid K-code. The MTCE flip-flop is reset; the CRI flip-flop is set; and the RO flip-flop is set to the active bus. The contents of the in-service store with the valid K-code is copied into the test store. The refresh data parity check (RDPC) flip-flop is set to allow data parity check on refresh during the slave time. The store is slaved for 10 seconds. At the end of the slave period, the MTCE flip-flop of the test store is set and CRI is reset. The test store is then checked for parity failures by using the refresh data parity check circuitry. All discrepancies are recorded and analyzed by the pattern analysis routine.

(2) DGN CS 2:PH 94,SLTIM 2, KCODE 1! (at terminal)
STSLAVE3 (in PH 94)

The input message will cause PH 94 of the diagnostic to be run on call store member number 2. The STSLAVE3 statement will set CS 2 to K-code 1. The MTCE flip-flop of CS 2 is reset; the CRI flip-flop is set; and RO is set to the active bus. The contents of the in-service store with the K-code of 1 will be copied into CS 2. The RDPC is set and the store is kept in this configuration for 2 minutes. At the end of slave time, the MTCE flip-flop in CS 2 is set and CRI is reset. The CS 2 is then checked for parity failures. All discrepancies are recorded and analyzed by the pattern analysis routine.

STSLWRD2

4.223 The description of the STSLWRD2 statement includes:

Function:

The STSLWRD2 store slow read statement reads all memory locations. The associated macro routine uses the maintenance load (ML) instruction to perform the memory reads. A memory read is executed every fourth store cycle. The results of the read are ignored for the CRI flip-flop is not reset. This statement will help find faults which occur when the store is operating at slow speed. Segment breaks occur automatically within the associated macro routing after a predetermined number of addresses have been read.

Format:

STSLWRD2

Characteristics of Parameters:

This statement has no parameters.

Example:

STSLWRD2

This store slow read statement reads all memory locations. The associated macro routine starts at address 0 and runs through octal 177777.

STSNAP

4.224 The description of the STSNAP statement includes:

Function:

The STSNAP snapshot control read statement reduces to three macros:

(1) STAWRITE with the W, M, and C bits set

(2) STAREAD with options set and the NOSTORE parameter specified

(3) STAREAD with the R, M, and C bits set.

The W, R, M, and C bits are options which specify the mode of the ML/MS instruction used by this macro. Refer to Table A.

Format:

```
        ITEM($1)
STSNAP ITEMS($2),AT($4),EXPECT($4),MASK($6)
        WORD($3)
```

Characteristics of Parameters:

WORD — Specifies the address of the read. The $3 is the address.

ITEM — Specifies the address of the read which is associated with a Datapool layout. The $1 is an item name.

ITEMS — Specifies the address of the read which is associated with a Datapool layout. The $2 is a list of items all in the same word.

AT — Specifies the data to be written by the STAWRITE macro. The $4 is any arithmetic expression which expresses the data.

EXPECT — Specifies the expected results in a 24-bit word, used by the second STAREAD macro. The $5 is any arithmetic expression which expresses the expected results.

MASK — Specifies the mask to be used. If not specified with WORD, a mask of all 1s is used. If not specified with ITEM(S), the mask of the item(s) is used. The $6 is any arithmetic expression which expresses the mask.

Note: Also see the STAWRITE and STAREAD descriptions.

Example:

```
STSNAP ITEM(MU1DRING1),AT(1MUDSNP08),
       EXPECT(M(MU1DRC091)),MASK(1MUDRCM91)
```

Reduces to:

```
STAWRITE WORD(MU1DRW10),DATA(1MUDSNP08*I(MU1DWTRPREG)),
         OPTIONS(W,M,C)
STAREAD ITEM(MU1DRING1),NOSTORE,OPTIONS(M,C)
STAREAD WORD(MU1DRW11)EXPECT(M(MU1DRC091)),
         OPTIONS(R,M,C),MASK(1MUDRCM91)
```

STSTATUS

4.225 The description of the STSTATUS statement includes:

Function:

The STSTATUS statement provides the means of executing a pulse read of the store. The routine executes two or four GCPs.

Format:

```
                ITEM($1) NOSTORE
STSTATUS ITEMS($2),EXPECT($3),BUS($4),ADDGCP
```

Characteristics of Parameters:

ITEM — Specifies the mask. The $1 is an item name.

ITEMS — Specifies the mask. The $2 is a list of items all in the same word.

EXPECT — Specifies the expected results in a 24-bit word. The $3 is any arithmetic expression which expresses the expected results. The $3 may also be PWR to indicate power-up K-code.

NOSTORE — Indicated to ignore the results of the GCPs.

BUS — If specified, specifies which GCP results are to be looked at. The $4 is ACT—active bus (first GCP) or STB—standby bus (second GCP).

ADDGCP — If specified, specifies that two additional GCPs will be executed (for a total of four GCPs) prior to the normal two GCPs. If not specified, only two GCPs are executed.

Example:

STSTATUS ITEM(MU1DWMTCE1),EXPECT(0),BUS(STB)

This statement executes two pulse reads of the store and compares the result of the second GCP to zero through the mask of the MU1DWMTCE1 item.

STSTATUS3

4.226 The description of the STSTATUS3 statement includes:

Function:

The store status macro is used to pulse read the store. The routine executes two or four GCPs.

Format:

```
                ITEM($1) NOSTORE
STSTATUS3 ITEM($2),EXPECT($3),BUS($4),ADDGCP
```

Characteristics of Parameters:

ITEM — Specifies the mask for the GCP read. The $1 is any Datapool-defined item.

ITEMS — Specifies the mask for the GCP read. The $2 is a list of Datapool-defined items in the same word.

EXPECT — Specifies the expected result of the GCP read in a 24-bit word. The $3 is any valid arithmetic expression which represents the expect value.

NOSTORE — Specifies that the result of the GCP is to be ignored.

BUS — Specifies which GCP results are to be looked at. The $4 is ACT for active bus (first GCP) or STB for standby bus (second GCP).

ADDGCP — Specifies that two additional GCPs will be executed prior to the two normal GCPs.

Example:

STSTATUS ITEM(MS1GCPMTCE1),EXPECT(1*I(MS1GCPMTCE1)), BUS(STB)

This statement will execute two GCP reads of the store and compare the results of the second GCP to 1*I(MS1GCPMTCE1) through a mask of MS1GCPMTCE1.

STTRAP

4.227 The description of the STTRAP statement includes:

Function:

The STTRAP trap control read statement reduces to three macros:

(1) STAWRITE with the W, M, and C bits set.

(2) A macro statement is executed which is specified by the EXECUTE parameter.

(3) STAREAD with R, M, and C bits set.

The W, R, M, and C bits are options which specify the mode of the ML/MS instructions used by this macro.

Format:

```
        ITEM($1)
STTRAP  ITEM($2),AT($4),EXECUTE($5),EXPECT($6),MASK($7)
        WORD($3)
```

Characteristics of Parameters:

WORD — Specifies the mask of the read if the mask parameter is not used. The $3 is the mask.

ITEM — Specifies the mask of the read which is associated with a Datapool layout. The $1 is an item name.

ITEMS — Specifies the mask of the read which is associated with a Datapool layout. The $2 is a list of items all in the same word.

AT — Specifies the data to be written by the STAWRITE macro. The $4 is any arithmetic expression which expresses the data.

EXECUTE — Specifies the next macro call to be executed.

EXPECT — Specifies the expected results in a 24-bit word, used by the STAREAD macro. The $6 is any arithmetic expression which expresses the expected results.

MASK — Specifies the mask to be used. If not specified with WORD, this mask is used. If not specified with ITEM(S), the mask of the item(s) is used. The $7 is any arithmetic expression which expresses the mask.

Note: Also see the STAWRITE and STAREAD descriptions.

Example:

```
STTRAP ITEMS(MU1DOD1),AT(1MUDTRP07),
       EXECUTE(STAREAD WORD(1MUDADD1S),NOSTORE,
       OPTIONS(R,M)),EXPECT(O(0))
```

Reduces to:

```
STAWRITE WORD(MU1DRW10),
         DATA(1MUDFRHTRAP/IMUDTRP07*I(MU1DWTRPREG)),
         OPTIONS(W,M,C)

STAREAD WORD(1MUDADD1S),NOSTORE,OPTIONS(R,M)
STREAD WORD(MU1DRW11),EXPECT(O(0)),OPTIONS(R,M,C)
       MASK(M(MU1DOD1))
```

STVERMEM

4.228 The description of the STVERMEM statement includes:

Function:

The STVERMEM statement is the second half of a checkboard memory test. All memory locations are read and compared with the expected result.

Format:

STVERMEM EXPECT($1)

Characteristics of Parameters:

EXPECT — Specifies the expected result in a 24-bit word. The $1 is any arithmetic expression which expresses the expected result.

Example:

STVERMEM EXPECT (1DGALT10)

This statement reads all memory locations and compares them to the value of 1DGALT10.

STVRMEMS

4.229 The description of the STVRMEMS statement includes:

Function:

The STVRMEMS statement is the second half of a memory test. The associated macro routine uses the maintenance load (ML) instruction to read all memory locations expecting a specified data pattern. The data parity bits are also checked for correctness. All data failures are shown by the store histogram printout. Segment breaks occur automatically within the routine after a predetermined number of addresses have been verified.

Format:

STVRMEM2 EXPECT ($1)

Characteristics of Parameters:

EXPECT — Specifies the expected result in a 24-bit word. The $1 is any arithmetic expression which expresses the expected result.

Example:

STVRMEM2 EXPECT(1DG_ONES)

This statement reads all memory locations and compares them to the value of 1DG_ONES.

STVRMEM2

4.230 The description of the STVRMEM2 statement includes:

Function:

The STVRMEM2 statement is the second half of the checkerboard/column bar type memory test. All memory locations are read and compared with the expected results of a checkerboard or column bar data pattern. The data parity bits are also checked for correctness. All data failures are shown by the store histogram printout. Segment breaks occur automatically within the associated macro routine after a predetermined number of addresses have been verified.

Format:

STVRMEM2 EXPECT($1),CHECKER,COLUMN

Characteristics of Parameters:

EXPECT — Specifies the expected result in a 24-bit word. The $1 is an arithmetic expression which expresses a data pattern of alternate one-zero or zero-one.

CHECKER — Specifies that a checkerboard pattern is expected in all memory chips. Due to the structure of the memory (TI) chip, the expected data pattern is rotated after every 64th address but not after the 512th address. This parameter cannot be specified along with COLUMN.

COLUMN — Specifies that a column bar pattern is expected in all memory chips. Due to the structure of the memory (TI) chip, the expected data pattern is rotated after every 512th address. This parameter cannot be specified along with CHECKER.

Example:

STVRMEM2 EXPECT(1DGALT10),CHECKER

This statement reads all memory locations expecting a checkerboard data pattern.

STWRGCP

4.231 The description of the STWRGCP statement includes:

Function:

The STWRGCP statement executes an abnormal write to the store (MS instruction). The results of the write are checked by a GCP. The MS and GCP instructions are described in Section 254-280-020, Assembly Language—Description, 1A Processor.

Format:

```
        ITEM($1)
STWRGCP ITEMS($2),EXPECT($4),DATA($5),OPTIONS($6),MASK($7),KCODE($8),BUS($9),STATE($10)
        WORD($3)
```

Characteristics of Parameters:

WORD — Specifies the address to be written. The $3 is the address.

ITEM — Specifies the address to be written associated with its layout. The $1 is the item name.

ITEMS — Specifies the address to be written associated with its layout. The $2 is a list of items all in the same word.

EXPECT — Specifies the expected results in a 24-bit word. The $4 is any arithmetic expression which expresses the expected result.

DATA — Specifies the data to be written to the store. The $5 is any arithmetic expression which expresses the data.

OPTIONS — Specifies the mode, parity, and store timing during the write (MS) instruction. The MS instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor. The $6 is a list of options. Table A is a list of options for the MS and ML instructions.

MASK — Specifies the mask to be used. If not specified with WORD, a mask of all 1s is used. If not specified with ITEM(S), the mask of the item(s) is used. The $7 is any arithmetic expression which expresses the mask.

KCODE — Specifies the K-code to be used. If not specified, current-test KCODE is used. The $8 is the K-code.

BUS — Specifies the GCP results: $9 is ACT—active bus, first GCP. STB—standby bus, second GCP.

STATE — If specified, specifies the state of the maintenance (MTCE) flip-flops. The $10 is one of the following:

MTCER — Reset MTCE

MTCES — Set MTCE

If not specified, $10 is MTCES.

Example:

```
STWRGCP_ WORD(MU1DRW00),EXPECT(M(MU1DWVNSYNC1)),DATA(0),
ME       OPTIONS(C,M,W,IST(4T6),IPKA),MASK(M(MU1DWVNSYNC1))
```

This statement executes two GCPs and sets the MS (write) options to C, M, W, ITS(4T6), and IPKA. Then the location MU1DRW00 is written with zeros. A GCP is executed and the reply is masked through the mask of the MU1DWVNSYNC1 item and compared with the mask of the MU1DWVNSYNC1 item. This is then followed by a second GCP.

STWRGCP2

4.232 The description of the STWRGCP2 statement includes:

Function:

The STWRGCP2 statement executes an abnormal write to the store. The results of the write are checked by a GCP. The associated macro routine uses the maintenance store (MS) instruction which gives the user complete control over the mode, parity, and store timing pulses during the write. The macro is, briefly, expanded as follows:

GCP

GCP

MS instruction

GCP

GCP.

Format:

```
          ITEM($1)
STWRGCP2 ITEMS($2),EXPECT($4),DATA($5),OPTIONS($6),MASK($7),KCODE($8),BUS($9),STATE($10)
          WORD($3)
```

Characteristics of Parameters:

WORD — Specifies the address (bits 0 through 15) to be written. The $3 is the address.

ITEM — Specifies the address (bits 0 through 15) to be written associated with its layout. The $1 is the item name.

ITEMS — Specifies the address (bits 0 through 15) to be written associated with its layout. The $2 is a list of items all in the same word.

EXPECT — Specifies the expected results in a 24-bit word. The $4 is any arithmetic expression which expresses the expected result.

DATA — Specifies the data to be written to the store. The $5 is any arithmetic expression which expresses the data.

OPTIONS — Specifies the mode, parity, and store timing during the write (MS) instruction. A delay is taken after the MS instruction if keyword DELAY is specified. The two GCP instructions prior to the MS instruction are not executed if keyword NGCP is specified. Figure 1 shows the list of options for the MS instruction. The $6 is a list of options.

MASK — Specifies the mask to be used. If not specified with WORD, a mask of all ones is used. If not specified with ITEM(S), the mask of the item(s) is used. The $7 is any arithmetic expression which expresses the mask.

KCODE — Specifies the K-code to be used. If not specified, current-test KCODE is used. The $8 is the K-code.

BUS — Specifies what GCP results to look at: $9 is ACT—active bus, first GCP, STB—standby bus, second GCP.

STATE — Specifies the state of the maintenance (MTCE) flip-flop during the MS instruction. The $10 is one of the following:

MTCER — Reset MTCE

MTCES — Keep MTCE set.

If not specified, $9 is MTCES. If MTCER is specified an MS instruction is executed prior to the normal MS instruction to reset the MTCE flip-flop.

Example:

```
STWRGCP2 WORD(MU1DRW00),EXPECT(M(MU1DWVNSYNC1)),DATA(0),
         OPTIONS(C,M,W,IST(4T6),IPKA),MASK(M(MU1DWVNSYNC1))
```

This statement executes two GCPs and sets the MS (write) options to C, M, W, IST(4T6), and IPKA. Then the location MU1DRW00 is written with zeros. A GCP is executed and the reply is masked through the mask of the MU1DWVNSYNC1 item and compared with the mask of the MU1DWVNSYNC1 item. This is then followed by a second GCP.

STWRGCP3

4.233 The description of the STWRGCP3 statement includes:

Function:

The store write and GCP macro executes an abnormal write to the store followed by a GCP read to verify the write operation. A general expansion of the macro would appear as follows:

GCP

GCP

MS instruction

GCP

GCP.

Format:

```
          ITEM($1)
STWRGCP3  ITEMS($2),EXPECT($4),DATA($5)OPTIONS($6),
          WORD($3)

          ---------------------------------------
          MASK($7),KCODE($8),BUS($9),STATE($10)
```

Characteristics of Parameters:

ITEM — Specifies the address (bits 0 through 17) to be written associated with the address of the item layout. The $1 is any Datapool-defined item name.

ITEMS — Specifies the address (bits 0 through 17) to be written associated with the address of the items layout. The $2 is any Datapool-defined item names in the same word.

WORD — Specifies the address (bits 0 through 17) to be read. The $3 is any valid arithmetic expression which represents the expect value.

EXPECT — Specifies the expected result of the GCP read in a 24-bit word. The $4 is any valid arithmetic expression which represents the expect value.

DATA — Specifies the data to be written to the store on the MS instruction. The $5 is any valid arithmetic expression which represents the data.

OPTIONS — Specifies the mode, parity, and store timing during the write (MS instruction). If DELAY is specified, a delay of 25 microseconds is taken after the MS instruction. If NGCP is specified, the two GCPs before the MS are not executed. Figure 3 provides a list of options.

MASK — Specifies the mask to be used for the GCP read. If not specified with the WORD parameter, a mask of all ones is used. If not specified with the ITEM or ITEMS parameters, the mask of the ITEM or ITEMS is used. The $7 is any valid arithmetic expression which represents the mask value.

KCODE — Specifies the K-code to be used during the MS instruction. If not specified, the current K-code will be used. The $8 is the K-code.

BUS — Specifies the bus to be used for the GCP results. ACT specifies the active bus (first GCP result) and STB specifies the standby bus or second GCP result.

STATE — Specifies the state of the MTCE flip-flop during the MS instruction. The $10 is one of the following:

MTCER — Reset MTCE flip-flop.

MTCES — Keep MTCE set.

If not specified, MTCES is assumed.

If MTCER is specified, an MS instruction is executed prior to the normal MS to reset the MTCE flip-flop.

Example:

```
STWRGCP3 WORD(MS1DRW00),EXPECT(1*I(MS1GCPTMD1)),
         DATA(0),OPTIONS(M,IST(5T7),IPKA),
         MASK(M(MS1GCPTMD1))
```

This macro call will execute two GCP instructions. The MS options will be set to M,IST(5T7) and IPKA. Location MS1DRW00 is written with data of 0. A GCP is executed and the reply is masked through MS1GCPTMD1 and compared with 1*I(MS1GCPTMD1). A second GCP is then executed.

STWRITE

4.234 The description of the STWRITE statement includes:

Function:

The STWRITE normal write statement reduces to a STAWRITE with the W bit set. The W bit is an option which specifies the mode of the MS instruction used by this macro. Refer to Table A.

Format:

```
STWRITE WORD($1),DATA($2)
```

Characteristics of Parameters:

WORD — Specifies the address to be written into. The $1 is the address.

DATA — Specifies the data to be written. The $2 is any arithmetic expression which expresses the data.

Example:

```
STWRITE WORD(MU1DRW11),DATA(1MUDPARADD2)
```

Reduces to:

```
STAWRITE WORD(MU1DRW11),DATA(1MUDPARADD2),
         OPTIONS(W)
```

STWRMEMS

4.235 The description of the STWRMEMS statement includes:

Function:

The STWRMEMS statement is the first half of a memory test. The associated macro routine uses the maintenance store (MS) instruction to write all memory locations with specified data. Segment breaks occur automatically within the routine after a predetermined number of addresses have been written.

Format:

STWRMEMS DATA($1)

Characteristics of Parameters:

DATA — Specifies the data to be written. The $1 is any arithmetic expression which expresses the data.

Example:

STWRMEMS DATA(1DG_ONES)

This statement writes all memory locations with the value of 1DG_ONES.

STWRMEM2

4.236 The description of the STWRMEM2 includes:

Function:

The STWRMEM2 statement is the first half of the checkerboard/column bar type memory test. All memory locations are written using the maintenance store (MS) instruction. Segment breaks occur automatically within the associated macro routine after a predetermined number of addresses have been written. One module is written at a time.

Format:

STWRMEM2 DATA($1),CHECKER,COLUMN

Characteristics of Parameters:

DATA — Specifies the data to be written. The $1 is an arithmetic expression which expresses a data pattern of alternate one-zero or zero-one.

CHECKER — Specifies that a checkerboard pattern be written into all memory chips. Due to the structure of the memory (TI) chip, the data is rotated after every 64th address but not after the 512th address. The COLUMN parameter cannot be specified along with CHECKER.

COLUMN — Specifies that a column bar pattern be written into all memory chips. Due to the structure of the memory (TI) chip the data is rotated after every 512th address. The CHECKER parameter cannot be specified along with COLUMN.

Example:

STWRMEM2 DATA(1DGALT10),CHECKER

This statement writes all memory chips with a checkerboard pattern. The data specified 1DGALT10 is equal to an alternate one-zero pattern, O(52525252).

STWRMEM3

4.237 The description of the STWRMEM3 statement includes:

Function:

The store write memory macro is used to write all 256K of memory to a constant data pattern.

Format:

STWRMEM3 DATA($1)

Characteristics of Parameters:

DATA — Specifies the constant data pattern to be written into all 256K addresses. The $1 is any valid arithmetic expression which represents the data.

Example:

STWRMEM3 DATA(1DGALT01)

This macro call will write all 256K addresses to a value of 1DGALT01. The routine will automatically take segment breaks.

STWRNAM2

4.238 The description of the STWRNAM2 statement includes:

Function:

The STWRNAM2 statement provides the means of altering the internal K-code name register of the store. All following test macros will use this K-code name as the "current test K-code" unless another STWRNAM2 macro is encountered. A common K-code can be specified for program store and call store or separate K-codes for program store and call store. The macro expands as a maintenance store (MS) instruction to write the new test K-code. The remaining STATUS bits MTCE, CRI, RO, ANS0, and ANS1 remain unchanged.

Format:

(1) STWRNAM2 KCODE($1)
or
(2) STWRNAM2 PSKCODE($2),CSKCODE($3)

Characteristics of Parameters:

KCODE — Specifies the new test K-code for program store and call store. The $1 is the K-code; it may be a number or Datapool-defined symbol. If $1 is PWR, power-up K-code is used, octal 36 for program store and octal 17 for call store.

PSKCODE — Specifies the new test K-code for program store. The $2 is the K-code; it may be a number or Datapool-defined symbol.

CSKCODE — Specifies the new test K-code for call store. The $3 is the K-code; it may be a number or Datapool-defined symbol.

Example:

STWRNAM2 KCODE(1MUDTSTNAME)

This statement executes a control write (MS) to change the K-code of the unit under test to the value of 1MUDTSTNAME.

STWRNAM3

4.239 The description of the STWRNAM3 statement includes:

Function:

The store write name macro is used to change the internal K-code name of the store. Once the name has been written, all the following macros will use this name as the "current K-code" value. Separate values can be specified for both call store and program store or the same name can be used for both. The K-code is changed by doing an MS instruction to write the new K-code name. All other status bits remain unchanged. The macro can also be used to specify that the K-code not be restored to power-up K-code over segment boundaries. This is done to prevent getting parity failures when writing the entire store. The macro also can provide the capability of storing a K-code name in a scratch word and using the scratch word to write the K-code name. This feature is used with the error analysis routine.

Formats:

STWRNAM3 KCODE($1),TSTKCSEG,SWORD($2)

STWRNAM3 PSKCODE($3),CSKCODE($4),TSTKCSEG,SWORD($2)

Characteristics of Parameters:

KCODE — Specifies the new K-code name for both program store or call store. The $1 is a number or valid Datapool-defined symbol. If $1 is PWR, the power-up K-code for program store (K-code 34) and for call store (K-code 14) will be used.

PSKCODE — Specifies the new K-code name to be used for the program store. The $3 is a number or valid Datapool-defined symbol.

CSKCODE — Specifies the new K-code name to be used for the call store. The $4 is a number or valid Datapool-defined symbol.

TSTKCSEG — Specifies that the K-code will not be changed over the segment break.

SWORD — Specifies a scratch word address which contains the K-code name which is to be written into the store. The $2 is a valid Datapool-defined address.

Example:

STWRNAM3 KCODE(1MSDTSTNAME)

This macro will cause an MS instruction to be executed to change the K-code name to the value of 1MSDTSTNAME.

STWRSTAT

4.240 The description of the STWRSTAT statement includes:

Function:

The STWRSTAT statement changes the status of the store, checks the results by a GCP, and then puts the store back to its original status. The GCP instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor.

Format:

```
          ITEM($1)  NOSTORE
STWRSTAT ITEMS($2),EXPECT($4),DATA($5),OPTIONS($6),MASK($7),STATE($8)
          WORD($3)
```

Characteristics of Parameters:

WORD — Specifies the address to be written into. The $3 is the address.

ITEM — Symbolic reference of the address to be written into. The $1 is the item name.

ITEMS — Symbolic reference of the address to be written into. The $2 is a list of items all in the same word.

EXPECT — Specifies the expected results in a 24-bit word. The $4 is any arithmetic expression which expresses the expected results. The $4 may also be PWR to indicate power-up K-code.

NOSTORE — Indicates to ignore the results of the GCP.

DATA — Specifies the data to be written to the store. The $5 is any arithmetic expression which expresses the data.

OPTIONS — Specifies the mode, parity, and store timing during the write (MS) instruction. The MS instruction is described in Section 254-280-020, Assembly Language—Description, 1A Processor. The $6 is a list of options. Table A is a list of options for the MS and ML instructions.

MASK — Specifies the mask to be used. If not specified with WORD, a mask of all 1s is used. If not specified with ITEM(S), the mask of the item(s) is used. The $7 is any arithmetic expression which expresses the mask.

STATE — Specifies the states of the communications reply inhibit (CRI) and maintenance (MTCE) flip-flops. The $8 is two of the following:

CRIR — Reset CRI

CRIS — Set CRI

MTCER — Reset MTCE

MTCES — Set MTCE.

If not specified, $8 is (CRIR,MTCES).

Example:

```
STWRSTAT_ WORD(MU1DRW10),EXPECT(0*I(MU1DWVSASWF1)),DATA(1MUDINPR),
ME        OPTIONS(W,C,M,NGCP),MASK(M(MU1DWVSASWF1))
```

This statement sets the MS (write) options to W, C, and M. No GCP is executed before the location MU1DRW10 is written with 1MUDINPR. The GCP reply is masked through the mask of the MU1DWVSASWF1 item and compared to 0*(MU1DWVSASWF1).

STWRTMEM

4.241 The description of the STWRTMEM statement includes:

Function:

The STWRTMEM statement is the first half of a checkerboard memory test. All memory locations are written.

Format:

STWRTMEM DATA($1)

Characteristics of Parameters:

DATA — Specifies the data to be written. The $1 is any arithmetic expression which expresses the data.

Example:

STWRTMEM DATA(1DGALT10)

This statement writes all memory locations with the value of 1DGALT10.

STWRTNAM

4.242 The description of the STWRTNAM statement includes:

Function:

The STWRTNAM statement alters the K-code name register of the store.

Format:

STWRTNAM KCODE($1)

Characteristics of Parameters:

KCODE — Specifies the new test K-code. The $1 is the K-code; it may be a number or Datapool-defined.

Example:

STWRTNAM KCODE(1MUDTSTNAME)

This statement changes the K-code of the unit under test to the value of 1MUDSTNAME.

STWRTRTF

4.243 The description of the STWRTRTF statement includes:

Function:

The STWRTRTF statement provides the only means of altering the store answer flip-flops.

Format:

STWRTRTF ROUTE($1)

Characteristics of Parameters:

ROUTE — Specifies the state of the answer on bus 0 (ANS0) and answer on bus 1 (ANS1) flip-flops. The $1 is two of the following:

ANS0S — Set ANS0 flip-flop

ANS0R — Reset ANS0 flip-flop

ANS1S — Set ANS1 flip-flop

ANS1R — Reset ANS1 flip-flop.

If not specified, $1 is (ANS0R,ANS1R)

Example:

STWRTRTF ROUTE(ANS0R,ANS1R)

This statement resets the store answer flip-flops.

STWRUPDN

4.244 The description of the STWRUPDN statement includes:

Function:

The STWRUPDN statement is the first half of a memory test in which all addresses are written. The data written into each location is equal to its address or its address complement. The write either starts at address octal 0 and runs through address octal 177777 (up direction) or starts at address octal 177777 and runs through octal 0 (down direction).

Format:

STWRUPDN DIRCTN($1),OPTN($2)

Characteristics of Parameters:

DIRCTN — Specifies direction of write (up or down). The $1 is up or down.

OPTN — If specified, specifies data equals complement of address. The $2 is C. If not specified, data equals address.

Example:

STWRUPDN DIRCTN(UP),OPTN(C)

This statement writes the complement of the address starting at address octal 0 and running through octal 177777.

STWRWPM3

4.245 The description for the STWRWPM3 statement includes:

Function:

The store write protect memory macro is used to initialize the 1K write protect RAM to a constant value.

Format:

STWRWPM3 DATA($1)

Characteristics of Parameters:

DATA — Specifies the data to write into all 1K of write protect memory. The $1 must be either 0 or 1.

Example:

STWRWPM3 DATA(0)

This macro will cause all the 1K RAM to be written with data of 0.

STWR512

4.246 The description of the STWR512 statement includes:

Function:

The STWR512 statement is the first half of a short version of the regular checkerboard memory tests. The statement writes the contents of 512 memory locations using a special algorithm where the combination of the addresses used check most of the current drivers, switches, bridge rails, and diode matrices of the store.

Format:

STWR512_ START($1),ADD1($2),ADD2($3),ADD3($4),COUNT($5),DATA($6)

Characteristics of Parameters:

START — Specifies the starting address to be used by algorithm. The $1 is the address.

ADD1,ADD2,ADD3 — Specifies the address increments. The $2, $3, and $4 are increments used by the algorithm.

COUNT — Specifies the total number of addresses to be read. The $5 is any arithmetic expression which expresses the total number of addresses.

DATA — Specifies the data to be written. The $6 is any arithmetic expression which expresses the data.

Example:

```
STWR512_ START(1MUDACC1STT),ADD1(1MUDACCADD1),ADD2(1MUDACCADD2),
ME       ADD3(1MUDACCADD3),COUNT(1MUDACCNT),DATA(1MUDDATA1S)
```

This statement checks most of the current drivers, switches, bridge rails, and diode matrices of the store. It starts at address 1MUDACC1STT and writes a total number of addresses equal to the value of 1MUDACCNT. The addresses to be written are calculated by the algorithm using 1MUDACCADD1, 1MUDACCADD2, and 1MUDACCADD3 as increments. The results are compared to the value of 1MUDDATA1S.

STWSLOW3

4.247 The description of the STWSLOW3 statement includes:

Function:

The store write slow macro is used to control the SLOW flip-flop. If the slow flip-flop is set, the store will operate in the slow mode (1400 ns central control cycle time). If the slow flip-flop is reset, the store will operate fast (700 ns central control cycle time). All other status bits are unchanged by the MS operation used to change the SLOW flip-flop.

Format:

```
STWSLOW3 SPEED($1)
```

Characteristics of Parameters:

SPEED — Specifies the speed which the store will operate after the MS instruction is performed. The $1 is SLOW or FAST.

Example:

```
STWSLOW3 SPEED(FAST)
```

This macro call will cause the store's SLOW flip-flop to be reset. The store will then operate in the fast mode.

STWSTAT2

4.248 The description of the STWSTAT2 statement includes.

Function:

The STWSTAT2 statement changes the status of the store, checks the results by a GCP, and then puts the store back to its original status. The associated macro routine uses the maintenance store (MS) instruction which gives the user complete control over the mode, parity, and store timing pulses during the control write to the status of the store. The macro is, briefly, expanded as follows:

GCP

GCP

MS instruction change status

GCP

GCP

MS — instruction to write back the tested store's original status.

Format:

```
          ITEM(S1)  NOSTORE
STWSTAT2 ITEMS(S2),EXPECT(S4),DATA(S5),OPTIONS(S6),MASK(S7),STATE(S8)
          WORD(S3)
```

Characteristics of Parameters:

WORD — Specifies the address (bits 0 through 15) to be written into. The $3 is the address.

ITEM — Symbolic reference of the address (bits 0 through 15) to be written into. The $1 is the item name.

ITEMS — Symbolic reference of the address (bits 0 through 15) to be written into. The $2 is a list of items all in the same word.

EXPECT — Specifies the expected results in a 24-bit word. The $4 is any arithmetic expression which expresses the expected results. The $4 may also be PWR to indicate power-up K-code.

NOSTORE — Indicates to ignore the results of the GCP.

DATA — Specifies the data to be written to the store. The $5 is any arithmetic expression which expresses the data.

OPTIONS — Specifies the mode, parity, and store timing during the write (MS) instruction of the store status. A delay is taken after the MS instruction if keyword DELAY is specified. The two GCP instructions prior to the MS instruction are not executed if keyword NGCP is specified. The $6 is a list of options. Figure 1 shows the list of options for the MS instruction.

MASK — Specifies the mask to be used. If not specified with WORD, a mask of all ones is used. If not specified with ITEM(S), the mask of the item(s) is used. The $7 is any arithmetic expression which expresses the mask.

STATE — Specifies the states of the communications reply inhibit (CRI) and maintenance (MTCE) flip-flops. The $8 is two of the following:

CRIR — Reset CRI

CRIS — Keep CRI set

MTCER — Reset MTCE

MTCES — Keep MTCE set.

If not specified $8 is (MTCES, CRIS). If MTCER and/or CRIR is specified, an MS instruction is executed prior to the write of the store's status to reset the proper flip-flop(s).

Example:

```
STWSTAT2 WORD(MU1DRW0)1,EXPECT(0*I(MU1DWVCRI0)),
         DATA(1MUDCRIR0),OPTIONS(W,C,M,NGCP),
         MASK(M(MU1DWVCRI0)
```

This statement sets the MS (write) options to W, C, and M. No GCP is executed before the location MU1DRW01 is written with 1MUDCRIR0. A GCP is then executed and the reply is masked through the mask of the MU1DWVCRI0 item and compared to 0*I(MU1DWVCRI01). This is followed by one or two GCPs; then a control write (MS) is executed to write back the store's original status.

STWSTAT3

4.249 The description of the STWSTAT3 statement includes:

Function:

The store write status macro is used to change the status of the store via the MS instruction. The results are checked by a GCP, then the original status is restored by an MS instruction. A general expansion of the macro would appear as follows:

GCP

GCP

MS instruction with new status

GCP

GCP

MS instruction to restore the original status.

Format:

```
          ITEM($1)  NOSTORE
STWSTAT3 ITEMS($2),EXPECT($4),DATA ($5),OPTIONS($6),MASK($7),STATE($8),BUS($9)
          WORD($3)
```

Characteristics of Parameters:

ITEM — Specifies the address (bits 0 through 17) to be written associated with the address of the item layout. The $1 is any Datapool-defined item name.

ITEMS — Specifies the address (bits 0 through 17) to be written associated with the address of the items layout. The $2 is any Datapool-defined item names in the same word.

WORD — Specifies the address (bits 0 through 17) to be written. The $3 is any valid arithmetic expression which represents the expect value.

NOSTORE — Specifies to ignore the results of the GCP read.

EXPECT — Specifies the expected result of the GCP read in a 24-bit word. The $4 is any valid arithmetic expression which represents the expect value.

DATA — Specifies the data to be written to the store on the MS instruction to change the status. The $5 is any valid arithmetic expression which represents the data.

OPTIONS — Specifies the mode, parity, and store timing during the write of the status. If DELAY is specified, a delay of 25 microseconds is taken after the MS instruction. If NGCP is specified, the two GCPs before the MS instruction are not executed. Figure 3 provides a list of options.

MASK — Specifies the mask for the GCP read. If not specified with WORD parameter, a mask of all ones is used. If not specified with ITEM or ITEMS, the mask of the item or items is used. The $7 is any valid arithmetic expression which represents the mask value.

STATE — Specifies the state of the CRI and MTCE flip-flops. The $8 is two of the following:

CRIR — Reset CRI flip-flop

CRIS — Keep CRI set

MTCER — Reset MTCE flip-flop

MTCES — Keep MTCE set.

If $8 is not specified, the default is CRIS and MTCES. If MTCER and/or CRIR is specified, an MS is executed prior to the write of the status to reset the proper flip-flops.

BUS — Specifies the bus to be used for the GCP results. ACT specifies the active bus (first GCP result) and STB specifies the standby bus or second GCP. If $9 is not specified, ACT is used.

Example:

```
STWSTAT3 WORD(MS1DRW10),EXPECT(0*I(MS1GCPSASWF1)),
         DATA(1MSDFPR2),OPTIONS(W,C,M,NGCP,DELAY),
         MASK(M(MS1GCPSASWF1))
```

This macro call will set the write options to W, C, and M. No GCPs will be executed before address MS1DRW10 is written with data 1MSDFPR2. A delay of 25 microseconds is then taken before the first GCP is executed. The results of the GCP read is masked through the mask of MS1GCPSASWF1 and then compared to 0*I(MS1GCPSASWF1). This is followed by another GCP and a control write (MS) to restore the original status.

STWTKBR3

4.250 The description of the STWTKBR3 statement includes:

Function:

The store write K-code blocking register is used to change the state of the K-code blocking flip-flops. The macro does an MS to change the status of the blocking flip-flops. The state of all other status bits is unchanged.

Format:

STWTKBR3 BLOCK($1)

Characteristics of Parameters:

BLOCK — Specifies which of the four K-code block flip-flops to set or reset. The $1 is any four of the following (one for each flip-flop).

0S — Block 0 set
0R — Block 0 reset
1S — Block 1 set
1R — Block 1 reset
2S — Block 2 set
2R — Block 2 reset
3S — Block 3 set
3R — Block 3 reset.

Example:

STWTKBR3 BLOCK(0S, 1S, 2S, 3R)

This macro call will generate an MS instruction to change the status of the K-code blocking flip-flops. Flip-flops 0, 1, and 2 will be set and 3 will be reset.

STWTRTF2

4.251 The description of the STWTRTF2 statement includes:

Function:

The STWTRTF2 statement provides the only means of altering the store answer flip-flops. The associated macro routine uses the maintenance load (MS) instruction to write the new status of the answer flip-flops. The remaining STATUS bits MTCE, CRI, RO, and K-code remain unchanged.

Format:

STWTRTF2 ROUTE($1)

Characteristics of Parameters:

ROUTE — Specifies the state of the answer on bus 0 (ANS0) and answer on bus 1 (ANS1) flip-flops. The $1 is two of the following:

ANS0S — Set ANS0

ANS0R — Reset ANS0

ANS1S — Set ANS1

ANS1R — Reset ANS1.

Example:

STWTRTF2 ROUTE(ANS0R,ANS1R)

This statement executes a control write (MS) to reset the store answer flip-flops.

STWTRTF3

4.252 The description of the STWTRTF3 statement includes:

Function:

The store write routing flip-flops macro is used to change the store answer flip-flops. The macro generates an MS instruction to change the state of the answer flip-flops. All other status bits are unchanged.

Format:

STWTRTF3 ROUTE($1)

Characteristics of Parameters:

ROUTE — Specifies the state of the answer flip-flops, answer on bus 0 (ANS0) and answer on bus 1 (ANS1). The $1 is two of the following:

ANS0S — set ANS0
ANS0R — reset ANS0
ANS1S — set ANS1
ANS1R — reset ANS1.

Example:

STWTRTF3 ROUTE(ANS0S,ANS1R)

This macro call will generate an MS instruction to set ANS0 and reset ANS1 status bits.

STWUPDN2

4.253 The description of the STWUPDN2 statement includes:

Function:

The STWUPDN2 statement is the first half of a memory test in which all addresses are written. The data written into each location is equal to its address or its address complement. The write either starts at address 0 and runs through address octal 177777 (up direction) or starts at address octal 177777 and runs through 0 (down direction). Segment breaks occur automatically within the associated macro routine after a predetermined number of addresses have been written.

Format:

STWUPDN2 DIRCTN($1),$\overline{\text{OPTN}}$($2)

Characteristics of Parameters:

DIRCTN — Specifies direction of write (up or down). The $1 is up or down.

OPTN — If specified, specifies data equals complement of address. The $2 is C. If not specified, data equals address.

Example:

STWUPDN2 DIRCTN(UP),OPTN(C)

This statement writes the complement of the address starting at address 0 and running through octal 177777.

STWUPDN3

4.254 The description of the STWUPDN3 statement includes:

Function:

The store write up down macro is used to write all of memory with data equal to the address or the address complemented. The data is written in either the UP direction (address 0 through 777777) or the DOWN direction (address 777777 through 0). Segment breaks are automatically initiated by the routine.

Format:

STWUPDN3 DIRECTN($1),OPTN($2)

Characteristics of Parameters:

DIRECTN — Specifies the direction of the write operation. The $1 is either UP or down.

OPTN — Specifies the complement of the address. The $2 is C. If not specified, the data is equal to the address.

Example:

STWUPDN3 DIRECTN(UP)

This macro will write all of the 256K addresses with data equal to the address starting with address 0.

ST2EXTST

4.255 The description of the ST2EXTST statement includes:

Function:

The ST2EXTST statement exercises, reads, a store memory test address at system speed. The test address range is specified by the DGN and EX input message parameters explained in Fig. 2. The memory reads are executed from the opposite store community (ensuring the orders are consecutive) using the LOAD instruction in the normal mode. The LOAD command is a 1-word instruction which is necessary to use when executing out of call store. The memory reads are executed in bursts of 512 addresses. If the address length is less than 512, the start address (or addresses) will be read again to make full use of the burst of reads. If the start test address is octal 77776 with an address length of three the burst of reads will be 77776, 77777, 100000, 77776, 77777, 100000, etc, until 512 reads are executed. Segment breaks occur automatically within the routine after a predetermined number of addresses have been read.

Format:

ST2EXTST

Characteristics of Parameters:

This statement has no parameters.

ST2RD512

4.256 The description of the ST2RD512 statement includes:

Function:

The ST2RD512 statement is the second half of a short version of the regular checkerboard memory tests. The statement verifies the content of 512 locations using a special algorithm where the combination of addresses used check all combinations of the address decoding paths. Segment breaks occur automatically within the associated macro routine after a predetermined number of addresses have been written.

Format:

ST2RD512 START($1),ADD1($2),ADD2($3),ADD3($4),COUNT($5),EXPECT($6)

Characteristics of Parameters:

START — Specifies the starting address to be used by the algorithm. The $1 is the address.

ADD1,ADD2,ADD3 — Specifies the address increments. The $2, $3, and $4 are increments used by the algorithm.

COUNT — Specifies the total number of addresses to be read. The $5 is any arithmetic expression which expresses the total number of addresses.

EXPECT — Specifies the expected results in a 24-bit word. The $6 is any arithmetic expression which expresses the expected result; $6 may also be PWR to indicate power-up K-code.

Example:

ST2RD512 START(1MUDACC1STT),ADD1(1MUDACCADD1),ADD2(1MUDACCADD2),
ADD3(1MUDACCADD3),COUNT(1MUDACCNT),EXPECT(1DGACT01)

This statement checks all combinations of the address decoding paths of the store. It starts at address 1MUDACC1STT and reads a total number of addresses equal to the value of 1MUDACCNT. The addresses to be read are calculated by the algorithm using 1MUDACCADD1, 1MUDACCADD2, and 1MUDACCADD3 as increments. The results are compared to the value of 1DGACT01.

ST2STCC

4.257 The description of the ST2STCC statement includes:

Function:

The ST2STCC statement uses the standby (STB) central control as a helper unit to test for the store's proper response. The associated macro routine will do one of the following:

(1) Read the tested program store with specified read options from the active (ACT) central control. This is then followed by a read of the STB CC ABL (mod 0) or ABR (mod 1) registers to check data bits 0 through 23. If the data parity bits are to be checked, then the STB CES register is read. This will ensure when one mod is referenced on a memory read the other mod will also respond. For this operation the macro is, briefly, expanded as follows:

(a) Control write to reset the CRI flip-flop

(b) ML instruction

(c) Read STB CC ABL, ABR or CES registers

(d) GCP

(e) GCP.

(2) The ACT CC will execute an MS or ML instruction and at the same time the STB CC will GCP the tested store. The correct response of the store is tested from the results of the ML instruction (if executed) in the ACT CC. This is followed by a GCP instruction of which the results are compared with specified expected results. This GCP is executed by the ACT CC. For this operation the macro is, briefly, expanded as follows:

(a) Control write to reset the CRI flip-flop

(b) ACT CC executes MS or ML instruction at the same time the STB CC GCPs the tested store

(c) GCP

(d) GCP.

Format:

```
ST2STCC WORD($1),DATA_EXPECT($2),OPTIONS($4),MASK($5)
                EXP_P2P1($3)

        ,MOD($6),STATE($6)*,EXML*,GCPEXP($8)*, GCPMSK($9)*
```

WORD — Specifies the address (bits 0 through 15) of the ML or MS instruction. The $1 is the address.

DATA_EXPECT — Specifies the data to be written and the expected results of the read (ML) unless EXP_P2P1 is specified. The $2 is any arithmetic expression which expresses the data.

EXP_P2P1 — Expected results data parity bits, P2 and P1, data bits 24 and 25 are rotated into bits 1 and 0 of the 24-bit word. The $3 is any arithmetic expression which expresses the expected results data parity bits.

OPTIONS — Specifies the mode and parity during the MS or ML instructions. The $4 is a list of options. Figure 1 shows a list of options for the MS and ML instructions. The inhibit store timing pulses are not available for this macro routine.

MASK — Specifies the mask to be used for the ML instruction. The $5 is any arithmetic expression which expresses the mask.

STATE — Specifies the state of the maintenance (MTCE) flip-flop. The $6 is one of the following:

MTCER — Reset MTCE during test.

MTCES — Keep MTCE set. If not specified, $7 is MTCES.

EXML — Execute ML instruction. If not specified and the MOD PAR also is not specified, the MS instruction is executed (from the ACT CC). During the ML/MS instruction the standby CC executes GCP order.

Note: The MOD and EXML parameters cannot be specified on the same macro call.

When the MOD PAR is specified, an ML instruction is executed from the ACTIVE CC. The standby CC ABL or ABR REG is then read to check the program store response.

When the MOD PAR is not specified, an ML or MS instruction is executed from the active central control at the same time a GCP is executed from the standby central control.

MOD — Module to read response in the STB CC. The $6 is one of the following:

1 — Read mod 1 which would be the ABR register.

0 — Read mod 0 which would be the ABL register.

GCPEXP — Specifies the expected results of the GCP after the MS/ML instruction. The $8 is any arithmetic expression which expresses the GCP expected results.

GCPMSK — Specifies the mask to be used for the GCP after the MS/ML instruction. The $9 is any arithmetic expression which expresses the GCP mask.

Example:

The following is performed only on program store:

```
(1)  ST2STCC WORD(MU1DM200),DATA_EXPECT(1DGALT10),OPTIONS(M,R),
          MASK(1MUDDATA1S),MOD(1)
```

This statement does a control write (MS) to reset the CRI flip-flop. The STB CC is then stopped and set up to only run during the read (ML) order. The ML order reads the word at location MU1DM200 sending the M and R bits. Mod 1 response is tested; therefore, the STB CC ABR register is read. The result is compared to the value of 1DGALT10 with a mask of 1MUDDATA1S. This is then followed by two GCPs. The STB CC is restored in the matching and in-step mode.

The following is performed on call store and program store:

(2) ST2STCC WORD (MU1DM200),DATA_EXPECT(1DGALT01),
OPTIONS(M,R),MASK(1MUDDATA1S),EXML,
GCPEXP(O(20000014),GCPMSK(O(20000014))

This statement does a control write (MS) to reset the CRI flip-flop. The STB CC is then stopped and set up to GCP the tested store at the same time the ACT CC executes an ML instruction sending the M and R bits. The results of the ML instruction are compared to the value of 1DGALT01 with a mask of 1MUDDATA1S. After the ML instruction the ACT CC performs a GCP of the tested store and compares the result to octal 20000014 with a mask of octal 20000014. A second GCP is then performed and the STB CC is restored in the matching and in-step mode.

ST2VRTST

4.258 The description of the ST2VRTST statement includes:

Function:

The ST2VRTST statement verifies a store memory test address range specified by the DGN and EX input messages. Figure 2 explains the input parameters used to specify the store start address, address length, test data pattern, to rotate the data and to inhibit the store's refresh. The input parameter information is obtained from the diagnostic buffer table. The associated macro routine uses the maintenance load (ML) instruction to verify the specified address range. The expected result is the test data pattern. The data pattern is rotated after each address, if specified. The data parity bits are also checked for correctness. All data failures are shown by the store histogram printout. Segment breaks occur automatically within the associated macro routine after a predetermined number of addresses have been verified.

Format:

ST2VRTST

Characteristics of Parameters:

This statement has no parameters.

ST2WRTST

4.259 The description of the ST2WRTST statement includes:

Function:

The ST2WRTST statement writes a store memory test address range specified by the DGN and EX input messages. Figure 2 explains the input parameters used to specify the store start address, address length, test data pattern, to rotate the data and to inhibit the store's refresh. The input parameter information is obtained from the diagnostic buffer table. The associated macro routine uses the maintenance store (MS) instruction to write the specified address range. The test data pattern is rotated after each address if specified. Segment breaks occur automatically within the routine after a predetermined number of addresses have been written.

Format:

ST2WRTST

Characteristics of Parameters:

This statement has no parameters.

ST2WR512

4.260 The description of the ST2WR512 statement includes:

Function:

The ST2WR512 statement is the first half of a short version of the regular checkerboard memory tests. The statement writes the contents of 512 memory locations using a special algorithm where the combination of the addresses used check all combinations of the address decoding paths. Segment breaks occur automatically within the associated macro routine after a predetermined number of addresses have been written.

Format:

ST2WR512 START($1),ADD1($2),ADD2($3),ADD3($4),COUNT($5),DATA($6)

Characteristics of Parameters:

START — Specifies the starting address to be used by algorithm. The $1 is the address.

ADD1,ADD2,ADD3 — Specifies the address increments. The $2, $3, and $4 are increments used by the algorithm.

COUNT — Specifies the total number of addresses to be read. The $5 is any arithmetic expression which expresses the total number of addresses.

DATA — Specifies the data to be written. The $6 is any arithmetic expression which expresses the data.

Example:

ST2WR512 START(1MUDACC1STT),ADD1(1MUDACCADD1),ADD2(1MUDACCADD2),
ADD3(1MUDACCADD3),COUNT(1MUDACCNT),DATA(1MUDDATA1S)

This statement checks all combinations of the address decoding paths of the store. It starts at address 1MUDACC1STT and writes a total number of addresses equal to the value of 1MUDACCNT. The addresses to be written are calculated by the algorithm using 1MUDACCADD1, 1MUDACCADD2, and 1MUDACCADD3 as increments.

ST2WVTST

4.261 The description of the ST2WVTST statement includes:

Function:

The ST2WVTST statement will write, then verify, each address of a store memory test address range specified by the DGN and EX input messages. Figure 2 explains the input parameters used to specify the store start address, address length, test data pattern, to rotate the data and to inhibit the store's refresh. The input parameter information is obtained from the diagnostic buffer table. The associated macro routine uses the maintenance store (MS) instruction to write the specified address and the maintenance load (ML) instruction to verify the specified address of the address range. The data written is the specified test data pattern which is also the 24-bit expected result for the verify. If specified the test pattern is rotated after each address is written, then verified. The data parity bits are also checked for correctness. All data failures are shown by the store histogram printout. Segment breaks occur automatically within the associated macro routine after a predetermined number of addresses are written/verified.

Format:

ST2WVTST

Characteristics of Parameter:

This statement has no parameters.

ST3EXTST

4.262 The description of the ST3EXTST statement includes:

Function:

The store exercise test macro is used to do a continuous exercise of a specified address range at system speed. The address range is specified on the DGN and EX input messages explained in Fig. 4. The memory reads are executed from the opposite community to guarantee the orders are consecutive. The memory reads are executed in groups of 512 addresses. If the address range is less than 512, the address will be looped through until 512 addresses have been read. Segment breaks occur automatically within the routine after a predetermined number of addresses have been read.

Format:

ST3EXTST

Characteristics of Parameters:

This statement has no parameters. All input is from the DGN and EX input messages.

Example:

See Fig. 4 for DGN and EX examples.

PS x
DGN: CS y: PH Z, TSTADR $1, L $2, TSTPAT $3, INHREF, ROTPAT

TSTADR — The start test address. Bits 0 through 17 of the store to be exercised. $1 is any store address in octal. Address Bit 15 is the word select bit, 1 = word 1 (odd word) and 0 = word 0 (even word). The default start address is zero.

L — Length of consecutive addresses to be exercised. If the highest store address (777777) is reached, the next address is 0, 1, etc, until the length is reached. The $2 is the number of addresses in decimal. The default is one.

TSTPAT — Specifies the 24-bit data pattern to be used for the test. The $3 is the data pattern in octal. The default test pattern is zero.

INHREF — Specifies that refresh will be inhibited during the testing. The default is to not inhibit refresh.

ROTPAT — Specifies to rotate the 24-bit data pattern after each address. The default is to not rotate the data.

The following example will diagnose PS 0 using Phase 97 of the diagnostic. The start test address will be 0(1402). The length of addresses tested will be 50 and test data pattern will be 0(25252525).

DGN:PS 0:PH 97, TSTADR 1402, L 50, TSTPAT 25252525!

If the exercise routine (ST3EXTST) is to be looped over, this can be done by the following examples.

(1) EX:CS 2; START!

(2) EX:CS 2:PH 97, ADR a-b, L 20, INHREF!

(3) EX:CS 2!

Input message 1 will start the looping process. Input message 2 will then loop over the segment bounded by "a" and "b" where "a" is the data table (DT) address of the seginit before the ST3EXTST macro and "b" is the address of the siginit macro following. The start test address is zero with a length of 20. Refresh will be inhibited during the looping process. Input message 3 will stop the loop.

Fig. 4—Description of the DGN and EX Input Messages

ST3MRH2K

4.263 The description of the ST3MRH2K statement includes:

Function:

The store march 2K macro is used to test a cross section of 2000 addresses in the 256K store. All 2K addresses in a row or column are previously initialized using ST3WR2K. Addresses in a row or column are read starting at the bottom or top of the row or column. The complement data is then written back into the address. The address is incremented or decremented and the process is repeated. This tests the decoding and chip select for all bits of the 25-bit words. Segment breaks occur automatically within the routine after a predetermined number of addresses.

Format:

ST3MRH2K EXPECT($1),DIRECTION($2),TYPE($3)

Characteristics of Parameters:

EXPECT — Specifies the data which is expected when an address is read. The $1 is any valid arithmetic expression which represents the expect value.

DIRECTION — Specifies the direction to read the addresses in a row or column. The $2 is UP for starting at the bottom (low address) of a row or column or DOWN for starting at the top (highest address) of a row or column.

TYPE — Specifies the type of addresses being read. The $3 is ROW for a row of addresses and COLUMN for a column of addresses.

Example:

ST3MRH2K EXPECT(1DGALT01),DIRECTION(UP),TYPE(ROW)

This macro call will start reading at the bottom of a ROW. The expect for the read will be 1DGALT01. The complement of 1DGALT01 is then written back to the same address that was read. The address is incremented and the process repeated until 2K addresses have been marched through.

ST3PATAN

4.264 The description of the ST3PATAN statement includes:

Function:

The store pattern analysis routine is used to analyze memory failure information obtained from the failure distribution data from a previous memory exercise (STMARCH3, ST3WR2K, or ST3ERRAN) routine. The pattern analysis routine takes the failure data and formats it into the histogram.

Format:

ST3PATAN HISTOGRAM

Characteristics of Parameters:

HISTOGRAM — Specifies that a histogram will be printed from the failure distribution data. If not specified, no histogram is printed and the failure data is stored into decision vectors only.

Example:

ST3PATAN HISTOGRAM

This macro call will cause any failure data to be printed as a histogram printout. Figure 5 shows a sample histogram.

M 16 DGN: PS 0 PH 18 MSG STARTED
MEMORY FAILURE RESULTS:
TSTPAT=77777777; THRESHOLD=NORM
NUMBER OF FAILING ADDRESSES= 2048; ERROR SUMMARY=00000000

ADDRESS BIT	COUNT	DATA BIT	WORD1	WORD0	
0	0(0)	0	1024	1024	
1	0(0)	1	1024	1024	
2	0(0)	2	1024	1024	
3	0(0)	3	1024	1024	
4	0(0)	4	0	0	
5	0(0)	5	0	0	
6	0(0)	6	0	0	
7	0(0)	7	0	0	
8	2048(0)	8	0	0	
9	2048(0)	9	0	0	
10	2048(0)	10	0	0	
11	2048(0)	11	0	0	
12	2048(0)	12	0	0	
13	2048(0)	13	0	0	
14	2048(0)	14	0	0	
15	0(0)	15	0	0	
16	0(0)	16	0	0	
17	0(0)	17	0	0	
		18	0	0	
		19	0	0	
		20	0	0	
		21	0	0	
		22	0	0	
		23	0	0	
		24	0	0	PARITY 2
		25	0	0	PARITY 1

Fig. 5—Example of Histogram Printout

ST3STCC

4.265 The description of the ST3STCC statement includes:

Function:

The store standby central control routine uses the standby central control as a helper unit. The two central controls execute the separate instructions specified in the macro for each central control. A read of some register in the standby central controls is then performed to verify that the operations are performed as expected.

Format:

```
ST3STCC ACTIVE((($1),BOTHBUS),STANDBY(($2),
        ACTBUS, BLKE),ADDRESS($3,MTCE,STBBUS,
                                 WORD($6)
        CRI),DATA($4),RUN($5),ITEM($7),
                              GCP($8)
        EXPECT($9),IDR_DATA($10),EXP_IDR
```

Characteristics of Parameters:

ACTIVE — Specifies the active central control. The $1 is the instruction which the active central control will execute.

BOTHBUS — Specifies that the active central control will send on both buses. The standby central control will not send at all.

STANDBY — Specifies the standby central control. The $2 is the instruction which the standby central control will execute.

ACTBUS — Specifies that the standby central control will receive on the active bus.

BLKE — Specifies that the standby central control execution sequencer will be blocked and the standby central control will not run.

ADDRESS — Specifies the address of the store operation. This is placed in the K register in the central control. The $3 is any valid store address.

MTCE — Specifies the store is in the maintenance mode.

STBBUS — Specifies the bus the store receives on. If specified the standby bus is used. If not specified, the active bus is used.

CRI — Specifies the state of the CRI flip-flop during the test. If specified, the CRI flip-flop will be set. If not specified, CRI is reset for the test.

DATA — Specifies the data which is put in the G register. If an MS instruction is being performed, this is the data which is sent to the store. The $4 is any valid arithmetic expression which represents the data.

RUN — Number of cycles the standby central control will run. The $5 is any valid number of central control cycles.

WORD — Specifies the address of the read. The $6 is any valid arithmetic expression which represents the address.

ITEM — Specifies the address of the read. The $7 is any valid Datapool-defined item.

GCP — Specifies the read is a GCP read. The $8 is any valid arithmetic expression or Datapool-defined items which define the GCP read items.

EXPECT — Specifies the expected results of the test in a 24-bit word. The $9 is any valid arithmetic expression which represents the expect data.

IDR_DATA — Specifies the data which the Input Data Register (IDR) in the store will be initialized to. The $10 is any valid arithmetic expression which represents the data. If this parameter is not specified, the IDR is not initialized.

EXP_IDR — Specifies the expect value is IDR data. If not specified the expect value is test data.

Example:

```
ST3STCC ACTIVE((ML 0(K),R,C,M)),STANDBY((ML 0(K),R,C,M)),
        ADDRESS(MS1DRW00,MTCE),DATA(0),RUN(1),
        ITEM((ST1IEQ)),EXPECT(1DG_IEQ_04)
```

This macro call will set up the active and standby central control to perform an ML 0(K), R, C, M instruction. The address of the ML which is in the K register will be MS1DRW00. The store will have maintenance set. The standby central control will run for 1 cycle and execute the ML at the same time the active central control is executing the ML. The standby central controls execution sequencer (ST1IEQ) will be read and expected to be equal to 1DG_IEQ_04.

ST3VRTST

4.266 The description of the ST3VRTST statement includes:

Function:

The store verify test routine verifies a store memory test address range specified by the DGN and EX input messages. Figure 4 explains the input parameters used to specify the store start address, address length, test data pattern, and options to rotate the data and to inhibit the stores refresh. The input parameter information is obtained from the diagnostic buffer table. The routine uses the maintenance load (ML) instruction to verify the specified address range. The expected result is the test data pattern. The data is rotated after each address if the rotate option is specified. The data parity bits are also checked for correctness. All data failures are shown by the histogram printout. Segment breaks occur automatically within the routine after a predetermined number of addresses have been verified.

Format:

ST3VRTST

Characteristics of Parameters:

This statement has no parameters.

Example:

See Fig. 4 for DGN and EX examples.

ST3WRTST

4.267 The description of the ST3WRTST statement includes:

Function:

The store write test routine writes a store memory address range specified by DGN and EX input messages. Figure 4 explains the input parameters used to specify the store start address, address length, test data pattern and options to rotate the data and to inhibit the stores refresh. The input parameter information is obtained from the diagnostic buffer table. The routine uses the maintenance store (MS) instruction to write the specified address range. The test data pattern is rotated after each address if the rotate option is specified. Segment breaks occur automatically after a predetermined number of addresses have been written.

Format:

ST3WRTST

Characteristics of Parameters:

This statement has no parameters.

Example:

See Fig. 4 for DGN and EX examples.

ST3WR2K

4.268 The description of the ST3WR2K statement includes:

Function:

The store write 2K addresses routine is used to write the cross section of memory to a constant data pattern. A total of 2000 addresses are written with the data pattern. Segment breaks are taken automatically after a predetermined number of addresses.

Format:

ST3WR2K DATA($1),TYPE($2),NOZRFD

Characteristics of Parameters:

DATA — Specifies the data pattern to be written into memory. The $1 is any valid arithmetic expression which represents the data.

TYPE — Specifies whether the writes will be done on a row or column of addresses. The $2 is ROW if writing a row or COLUMN if writing a column.

NOZRFD — Specifies that the existing fault distribution memory should not be zeroed. If not specified, all fault distribution data is zeroed.

Example:

ST3WR2K DATA(1DGALT01),TYPE(ROW)

This macro call will cause the 2K addresses in the row select to be written with the data pattern 1DGALT01.

ST3WVTST

4.269 The description of the ST3WVTST statement includes:

Function:

The store write and verify test routine is used to write and then verify each address of a store memory test address range specified by the DGN and EX input messages. Figure 4 explains the input parameters used to specify the store starting address, the address length, the test data pattern and options to rotate the data and to inhibit the stores refresh. The input parameter information is obtained from the diagnostic buffer table. The routine uses the maintenance store (MS) instruction to write the memory and the maintenance load (ML) instruction to verify the address of the given range. The data which is written is also the expected value of the verify. The data is rotated after each address if the rotate option is specified. The data parity bits are also checked for correctness. All data failures are shown by the histogram printout. Segment breaks occur automatically within the routine after a predetermined number of addresses have been written and verified.

Format:

ST3WVTST

Characteristics of Parameters:

This statement has no parameters.

Example:

See Fig. 4 for DGN and EX examples.

SYNCDET

4.270 The description of the SYNCDET statement includes.

Function:

The SYNCDET statement tests the synchronization circuit of the standby central control operational clock error detector. The operational clock in the standby central control is started and then an unsynchronizing loop is entered. In this loop, an operational clock phase is inhibited and then restored in an attempt to unsynchronize the clock error detector. If, after a number of attempts, the unsynchronizing loop fails to unsynchronize the error detector, a passing test result is generated and stored in diagnostic scratch call store. If the unsynchronizing loop was successful in unsynchronizing the clock error detector, the error detector is control written to synchronize it. The error detector indicator is then read and the results stored in diagnostic scratch call store for interrogation by other DL-1 statements. The operational clock in the standby central control is stopped.

Format:

SYNCDET

Characteristics of Parameters:

This statement has no parameters.

Example:

SYNCDET

This statement starts the operational clock in the standby central control, then enters an unsynchronizing loop in which an operational clock phase is inhibited and then restored in an attempt to unsynchronize the clock error detector. If the unsynchronizing loop fails to unsynchronize the error detector, a passing test result is generated and stored in diagnostic scratch call store. If the unsynchronizing loop is successful in unsynchronizing the clock error detector, the error detector is control written to synchronize it. Then the error detector indicator is then read and stored in diagnostic scratch call store. The operational clock in the standby central control is stopped.

TAPERETN

4.271 The description of the TAPERETN statement includes:

Function:

The TAPERETN statement calls a DUAD subroutine to inform DUAD that the tape on the TUC transport has been returned to its original position by the program.

Format:

TAPERETN

Characteristics of Parameters:

This statement has no parameters.

Example:

TAPERETN

This statement calls a DUAD subroutine to inform DUAD that the tape on the TUC transport has been returned to its original position.

TBLDELY

4.272 The description of the TBLDELY statement includes:

Function:

The TBLDELY statement will verify the effective enable and disable recovery times of the auxiliary unit bus drivers within the API. Drivers are expected to be enabled in 1.4 microseconds and disabled in 42 microseconds. Disabling and enabling bus drivers are accomplished by setting and resetting the API trouble flip-flop, respectively.

Format:

TBLDELY

Characteristics of Parameters:

This statement has no parameters.

Example:

TBLDELY

This statement will verify the timing specification of the API trouble flip-flop.

TESTDMA

4.273 The description of the TESTDMA statement includes:

Function:

The TESTDMA statement performs one of the following:

(1) Initializes a block of 511 words in 1A call store with unique patterns. This block is then read by the API DMA sequencer.

(2) Verifies a DMA write of 1A call store by the API.

Format:

TESTDMA READ
TESTDMA WRITE

Characteristics of Parameters:

READ — Specifies that the initialization of 1A call store is to be performed. A block of 511 words in the diagnostic raw data area, starting at address F + DG1TASKE, is initialized to unique patterns.

WRITE — Specifies that the verification of a DMA write job by the API is to be performed. A block of 511 words is read and verified starting at address F + DG1TASKE.

Example:

This statement will verify a DMA write job of 511 words performed by the API.

TESTDMA READ

This statement will initialize a block of 511 words with unique patterns. The API then reads this block using the DMA sequencer.◆

TPMOTCHK

4.274 The description of the TPMOTCHK statement includes:

Function:

The TPMOTCHK statement writes a command in the TUC command register and then keeps track of the tape motion indicators in the tape transport status (TTS) register as the command is performed. The transport should be stopped at the completion time indicated. The TIMEOUT parameter causes the transport to time out during the specified command. The DELYSTOP parameter causes a stop command to be issued during the execution of the specified command. The SETMIS parameter causes a write to the TUC maintenance interject status (MIS) register during the execution of the specified command. The BUSRQINH parameter causes the bus requests to be inhibited via the request inhibit group (RIG) register during the execution of the specified command.

Format:

```
                                                    SETMIS
                                                    BUSRQINH
                                                    TIMEOUT
TPMOTCHK COMMAND($1),TTSEXPCT($2),CMPLTIME(MSEC($3)),DELYSTOP
```

Characteristics of Parameters:

COMMAND — Specifies the command. The $1 is the mask of the command as it is in the TUC command register.

TTSEXPCT — Specifies the expected results of the tape motion indicators while the tape is in motion. The $2 is the mask of the TTS register tape motion item(s) expected for the specific command.

CMPLTIME — Specifies the total time in milliseconds to execute the command and stop the transport. The $3 is a number 13 to 8192.

TIMEOUT — Causes a timeout during the specified command.

DELYSTOP — Causes a stop command to be issued during the execution of the specified command.

SETMIS — Causes a write to the TUC MIS register during the execution of the specified command.

BUSRQINH — Causes the bus requests to be inhibited via the RIG register during the execution of the specified command.

Example:

```
TPMOTCHK_COMMAND(M(TU1REVEOF)),
MC          TTSEXPCT(M(TU1LOWSPD,TU1BSYIDL,TU1REVMOT,TU1TTRDY)),CMPLTIME(MSEC(39)),
ME          DELYSTOP
```

This statement writes the command (the mask of TU1REVEOF) in the TUC command register and compares the TTS register to the mask of the TU1LOWSPD, TU1BSYIDL, TU1REVMOT, and TU1TTRDY items after the command has been executed. The transport should be stopped in 39 ms. A stop command is issued during the execution of the TU1REVEOF command.

TUCARIN

4.275 The description of the TUCARIN statement includes:

Function:

The TUCARIN statement adds the address of the start of scratch memory, used for autonomous block transfers for the ADS, to the address of scratch memory (in register F) and puts the sum into the TUC — address register (AR).

Format:

TUCARIN $1

Characteristics of Parameters;

$1 — The Datapool name given to the start of scratch memory for ADS autonomous block transfers.

Example:

TUCARIN DG1ADSBLK

This statement adds DG1ADSBLK, the Datapool name given to the start of scratch memory for ADS autonomous block transfers, to the relative address of scratch memory (in register F) and puts the sum in the TUC-AR register.

TUCAROUT

4.266 The description of the TUCAROUT statement includes:

Function:

The TUCAROUT statement reads the contents of the TUC — address register (AR) and compares it with the parameter supplied with the statement after adding the address of scratch memory (in register F) to the parameter supplied with the statement.

Format:

TUCAROUT $1

Characteristics of Parameters:

$1 — The Datapool name given to the start of scratch memory for ADS block transfers (ie, DG1ADSBLK) plus an offset value if any.

Example:

TUCAROUT DG1ADSBLK+O(132)

This statement reads the TUC-AR register and compares it to the address of DG1ADSBLK plus octal 132 plus the address of scratch memory (in register F).

TUCMREAD

4.277 The description of the TUCMREAD statement includes:

Function:

The TUCMREAD statement is a maintenance-read instruction for the TUC.

Format:

```
          ITEM($1)  NOSTORE
TUCMREAD  ITEMS($2),EXPECT($4)
          WORD($3)
```

Characteristics of Parameters:

ITEM — The internal TUC location to be read. This supplies bits 0 through 6 of the auxiliary unit address bus; K-code, A-code, and bit 10 are supplied by the task routine. The mask for the results is also generated from the attributes of this parameter. The $1 is the item name.

ITEMS — The internal TUC location to be read. This supplies bits 0 through 6 of the auxiliary unit address bus; K-code, A-code, and bit 10 are supplied by the task routine. The mask for the results is also generated from the attributes of this parameter. The $2 is a list of items all in the same word.

WORD — The internal TUC location to be read. This supplies bits 0 through 6 of the auxiliary unit address bus; K-code, A-code, and bit 10 are supplied by the task routine. The mask for the results is also generated from the attributes of this parameter. The $3 is the address.

NOSTORE — If specified, nothing is done with the reply from the TUC.

EXPECT — If specified, this is matched with the reply from the TUC. The $4 is any arithmetic expression which expresses the expected result.

Example:

TUCMREAD WORD(TU1OIS),EXPECT(M(TU1OPCOM))

This statement reads the TU1OIS word in the TUC and compares the result to the mask of TU1OPCOM.

TUCMWRITE

4.278 The description of the TUCMWRITE statement includes:

Function:

The TUCMWRITE statement is a maintenance-write instruction for the TUC.

Format:

TUCMWRITE WORD($1),DATA($2)

Characteristics of Parameters:

WORD — Specifies the internal TUC location to be written. This supplies bits 0 through 5 of the auxiliary unit address bus. K-code, A-code, and bit 10 of the address are supplied by the task routine. The $1 is the address of the TUC location.

DATA — Specifies 24 bits of data to be written into the TUC location. This supplies bits 0 through 23 of the auxiliary unit write bus. The $2 is any arithmetic expression which expresses the data.

Example:

TUCMWRITE WORD(TU1COM),DATA(M(TU1STOP))

This statement writes the TUC location TU1COM with the mask of the TU1STOP item.

TUCMICHK

4.279 The description of the TUCMICHK statement includes:

Function:

The TUCMICHK statement looks for a maintenance interject (MI) in the TUC—control bus register (CBR). It waits a designated time (DELAY), does a no-store read of the TUC-CBR, and then reads the DUS—output buffer register (OBR) looking for an MI. If it finds an MI, TUCMICHK passed. If an MI is not found, it reads the TUC-CBR again and repeats the check. This procedure is repeated some number of times (NUM), waiting one delay time (DELTA) between each check. If the loop occurs more than NUM times, the timed out failing data = 76543210. If a data transfer is to take place, DATAFXR causes auxiliary unit hard and soft stops to be inhibited.

Format:

```
                          SEC($1)          SEC($1)                    _________
TUCMICHK DELAY(MSEC($2)),DELTA(MSEC($2)),NUM($3),MSG($4),DATAXFR
```

Characteristics of Parameters:

DELAY — Specifies the time delay before the first read of the TUC-CBR. The $1 specifies the time in seconds. The $2 specifies the time in milliseconds. Delay time is between 100 ms and 186 seconds inclusive.

DELTA — Specifies the time delay between subsequent reads of the TUC-CBR. The $1 specifies the time in seconds. The $2 specifies the time in milliseconds. Delay time is between 10 ms and 256 seconds inclusive.

NUM — Specifies the number of times a check for an MI is made. The $3 is between 1 and 31 inclusive.

DATAXFR — Inhibits auxiliary unit hard and soft stops if a data transfer is to take place while TUCMICHK is waitng for an MI.

Example:

TUCMICHK DELAY(SEC(17)),DELTA(MSEC(200)),NUM(10),MSG(8),DATAXFR

This statement waits 17 seconds, then reads the TUC-CBR register and looks for an MI. If an MI only is found, TUCMICHK passed. If an MI is not found, it waits 200 ms, reads the TUC-CBR register again, and looks for an MI. If an MI only is found, TUCMICHK passed. If it still does not find an MI, the process of waiting 200 ms, reading the TUC-CBR and looking for an MI is repeated until 10 reads are completed. If an MI is not found after 10 reads, TUCMICHK timed out. Data is to be transferred during the time break, and auxiliary unit soft and hard stops are inhibited.

TUCOICHK

4.280 The description of the TUCOICHK statement includes:

Function:

The TUCOICHK statement looks for an operation-complete indication (OI) in the TUC—control bus register (CBR). It waits the designated time (DELAY), does a no-store read of the TUC-CBR, then reads the DUS—output buffer register (OBR), looking for an OI. If it finds an OI only, TUCOICHK passed. If it finds an MI, the MI failing data = 01234567. If neither an MI nor OI is found, it reads the TUC-CBR again and repeats the checks. This procedure is repeated some number of times (NUM), waiting one time delay (DELTA) between each check. If the loop occurs more than NUM times, the timeout failing data = 76543210. If a data transfer is to take place, time break, DATAXFR causes auxiliary unit hard and soft stops to be inhibited.

Format:

```
                    SEC($1)          SEC($1)
TUCOICHK DELAY(MSEC($2)),DELTA(MSEC($2)),NUM($3),MSG($4),DATAXFR
```

Characteristics of Parameters:

DELAY — Specifies the time delay before the first read of the TUC-CBR. The $1 specifies the time in seconds. $2 specifies the time in milliseconds. Delay time is between 100 ms and 186 seconds inclusive.

DELTA — Specifies the time delay between subsequent reads of the TUC-CBR. $1 specifies the time in seconds. $2 specifies the time in milliseconds. Delay time is between 10 ms and 256 seconds inclusive.

NUM — Specifies the number of times a check for an OI is made. The $3 is between 1 and 31 inclusive.

DATAXFR — Inhibits auxiliary unit hard and soft stops if a data transfer is to take place while TUCOICHK is waiting for an OI.

Example:

TUCOICHK DELAY(SEC(1)),DELTA(MSEC(400)),NUM(25),MSG(0)

This statement waits 1 second, then reads the TUC-CBR and looks for an OI. If it finds an OI only, TUCOICHK passed. If it finds an MI, TUCOICHK failed. If it finds neither, it waits 400 ms, then reads the TUC-CBR again and looks for an OI. If an OI is found, TUCOICHK passed. If it finds an MI, TUCOICHK failed. If it still does not find either, the process of waiting 400 ms, reading the TUC-CBR, and looking for an OI will be repeated until 25 reads are completed. If an OI is not found after 25 reads, TUCOICHK timed out.

VLDWRTPT

4.281 The description of the VLDWRTPT statement includes:

Function:

The VLDWRTPT statement calls a DUAD subroutine to determine if the tape on the transport is at a valid write point. A diagnostic scratch word will be set according to the response from DUAD.

Format:

VLDWRTPT

Characteristics of Parameters:

This statement has no parameters.

Example:

VLDWRTPT

This statement calls a DUAD subroutine to determine if the tape on the transport is at a valid write point.

XCR

4.282 The description of the XCR statement includes:

Function:

The XCR statement sets up call store for tests that follow. The XCR statement is concerned with internal transmission tests for central control. It stores in call store the number of cycles the central control is to run, the expected results, and the address of the part of the central control to be tested. The statement also allows a clock to be inhibited on inactive phases.

Format:

```
     ITEM($1)
XCR ITEM($2),CYCLES($4),INHPHASE($5)
     WORD($3)
```

Characteristics of Parameters:

ITEM — Specifies the address and mask to be stored. The $1 is an item name.

ITEMS — Specifies the address and mask to be stored. The $2 is a list of items all in the same word.

WORD — Specifies the address and mask to be stored. The $3 is the address.

CYCLES — Specifies the number of cycles the standby central control is to run. The $4 is a decimal number.

INHPHASE — Specifies the clock to be inhibited on inactive phases. The $5 is the clock.

Example:

XCR ITEM(ST1A),CYCLE(1),INHPHASE(OOT04)

This statement stores in call store the address and mask of the ST1A item and the number of cycles the standby central control is to run (1). It also inhibits the OOT04 clock during inactive phases in subsequent tests.

XREAD

4.283 The description of the XREAD statement includes:

Function:

The XREAD statement is a general purpose read of the program store or call store . The result is passed to the Diagnostic Control Program (DCON).

Format:

```
        ITEM($1)    ___
▶XREAD ITEMS($2),($3),EXPECT($5)◀
        WORD($4)
```

Characteristics of Parameters:

ITEM — Specifies the location to be read. The $1 is the name of the item.

ITEMS — Specifies the location to be read. The $2 is a list of items all in the same word. The $3 is CSREL, PSREL, or NONREL. If $3 is CSREL, the location specified is indexed by the base address (in register F). If $3 is PSREL, the location specified is indexed by the starting address of the paging area. If $3 is NONREL, the location is not indexed. If $3 is not specified, it is CSREL.

WORD — Specifies the location to be read. The $4 is the address.

EXPECT — Specifies the expected result of the read. The $5 is any arithmetic expression which expresses the expected result.

Example:

XREAD ITEMS(MU1ITEMA,MU1ITEMB),EXPECT(0)

This statement reads the location specified by the address of the word containing the MU1ITEMA and MU1ITEMB items plus the base address (in register F) and compares the results to zero.

5. STATEMENT INDEX—ALPHABETICAL LISTING OF STATEMENTS FOR TESTING BY EQUIPMENT TYPE

5.01 This part provides an alphabetical listing of the DL-1 statements by the type of equipment to which the statements apply. The page number on which the explanation of each statement starts is also given.

(a) Central Control

EQUIPMENT TYPE/STATEMENT (Contd) PAGE

(a) Central Control (Contd)

(b) Call Store/Program Store Bus and Call Store/Program Store

(1) Call Store/Program Store Bus

(2) Call Store/Program Store

EQUIPMENT TYPE/STATEMENT (Contd) PAGE

(2) Call Store/Program Store (Contd)

EQUIPMENT TYPE/STATEMENT (Contd) PAGE

(2) Call Store/Program Store (Contd)

(c) Peripheral Unit Bus

(d) Input/Output Unit Selector

(e) Processor Peripheral Interface and Master Control Console

EQUIPMENT TYPE/STATEMENT (Contd) PAGE

(e) Processor Peripheral Interface and Master Control Console (Contd)

(f) Auxiliary Units and Auxiliary Unit Bus

(1) Auxiliary Unit Bus

(2) Auxiliary Units

(3) File Store

EQUIPMENT TYPE/STATEMENT PAGE

(4) Auxiliary Data System

(5) Data Unit Selector

(6) Data Unit Controller

(7) Tape Unit Controller

(8) Attached Processor System

(g) Miscellaneous (Data Manipulation, Control and Decision, Defining and Calling Subroutines and General Testing)

6. REFERENCES

6.01 The following document provides further information in related areas:

NUMBER	TITLE
PK-5A001	1A Processor Datapool

6.02 The introductory BSP to the application programs (Section 234-180-000) provides a complete list of 1A Processor and application programs and the sections in which they are described. More detailed information about all programs referenced in this section may be found by referring to this BSP.

7. GLOSSARY

7.01 This part of the section provides definitions of terms associated with DL-1.

A — An "A" preceding an item name in parentheses indicates the address attribute. The address of an item in a memory location is the absolute address of the memory location in which the item appears. For example, A (DG1TEMP) indicates the address of the DG1TEMP item.

Aborted — A task is aborted if it is terminated (abnormally) prior to normal end; eg, due to a maintenance interrupt.

Alphabetic — Alphabetic characters include all the letters in the alphabet, A through Z.

Alphanumeric — Alphanumeric characters include all the letters in the alphabet and all the numbers in the decimal numbering system A through Z and 0 through 9.

AND — When two binary numbers are combined by the logical product (AND) operation, each bit of one binary number is matched with the corresponding bit of the other binary number. When corresponding bits are 1s, the result is a 1. When either of the corresponding bits is a 0, the result is a 0.

Arithmetic Expression — An arithmetic expression is a number, symbol, function, attribute, indirect symbol, or a string of these items separated by arithmetic operators. The resulting expression yields a value of a 24-bit binary integer. 6, STM6, WRT*6+3 are examples of arithmetic expressions.

Arithmetic Operator — The allowable arithmetic operators (in order of highest to lowest procedure) are:

**	exponentiation
+,—,¬	unary plus, unary minus, complement

*,/	multiplication, division
+,–	addition, subtraction

B — A "B" preceding a number in parentheses indicates that the number is of the binary number system (2 numbers). In the binary numbering system, only 0s and 1s are used. B(10010) indicates a binary number.

Binary — See B.

Block — A block of memory is several adjacent words. A task block is a block of memory containing the machine language code required to execute the task.

C — A "C" preceding a group of characters in parentheses indicates characters. For example C(ABC) indicates the characters ABC.

Client — A client program runs under the supervision of another program.

Complement — When any binary number is involved in a complementary (NOT) operation, each of the bits in the number is changed to its opposite binary representation.

D — A "D" preceding a number in parentheses indicates that the number is of the decimal numbering system (10 numbers). In the decimal numbering system, numbers 0 through 9 are used. D(689) indicates a decimal number. If no numbering system is indicated for a number, it is decimal.

Decimal — See D.

Default — Most optional parameters have default values. If the parameter is not specified, the default value is used.

E — An "E" preceding a number in parentheses indicates a 1 in that bit position. E(8) indicates a 1 in bit position 8, all other bits in the word 0.

Exclusive OR — When two binary numbers are combined by the Exclusive OR operation, each bit of one binary number is matched with the corresponding bit of the other binary number. When the corresponding bits agree (both bits are 1s or both bits are 0s), the result is a 0. When the corresponding bits do not agree, the result is a 1.

Execution Time — Execution time is the point in time during which a task is run. Execution time is also the length of time required to run a task.

Format — The format of a statement is the arrangement of parameters and variables allowed in the statement as it appears on a program listing.

H — An "H" preceding an item name in parentheses indicates the displacement attribute. The displacement of an item in a memory location is equal to the number of the bit positions the item is from bit 0 (the rightmost bit in the item). For example H(DG1TEMP) indicates the displacement of the DG1TEMP item.

Hexadecimal — See X.

I — An "I" preceding an item name in parentheses gives a value of 1 in the rightmost bit of the item. For example, I(DG1TEMP) gives the value 1 to the rightmost bit in item DG1TEMP.

Insertion Mask — To insertion mask an item from a CC register into a memory location, the bit positions in the data buffer register corresponding to the bit positions in the logic register that contain 1s are replaced by the contents of the corresponding bit positions of the specified CC register. The other bit positions in the data buffer register remain unchanged, and the new contents of the data buffer register replace the contents of the memory location.

Item — An item is a group of adjacent bits within a word. The item may contain any number of bits up to and including 24. Each item has the following attributes: address, size, displacement, mask, represented by A, S, H, and M, respectively.

Label — A label is a symbolic name given to the address of a memory location; especially a label is the symbolic name appearing in the location field of a program statement.

M — An "M" preceding an item name in parentheses indicates the mask attribute. The mask of an item in a memory location is a set of 24 bits having 1s in the position occupied by the item and 0s elsewhere. For example, M(DG1TEMP) indicates the mask of the DG1TEMP item.

Macro — A macro is a model for generating assembly language code or pseudo-operations at program assembly time. In diagnostic languages, all macros (statements) generate the DATA pseudo-operation and result in a table of data.

Mask — See M.

Masking — Masking is the process of performing a logical AND of a MASK on a word. Product masking and insertion masking are used.

Mnemonic — A mnemonic is an abbreviation or shorthand notation which helps the user understand the operation. For instance, ML is the mnemonic for MAINTENANCE LOAD; MS is the mnemonic for MAINTENANCE STORE.

O — An "O" preceding a number in parentheses indicates that the number is of the octal numbering system (8 numbers). In the octal numbering system, numbers 0 through 7 are used. O(777) indicates an octal number.

Octal — See O.

Op-code — An op-code is an abbreviation for operation-code. A large number of operations have been assigned a numeric code.

OR — When two binary numbers are combined by the logical union (OR) operation each bit of one binary number is matched with the corresponding bit of the other binary number. When corresponding bits are 0, the result is a 0. When either or both of the corresponding bits is a 1, the result is a 1.

Page — A program page is a block of machine language code or data which resides on file store and must be constructed in main memory before it can be executed. Paging is the process of reading program pages from file store and reconstructing them in available main memory for execution.

Parameter — A parameter is a quantity that is assigned a temporarily constant value in order to conduct a diagnostic test. A parameter followed by a variable in parentheses is assigned the value of that variable. If a parameter has no variable, the presence or absence of the parameter assigns the value.

Pest — A pest is a flip-flop that inhibits an interrupt.

Phase — A phase of a diagnostic program is designed to test part of a unit. A phase is a convenient division of a diagnostic program.

Product Mask — Product masking is the operation of taking some constant, the contents of a register (other than the logic register) or the contents of a memory location and ANDing it with the contents of the logic register before it reaches its destination.

Rawdata — Raw data is generated as a result of diagnostic tests. This raw data is messaged by the Diagnostic Control Program and passes to the Input/Output Control Program to be printed in an output message.

S — An "S" preceding an item name in parentheses indicates the size attribute The size of an item in a memory location is equal to the number of bits occupied by the item. For example, S(DG1TEMP-indicates the size fo the DG1TEMP item.

Segment — A segment is a series of tests executed without a time break. A segment runs for no longer than 3 ms. Diagnostic phases are divided into segments.

Size — See S.

Statement — A diagnostic statement is a macro call.

Syntax — Syntax is the set of rules needed to construct a statement.

Subroutine — A subroutine is a block of machine language code used by other programs. A calling program transfers to an entry point in the subroutine; upon completion of execution, the subroutine returns control to the statement immediately following the transfer instruction in the calling program. Diagnostic subroutines differ from other subroutines in that they contain no executable code. They are data tables; the first word in each entry in the data table is an index into the task routine which contains executable code.

Unary — Unary is single.

V — A "V" preceding an arithmetic expression in parentheses indicates the value of the expression. For example, V(1*8*1DG1TEMP) indicates the value of the expression 1*81DG1TEMP.

Value — See V.

Variable — A variable is used to assign a temporary constant value to a parameter. The value a particular parameter is to take during a diagnostic test is in parentheses following the parameter.

Word — A word is 24 consecutive bits in memory addressable by central control.

X — An "X" preceding a number in parentheses indicates that the number is of the hexadecimal numbering system (16 numbers). In the hexadecimal numbering system, numbers 0 through 9 and the letters A, B, C, D, E, and F are used. (XFO) indicates a hexadecimal number.

8. ABBREVIATIONS AND ACRONYMS

8.01 The following is a defined list of abbreviations and acronyms used in this section.

ADS	Auxiliary Data System
API	Attached Processor Interface
ASW	All Seems Well
ASWF	All-Seems-Well Failure
AUA	Auxiliary Unit Access Bus
AUFR	Auxiliary Unit Fault Recovery Program
BC	Bit Control
BPS	Buffer Pulse Source
CC	Central Control
CATP	Conditional All Tests Pass
CPD	Central Pulse Distributor
CRI	Communications Reply Inhibit
DCON	Diagnostic Control Program
DUAD	Data Unit Administration Program
DUC	Data Unit Control
DUFR	Data Unit Fault Recovery Program
DUS	Data Unit Selector
GCP	Generate Control Pulse
IOUC	Input/Output Unit Controller
IOUS	Input/Output Unit Selector
LDI	Laboratories Design Information
LED	Light Emitting Diode
MA	Maintenance Access
MACP	Maintenance Control Program
MB	Masked Bus
MC	Macro Continue
MCL	Millisecond Clock
ME	Macro End
MI	Maintenance Interject
ML	Maintenance Load
MS	Maintenance Store
OI	Operation Complete Indication
PAGS	Paging Program
PKA	Address Parity Bit
PPI	Processor Peripheral Interface
SAP	Store Access Permitted
SD	Signal Distributor
SR	System Reinitialization
SWAP	Switching Assembly Program
TUC	Tape Unit Controller
UB	Unmasked Bus