SENTRY

FACTOR PROGRAMMING LANGUAGE Reference Manual

FAIRCHILD
SYSTEMS TECHNOLOGY
A DIVISION OF FAIRCHILD CAMERA AND INSTRUMENT CORPORATION

SENTRY

FACTOR PROGRAMMING LANGUAGE

Reference Manual

Part Number: 67095738
Date Issued: March 1977

1725 Technology Drive, San Jose, California 95110

PREFACE

This manual discusses the Fairchild Sentry II and Sentry VII Programming Language called FACTOR (Fairchild Algorithmic Compiler-Tester ORiented). FACTOR is a procedural programming language that consist of two basic types of statements:

(1) Arithmetic and logical control

(2) Test control statements which set up and execute functional/parametric tests on electronic elements or devices.

This manual is intended for a person with a general knowledge of electronics and a working knowledge of programming concepts. It is suggested that the user become familiar with the glossary of terms before reading this manual. Sections 1 and 2 discuss elements and expressions as well as general format of the FACTOR language. Sections 3 through 5 discuss control statements, blocks and subprogram concepts and statements. Section 6 discusses FACTOR test statements and their format. Sections 7 through 9 discuss FACTOR I/O and operating procedures. A glossary of terms and acronyms, appendices and a a comprehensive index are also included for user convenience.

The following manuals are suggested as supplementary reading and/or reference:

Description	Publication Number
Sentry VII User Reference Manual Reference Manual	67095733
Sequence Processor (SPM) Reference Manual	67095589
Pattern Processor (PPM) User and Programming Reference Manual	67095583
Sentry Utility Reference Manual	67095661
FST-2 Computer Manual	67095701

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF APPENDICES

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS (Continued)

SECTION 1.0

ELEMENTS OF FACTOR

This section defines the basic elements used in the FACTOR programming language, and the syntax information used to prepare programs for input to the compiler.

1.1 CHARACTER SET

Letters	A through Z and $ #
Digits	0 through 9
Special	() * + -/ , . : ; = @ [ ] space ↑ &
Other	! % ? → SEE PG 108 in work book For complete list.

The above characters are valid for a FACTOR source program. The "special" characters have meanings in FACTOR, the "other" characters may be used only in a FACTOR program REMARK statement. The meaning of a special character depends on the context in some cases. For example, the colon is used to define the immediately preceding identifier as a statement label; it also may be used in the binary pin pattern definition in a functional test statement. The correct meaning is chosen by the compiler from adjacent information in the statement. The meaning of all special characters is discussed in the text of this manual. Appendix A tabulates the internal code for the character set.

1.2 FACTOR STATEMENTS

A FACTOR statement is the basic functional entity in a FACTOR program. Except for the IF statement, all statements are terminated by a semicolon.

Example:

```
LABEL:  A = A + 1;
```

1.3 PROGRAM PREPARATION

A FACTOR program is a group of statements designed to do a specific task, i.e., test a particular device. The program is compiled or translated into a set of object codes, which are interpreted by the system. The result of this translation is called the binary test program, test plan, or data file. The compiler input is called the source version of the test plan.

A SII/SVII FACTOR program begins with a SET PAGE statement followed by the main body of the program, and terminated with an END statement. A program is executed in the order written unless a specific statement alters the flow of control.

1.3.1 Record Format

A record is an arbitrary amount of data read from or written into an input/output device. A record typically contains from 1 to 80 characters, depending on the input/output device type. For a VKT, a record is the amount of data from one carriage return character until the next carriage return character. For a line printer, a record is one line. Character position within the record is frequently called a column; independently of the the medium on which the record is written - even if it is not punched cards. Only the first 72 columns of a record are used for FACTOR input. The next 8 columns are reserved for sequence numbers.

FACTOR provides free field input. That is, there is no implied correspondence between the end of the record and the end of the statement. Also, wherever one space is legal, as many spaces as desired can be used. Hence, a statement may start in the middle of a record and continue for as many records as desired. Conversely, more than one statement can be placed in one record. Statement labels need not start in a fixed field such as column 1, and so on. The restrictions are: (1) individual terms, such as variable names, reserved words, and noise words cannot be divided between two records, and the record may be terminated at any point where a space is a legal character; (2) only the first 72 columns of the record may be used for statements.

1.3.2 Cards

When the user prepares the card deck of source statements there are several options. Up to 72 columns of the card may be used for one or more statements, providing a semicolon delimits each statement. Sequence characters are placed in columns 73 through 80 otherwise a portion of the statement is interpreted as a sequence symbol. A FACTOR statement may be started on one card and be carried over to the next, provided that individual words or tester instructions are not divided between two cards.

Cards can be seqeunced either alphabetically, numerically, or both. The normal form of sequence numbers is a fixed alpha identifier in columns 73-75, followed by numeric digits in the remaining columns through column 80. These (five) digits ascend in sequence through the program by a convenient increment, one, ten, etc. Sequence symbols are checked for progression. Gaps (e.g. sequencing by tens) in the sequence may be left, so that program corrections and additions may be made without changing every sequence number in the deck. If a single deck contains more than one alpha identifier, these identifiers must be chosen so that the TRASCII (TRuncated ASCII) ascending collating sequence is maintained, otherwise a sequence error is produced.

When an error is detected, the compiler types the full current record and the warning message "SEQUENCE ERROR".

Examples:

	Legal Sequences						Illegal Sequences					
NUMERIC	1	3	3	10	20	99999999	0	1	3	2	5	4
COMBINED	1LB	2LB	3LB				1A	0A	3A	1A		
ALPHA	A	C	D	E	F		A	B	P	E	C	

1.3.3 Disc

Disc files may be used as a source program input to FACTOR. They must be type "string".

1.3.4 Video Keyboard Terminal

Source programs may be entered directly via the keyboard. In this mode of operation the statements in each input record are compiled as they are entered.

Two editing characters may be entered from the VKT keyboard. A character back space is obtained by typing the "control" and "B" keys simultaneously. The number of times this key combination is typed corresponds to the number of previously entered characters which are to be ignored. A line delete is obtained by typing "control" and "L" simultaneously. This character deletes the current line only. It is necessary to type "carriage return" and "line feed" after the last END statement in the program.

1.4 SYNTAX

Many FACTOR statements have numerous possible forms. Syntax notation provides a convenient method of identifying all options precisely and succinctly. Special syntactical characters are used to identify alternative forms of the statement. These characters are not entered as part of the statement; they simply define the statement's structure. Throughout this manual, the term "General Format", is used as notification that the next line is presented in syntax notation.

1.4.1 Constant Parameters

Any word shown in upper case in the general form is a constant parameter and is always entered exactly as shown. The general form of a statement required to disconnect the precision measuring unit is as shown below.

Example:

```
XPMU  PIN;
```

This statement has no options.

1.4.2 Variable Parameters

Any word shown in lower case in the general form is a variable parameter; the word used indicates what kind of information is required. The limits on the value of the variable depend on the statement in which it is used.

Example:

CPMU PIN expression;

The word expression indicates a number or variable must be entered and not the ten characters that comprise the word expression.

CPMU PIN 3; is a legal statement.

1.4.3 Required Parameters

Brackets are used to enclose a set of parameters where one, and only one, of the parameters in the set must be used. The parameters in the set are separated by the slash character. An underlined parameter identifies the default case if the entire statement is omitted.

Examples:

1. ON [FCT/DCT/TRIP], label;

   Either FCT, DCT or TRIP must be specified.

2. [ENABLE/DISABLE] RELAY;

   Either ENABLE or DISABLE must be used in this statement. If the entire statement is omitted, however, the disable relay state is assumed.

1.4.4 Optional Parameters

Parentheses are used to enclose a set of optional parameters where, at most, one of the parameters in the set may be used. A slash is used to seperate parameters in the set. An underlined parameter identifies the default case if only that parameter is omitted.

Examples:

1. MEASURE VALUE (, LOG);

   The ", LOG" is optional in this statement, if LOG is used, the comma must be used.

2. . . . (, RNG2/,RNG3);

   If the range is programmed, either ,RNG2 or ,RNG3 is used. If the range parameter is omitted, RNG3 is assumed.

1.4.5 Syntax Characters

The brackets and parentheses characters are also used as a required part of the syntax for certain statements.

The brackets and parentheses are used to define pattern replication of binary pin pattern control statement (refer to section 6.4.1.1). In those patterns where the brackets and parentheses are shown, they are required and are not part of the syntax notation.

Parentheses are a required part of the input/output statements syntax (refer to section 8.0).

Brackets are required in the array declaration statements to designate array size (refer to section 7.1.2).

Examples:

1) SET F [8] (2:1);
   The previous pin pattern is preserved to pin 8 and a new pattern starting at pin 8 is specified by the following pattern of two "ones".

2) READ (CR) & DEVNAM, &STAT;
   Information is read from the card reader and input data for literal variable DEVNAM and STAT is extracted.

3) DCL ARR [10] ;
   Defines array size as 10 elements.

SECTION 2.0

EXPRESSIONS

An expression is a grouping of one or more numbers, variables, and functions combined with arithmetic or Boolean operators and parentheses so as to represent a quantity or an operation. Note that a single number or variable is considered an expression by this definition.

2.1 NUMBERS

FACTOR accepts numbers in three forms:

(1) integers
(2) decimal fractionals
(3) exponentials

In all cases, numbers are converted to a floating point internal representation for manipulation in the computer (refer to Appendix J). The range of allowable decimal numbers is:

$$2.7105\ \left(10^{-20}\right) \leq |n| \leq 9.2228\ \left(10^{18}\right) \text{ or } n = 0$$

where: $|n|$ means the "magnitude of n".

2.1.1 Integers

An integer is defined as a whole number, including zero. It is interpreted as octal if it is immediately followed by a B, otherwise it is interpreted as decimal. It may be either signed (preceded by a + or -) or unsigned. If unsigned, it is interpreted as positive.

The limits for decimal and octal integers are:

Decimal integer	$-8388607 \leq n \leq +8388607$
Octal integer	$-37777777B \leq n \leq +37777777B$ (in two's complement form)

The following are examples of integers:

ACCEPTABLE	UNACCEPTABLE	
0	4,000,000	Commas not allowed
4000000	10000000000000000000	Too large
+2361	125 B	Imbedded blank between number and octal identifier
-5		
6B		

2.1.2 Decimal Fractionals

A decimal fractional is any signed or unsigned decimal number with a fractional part preceded by a period. Decimal fractionals cannot be octal. An attempt to use octal notation in combination with a decimal fractional results in an error message.

The following are examples of fractional numbers:

ACCEPTABLE	UNACCEPTABLE	
4.0	4.	A number cannot end with a period
0.0		
.671	.1234B	A fractional number cannot be specified as octal
+.734650		
42.0		
0.734650		

2.1.3 Exponentials

Exponentials may be signed or unsigned decimal integers or decimal fractionals followed immediately by an E and a positive or negative decimal integer.

The following are examples of exponential numbers:

ACCEPTABLE	UNACCEPTABLE	
0.1E2	0.1E2+	The sign must come between and its integer
+1.23E-5		
7E-3	1E	The exponent must have a number
-1.0E+5		
-5E+2	.234 E5	Imbedded spaces are illegal
	2.BE2	Octal numbers may not be exponentially specified

2.2 VARIABLES

In FACTOR, a variable denotes any quantity which is referred to by a name rather than by an explicit value. A variable may take on many values. The values

assigned may be any of the forms discussed above or they may be either scalar or Boolean. A variable identifier may reference either a single variable or a set of variables considered as an array.

Two general classes of variables may be referenced by the FACTOR program; system global variables and user variables. The values of the system global variables are retained from one execution of the program to the next. This allows the programmer to accumulate such information as total number of devices tested, number of parts passing or failing certain tests within the program. The values of all user variables are lost at the end of the test sequences and are reset to zero at the start of the next sequence. The system global variables are saved for each station or test position. System global variables are automatically declared.

2.2.1 System Global Variables

There are twenty-three system global variables which are accessable from a FACTOR program. The names and special uses of these system global variables are:

SWITCH	SWITCH is normally accessed through the TOPSY command SWITCH from the system PID. The global variable SWITCH may then be interrogated using a FACTOR statement to perform such operations as conditional branching.
VALUE	Contains the last value obtained by executing the statement: MEASURE VALUE/NODE/VARIABLE
TIME	Value set by TIME is retained from one execution to the next execution.
GLOB1 through GLOB20	May be assigned a single value each through a FACTOR variable assignment statement. May be set by a TOPSY command for program control.

SN Serial Number

In addition to the initialization procedures the TOPSY command LOAD also affects the system global variables contents depending upon whether or not the modifier SAVE is used with the command. If it is not used, the global variables are initialized to zero. If it is used, the current values of the global variables are retained and may be used by the new program being loaded.

Examples:

```
GLOB1 = GLOB1 + 1;
WRITE 'DEVICE SERIAL NUMBER', GLOB1;
```

outputs an ascending device serial number to the POD each time the program is executed.

```
IF SWITCH EQ 2 THEN GOTO LOOP;
```

The global variable SWITCH may be interrogated using the following FACTOR statement to perform conditional branching.

```
SWITCH = SWITCH + 1
```

SWITCH may also be assigned a value from within the user's FACTOR program using a variable assignment statement.

```
MEASURE VALUE;
USERVAR = VALUE;
```

The content of the global variable VALUE may then be assigned to a user declared variable, if desired, using a FACTOR variable assignment:

```
SET PAGE 512;
      :
      :
TIME = TIME + 1;
XYZ = XYZ + 1;
WRITE TIME, XYZ;
END;
```

The results of TIME and XYZ would look as follows for 3 pressings of the START buttom:

TIME	XYZ
1	1
2	1
3	1

This is because TIME is a global and its value is kept from one executio to the next, where as XYZ is a user variable and is set to 0 after each execution.

2.2.2 User Variable Identifiers

Variable identifiers are names given to variables to aid the user's memory and make the program more intelligible. The identifier must begin with a letter ,#, or a dollar sign and may contain only letters, periods, dollar signs, pound signs, and digits. Identifiers can be of any length, however, FACTOR retains only the first eight characters. The first eight characters must be unique. System global variable names or reserved words must not be used as identifiers. The following is a list of reserved words:

AND	FOR	PAUSE
AT	FORCE	PGEN
BEGIN	FUNCT	PGM
BLOCK	GE	RD
BRANCH	GOTO	READ
BY	GT	REM
CALL	IF	RESET
CGEN	INSERT	REXEC
CLEAR	LCGEN	SET
CONN	LEQ	SOCKET
CPMU	LSET	SPEC
DCL	LSUBR	SUBR
DISABLE	LT	THEN
DO	MEASURE	THRU
ELSE	NEG	UPDATE
ENABLE	NEQ	WR
END	NOISE	WRITE
EOR	NOT	XCON
EQ	ON	XPMU
EXEC	OR	

It is good programming practice to use identifier names that represent the meaning or use of a variable. For instance, TEMP could be the name given to a working variable. COUNTER might be the name given a variable that is used as a general purpose counter, and so on. Variables are assigned an initial value of zero. Arrays must be declared before the array is referenced. Refer to Section 7 for variable and array name declaration information.

The following are examples of variable identifiers:

ACCEPTABLE	UNACCEPTABLE	
A	123	Identifiers may not start with a digit
CHISQUARE		
ALARGEIDENTIFIER	AB*C	Special characters, including blanks are not allowed
A1B2C3D4		
DARRYL	END	Reserved words are illegal

2.2.3 Scalar Values

The FACTOR variable in its simplest form is scalar. Scalars are defined as quantities having magnitude and no direction (i.e., no vectors). (Note that, as defined below, an array element may be a scalar value and/or a Boolean value.) In addition scalars may be any legal numbered value.

Example:

```
2.352
3.1414
+6
-3
```

To use the scalar value currently assigned to a variable, the user writes the variable's identifier in the program statements or expressions.

Example:

```
VCC=5;
FORCE VF1 VCC,RNG2;
```

2.2.4 Boolean Values

Boolean values are quantities which when evaluated have a value of either one (true) or zero (false). Expressions involving Boolean operators can only take on a true or false value thus expressions are not evaluated for any other absolute value. Whenever the user references the variable identifier the current Boolean value is returned.

2.2.5 Array Values

An array is an ordered series of values which are grouped together positionally with respect to some variable identifier (usually the first array element identifier). The elements of the array are restricted to either signed or unsigned numbers (except alpha values are legal for literal variables). FACTOR arrays are restricted to one dimension.

To obtain an array value, the user must follow the array identifier with an expression which is enclosed in brackets. The value of the expression is the subscript which tells FACTOR which element of the array is wanted. If the subscript is zero, (i.e.,A [0]) FACTOR returns the array size. Any other value of the subscript refers to the appropriate element in the array of values. For example, A [2] would reference the second element in array A.

NOTE

If the value of the expression is negative or greater than the array size, a terminal error results during execution.

2.3 FUNCTIONS

Functions are parameterized calls and are used to obtain a value through a standardized set of operations. The FUNCT statement is described in Section 4.

2.4 ARITHMETIC EXPRESSION EVALUATION

Arithmetic expressions are evaluated left-to-right according to the following rules.

(1) Parenthesized expressions are evaluated first. If parenthesized expressions are nested, the innermost expression is evaluated, then the next innermost until the entire expression has been evaluated.

(2) Within parenthesis and/or whenever parenthesis do not govern the order or evaluation, the hierarchy of operations in order of precedence are shown in Table 2-1. The arithmetic expressions are:

(a) Negation (NEG)
(b) Exponentiation (↑)
(c) Multiplication or division (*,/),
(d) Addition or subtraction (+,-).

Example:

The expression

A*(Z-((B↑C)/T)) + VAL

is evaluated in the following sequence.

$B \uparrow C \rightarrow e_1$

$e_1/T \rightarrow e_2$

$Z-e_2 \rightarrow e_3$

$e_3 * A \rightarrow e_4$

$e_4 + VAL \rightarrow e_5$

NOTE

The value of B is limited to positive or zero, since the general result for a negative quantity raised to a power is a complex number. C may be positive, negative or zero.

2.5 LOGICAL EXPRESSIONS

2.5.1 Logical Operators

The logical operators defined for the Test System operate on full word integers. The logical operators are:

Symbol	Operation
AND	Logical and
OR	Inclusive or
EOR	Exclusive or
NOT	One's complement

With two expressions or constants P and Q, for each bit of P matched with the corresponding bit of Q, the following truth table holds:

P	Q	P AND Q	P OR Q	P EOR Q	NOT P
1	1	1	1	0	0
1	0	0	1	1	0
0	1	0	1	1	1
0	0	0	0	0	1

The order of precedence for the logical operators is shown in Table 2-2

Example:

Calculate a percentage and form an integer before printing:

```
PC = 100*X/Y;
PC = PC AND NOT 0;
WRITE 'PERCENT GOOD=', PC;
```

Prior to performing the logical operation the expression is evaluated and then fixed as an integer. The integer form is limited to 16 bits, therefore conversion underflow or overflow from floating point to fixed format must be considered by the programmer.

2.5.2 Logical Expression Evaluation

Logical expressions are evaluated in order of operator precedence and from left to right when two or more operators of the same precedence exist.

Logical expressions are useful for specifying more than one option in a single variable. Using octal notation, suppose that the "hundreds" digit of the system variable SWITCH is used to select ABORT on first fail if the digit is non-zero, the tens digit is used to select a device grade, and the units digit is another quantity of interest. The logical AND operator can separate this information as follows:

```
SWITCH AND 700B extracts the "hundreds" octal digit.
SWITCH AND  70B extracts the "tens" digit.
SWITCH AND   7B extracts the "units" digit.
```

Example:

```
IF (SWITCH AND 700B) NEQ 0 THEN BEGIN

    ON FCT, ABORT;
    ON DCT, ABORT;
    ON TRIP, ABORT:
                                END;
```

where ABORT is a label at the end of the program.

2.6 BOOLEAN EXPRESSIONS

2.6.1 Relational Operators

Relational operators deal with the comparison of two logical expressions, arithmetic expressions, variables, or constants in any combination. The result of the comparison is either true or false. The relational operators are:

Symbol	Operation
EQ	equal
GE	greater than or equal
GT	greater than
LT	less than
LEQ	less than or equal
NEQ	not equal

Consider the following examples:

```
A=16
B=25
A LT B

16 LT 25  This is a true statement.

B LT A    This is a false condition.
```

All relational operators have the same precedence level. (Refer to Table 2-2).

2.6.2 Evaluation of Boolean Expressions

A Boolean expression uses logical and relational operators and defines whether a true or a false condition exists. True and false conditions are represented by non-zero and zero respectively.

The order of operations for Boolean expressions depends upon the precedence values of the operators. The precedence order is shown in Table 2-1. The Boolean expressions are:

(a) NOT
(b) relational operators (LT, LEQ, EQ, GE, GT, NEQ)
(c) AND
(d) OR and EOR

The following are examples of Boolean expressions:

1. A (where A is either true or false)
2. A OR B EOR C
3. A GE B OR A LT C

In example 2, the expression is evaluated from left to right. A is ORed with B and then the result is EORed with C. In example 3, the expression A GE B is evaluated for a true or false condition: the expression A LT C is evaluated: the results of these two operations are ORed together.

2.7 MIXED EXPRESSIONS

FACTOR allows mixing of arithmetic and Boolean expressions, without adhering to pure Boolean values. It is the responsibility of the programmer to ensure that values in mixed expressions are valid integers when they are involved in a Boolean expression. Arithmetic operators take precedence over Boolean operators in mixed expressions.

TABLE 2-1 PRECEDENCE VALUES OF OPERATORS

Symbol	Operation	Precedence Value
NEG NOT	Unary negate Not	1 (highest precedence)
↑	Exponentiation	2
/ *	Division Multiplication	3
+ -	Addition Subtraction	4
LT LEQ EQ GE GT NEQ	Less than Less than or equal Equal to Greater than or equal Greater than Not equal	5
AND	Logical and	6
EOR OR	Exclusive or Inclusive or	7 (lowest precedence)

SECTION 3.0

CONTROL STATEMENTS

Control statements are used to direct the flow of the program by a transfer of control to different parts of the program. Such a transfer may be imperative (e.g., GOTO) or conditional (e.g., IF).

The control statements discussed in this section are PAUSE; GOTO; IF; BEGIN; and FOR.

3.1 PAUSE

The PAUSE statement is used to stop the execution of further statements until the START button is depressed. The general form is:

```
PAUSE expression;
```

The statement number and the value of the expression is output to the POD. The EIR register displays the pass/fail results of functional and DC tests which were executed since the last pause or beginning of test.

The PAUSE statement can be used to provide a programmed halt when debugging new FACTOR programs

NOTE

Refer to the Sentry User's Manual for use of the manual analysis PAUSE command

3.2 GOTO

A program is essentially a series of statements which, in general, are executed sequentially, and thereby accomplish a particular task. The computer thus operates one step at a time. However, it is essential to be able to enter or leave the sequence of instructions at any desired point

This is the function of the GOTO statement. When executed, a GOTO statement always changes the program flow from the statement immediately following it to the one specified in the GOTO statement.

The simplest form of the GOTO statement is:

```
GOTO label;
```

where label is an identifier up to eight characters specifying the statement to be executed next. The statement label is terminated by a colon.

The label must be in the same block or a lower block (refer to Section 4), it is not permissible to jump into a subroutine from the main block or from another subroutine.

3.2.1 Indexed GOTO

General Form:

GOTO (label1, label2, ..., labeln) expression;

As in a GOTO statement, program control is transferred to a specified location (defined by a label) in the FACTOR program. In the indexed GOTO statement, a multiple of such locations are allowed. The choice of one of the locations depends upon the result from evaluating the expression.

GOTO Location	Expression Value
Label 1	1
Label 2	2
Label 3	3
Label n	n
Next consecutive statement is fetched and executed.	0, negative or > n

In cases where the expression contains variables, the same indexed GOTO statement may cause branching to different locations at different times, depending upon the changing expression value.

3.3 IF

The GOTO statement provides one method for altering the sequence of statement executions. It is also essential to be able to change the sequence of execution based on what happens as the program executes, i.e., a conditional change of execution. This is the principal use of the IF statement.

The simplest form of the IF statement is;

IF relation THEN statement1;

statement2;

Upon execution of the IF statement, if the relation is true, statement1 is executed followed by statement2 (unless statement1 carries control away from statement2). If the relation is false, statement2 is executed instead and statement1 is skipped.

Example:

IF A EQ 3 THEN GOTO LABEL;

If A is equal to 3, the sequence of execution is changed to the point in the program having a statement labeled LABEL. If A is not equal to 3 the next sequential statement, after the IF statement, is executed.

The true-false nature of the above relation gives a clue to the second general form of the IF relational clause. It is:

IF Boolean-expression THEN statement;

where Boolean-expression is any legal expression.

Example:

To continue doing something until the value of A and B, two variables being manipulated, both become less than some terminal value, 0. We could make this decision and monitor the values of A and B with one IF statement as follows:

IF A LT 0 AND B LT 0 THEN GOTO DONE;

In all cases of IF statement usage, the statements following THEN can introduce any type of FACTOR statement.

3.3.1 The Conditional ELSE

The simple IF statement is one which causes a statement to execute if a relation or Boolean expression is true and skips statement execution if the relation or expression is false. A complete conditional statement does more. It specifies a second statement to be carried out if, and only if, the relation or expression is false. The general forms are:

IF relation THEN S1 ELSE S2;
IF Boolean expression THEN S1 ELSE S2;

where S1 and S2 are any two statements. When the result of the IF operation is true, S1 executes and S2 is ignored. Notice here that ELSE terminates the first statement (S1) instead of a semicolon. When the result is false, S1 is skipped over and S2 executed. S2 may be any statement, including another IF statement. This nesting of conditionals can go to any depth.

Example:

IF relation THEN S1 ELSE IF relation THEN S2 ELSE S3;

3.4 BEGIN

FACTOR allows the grouping of a series of statements between the statements BEGIN and END. The END, must immediately follow the last statement executed. Note that the semicolon is an integral part of the END; bracket. One purpose of this is to allow a compound statement to follow the THEN or the IF statement.

Example 1:

```
IF relation THEN
      BEGIN
            statement;
            statement;
            statement;
      END;
```

Example 2:

```
IF relation THEN
      BEGIN

      statement 1;
      statement 2;
END ELSE BEGIN
      statement 3;
      statement 4;
END;
```

The above are examples of compound statements, and are an acceptable method of writing the IF statement. The statements between BEGIN and END; are legal and, as far as the IF statement is concerned, are considered to be one statement. In other words, if the relation is false the statement after the BEGIN-END; block is executed next. Any FACTOR statement may be part of the compound statement, including another IF statement.

3.5 FOR

One of the techniques most widely used in programming is that of the program loop. This is the repetition of some program statement or statements over and over with different parameters. The FOR statement is the looping mechanism within FACTOR.

The general format of the FOR statement is:

```
FOR variable = expression THRU expression DO statement;
```

where variable expression and statement may be in any legal form defined in this manual.

Several statements may be included in the DO loop portion of the FOR statement by specifying a compound statement with BEGIN and END.

Example:

FOR variable=expression THRU expression BY expression DO

```
BEGIN
statement 1;
statement 2;
END;
```

An example of a typical loop is one designed to solve the following problem. Suppose it is desired to set the elements of an array to zero This can be achieved with the following IF statement sequence of statements:

```
        I = 1
NEXT:   A [I] =0;
        I = I+1
        IF I LEQ A [0] THEN GOTO NEXT;
```

but it is more easily programmed with the statement:

```
FOR I = 1 THRU A [0] DO  A [I] =0;
```

The simple FOR statement provides an index value which has three important features:

(1) an initial value,
(2) an (assumed) increment of +1,
(3) a limit

In the above example, I takes on the values 1, 2, 3, . . ., A [0] , where A [0] is the last value corresponding to the size of the array.

The implementation of the FOR causes the address of the index, the increment and the limit to be evaluated each time the loop is executed. Therefore, caution must be exercised within the loop when changing values that might affect this evaluation.

The loop is executed the number of times specified by the initial value, limit, and increment. (This may be zero.) Also, there is no restriction on transfers of control into or out of the loop. When the loop has finished its specified number of executions, control passes to the next sequentially executable statement, unless this sequence is interrupted by a statement in the DO loop.

In the above discussion an automatic increment of +1 from the initial value to the final value was assumed. There is a second form of the FOR statement which uses BY; this allows the user to specify some value which is used as the increment.

Example:

```
FOR I = 1 THRU A [0] DO A [I] =0 BY 2;
```

It should be pointed out that because the values may be all positive, all negative, or mixed positive and negative, the user should consider the range of possible values expected. It makes sense to go from a negative number to a more negative number in negative increments or from a positive number to a negative number by negative increments. Going from positive to more positive or negative to positive, the increment must be positive. Going from -2 to +6 in increments of -2, as from +8 to +2 in increments of +2 is not logical and is flagged as an error at execution time.

Caution must be exercised when using fractional values for the index, since it is possible to introduce a step error. For example, a statement such as:

```
FOR I = 0 THRU 1000 BY 0.1 DO X = I + 1;
```

may operate the DO statement more than 10,000 times because of a rounding error in the floating point conversion of 0.1.

FOR statement also applies to a compounded set of statements written within a "BEGIN" and "END" pair. In the examples shown thus far, simple statements are used with the FOR statement. Consider the following example:

```
DCL ARRAY [100];
J = 1;
FOR I = 15E-6 THRU 5E-6 BY-100E-9 DO
        BEGIN
        ARRAY  [J] = I;
        J = J + 1;
        WRITE I;
        END;
WRITE ARRAY;
```

The FOR loop contains 3 statements compounded within the BEGIN-END pair of statements. The three statements are repeated for the number of times taken to reach the terminating value of I, the loop's variable. The FOR loop variable, I, is decrementing (instead of incrementing as previously shown) by 100E-9. The initial, the ending, and the incrementing values of I are all exponentiated instead of integers. The total number of looping times is 100.

Restriction:

The DO BEGIN words, even though they are written on two separate lines, must not have any other word or symbol in between them. Violation of this rule results in the error message, "WARNING NO STATEMENT INSIDE DO LOOP".

SECTION 4.0

SUBPROGRAMS AND BLOCK PROGRAM CONCEPTS

This section describes subprograms, and block program concepts.

4.1 BLOCKS

Blocks are groups of program statements between the delineators, BLOCK and END. Local variable storage and local labels do not exist outside of the parent block they are in and cannot be referenced outside of the parent block. A block is an independent compilation. A program can consist of several completely independent blocks.

A block must have a beginning and a closing statement. In addition, a block can be either independent or dependent.

A block can be established in two ways:

(1) It may be opened directly by writing the command BLOCK and closed by the command END.

(2) A block may also be opened following the FUNCT and SUBR commands.

The initial BLOCK declaration need not be specified because FACTOR assumes a BLOCK 0.

4.1.1 Nesting Blocks

Blocks do not need to be completely independent. One of the easiest methods of introducing block dependence is by "nesting" one block within another. This results in the execution of the inner block being dependent on the execution of the outer block. Nesting can occur up to eight levels on the Sentry. Nesting is illustrated in the following example:

```
BLOCK
      BLOCK
      END;
      BLOCK
            BLOCK
            END;
      END;
END;
```

The inner block of a nested set is considered part of the enclosing blocks. Another form of dependence is that of global variables. A global quantity is one that is accessible to a block, but is not necessarily contained in (i.e., is not local to) that block. Variables and labels can be either local or global. This is illustrated in the following example:

```
BLOCK
L: DCL A, B/10/;
       BLOCK
       DCL A,C;
       END;
END;
```

Each block in the above example contains the local variable A. The A in the inner block cannot be accessed from the outer block and vice versa. The variable B in the outer block is accessible from either block, but the variable C can be accessed only from the inner block. In this example, then, B is a global variable but C and the two variables A are all local.

Note, that if there had been a label L in the inner block, any reference to it within the inner block would have used that rather than the one in the outer block. Any nested set of blocks establishes a block context; i.e., a relationship of local and global variables. From the example, it can be seen that a reference to a variable or label is associated with the occurrence of that identifier or label in the same block, if it is present. If it is not, then the next outer block is examined, etc.

It should be noted that it is possible to make variables global from within a nested block in FACTOR by simply never declaring the variable as local. When the nested block is closed, the variable, and any residual value is relocated to the next outer block, where it may now be considered as global to any further nesting. When this outer block is closed, if it was nested, the variable again relocates to the next outer block and so forth until block 0 is closed.

NOTE

Compilation time can be decreased by declaring (in block 0) all the variables which are not local.

The fact that the declaration of a variable within a block makes it local has important implications for the FACTOR user. After leaving a block, i.e., closing it with an END command, the values of all variables declared within the block, and thus made local, are lost. Upon re-opening the block, the values of these variables are initialized to 0.

4.2 SUBPROGRAMS

Programs frequently have groups of statements which can be used several times with different parameters. The required statements could be duplicated wherever they are needed in the program, but to do so is error prone, and wastes user time and machine storage. Therefore, it is desirable to be able to write statements so that they may be executed from any point in the program with a different set of parameters each time they are executed. The subroutine statement makes this possible.

4.2.1 SUBR

The general formats of the subroutine declaration are as follows:

Format One:

```
SUBR    'entifier;
        statement 1;
        statement 2;
        .
        .
        .
        statement n;
END;
```

Format Two:

```
SUBR  identifier (VI1, VI2, . . . , VIn)
        statement 1;
        .
        .
        .
        statement n;

END;
```

The identifier after the SUBR command is used to reference the subroutine from the main program. The statements within the subroutine are not executed until the subroutine is called from the main program by the SUBR identifier. Any number of statements are allowed within the subroutine.

The END; statement is necessary because the SUBR command effectively opens a new block. When it is completed, it must be closed. The END; indicates the last statement in the subroutine.

Format Two indicates another important feature of the SUBR statement. The terms VI1 through VIn represent variable identifiers 1 through n. They are enclosed in parentheses and indicate to FACTOR that whenever a reference is made to this subroutine, the reference specifies actual values which are to be substituted at specific places within the subroutine body. These identifiers are called formal parameters There is a one for one correspondence between the position of the formal parameters and the position of the parameters or values used in the call. For reference and further explanation, see the next section on the CALL statement. The manner in which values transferred to the subroutine are used in the subroutine's statements is illustrated in the following example:

```
SUBR TOTAL  (VI1, VI2, VI3);
        VI1 = VI2 + VI3;
END;
```

When the above subroutine is referenced:

```
CALL TOTAL (A1, A2, A3);
```

the values passed to it, obtained from the actual parameters A1, A2, and A3, positionally replace VI1, VI2, VI3 and are used in the arithmetic expression and assignment. The value of the variable represented by VI2 is added to that represented by VI3 and the total is assigned to the variable represented by the formal parameter VI1.

The following is an example of a subroutine with no formal parameters specified:

```
SUBR TOTAL2;
        A = B+C;
END;
```

When TOTAL2 is called, the current values of the variables B and C are added and the total is assigned to the variable A. In this case A, B, and C are not formal parameters. They are working variables with current values in the outer blocks to the SUBR statement block.

Because the subroutine forms a new block, it must be remembered that any variables which are formally (i.e., DCL variable name/value;) declared in the subroutine are local.

4.3 CALL

A subroutine is executed by using a CALL statement, which can be placed at any point in the program where the programmer can legally place a statement. The general formats are as follows:

```
CALL SI;
CALL SI (expression 1, expression 2,. . ., expression n);
```

SI is the identifier of the subroutine block to be activated. The values, changed by the subroutine statements and by any other task executed, are accomplished as if the subroutine's body of statements has been placed at the point of the CALL statement. Then, the next sequential statement, following the CALL, is executed.

The expressions are evaluated at the time of the execution of the call and therefore, removes many constraints which are ordinarily placed on the CALL values. As the subroutine statements are executed, the value of expression 1 in the formal parameter list of the CALL statement is used wherever dummy parameter 1 was used. The same holds true for other dummy parameters and expressions. The only restriction is that when a dummy parameter recieves a result, the corresponding actual parameter from the formal parameter list in the CALL should not be an expression but a single variable identifier.

4.4 FUNCT

The subroutine call, when encountered in the program execution, brings the subroutine statements into action to accomplish whatever processing is specified (ordinarily assigning new values to outer block variables). Control then usually passes to the next sequential statement.

When only one variable is assigned a new value, as a result of executing a subprogram, the call can be simplified by making the subprogram a function. When FUNCT is used, simply writing the identifier of the function causes its statements to execute. However, the identifier now represents a value that may be used wherever a variable is legal. Thus, it is as though the function call represents a variable of the same name.

The general form of the function statement is:

```
FUNCT        identifier (VI1, . . . , VIn);
      statement 1;
           .
           .
           .
      statement n;

END;
```

The format, except for the FUNCT command portion, is exactly like the SUBR statement. It also defines a new block and it is called by the identifier, following the FUNCT. The difference is in the way the function is activated and also that it always returns a value for the function identifier. If no assignment of a value to the function identifier is made within the statements following FUNCT and before END; is encountered, a value of zero is returned. Assignments of values to a function, which are external to the function declaration statement are illegal, i.e., the function name must not be used on the left hand side of an assignment statement.

The function identifier is not local to its function block and therefore must not be declared within the function statements.

Like the subroutine call, the FUNCT statement, with its compound tail of statements, is not executed until the function identifier is used in an expression.

4.4.1 Function Call

Using the function identifier as a variable identifier in an expression causes the function statement block to activate and to return a value for the identifier.

Example:

```
FUNCT TOTAL3 (VI1,VI2,VI3);
      ' TOTAL3 = VI1+VI2-VI3;
END;
```

When it is desired to reference a value by executing the above function, the identifier is used as follows:

```
NEWTOTAL = TOTAL3(A,2,B+C)+(A-B);
```

The function TOTAL3 is evaluated using the current value of A for dummy parameter VI1, 2 for dummy parameter VI2 and the current value of B+C for parameter VI3. This overall value is then added to the value calculated for the subexpression, A-B, using current values for A and B. Finally, the end value arrived at is assigned to the variable NEWTOTAL.

The same symbol must not be used for dummy and actual call parameters.

A further example illustrates the range of versatility of a function. Assume that the following sum is to be evaluated:

$$A = \sum_{B=1}^{10} B^2$$

The function statement to handle the sum is shown below:

```
FUNCT       TOTAL (X,Y);
      TOTAL = 0;
      FOR Z = X THRU Y DO TOTAL = TOTAL + (Z*Z);
      END;
```

to evaluate the sum, the call is written as:

```
      SUMB = TOTAL (1, 10);
```

The parameters of a function call can also include other function calls. In fact, a function may even call itself recursively. For instance, a factorial could be calculated as follows:

```
FUNCT  FACTORIA (N);
      IF N EQ 1 THEN FACTORIA = 1
      ELSE FACTORIA = N * FACTORIA (N-1);
  END;
```

This is an example of recursion, the use of a function within the same function. The user should remember that the number of recursive calls is determined (and limited) at run time by the size of memory. Also, because new blocks are opened whenever a call activates a function, this recursion of the function causes nesting. Since nesting can go only 8 blocks deep, this example could easily exceed this depth in most cases.

4.5 EXEC

An assembly language program is executed by using an EXEC statement. The general formats are as follows:

```
EXEC PROGRM;

EXEC PROGRM (parameter 1, parameter 2 . . parameter n);
```

PROGRM is a FACTOR identifier which is also the name of a core image file on disc. Each parameter is evaluated at the time of the EXEC, and may be a global variable, variable, array name array-element, function, formal parameter or arithmetic expression. A maximum of 63 parameters is allowed.

After the assembly language program has been written, it is assembled and an object file is created from which a core image file is created at a particular origin. This must be done either under the system job or under the same job used to run the FACTOR program. The origin, must be greater than the top of $TOPSY. Up to six assembly language linkage files (ALLINK files) may be resident in memory simultaneously. If any ALLINK file overlaps a currently resident file all resident files are removed from memory and their next execution forces a disc access. Refer to the Sentry User's Reference Manual for a description of how TOPSY handles EXEC programs.

The assembly language program is brought into memory at its origin location. All index regsiter values are saved, and then the index regsiters are set to the values described below. Control is passed to the assembly language program. The assembly language program may return normally after executing, or it may take the ABORT exit causing a terminal error message to be printed. After a normal return the index registers are restored and the state switches reset.

It is desirable to keep a short assembly language program in memory after execution so that it may be repeatedly executed without having to access the disc each time. However, since an assembly language program reduces the space available for the FACTOR program, it may be desirable to remove a long program or one not called often to mimimize the number of disc accesses required for executing the FACTOR program.

The ALLINK file can request that the space for it and all other resident ALLINK files be returned to the test program by calling the subroutine ALPCLR. This call is coded as follows:

```
ALPCLR      EQU       1244B
              .
              .
              .
              .
            BSM*      ALPCLR
```

The A register is destroyed by this call.

There also exists a system constant which can be modified to fit the needs of each installation. If an assembly language program is longer than the system constant (set at 4K or 10000B) a disc access is made after execution, allowing the memory area to be used to store the FACTOR program. If the program is shorter than the system constant it remains in memory until a different assembly language program is executed, and its space does not become available for FACTOR program storage. If it is desired to change the maximum size of ALLINK files which remains resident, cell 1243B of $TOPSY may be changed from 10000B to the desired length by PATCH.

4.5.1 Writing the Assembly Language Program

For additional information on writing assembly language programs refer to Appendix J and the FST-2 Assembler Manual.

There must be an entry point. It does not have to be the first statement in the file, however, the file must begin with a BSS statement.

```
ENTRY PROC 0
```

Normal return is:

```
BRU* ENTRY
```

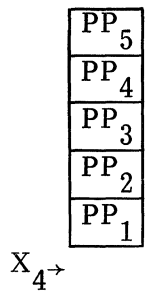
An error exit is provided and should be coded as follows:

```
AOM ENTRY
LDA message number
BRU* ENTRY
```

where $0 \leq$ message number < 900. This causes a terminal error message (message number + 100) to be printed, and the FACTOR test program is aborted.

4.5.2 Referencing Parameters

In order to reference the assembly language program parameters, the working stack contains pointers to the values with Index Register 4 (X4) pointing in front of the first parameter location and Index Register 3 containing the parameter count. Thus, for a routine with five parameters the working stack is as follows:




```
LDA*  1, 4        gets the value of the 1st parameter
STA*  1, 4        stores a new value into the 1st parameter
```

If the second parameter is an array of size 10

```
ARRAY LDA*   2,4
      LXA    5
      LDA    10, 5          gets the value of the 10th element of
                            the array.
```

If it is not known if a parameter is an array or a variable a check may be made.

```
      LDA    2, X4          if the contents are negative it is an
                            array
      BN     ARRAY          array
      LDA*   2, X4          variable, get value
```

4.5.3 Accessing System Routines

Input and output routines, the floating point routines, and other utilities may be used by ALLINK files. However, the input/output routines (or any routine with an interrupt address) must not be called directly. Other routines may be called directly, however, if they are already resident as part of TOPSY the ALLINK file can be kept smaller by calling routines indirectly.

The address in memory of each resident routine is stored in low memory locations in TOPSY called GLOBAL. To access these routines indirectly the user programs a BSM* (branch and store memory-return-address, indirect) to the address of the memory location of the routine. (A direct call is a branch and store memory-return-address to the memory location of the routine.)

There are two methods for coding this calling sequence. The first is the preferable method because it is easier to read. TTRIO, whose address in memory is always stored at location 520B, may be called as follows:

Method 1

```
TTRIO     EQU    520B
          .
          .
          .
          BSM*   TTRIO
          DATA   1
          DATA   DCB
```

Method 2

```
          BSM*   520B
          DATA   1
          DATA   DCB
```

The remainder of the calling sequence is identical to that of a direct call.

The I/O driver, floating point, and other miscellaneous system routines and their memory address locations are shown in Table 4-1.

TABLE 4-1 SYSTEM ROUTINES AND MEMORY ADDRESS LOCATIONS

System Routine	Address of Memory Location
I/O DRIVERS	
CLIO	1232B
CRIO	553B
DISCIO	554B
LPIO	543B
MTIO	552B
TAPIO	541B
TTPIO	512B
TTRIO	520B
FLOATING POINT ROUTINES	
FADD	540B
FDIV	521B
FFIX	542B
FFIXS	514B
FFLTS	515B
OTHER SYSTEM ROUTINES	
ADRXLATE	1236B
ALPCLR	1244B
BINDEC	513B
CILOAD	510B
CLOSE	537B
ENTRFN	1235B
FIND	511B
GET	517B
GFREC	1205B
INREC	551B
OPEN	547B
PFREC	1204B
PUT	516B
PUTN	1210B
READ	545B
SRCH	1233B
WRITDS	1234B
WRITE	544B

SECTION 5.0

NOTATIONAL STATEMENTS AND COMPILER DIRECTIVES

Notational statements and compiler directives are used to enhance the readability and clarity of programs as well as direct the compiler. Notational and compiler statements discussed in this section are NOISE, REM, PAGE, LIST, NOLIST, and INSERT.

5.1 NOISE

The NOISE statement is used to define words that make FACTOR statements read like English sentences. Its general forms are:

```
NOISE word1;
NOISE word1, word2, . . .wordn;
```

The command NOISE is followed by at least one space and the noise word, or words which are separated by commas. After defining the noise words they are ignored by the FACTOR compiler. This provides a means for adding clarity to FACTOR statements. An example is shown below:

```
NOISE VOLTS, AMPS;

FORCE VF1 5.0 VOLTS;
```

Noise words are restricted to the format of identifiers, however, the reserved words, as well as user-declared identifiers, are not allowed as noise words.

5.2 REM

General Form:

```
REM text;
```

The REM (remark) statement provides a means for adding commentary to a program listing. It is not executable and it can occur anywhere in the context of a program provided its format rules are followed. The text may be any alphanumeric or special character except a semicolon.

Example:

```
REM  THIS SECTION PERFORMS VOL MEASUREMENTS;
```

All elements of the character set except the semicolon are positionally listed as located in the REM statements at compile time.

A REM statement is not allowed between a DO and BEGIN of a FOR loop. A warning message is generated if this occurs.

5.3 PAGE

General Form:

```
PAGE;
```

PAGE causes an ejection to top-of-form when the compiled output is being listed on the line printer. Subsequent compiled listings start on a new page. PAGE does not generate any object code.

5.4 NOLIST/LIST

General Form:

```
NOLIST;
LIST;
```

NOLIST suppresses the source listing from the point it appears in the source program until a LIST appears in the source program or until the end of the source is reached.

LIST removes the suppression imposed by NOLIST. If LIST was not specified in the compile command it has no effect.

Example:

```
SET PAGE 1024;
CONN DPS1 12;
CONN DPS3 11;
SET PMU SENSE, RNG3;
NOLIST;
SET F 1010;
SET F 1010;
    .
    .
    .
LIST;
FORCE VF1 4.5;
    .
    .
    .
END;
```

These statements are suppressed and do not appear in the source listing

5.5 INSERT

General Form:

INSERT string file name;

This statement causes the FACTOR statements of the referenced string file name to be compiled, with the resultant data code inserted into working storage, along with the current FACTOR program compilation.

The string file name must be a file under the current job name. In addition, the string file must not contain an END statement as a block end. The inclusion of one causes the compilation to terminate incorrectly at that point. The INSERT string file must not contain a SET PAGE statement.

SECTION 6.0

TEST STATEMENT FORMATS

This section describes the test statements available for testing electronic devices. The statements are grouped within this section by their logical subsystem function. The subsystem functions are:

6.1 PROGRAM INITIALIZATION

This section describes the statement used to initialize the tester prior to performing actual tests.

6.1.1 Set Page

General Form:

SET PAGE integer (,SPM);

Description:

This statement identifies the program being compiled as a high speed test program and sets the number of words of local memory available. The integer should be in the range of 1 to 4096, not to exceed, however, the actual local memory size. The SET PAGE statement is required for running on the Sentry II or VII. If a test program is to be run without a tester, no SET PAGE statement should be used.

If during compilation of the program, a block of consecutive test patterns is found which would require more local memory than was specified, the compiler generates an ENABLE TEST statement. Refer to Section 6.5.

Upon execution of the SET PAGE statement, the integer given is compared with the actual local memory size of the station currently on line. Terminal error 71 is issued if the memory is not at least as large as specified.

The SET PAGE statement is also an indicator to the compiler and the operating system that the current test program is for a high speed station. It is the key for allowing the high speed instruction set and for noting any illegal FACTOR statements.

If the program is to be run on a system with the SPM (Sequence Processor Module) then SPM must be included in the statement. (Refer to the SPM reference manual for a description of the SPM statement.)

For a selected memory size of 1024 words, CHAIN TWO or CHAIN FOUR mode is available. If the memory size specified is between 1025 and 2048 only CHAIN TWO mode is available if SPM is not included in the SET PAGE statement. If the memory size is greater than 2048, chaining is not allowed. (Refer to Section 6.4.8.1 for a description of the SET CHAIN statement.)

During execution, when the last address of local memory is reached, wrap around to location 0 occurs. The effective address, CHAIN TWO and CHAIN FOUR modes are determined by the SET PAGE statement as shown in Table 6-1.

TABLE 6-1 RESULTS OF SET PAGE INTEGER STATEMENT

SET PAGE integer;	Local Memory Wrap Around	CHAIN TWO Allowed	CHAIN FOUR Allowed	Required Local Memory Sizes
1-1024	1023	YES	YES	1k, 2k, or 4k
1025 - 2048	2047	YES	NO	2k , or 4k
2049 - 4096	4095	NO	NO	4k

6.2 ANALOG SUBSYSTEMS

6.2.1 Digital Power Supplies

6.2.1.1 DIRECT LOAD BOARD CONNECTION vs PIN ELECTRONICS CONNECTION

There are two methods for the user to route the digitally programmable power supplies to the device under test. For applications where the load current is under 100 milliamperes, the most convenient connection is via the tester pin electronics. When using this method (see CONN statement description), the digital to analog connector for the DPS supplies gives a programmable reference voltage to the analog reference subsystem for pins specified in this mode. A relay (K1) on the pin electronics card routes the "zero" analog reference buffer directly to the driver connect relay (K3-controlled by SET DA/DB) thus shunting the pin driver switching circuitry and giving a low impedance source for power. Hence, no special load board relay need be used, however, normal supply decoupling at the test socket is recommended. Power supply current measurements in this mode are made directly with the precision measurement unit (PMU).

The second method of power supply routing is the "direct load board connection." This method is normally used for greater than 100 milliampere applications. Each DPS has a one ampere buffer following the D/A connector. The one ampere output is brought out to the test station load board edge connectors. Wiring on the load board makes the final connection to the test socket. It is recommended that direct DPS connections be made via a utility relay on the load board for software controllable connect and disconnect. Also the DPS should be decoupled at the device test socket. Power supply current is measured in this mode with the PMUs internal node sensing capability to allow up to a full one ampere scale.

DPS related statements below point out the differences in these two usage modes.

6.2.1.2 FORCE DPS VOLTAGE SUPPLIES

General Form:

FORCE [VF1/VF2/VF3] expression (,RNG1/,RNG2/,RNG3);

Description:

This statement forces the programmable voltage forcing supplies to the value specified. If the range is not specified, then the highest range is set. Refer to Appendix L. FORCE automatically connects the specified unit to the test station performance board.

When the DPS is in the voltage forcing mode (initiated by the FORCE statement) a current sensing circuit monitors the DPS load currents (when a direct performance board connection is used). The current sense circuit has two purposes:

1) To activate an automatic disconnect function if the current exceeds the current range by 50% (refer to Section 6.2.1.3 for a definition of the current ranges).

2) To be used by the programmable current trip function to detect an overcurrent condition less than the hardware disconnect limit.

For 5MHz and 10MHz test stations, direct DPS voltages are referenced to system ground, which is eleven volts above tester common. Hence, one should subtract eleven volts from the desired programmed level when using direct performance board connections.

If a current trip has been previously enabled refer to the ENABLE TRIPx statement for the affect on the DPS.

Examples:

Force units VF1,VF2,VF3, to +8, 5 and -30 volts respectively.

```
FORCE VF1 8,RNG2;
FORCE VF2 -5,RNG2;
FORCE VF3 -30;      REM RNG3 IS DEFAULT RANGE VALUE;
```

The following is an example of a program that has the ability to branch upon detecting a trip condition by using the ON TRIP statement. Upon branching the program can then measure the DPS load current and record the resultant measurement on an output device by using a FACTOR WRITE statement.

```
        SET PAGE 1024;
          .
          .
          .
        ON TRIP,TRIP1;
        ENABLE TRIPI1 GT 8OE-3,RNG2;
        FORCE VF1 5,RNG2;
          .
        GOTO EOT;
TRIP1:  DISABLE DCT0;
        DISABLE DCT1;
        MEASURE NODE 143;
        WRITE (LP) 'DPS1 TRIP - DPS1 LOAD CURRENT IS',value;
        GOTO EOT;
          .
          .
          .
 EOT:   END;
```

6.2.1.3 ENABLE CURRENT TRIP (DPS)

General Form:

ENABLE [TRIPI1/TRIPI2/TRIPI3] [LT/GT] expression (,RNG2/,RNG3);

Description:

This statement enables the current-trip detector of the corresponding voltage forcing unit (VF1, VF2, VF3). If the source/load current of the forcing unit (VF) exceeds the enabled trip value during a test sequence, indicating a DC failure, then program control is transferred to the instruction as specified by the ON TRIP statement, if issued and the DC FAIL indicator is set. If an ON TRIP has not been executed prior to the trip, the program proceeds normally. At a pause or end-of-test, the DC FAIL indicator is on if a trip occurred. If the DATALOG ON TRIP is set, the value specified by this instruction is written on the output device.

The trips are ignored while the Tester Busy status is 'ON', i.e., until the time delay generated has expired. This feature allows surge currents without setting the trip when the VF supplies are driving capacitive loads. For additional time without increasing the value programmed in the SET DELAY, DC statement, the ENABLE TRIP may be programmed several statements after the FORCE.

The VF units automatically disconnect under the following conditions:

(1) When the magnitude of the current is greater than 150 milliamps and the trip register is in range 2.

(2) When the magnitude of the current is greater than 1.50 amps and the trip register is in range 3.

NOTE

The initial condition of the current trip register is RNG2.

The automatic disconnect from the test station is a safety feature that protects both the device under test and the DPS units. The trips are processed, provided they have been enabled, even though the automatic disconnect occurs.

Examples:

Enable the voltage forcing unit VF1 so that it trips on load currents exceeding 100 milliamps and VF2 trips on currents more negative than -50 milliamps.

```
ENABLE TRIPI1  GT 100E-3, RNG3;
ENABLE TRIPI2  LT -0.05,  RNG2;
```

NOTE

If any trip is enabled, then all trips are enabled by implication. The trip specified is enabled for the particular value; the other is enabled for less than -1 ampere if the DPS is in voltage force mode or for less than -40 volts if the DPS is forcing current.

6.2.1.4 FORCE DPS CURRENT

General Form:

FORCE [IF1/IF2/IF3] expression (,RNG2/,RNG3);

Description:

This statement directs the programmable power supplies to force the specified currents. If the range is not specified, RNG3 is used. The direction of the positive current is out of the supply. This feature may only be used with a direct performance board connection to a load circuit or device under test. This statement automatically connects the specified unit to the test station performance board.

If a voltage trip has not been programmed, an "impossible" trip of less than -40 volts is set, but not enabled.

Examples:

```
FORCE IF1 100E-3,RNG3;
FORCE IF3 -5E-3,RNG2;
FORCE IF2 30E-2;          REM  RNG3 IS DEFAULT VALUE;
```

6.2.1.5 ENABLE VOLTAGE TRIP (DPS)

General Form:

ENABLE [TRIPV1/TRIPV2/TRIPV3] [LT/GT] expression (,RNG2/,RNG3);

Description:

This statement enables the voltage trip detector of the corresponding power supply in the current forcing mode. Voltage trips are processed in the same fashion as current trips. This statement automatically connects the specified unit to the test station performance board.

Example:

Enable trip interrupts occur if the voltage on DPS1 is more positive than -9 volts or if the voltage on DPS3 is more negative than +30 volts.

```
ENABLE TRIPV1 GT -9.0,RNG2;
ENABLE TRIPV3 LT -30;     REM RNG3 IS THE DEFAULT VALUE
```

6.2.1.6 DISABLE CURRENT OR VOLTAGE TRIPS

General Form:

DISABLE TRIPS;

Description:

This statement causes all voltage and current trips to be disabled.

6.2.1.7 DISCONNECTION FROM LOAD BOARD AND OVERCURRENT DISCONNECTION

General Form:

XCON [VF1/VF2/VF3] ;

Description:

This statement disconnects the specified DPS unit from the load board when using direct load board connection. If current forcing, the magnitude of the specified unit is automatically set to 0 in the low range prior to disconnecting. When forcing a voltage, the user should force a value which minimizes current flow before disconnecting. After execution of this statement, the DPS automatically is set to voltage force mode and an impossible current trip condition. The trip function for the specified unit is disabled.

Example:

Disconnect all VF units.

```
XCON VF1;
XCON VF2;
XCON VF3;
```

6.2.1.8 DPS CURRENT OR VOLTAGE MEASUREMENT

General Form:

MEASURE NODE number (,LOG);

Description:

This statement causes the precision measurement unit (PMU) to measure the voltage or current of a DPS at an internal tester monitor node. The DPS as well as the voltage or current measurement function (including automatic measuring and ranging conditions) is defined by the node number (octal or decimal) as summarized below.

NODE NUMBER		DESCRIPTION
DECIMAL	OCTAL	
140	214	DPS1 voltage output
141	215	DPS2 voltage output
142	216	DPS3 voltage output
143	217	DPS1 load current
144	220	DPS2 load current
145	221	DPS3 load current

At the conclusion of the measurement cycle, the PMU is automatically disconnected from the DPS mode and then initialized to force 0 current in range 1. For a more complete description and examples refer to Section 6.2.3.8.

6.2.1.9 MODE CHANGE AND TIME DELAYS

Mode Change

Changing from voltage force to current force implies a disconnect. The user should minimize current flow with a FORCE VFx before a FORCE IFx or ENABLE TRIPx is programmed.

Programmed Delays

A hardware delay is initiated when necessary for relay changes, an additional delay may be desirable in some cases. For example, when forcing voltage into a capacitive load and a trip is programmed, an additional delay may be used to prevent a trip while the capacitance is being charged. The trip can then be used for the steady state load.

The first time the DPS is programmed to a certain mode the delay generated is 5.37 milliseconds or the programmed DC delay, whichever is greater. Subsequent DPS programming in the same mode results in a delay of 0.56 milliseconds or the programmed DC delay, whichever is greater.

Trips

Enabling a trip for any DPS enables trips for all DPS's. If trips are not programmed for remaining power supplies, an "impossible" value of less than -1 ampere or -40 volts is used. Note that this enables the supply to deliver maximum power. Disconnecting any or all supplies does not disable the general trip interrupt in all cases. The disconnected supply can not trip until it is reprogrammed. (Actually, not until its DPS register and any DPT register are reprogrammed.) The general trip interrupt enable can be turned off only with a DISABLE TRIP statement.

Note that programming an ENABLE TRIPV also causes the supply to be connected to the performance board.

6.2.1.10 CONNECTION OF DPS VIA PIN ELECTRONICS (POWER PIN MODE)

General Form:

CONN [DPS1/DPS2/DPS3/TCOM] decimal pin list;

Description:

This statement provides the means for connecting the pins listed to power supplies DPS1/DPS2/DPS3 or to tester common TCOM via the pin electronics card.

The pin list should be comprised of decimal pin numbers separated by blanks or a comma.

Pins which are connected to one supply need not be disconnected before being connected to a different supply. This statement only defines the mode of the pin electronics card and analog reference selection. The final connection of a pin and driver to the device under test must be made with the SET DA/DB statement. Refer to figure 6-3 (Section 6.4.2.1) for the pin relay sequence.

NOTE

When using this method of applying power to the device under test, the DPS supply is acting as a programmable reference voltage to the analog reference buffers which supply current to the pin electronics. Hence, the current monitoring circuits of the DPS do not see the device load current and voltages at the pin are limited to +6/-30(-16) volt swing of the pin cards.

6.2.1.11 DISCONNECTION OF PIN FROM DPS (RETURN TO DATA MODE)

General Form:

XCON PIN decimal pin list;

Description:

This statement causes the listed pins to be disconnected from their current power supplies, bias supplies, tester common, or clock reference supplies and reverts to an NRZ data pin state.

The pin list should be comprised of decimal pin numbers separated by blanks or a comma and become data pins.

When a pin is disconnected, it is reconnected to the selected data reference pair as defined by the SET S statemen. Also, the data driver mode is NRZ (refer to Section 6.4.5.3).

6.2.1.12 POWER UP SEQUENCE EXAMPLE

A. For power supply connection via the pin electronics:

```
FORCE VF1  0;
FORCE VF2  0;
SET DA 00100  00100  00000  1;
ENABLE DA;
SET F  0;
SET PERIOD  1E-6;
ENABLE TEST;              REM  CONNECT  PIN  DRIVERS,  PRE-
                          CHARGE DECOUPLING  CAPACITORS VIA
                          I/O SWITCH;
CONN TCOM 8;              REM SUBSTRATE;
CONN DPS1 16;             REM VDD;
CONN DPS2 3;              REM VBB;
FORCE VF1 VDD;
FORCE VF2 VBB;
```

The above procedure allows the solid state I/O switch to make initial connection. This is desirable because decoupling capacitors are usually tied to system ground (+11 volts relative to tester common) and electro-mechanical reed relays are degraded if required to switch large transient currents. After closing the I/O switch with the ENABLE TEST statement, the CONN statements cause closure of relays bypassing the I/O switch and thus creating the lowest impedance path for power supply currents.

B. For direct load board connection:

```
FORCE VF1 -11;
FORCE VF2 -11;
SET R 00100  00100  00000 1; REM CONNECT SUPPLIES ON LOAD
                             BOARD;
SET VF1 VDD-11;
FORCE VF2 VBB-11;
```

6.2.2 Reference Voltage Supplies (RVS) and Clock Selection

This section describes the statements that control selection and forcing of reference voltage supplies or the alternate or interpretative forms of the reference supplies. This section also describes the statements used for setting output levels and defining logic polarity.

6.2.2.1 CLOCK MODE SELECTION

General Form:

```
CONN CLK decimal pin list;
```

Description:

This statement defines the listed pins as clock pins. The pin list should be comprised of decimal pin numbers separated by blanks or a comma.

The CONN CLK option along with appropriate bit settings in SET F and SET S statements may be employed to connect ~~either the data~~ one of the four reference power supply pair or ~~either~~ alternate clock reference power supply pair to any particular testing pin according to Table 6-2. Refer to figure 6-3 (Section 6.4.2.1) for the pin relay sequence.

TABLE 6-2 CLOCK OPTION, POWER SUPPLY, AND SET F AND SET S STATEMENTS

CLK	S REG	F REG	SUPPLY
0	0	0	E0 Data Reference Pair
0	0	1	E1
0	1	0	EB0 Data Reference Pair
0	1	1	EB1 (Alternate)
1	0	0	EA0 Clock Reference Pair
1	0	1	EA1
1	1	0	EC0 Clock Reference Pair
1	1	1	EC1 (Alternate)

When a pin is defined as a clock pin, a relay is closed which shunts the solid state I/O switch in order to provide a lower source impedance and a greater capacitive load drive capability. A clock pin is automatically put into the Return-to-Zero (RZ) mode, however, a SET RZ to zero statement can be used to put a clock pin in the NRZ mode ~~after~~ if it precedes the CONN CLK statement.

To remove a pin from clock mode use the previously defined XCON PIN statement.

6.2.2.2 SELECTING ALTERNATE REFERENCE SUPPLIES

General Form:

SET S binary pin pattern;

Description:

This statement loads the S register. A binary 0 in the pin pattern selects the standard logic level pair E1/E0 as input forcing voltages for data pins and the standard comparator reference pair S0/S1 for the corresponding test pin. A binary 1 selects the optional alternate logic level pair EB1/EB0, and the optional comparator reference SA0/SA1. When switching from one comparator reference pair to the other, the user must program a one millisecond delay to allow switching of the relay multiplexer for the references.

For CLOCK pins, a binary 0 selects EA0/EA1 reference pair and a binary 1 selects the EC0/EC1 reference pair.

This statement always loads a complete rank of 15 pins. Pins whose status are not specified are assumed by the compiler to be zero or in the last programmed state.

6.2.2.3 INTERPRETIVE REFERENCE VOLTAGE SUPPLIES BIT CHANGES

General Form:

SET SI binary pin pattern;

Description:

This statement is a companion to the SET S statement, except that this statement generates interpretive code.

When this statement is used, all the bits of the S register which are to be modified, must be explicitly specified in the binary pin pattern using the normal SET S coding techniques. Bits not specified in the SET SI statement remain in their current state.

6.2.2.4 FORCING REFERENCE SUPPLIES

General Form:

FORCE [E0/E1/EA0/EA1/EB0/EB1/EC0/EC1]expression (,RNG1/,RNG2/,RNG3);

Description:

This statement forces the reference voltage supplies to the programmed values. If the range is not specified the default (RNG3) is selected. Refer to Appendix L.

E0, E1, EA0, and EA1 are primary voltage references. The truth table in Section 6.2.2.1 shows the assignment of RVS supplies to the tester pins. The supplies EB0, EB1, EC0, and EC1 are optional.

Example:

Force the standard reference pair to 3.5 and .5 volts respectively for the "1" and "0" levels.

```
FORCE  E1 3.5, RNG2;
FORCE  E0  .5, RNG2;
```

6.2.2.5 SETTING OUTPUT REFERENCE LEVELS

General Form:

SET [S0/ S1/SA0/SA1] expression (,RNG1/,RNG2/,RNG3);

Description:

S0 and S1 (and optionally SA0 and SA1) are reference supplies for the functional test comparators. S1(SA1) is the reference level for the expected logic "1" levels (F(i)=1) and S0(SA0) is the reference level for the logic "0" levels (F(i)=0). The programmed value is loaded into the functional test comparator reference voltage suppply register.

The default range (RNG3) is selected if no range is specified.

Example:

Set S0 comparator reference voltage to -5 volts:

```
SET S0 -5, RNG2;
```

6.2.2.6 RVS VOLTAGE MEASUREMENT

General Form:

MEASURE NODE number (,LOG);

Description:

This statement causes the precision measurement unit (PMU) to measure the voltage output of an RVS at an internal tester monitor node. The RVS to be measured is defined by the node number (octal or decimal) as summarized below. At the conclusion of the measurement cycle the PMU is automatically disconnected from the RVS node and then initialized to force 0 current in range 1.

NODE NUMBER		DESCRIPTION
DECIMAL	OCTAL	
128	200	Comparator S1 reference voltage
129	201	Comparator S0 reference voltage
130*	202	Data driver E1 reference voltage
131*	203	Data driver E0 reference voltage
132*	204	Clock driver EA1 reference voltage
133*	205	Clock driver EA0 reference voltage
134	206	Data driver EB1 reference voltage
135	207	Data driver EB0 reference voltage
136	210	Clock driver EC1 reference voltage
137	211	Clock driver EC0 reference voltage
138	212	Comparator SA1 reference voltage
139	213	Comparator SA0 reference voltage

* Values measured at internal nodes 130 through 133 are actually 1/8 of the programmed values due to an internal hardware design of these particular RVS supplies.

6.2.2.7 DEFINE OUTPUT LOGIC POLARITY

General Form:

SET LOGIC [POS/NEG] ;

Description:

This statement initializes the functional test comparator logic pass conditions for either positive or negative voltage logic for the device under test (DUT). If this statement is not used, the positive logic condition is assumed.

Table 6-3 shows the pass/fail decisions made by the comparators. For testing positive logic S1 should be greater than S0. For testing negative logic S1 should be less than S0.

For SET LOGIC POS the pass conditions are defined as follows:

(1) F(i) = 1 (expected output function for pin(i) = 1)
Pass = DUT OUTPUT SIGNAL S1, otherwise fail

(2) F(i) = 0 (expected output function for pin (i) = 0)
Pass = DUT OUTPUT SIGNAL S0, otherwise fail

Where F(i) is the expected functional state of pin(i), refer to Section 6.4.5.1.

For SET LOGIC NEG pass conditions are defined as follows:

(1) F(i) = 1 (expected output function for pin (i) = 1)
Pass = DUT OUTPUT SIGNAL S1, otherwise fail.

(2) F(i) = 0 (expected output function for pin (i) = 0)
Pass = DUT OUTPUT SIGNAL S0, otherwise fail.

TABLE 6-3 COMPARATOR PASS/FAIL CONDITIONS

POS LOGIC:	NEG LOGIC:
F=1 pass S1 ↑↓ F=1 fail	F=0 pass S0 ↑↓ F=0 fail
F=0 fail S0 ↑↓ F=0 pass	F=1 fail S1 ↑↓ F=1 pass

6.2.3 Precision Measurement Unit (PMU)

This section describes the statements that control the operation of the Sentry Test System precision measuring unit. This unit is used for measuring device under test DC parameters and for system self-check. DC test macro statements are provided to reduce programming and increase test throughput.

6.2.3.1 PMU FORCING MODE

General Form:

FORCE CURRENT expression (,RNG0/,RNG1/,RNG2/,RNG3);
FORCE VOLTAGE expression (,RNG1/,RNG2/,RNG3/,RNG4);

Refer To Appendix B-14

Description:

These statements are used to force a programmed current or voltage via the precision measurement unit. Upon execution of these statements, the output of the PMU begins to slew to the desired value. If the range is not specified, the default range is automatically set. If the expression contains only a constant, these statements are executed in DMA mode.

Example:

Force the output of the precision measurement unit to -1 microamp.

```
FORCE CURRENT -1E-6,RNG1;
```

6.2.3.2 DEFINE PMU SENSING RANGE

General Form:

```
SET PMU  SENSE (,RNG0/,RNG1/,RNG2/,RNG3,
                RNG4/,AUTO);
```

Description:

This statement initializes the PMU sensing range. The sense function is complementary with respect to voltage and current (i.e., when the unit is set to force voltage (or current) it is automatically initialized to sense current (or voltage)). When sensing voltage, the legal ranges are 1, 2, 3, and 4. When sensing current the legal ranges are 0, 1, 2, and 3. Refer to appendix L for range limits.

When sensing with autorange is requested the initial measurement is made in the highest range and the range is decreased automatically one range at a time if necessary to provide the best measurement resolution.

Example:

```
SET PMU SENSE,RNG2;
```

This example senses the PMU in range 2.

```
SET PMU SENSE,AUTO;
```

This example senses the PMU in the range that provides the best measurement resolution.

6.2.3.3 CONNECTION OF PMU TO PIN UNDER TEST

General Form:

```
CPMU PIN expression;
```

Description:

This statement connects the PMU to the pin number specified by the expression (or equivalent value). When the expression is a constant, the PMU is connected in DMA mode. The PMU is automatically removed from a previous connection before a new connection is completed.

6.2.3.4 DISCONNECTION OF PMU

General Form:

XPMU PIN;

Description:

This statement disconnects the PMU from its present pin connection and reconnects it to pin 0, an internal tester pin. If a forcing current condition exists pin 0 is connected as an internal tester pin which has a 10Ω resistor relative to tester common. If a forcing voltage condition exists tester pin 0 is in an open state.

6.2.3.5 DEFINING GO/NO-GO PMU TEST LIMITS

General Form:

SET DCT [LT/GT] expression (,RNG0/,RNG1/,RNG2/,RNG3/ ,RNG4);

Description:

The SET DCT forms a pass or fail threshold. When a MEASURE PIN does not pass the level specified by the DCT function, a DC fail is indicated and program control is transferred to the statement specified by the ON DCT statement. If an ON DCT has not been previously executed, then the next statement following the MEASURE is executed. A failure causes the DC FAIL indicator to be lit at the next pause or at the end-of-test. The range specified in this statement overrides that specified in the SET PMU SENSE statement.

Example:

Enable DC trip limits which pass all measured values between -2 milliamps and +2 microamps.

```
SET DCT GT 2E-6,RNG1;
MEASURE PIN;
SET DCT LT -2E-3,RNG2;
MEASURE PIN;
```

This statement sequence is executed at DMA speed.

6.2.3.6 PMU GO/NO-GO Test

General Form:

MEASURE PIN;

Description:

MEASURE PIN allows fast go/no-go DC parameter tests. It is similar to the MEASURE VALUE statement (Section 6.2.3.8) except that go/no go comparisons are made against the SET DCT limit. No floating point conversion is made, nor is the result stored in VALUE.

No autoranging occurs when MEASURE PIN is used.

If any datalogging conditions are specified and met (DATALOG DCT or MEASURE), the resultant measurement is determined by successive approximation.

Example:

```
FORCE VOLTAGE 4.5,RNG2;
SET PMU SENSE,RNG1;
SET DCT GT 6.0E-6,RNG1;
CPMU PIN 5;
FORCE VOLTAGE 4.5, RNG2;        REM ACTIVATES A DC TIME
                                DELAY IN DMA MODE;
MEASURE PIN;
```

6.2.3.7 SOFTWARE DUAL PMU MEASUREMENT LIMITS

General Form:

```
ENABLE  [DCT0/DCT1] [LT/GT] expression;
DISABLE [DCT0/DCT1] ;
```

Description:

Execution of ENABLE DCT0/DCT1 forms a pass or fail threshold for level DCT0 and/or DCT1. Either one or both thresholds may be specified.

Using both thresholds specifies a "pass window" for subsequent DC measurements (see MEASURE VALUE, Section 6.2.3.8). The pass region may be specified by the operators LT/GT and by the value of the expression in the ENABLE statement.

When a measurement caused by the statement MEASURE VALUE does not fall within the "pass window" specified by the DCT function, a DC fail is indicated and program control is transferred to the statement specified by the ON DCT statement (Section 6.2.3.14). If an ON DCT has not been executed, then the statement following the MEASURE is executed. A failure causes the DC FAIL indicator to be lit at the next pause or end-of-test.

The DISABLE statement disables the specified comparison limit.

NOTE

This statement provides a measurement limit comparison for the statement:

MEASURE VALUE/NODE/VARIABLE ;

Example:

Enable DC limits which pass all measured values in the range from -2 milliamperes to +2 microamperes.

```
ENABLE DCT1 GT 2E-6;
ENABLE DCT0 LT  -2E-3;
MEASURE VALUE;
```

6.2.3.8 ANALOG TO DIGITAL CONVERSION MEASUREMENTS

General Form:

```
MEASURE [VALUE/NODE number]  (,LOG);
```

Description:

This statement initiates a measurement within the precision measurement unit (PMU).

When the VALUE option is specified, the measurement is made according to the existing state of the PMU. The measured value is converted and scaled to floating-point, using a successive approximation algorithm, and stored in the system global variable, VALUE. If DC limits are enabled (see Section 6.2.3.7), VALUE is tested to determine if it falls within the pass window (defined by the previous ENABLE DCT0/1 statements, regardless of the current DCT register contents). If it fails the DC FAIL light is on at the next pause or end-of-test. Conditions required for datalogging the resulting value are described in the note at the end of this section.

If AUTO ranging has been specified (see Section 6.2.3.2), the system automatically selects the measuring range which gives best resolution. Autoranging begins with the highest range and ranges downward until best resolution is obtained or until the lowest range is reached.

Use of the NODE option provides a means for measuring a parameter at a system internal monitor node. The node numbers and their descriptions are listed in Appendix G. Ranging and measuring conditions are automatically controlled.

For internal nodes, the measured value, logging, and pass/fail conditions are the same as previously described. At the conclusion of an internal node measurement cycle, the PMU is automatically disconnected from the RVS mode and initialized to force 0 current in Range 1.

Values measured at internal nodes 130 thru 133 are actually 1/8 of the programmed values due to an internal hardware design of those RVS supplier.

Values measured for DPS current nodes (143, 144, 145) are automatically scaled so the DCT 1/DCT0 limits should be expressed in amperes for this case.

The LOG option allows datalogging to occur for only those measurements specified with the parameter LOG. Refer to Section 3 of the Sentry User's Reference Manual for a complete description of DATALOGGING.

Example:

Measure and log the current from VF1:

```
ENABLE DCT1 GT ICCMAX;
ENABLE DCT0 LT ICCOPEN;
MEASURE NODE 217B, LOG;
```

6.2.3.9 INDIRECT MEASUREMENTS

General Form:

MEASURE VARIABLE variable (,LOG);

Description:

The value of the variable, in floating point format, is stored in the system global variable: VALUE. Variable may be a variable identifier or an array element.

If DC limits are enabled (see Section 6.2.3.7), VALUE is tested to determine if it falls within the pass window. If it fails, the DC FAIL light is on at the next pause or end-of-test. The value of the variable is logged according to the Monitor logging command conditions.

The LOG option allows datalogging to occur for only those measurements specified with the parameter LOG. Refer to Section 3 of the Sentry User's Reference Manual for a complete description of DATALOGGING.

Example:

Measure and selectively log the value of variable XVAL.

```
MEASURE VARIABLE XVAL, LOG;
```

6.2.3.10 CONNECTION OF THE PMU TO FUNCTIONAL PIN DRIVERS

General Form:

[ENABLE/DISABLE] RELAY;

Description:

The ENABLE statements initializes the pin address control logic such that the driver for pin (n) remains connected, even though the precision measurement unit is connected to pin (n). This assumes a driver was previously connected by a SET DA/DB statement.

The DISABLE statement initializes the pin address control logic such that the driver for pin (n) is automatically disconnected when the precision measurement unit is connected to pin (n). If no relay statement is made, the disable mode is assumed.

Example:

Connect the precision measurement unit to pin (10) with the driver connected and then disconnect the driver (make before break sequence in order to maintain bias on a pin).

```
ENABLE RELAY;
CPMU PIN 10;
DISABLE RELAY;
```

6.2.3.11 OTHER PMU FORCE FORMS

General Form:

SET PMU [FORCEV/FORCEI] (,RNG0/,RNG1/,RNG2/,RNG3/,RNG4/,AUTO);
FORCE PMU expression;

Description:

Refer to Section 6.2.3.2 for a detailed description of the SET PMU statement.

The FORCE PMU statement is especially useful when the PMU is to be forced to several values in the same range, however, it has a slightly longer execution time than the FORCE VOLTAGE or FORCE CURRENT statements. The expression in the FORCE PMU statement is scaled according to the mode, V or I, and range. When forcing voltage, the legal ranges are 1, 2, 3, and 4. When forcing current, the legal ranges are 0, 1, 2, and 3. Refer to appendix L for range limits.

If AUTO ranging has been preset the range which gives the best resolution is automatically determined (at run time) prior to loading the forcing register (PPS).

Example:

Force the output of the PMU to -1 microamps.

```
SET PMU FORCEI,AUTO;
A=5E-6;
B=5;
FORCE PMU -A/B;
```

6.2.3.12 MACRO SEQUENCES FOR DC PARAMETERIC TESTING

Most compiler languages provide the programmer with MACRO programming capabilities - that is, the ability to write one compiler statement which causes a predefined set of operations to be performed. The way in which the MACRO call and subsequent execution sequence is performed, however, varyies between languages.

The FACTOR language provides the test programmer with the ability to write a single statement which causes a set of PMU related operations to be performed. The PMU operations, performed at execution time under control of TOPSY, constitute a complete DC parametric measurement sequence - hereafter referred to as a MACRO DC measurement sequence, or simply, a DC MACRO.

In many respects, this DC macro sequence appears very similar to a subroutine but has several distinct advantages over using a subroutine or even writing the measurement sequence in line.

A total of nine different DC MACROs are available to support a variety of commonly used DC measurement sequences. The macros provide less danger of damaging the device-under-test (DUT) and potentially more accurate test results as each DC macro is designed to provide an optimized measurement sequence. Each sequence insures that the PMU forcing and sensing ranges and modes (current and voltage) are not changed while the PMU is connected to the DUT, thereby preventing possible harmful transients to the DUT. The DC macro sequences also insure that tester pin electronics relays are switched at minimum current flow conditions, thereby prolonging the life and reliability of the test system. Many of the DC macro sequences also allow the PMU to be "pre-charged" to any desired potential before connecting to the DUT. This is useful in preventing the functional state of the DUT from being disturbed when the PMU connection is made. In addition to providing a means of preventing possible harmful transients to the DUT, pre-charging can also reduce the response time of both the PMU and DUT after the PMU connection before making the actual measurement. The latter produces a reduction in test time which results in greater test throughput, often times a critical factor for digital IC manufacturers.

When using a DC macro in place of in-line measurement sequences potential programming errors and subsequent program debugging time can be reduced since fewer statements are written. The execution time of the DC macro sequence is also generally faster than in-line sequences which, of course, results in faster program execution time.

When using a DC macro sequence in place of a FACTOR subroutine, the program execution time is faster as there is less software overhead time involved.

Macro Definition and Description

General Form:

```
SET TEST number;
MEASURE PIN number(,LOG);
```

Description:

The statement, SET TEST number, together with an immediately preceeding setup procedure provides a macro definition for performing a particular type of DC measurement. The DC measurement sequence is then initiated with the statement 'MEASURE PIN number'. Depending on the test type, the setup procedure may contain the PMU connected to a precharge pin, open, or short circuit. The PMU is forced to the desired current or voltage value in the desired sense range. The LOG option on the MEASURE PIN statement allows datalogging for the particular pin being measured.

6.2.3.13 DC MACRO MEASUREMENT

There are nine available macro DC measurements. They are:

• SET TEST 1 = VOH	Voltage output high
• SET TEST 2 = VOL	Voltage output low
• SET TEST 3 = IIH (IR)	Input high current
• SET TEST 4 = ICEX	Output leakage current
• SET TEST 5 = IIL (IFX)	Input low current
• SET TEST 6 = VCD	Input diode clamp
• SET TEST 7 = ISC	Short circuit current with outputs high
• SET TEST 9 = IOL	Output current with outputs low
• SET TEST 10 = VBD	Voltage breakdown

Each macro is discussed below along with the resultant macro measurement sequence and the state of the PMU after the MEASURE PIN number statement. A sample setup is provided for a VOH test to illustrate how the macro works.

1. SET TEST 1; OUTPUT HIGH VOLTAGE TEST (VOH)

   Set-up Procedure:

```
ENABLE RELAY;
CPMU PIN r;                 REM CONNECT TO REST
                            PIN - PRECHARGE;
FORCE CURRENT i, RNGx;
SET PMU SENSE, RNGy;
SET TEST 1;                 REM SELECT TEST MACRO;
ENABLE DCT0 LT lower limit;
ENABLE DCT1 GT upper limit;
```

```
MEASURE PIN t;          REM EXECUTES THE TEST
                        MEASUREMENT
MACRO ON PIN T;
```

Macro Measurement Sequence:

```
CPMU PIN t;
DISABLE RELAY;
FORCE CURRENT i, RNGx;
MEASURE VALUE;          REM PERFORM MEASUREMENT;
FORCE CURRENT 0, RNGx;
ENABLE RELAY;           REM RECONNECT THE REST PIN;
CPMU PIN r;
```

State of PMU after MEASURE PIN t statement:

FORCING: 0 Current
PIN: Set up pin

2. SET TEST 2; OUTPUT LOW VOLTAGE TEST (VOL)

Set-Up Procedure:

```
ENABLE RELAY;
XPMU PIN;
FORCE CURRENT i, RNGx;
SET PMU SENSE, RNGy;
SET TEST 2;
ENABLE DCT1 GT limit;
ENABLE DCT0 LT limit;

MEASURE PIN t;
```

Macro Measurement Sequence:

Same as SET TEST 1.

State of PMU after MEASURE PIN t statement:

FORCING: 0 Current
PIN: Set up pin

3. SET TEST 3; INPUT HIGH CURRENT TEST (IIH (IR))

Set-up Procedure:

```
DISABLE RELAY;
XPMU PIN;
FORCE VOLTAGE v, RNGx;
SET PMU SENSE, RNGy;
SET TEST 3;
ENABLE DCT0 LT limit;
ENABLE DCT1 GT limit;

MEASURE PIN t;          REM EXECUTE TEST MEASUREMENT
                        MACRO;
```

Macro Measurement Sequence:

```
CPMU PIN t;
FORCE DELAY;
MEASURE VALUE;          REM PERFORM MEASUREMENT;
```

State of PMU after MEASURE PIN t statement:

FORCING: vih Voltage
PIN: t

4. SET TEST 4; OUTPUT LEAKAGE CURRENT TEST (ICEX)

The setup procedure and measurement sequence and PMU state are identical to SET TEST 3; IIH (IR). It was originally designed to test a tri-state TTL device with an open collector on the output and the output transistor in the "off" condition.

5. SET TEST 5; INPUT LOW CURRENT TEST (IIL(IFX))

Set-up Procedure

```
DISABLE RELAY;
XPMU PIN;
FORCE VOLTAGE v, RNGx;
SET PMU SENSE, RNGy;
SET S1 vih, RNGx;          REM SET REFERENCE S1 SLIGHTLY
                           GREATER THAN DEVICE VIH ON
                           SAME RANGE AS PMU FORCE;

SET TEST 5;
ENABLE DCT0 LT limit;
ENABLE DCT1 GT limit;

MEASURE PIN t;             REM EXECUTE TEST MEASUREMENT
                           MACRO;
```

Macro Measurement Sequence:

```
CPMU PIN t;
FORCE VOTLAGE v, RNGx;
MEASURE VALUE;
FORCE VOLTAGE vih, RNGx;
XPMU PIN;                  REM IF XPMU PIN WAS GIVEN IN THE
                           SET-UP PROCEDURE, OTHERWISE
                           THE LAST PIN NUMBER;
```

State of PMU after MEASURE PIN t statement:

FORCING: vih Voltage
PIN: t

6. SET TEST 6; INPUT DIODE CLAMPS (VCD)

Set-up Procedure:

```
ENABLE RELAY;
XPMU PIN;
FORCE CURRENT i  RNGx;
```

```
SET PMU SENSE, RNGy;
SET TEST 6;
ENABLE DCT0 LT limit;
ENABLE DCT1 GT limit;

MEASURE PIN t;          REM EXECUTE TEST MACRO;
```

Macro Measurement Sequence:

```
CPMU PIN t;
DISABLE RELAY;
FORCE CURRENT i, RNGx;
MEASURE VALUE;
FORCE CURRENT 0, RNGx;
ENABLE RELAY;
```

State of PMU after MEASURE PIN t statement:

```
FORCING: 0 Current
PIN: t
```

7. SET TEST 7; OUTPUT SHORT CIRCUIT CURRENT TEST (ISC)

Set-up Procedure:

```
DISABLE RELAY;
XPMU PIN;
FORCE VOLTAGE v, RNGx;
SET PMU SENSE, RNGy;
SET S1 voh, RNGx;    REM SET REFERENCE S1 SLIGHTLY
                     GREATER THAN VOH ON SAME RANGE AS
                     PMU FORCE;
SET TEST 7;
ENABLE DCT0 LT limit;
ENABLE DCT1 GT limit;

MEASURE PIN t;       REM EXECUTE TEST MACRO;
```

Macro Measurement Sequence:

```
CPMU PIN t;
FORCE VOLTAGE v, RNGx;
MEASURE VALUE;
FORCE VOLTAGE voh, RNGx;
XPMU PIN;            REM IF XPMU PIN WAS GIVEN IN THE SET
                     UP PROCEDURE, OTHERWISE THE LAST
                     CONNECT PIN NUMBER;
```

State of PMU after MEASURE PIN t statement:

```
FORCING: voh voltage
PIN: Set up pin
```

8. SET TEST 8; Not Available

9. SET TEST 9; OUTPUT LOW CURRENT TEST (OUTPUTS LOW (IOL))

Set-up Procedure

```
DISABLE RELAY;
XPMU PIN
FORCE VOLTAGE v, RNGx;
SET PMU SENSE, RNGy;
SET S0 vol, RNGx;      REM SET REFERENCE S0 SLIGHTLY LESS
                       THAN DEVICE VOL ON SAME RANGE AS
                       PMU FORCE;
SET TEST 9;
ENABLE DCT0 LT LIMIT;
ENABLE DCT1 GT LIMIT;
MEASURE PIN t;         REM EXECUTE TEST MACRO;
```

Macro Measurement Sequence:

```
CPMU PIN t;
FORCE VOLTAGE v, RNGx;
MEASURE VALUE;
FORCE VOLTAGE vol, RNGx;
XPMU PIN;
```

State of PMU after MEASURE PIN t statement:

```
FORCING:  vol Voltage
PIN: Set up pin
```

10. SET TEST 10; VOLTAGE BREAKDOWN (VBD)

Set-up Procedure:

```
XPMU PIN;              REM SAME PROCEDURE AS TEST 6;
ENABLE RELAY;
FORCE CURRENT i, RNGx;
SET PMU SENSE, RNGy;
SET TEST 10;
LIMIT 2;
ENABLE DCT1 GT LIMIT 2;
MEASURE PIN t;
```

Macro Measurement Sequence

```
CPMU PIN t;            REM SAME PROCEDURE AS TEST 6;
DISABLE RELAY;
FORCE CURRENT i, RNGx;
MEASURE VALUE;
FORCE CURENT 0, RNGx;
ENABLE RELAY;
```

State of PMU after MEASURE PIN t statement:

```
FORCING: 0 Current
PIN: t
```

EXAMPLE:

A sample setup for a VOH test is as follows:

```
ENABLE RELAY;
CPMU PIN 16;                REM PIN 16 = VCC FOR PRECHARGE;
FORCE CURRENT -360E-6, RNG2;
SET PMU SENSE, RNG2;
SET TEST 1;
ENABLE DCT1 GT 3.5;
ENABLE DCT0 LT 2.7;
MEASURE PIN 12;
```

The execution of the SET TEST 1 statement causes the connected test pin, pin 16, and the forcing mode and value, current -360E- 6 RNG2, to be saved. In addition, when current is being forced, the PMU is reset to force zero current on the same range whenever relay switching occurs. The statement MEASURE PIN 12 causes the following measurement sequence to be performed.

```
CPMU PIN 12;                        REM THE PMU IS SWITCHED FROM
DISABLE RELAY; †                    THE PRECHARGE PIN TO THE PIN
FORCE CURRENT 360E-6, RNG2;         UNDER TEST;

MEASURE VALUE;                      REM THE MEASUREMENT IS MADE;

FORCE CURRENT 0,RNG2;               REM THE PMU IS RECONNECTED
ENABLE RELAY; †                     TO THE REST PIN WHILE THE PMU
CPMU PIN 16;                        IS FORCING 0 CURRENT;
```

NOTE

The FACTOR statement MEASURE PIN is different from MEASURE PIN number. MEASURE PIN is retained for performing a go/no–go comparison on the currently connected tester pin. The MEASURE VALUE performed by the macro does not update the variable VALUE.

†The Enable Relay-Connect-disable relay sequence is used to allow the input pin buffer of the pin electronics to clamp the current forcing PMU when initially connected to the pin. That is, a make-before-break sequence between the PMU relay and D relay occurs. To make use of this feature, the pins must be previously defined as input by the SET D instruction.

6.2.3.14 PROGRAMMABLE PMU VOLTAGE CLAMP

General Form:

SET CLAMP [SYM/POS/NEG] number;
SET CLAMP OFF;

Description:

The purpose of the SET CLAMP statement is to define a limited range within which PMU voltages may occur when the PMU is forcing a current into a load.

The PMU voltage clamp may be used when forcing voltage (to protect a device from a programming error) or when forcing current and sensing voltage.

POS means no voltages less than -0.7 volts are allowed and NEG means that no voltages greater than 0.7 volts are allowed. SYM allows both positive and negative voltages to occur. At the other limit, number defines the absolute value of the maximum PMU voltage which is allowed.

There are 30 values at which the PMU may be clamped (positive or negative). They are:

1.5, 3.0, 4.5, 7.5, 9.0, 10.5, 13.5, 15, 16.5,
19.5, 21.0, 22.5, 25.5, 27.0, 28.5, 31.5, 33.0,
34.5, 37.5, 39.0, 40.5, 43.5, 45.0, 46.5, 51.0,
57.0, 63.0, 60.0, 75.0, 81.0, 87.0, 93.0, volts.

Any value is accepted, however, the next higher (if not equal) clamp value is selected.

Example:

SET CLAMP POS 8.2;

The allowed voltages from the ~~command~~ STATEMENT are between -0.7 volts and 9.0 volts. The actual clamp values produced by the hardware are +1 volt or ±10% of the value specified. When sensing voltage, extra delay should be provided for clamp stabilization for each FORCE CURRENT statement executed.

6.2.4 Analog System Time Delays

There are two sources of time delays to allow for relay and analog settling times. One source is fixed delays which are appropriate for whatever hardware activities need them. The second source is a user programmable time delay generator to allow for special cases where increased settling time is needed for best accuracy. Refer to Appendix D.

6.2.4.1 INITIALIZING PROGRAMMABLE TIME DELAYS

General Form:

SET DELAY expression, DC;

Description:

The value of the required time delay is loaded in the tester time delay register. The resolution is 0.35 milliseconds with a maximum value of 5.734 seconds. When a DC delay is invoked, the Tester Busy status remains "ON" for the amount of time defined by the SET DELAY statement or for the fixed time delay generated by the hardware, whichever is greater. The DC delay is used in execution of FACTOR statements for power supplies and precision measuring unit programming (refer to Section 6.3). It is also used for expressing MATCH or EXT SYNC mode time out duration (refer to Section 6.5.2.4).

Example:

SET DELAY 0.005, DC;

6.2.4.2 TIME DELAY GENERATION

General Form:

FORCE DELAY;

Description:

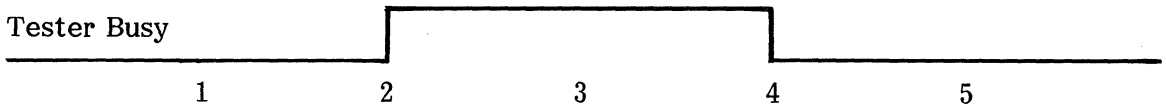
This statement forces TESTER BUSY for the time initialized by the last "SET DELAY" statement that was executed. No time-delay-dependent statement can be executed during TESTER BUSY. Hence if a time delay dependent statement follows a FORCE DELAY statement then a time delay is actually encountered. If the statement following the FORCE DELAY is a non-time delay dependent statement it is executed immediately regardless of the tester busy/not busy condition. Then the next statement is examined for execution.

Example:

Provide a 3 millisecond delay after connecting pin 10 to provide settling time for the change of programmed voltage prior to executing a MEASURE VALUE.

```
SET DELAY 3 E-3, DC;
CPMU PIN 10;
FORCE DELAY;
MEASURE VALUE;
```

The following diagram shows the sequence of the statements listed in reference to a timing diagram.



1. SET DELAY 5E-3, DC;
2. FORCE DELAY;
3. A=B+C;
4. MEASURE VALUE;
5. RESULT=VALUE-A;

6.2.4.3 TIME DELAY DEPENDENCY

General Form:

FORCE WAIT;

Description:

This statement forces the tester to wait until the status of the tester is not-busy before processing the next statement. This statement is not necessary prior to time delay statements such as MEASURE VALUE.

Example:

Provide a delay until the tester is not busy after forcing a power supply voltage.

FORCE VF1 5.0, RNG2;
FORCE WAIT;

6.2.5 Miscellaneous Analog Subsystem Statements

6.2.5.1 DEFINING PROGRAM OFFSET VOLTAGES

General Form:

SET VOFFSET number;

Description:

This statement allows the specification of a voltage offset value which is automatically added to the values programmed by all subsequent voltage forcing statements.

The VOFFSET level can be used as a reference level for all other forcing voltages, thus reducing the programming task and improving the readability of the listing.

If range modifiers are used, they must be consistent with the sum of the programmed voltage and VOFFSET.

Example:

If a supply absolute voltage range is +6 to -30 volts (5MHz) and a device requires greater than +6 volts with voltage swings within the 36 volt range, the offset value and supplies can be programmed as follows:

SET VOFFSET -18;
VSS=20;
FORCE VF1 0, RNG 3;
FORCE VF2 VSS, RNG 2;

The actual voltages programmed are:

VF1=-18 volts
VF2=+2 volts

The voltage statements affected by the voltage offset value are listed in Table 6-4.

TABLE 6-4 VOLTAGE STATEMENTS

FORCE [VF1/VF2/VF3] expression (,RNG1/,RNG2/,RNG3);
FORCE [E0/E1/EA0/EA1/EB0/EB1/EC0/EC1] expression (,RNG1/,RNG2/,RNG3);
FORCE PMU expression;
FORCE VOLTAGE expression (RNG1/,RNG2/,RNG3/,RNG4);
SET [S0/S1/SA0/SA1] expression (,RNG1/RNG2/,RNG3);
ENABLE [TRIPV1/TRIPV2/TRIPV3] [LT/GT] expression (,RNG2/,RNG3);
SET DCT [LT/GT] expression (,RNG0/,RNG1/,RNG2/,RNG3/,RNG4);
ENABLE [VLO/VHI] [LT/GT] number;
ENABLE [DCT0/DCT1] [LT/GT] expression;
Note: Voltage offset does not apply to the SET CLAMP statement as there is no D to A converter in the hardware.

6.2.5.2 ESTABLISHING ABSOLUTE PROGRAM ANALOG LIMITS

General Form:

ENABLE [ILO/IHI/VLO/VHI] [GT/LT] number;

Description:

This statement enables limit comparisons to be made on all programmed current/voltage operands prior to an instruction execution. If the operand fails to be within the LIMIT bounds, a system terminal error of 21 or 22 is issued. (Refer to Appendix E for terminal error numbers and descriptions.)

Statements with operands in the LIMIT bounds are executed. The "allowed" regions are established by the LIMIT pairs ILO and IHI for currents and by the LIMIT pairs VLO and VHI for voltages. The absence of ENABLE LIMIT statements in a program allows all magnitudes less than the hardware range limits to be programmed. The function of the limit compairsons is to protect the device under test where source forcing parameters are calculated or may be unknown. Once the program is operational and safe parameters are known, these statements may be removed for execution time efficiency.

Examples:

Enable limits to "pass" voltages which are between the values of +5.0 and 0 volts.

```
ENABLE VHI GT 5;
ENABLE VLO LT 0;
```

Enable limits to pass currents which are between the values of +100mA and -5mA.

```
ENABLE IHI GT 100E-3;
ENABLE ILO LT -5E-3;
```

6.3 FUNCTIONAL TEST TIMING SUBSYSTEMS

This section describes the FACTOR programming statements for the Sentry function test timing subsystem. The timing subsystem consists of the Test Rate Generator and delay/width generators.

6.3.1 Time Delay and Width Generators

Time delay and width generators are used to define functional test driver switching points of an input waveform and output sampling times.

6.3.1.1 DEFINING TIMING GENERATOR DELAY AND WIDTH

General Form:

SET TGx [DELAY/WIDTH] expression (,RNG0/,RNG1/,RNG2/,RNG3);

Description:

Eight timing generators are available for use, six of which may be used for data and clock timing (TG1 through TG6) and two of which may be used for strobing (TG7 and TG8).

This statement set provides the user with the capability of programming the delay and width of any one of the timing generators. The full range for pulse delay and pulse width is 10 nsec to 10 msec with a minimum resolution of .16 nsec shown in Table 6-5.

TABLE 6-5 TIMING GENERATOR RANGE SCALE AND RESOLUTION

Range	Full-Scale	Resolution
0	10 usec.	.16 nsec
1	100 usec.	100 nsec.
2	1 msec.	1 usec.
3	10 msec.	10 usec.

If range is not specified in the statement, the lowest range consistent with the specified magnitude is used.

6.3.1.2 TIMING RANGE RESTRICTIONS

The range of pulse width must be less than or equal to the range of pulse delay, since all time base outputs are synchronized to T0 rather than to the end of delay. The range of the pulse delay must be less than or equal to the range of the period.

Pulse width and pulse delay must be less than or equal to test period as programmed by the statement SET PERIOD.

6.3.1.3 ASSIGNING TIMING GENERATORS TO PINS

General Form:

CGEN [TG0/TG1/TG2/TG3/TG4/TG5/TG6/TG12] decimal pin list;

Description:

There are eight timing generators available, two of which are used for generating comparator strobes (refer to Section 6.4.4.1 and Section 6.3.2.1). Any one of the remaining six (TG1 to TG6) may be connected to one or more tester pins specified in the decimal pin list by issuing this statement. These six timing generators are used for data and clock timing during functional testing.

Connecting a pin to TG0 results in that pin being clocked at reference time T0. Pins not connected to a timing generator are clocked at time T0.

The decimal pin list is comprised of one or more decimal pin numbers. Connecting a specific pin to a timing generator causes that pin to be automatically disconnected from any other timing generator. Otherwise, a pin remains connected until it is specified in a CGEN TG0 statement. Figure 6-1 illustrates timing generator pulse generation.

Connecting a pin whose mode is RZ to TG12 results in that pin being provided with an effective input of the logical OR of TG1 and TG2 which produces a double pulse. Double pulse has no effect on pins whose mode is NRZ, as the definition of NRZ is retained.

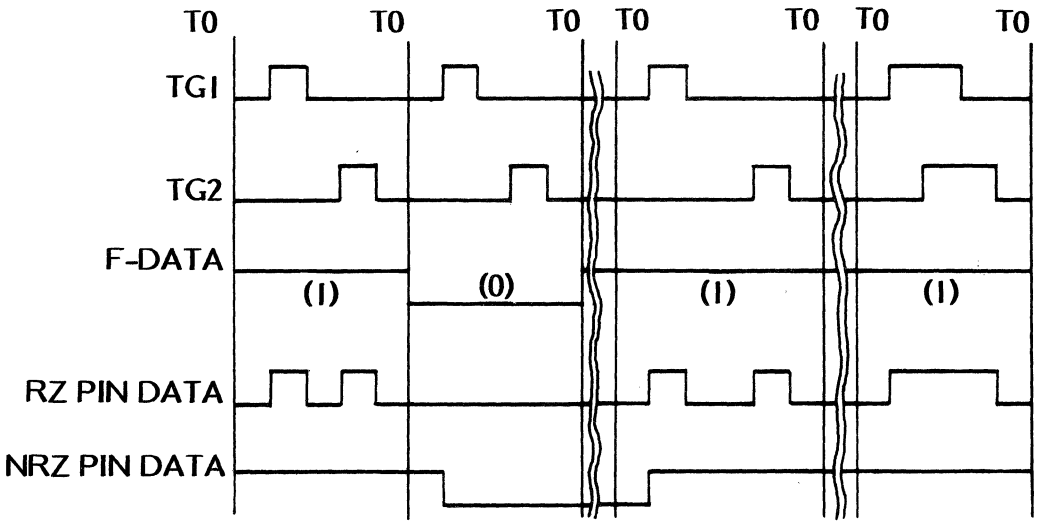


Figure 6-1 Timing Generator Pulse Generation For TG12

6.3.2 Test Rate Generator

6.3.2.1 DEFINING TEST RATES

General Form:

SET PERIOD expression (,RNG0/,RNG1/,RNG2/,RNG3);

Description:

This statement is required prior to an ENABLE TEST statement and defines the functional test period or test rate between 100 nsec and 40 msec. The full-scale value and resolution for each range is shown in Table 6-6.

TABLE 6-6 PERIOD RANGE, SCALE AND RESOLUTION

Range	Scale	Resolution
0	40 usec	10 nsec
1	400 usec	100 nsec
2	4 msec	1 usec
3	40 msec	10 usec

The period and range specified is tested for consistency at execution time. If no range is specified, the range giving the best resolution for the selected period is chosen by the compiler. Full scale values are 10 usec, 100 usec, and 1 msec for ranges 0, 1, and 2 respectively for autoranging. For example, in order to obtain the finest resolution for periods greater than 10 usec and less than 40 usec, the RNG0 modifier must be used. Otherwise autoranging selects RNG1. The autorange points are set to best prevent range violations as described in Section 6.3.2.2.

If a pin in the NRZ mode is driven by a timing generator with a delay plus width exceeding the test period, the F-data becomes 0 at T0 on a 1 to 0 transition of memory data. Figure 6-2 shows data and pulse response for RZ and NRZ F-data during set periods.

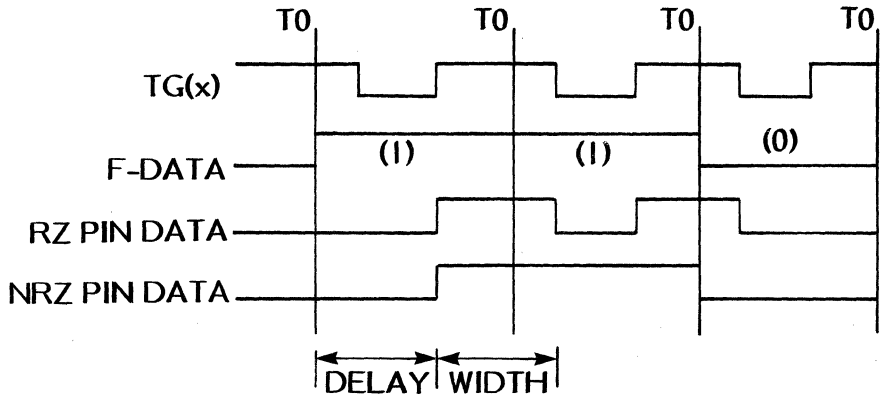


Figure 6-2 Period and Pulse Generator For Pulse Exceeding Period

6.3.2.2 RANGES AND RESTRICTIONS

The value of the test rate must be greater than or equal to the pulse width or the pulse delay values specified for any programmed timing generator. Also the range of the test rate generated must be greater than or equal to the range of range of the test rate generated must be greater than or equal to the range of any delay generated.

Specifications less than 100 nsec (200 nsec for a 5 MHz test station) or more than 40 msec for period, results in a terminal error message.

6.4 PIN CONTROL LOGIC - FORMATTING AND FAIL RESPONSE

6.4.1 Long Register Formatting

This section describes two basic forms of tester statements which fundamentally manipulate tester pin registers interfaced to the tester long register data bus.

The binary register (long registers) are used to set up local memory and other hardware conditions for functional testing.

The general format of the statements for programming functional test conditions is as follows:

SET register (*) binary pin pattern (,binary pin pattern,...);

where the register may be one of the following:

DA,DB,MA,MB,S,R,F,INVERT,I,STROBE,RZ, or XOR

The binary pin pattern is defined by a binary value which has a one to one correspondence between the pin and the pattern bit location. Only pins that change from the previous state need be specified by the SET statement. Even if all pins are specified and the asterisk is omitted (refer to Section 6.4.1.4 for asterisk notation), only codes for the ranks (refer to Appendix B) in which pin data has changed are generated for maximum test rate. In the first statement of a type executed by a program, or for SET F statements the first of each local memory load, all pin states not specified are assumed to be zero.

6.4.1.1 PIN ORIGIN AND PATTERN REPLICATION

The binary pin pattern can be programmed by either specifying each bit of the pattern or by using one of the following operators:

[n] Pin origin operator where n is an integer.
(m:bpp) Binary pin pattern replicator operator, where m is an integer and bpp is the binary pin pattern.

A sequence of binary pin patterns seperated by commas represent a series of functional tests. To better illustrate the use of these statements, two examples are shown below. The first example uses the pin origin and replicator operators. The second example yields the same binary pin patterns, however, it does not make use of the operators.

Example 1:

```
SET F (3:0)      (13:1)     (2:0),
      (3:101)    010        (6:1),
      [8]        (2:1)      (9:0);
```

Example 2:

```
SET F 000  111  111  111  111  100,
      101  101  101  010  111  111,
      101  101  111  000  000  000;
```

Explanation:

(3:0) means three 0 values specified
(13:1) means thirteen 1 values specified
(2:0) means two values of 0 specified
(3:101) means three sets of 101 values
010 means set the next three pins to this pattern
(6:1) means six values of 1 specified
[8] means preserve the previous pattern and start at pin 8 with the following specifications (Note the pins are numbered starting with 1.)
The (2:1) and (9:0) follow the same format as the other pattern replicators specified

It should be noted that blanks are ignored within the binary pin patterns, however, when using the pattern replicator no blanks are allowed within the binary pin pattern. A system error message is issued if blanks are entered.

6.4.1.2 RANK ORGANIZATION

The tester is organized into ranks of 15 pins each, as follows:

rank 1 = pins 1-15 and ENABLE DA/DB
rank 2 = pins 16-30 and ENABLE MA/MB
rank 3 = pins 31-45
rank 4 = pins 46-60

When loading the long registers, all 15 pins of the rank are loaded simultaneously from one 24 bit word of the CPU's memory. Due to this, all 15 pins must be determined by the compiler in order to fully define the rank. All pins not specified in a given statement are assumed to be zero or in the last defined state.

The SET FI and SET SI statements also operate on a rank basis; however, since they are interpretive rather than DMA, they are first read back during test execution and the specified pin definitions only are changed.

6.4.1.3 COMPILER GENERATION OF MINIMUM DATA

The compiler minimizes the number of ranks of data generated in order to reduce the size of the test program, and also reduce the time to load local memory.

The following decisions are made in the compiler:

(1) If all pins of any rank are in the same state as the last SET register and there is no asterisk, then no code is generated for that rank.

Example:

```
SET F (60:1);              REM GENERATE RANK 1,2,3,4;
SET F (30:0) (30:1);       REM GENERATE RANK 1,2 ONLY;
```

If there are no pin changes in any rank, code for rank 1 is generated.

(2) If the statement contains an asterisk, then data is generated for each rank for which a pin is specified, even if it is not a change in the pin condition.

Example:

```
SET F (60:1);                REM GENERATE RANKS 1,2,3,4;
SET F* (30:0) (30:1);        REM GENERATE RANKS 1,2,3,4;
SET F* [1]  1  [45]  0;      REM GENERATE RANKS 1,3;
```

(3) Regardless of pin data, a SET F following a change of the enabled D register (ENABLE DA/DB) causes code to be generated for rank 1. A change of the enabled M register (ENABLE MA/MB) causes code to be generated for rank 2.

(4) All ranks are automatically generated at the first SET F of the program, the first SET F following an AT statement, after an ENABLE TEST, and after SPM statements loading local memory.

A four rank holding register is used for loading local memory. Thus, ranks that are not loaded from CPU memory are replicated into each local memory location.

(5) Unless defined otherwise, the compiler generates a pin state of binary 0 for the following conditions:

(1) beginning of the program
(2) after a FORCE RESET statement
(3) after an AT statement
(4) after an ENABLE TEST statement

6.4.1.4 ASTERISK NOTATION

General Form:

SET register* binary pin pattern (,binary pin pattern);

Description:

The asterisk form of the statement forces data codes to be generated for all ranks in which pins are specified. It should be used whenever the flow of control is altered.

Example:

Statement Number		Statement	Compile Ranks
1		SET S (30:1);	1 and 2
2		SET S (29:1)0;	2
3	X:	SET S* (30:1);	1 and 2
.		.	
.		.	
.		.	
n		SET S (30:0);	1 and 2
		GOTO X;	

Explanation:

Statement 1 is included in order to get the S register into a known state. Since pins 1 to 15 are already 1's, only rank 2 is generated for statement 2. Without the asterisk in statement 3, only rank 3 would be generated for the same reason, however, since the first 15 pins are 0's at statement n, execution of statement 3 as the result of a branch statement from statement n+1 would be an error. Using the asterisk form of the SET S statement corrects the problem in the above example

The asterisk form with all the pins specified should always be used after a subroutine call, subroutine entry, conditional statement, or after a label in order to force generation of code for all ranks and account for any changes which may have occurred outside of the mainstream flow of the program.

Example:

Statement	Compiled Data
SET F (60:1);	06077777 06177777 06277777 06377777
SET F (59:1) 0;	26337777
CALL X;	
SET F* (60:1);	06077777 06177777 06277777 26377777

Explanation:

In the above example, note the second SET F statement generated only one word. Normally, words are generated only for ranks with data that changes. However, the fourth statement follows a subroutine call and, at run time, subroutine X may alter the F register. To insure that all 60 pins were returned to "one", the programmer used an asterisk to force the compiler to generate data for all the ranks explicitly mentioned in the statement.

6.4.1.6 MINIMUM PIN DEFINITION

General Form:

SET MPIN number;

Description:

This statement allows a user to define the maximum pin count allowed in their FACTOR program and to control the maximum number of ranks of long register data that the compiler generates. If subsequent pin-related statements contain a pin count greater than the specified limit, the error message:

NUMBER EXCEEDS LIMIT

is issued for each statement.

Example:

```
000001        SET PAGE 1024;
000002        SET MPIN 30;
000002        SET F  25  1010101;
NUMBER EXCEEDS LIMIT          ↑
000002        END;
0001B         COMPILATION ERRS
```

6.4.1.7 PIN LIST FORMAT - PROGRAMMING CONDITIONS

This discussion explains the compiler handling of the following tester statements:

```
CONN CLK
XCON PIN
CGEN TGx
```

(1) The compiler keeps a buffer in memory of all clock, DPS, and timing generator conditions. This allows the compiler to generate the correct code when reassignments are made. For example:

```
CGEN TG1    1,2,3;
.
.
.
CGEN TG1    4,5,6;
REM PINS 1 THRU 6 ARE NOW CONNECTED TO TG1;
```

(2) The compiler makes the pin assignments in the order of presentation to the compiler. This can cause complications when the program is executed in a different sequence. For example:

```
IF A EQ 0 THEN GOTO LABEL;
CGEN TG1    1,2,3;
LABEL:
CGEN TG1    4,5,6;
REM PINS 1 THRU 6 ARE NOW CONNECTED TO TG1;
```

(3) The compiler assumes that all assignments will occur, even if a conditional expression causes the tester statement to be skipped at execution time. For example:

```
IF A EQ 0 THEN CGEN TG1  1,2,3
          ELSE CGEN TG1  4,5,6;
REM PINS 1 THRU 6 ARE NOW CONNECTED TO TG1;
```

(4) A possible programming technique to avoid the pitfalls of examples 2 and 3 is to program the disconnection. For example:

```
IF A EQ 0 THEN CGEN TG1  1,2,3
          ELSE BEGIN
CGEN TG0 1,2,3;
CGEN TG1 4,5,6;
REM PINS 4 THRU 6 ARE NOW CONNECTED TO TG1;
END;
```

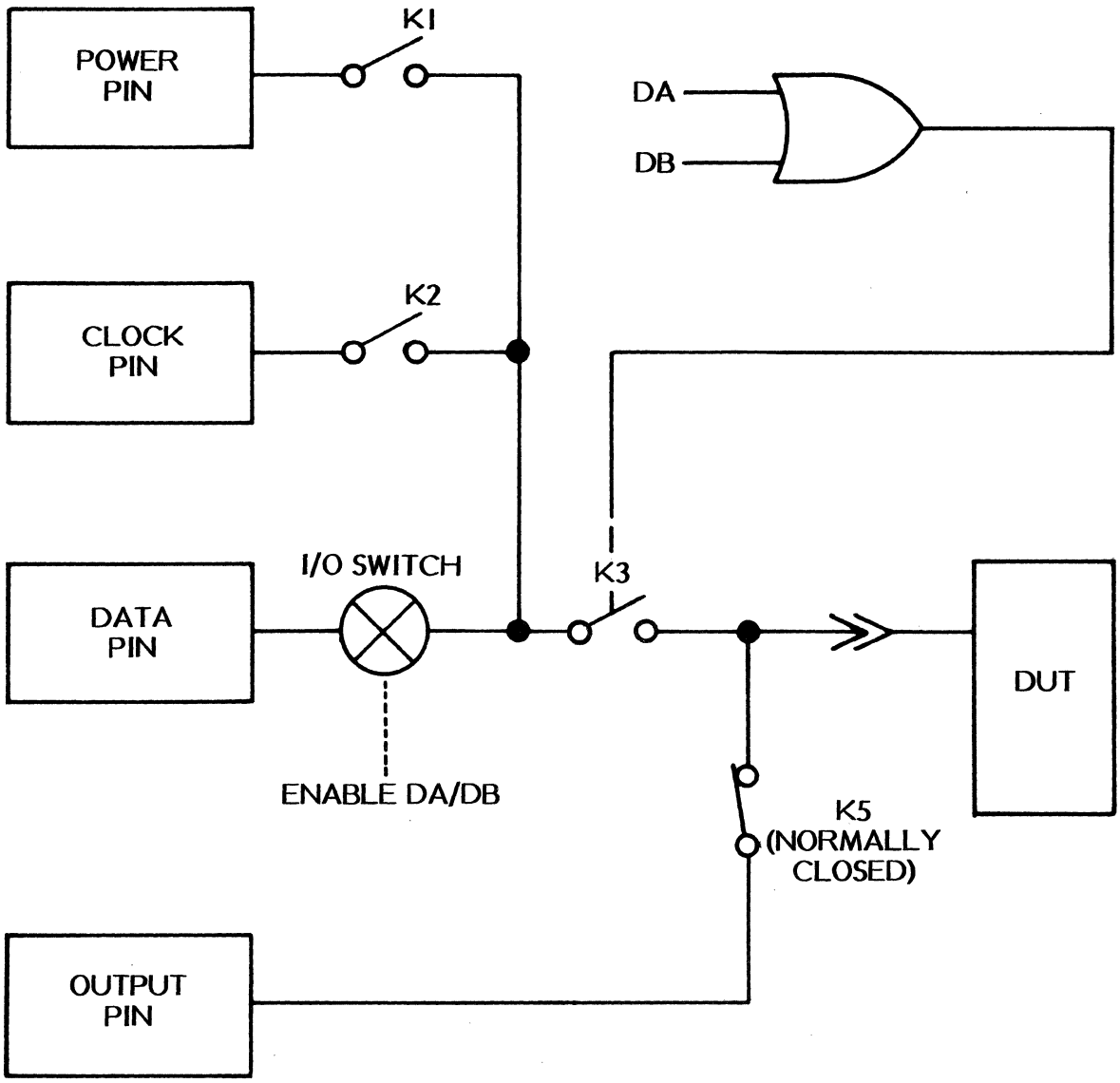


Figure 6-3 Pin Relay Sequence

6.4.2 Input Pin Definition Registers

There are two registers which provide the capabilities of defining input pins. They are the DA and DB registers. Only one of the two register is effective or enabled at any time during a functional test execution. This implementation allows the high speed functional pattern to control the connection or disconnection of pin electronics drivers from the pin under test. The selection of which input definition register is active is under control of the local memory or other optional pattern generators. In the case of local memory operation the ENABLE DA/DB statement (Section 6.4.2.2) defines the selection of DA or DB within the functional test truth table.

6.4.2.1 DEFINING INPUT PINS

General Form:

SET [DA/DB] binary pin pattern;

Description:

This statement provides the capability for setting either the primary DA or alternate DB pin definition register. The I/O pin definition register in use at any particular time may be specified by the ENABLE DA/DB statement.

This statement loads one of the input definition registers. A binary 1 in the pattern field declares the associated test pin to be an input pin and a binary 0 in the pattern field declares the associated pin not to be an input pin. The OR of the pin data from the DA and DB registers energizes a relay (K3) through which inputs may be programmed (refer to figure 6-3 for relay sequence). If the DA register contains a binary 1 for a certain pin and ENABLE DA is effective in the functional test, the high speed I/O switch in the pin electronics for that pin is on, thus completing the connection of the functional driver to the device pin. If the DA or DB register contains a 0 for a certain pin, and that DA or DB register is effective then the high speed I/O switch is off (open) thus disconnecting a data driver from the device pins. In any case the test pin remains connected to the corresponding pin voltage level detector.

6.4.2.2 SELECTION OF INPUT DEFINITION REGISTER FROM LOCAL MEMORY

General Form:

ENABLE [DA/DB] (,MA/MB);

Description: Default condition

This statement selects input definition register and optionally selects the pin mask register to be used in subsequent functional tests (in the order in which the D or M registers appear in the program). When executing a test pattern, changes in enabling DA or DB always occur at time T0 (the beginning of a test cycle period). DA is enabled prior to execution of this statement.

This statement generates no object code but is merely an informational statement for the compiler and affects the generation of subsequent SET F statements. If the statement is referenced by a local memory label, the label refers to the next available location in local memory. Therefore, this statement must be immediately followed by at least one SET F statement.

Refer also to Section 6.4.3.2 for the form and description of the ENABLE MA/MB statement for selection of the output mask definition registers.

6.4.3 Output Mask Definition Registers

There are two registers which provide the capabilities of defining output pins. They are the MA and MB registers. Either one of the two registers is effective or enabled at any time during a functional test exercise. This implementation allows the high speed functional pattern to control the enabling and disabling of pin electronics comparators. The selection of which mask definition register is active is under control of the local memory or other optional pattern generators. For local memory usage ENABLE MA/MB statement defines the selection of MA or MB within the functional truth table.

6.4.3.1 DEFINING CARE PINS

General Form:

SET [MA/MB] binary pin pattern;

Description:

The function of this statement is to allow the setting of the primary MA or alternate MB pin mask register. The pin-mask register in use at any particular time it is specified by the ENABLE MA/MB statement.

This statement loads one of the mask registers. A binary 1 in the binary pin pattern enables the associated pin level detector. A binary 0 in the pin pattern disables the associated pin level detector. These two states are referred to respectively as the "care" and "don't care" conditions. The functional test results (pass/fail) are strobed from the level detector outputs into the comparison register (C) for "care" pins. The transfer of the fail signal into the C register is inhibited for "don't care" pins.

6.4.3.2 SELECTION OF OUTPUT MASK REGISTER FROM LOCAL MEMORY

General Form:

ENABLE [MA/MB] (,DA/DB);

Description: Default condition

This statement selects the pin mask register and optionally selects the input definition register to be used in subsequent functional tests (in the order in which they appear in the program). MA is enabled prior to execution of this statement.

This statement generates no object code but is merely an informational statement for the compiler and affects the generation of subsequent SET F statements. If the statement is referenced by a local memory label, the label refers to the next available location in local memory. Therefore, this statement must be immediately followed by at least one SET F statement.

Refer to Section 6.4.2.2 for the form and description of ENABLE DA/DB statement for the selection of the input definitions register.

3.4.3.3 MASK CONTROL RELATING TO I/O DEFINITION

General Form:

[ENABLE/DISABLE] IMASK;

Description:

This statement provides automatic masking (i.e., specifying "don't care") of all input pins. The input pins are defined by either the SET DA or SET DB statements, whichever is effective. Any pin programmed as an input pin in the enabled D register is not functionally tested as long as ENABLE IMASK is active, regardless of the contents of the enabled M register.

Note IF Dn=0 then Case only IF MA/B = 1

6.4.4 Output Comparator Strobes

Timing generators 7 and 8 are used to define the time during which output pin comparator data is sampled (or strobed) and compared to expected results.

6.4.4.1 STROBE TIMING GENERATOR SELECTION

General Form:

SET STROBE binary pin pattern;

Description:

This statement may be used to select one or both of two possible strobe times for each tester pin. A binary 0 in the pattern field connects TG7 to the pin, while a binary 1 connects TG8 to the pin.

If the statement ENABLE DOUBLE STROBE is executed by the program, then all pins indicated by a binary 0 is strobed by both TG7 and TG8. A single strobe (TG8) still exists on pins indicated by a binary 1 in the pattern field.

If either TG7 and TG8 is being used for strobing, they must have a programmed delay and width (refer to Section 6.3.1.1). The strobe window is equal to the programmed width.

6.4.4.2 DOUBLE STROBING

General Form:

[ENABLE/DISABLE] DOUBLE STROBE;

Description:

This statement is used in conjunction with the SET STROBE statement described above. When double strobing is enabled, all pins may be strobed by both TG7 and TG8. The composite strobe pattern allows failures to be detected during both of the strobe windows. All pins with a binary 1 in the pin pattern are strobed by TG8 only, overriding the DOUBLE STROBE statement.

6.4.5 Input Waveform Control

This section describes the FACTOR statements which define the functional test pattern and associated waveform format controls.

6.4.5.1 DEFINITION OF FUNCTION DATA

General Form:

SET F binary pin pattern (,binary pin pattern,...);

Description:

This statement loads a binary functional test pattern into the local memory in the high speed test station controller. After each SET F statement, the local memory load address register (MCS) is incremented to the next location.

The binary pin pattern consists of binary 1's and 0's which have a one to one correspondence between the pattern bit location and the tester pin number.

The binary pin pattern contains the logic levels to be forced as data or clock on those pins defined as inputs as well as the expected logical response for those pins to be used as outputs.

The voltage levels corresponding to the input logic levels in the binary pin pattern are obtained from the programmable reference supplies E0,E1,EA0,and EA1 (or alternate EB0,EB1,EC0, and EC1). The reference voltage levels corresponding to the expected output logic levels are obtained from the programmable S0 and S1 (or alternate SA0 and SA1) reference supplies.

6.4.5.2 DEFINITION OF LOGIC INVERSION

General Form:

SET [INVERT/I] binary pin pattern;

Description:

This statement provides the capability of inverting the functional data on any pin. A binary 1 in the pin pattern enables that pin's F-data to be inverted (return-to-one condition). A binary 0 in the pin pattern disables the F-data inversion. The pins remain in their programmed state until reprogrammed. The F-data inversion for both RZ and NRZ pins is shown in Figure 6-4.

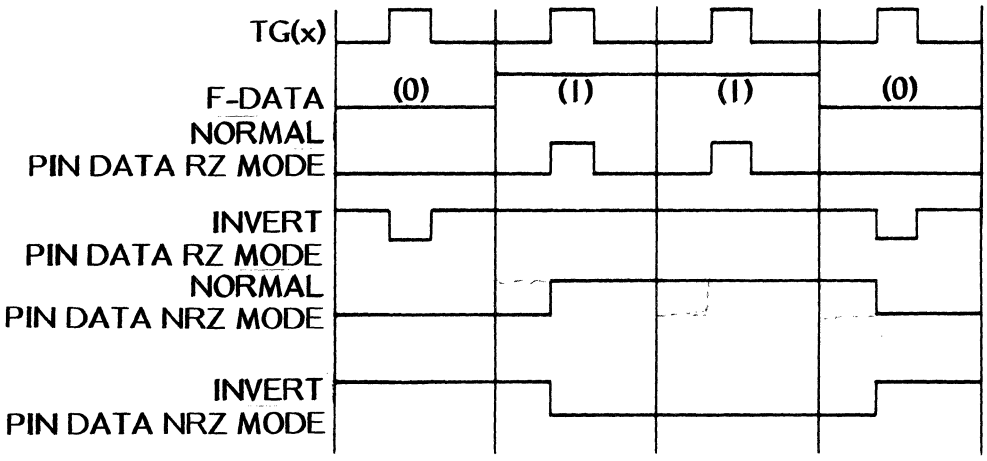


Figure 6-4 F Data Inversion of RZ and NRZ Pins

6.4.5.3 RETURN-TO-ZERO FORMAT

General Form:

SET RZ binary pin pattern;

Description:

This statement allows any tester pin to be programmed to either of two data driver modes. The normal condition is for all data pins to be in a Non-Return-to-Zero (NRZ) mode. By inserting a one at the proper place in the bit pattern, the user may program the pin to a Return-to-Zero (RZ) mode. All pins programmed to a given mode remain in that mode until reprogrammed. Pins defined as clock pins by the CONN CLK statement (refer to Section 6.2.2.1) are set to RZ mode because the compiler automatically generates data for the RZ register. However, if it is desired to have a clock pin be in NRZ mode, the pin should be set to 0 using the SET RZ statement after the CONN CLK statement.

6.4.5.4 RETURN-TO-ONE FORMAT

General Form:

[ENABLE/DISABLE] RTO;

Description:

This statement allows inverting the entire waveform produced when RZ mode and INVERTED data have been programmed for a particular pin. With the RZ and the INVERT bits set to binary 1's, the resultant waveform is essentially a Return-To-One function, hence the name of the mode.

The RTO waveform and its inversion which results from a DISABLE RTO statement is illustrated in Figure 6-5.

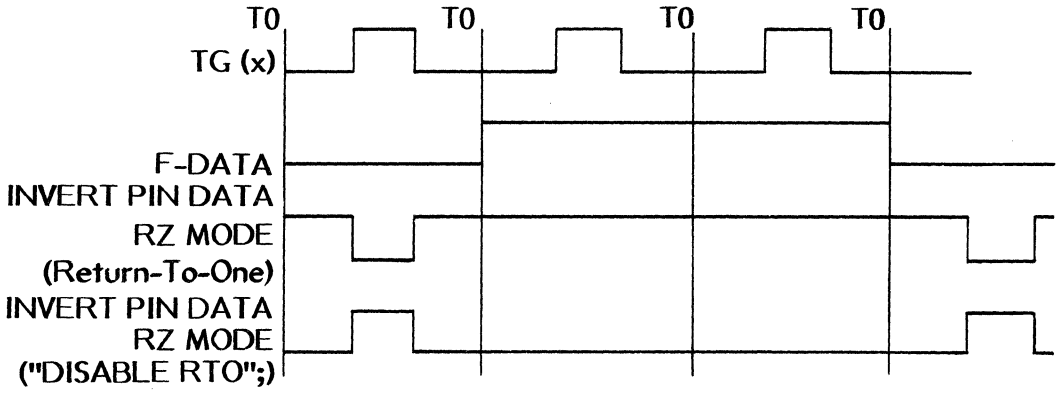


Figure 6-5 RTO Waveform and Inverted RTO Waveform

6.4.5.5 RETURN TO COMPLIMENT FORMAT (EXCLUSIVE OR)

General Form:

SET XOR binary pin pattern;~~(,binary pin pattern,...);~~

Description:

All pins having a binary 1 in this register are in either of two XOR modes depending on the INVERT register. If the INVERT bit for a particular pin is a binary 0, the waveform applied to that pin is equivalent to the logical coincidence (i e., the inversion of an exclusive-Or) of the F-data and the programmed timing generator. In Boolean symbols, the logical coincidence can be expressed as:

$$(A \cdot B) + (\overline{A} \cdot \overline{B})$$

where A•B stands for A "AND" B,

and $\overline{A} \cdot \overline{B}$ stands for inverted A "AND" with inverted B,

and + is the logical "OR" between the two expressions

All possible combinations for this expression are as follows:

A	B	Logical Coincidence
0	0	1
0	1	0
1	0	0
1	1	1

If the INVERT bit for that pin is a binary 1, the waveform applied to the pin is the true exclusive-or of the F-data and the programmed timing generator.

CAUTIONS:

a For any pin, the RZ bit must be a binary 0 whenever the XOR bit is programmed to be a binary 1. Else the waveform applied to the pin is undefined.

b. The timing generator pulse must not exceed the test period when the XOR bit is a binary 1.

Waveforms generated when the XOR bit is a binary 1 are shown in Figure 6-6.

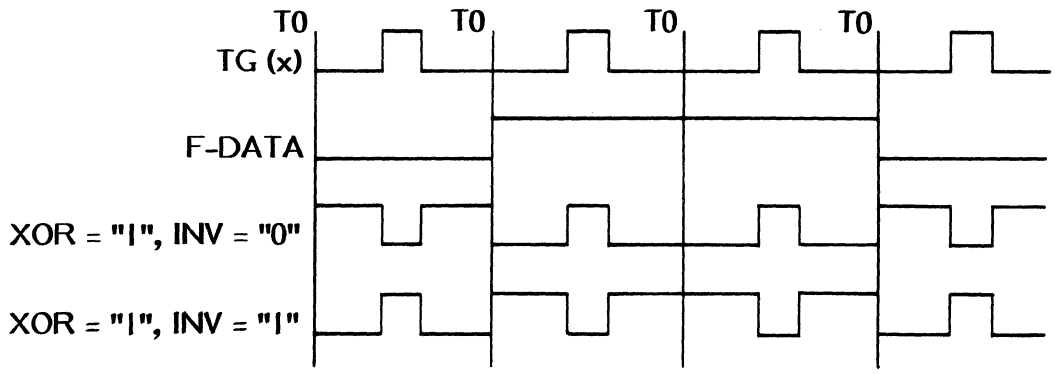


Figure 6-6 XOR Waveforms For a Binary 1

6.4.6 Input/Output Modes

This section describes two methods of controlling I/O pin states other than the previously described ENABLE DA/DB methods. These methods do not change relay states.

6.4.6.1 REGULAR I/O MODE

General Form:

SET IOMODE [OFF/surviving pin list] ;

Description:

This statement provides the capability for selecting certain tester pins to provide I/O definition independent of the definition specified by the SET DA and SET DB statements. The tester pins which the I/O definition specifies are known as the "surviving pins". These are the pins which may be connected to a device while using IOMODE. The tester pins which supply I/O definition are known as "conditioning pins". The pin list consists of the surviving pins separated by spaces or commas. Table 6-7 shows which tester pins may be specified as surviving pins and their corresponding conditioning pins.

TABLE 6-7 TESTER SURVIVING AND CONDITIONING PINS

SURVIVING PINS	1	5	9	13	16	20	24	28	31	35	39	43	46	50	54
CONDITIONING PINS	2	6	10	14	17	21	25	29	32	36	40	44	47	51	55

The state of the F-data of the conditioning pins determines the I/O definition of its surviving pin as shown below:

F=0 Output
F=1 Input

Furthermore, when the conditioning pin specifies the surviving pin to be an input, then the surviving pins comparator response is disabled (i.e., its output is a "don't care"). For each surviving pin specified, the mask registers MA and MB should be set to a zero for the associated conditioning pin and the surviving pin must be specified as an input pin by setting a 1 in either the DA or DB register. IOMODE may be used to control the mask for surviving IOMODE pins. By setting both the DA and DB registers to zero for surviving pins (outputs only), their mask can be made "don't care" by the conditioning pin's F-data set to a 1. Also note that the conditioning pin can have a timing generator assigned to it to delay the time at which the I/O switch changes state within a test cycle period. Furthermore, the controlling pin could be in the RZ mode thus causing on-off-on or off-on-off I/O switch sequences within a period.

IOMODE and CHAIN MODE are mutually exclusive modes. Therefore, only one or the other may be active at any one time.

Example:

```
SET MA   x0xxx0;      REM X = DON'T CARE;
SET MB   x0xxx0;
SET DB   1xxx1x;
SET IOMODE  1, 5;
SET F    x0xxx0;      REM PIN 1 = OUTPUT      PIN 5 = OUTPUT;
SET F    x1xxx0;      REM PIN 1 = INPUT       PIN 5 = OUTPUT;
SET F    x0xxx1;      REM PIN 1 = OUTPUT      PIN 5 = INPUT;
SET F    x1xxx1;      REM PIN 1 = INPUT       PIN 5 = INPUT;
```

6.4.6.2 THREE FOR ONE I/O MODE

General Form:

SET IOM3 [OFF/surviving pin list] ;

Description:

This statement is similar to the SET IOMODE statement. For a given conditioning pin, the list of surviving pins has been extended. Table 6-8 shows the extended 3 for 1 pin list.

TABLE 6-8 IOM3 EXTENDED PIN LIST

Surviving Pins														
1	5	9	13	16	20	24	28	31	35	39	43	46	50	54
3	7	11	15	18	22	26	30	33	37	41	45	48	52	56
4	8	12		19	23	27		34	38	42		49	53	57
Conditioning Pins														
2	6	10	14	17	21	25	29	32	36	30	44	47	51	55

The pin list, however, contains only the first of the three surviving pins for a particular conditioning pin, thereby retaining the same format as the SET IOMODE statement.

Like in the SET IOMODE statement at least one of the I/O mode definition registers DA or DB has to be set to a 1 (in order to close the relay that connects the pin electronics driver to the DUT).

The effect of IOM3 on the "care/don't care" state of the controlled pins is as follows:

(1) If F data on pin 2 is a binary 1, pins 1, 3, and 4 are "don't care" pins. Being the controlling pin in IOM3, pin 2 defines pins 1, 3, and 4 as input pins. Pins defined as input are automatically "don't care" pins in the data comparison.

(2) If F data on pin 2 is a binary 0, "care or don't care" on pins 1, 3, and 4 is determined by the effective "care" mask (i.e., MA or MB whichever is being ENABLED at that time).

The effect of IOM3 on the "I/O definition" of the controlled pins is as follows:

(1) If F data on pin 2 is a binary 1: Pin 1, 3, or 4 is an input pin, if either DA = 1 or DB = 1 for the respective pins, regardless which is in effect (i.e., ENABLED) at the time.

Pin 1, 3, or 4 is an output pin, if both DA = 0 and DB = 0 for the respective pin.

(2) If F data on pin 2 is a binary 0:

Pin 1 is an output pin regardless of DA or DB.

Pin 3 and pin 4 are controlled by the state of the ENABLED D register rather than by the I/O control pin 2. That is a binary 1 in the ENABLED D register defines the pin as an input pin and a 0 as an output pin.

6.4.7 Multiplexing Pin Channels

6.4.7.1 MUXMODE

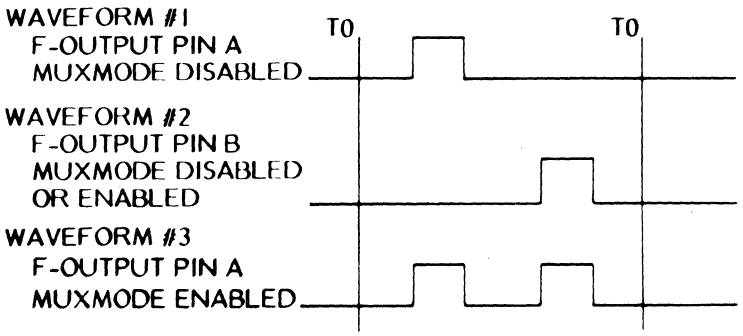
General Form:

[ENABLE/DISABLE] MUXMODE;

Description:

This statement enables the alternate output gates for all pins such that the functional waveform generated for one pin B is hard-wired OR'ed with that of another pin A. This produces an effective multiplexing of the waveforms of the pins involved on pin A.

This is most useful for the case where the pins involved are in the RZ mode and their timing generators do not overlap. An example of this is shown in Figure 6-7.



THE MULTIPLEXING IS DONE ON 16 PAIRS OF PINS:

PIN A	1	2	3	4	5	6	7	8	31	32	33	34	35	36	37	38
PIN B	16	17	18	19	20	21	22	23	46	47	48	49	50	51	52	53

Figure 6-7 Muxmode Example

The third waveform is the F-output waveform going to pin A when the ENABLE MUXMODE statement has been executed.

6.4.8 Chaining Local Memory Channels

6.4.8.1 CHAINING †

General Form:

```
SET CHAIN [TWO/FOUR] surviving pin list;
SET CHAIN OFF;
```

Description:

This statement allows 2 or 4 test patterns to be generated for each following SET F statement, thus utilizing local memory more efficiently. When a sequence of test patterns is executed in the Chain 4 mode, each memory access generates 4 test patterns by using the memory data for the 4 chained pins in sequence and applying them to the surviving pin. Non-chained pins repeat the same test pattern 4 times.

The pin list consists of "surviving pin" members, separated by spaces or commas. Surviving pins are those that may be connected to a device. Table 6-9 shows which pins may be chained together.

TABLE 6-9 SET CHAIN SURVIVING PIN LIST

"Chained Surviving" Pin	1	5	9	13	16	20	24	28	31	35	39	43	46	50	54
Pin chained to it in "Chain 2" mode	2	6	10	14	17	21	25	29	32	36	40	44	47	51	55
Pins chained to it in "Chain 4" mode	2	6	10	-	17	21	25	-	32	36	40	-	47	51	55
	3	7	11	-	18	22	26	-	33	37	41	-	48	52	56
	4	8	12	-	19	23	27	-	34	38	42	-	49	53	57
Rank	1				2				3				4		

SET CHAIN OFF restores the non-chaining mode whereby each memory access generates only 1 test pattern. This form of the statement does not have a pin list.

Only Chain Two mode is available when the local memory size requested by the SET PAGE statement is greater than 1024 but less than or equal to 2048. Chaining is not allowed when the local memory size requested is greater than 2048.

† This statement is not allowed for programs using SPM.

6.5 LOCAL MEMORY TEST SEQUENCE LOGIC

The high speed test station controller contains a local memory which allows the test station to execute high speed functional test. This local memory is available in 30 and 60 pin widths and contains 2048 or 4096 words.

Functional testing patterns (F-data) are loaded into local memory with the SET F and SET FI statements (refer to Sections 6.4.5.1 and 6.5.1.3 respectively). The actual functional testing takes place when the ENABLE TEST statement is executed (refer to Section 6.5.2.4). While the local memory is executing the F-data, the test plan may continue to execute. For example, the device under test can be exercised in a continuous loop while the F-data is being altered outside of the continuous loop.

It is possible to loop within a major and/or a minor loop. Local memory can be loaded with F-data and then have the test start, test end, looping conditions, and the F-data altered by local memory management statements. F-data can also be altered in local memory while in a continuous loop.

6.5.1 Loading Local Memory

Each SET F statement generates code that occupies one word in local memory. Therefore, SET F 1,1,1,1; is actually considered to be four statements even though written as one and occupies four locations (words) in local memory. Any pins not programmed to one (1) in the initial SET F statement of the load are programmed to the zero state. Care should be taken especially when using the AT statement, as all pins in the following SET F statement not specified are set to zero

Any statement in a local memory load may be labeled. The purpose of such labels is to allow symbolic referencing of local memory locations when designating start addresses, loop limits, and locations of modifications. Local memory labels must meet the requirements imposed on regular labels and they must be unique. Local memory labels may only be used in local memory statements and they must be defined before being referenced in statements other than LGOTO. Local memory labels can only be referenced within the block in which they are defined. A local memory label preceeds the statement referenced and must be terminated by an @ symbol.

Example:

```
LABEL@
```

In any statement in which a local memory label is used as an operand, the acceptable forms are:

(1) label
(2) label $\pm$ constant
(3) constant
(4) label $\pm$ expression
(5) expression

The first three forms are evaluated at compile time and produce DMA code (refer to Appendix C). The fourth and fifth forms cause interpretive codes to be produced and are evaluated at execution time.

It is important to consider that because of the domain of the definition of local memory labels and because of the possible variation in sizes of local memory loads, it is possible to have a legal label above the last location in a particular load. For example, in the following example in the second load the end of the major loop is set at location 750 (label AA) although the load ended at location 500.

Example:

```
        SET PAGE 1024;
        SET F .....;            REM FIRST MEMORY LOAD;
        SET F .....;
        .
        .
        .
AA@     ENABLE MB;              REM LOCATION 750
        SET F .....;            LAST SET F OF LOAD;
        ENABLE TEST;
        SET F .....;            REM SUBSEQUENT LOAD;
        .
        .
        .
        SET F .....;            REM LAST SET F LOCATION 500;
        SET MAJOR 25, AA;
```

The high speed test station operates in one of four modes while executing functional tests, they are:

NORMAL
CONTINUOUS
MOMENTARY
MATCH

In the simplest form (NORMAL mode), only the test start address (SET START or location 0) and test end address (SET MAJOR or last location of the load) need be provided in order to bound the execution of a series of functional tests. Testing is initiated by an ENABLE TEST statement and begins at the start address indicated by a SET START statement or at location 0 if no other address is provided. Execution proceeds sequentially until the test end address is reached, wrapping around from the end of physical memory if necessary.

The user may also program one or two loops into the test sequence described above. The minor loop may contain all or part of the test sequence and is programmed with the SET MINOR statement. This loop may be executed from 1 to 4096 times. The major loop may contain all or part of the test sequence but always contains the test end address as the end of the major loop. It is programmed with the SET MAJOR statement and also may be executed up to 4096 times. The major loop branches to location 0 from the test end address. When the major loop execution is terminated (i.e., when the location indicated by the test end address is executed) and the major loop count is zero the functional test sequence is terminated.

After execution of a series of functional tests, the user may wish to modify selected locations in local memory before executing further tests. This modification may take place while the local memory is stopped or in continuous mode.

This modification may be performed either in the DMA (SET F) mode for fast executions or in the interpretive (SET FI) mode which allows setting of each pin independently.

A modification may be initiated by specifying an AT local memory label expression, followed by either of the following statements:

(1) SET F binary pin pattern, . . .;
(2) SET FI binary pin pattern;

The local memory label expression may be any of the five forms specified earlier. The same constraint regarding DMA versus interpretive execution applies. Following the AT statement, it is suggested that an ENABLE([MA/MB])([DA/DB]); statement appear to explicitly select the effective mask and I/O definition registers.

The appearance of the AT statement causes the setting of the local memory load address register (MCS) and software location counter if the address is in DMA format. Also, for the initial SET F statement following, all pins which are not programmed to a "1" state are automatically forced to 0. Subsequent SET F statements may assume that the compiler remembers pin states.

6.5.1.1 POSITIONING START ADDRESS OF LOAD

General Form:

AT [label/label ± constant/label±expression/constant/expression];

Description:

The AT statement may be used to designate a local memory address at which it is desired to make one or more modifications. In other words, the load of the subsequent F-data (until the load is terminated) begins at the location defined by the label.

The only statements which may be used to modify local memory are the SET F and the SET FI statements. Since the SET F pattern memory is reset by the AT statement, the first SET F following the AT must specify all pins which are to be set to the "1" level. Those pins not specified are set to 0.

If no variable is contained in the expression, the code generated is DMA.

Example:

```
      AT  0;
      SET F  1011. . . 1101;
      .
      .
      .
      SET F  1101.  . 1000;
L1@   SET F  1111. . . 1010;
      .
      .
      .
```

If a variable is contained in the expression the resulting code generated is interpretive.

6.5.1.2 LOADING THE FUNCTIONAL TEST PATTERN

There are two statements for loading of functional test patterns, they are:

ENABLE ([DA/DB])([MA/MB])
SET F binary pin pattern (,binary pin pattern, . . .);

The ENABLE DA/DB statement is used to select the input definition register to be used in conjunction with subsequent SET F or SET FI functional patterns loaded into local memory. This statement is discussed in detail in Section 6.4.2.2.

The ENABLE MA/MB statement is used to select the output mask register to be used in conjunction with subsequent SET F or SET FI functional patterns loaded into local memory. This statement is discussed in detail in Section 6.4.3.2.

The SET F statement is used to load a binary functional test pattern into local memory in the high speed test station controller. The SET F statement is discussed in detail in Section 6.4.5.1.

6.5.1.3 INTERPRETIVE LOCAL MEMORY BIT CHANGES

General Form:

SET FI binary pattern;

Description:

This statement is used to modify one or more bits of data at a local memory address location rather than the entire location contents as done by a SET F statement. When this statement is used, all bits which are to be modified must be explicitly specified in the binary pin pattern using the normal SET F coding techniques (refer to Section 6.4.1). Bits not specified in the SET FI statement remain in their current state.

This statement must be preceeded by an AT statement in order to specify the location "at" which the change is to be made. Since the SET FI statement does not increment the local memory address register (MCS register) each succeeding SET FI statement must also be preceeded by an AT statement.

Examples:

(1) Current state of memory location 500:
1010110. . .00101

(2) AT 500;
SET FI [2] 1010;

(3) Final state:
1101010. . .00101

6.5.2 Initiating Local Memory Function Tests

This section describes the FACTOR statements which are used to initiate functional testing from the local memory.

6.5.2.1 TEST START ADDRESS DEFINITION

General Form:

SET START [label/constant/label±constant/expression/label±expression];

Description:

This statement is used to specify the test start address for the next functional test execution. The evaluated start address must be a positive integer within the range of local memory addresses established by the SET PAGE statement.

If no address is specified for a particular functional test execution, the test starts at location zero (0) in local memory.

If the operand does not contain a variable, the instruction produced by this statement executes in DMA mode.

6.5.2.2 TEST STOP ADDRESS DEFINITION AND LOOPING

General Form:

SET MAJOR expression [,label/label±constant/constant/label±expression/expression];
SET MAJOR N,L;

Description:

This statement defines the major loop within local memory (Figure 6-8) by setting:

(1) Major loop count (N) expression. This is the number of times the functional patterns within the major loop is executed ($1 \leq N \leq 4096$).

If N is greater than 1, when the local memory address equals L during pattern execution, a branch to location 0 is made and the major loop count decremented by one. Functional execution continues in the major loop (0 to L). Until the loop count is exhausted. N-1 branches from L to 0 are made.

(2) Major loop end address (L). This is the last memory location to be executed unless a fail occurs first, interrupting testing. This is defined by the last local memory word loaded in the previous load or the one defined by the SET MAJOR statement.

Operation and terminology relating to the local memory is explained in Section 6.5.1.

The evaluated loop count (N) must be in the range $1 \leq N \leq 4096$ and indicates the number of times the code within the major loops is executed. A SET MAJOR statement must occur prior to each functional test during which a major loop is desired. Otherwise, the major loop code is executed only once and testing terminates after execution of the last local memory word loaded.

If no variables are contained in either operand, the generated statements are executed in DMA mode.

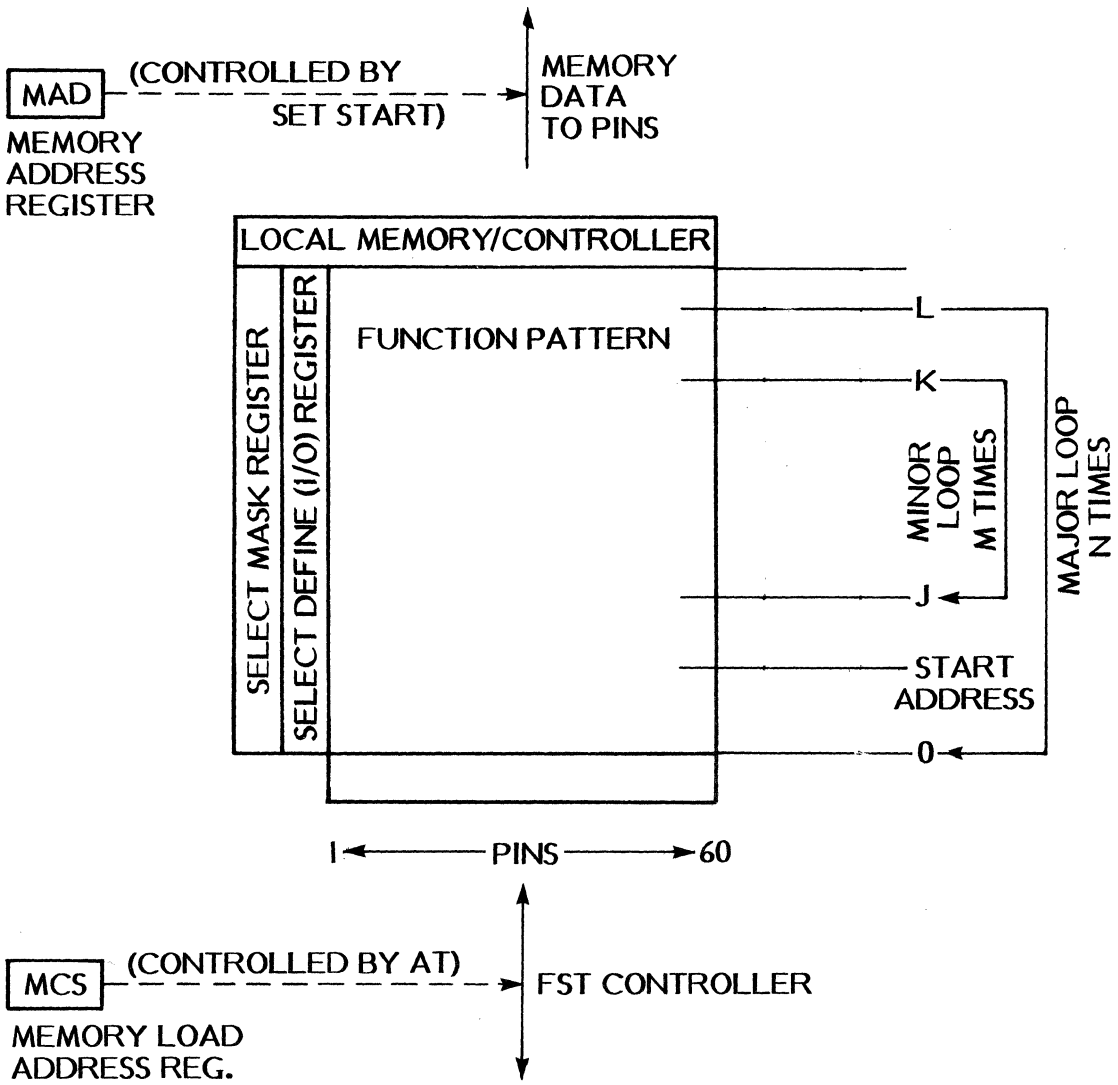


Figure 6-8 Local Memory

6.5.2.3 MINOR LOOP DEFINITION †

General Form:

```
SET MINOR expression([,label/label+constant/constant/label±expression/
                      expression][label/. . .expression] );
SET MINOR M  (,J,K);
```

Description:

This statement defines the minor loop within local memory by setting:

(1) Minor loop count (M) expression. This is the number of times the functional patterns within the minor loop are executed ($1 \leq M \leq 4096$).

(2) Minor loop start address (J). This is the local memory address for the start of the minor loop.

(3) Minor loop end address (K). This is the last address in the loop before execution controls returns to the minor loop start address.

Once programmed, these values remain set until reprogrammed. The minor loop may be removed by programming the minor loop count to 1. If no minor loop is required for a function test the SET MINOR statement need not be included for the default value of M is one.

If the loop boundaries are not altered, the statement may indicate only a new loop count (operand 1). However, if either boundary is to be altered, both must be programmed in the statement.

If no variables are contained in the statement, the code is executed in DMA mode.

6.5.2.4 IGNORING TEST COMPARISONS

General Form:

```
SET IFAIL [label/constant/label±constant/expression/label±expression];
SET IFAIL expression, COUNT;
```

Description:

These statements indicate a local memory address in the first form or the number of test cycles in the second where functional test failures are ignored.

The SET IFAIL [label/constant/. . .] ; statement designates a local memory address for which, after local memory function testing commences (via the ENABLE TEST IFAIL statement) failures are ignored until a test vector is accessed from the IFAIL address. After the test at the IFAIL location is executed, the next test strobes enabled comparators for failures. Once the IFAIL location has been executed, fail strobing is enabled outside continuous loops as long as local memory execution continues.

The SET IFAIL expression COUNT statement indicates that functional failures are to be ignored up to the number of test ~~cycles~~ executions (LTD's) specified for a test sequence initiated by the ENABLE TEST IFAIL statement.

The ignore fail count must be in the range of 0 to 37777777B or 8,388,607 decimal. The count may also be specified as an expression whose value can be program controlled throughout a test plan execution. In this case, the user should be aware that the count value loses its order of significance (i.e., truncation errors occur) once it is greater than 177777B.

†This statement is not allowed for programs using SPM.

The SET IFAIL COUNT and the SET IFAIL statements are mutually exclusive for an ENABLE TEST IFAIL statement. The last one before the ENABLE TEST IFAIL statement prevails.

6.5.2.5 LATCHING FAIL COMPARE RESULTS

General Form:

[ENABLE/DISABLE] LATCHES;

Description:

The ENABLE statement initializes the functional test control so that the C register is not cleared prior to strobing the functional test comparators. In this mode, when testing is enabled, (each ENABLE TEST) the C register is initially cleared and functional testing is not terminated regardless of the number of failures which occur and until local memory reaches the major loop end address and the major loop count is exhausted. The C register accumulates failures on all care pins during functional testing.

The DISABLE statement initializes the functional test control so that the C register is cleared prior to strobing the functional test comparators for each functional test.

If no latch statement is made, the disable latch mode is active.

6.5.3 Function Test Execution

This section describes the statements which initiate functional test execution and control various function test modes.

6.5.3.1 NORMAL MODE TESTING

General Form:

ENABLE TEST (NORMAL);

Description:

This statement is the simplest form of the test execution statements. The purpose of this statement is to initiate functional testing. The NORMAL mode of execution is assumed.

In NORMAL mode testing begins at the test start address and continues to the test end address. The test end address is the one specified in the SET MAJOR statement.

The local memory loop capability may be used in NORMAL mode (refer to Section 6.5.1).

If the wrap around location in local memory (Table 6-10) is executed without reaching the test end address, control is transferred to local memory address 0 and testing continues until the test terminates.

TABLE 6-10 RESULTS OF SET PAGE INTEGER STATEMENT

SET PAGE integer;	Local Memory Wrap Around	CHAIN TWO Allowed	CHAIN FOUR Allowed	Required Local Memory Sizes
1-1024	1023	YES	YES	1k, 2k, or 4k
1025 - 2048	2047	YES	NO	2k , or 4k
2049 - 4096	4095	NO	NO	4k

6.5.3.2 LOCAL MEMORY CONTINUOUS LOOPS

General Form:

ENABLE TEST CONTINUOUS (IFAIL); †

Description:

This statement initiates functional testing in CONTINUOUS mode. This mode is identical to NORMAL test execution except that testing remains in a loop until the tester mode is changed by execution of ENABLE TEST MOMENTARY. In this mode the FACTOR statements following the ENABLE TEST CONTINUOUS are executed while the local memory data from the minor loop is applied to the device under test.

Upon execution of ENABLE TEST CONTINUOUS, the local memory starts sending out data from consecutive local memory locations. If the major loop end address, L, is encountered, a branch to location 0 occurs. This is a continuous major loop. The strobing of enabled output pins occurs in this loop. If the beginning of the minor loop, J, is passed, output comparator strobing is suppressed. If the minor loop end, K, is reached, a branch to J occurs. This is a continuous minor loop.

When the tester is operating in continuous mode in the minor loop, local memory can be altered to contain a new test sequence. Execution of an ENABLE TEST MOMENTARY then causes the tester to fall out of the minor loop and execute the new test sequence.

6.5.3.3 EXIT FROM CONTINUOUS LOOP

General Form:

ENABLE TEST MOMENTARY (IFAIL);

Description:

The purpose of the ENABLE TEST MOMENTARY statement is to allow the user to exit from CONTINUOUS mode. After the ENABLE TEST MOMENTARY statement is executed the first time the minor loop end address (K) is reached, the branch to the minor loop start address (J) is inhibited, and NORMAL testing is resumed as follows:

(1) When the minor loop end address (K) is not equal to the major loop end address (L), the system goes from (K) to (K+1).

(2) When (K) and (L) are equal and the major loop count (N) is zero, the system stops at (K,L).

(3) When (K) and (L) are equal and the major loop count (N) is not zero, the system goes from (K,L) to location 0 and (N) is decremented by 1.

† This statement is not allowed for programs using SPM.

6.5.3.4 CONSECUTIVE CONTINUOUS LOOPS

General Form:

ENABLE TEST MOMENTARY CONTINUOUS (IFAIL):

Description:

The ENABLE TEST MOMENTARY CONTINUOUS statement combines the functions previously described for ENABLE TEST CONTINUOUS and ENABLE TEST MOMENTARY. This statement is used after the local memory is already in the continuous loop mode and allows the normal execution of data outside of the minor loop and reentry into a continuous loop without stopping.

After the ENABLE TEST MOMENTARY CONTINUOUS statement is executed, the first time the minor loop end address (K) is reached, the branch to the minor loop start address (J) is inhibited. Then the following events occur:

(1) If the major loop end address (L) is greater than (K), the address sequence goes from (K) to (K+L). Since (K+L) is outside of the minor loop, the comparator strobing is no longer suppressed and since continuous mode is on, a major loop branch to 0 occurs when L is encountered. The sequence continues and when location J is encountered, a continuous monitor loop has been reentered with looping from K to J and comparator strobing inhibited.

(2) When (K) and (L) are equal, the address sequence goes from (K) to 0 and comparator strobing is no longer suppressed. When (J) is encountered, the continuous minor loop is reentered as described above.

Note that in both cases above, the major loop count (N) has no meaning since testing is always in a continuous major or minor loop.

The ENABLE TEST MOMENTARY CONTINUOUS statement is primarily used for continuous, or non stop execution of several local memory loads while testing dynamic devices. The minor loop functional exercise is normally called a "keep alive loop" and contains the necessary data to "refresh" the device under test.

Final termination of functional testing after a series of ENABLE TEST MOMENTARY CONTINUOUS statements would be executed by ENABLE TEST MOMENTARY.

Example:

```
Page = 1;
CALL LOAD (PAGE); REM LOAD FIRST PAGE OF LOCAL MEMORY DATA;
SET START 5;
SET MINOR 1, J, K;
SET MAJOR 1, L;
ENABLE TEST CONTINUOUS: REM TEST FIRST PAGE AND FALL INTO
                                MINOR LOOP;
FOR PAGE = 2 THRU 10 DO BEGIN
CALL LOAD (PAGE); REM LOAD NEW PAGE;
ENABLE TEST MOMENTARY CONTINUOUS; REM TEST NEW PAGE;
END;
PAGE = 1;
CALL LOAD (PAGE); REM LOAD LAST PAGE;
ENABLE TEST MOMENTARY; REM TEST LAST PAGE AND STOP;
```

6.5.3.5 MATCH MODE

General Form:

ENABLE TEST MATCH (IMMEDIATE);

Description:

The purpose of MATCH mode is to functionally exercise a device under test whose outputs are initially in an indeterminent state, and when a certain output state is sensed, branch to a normal functional exercise pattern.

When ENABLE TEST MATCH is executed, functional testing starts at location (S) in the MATCH mode. The match search continues until a match is found. This means that the pass/fail logic from the C register is reversed, and a "functional fail" on any care pins is a "non-match" condition. As long as a "non-match" condition is found, the functional exercise keeps executing with the local memory address incrementing each cycle. Normally a minor loop is used for a match search to confine the local memory address between J and K inclusive. Whenever all care pins pass, a match has been found and the local memory address is set to zero on the next cycle. Normal functional testing then proceeds from location 0 until the test terminates.

A bad device under test could possibly never generate a functional state which matches a test pattern in the match loop. This would mean that match mode would never terminate. The match search can be set up to terminate after a defined length of time.

The timer value used with the ENABLE TEST MATCH statement is set by the statement SET DELAY expression, DC. The timer value should be at least as long as the match mode test time + normal functional test time. If it is known that a device always finds a match, the timer delay can be eliminated by setting a delay of zero or by using the statement ENABLE TEST MATCH IMMEDIATE. ENABLE TEST MATCH IMMEDIATE executes in DMA mode and therefore has a short ~~in~~ execution time.

While in the minor loop under MATCH mode, testing proceeds at the programmed rate. Timing in internal sync MATCH mode is shown in Figure 6-9.

Use of match mode:

(1) Minimum test period = 800 ns

(2) Allowable Strobe region: TG7 DELAY + WIDTH = PERIOD MINUS 600 ns. Only TG7 is used during search for match.

TG8: Disabled during search for match. After match is found, TG8 is enabled if selected by the SET STROBE and ENABLE DOUBLE STROBE statements.

(3) For the first two test periods, finding a match is inhibited. (This is required for external match).

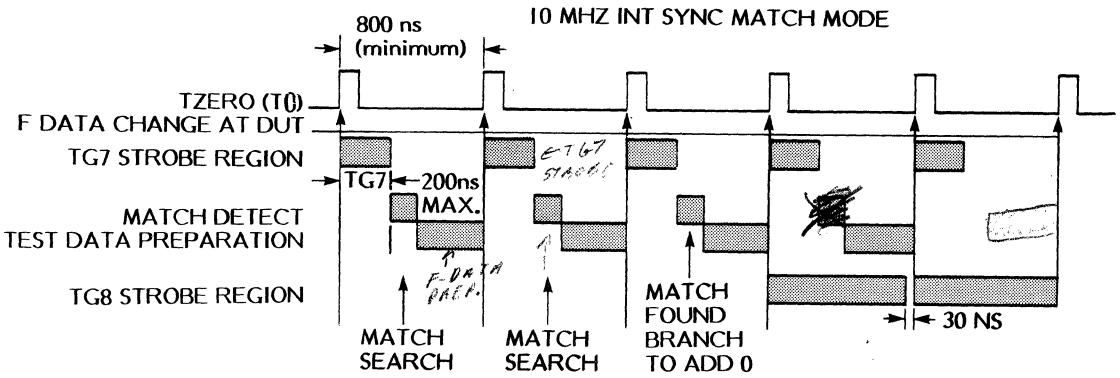


Figure 6-9 Timing in Internal Sync MATCH mode

6.5.4 Function Test Execution Options

6.5.4.1 INITIATING TESTS IN THE IGNORE FAIL MODE

General Form:

ENABLE TEST IFAIL;

Description:

This statement initiates testing in the Ignore Fail mode. This mode is identical to ENABLE TEST NORMAL except that it allows functional failures to be ignored until the IFAIL address or IFAIL count is executed (refer to section 6.5.2.4). The Ignore Fail mode may be activated by appending IFAIL to any of the ENABLE TEST modes except MATCH mode.

EXAMPLE:

ENABLE TEST IFAIL;
ENABLE TEST CONTINUOUS IFAIL;
ENABLE TEST MOMENTARY CONTINUOUS IFAIL;
ENABLE TEST MOMENTARY IFAIL.

6.5.4.2 DEFINING EXTERNAL SYNC OPERATION

General Form:

ENABLE TEST(NORMAL/MOMENTARY/CONTINUOUS) (EXT/EXTA) (IFAIL);
ENABLE TEST MATCH (EXT/EXTA);

Description:

For those devices which have their own internal clock, ENABLE TEST, EXT provides the capability of replacing the system test rate generator with the device internal clock to act as a sync for starting functional test and timing generators. Tester pin 1 is used to provide the sync signal to the system. Pin 1 is programmed as an output pin and is connected to the clock pin of the device under test (DUT), or to an external generator if required.

External Sync Pulse Characteristics:

Logic levels of the External Sync signal must be compatible with other outputs of the DUT. Level shifters are otherwise required at the Performance Board to provide appropriate levels. Figure 6-10 shows the external pulse characteristics for the following conditions:

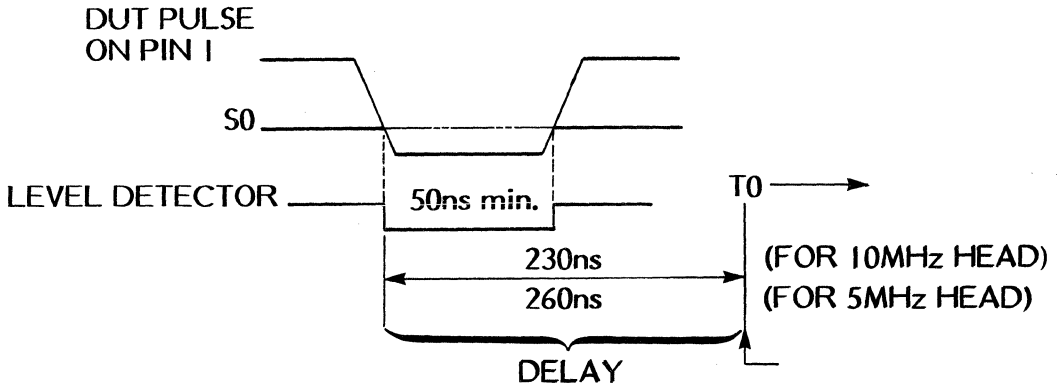
Trigger is on the falling edge of the level detector output for pin 1.

Pulse width 50 nsec. min.

Pulse amplitude - higher than S0/S1 at high state and lower than S0/S1 at low state.

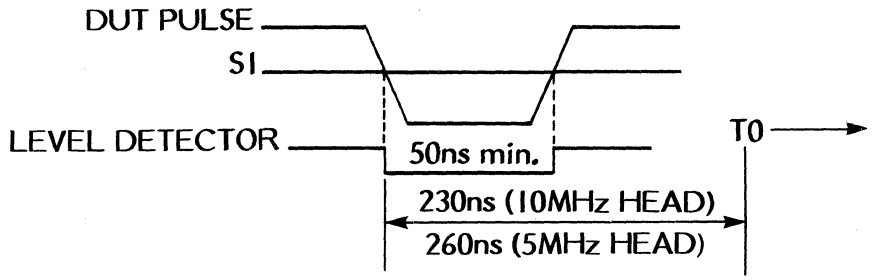
Use of pin 1 as trigger pin:

(a) Set DA = 0, DB = 0 for Pin 1 (always output)
(b) Set MA = 0, MB = 0 for Pin 1 (always don't care)
(c) If F = 0 on Pin 1:



When the DUT pulse on Pin 1 crosses the S0 level, it triggers the level detector pulse which in turn starts the test cycle, T0 after the respective delay.

(d) If F = 1 on Pin 1:



Here F-data is programmed to 1 on Pin 1. The determining factor, then, is when the DUT pulse crosses the S1 level.

(e) Negative logic does not make a difference.

Figure 6-10 External Sync Pulse Characteristics

External Sync Mode:

Since the device in the DUT socket may fail to give external sync pulses, the internal test rate generator is used to prepare the tester and to send out the F-data for the first test. The tester then switches over to external sync mode. At the end of a test or in the case of a fail, the internal test rate generator is used again to shut down the tester. The DUT however, does not see the internal rate.

External Sync Error Conditions:

If the external sync fails to occur or fails to continue after its first occurrence and causes the functional test to halt, the failure is detected after a programmed delay has expired and a functional test failure is indicated. The value of the delay is obtained from the FACTOR statement:

SET DELAY expression, DC;

and must be programmed longer than the time expected for the entire functional test sequence. A DC delay value of zero disables the external sync failure check.

Internal and External sync cannot be mixed in the ENABLE TEST CONTINUOUS (MOMENTARY/IFAIL/NORMAL) modes.

Use of External Sync Mode:

(1) Minimum test period of external sync pulses is 200 ns. (Equal to a 5 MHz maximum test rate.)

(2) The internal test rate generator has to be programmed in range 0 to 300 ns or greater.

(3) Test data appears at the DUT 230 ns on a 10MHz test head and 260 ns on a 5MHz test head after the external sync pulse is applied to Pin 1.

(4) For the first F-data sent out, the timing generators are disabled. The pins used for the first F-data must be in NRZ mode, and the first test has to be programmed as a dummy test.

This avoids the problem of an unpredictable period occurring between the last internal sync and the first external sync.

(5) The minimum period during switch-over from internal to external sync can be $\approx$ 170 ns.

When External Sync mode is used with CONTINUOUS mode the maximum test time cannot be programmed to detect failure of the external sync signal. The proper funcioning of the external sync should be verified by the FACTOR program prior to entering the CONTINUOUS mode.

When External Sync is used with MATCH mode, MATCH mode timing is changed. The test data appears at the DUT $\approx$ ~~580~~ 330 nsec (10 MHz head) after the external sync. The minimum test period remains at 800 nsec. Only TG7 can be used as a strobe during search for match. TG8 is enabled after a match has been found. Matching in the first two tests is inhibited. Timing in external sync MATCH is shown in Figure 6-11.

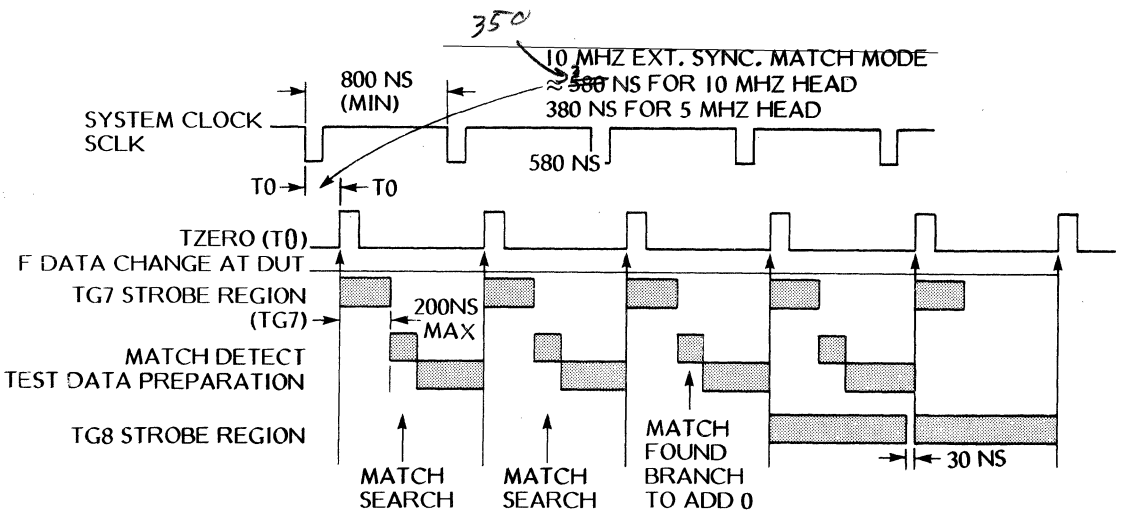


Figure 6-11 Timing in External Sync Match Mode

External Sync Alternate

The ENABLE TEST EXTA statement does not disable the timing generators during the first F-data sent out. Also no dummy test is required for the first test. The user must be responsible for using this mode only with predictable devices which give an external sync pulse at a known time, so that the period is not shorter than the delay or width of any timing generators which are programmed. For a free-running external clock the minimum period between the first and second tests still could be ≈170ns.

6.5.5 Function Test Termination

The tester terminates testing in NORMAL mode under three conditions:

(1) The test end address is reached and the major loop counter is zero.

(2) A functional fail has occurred.

(3) In external sync or MATCH mode, execution of the functional test sequence exceeded the programmed value of maximum test time (via the SET DELAY expression, DC; statement). A functional fail condition occurs to terminate test execution.

6.6 MISCELLANEOUS TEST STATEMENTS

6.6.1 Branch on Fail

General Form:

ON [DCT/FCT/TRIP] ,label;

Description:

These statements establish program branch control on DC test failures (DCT), functional test failures (FCT) and DPS trip failures (TRIP). The label specifies the branch location to which program control is transferred when the event occurs. The label must be in the outermost block of the test program in all cases, (i.e., the label must be in the main program, not in a subroutine).

The three branch conditions are programmed independently. Once a branch is taken, the ON statement is cancelled; a second failure of the same type does not alter program control unless the branch is reprogrammed. In the event of multiple fails on the same test, a DPS trip branch has priority over a DC or functional fail branch.

When a trip, functional or DC failure occurs at statement n and, if the corresponding ON statement has not been processed or if the branch has been used and not reprogrammed, program control resumes at statement n+1. The ON statement is cleared by the CLEAR [FCT/DCT/TRIP] ; statement and is overridden by a subsequent ON statement of the same type.

Example:

```
          ON DCT,DFAIL1;
          ON FCT,FFAIL;
          ON TRIP,CFAIL;
DFAIL1:   ON DCT,DFAIL2;
```

6.6.2 Clearing Branch on Fail Flags

General Form:

CLEAR [FCT/DCT/TRIP] ;

Description:

This statement clears a corresponding previously programmed ON [FCT/DCT/TRIP] statement if the program branch has not yet been taken.

Example:

```
SET F 1;
SET F 11;
SET F 11;
SET F 111;
SET F 1111;
ON FCT,FAIL1;
ENABLE TEST;
CLEAR FCT;          REM NO FAILURE, PREVENT GOING TO FAIL1
                    IF AN FCT FAILURE OCCURS LATER;
```

```
LOOP:
        .
        .
        .
        .
FAIL1:      FFLAG=1;     REM REMEMBER FAIL OCCURRED;
            GOTO LOOP;
```

In the example, a functional failure occurring on the ENABLE TEST statement causes a branch to the label FAIL1. The test program sets a flag to indicate that the failure occurred, and then returns to the test sequence at the label LOOP.

If no functional failure occurs the CLEAR FCT statement rests the ON FCT, FAIL1 request because it is not wished for it to go to FAIL1 if a functional failure occurs later in the program. Another ON FCT statement would also clear the previous ON FCT but may not be desired.

6.6.3 Clearing Fail Indicators

General Form:

CLEAR FAIL [FCT/DCT/TRIP] (,FCT/DCT/TRIP);

Description:

This statement clears the system software fail flags. When a fail is normal (e.g., for device pre-conditioning, this statement may be used to inhibit this fail from being displayed at end of test. More than one option may be specified, separated by commas.

6.6.4 Controlling Load Board Utility Relays

General Form:

SET R binary pin pattern;

Description:

This statement provides the control for opening or closing the utility relays. There is one relay associated with each tester pin. A binary 1 in the binary pin pattern closes the utility relay and a binary 0 opens the relay.

6.6.5 System Reset During Program Execution

General Form:

FORCE RESET;

Description:

This statement forces the test system into the reset state, thus clearing all programmable test conditions. The programmable supplies are forced to a reset condition in the following sequence prior to being disconnected from the device under test:

Clear DPS 1, 2, and 3 in that order without changing the range.

Clear DPS trips (DPT register) 1, 2, and 3 in the order without changing the mode and range.

Forces PMU to 0 voltage.

Clear PPS register without changing the mode and range.

Clear E1, S1, EA1, EB1, EC1, SA1, E0, S0, EA0, EB0, EC0, SA0 in that order without changing the range.

Wait for 3.36MS

Clear all short and long registers
(all relays are disconnected)

6.6.6 Enable Access

General Form:

ENABLE ACCESS;

Description:

This statement forces a disc access at run time to reload the memory buffer with the test program. Normally TOPSY dynamically allocates the amount of data in the memory buffer. With a 16K memory, the maximum buffer size is about 4000 words. A disc access and reload of this buffer takes place at the end of the buffer and takes about 70 milliseconds. When testing dynamic MOS devices with large programs, it may be desirable to control when the buffer is refreshed so some initializing data for the device under test can follow the disc access period. This may be done with the ENABLE ACCESS instruction.

SECTION 7.0

VARIABLE DECLARATION AND VALUE ASSIGNMENT

As described in Section 2.2, variables may be used in expressions without giving them initial values or by declaring them. If they are not declared, they are assumed to be a single variable. If they are not given an initial value, they are automatically given an initial value of zero. Variables may be declared and assigned values at any point in a program.

A variable may also be used as an array reference, but then it must be declared. Thus, a declaration (DCL) statement must always be executed before any references are made to the declared arrays. If this rule is violated, TOPSY indicates the programming error with a terminal error at run time (see Appendix E).

7.1 DCL

The DCL command is used to reserve storage for variables, assign initial values, and to make a variable local to the block in which it is declared.

If the DCL statement is executed more than once in a currently open block, all but the first execution is ignored; however, a value assignment always occurs at every execution. It means that the evaluation of an array size occurs only once at the first DCL for that array. However, values are assigned for every DCL specified.

Two types of variables may be declared: single variable and arrayed (one dimensional variables).

7.1.1 Single Variable Declaration

The general formats on the scalar declaration are as follows:

```
DCL  V1;
DCL  V1 ,V2,. . . . .Vn;
DCL  V1/value 1/,V2/value 2/, . . .Vn/value n/;
DCL  V1,V2/value 2/,V3 . . .,Vn;
```

V1. . .Vn stand for variable numbers 1 through n. value 1. . . value n stand for single signed or unsigned numbers which declare the value of a variable. When declared without a value, the variable is set equal to 0. Multiple declaration and assignment can be made with one statement. As shown in the last two examples, each variable of a multiple declaration can be optionally assigned an initial value.

7.1.2 Array Declaration

The general formats of the array declaration are as follows:

```
DCL  V1 [asize 1] ;
DCL  V1 [asize 1] , . . .Vn [asize n] ;
DCL  V1 [asize 1] /AE1, AEm/, . . .,Vn [asize n] /AE1, . .AE m/;
DCL  V1 [asize 1] ,V2 [asize 2] /AE1. . .AE m/,. . .,Vn [asize n] ;
```

The formats are similar to those for a single variable declaration; however, the array identifier, V1, requires an argument to specify the number of elements, i.e., the size of the array. This quantity (asize) is enclosed in square brackets. The size is specified by an expression which allows it to be variable or fixed. The evaluation of array size and allocation of storage is performed by TOPSY at run time. The array size, of necessity, is automatically truncated to the nearest integer if it should be expressed as a non-integer value.

The elements of an array may be optionally assigned initial values when declared. The assignment is specified by the terms AE1 through AEm as shown above; m is the size of the array. If the size and number of initial value assignments do not agree, the missing (trailing) elements are set to zero (or blanks in the case of literals). If too many elements are specified, a compiler error (if the size is a constant) or, a run time error (if the size is an expression) occurs.

NOTE

> The distinction must be drawn between the value in square brackets used in a DCL statement, where it represents the array size, and the value in square brackets used in a non-DCL statement, where it specifies the array element desired.

Examples:

```
DCL  ARR [10] ; REM ARRAY SIZE = 10 ELEMENTS;
FOR J = 1 THRU 10 DO
ARR [J] = 2*J; REM COMPUTE ARRAY ELEMENT VALUES;
FIFTH = ARR [5] ; REM ASSIGN VARIABLE FIFTH
                    THE VALUE OF THE FIFTH ARRAY ELEMENT;
```

7.1.3 Literal Variable Capability

A variable or an array may be used as a literal (i.e., as a string variable or a string array response) in FACTOR READ, WRITE, and assignment statements.

The "&" (ampersand) sign indicates a single variable or an arrayed variable as being literal. Without the "&", a single variable or an arrayed variable is treated as being numeric. All displayable characters (except a semicolon(;) and another single-quote(') are allowed within a literal variable declaration. Mixed mode usage causes indeterminate results. For a literal array declaration, the size of the array (in terms of word count) must be large enough to include the whole string. Four characters fit into 1 word.

The text within the literal variable declaration may be grouped into 4-character groups by single quotes and commas.

Example:

```
DCL MSG [6] / 'NAME', 'OF P', 'ROGR', 'AM I', 'S'/;
is the same as
DCL MSG [6] /  'NAME OF PROGRAM IS'/;
```

Grouping a message text this way may help the user in checking for the proper size of the array being declared.

Example #1:

```
000001          SET PAGE 1;
000002          DCL MSG [6] /'THE DEVICE HAS'/;
000003          DCL PASS [2] /'PASSED'/, FAIL [2] /'FAILED'/;
000004          FORCE RESET;
000005          IF SWITCH EQ 1 THEN BEGIN
000006          SETMODA 1;
000007          END;
000007          ON FCT, DFAIL;
000010          SET F 1;
000011          SET PERIOD 100E-6;
000012          ENABLE TEST;
000013          MSG [5] = PASS [1] ;
000014          MSG [6] = PASS [2] ;
000015          WRITE &MSG;
000016                          REM 'THE DEVICE HAS PASSED';
000016          GOTO PEND;
000017        DFAIL:
000017          MSG [5] = FAIL [1] ;
000020          MSG [6] = FAIL [2] ;
000021          WRITE &MSG;
000022                          REM 'THE DEVICE HAS FAILED';
000022        PEND:
000022          END;
0000B           COMPILATION ERRS
```

The following are the results from executing the test program. At first, SWITCH was set to 0

```
STAT3D      TEST PLAN      C      SN          16
THE DEVICE HAS PASSED
```

Then /. SWITCH 1 STAT3D was entered to set SWITCH to 1

```
STAT3D      TEST PLAN      C      SN          17
THE DEVICE HAS FAILED
```

Example #2:

```
00001          SET PAGE 1;
00002          DCL MSG [6] /'NAME OF PROGRAM IS'/;
00003          WRITE 'TYPE IN NAME OF PROGRAM';
00004          READ (TTK) &NAME;
00005                        REM NAME IS LIMITED TO 4 CHARACTERS;
00005          MSG [6] = NAME;
00006          WRITE &MSG;
00007          END;
0000B          COMPILATION ERRS
```

The following are results from executing the test program. ABCD, 1234 and D808 were the inputs respectively for the three times.

```
STAT3D        TEST PLAN        D        SN          2
TYPE IN NAME OF PROGRAM
NAME OF PROGRAM IS ABCD

STAT3D        TEST PLAN        D        SN          3
TYPE IN NAME OF PROGRAM
NAME OF PROGRAM IS 1234

STAT3D        TEST PLAN        D        SN          4
TYPE IN NAME OF PROGRAM
NAME OF PROGRAM IS D808
```

7.2 VARIABLE ASSIGNMENT STATEMENT

The variable assignment (or replacement) statement is the most fundamental of all FACTOR statements. It takes the general form:

```
variable = expression;
```

This statement results in the replacement of the value of the variable on the left by the value of the expression on the right. (In general, it is not an equation, since the variable on the left may form part of the expression on the right).

Thus the statement:

```
A = A+B;
```

means take the value of the variable A, add the value of the variable B and replace the value of the variable A with the result.

SECTION 8.0

READ/WRITE STATEMENTS

The READ and WRITE statements that control data-flow in and out of the computer during execution are described in this section. They use the following syntax notation.

(1) A 0 indicates that none of the elements of the set need be chosen. When none are selected, the system assigns the current DOPSY primary input device or the primary output device unless a current DATALOG statement requests another output device.

(2) A file name identifier is shown in lower case letters and is enclosed by double quotation marks.

8.1 READ

The general formats of the input statement are:

READ((CR) / (FDIF) / (EIR) / (TTK) / (TTR) / (MTR) "name") V1,V2,. . . Vn;

READ((CR) / (TTK) / (TTR) / (FDIF) / 0)&Vi, &Vj . .&Vn;

The items enclosed by parentheses are peripheral devices defined as follows:

Card Reader	(CR)
Video Terminal Keyboard	(TTK)
Magnetic Tape	(MTR)
External Interface Registers	(EIR)
Disc Input File	(FDIF)

When magnetic tape, MTR, is specified the statement must include a file name which is enclosed by double quotation marks. The file name syntax is defined in the same manner as identifiers (see paragraph 2.2.2).

Magnetic tape file "names" are used to uniquely identify data segments on the tape. These names are assigned with the WRITE statement (Appendix I).

The terms V1 through Vn may be any legal variable identifier, including arrays. As the input numerical data is read from the peripheral, it is assigned to the specified variable(s).

The input data for literal variables (variables preceded by '&') must be of the form:

$C_1C_2C_3C_4$	for a ~~simple~~ single variable
$C_1C_2C_3C_4 \ldots C_{4n}$	for an array of size n

where:

C belongs to the FACTOR character set.

All of the characters must fit on one card (or one record of a different media). The first or single value for each new variable identifier must start on a new card. The characters are converted to TRASCII and stored into the variable without further conversion. C_1 must appear in the first column of the card. This capability has not been implemented for magnetic tape.

When ARITHMATIC values of an array are to be read, they must be separated by A COMMA at least one space (for TTK, CR, FDIF). ~~More than one card may be used to enter these values.~~ All numbers following the last array element number on a card are ignored.

When the input peripheral is the magnetic tape unit, the tape is searched forward until the file "name" is located. The numerical data from this file is read and assigned to the variables V1, . . .etc. as specified by the READ statement. For magnetic tape, the variables must be arrays which have no less than 7 elements. The maximum array size is limited by the amount of core memory available when the array is declared. It is recommended that arrays be no larger than 512 elements. Appendix I gives a detailed description of the magnetic tape operation and responses to the READ (MTR) and WRITE (MTW) statements.

When the input is a disc input file, the information is sequentially read from the disc and stored in variables V1, through Vn. (No formatting occurs.) The Disc Input file (FDIF) must have been opened and each READ continues processing the file where the previous READ left off. The assumption is made that the file is composed of records written by a FACTOR WRITE (FDOF) statement, and therefore consists of floating point numbers and alphanumeric text.

If an attempt is made to read beyond the information written in the file, a flag is transferred to the statement label, which appears in the ON DIFEOF statement. If no ON DIFEOF statement has been encountered, a terminal error message #68 is issued. If the DIF is not open, terminal error message #67 is issued.

8.2 WRITE

The formats for output statements are:

```
WRITE ((FDOF)/(TTP)/(LP)/CLO)) expression, Vi, 'Si',&Vj,'Sj',
      /col/V1,/col/'S1', . . .Vn;
```

WRITE expression; → TO POP BY default

WRITE (E14) expression;

WRITE (****D) expression;
Fmt. #

Where: Vi. . .Vj. . .Vn. . .V1. . .V4 are legal variable identifiers including arrays which may occur in any sequence. Si, Sj. . .are strings of alphanumeric characters, col is a numeric column number between 1 and 80, enclosed by slashes

The items enclosed by parentheses are peripheral devices defined as follows:

VKT	(TTP)
Line Printer	(LP)
External Interface Register	(EIR)
Disc Output File	(FDOF)
Magnetic Tape	(MTW)
Communications Link	(CLO)

```
WRITE (MTW) "name" V1, V2, V3, V4;
```

When magnetic tape (MTW) is specified the statement must include a file segment name which is enclosed by double quotation marks. When writing to magnetic tape, the variables Vi must be arrays which have no less than seven (7) elements and are recommended to be no larger than five hundred and twelve (512) elements as described in paragraph 9.1. Appendix I gives a detailed description of the magnetic tape operation and responses to the WRITE (MTW) statement.

When the line printer is specified as the output device there may be one or more strings of alphanumeric characters and one or more variables in a single WRITE statement. All strings must be enclosed by single quotes and must not contain semicolons (;). Multiple variables are separated by commas as are intermixed combinations of strings and variables.

When the output is to a disc output file, the information stored in the variables and the string is output to the disc. (No formatting occurs.) As each word is output, it is added to that file which has been previously specified to be the Disc Output File (DOF). If an attempt is made to write beyond the end of the file, a terminal error message #69 is issued and the test program is aborted. If the DOF is not open, terminal error message #67 is issued.

8.2.1 Numeric Variables

Numeric variables are output in one of three forms. If the numeric value of the variable is a positive integer whose magnitude is less than ten thousand (10000), it is printed in the form 9999.

If the value is negative and of magnitude less than one thousand (1000), it is printed in the form S999.

S is the minus sign (-) and the '9's' are decimal digits. Leading zeroes in a positive number print as spaces.

Integers and non-integers whose magnitudes exceed 999, or 9999, print in the following format:

```
S9.999EP99
```

where S is the sign of the value and the '9.999' represents the decimal digits of the mantissa, the '99' represents the decimal digits of the exponent of the value and P is the sign (+ or -) of the exponent. The character 'E' prints as shown. For example, 8.979 x 10-6 prints as '8.979E-06'.

Numeric values as described above occupy a field of twelve (12) characters and are left justified within this field.

8.2.2 Literal Variables

Literal variables (variables preceded by '&') are output as a string of characters. Four charactes are output for a simple variable, 4n characters for an array of size n. The string of characters is followed by four blanks.

Strings of characters are printed as they appear in the enclosed quotes. The characters may be any of those in the character set excluding single quotation marks and semicolons(;). Leading spaces are printed according to the number of spaces following the single quote of a string. The total number of printed characters is an integral multiple of four (4). (The restriction is automatically imposed at run-time with the addition of no more than three (3) spaces following the character preceding the trailing quote of a string.)

NOTE

> Where a 'data string' or a literal variable array is written under column format control, the entire string or array must fit on one line.

The maximum number of variables printed per line is five (5) with the first character field left justified, unless column formatting is specified.

When a variable is an array, its current values are printed five (5) per line beginning with array element one (1) left justified on the line.

More than five (5) variables can be specified per WRITE statement with the result that five values per line is printed on all lines including the last, unless there are fewer than five values to fill the last line.

A single string of seventy-two (72) characters may be printed on a single line when the VKT is the output device. When the line printer is the output device, a string of eighty (80) characters† may be printed on one line. Single strings which extend beyond column seventy-two (72) of a punched card can be continued beginning with column one (1) of the next card, etc. When the single string exceeds the character counts described above, the excess characters are printed on the following line. The VKT ignores characters between column seventy-three (73) and eighty (80).

† Standard line printer - other line printers are available.

When variables and strings are intermixed in a single statement, without column formatting specified, the following output rule holds:

> If the count of characters printed on the current line exceeds fifty-six (56), then the first character of the next entity (either a variable or string) is printed left justified beginning on the next line. Otherwise, it is printed beginning on the current line and character position. Overflow to the next line occurs whenever the character count of a string exceeds the number of available characters on the line.

Example:

(FACTOR Code):

```
WRITE 'DATALOG';
WRITE '  ';
WRITE 'TEST#=',N,'                VALUE=',VALUE;
WRITE 'NODE=',PINN,               EXPECTED VALUE=',EV;
```

(Output Data):

DATALOG

```
TEST#= + 6                VALUE=',VALUE;
NODE= -0                  EXPECTED VALUE= + 2
```

In this example, the variables are N, VALUE, PINN, and EV. At the time the WRITE statements are executed, these variables had the following numeric values; 6, 1.2 x 10-6, 0, and 2, respectively.

Whenever column formatting is specified, the fifth character of the output is right justified in the specified column (except column 0). This capability is primarily oriented toward outputting integer values, which always are right justified in the specified column. By specifying columns, the programmer can concatenate values.

8.3 FACTOR DISC I/O

8.3.1 ON DIFEOF, Label ;

When a READ (FDIF) statement encounters the end of the written file, control is transferred to the statement Label.

8.3.2 Reset FDIF RESET FDIF;

Reset FDIF initializes the DIF pointer to the beginning of the DIF file.

8.3.3 Programming Conventions for use with FACTOR Disc I/O

The following conventions are suggested to simplify use of disc I/O. It should be kept in mind that several different programs on different stations can be writing to the disc file and that the records from each station are intermixed.

1. All WRITE FDOF statements in all programs should write the same number of words to the disc (i.e., the record size should be constant).

2. Each WRITE FDOF statement should output at least 3 words of identifying information at the beginning of the record:

   Words 1 and 2 - Device name
   Word 3 - Station number

   Other identifying information could be included, such as the current date.

3. The READ/WRITE and variable capability can be used to read in identifying alpha information such as the device name, the station number, or the date. This information may then be output to the disc in a WRITE statement, or may be used in a data reduction program to compare the desired device name, station number, date, etc., against the corresponding characteristic read from the disc file.

8.4 EXAMPLES OF PROGRAMS THAT READ AND WRITE TO DISC

```
REM PROGRAM THAT WRITES TO DISC;

FACT1:          DCL DEVNAM [2], ARRAY [10];

        READ (CR) &DEVNAM, &STAT;
        .
        .
        .
        MEASURE VALUE;

        X1 = VALUE;
        .
        .
        .
        MEASURE VALUE;

        X2 = VALUE;
        .
        .
        .
        WRITE (FDOF) &DEVNAM, &STAT, X1, X2, ARRAY;
        .
        .
        .
        END;
```

```
REM DATA REDUCTION PROGRAM;
FACT2:          DCL DEV [2] DEVNAM [2], ARRAY [10], SUM;
        RESET FDIF;
        ON DIFEOF, AVER;
        I = 0;
        READ(CR) &DEV;
        WRITE (LP) 'DEVICE IS' &DEV;
LOOP:   READ(FDIF) &DEVNAM, &STAT, X1, X2, ARRAY;

REM ONLY PROCESS DATA FOR CURRENT DEVICE;

        IF DEV [1] NEQ DEVNAM[1] THEN GOTO LOOP;

        IF DEV [2] NEQ DEVNAM [2] THEN GOTO LOOP;

        WRITE (LP) '          ',X1, X2;
        SUM = SUM + ARRAY  1
        .
        .
        .
        GOTO LOOP;
AVER:   SUM = SUM/I;
        WRITE (LP) 'AVERAGE IS,' SUM;
        .
        .
        .
        END;
```

SECTION 9.0

FACTOR OPERATING PROCEDURES AND ERROR MESSAGES

A program written in FACTOR must be compiled before it can be executed on the Sentry Test System. The DOPSY compiler converts the FACTOR English-like statements into a program file of object codes. The object code is then used by the TOPSY system, which interprets and executes the program.

9.1 PROGRAM INITIATION

The syntax of the DOPSY command to request the FACTOR compiler is:

// COMPILE ('filename1' /CR/TTK/MTR) ('filename2') (TTP/LP/MTW)
(LIST/LISTOBJ) (OBJ/NOOBJ) (ADDR/NOADDR) ("title")

9.1.1 Input

The first group of enclosed options in the above command indicates that the compiler input may be specified from the keyboard (TTK) from cards via the card reader (CR), from magnetic tape (MTR) via the magnetic tape unit, or from a file on the disc ('filename'). If a second file name is included, then at the end of the compilation 'filename2' is deleted and a data file is created automatically on the disc with the specified file name.

Not more than one of these options may be specified. The user may, however, elect not to specify any option, in which case the compiler expects its input from the current principal input device (PID) assigned to DOPSY.

9.1.2 Output

The remaining options define the output generated. A listing may be output to either the console (TTP), magnetic tape (MTW), or the line printer (LP). No more than one output device may be entered with the command. If no entry is made, the output, if any, goes to the principal output device (POD) which is currently assigned to DOPSY. MTR and MTW may not be specified together. If LIST is selected, then source statements only are listed. If LISTOBJ is selected, then both source statements and their resulting object codes are listed. Unless NOOBJ is selected, the compiler places its translated program in working storage on the disc. The ADDR option causes the local memory address to be listed for all SET F statements. This is the default case. If a title is included in double quotes, then it is printed at the top of the compiled listing if LIST is also selected.

A typical initiation command might be:

// COMPILE '*TEST' 'TEST' LIST LP

followed by a carriage return, if entered from the VKT keyboard. This command causes the source program to be read from the file on disc name '*TEST' and produces both a data file on disc named 'TEST' and also a listing of the source statements on the line printer. Local memory addresses are printed beside local memory load statements.

When a program error is detected, one of two procedures is taken:

(1) If the error is recoverable, i.e., if the compiler can continue, FACTOR continues to compile and notifies the user of further errors.

(2) If the error is not recoverable, the DOPSY monitor is called and an asterisk is typed to notify the user that DOPSY is in control again.

9.2 INTERPRETER INTERFACING

FACTOR produces a data file which must be saved by the user if it is to be executed. Once a compilation has been completed return is made to the DOPSY system monitor with the compiled program in working storage. The user then has the option of correcting any errors in the source program and redoing the compilation, or if the program compiled error free, save the object program by creating a type "DATA" file on the disc:

```
// CREATE DATA 'filename'
```

The user program may now be executed under the control of the TOPSY interpreter.

When two filenames are entered on the compile command this step is handled automatically. The following three DOPSY commands are issued:

```
// DELETE 'filename2'
// CREATE 'filename2' DATA
// SET TTK TTP
```

This procedure always returns control to the TTK and TTP.

TOPSY is called by typing

```
// TOPSY
```

followed by a carriage return. The operation of the program from this point, using the /. LOAD command, etc., is described in the Sentry User's Manual.

9.3 ERROR MESSAGES

Most of the error messages issued by FACTOR are self-explanatory. They are listed in Table 9-1 with some comment for clarification. The error messages are accompanied by an up-arrow "↑", where appropriate, to indicate the position in the statement text where the error was detected. Two parentheses or brackets may be used in the text of the error message, since a single symbol might be obliterated by the up-arrow, making the message illegible. The total number of errors is output to the POD at the end of compilation.

TABLE 9-1 FACTOR ERROR MESSAGES

TEST	DESCRIPTION
"variable name" ALREADY DEFINED	A duplicate definition of a variable within the same block.
SEQUENCE ERROR	It is a warning message. The sequence numbers punched in columns 73-80 of the source card deck are out of order.
SS FULL	There is not enough space in memory for the number of symbols used.
NW FULL	There are too many noise words.
WORK FULL	A compound statement is too long and exceeded work area of memory.
DISC OVERFLOW	There is not enough space on the disc for the data file to be built up in working storage.
EXCESS BLOCK - - STOP OBJ	The allowable maximum number of nested blocks has been exceeded. Blocks may be nested to a depth of 8 (including Block 0).
SYSTEM 2 ERROR	Disc error. File cannot be read or closed. Compiler returns to DOPSY.
PROGRAM TOO BIG	Object code generated exceeds 777777B.
MISSING))	A left or right parenthesis has been left out.
EXPRESSION SYNTAX	An expression has been written incorrectly.
MISSING	A left or right bracket has been left out.
MISSING NAME	An identifier should have been specified in this syntactical position.
MISSING NUMBER	A number should have been specified.

TABLE 9-1 FACTOR ERROR MESSAGES (Continued)

TEST	DESCRIPTION
STATEMENT SYNTAX	A statement has been incorrectly written.
USE ERROR -- DEFINED USAGE-- [SCALAR/FOR PAR/ ARRAY/FUNCT/SUBR LABEL/LM LABEL]	Incorrect usage of variable; where SCALAR = simple variable, FOR PAR - arguments in CALL or FUNCT, ARRAY = array variable, FUNCT = function, SUBR = subroutine, LABEL = statement label, and LM LABEL = local memory label.
NUMBER SYNTAX	A number has been specified incorrectly.
INVALID TERMINATOR	An expected terminator or delimiter is incorrectly specified or missing.
I/O SPECIAL ERROR	Format used for READ or WRITE statement is incorrect.
END OF FILE INPUT	The input file has been exhausted without finding an END statement.
EXCESS VARIABLES -- STOP OBJ	The allowable maximum number of variables per block has been exceeded or too many parameters in an EXEC statement. (Maximum number of variables/block is 127, except Block 0 which is 104. The maximum number of parameters in an EXEC statement is 63.)
NUMBER EXCEEDS LIMIT	Number exceeds hardware capabilities.
PIN MISSING	No pin list with SET TG(x) statement.
SET PAGE ERROR	More than one SET PAGE statement appeared in the program, or it is preceded by something other than PGMID or REM, or a CHAIN mode was selected that is not compatible with the page size.
ILLEGAL INSTRUCTION	Instruction is not applicable to system being operated on.
LOCAL MEMORY NOT LOADED	No last local memory address defined i.e., no SET F statements before ENABLE TEST.
LABEL NOT IN BLOCK 0	ON fail-type label is not defined in Block 0.

TABLE 9-1 FACTOR ERROR MESSAGES (Continued)

TEXT	DESCRIPTION
COMPILER GENERATED "ENABLE TEST"	Number of SET F statements comprising a local memory load has exceeded the local memory size defined by SET PAGE. Compiler has generated code for an ENABLE TEST statement.
FILE NAME ERROR	Incorrect file name was used with an INSERT statement.
NUMBER EXCEEDS RANGE	The value specified exceeds the limit of the range used in the tester instruction.
WARNING NUMBER EXCEEDS LIMIT	The number used is greater than 16 bits (177777B).
FILE TYPE ERROR	Wrong file type was used for an INSERT file.
RESERVE WORD USE ERROR	A reserved word was used in place of an identifier.
WARNING - LOG IGNORED	LOG specified in MEASURE PIN instruction is ignored since it is a DMA instruction.

APPENDIX A

CHARACTER CODING (TRASCII)

Code	Char.	029 Special Character	Code	Char.	029 Special Character
00	SPACE	BLANK	40	@	4-8
01	!	11-2-8	41	A	12-1
02	"	7-8	42	B	12-2
03	#	3-8	43	C	12-3
04	$	11-3-8	44	D	12-4
05	%	0-4-8	45	F	12-6
06	&	12	46	F	12-6
07	'	5-8	47	G	12-7
10	(	12-5-8	50	H	12-8
11	)	11-5-8	51	I	12-9
12	*	11-4-8	52	J	1-1
13	+	12-6-8	53	K	11-2
14	,	0-3-8	54	L	11-3
15	-	11	55	M	11-4
16	.	12-3-8	56	N	11-5
17	/	0-1	57	O	11-6
20	0	0	60	P	11-7
21	1	1	61	Q	11-8
22	2	2	62	R	11-9
23	3	3	63	S	0-2
24	4	4	64	T	0-3
25	5	5	65	U	0-4
26	6	6	66	V	0-5
27	7	7	67	W	0-6
30	8	8	70	X	0-7
21	9	9	71	Y	0-8
32	:	0-8-2 Shift T	72	Z	0-9
33	;	11-6-8	73	[	12-4-8 <
34	<	12-0	74	\	11-7-8 ¬
35	=	6-8	75	]	0-6-8 >
36	>	11-0	76	↑	12-7-8 |
37	?	0-7-8	77	←	0-5-8 _

Note difference

APPENDIX B

READING AND WRITING OF LONG AND SHORT REGISTERS

B.1 INTRODUCTION

The Sentry Test System READ and WRITE capabilities reference the system's long and short registers. (The long and short registers consist of one bus each). The following is a description of the operation of the long and short registers.

B.1.1 Long Registers

The long registers are interfaced to the memory interface unit, called the Instruction Register, which sends information to and from the test station. It has a bus, which is used primarily for transmitting functional test data and PMU control data to the test station. The S, F, D, M, C, RZ, STROBE, INVERT and R registers are the only registers programmable with functional test data. There are other registers that are essentially like the short registers, but since the hardware resides in the test station, they are interfaced to the long register data bus and use rank address bits for identification.

B.1.2 Short Registers

The short register is interfaced to the computer accumulator. The register is used for controlling the digital to analog converter subsystems and for communicating the tester status, mode and interrupt information. Therefore, the E1, E0, EA1, EA0, etc., registers, that contain data for reference supplies are interfaced to the short register data bus.

B.2 ADDRESSING SHORT REGISTERS

Each register can be addressed by the three computer Select Peripheral Unit (SPU) commands: READ, WRITE and SPECIAL. They are each discussed as follows:

READ

> To examine the contents of a register, the register is first addressed with an SPU READ command. The contents of the register are then read into the CPU accumulator.

WRITE

To write a bit pattern into a register, the register is first addressed with an SPU WRITE command. The command is followed by the bit assignment for that register.

SPECIAL

An SPU SPECIAL command is defined as an instruction that executes some function, but no read or write data transfer is involved, e.g., Increment IND Counter or Disconnect DPS.

Each register command consists of an 8 digit octal code. The code, in effect, identifies a specific register in a specific unit and informs this register that it is about to either receive or transmit data. The data is either read out of the register, written into the register or the register performs a special function. The command format is shown in Table B-1.

Consider an example of an SPU command so that its mode and function within the system can be understood. The following table shows the MODE register SPU READ command and its octal and binary equivalents:

TABLE B-1 SPU COMMAND FORMAT

Bit Location:	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Octal Value:	0			6			6			0			0			5			2			0		
Binary Value:	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0
	OP CODE (SPU)						6=Read 4=Write 2=Special 0=No-op				TESTER REGISTER						UNIT ADDRESS							

Starting from the left, the high-order six binary bits (23-18) represent the octal code 06. This octal code is the SPU op code for the READ, WRITE, or SPECIAL command functions. The op code informs the system that it is about to address a register in a unit with a READ, WRITE or SPECIAL command.

The 3-bit value in bits 17-15 defines the command as READ, WRITE, or SPECIAL transfer. Octal 6 = READ; 4 = WRITE; 2 = SPECIAL and 0 = No op.

The six bits shown for the tester register (13-8) specify one of 64 unique registers. The remaining bits (7-0) are used to form the unit address. The tester is unit 120B.

B.2.1 Short Register Descriptions

The register number is the octal equivalent of bits 13-8 of the SPU command. The following paragraphs summarize the short registers, their addresses and special functions.

B.2.1.1 Mode Register (MR) Address 01

The mode register controls mode functions affecting the total test system as shown in Table B-2.

TABLE B-2 MODE REGISTER

BIT	FUNCTION	READ	WRITE
0	Reset Tester Short Register		X
1	Reset Tester Long Register		X
2	Monitor Mode	X	
3	Auto Mode	X	
4	Negative Logic Mode	X	X
5	Latch C Mode	X	X
6	Strobe Inhibit Mode	X	X
7	Force Strobe		X
8	Force Sync Pulse		X
9	DMA Mode		X
10	DMCRS Last function test register used	X	X
11	Function Test Suspended	X	X
12	Trip Fail	X	X
13	Functional Fail	X	X
14	Pass (Cleared by 2 or 13)	X	X
15	Spare	X	X

NOTE: A SPECIAL command clears the mode register.

B.2.1.2 Status Register (SR) Address 02

The status register contains interrupt information as shown in Table B-3.

TABLE B-3 STATUS REGISTER

BIT	FUNCTION	READ	WRITE
0	Instruction Number Compare Interrupt Enable	X	X
1	Instruction Number Compare	X	X
2	Delay Complete Interrupt Enable	X	X
3	Delay Complete Interrupt	X	X
4	Trap Interrupt Enable	X	X
5	Trap Interrupt	X	X
6	Fail Interrupt Enable	X	X
7	Fail Interrupt	X	X
8	Trip Interrupt Enable	X	X
9	DPS #1 Trip (8 must be on)	X	X
10	DPS #2 Trip (8 must be on)	X	X
11	DPS #3 Trip (8 must be on)	X	X
12	Stop Interrupt Enable	X	X
13	Stop Interrupt	X	†
14	Interrupt in Process	X	
15	Spare	X	X
	†Set by short register reset.		
NOTE: A SPECIAL command clears the status register. (B0-B13)			

Master ITest

Consol switch 6 Run Test 7

STATUS Reg. Function

0 - INC Enable
1 - INC INT
2 - Clock Enable
3 - Clock INT
4 - DMA EMABLE
5 - DMA INT
6 - FAIL Enable
7 - FAIL INT
8 - Trip Enable
9 - DPS 1 Trip
10 - DPS2 Trip
11 - DPS3 Trip

12 - Reset Enable
13 - Reset INT
14 -
15 -
16 - TSTBSY Enable
17 - TSTBSY INT
18 - STAMT Enable
19 - STAMT INT

Console switch 1
Reads out ~~with~~ All Bits
for Test 7

Console switch 3
Loops

B.2.1.3 Instruction Register (IR) Address 03

The instruction register is a buffer between B memory and the long registers via the accumulator. It contains the information listed in Table B-4.

TABLE B-4. INSTRUCTION REGISTER

BIT(S)	FUNCTION	Read	Write
0	Data	X	X
.	.	.	.
.	.	.	.
.	.	.	.
14	Data	X	X
15-18	Rank Address		X
19-21	Register Address		X
22-23	Long Reg. Read/Write Control		X

00 = WRITE & HOLD BITS 0-14
01 = WRITE & EXEC BITS 0-14
10 = READ BITS 0-14

WRITE & EXEC in DMA mode advances the Instruction Number Counter (IND) and waits for 'tester not busy'.

B.2.1.4 Memory Address Register (MAR) Address 04

The memory address register contains the memory address for the tester DMA mode. The fourteen bits are all read and written. When the tester is in the DMA mode, phase loop control automatically increments MAR.

B.2.1.5 Test Station Control Register (TSC) Address 05

The test station control register controls four channel multiplexing as listed in Table B-5.

TABLE B-5. TEST STATION CONTROL REGISTER

BITS	FUNCTION	READ	WRITE
0-1	Station Address (Write places on-line)	X	X
2-5	Start Requests from Stations 1 to 4	X	
6-9	Manual Mode from Stations 1 to 4	X	
10-13	Reset Request from Stations 1 to 4	X	

Reset and Start are written only by addressing the associated station.

B.2.1.6 Clock Burst Count Register (CBC) Address 10

The clock burst count register consists of eight bits, all read/write, which contain the count of the number of clock syncs generated per function test. This register is not used on the Sentry 600.

B.2.1.7 Time Delay Register (TD) Address 11

The time delay register consists of fourteen bits, all read/write, representing the value of a functional or DC time delay to be generated by certain tester instructions. For function tests, the least significant bit represents 0.35 microsecond and full scale is 5.734 milliseconds. The phase loop counter triggers the time delay counter when a SET F instruction is executed. For DC tests, the least significant bit represents 0.35 millisecond and the full scale value is 5.734 seconds. An SPU SPECIAL command triggers the DC time delay. The functional delay is not used by the Sentry 600.

B.2.1.8 Instruction Number Compare Register (INC) Address 14

The instruction number compare register consists of sixteen bits, all read/write, representing the test instruction number at which a pause or external sync pulse occurs. A compare interrupt is generated if the INC Interrupt is enabled, else a sync pulse occurs.

B.2.1.9 Instruction Number Display Register (IND) Address 15

The instruction number display register is a sixteen bit register, all sixteen bits read/write, representing the test instruction being executed. An SPU SPECIAL command increments the contents of the register by one. It is also incremented in DMA mode when bits 23 and 22 = 01, WRITE and EXECUTE, are set as shown.

B.2.1.10 Digital Programmable Power Supply Registers DPS1, DPS2, and DPS3 Addresses 21, 22, 24

Registers DPS1, DPS2, and DPS3 contain the range, polarity and magnitude of the DPS voltage being forced or the voltage trip point (Refer to Table B-6).

TABLE B-6. DIGITAL PROGRAMMABLE POWER SUPPLY REGISTERS

BITS	FUNCTION	Read	Write
0-9	Voltage Magnitude (Forced or Trip Value) LSB = 0.01 volt in low range = 0.04 volt in high range	X	X
10	Polarity 0 = Pos 1 = Neg	X	X
11	Range 0 = low 1 = high	X X	X X
NOTE: An SPU SPECIAL command disconnects the corresponding supply. A DPS write connects the unit to the load board.			

B.2.1.11 DPS Trip Registers - DPT1, DPT2, and DPT3 Addresses 23, 25, 26

Registers DPT1, DPT2, and DPT3 contain the current trip point or the current being forced and the trip greater than or less than control plus the DPS forcing mode control (Refer to Table B-7).

TABLE B-7. DPS TRIP REGISTERS

BIT(S)	FUNCTION	Read	Write
0-9	Current Magnitude (Forced or Trip Value) LSB = 0.1mA in low range LSB = 1. mA in high range	X	X
10	Polarity 0 = Pos 1 = Neg		
11	Range 0 = low 1 = high	X	X
13	GT or LT 1 = GT 0 = LT	X	X
14	Voltage/current 0 = voltage force 1 = current force	X	X

B.2.1.12 Reference Voltage Supply Registers S0, S1, E0, E1, EA0, EA1, EB0, EB1, EC0, EC1, SA0, SA1 Addresses 32-37, 42-47

The reference voltage supply registers contain the range, polarity and magnitude of the reference voltage supply (Refer to Table B-8).

TABLE B-8. REFERENCE VOLTAGE SUPPLY REGISTERS

BIT(S)	FUNCTION	Read	Write
0-9	Voltage Magnitude LSB = 0.01 volt in low range LSB = 0.04 volt in high range	X	X
10	Polarity 0 = Pos 1 = Neg	X	X
11	Range 0 = low 1 = high	X	X

B.3 LONG REGISTER DESCRIPTION

The registers associated with the long register are divided into two groups. The first group consists of the D, M, S, R, F, RZ, ST, INVERT, TGA0, TGA1, TGA2 and C registers. The second group consists of Pin Address, Statement Number Display, Functional Test Rate, Precision Power Source, Precision Sense Level, External Interface, Slave TSC, DC Trip, Status and Mode A and B registers, and Local Memory Registers.

B.3.1 The D, M, S, R, F, RZ, ST, INVERT, TG and C Registers

The twelve registers of the first group are discussed below.

B.3.1.1 D/DA Register, Address 02

The D/DA register is termed the input/output register. If the D register is programmed as a binary 1, the associated pin is defined as an input pin. If the D register is programmed as a binary 0, the associated pin is defined as an output pin. When a pin is defined as an input pin in the DA or DB register, a relay is energized to connect the output of the driver to the pin.

B.3.1.2 DB Register, Address 03

As above, except DB is the alternate input/output register.

B.3.1.3 M/MA Register, Address 04

The M register is the care/don't care or "mask" register. If the programmer is interested (care) in knowing the output level of a pin, the M register is programmed as a binary 1. If the programmer is not interested (don't care) in the output level of a pin, the M register is programmed as a don't care, or a binary 0. An input pin, or an output pin with an undefined state, would normally be programmed as don't care to prevent false failure indications on that pin. The don't care condition inhibits any fail indication from the output of the detector.

B.3.1.4 MB Register, Address 05

As above, except MB is the alternate mask register.

B.3.1.5 S Register, Address 10

The select reference register selects which set of reference supplies are to be used by the functional test driver for each tester pin. A binary 0 selects the primary Data/Clock reference pairs, while a binary 1 selects the alternate Data/Clock reference pairs.

B.3.1.6 RZ Register, Address 00

The RZ register selects between two data driver modes. The normal condition, binary 0, is for the Non-Return-to-Zero (NRZ) mode. A binary 1, selects the Return-to-Zero (RZ) mode for the programmed pin.

B.3.1.7 ST Register, Address 01

The strobe (ST) register selects one of two possible strobe times for each tester pin. A binary 0 selects TG7 while a binary 1 selects TG8. If the statement ENABLE DOUBLE STROBE has been executed, then a binary 0 selects both TG7 and TG8 for that pin, and a binary 1 selects TG8 only.

B.3.1.8 R Register, Address 14

The utility relay register controls the utility relays, one relay per tester pin. A binary 1 indicates a closed relay and a binary 0 indicates an open relay. The utility relays can be used for such functions as connecting a load resistor for an output pin to a programmable power supply.

B.3.1.9 F Register, Address 06

Writing F data to the register address results in loading the data into the local memory at the local memory address set by the Local Memory Main Frame Access Register (see B.3.3.11). The local memory contains the logic patterns 1 or 0 to be applied to those pins which are defined as inputs pins by the DA DB register. If the pin data is programmed as a high level (1), the high output of the driver is applied to the associated pin. If the pin data is programmed with a low level (0), the low output of the driver is applied to the pin. The local memory also contains the expected logical output of those pins which are defined as output pins.

Local memory F data also contains bits to select the primary or alternate I/O registers and Mask registers (DA/DB and MA/MB). The selection is programmed via bit 18 when writing F data to ranks 1 and 2 and shown:

Rank	Bit 18
1	0 = DA 1 = DB
2	0 = MA 1 = MB

Reading F data from local memory is accomplished by first writing the local memory address to the Local Memory Main Frame Access Register and then reading the F data via the F register address. The bits to select DA/DB and MA/NB are read back through the status and Mode B register (see B.3.3.19).

B.3.1.10 C Register, Address 12 (Read Only)

The C register stores the go/no-go results of a comparison between the actual output of a device and the expected output. A binary 1 represents a comparison failure and a binary 0 represents a pass condition.

B.3.1.11 INVERT Register Address 12 (Write Only)

This register provides a means of inverting the functional data for any pin. A binary 1 in the pin pattern enables the F data for that pin to be inverted (Return-to-One).

B.3.1.12 TGA0/TGA1/TGA2 Registers (Address 11, 13, 15)

These registers provide selection of timing generators TG1-TG6 for each tester pin. A timing generator is selected for a pin by programming the timing generator number (1-2-4 code) in the pin position in each of the three registers. The 1-bit in TGA0, the 2-bit in TGA1 and the 4-bit in TGA2. The assignment is:

TGA2	TGA1	TGA0	
0	0	0	No generator assigned
0	0	1	TG1
0	1	0	TG2
.	.	.	
.	.	.	
1	1	0	TG6
1	1	1	TG1 and TG2

B.3.2 Format of Functional Test Word

The primary difference between the long and short registers is that the short registers are variable length words. The format of a 24-bit function test word is discussed below and it is illustrated in the accompanying Figure B-9. The format for all function test registers is the same, except that the address for each register is different.

TABLE B-9. TEST WORD FUNCTION FORMAT

24 Bit Functional Test Word

23 22	21 20 19	18 17 16 15	14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Control	Register Address	Rank Address	Pin Data Field

Rank Address

3 Bits = 1 to 8 Ranks
Maximum Register Length = 8 x 15 = 120 Bits

Register Address

Bits:	21	20	19	18	Meaning
	0	0	0	0	RZ RETURN TO ZERO
	0	0	0	1	ST STROBE SELECT
	0	0	1	0	D/DA DEFINE I/O PINS
	0	0	1	1	DB DEFINE I/O PINS (ALTERNATE)
	0	1	0	0	M/MA MASK
	0	1	0	1	F FUNCTIONAL PATTERN
	1	0	0	0	S SELECT ALTERNATE REFERENCE
	1	0	0	1	TGA0 TIMING GENERATOR PIN ADDR (1)
	1	0	1	0	C COMPARE (FAIL PATTERN)/INVERT
	1	0	1	1	TGA1 TIMING GENERATOR PIN ADDR (2)
	1	1	0	0	R UTILITY RELAY
	1	1	0	1	TGA2 TIMING GENERATOR PIN ADDR (4)
	1	1	1	X	TEST STATION REGISTERS

Control

Bits:	23	22	Meaning
	0	0	WRITE AND HOLD
	0	1	WRITE AND EXECUTE
	1	0	READ
	1	1	TRAP

Register Load Times:

1.75 (1 + N) microseconds where: N = Number of ranks changing

Starting from the right, the 0 bit represents pin 1, the 1 bit represents pin 2, etc., up to bit 14 which represents pin 15.

Rank Address:

Bits 15 through 17 represent the rank to which the first 15 bits have been assigned. The ranks are determined by the normal 4-2-1 binary code minus one. 000 inserted in bits 15 through 17 represent rank 1; 111 represent rank 8. In this manner, 8 ranks of 15 pins can be programmed, thus providing a capacity of 120 pins.

Register Address:

Bits 18, 19, 20, and 21 determine the register to which the rank of 15 bits is to be sent.

Control:

Bits 22 and 23 control the read/write function. Assume the device under test is a 15-pin device; all 15 pins are programmed on one line and assigned to rank 1. The correct address code is inserted and bits 22 and 23 are programmed as 01 (write and execute), i.e., there are no more pins to program. If the device is a 45-pin device, the first 15 pins are programmed as described above, except for bits 22 and 23 which are programmed as 00 (write and hold), i.e., there are no more pins to program. The second group of 15 pins is assigned to rank 2 and, again, bits 22 and 23 are programmed as 00, i.e., there are more pins to program. The third group of 15 pins is assigned to rank 3 and bits 22 and 23 are programmed as 01 (write and execute), i.e., there are no more pins to program-execute the command. The F and S registers are MASTER/SLAVE so that all bits execute together.

B.3.3 Special Test Station Registers

The registers of the second group are discussed below.

B.3.3.1 Pin Address Register (Address = 160)

The pin address register addresses the precision measuring unit to one of the pins of the device under test or to an internal node (Refer to Table B-10). The reset state of this register is 0.

TABLE B-10. PIN ADDRESS REGISTER

BIT(S)	FUNCTION	Read	Write
0-3	Pin Number 1-15 0000 = disconnect 0001 = pin 1	X	*
4-6	Rank Number 1-8 000 1111 = pin 15 001 0001 = pin 16	X	*
7	Internal Node Address	X	*
8	Connect Voltage Conditioner (ENABLE RELAY FACTOR Instruction)	X	**
14	Write Protect Bit		X
	*Write Protected if bit 14 is a 1. **Write Protected if bit 14 is a 0.		

B.3.3.2 Statement Number Display Register (Address = 162)

The statement number display register contains the statement number to be displayed on the test station control panel. This register is interfaced to the IND register with software in TOPSY. Whenever the test sequence pauses, the software updates SND. The register is 15 bits, both read and write. This register is not used on the Sentry 600.

B.3.3.3 Functional Test Rate Register (Address = 163)

The test rate register contains the magnitude and the range of the functional test period. It controls the rate at which functional testing is executed.

TABLE B-11. TEST RATE REGISTER

BIT(S)	FUNCTION			Read	Write
0-11	Magnitude			X	X
	LSB	Range	Full Scale		
	20 ns (10ns)*	0	80 us (40us)*		
	100 ns	1	400 us		
	1 us	2	4 ms		
	10 us	3	40 ms		

*10MHz System

TABLE B-11. TEST RATE REGISTER (Continued)

BIT(S)	FUNCTION	Read	Write
12-13	Range 00 = Range 0 01 = Range 1 10 = Range 2 11 = Range 3		

B.3.3.4 Precision Power Source Register/Precision Measurement Unit Forcing Register (Address = 164)

This register contains the magnitude, polarity and range information of the PMU forcing value. It also contains the voltage or current force mode bit (Refer to Table B-12).

TABLE B-12. PPSR/PMUF REGISTER

BIT(S)	FUNCTION			Read	Write
0-9	Magnitude			X	X
	LSB	Range	Full Scale		
	1mV	1	1.023V		
	10mV	2	10.23V		
	40mV	3	40.92V		
	100mV	4	102.3mV		
	1nA	0	1.023uA		
	100nA	1	102.3uA		
	10uA	2	10.23mA		
	100uA	3	102.3mA		
10	Polarity		0 = POS 1 = NEG	X	X
11-12	Forcing Range		00 = Range 0 (Current Force only) 01 = Range 1 10 = Range 2 11 = Range 3 00 = Range 4 (Voltage Force only)	X	X
13	FV/IF		1 = Voltage Force 0 = Current Force	X	X

B.3.3.5 Precision Sense Level Register (Address = 165)

This register contains the PMU voltage clamp levels and the measuring range (Refer to Table B-13).

TABLE B-13. PRECISION SENSE LEVEL REGISTER

BIT(S)	FUNCTION	Read	Write
0-3	Voltage Clamp Magnitude	X	*
4	Clamp Control (1 = on, 0 = off)	X	*
5	Clamp Range	X	*
	Range Clamp Voltage Range of Values		
	0 (1.5+3n)V $0 \leq n \leq 15$ 1.5 to 46.5		
	1 (3.0+6n)V $0 \leq n \leq 15$ 3.0 to 93.0		
6	Allow Negative Voltages (SET CLAMP NEG)	X	*
7	Allow Positive Voltages (SET CLAMP POS)	X	*
8-9	Not used		
10	Write Protect Bit		
11-12	Sense Range	X	**
13	VM/IM (complement of PPS bit 13)	X	
	*Write Protect if bit 10 is a 1. **Write Protect if bit 10 is a 0.		

B.3.3.6 External Interface Register, Address 166B

This fifteen bit register is used to display test results and control external equipment (Refer to Table B-14). All bits are read/write. Bits 10-14 are defined by the system software. Bits 0-9 are available to the user to use in any form. The user may wish to interface these bits to an external unit (Refer to Sentry Display Control PCB, Dwg. # 97233401-04 and Wafer Prober Interface PCB, Dwg. #97239919). Signal Levels should be TTL level compatible. Bits 0-9 have a drive capability of 24 L and present a load of 1 L. Consult FST Custom Produce Engineering for interfacing to non-standard handlers, wafer probers or custom equipment.

TABLE B-14. EXTERNAL INTERFACE REGISTER

BIT(S)	FUNCTION	Read	Write
0-9	Defined by User Displayed on Station Control Panel	X	X
10	D.C. Fail Lamp*	X	X
11	D.C. Pass Lamp*	X	X
12	Functional Fail Lamp	X	X
13	Functional Pass Lamp	X	X
14	End-of-Test	X	X
	*If both DC fail and DC pass are set, the Terminal Error lamp will go 'ON'.		

B.3.3.7 Slave Test Station Control (Address = 167)

This register is a copy of the TSC short register and is used at each station equipped with relay mux. It is used to put the test head on line; and contains Reset, Manual and Start data from each of the substation test heads.

B.3.3.8 Local Memory Test Start/Delayed Memory Address Register (Address 1700)

The test start address in local memory is set by writing to this register. The address at which local memory stopped execution is obtained by reading this register. This register is not on the system reset line.

TABLE B-15. LOCAL MEMORY TEST START/DELAYED MEMORY ADDRESS REGISTER (ADDRESS 1700)

BIT	FUNCTION	Read	Write
0-9	Memory Address* Read - Halted Location Write - Start Location	X	X
10-11	Chain Address* 00 = No Chain Address 10 = Chain 2 Address 11 = Chain 4 Address	X	X
	*Bit 10 becomes the most significant bit of the local memory address if local memory size is set to >1024.		

B.3.3.9 Local Memory Minor Loop Count Register (Address 1701)

This 12 bit register contains the number of loops through the minor loop in local memory. The loop count ranges from 0 to 4095. This register is not on the system reset line. All bits are read/write.

B.3.3.10 Local Memory Major Loop Count Register (Address 1702)

This 12 bit register contains the number of loops through the major loop in local memory from the end address to location 0. The loop count ranges from 0 to 4095. This register is not on the system reset line. All bits are read/write.

B.3.3.11 Local Memory Main Frame Access Register (Address 1703)

This register controls the local memory address to which succeeding functional patterns will be loaded. It must also be set to the local memory address from which it is desired to read back a functional pattern. This register is not on the system reset line.

TABLE B-16. LOCAL MEMORY ADDRESS REGISTER

BIT	FUNCTION	Read	Write
0-9	Local Memory Address*	X	X
10-11	Chain Address* 00 = No Chain Address 10 = Chain 2 Address 11 = Chain 4 Address *Bit 10 becomes the most significant bit of the local memory address if the memory size is set to >1024.	X	X

B.3.3.12 Local Memory Minor Loop Start Address Register (Address 1704)

This register defines the minor loop start address within local memory. This register is not on the system reset line. Refer to Table B-17.

B.3.3.13 Local Memory Minor Loop End Address Register (Address 1705)

This register defines the minor loop end address within local memory. This register is not one the system reset line. Refer to Table B-17.

B.3.3.14 Local Memory Loop End/Test End Address Register (Address 1706)

This register defines the major loop end or test end address within local memory. This register is not on the system reset line. Refer to Table B-16.

B.3.3.15 Local Memory Ignore Fail Register (Address 1707)

This register sets the local memory address up to and including which fails are to be ignored. Status and Mode register A, bit 8 must be set to enable this mode. This register is not on the system reset line. Refer to Table B-16.

B.3.3.16 DC Trip Limit Register (Address = 171)

The DC trip limit register holds a go/no-go limit for PMU tests made in DMA mode (MEASURE PIN). It is also used to hold trial limits in the software A to D conversion routine. (Reference Table B-17).

TABLE B-17. DC TRIP LIMIT REGISTER

BIT(S)	FUNCTION	Read	Write
0-9	Magnitude (See PPS register)	X	*
10	Polarity (0 = POS, 1 = NEG)	X	*
11-12	Sense Range (Set by PSL register)	X	*
13	LT/GT Bit (0 = LT, 1 = GT)	X	*
14	D.C. Fail	X	
14	D.C. Strobe		X
	*Write Protect if DC strobe is 1.		

NOTE

A DC strobe is generated either by writing bit 14 or reading the register. A comparator fail on a strobe sets on a latch on bit 14; a comparator pass during a strobe resets the latch.

B.3.3.17 Chaining Register (Address 172)

This register controls the chaining of data and I/O modes for the selected pins. A binary 1 denotes Chain I/O mode.

TABLE B-18. CHAINING REGISTER

BIT	FUNCTION	Read	Write
0	Surviving Pin 1	X	X
1	Surviving Pin 5	X	X
2	Surviving Pin 9	X	X
3	Surviving Pin 13 (Chain 2 only)	X	X
4	Surviving Pin 16	X	X
5	Surviving Pin 20	X	X
6	Surviving Pin 24	X	X
7	Surviving Pin 28 (Chain 2 only)	X	X
8	Surviving Pin 31	X	X
9	Surviving Pin 35	X	X
10	Surviving Pin 39	X	X
11	Surviving Pin 43 (Chain 2 only)	X	X
12	Surviving Pin 46	X	X
13	Surviving Pin 50	X	X
14	Surviving Pin 54	X	X

B.3.3.18 Status and Mode Register A (Address = 1730)

The Status and Mode A Register contains control and status information to supplement the functional/DC fail interrupt in the status register and for local memory control (Reference Table B-19).

TABLE B-19. STATUS AND MODE REGISTER A

BIT	FUNCTION		Read	Write
0	R - Local Memory Busy W - Start Local Memory		X	X
1	R - No match W - Match Mode		X	X
2	Momentary Mode		X	X
3	Continuous Loop State		X	
4	Functional Fail		X	
5	Parametric Fail		X	
6	Functional Fail Enable		X	X
7	Parametric Fail Enable		X	X
8	Enable Ignore Fail		X	X
9	Rank Load Mode		X	X
10-11	Memory Size*		X	
12	Enable Fail in Continuous Loop		X	X
13	Continuous Loop Mode		X	X
	*LOCAL MEMORY SIZE	(B2(SAMC))	B10 B11	
		1	1 1	512
		1	0 1	1024
		1	0 0	2048
		0	0 0	4096

B.3.3.19 Status and Mode Register B (Address 1734)

The Status and Mode B Register allows control of Chain mode, Sync, Double Strobe and Interrupt Enable for a DC pass. This register may not be read or written from a FACTOR program.

TABLE B-20. STATUS AND MODE REGISTER B

BIT	FUNCTION	Read	Write
0	I/O Mode	X	X
1	Sync Mode	X	X
2	2K Page Size	X	X
3	External Sync Function Test	X	X
4	Double Strobe	X	X
5	Chain 2 Mode	X	X
6	Chain 4 Mode	X	X
7	D/L Measure Interrupt Enable	X	X
8	Local Memory Data Rank 1 (DA/DB)	X	
9	Local Memory Data Rank 2 (MA/MB)	X	
10	Local Memory Data Rank 3	X	
11	Local Memory Data Rank 4	X	

B.3.3.20 Status and Mode Register C (Address 1735)

The Status and Mode Register C contains data denoting the test system configuration. All bits in this register are read only.

TABLE B-21. STATUS AND MODE REGISTER C

BIT	FUNCTION	Read	Write
4	One nanosecond option	X	
5	Ten MHz option	X	
6	Ten MHz test head	X	

B.3.3.21 Timing Generator Pulse Width Register (Address 175)

There are eight timing generator pulse width registers. Each contains the range and magnitude data for the associated timing generator.

TABLE B-22. TIMING GENERATOR PULSE WIDTH REGISTER

BIT	FUNCTION	Read	Write
0-9	Magnitude LSB Range Full Scale 10 ns 0 10 us 100 ns 1 100 us 1 us 2 1 ms 10 us 3 10 ms	x	x
10-11	Range 00 = Range 0 01 = Range 1 10 = Range 2 11 = Range 3		
12-14	Timing Generator Number 000 Timing Generator 8 001 Timing Generator 1 . . . . 110 Timing Generator 6 111 Timing Generator 7		

B.3.3.22 Timing Generator Pulse Delay Register (Address 176)

There are eight timing generator pulse delay registers. Each contains the range and magnitude data for the associated timing generator.

TABLE B-23. TIMING GENERATOR PULSE DELAY REGISTER

BIT	FUNCTION	Read	Write
0-9	Magnitude LSB Range Full Scale 10 ns 0 10 us 100 ns 1 100 us 1 us 2 1 ms 10 us 3 10 ms	X	X
10-11	Range 00 = Range 0 01 = Range 1 10 = Range 2 11 = Range 3		
12-14	Timing Generator Number 000 Timing Generator 8 001 Timing Generator 1 . . . . 110 Timing Generator 6 111 Timing Generator 7		

B.3.3.23 Power Pin Address Register (Address 177)

This register connects a tester pin to one of the following: Data Reference pair, Tester Common, Clock Reference pair, DPS1, DPS2, or DPS3. There is a 3 bit subregister for each tester pin. These registers are addressed through the Power Pin Address Register.

TABLE B-24. POWER PIN ADDRESS REGISTER

BIT	FUNCTION	Read	Write
0-3	Pin Number (1-15)*	X	X
4-6	Rank Number*	X	X
7	Not used		
8	P **		
9	PRSB **	X	X
10	PRSA **	X	X
11	PPA Register Sub Address (0)	X	X

* Pin sub register is addressed through the Rank and Pin Number
**P, PRSB and PRSA bits set the pin connection type:

PRSA	PRSB	P	CONNECTION
0	0	0	Data Reference Pair
0	0	1	Tester Common
0	1	0	Clock Reference Pair
0	1	1	DPS2
1	0	1	DPS1
1	1	1	DPS3

B.3.3.24 Timing Generator Delay/Width Vernier (Address 17704-17774)

The timing generator delay vernier and width vernier registers contain the delay and width value within the range of .16 usec to 10.24 nsec. There is a delay vernier and width vernier for each timing generator. These registers are used only for the One-Nanosecond Option.

TABLE B-25. TIMING GENERATOR DELAY/WIDTH VERNIER

BIT	FUNCTION	Read	Write
0-3	Strobe Generator Round Trip Delay (Timing Generator 1 Delay Vernier only) These bits are hardwired to a value of 0-9 ns (LSB = 1 ns) and represent the physical round trip delay time needed to deskew the timing generator pulses and strobes. This value is automatically used by TOPSY.	X	

TABLE B-25. TIMING GENERATOR DELAY/WIDTH VERNIER (Continued)

BIT	FUNCTION	Read	Write
4-9	Timing Generator Vernier Value LSB Full Scale .16 nsec 10.24 nsec	X	X
10	Width/Delay Register 0 = Width Vernier 1 = Delay Vernier	X	X
11	Subregister Address (1)	X	X
12-14	Timing Generator Number 0 = Timing Generator 8 1 = Timing Generator 1 . . . . 7 = Timing Generator 7	X	X

B.4 FORMATTING OF FACTOR WRITE AND READ STATEMENTS

The format for programming the short or long registers is:

WRITE (xxxxB) expression;

where: xxxx is any register number, and
B is the octal indicator.

Reading information from a short or long register is:

READ (xxxxB) variable;

Tables B-27 and B-28 provide the necessary information for reading from or writing to a specific register for a specific function on the short and long registers.

TABLE B-26. SHORT REGISTER READING AND WRITING CODES

A=1= SPECIAL
A=2= WRITE
A=3= READ

Reg. No.	Register	x x x x	SPECIAL Function
0	No-op		
1	MODE	A 0 0 2	Clear Status Reg
2	MODE	A 0 0 4	Clear Status Reg
3	Instruction	A 0 0 6	
4	Memory Address	A 0 1 0	
5	TSC	A 0 1 2	
10	Clock Burst Count	A 0 2 0	
11	Time Delay	A 0 2 2	Start D.C. Delay
14	Instruction Number Compare	A 0 3 0	
15	Instruction Number Display	A 0 3 2	
21	DPS1	A 0 4 2	Disconnect DPS1
22	DPS2	A 0 4 4	Disconnect DPS2
23	DPT3	A 0 4 6	
24	DPS3	A 0 5 0	Disconnect DPS3
25	DPT2	A 0 5 2	
26	DPT1	A 0 5 4	
32	E1	A 0 6 4	
33	E0	A 0 6 6	
34	S1	A 0 7 0	
35	S0	A 0 7 2	
36	EA1	A 0 7 4	
37	EA0	A 0 7 6	
42	EB1	A 1 0 4	
43	EB0	A 1 0 6	
44	EC1	A 1 1 0	
45	EC0	A 1 1 2	
46	SA1	A 1 1 4	
47	SA0	A 1 1 6	

TABLE B-27. LONG REGISTER READING AND WRITING CODES

Register (Pins)		Register No.	Write X X X	Read X X X
RZ	1-15	000	200	400
RZ	16-30	001	201	401
RZ	31-45	002	202	402
RZ	46-60	003	203	403
RZ	61-75	004	204	404
RZ	76-90	005	205	405
RZ	91-105	006	206	406
RZ	106-120	007	207	407
ST	1-15	010	210	410
ST	16-30	011	211	411
ST	31-45	012	212	412
ST	46-60	013	213	413
ST	61—75	014	214	414
ST	76-90	015	215	415
ST	91-105	016	216	416
ST	106-120	017	217	417
D/DA	1-15	020	220	420
D/DA	16-30	021	221	421
D/DA	31-45	022	222	422
D/DA	46-60	023	223	423
D/DA	61-75	024	224	424
D/DA	76-90	025	225	425
D/DA	91-105	026	226	426
D/DA	106-120	027	227	427
DB	1-15	030	230	430
DB	16-30	031	231	431
DB	31-45	032	232	432
DB	46-60	033	233	433
DB	61-75	034	234	434
DB	76-90	035	235	435
DB	91-105	036	236	436
DB	106-120	037	237	437
M/MA	1-15	040	240	440
M/MA	16-30	041	241	441
M/MA	31-45	042	242	442
M/MA	46-60	043	243	443
M/MA	61-75	044	244	444
M/MA	76-90	045	245	445
M/MA	91-105	046	246	446
M/MA	106-120	047	247	447

TABLE B-27. LONG REGISTER READING AND WRITING CODES (Continued)

Register (Pins)		Register No.	Write X X X	Read X X X
MB	1-15	050	250	450
MB	16-30	051	251	451
MB	31-45	052	252	452
MB	46-60	053	253	453
MB	61-75	054	254	454
MB	76-90	055	255	455
MB	91-105	056	256	456
MB	106-120	057	257	457
			3703.0W To get in From Debug	
F	1-15	060 (070)*	260 (270)*	460
F	16-30	061 (071)*	261 (271)*	461
F	31-45	062	262	462
F	46-60	063	263	463
F	61-75	064	264	464
F	76-90	065	265	465
F	91-105	066	266	466
F	106-120	067	267	467
*Note: F register Rank 1, Bit 18 = 1 Sets DB register selection Bit 18 = 0 Sets DA register selection F register Rank 2, Bit 18 = 1 Sets MB register selection Bit 18 = 0 Sets MA register selection				
S	1-15	100	300	500
S	16-31	101	301	501
S	31-45	102	302	502
S	46-60	103	303	503
S	61-75	104	305	504
S	76-90	105	305	505
S	91-105	106	306	506
S	106-120	107	307	507
TGA0	1-15	110	310	510
TGA0	16-30	111	311	511
TGA0	31-45	112	312	512
TGA0	46-60	113	313	513
TGA0	61-75	114	314	514
TGA0	76-90	115	315	515
TGA0	91-105	116	316	516
TGA0	106-120	117	317	517
C/INVERT	1-15	120	320*	520*
C/INVERT	16-30	121	321*	521*
C/INVERT	31-45	122	322*	522*

TABLE B-27. LONG REGISTER READING AND WRITING CODES (Continued)

Register (Pins)	Register No.	Write X X X	Read X X X
C/INVERT 46-40	123	323*	523*
C/INVERT 61-75	124	324*	524*
C/INVERT 76-90	125	325*	525*
C/INVERT 91-105	126	326*	526*
C/INVERT 106-120	127	327*	527*
*READ: C REGISTER/WRITE: INVERT REGISTER			
TGA1 1-15	130	330	530
TGA1 16-30	131	331	531
TGA1 31-45	132	332	532
TGA1 46-60	133	333	533
TGA1 61-75	134	334	534
TGA1 76-90	135	335	535
TGA1 91-105	136	336	536
TGA1 106-120	137	337	537
R 1-15	140	340	540
R 16-30	151	351	551
R 31-45	142	342	542
R 46-60	143	343	543
R 61-75	144	344	544
R 76-90	145	345	545
R 91-105	146	346	546
R 106-120	147	347	547
TGA2 1-15	150	250	550
TGA2 16-30	151	351	551
TGA2 31-45	152	352	552
TGA2 46-60	153	353	553
TGA2 61-75	154	354	554
TGA2 76-90	155	355	555
TGA2 91-105	156	356	556
TGA2 106-120	157	357	557
Pin Address	160	360	560
Statement Number Display	162	362	562
Test Rate	163	363	563
Precision Power Source	164	364	564
Precision Sense Level	165	365	565
External Interface Register	166	366	566

TABLE B-27. LONG REGISTER READING AND WRITING CODES (Continued)

Register (Pins)	Register No.	Write X X X	Read X X X
Slave Test Station Control	167	367	567
Test Start/Delayed Memory Address	170	370	570
READ: Delayed Memory Address/WRITE: Test Start.			
Minor Loop Count	1701		
Major Loop Count	1702		
Main Frame Access	1703		
Minor Loop Start Address	1704	These registers cannot be	
Minor Loop End Address	1705	read or written from FACTOR	
Major Loop End Address	1706		
Ignore Fail	1707		
DC Trip	171	371	571
Chaining	172	372	572
Status & Mode A	173	373	573
Status & Mode B	1734	These registers cannot be	
Status & Mode C	1735	read or written from FACTOR	
Pulse Width	175	375	575
Pulse Delay	176	376	576
NOTE: Only TG8 registers can be read or written from FACTOR.			
Power Pin Address	177	377	577
Timing Generator Vernier	17704	Cannot be read/written from FACTOR.	

B.5 LONG REGISTER ASSIGNMENT IN ALTERNATE BANK

For formats and detailed information refer to the Register Formats Reference Manual, publication number 67095504.

TABLE B-28. LONG REGISTER ASSIGNMENT IN ALTERNATE BANK

REG. ADDR	SYMBOL	FUNCTION	SPM	PPM
060	DR1	DATA READOUT #1		X
062	DR2	DATA READOUT #2		X
100	T0P0	T0P0LOGICAL SCRAMBLER		X
102	HLD1/IR1	HOLD/INDEX REG		X
104	HLD2/IR2	HOLD/INDEX REG		X
106	HLD3/IR3	HOLD/INDEX REG		X
110	MAX/CMP	MAXIMUM/COMPARE		X
112	DEL1	DELTA REG.		X
114	DEL2	DELTA REG.		X
116	DEL3	DELTA REG.		X
120	CD1	CONTROL RAM		X
122	CD2	CONTROL RAM		X
124	CA	CONTROL RAM		X
126	SD/CRA	SHIFT DATA/EXEC ADDR		X
130	DRAM	DATA RAM		X
132	CSMD	CHIP SELECT & MASK		X
134	SSA	STOP & STORAGE ADDR		X
136	RFC	REFRESH COUNT		X
140-147		RESERVED FOR USE BY C.P.E.		
1701	RA	RETURN ADDR	X	
1703	FC	CLOCK BURST	X	
1704	LCS	LOOP COUNT STACK	X	
1705	LC	LOOP COUNT	X	
1706	STAM	STACK ADDR	X	
1710	0L	SEQUENTIAL LENGTH	X	
1711	Q	SEQUENTIAL PATTERN	X	
1740	LMI	LOCAL MEMORY INST.	X	

APPENDIX C

DMA MODE STATEMENTS

FACTOR statements that cause a value to be loaded in a tester long register are normally executed in direct memory access (DMA) mode. Briefly, in this mode, the system software determines the start address of a sequence of these statements, loads the MAR register, and initiates DMA mode. The hardware then executes the test program directly until an instruction that cannot be processed in this mode is encountered. Such an instruction may require several operations to be performed; these instructions are executed interpretatively by the system software. Execution in DMA mode is more efficient, particularly if the programmer structures his program so that long DMA sequences are not broken by interpretative statements or statement labels.

Table C-1 indicates which FACTOR instructions are executed in DMA mode and which long registers may be affected.

TABLE C-1. STATEMENTS EXECUTED IN DMA MODE

INSTRUCTION	LONG REGISTER NUMBER
SET D/DA/DB	020 through 027 } Functional test statements load from 1 to 8 ranks per statement depending on the specified pins.
SET M/MA/MB	040 through 047
SET F	060 through 067
SET S	100 through 107
SET R	140 through 147
SET STROBE	010 through 017
SET RZ	000 through 007
SET INVERT	120 through 127
CPMU PIN†	160
XPMU PIN	160
ENABLE/DISABLE RELAY	160
SET PMU FORCEV/FORCEI†	164
FORCE VOLTAGE/CURRENT†	164
SET PMU SENSE	165
SET CLAMP number	165
MEASURE PIN†††	171, 173
ENABLE TEST††††	1700, 1703, 1706, 173
CGEN TG(x)	110-117, 130-137, 150-157
CONN DPS(x)/TCOM/CLK	177
XCON PIN	177
AT label ††	1703
SET START ††	1700
SET MAJOR ††	1702, 1706
SET MINOR ††	1701, 1704, 1705
SET DCT	171, 165

†Only when the expression is a simple constant. (If the expression must be evaluated at execution time, the statement is executed interpretively.

††Only when the label or constant can be evaluated at compile time.

†††Only when a SET DCT statement is programmed prior to MEASURE PIN statement.

††††Any enable test except with a modifier EXT or MATCH without IMMED.

APPENDIX D

TIME DELAY RELATED STATEMENTS

FACTOR statements that start a software delay, i.e., cause the time delay register countdown to be initiated, must also be time delay dependent. This is necessary in order to avoid disturbing a previous countdown in progress, if any, which could have a longer net value than the countdown to be started. Table D-1 lists all statements that are time delay dependent; the execution of these statements which generate time delays (a fixed number if hardware initiated or a variable if software initiated) are listed in Table D-2. Note that x implies a DPS number 1 through 3 and y implies a reference voltage supply (RVS) level of 0 or 1.

TABLE D-1. TIME DELAY DEPENDENT STATEMENTS

FORCE WAIT
SET DELAY
ENABLE TRIPVx (if changing to current force mode)
ENABLE TRIPIx (if changing to voltage force mode)
FORCE FVx (after a FORCE IFx)
FORCE IFx (after a FORCE VFx or FORCE RESET)
FORCE VOLTAGE
FORCE CURRENT
FORCE PMU
SET PMU FORCEI/FORCEV/SENSE
SET D
SET M
SET R
SET S
CPMU PIN expression
XPMU PIN
MEASURE PIN
MEASURE VALUE/NODE/PIN number
The first statement of any DMA mode sequence. (Refer to Appendix C).

TABLE D-2. TIME DELAY GENERATING STATEMENTS

FORCE DELAY	Programmed DC Time Delay
SET S	0.28 millisecond
SET D, SET CLAMP, CONN (DPS, TCOM) CPMU PIN, XPMU PIN, CONN CLOCK ENABLE RELAY, DISABLE RELAY	0.56 millisecond
SET R	1.75 millisecond
~~SET PMU FORCEV, FORCEI~~ FORCE VOLTAGE, FORCE CURRENT ~~FORCE PMU~~	Programmed DC Time Delay or 0.56 millisecond with no current range change or 4 millisecond (±1 millisecond) with current range change, whichever is greater.
SET PMU SENSE	0.56 millisecond with no current range change or 4 millisecond with current rannge change.
FORCE VFx, FORCE IFx, ~~ENABLE TRIPx,~~ → ENABLE TRIP Vx, ~~ENABLE TRIPx,~~ → ENABLE TRIP Ix, XCON VFx, FORCE RESET (DPS's)	Programmed DC Time Delay or 5.37 milliseconds, whichever is greater.
SET Sy, SET SAy, FORCE Ey, FORCE EAy, FORCE EBy, FORCE ECy (HVS's)	Approximately 300 microseconds per volt of change or 0.56 millisecond, whichever is greater.
SET DCT, MEASURE PIN	56 microseconds

APPENDIX E

EXECUTION TERMINAL ERROR NUMBERS

Certain set up and programming errors cannot be detected at compilation time; these errors are discoverable only while testing. The Terminal Error lamp goes 'ON' for errors described in the following table. The error number is logged. Both the Parameter FAIL and Parameter PASS lights are 'ON', but the EOT light is 'OFF'. The error number is displayed (in binary format) in the EIR register, bits 0-10; least significant bit to the left (bit 0).

Terminal Error Number	Description
1	A program has not been loaded for this station.
2	Station is disabled (power off)
3	Value programmed is negative or exceeds the hardware limit: SET DELAY, DC > 5.734 sec SET MAJOR loop count > 4096 SET MINOR loop count > 4096
4	DMA statement execution process did not start. (Hardware error).
5	Magnitude programmed exceeds hardware limit: FORCE [E0/E1/EA0/---/EC1] >10 bits → -1024 or 1023 SET [S0/S1/SA0/SA1] >10 bits FORCE [VF1/VF2/VF3] >10 bits FORCE [IF1/IF2/IF3] >10 bits ENABLE [TRIPI1/TRIPI2/TRIPI3] >10 bits ENABLE [TRIPV1/TRIPV2/TRIPV3]>10 bits SET DCT >10 bits SET TG [DELAY/WIDTH]>10 bits
6	Value programmed is negative, zero or is outside of limit: SET PERIOD >40 milliseconds SET PERIOD >12 bits → -4096 or 4095 SET PERIOD < 200 nanosec for 5 MHZ SET PERIOD < 100 nanosec for 10 MHZ SET TG(X) [DELAY/WIDTH]<10 nanosec
21	Value outside of limits set by ENABLE IHI/ILO
22	Value outside of limits set by VHI/VLO
23	Pin number is greater than 120; CPMU PIN
24	Value programmed for RVS exceeds hardware limit: [SET S0/S1/SA0/SA1] +6, -30V for 5 MHZ [SET S0/S1/SA0/SA1] +6, -16V for 10 MHZ FORCE [E0/E1/---EC1]→as above

Terminal Error Number		Description
26		Illegal OPCODE in FACTOR interpretive tester statement.
31†		FACTOR magtape read error (File skip forward executed to move tape to the next tape file).
33†		FACTOR magtape write error (File skip backward executed to move tape to the start of the last file. When start is pressed the program continues execution from this tape location).
35†		FACTOR magtape EOT on write.
36†		FACTOR magtape EOT on read.
37†		FACTOR magtape memory protect on tape read.
40†		FACTOR magtape data count less than 7 or greater than assigned buffer size. Also a memory overflow may have occurred.
42†		FACTOR magtape irrecoverable error.
50	(a)	No DCL statement appears before this reference to the array element.
	(b)	Array has zero or negative number of elements.
51	(a)	The number of actual parameters does not agree with the number of formal parameters.
	(b)	TOPSY internal address error during store of a value, array element or formal value.
52	(a)	Array subscript exceeds 8388607, is negative or greater than the array size.
	(b)	Attempt to change array element 0 (i.e., the array size)
53	(a)	The number of entries on the top of the working stack is less than required for current statement execution. (System error.)
	(b)	A block header memory address of zero has been encountered during update of Current Active Block pointer table. (System error.)
54	(a)	Array size declared exceeds 8388607 or available memory.
	(b)	Memory buffer available for the test program is less than 1 disc sector (48 words).
55		The statement to be executed is not within the FACTOR object file. (System error).
56		Illegal FACTOR data code. (System error).
57		The number of array elements being initialized exceeds the declared array size.
58	(a)	FACTOR I/O started without previous I/O being completed.
	(b)	Text is to be output without I/O being initialized.
	(c)	Column formatting outside of I/O process.
	(d)	Literal variable outside of I/O process.
	(e)	Column formatting is allowed on output only.

Terminal Error Number	Description
59	For statement loop control start value is less than end value.
60	Assembly Language Linkage program or PPM microprogram is not on the disc.
61	Assembly Language Linkage program or PPM microprogram load entry point overlaps the top of the working stack.
62	Arithmetic or logical operation overflow (ADD, SUBTRACT, MULTIPLY, DIVIDE, EXPONENTIATION, AND, OR, EOR, NOT, NEGATE).
67	DOF (disc output file) is not open.
68	Attempt to read beyond EOF (end of file) if DIF (disc input file) or DIF not open.
69	Attempt to write beyond EOF (end of file) of DOF (disc output file)
70	Attempt to execute a program without a SET PAGE statement on a SII and SVII or a SII and SVII program on a S200/400 or a system without a tester.
71	Local memory size requested exceeds local memory available.
72	Programmed timing generator delay/width error (checked on SET PERIOD): Delay + Width ≥ Period SII/SVII Delay or Width ≥ Period SII/SVII + 1 nsec. option Delay or Width range ≥ Period range Width range ≥ Delay range
74	Local memory address is negative or exceeds size requested by SET PAGE.
75	Attempt to execute a 10 MHZ FACTOR statement on a 5 MHZ system. (ENABLE/DISABLE LATCHES, ENABLE TEST, EXT)
76	Attempt to execute a program with SET PAGE, SPO on a Sentry II without the Sequence Processor Module.
77	Attempt to execute a PPO program on a Sentry II without the Pattern Processor Module, i.e., one of the following was encountered: SET APERIOD SET ATG4 [DELAY/WIDTH] SET PPM REXEC

† Tape status issued in octal. On terminal errors 35 or 36 a tape rewind is executed.

Terminal Error Number	Description
78	Attempt to execute a program with extended capabilities on a system without the hardware, i.e., one of the following was encountered: SET IFAIL, COUNT SET IOM3 SET Q ENABLE TEST AMATCH
79	Attempt to execute a 2v /2mv program on standard hardware.
81	The microprogram called by REXEC contains an assembly error.
82	The module number passed to the microprogram by the REXEC statement is negative.
83	The module number passed to the microprogram by the REXEC statement is greater than the number of modules in the microprogram.
84	Error in DMA loading the control RAM or in DMA loading the PPM registers when a PPM microprogram is executed.
100-999	Terminal errors generated by ALLINK programs.

APPENDIX F

TABLE F-1. CALIBRATION RESISTOR TABLE

TVFY LOAD BOARD PIN	CALIBRATION RESISTANCE	VOLTAGE RANGE	CURRENT RANGE
1	10	1 (1V)*	3 (100mA)*
3	100	1	2 (10mA)*
11	10 K	1	1 (100-a)*
17	1 M	1	0 (1 a)*
3	100	2 (10V)	3
7	1 K	2	2
13	100 K	2	1
19	10 M	2	0
5	400	3 (40V)*	3
9	4	3	2
15	400 K	3	1
21	40 M	3	0

*Full scale.

APPENDIX G

INTERNAL NODE MEASUREMENT

Internal nodes are listed in the table below. The PMU must be programmed to force zero current in range 2 before the PMU is connected to any internal node, including the load current nodes (143-145). The voltage sensing range and the limit is programmed according to the expected value.

TABLE G-1. INTERNAL NODES MEASUREMENT

NODE NUMBER			MEASURED PARAMETER
DECIMAL	OCTAL	NAME	DESCRIPTION
128	200	S1	COMPARATOR S1 REF. VOLTAGE
129	201	S0	COMPARATOR S0 REF. VOLTAGE
130	202	E1	FORCING LEVEL E1 REF. VOLTAGE
131	203	E0	FORCING LEVEL E0 REF. VOLTAGE
132	204	EA1	FORCING LEVEL EA1 REF. VOLTAGE
133	205	EA0	FORCING LEVEL EA0 REF. VOLTAGE
134	206	EB1	FORCING LEVEL EB1 REF. VOLTAGE
135	207	EB0	FORCING LEVEL EB0 REF. VOLTAGE
136	210	EC1	FORCING LEVEL EC1 REF. VOLTAGE
137	211	EC0	FORCING LEVEL EC0 REF. VOLTAGE
138	212	SA1	COMPARATOR SA1 REF. VOLTAGE
139	213	SA0	COMPARATOR SA0 REF. VOLTAGE
140	214	VF1	VOLTAGE FORCING UNIT 1 OUTPUT VOL.
141	215	VF2	VOLTAGE FORCING UNIT 2 OUTPUT VOL.
142	216	VF3	VOLTAGE FORCING UNIT 3 OUTPUT VOL.
143	217	TRIP1	VF1 LOAD CURRENT
144	220	TRIP2	VF2 LOAD CURRENT
145	221	TRIP3	VF3 LOAD CURRENT

Load currents are proportional to the voltage drop across an internally connected resistor chosen such that the full scale measurement value is 1.023 volts. If the power supply is in range 3, 1 millivolt of voltage drop corresponds to 1 milliamp of load current. If the power supply is in range 2, 1 millivolt of voltage drop corresponds to 0.1 milliamp of load current.

When load currents are measured, the measured voltage is automatically converted into the corresponding current value by the FACTOR "MEASURE NODE" statement.

APPENDIX H

STATEMENT LIST

The following statement forms are allowed in programming the SII or SVII.

H.1 BASIC STATEMENT FORMS

BLOCK

Creates groups of program statements.

SUBR identifier;
SUBR identifier (parameters);
FUNCT identifier (parameters);

Delineates a group of statements which can be repeated with a call statement.

END;

Closes BLOCK or subroutine or

CALL identifier;
CALL identifier (parameters);

Subroutine is executed and at completion control is returned to the calling routine.

INSERT filename;

Allows inclusion and compilation of the names source file at point specified.

NOISE xxx, xxx, -----;

"Words listed as noise may be used in any statement but is ignored by the compiler. (Must not include reserved words.)

REM ------;

Allows user to give documentation which is ignored by the compiler.

PAGE;

Ejects paper to top of form if listing to line printer.

LIST;
NOLIST; Controls listing.

EXEC identifier (parameters);

DCL V1 V2, VN;
DCL V1/value1/, V2/value2/;
DCL V1 asize /avalue1,avalue2, . . ./;

Declares variables and arrays which may be assigned values.

GOTO label;
GOTO (label1, label2, . . ./labeln) expression;

Causes unconditional branch.

LABEL: - - -; -

An address is assigned to label to allow branching to label.

IF relation THEN statement;
IF relation THEN BEGIN - - - - - END;
IF relation THEN statement1 ELSE statement2;

Statements are executed if the 'if' condition is met.

FOR variable = expression THRU expression
BY increment DO statement;

Allows looping under control of a variable.

PAUSE expression;

Program pauses -- value of expression printed on POD

FORMS OF ARITHMETIC STATEMENTS

variable = expression;

Variable is assigned a value.

ARITHMETIC EXPRESSIONS:

With parenthetical expressions:
Read from left to right only:

Arithmetic replacement statements may use the following operators:

+ ADDITION
- SUBTRACTION

SET CHAIN [TWO/FOUR] pin list;

SET CHAIN OFF:

Allows 2 or 4 test patterns to be generated for each SET F. 'OFF' restores non-chaining mode.

SET [D/M/S/R/F] (*) binary-pin-pattern;

Definition, mask, select, relay, or function registers are set to pattern.

SET SI binary-pattern;

Similar to set S, but generates interpretive code.

SET FI binary-pattern;

Selectively changes specific bits in local memory.

SET [M/MA/MB] binary-pattern;

Sets primary or alternate pin mask register.

SET [D/DA/DB]binary-pattern;

Sets either primary or alternate I/O pin definition register.

SET RZ binary-pattern;

Sets any pin (Data or Clock) to be either NRZ (0) or RZ (1).

SET PAGE integer (,SPM);

Indicates to compiler and TOPSY that program is for hi-speed station, specifies size of a local memory load, and selects SPM option.

SET TGx [DELAY/WIDTH] expression (,RNG0/,RNG1/,RNG2/,RNG3);

Sets delay and width of timing generators 1-8.

SET MPIN integer;

Defines maximum pin count allowed.

SET VOFFSET number;

Specifies an offset voltage to be added to all tester statements which control a voltage level.

SET DCT [LT/GT] expression (,RNG0/,RNG1/,RNG2/,RNG3/,RNG4);

Sets a hardware passfail limit for one DCT threshold at a time, for MEASURE PIN.

H.4 ENABLE FORMS

ENABLE [ILO/IHI/VLO/VHI] [GT/LT] number;

Enables limit comparisons to be made on all programmed current/voltage operands prior to an instruction execution.

ENABLE [TRIPV1/TRIPV2/TRIPV3] [LT/GT] expression (,RNG2/,RNG3);

Enables the voltage-trip detector of the corresponding current forcing unit.

ENABLE [TRIPI1/TRIPI2/TRIPI3] [LT/GT] expression (,RNG2/,RNG3);

Enables current-trip detector of the corresponding voltage forcing unit.

[ENABLE/DISABLE] LATCHES;

Determines if C register is to be cleared prior to strobing functional test comparators.

ENABLE ACCESS;

Forces a disc access to reload the memory buffer.

[ENABLE/DISABLE] RELAY;

Determines if voltage conditioner remains connected to a pin when the PMU is connected.

DISABLE TRIPS;

Clears trip limits set up with Enables.

ENABLE [DCT0/DCT1] [LT/GT] expression;

Forms a software pass-fail threshold, or if both DCT0 and DCT1 are specified, a pass-fail window, for 'MEASURE VALUE'.

DISABLE [DCT0/DCT1] ;

Disables comparison limits.

[ENABLE/DISABLE] DOUBLE STROBE:

Pins indicated by a 0 in 'SET STROBE' will be strobed by both TG7 and TG8.

ENABLE [MA/MB/DA/DB] (,MA/MB/DA/DB);

Specifies which I/O definition and/or mask register is to be used.

H.5 FORCE FORMS

FORCE [VF1/VF2/VF3] expression (,RNG2/,RNG3);

Forces DPS voltage supply to value specified.

FORCE [IF1/IF2/IF3] expression (RNG2/,RNG3);

DPS unit is to force current specified.

FORCE [E0/E1/EA0/EA1/EB0/EB1/EC0/EC1] expression (,RNG2/,RNG3);

Forces voltage conditioner reference supplies to programmed value.

FORCE PMU expression;

Forces output of PMU to value specified.

FORCE VOLTAGE expression (,RNG1/,RNG2/,RNG3/,RNG4);

Forces PMU to voltage specified.

FORCE CURRENT expression (,RNG0/,RNG1/,RNG2/,RNG3);

Forces PMU to current specified.

FORCE RESET;

Clears all programmable test conditions and causes a hardware reset.

FORCE DELAY;

Forces the time delay to occur and to wait until tester not busy.

FORCE WAIT;

Forces tester to wait until 'tester not busy'.

H.6 MISCELLANEOUS FORMS

ON [DCT/FCT/TRIP] , label;

Causes program branch to label on failure.

XCON [VF1/VF2/VF3] ;

Specified voltage forcing unit is disconnected from the test head.

CPMU PIN expression ;

PMU is connected to pin specified.

XPMU PIN;

Disconnects PMU.

MEASURE PIN ;

Pass-fail comparison is made with programmed limit. No floating point conversion.

MEASURE [VALUE/NODE number] (,LOG);

Measurement is made and a software analog-to-digital conversion takes place, with result stored in global variable 'VALUE'.

MEASURE VARIABLE variable (,LOG)

Similar to MEASURE VALUE except that no measurement is made but the value of the variable is used and compared against the enabled DCT0/1 limits.

CLEAR FAIL [DCT/FCT/TRIP];

Previous fail indicator is cleared.

CLEAR [DCT/FCT/TRIP] ;

Clear previous ON fail-type, label instruction.

CONN [DPS1/DPS2/DPS3/TCOM/CLK] pin list;

Connects listed pins to power supply or to tester common or defines them as clock pins.

XCON PIN pin list;

Reconnects pin to selected data reference.

CGEN [TG1/TG2/TG3/TG4/TG5/TG6/TG12] pinlist;

Connects listed pin to specified timing generator

H.7 LOCAL MEMORY MANAGEMENT

In the following instructions, addresses refer to locations in local memory and are of the following form:

label/constant/variable/label+-constant/label+-variable

SET START test-start-address ;

Specifies functional test start address.

AT memory-address ;

Designates memory address at which it is desired to make modifications.

SET MAJOR major-loop-count, major-loop-end-address;

Defines major loop within local memory. (Also used to redefine (L), test end).

SET MINOR minor-loop-count, minor-loop-start-address, minor-loop-end-address;

Defines minor loop within local memory.

ENABLE TEST [NORMAL/CONTINUOUS/MOMENTARY/IFAIL] (,EXT/,EXTA);
ENABLE TEST MATCH (,EXT/,EXTA/,IMMED);

Initiates testing in mode specified.

SET IFAIL memory-address (,COUNT);

Sets local memory address or step count up to which fails will be ignored.

KEY

[X/Y/Z]	one of options is required.
(X/Y/Z)	one of options may be used but none is required.
integer	user must select appropriate expression or number.
number	any floating point number but may not be a variable.
expression	any floating point number or variable or arithmetic combination of numbers and variables.

APPENDIX I

READ/WRITE MAGNETIC TAPE STATEMENTS

I.1 DEFINITION

The FACTOR READ (MTR) and WRITE (MTW) statements are defined as follows:

READ (MTR) "name" V1, V2, V3, V4;

WRITE (MTW) "name" V1, V2, V3, V4;

The terms V1 through V4 represent array identifiers which have been declared prior to executing the READ/WRITE statements. There may be one to four arrays per statement, of at least seven elements. The term "name", enclosed by double quotations specifies the file name of the data to be written on magnetic tape.

Execution of the WRITE (MTW) statement causes the Array Data Segment(s) to be written on magnetic tape at the tape's current position. Table I.1 gives the format specification of an Array Data Segment.

An EOF (End of File) tape mark is written under the following conditions:

(a) When End of Test occurs and the tester is in automatic mode, and at least one WRITE (MTW) statements has been executed.

(b) At the completion of each WRITE (MTW) statement when the tester is in manual mode.

(c) When the tester pauses as the result of a TOPSY "PAUSE" command or a FACTOR "PAUSE" and at least one WRITE (MTW) statement has been executed.

Only one magnetic tape unit may be used with the SENTRY II/VII even though the system may have more than one test station. Any of the four stations which execute programs containing READ (MTR) and/or WRITE (MTW) statements have access to the magnetic tape unit. To avoid having read/write conflicts which could destroy valid data, only one station of a multiple station system should execute programs which utilize magnetic tape.

TABLE I-1. ARRAY DATA SEGMENT

Physical Record Number	Word Number	Contents
1	1	8 character TRASCII code word
	2	for the file "name".
	3	Data record length = N (integer)
	4	0
	5	0
	6	0
2	1	Words 1 to N are the contents of
	2	one variable length FACTOR array.
	.	(FST floating point)
	.	
	.	The maximum number of words per each record is recommended to be 512,
2	N	(but must not be fewer than 7).

I.2 READ ERRORS

I.2.1 Array Element Count Error

If the word count of the tape data exceeds the number of elements in the specified array(s) or if the declared array has less than seven (7) elements, the system issues terminal error 40. If the array size is less than 7 elements, the tape is not advanced. When the tape data word count exceeds the array size, the tape will have advanced to the end of the excessive tape segment prior to accepting the next station "START".

I.2.2 Data Transfer Error

If a data transfer error is detected, terminal error 31 is issued and the tape is positioned forward to the beginning of the next file. The TOPSY program statement counter is reset such that when station "START" is depressed, the loaded program begins execution at statement one (1).

I.2.3 End of Tape Error

If the End Of Tape (EOT) mark is encountered before the specified segment is found, the tape is rewound to the Beginning of Tape (BOT) mark and terminal error 36 is issued. The TOPSY program statement counter is reset such that when station "START" is depressed, the loaded program begins execution at statement one (1).

I.2.4 Memory Protect

If the memory protect switch located on the tape controller is enabled, the system issues terminal error 37 and the TOPSY program counter is reset such that when station "START" is depressed, the loaded program begins execution at statement one (1).

I.3 WRITE ERRORS

I.3.1 Data Transfer Error

If a data transfer error is detected, terminal error 33 is issued and the tape is positioned backwards to the start of the current file. The TOPSY program counter is reset so that when station "START" is depressed, the loaded program will begin execution at statement one (1).

I.3.2 End of Tape Error

If the End of Tape (EOT) mark is encountered prior to completion of a WRITE operation, the tape is rewound and the system issues terminal error 35. The station is unloaded so that it cannot be restarted by pushing station "START". This avoids accidental writing over good data at the beginning of the tape.

I.3.3 Array Element Count Error

If an array of size less than seven (7) appears in the WRITE statement, terminal error 40 is issued.

I.3.4 Unrecoverable Errors

Any errors other than those described above are considered to be unrecoverable and the system issues terminal error 40. The TOPSY program statement counter is reset.

I.4 STANDARD MAG TAPE OPERATION IN TOPSY

Before executing a program employing mag tape read or write statements, the operator must set the tape at the BOT marker of the tape file the program is to read or write.

The instructions relating to the periodic maintenance of the mag tape should be attended to if error free operation is desired.

Before executing the TOPSY program, the REMOTE switch on the mag tape unit must be enabled.

After the above steps it is only necessary to execute the TOPSY program from the tester station. All mag tape controls are performed by TOPSY.

I.5 UNUSUAL MAG TAPE OPERATION IN TOPSY

I.5.1 Catastrophic Errors

If a 'catastrophic' error occurs during mag tape operation and the user desired to make some attempts to recover then the following course of action is recommended as a desperation procedure.

I.5.1.1 Write Operation

Go back to DOPSY manually and execute two tape mark writes, viz:

```
// MTAP TMARK
// MTAP TMARK
```

followed by:

```
// MTAP SKIP BACK 1 RECS
```

I.5.1.2 Read Operation

Go back to DOPSY manually. Rewind the tape via the tape transport REWIND switch and restart TOPSY.

I.5.1.3 Warning

The user should be aware that these recovery actions bypass the normal TOPSY-DOPSY return and, consequently, do not update the present state of TOPSY. When TOPSY is reentered, it is initialized to the state prior to the last return to DOPSY.

APPENDIX J

FLOATING-POINT PACKAGE

Calling sequences and timings for the individual subroutines of the floating-point package, and the internal format and range of floating-point values are discussed below.

Calling Sequences:

Type 1 Subroutines:

FCAM, FMUL, FDIV, FSUB, FADD, FAND, FOR, FEOR

All of these subroutines require two floating-point parameters. Value 1 must be in the A-register, value 2 is obtained indirectly, (the floating-point result is returned in the A-register):

```
CALL FPSUBR call each subroutine by name with value 1 in A-reg
NOP ADDRV2 address of value 2 in address field
```

Type 2 Subroutines:

(FNOT, FNEG, SQRTF, FLOG, FEXP):

All of these subroutines require one floating-point value in the A-register. (The floating-point result is returned in the A-register):

```
CALL FPSUBR  call each subroutine by name with value in A-reg
```

Type 3 Subroutine (One-Word Fixed-to-Floating):

On entry the A-register has a signed integer value. On exit the A-register has the required floating-point value:

```
CALL FFLT
```

Type 4 Subroutine (One-Word Floating-to-Fixed):

On entry the A-register has a floating-point value. On exit the A-register has the one-word (signed) integer value:

```
CALL FFIX
```

Type 5 Subroutine (Two Word Fixed To Floating)

On entry the A-register has a (signed) integer value and the E-register has the power of 10 multiplier (+ or - power). On exit the A-register has the required floating-point value:

CALL FFLTS

Type 6 Subroutine(Two-Word Floating-To-Fixed):

On entry the A-register has the floating-point value to be fixed. On exit the A-register contains a signed integer and the E-register contains the power of 10 multiplier:

CALL FFIXS

Timings:

FNEG: 3 cycles (constant)

FCAM: 24 cycles (constant)

FDIV: 103 cycles (maximum)..if either or both arguments negative
101 cycles for both arguments positive
96 cycles for overflow: both arguments negative
95 cycles for overflow: either argument negative
94 cycles for overflow: both arguments positive
89 cycles for underflow (0 result): both arguments negative
88 cycles for underflow (0 result): either argument negative
87 cycles for underflow (0 result): both arguments positive
31 cycles for 0 divisor (dividend negative): overflow bit set
30 cycles for 0 divisor (dividend positive): overflow bit set
3 cycles for 0 dividend

FMUL: 101 cycles (maximum)..if either or both arguments negative
99 cycles for both arguments positive
94 cycles for overflow: both arguments negative
93 cycles for overflow: either argument negative
92 cycles for overflow: both arguments positive
87 cycles for underflow (0 result): both arguments negative
86 cycles for underflow (0 result): either argument negative

85 cycles for underflow (0 result): both arguments positive
26 cycles for 0 multiplier (0 result): multiplicand negative
25 cycles for 0 multiplier (0 result): multiplicand positive
3 cycles for 0 multiplicand

FFLT: 36 cycles (maximum): values -1 through -7
35 cycles: -10 thru -377 octal,)-8 thru -255 decimal)
34 cycles: -400 thru -17777 octal
34 cycles (maximum for positive values): 1 thru 7 octal
33 cycles: -20000 thru -37777777 octal
33 cycles: 10 through 377 octal
32 cycles: 400 through 17777 octal
31 cycles: 20000 through 37777777 octal
3 cycles: value 0

FFIX: 41 cycles (maximum): 53100000 (-1000000) to 52200001 (-17777700)
40 cycles: 54400004 (-20000) to 53200001 (-777774)
39 cycles: 55600200 (-400) to 54400005 (-17777)
39 cycles: 24700000 (1000000) to 25577777 (17777700)
38 cycles: 57500000 (-1) to 55600201 (-377)
38 cycles: 23377774 (20000) to 24577777 (777774)
37 cycles: 22177600 (400) to 23377773 (17777)
36 cycles: 20300000 (1) to 22177577 (377)
24 cycles: negative overflow: threshold: 52100000
23 cycles: positive overflow: threshold: 25700000
17 cycles: negative underflow: threshold: 57600001
16 cycles: positive underflow: threshold: 20177777
3 cycles: value 0

SQRTF: 160 cycles (maximum) .. exponent even
156 cycles if exponent odd
10 cycles for negative argument: overflow bit set
3 cycles for 0 argument

Note 1: By 'signed' is meant that a negative value is the internal machine representation for negative values (two's complement of the corresponding positive value).

Note 2: Any error results in the overflow bit being set on exit.

CAUTION

It is the programmer's own responsibility to clear this bit before a subsequent subroutine call.

Note 3: Internal floating-point format:

Bit 23: Sign (of value, i.e., sign of 'MANTISSA')
0: value is positive,
1: value is negative (TCA of positive value)

Bits 22--16: Biased characteristic (7 bits)
(100)8 = BIAS, i.e, represents 0 characteristic
(177)8 represents (63)10 characteristic
(000)8 represents (-64)10 characteristic

Bits 15--0: Fractional mantissa, i.e., octal point is to the left of bit 15. Thus bit 15 = 1/2,
bit 14 = 1/4, etc.

Examples: Representation of (23.5)10:
(23.5)10 = (27.4)8 = 10111.1)2
= (0.101111)2 *2 5
Thus to create the floating-point number:-
number is positive, therefore sign = 0
characteristic = (105)8 = (1000101)2
i.e., bias + 5 --(100 + 5 from 2 5 above)
mantissa = (1011110000000000)2
i.e., the (0.101111)2 from above left-justified in 16 bits.
Putting the sign, characteristic and mantissa together: 0 1000101 1011110000000000
i.e., (0100010110111100000000000)2
i.e., (2 1 3 6 0 0 0)8
Thus (21336000)8 is the required value.

Representation of (-23.5)10:
Take the two's complement of the floating-point value for (23.5(10, i.e., 2's complement of (21336000)8
i.e., (56442000)8. This is the required value.

Note 4: Range of Floating-Point Values:

The smallest positive floating-point value which FPP can handle is represented by (11011111)8. The value of this is (.1)2 *2 (-64) = (.5)10 *2 (-64) = 2 (-65) = 2.711E-20 (approximately).

The largest positive floating-point value which FPP can handle is represented by (37777777)8. The value of this is (.777774)8 *2 (63) = 2 (62) = 2 (61) + 2 (60) + 2 (59).... + 2 (47) = 2 (47) = 2 (63) - 2 (47) = 9.445E18 (approximately).

The negative range exactly parallels the positive range.

APPENDIX K

COMPILER GENERATED TESTER OP CODES-DMA

OP CODE	OPTION	DESCRIPTION
00000000		SET RZ
00000000		SET XOR (Alternate Bank)
01000000	*	SET STROBE
02000000	*	SET DA/D
03000000	*	SET DB
04000000	*	SET MA/M
04000000	*	SET CRO (Alternate Bank)
05000000	*	SET MB
06000000	*	SET F
06710004	S	LSET IX
06730001	S	LSET STROBE
06730002	S	LSET RZ
06730003	S	LSET XOR
06730004	S	LSET INVERT/I
06730005	S	LCGEN TG1
06730006	S	LCGEN TG2
06730007	S	LCGEN TG3
06730010	S	LSET DB
06730011	S	LSET DA
06730012	S	LSET MB
06730013	S	LSET MA
06740000	S	SET F-LCALL
06750000	S	LSET IX-LCALL
06760000	S	SET F-LGOTO
06770000	S	SET F-LEND
07700000	S	LSUBR NORMAL
07710000	S	LSUBR MATCH
07720000	S	LSUBR CONTINUOUS
07740000	S	SET FC NORMAL
07750000	S	SET FC MATCH
07760000	S	SET FC CONTINUOUS
10000000	*	SET S
12000000	*	SET INVERT

Options

S - Sequence Processor
P - Pattern Generator

COMPILER GENERATED TESTER OP CODES-DMA

OP CODE	OPTION	DESCRIPTION
14000000	*	SET R
16000000	*	CPMU/XPMU PIN
16040000	*	DISABLE RELAY
16040400	*	ENABLE RELAY
16400000	*	FORCE CURRENT
16420000	*	FORCE VOLTAGE
16420002	*	SET PMU FORCEV
16500000	*	SET CLAMP
16502000	*	SET DCT (PSL REGISTER)
17000000	*	SET START
17010000	*	SET MINOR
17020000	*	SET MAJOR
17100000	*	SET DCT
17140000	*	MEASURE PIN (DCT. REGISTER)
17300000	*	ENABLE TEST
17300200	*	MEASURE PIN (2nd WORD) (SAMA REGISTER)
17360000	*	ENABLE/DISABLE SPLIT/RTO/MUX/IMAS
17374000	*	ENABLE ALTERNATE BANK
17400000	P	RD/WR ONE/ZERO
17400100	P	RD/WR CHECK/NCHECK
17400200	P	BRANCH UNLESS
17400300	P	BRANCH TO/RESET
17410000	P	PGEN LOAD/START/ENABLE/DISABLE PGEN
17420000	P	SET PGEN1 (WORD 2)
17430000	P	SET PGEN1 (WORD 3)
17440000	P	SET PGENA
17443200	P	SET PGEN1
17443400	P	SET PGEND
17443600	P	SET PGENDN
17444000	P	PIN CONFIGURE (WORD1)
17444400	P	PIN CONFIGURE (WORD2)
17443600	P	SET PGENCN
17447600	P	SET PGENC
17700000	*	XCON PIN
17700400	*	CONN TCOM
17701000	*	CONN CLK
17701400	*	CONN DPS2
17702400	*	CONN DPS1
17703400	*	CONN DPS3

Options

N - Sequence Processor
P - Pattern Generator
T - Time

COMPILER GENERATED OP CODES - INTERPRETIVE (TESTER)

OP CODE	S6	DESCRIPTION
50000001	*	FUNCTION TRAP
50000002	*	PMU TRAP
50100000	*	SET PAGE
50200000	*	SET START
50220000	*	SET IFAIL
50300000	*	AT
50400000	*	SET MAJOR (ADDR)
50500000	*	SET MINOR (ADDR)
50600000	*	SET MAJOR/MINOR (LOOP COUNT)
50700000	*	SET PERIOD
51000000	*	SET TG1
51100000	*	SET TG2
51200000	*	SET TG3
51300000	*	SET TG4
51400000	*	SET TG5
51500000	*	SET TG6
51600000	*	SET TG7
51700000	*	SET TG8
52014000	*	SET FI
52020000	*	SET SI
52101000	*	ENABLE TEST
52102000	*	SET CHAIN
52104000	*	DISABLE DOUBLE STROBE
52104020	*	ENABLE DOUBLE STROBE
52110000	*	SET IOMODE
52130000	*	SET IOM3 (ALTERNATE BANK)
52200000	*	SET Q
52300000	*	SET ATG4
60100000	*	ENABLE TRIP1/TRIPI1
60200000	*	ENABLE TRIP2/TRIPI2
60300000	*	ENABLE TRIP3/TRIPI3
60400000	*	FORCE CURRENT
60520000	*	FORCE VOLTAGE
60700000	*	FORCE PMU
61100000	*	SET DELAY, DC
61200000	*	CPMU PIN
61202000	*	*XCON VF1/VF2/VF3
61300000	*	ENABLE TRIPV1
61400000	*	ENABLE TRIPV2
61500000	*	ENABLE TRIPV3
61600000	*	MEASURE PIN
61600000	*	MEASURE VALUE/NODE
61640000	*	MEASURE VARIABLE
61700000	*	SET PMU
62000000	*	FORCE E0
62020000	*	FOREC E1
62100000	*	FORCE EA0

COMPILER GENERATED OP CODES - INTERPRETIVE (TESTER)

OP CODE	S6	DESCRIPTION
62120000	*	FORCE EA1
62200000	*	FORCE EB0
62220000	*	FORCE EB1
62300000	*	FORCE EC0
62320000	*	FORCE EC1
62400000	*	SET CLAMP
62460000	*	SET VOFFSET
62500000	*	FORCE VF1
62600000	*	FORCE FV2
62700000	*	FORCE VF3
63000000	*	SET S0
63020000	*	SET S1
63100000	*	DISABLE DCT0
63100001	*	ENABLE DCT0
63200000	*	DISABLE DCT1
63200001	*	ENABLE DCT1
63300000	*	ENABLE ILO
63400000	*	ENABLE IHI
63500000	*	ENABLE VLO
63600000	*	ENABLE VHI
64000003	*	FORCE RESET
64000020	*	SET LOGIC NEGATIVE
64100020	*	SET LOGIC POSITIVE
64206000	*	FORCE DELAY
64210000	*	FORCE WAIT
64300000	*	ENABLE/DISABLE RELAYS
64400000	*	ON DCT
64400001	*	ON FCT
64400002	*	ON TRIP
64400000	*	CLEAR DCT
64400001	*	CLEAR FCT
64400002	*	CLEAR TRIP
64500000	*	LIN
64600400	*	DISABLE TRIP
64700000	*	FORCE IF1
65000000	*	FORCE IF2
65100000	*	FORCE IF3
65200000	*	SET DCT
65300000	*	CLEAR FAIL
65400000	*	SET SA0
65420000	*	SET SA1
65500000	*	SET TEST #

COMPILER GENERATED OPCODES - ARITHMETIC

	OP CODE	S6	DESCRIPTION	
	70000000		ADD	
	70000001		SUB	
	70000002	*	MUL	
	70000003	*	DIV	
	70000004	*	EXP	
	70000005	*	AND	
	70000006	*	OR	
	70000007	*	EOR	
	70000010	*	NOT	
	70000011	*	NEG	
	70100002	*	LTN	
	70100004	*	EQ	
	70100006	*	LEQ	
	70100010	*	GTR	
	70100012	*	NEQ	
	70100014	*	GEQ	
	70200000	*	STR	Store
	70200001	*	CF	Call procedure with argu.
	70200002	*	PAE	Parameter array elements
	70200003		PEX	Parameter expression on STACK top
	70200004	*	STKC	Stack floating point constant
	70200005	*	STKCI	Stack integer constant
	70200006	*	INDR	Calculate index result
	70200007	*	INDA	Calculate index address
	70200010	*	PAUSE	PAUSE statement
	70200011	*	CIO	Complete IO statement (END OF READ/WRITE)
	70200012	*	FOR	FOR statement
	70200013	*	DO	DO statement
	70200014	*	EAS	end of DCL statement
	70200015	*	COL	Column format for WRITE
	70200016	*	LIT	Literal variable for READ/WRITE
*	70300000	*	DCLV	declare array
***	70400000	*	CPN	check parameter count
***	70500000	*	SIO	Start IO
*	70600000	*	INV	Initialize array
*	70700000	*	INS	Initialize scalar
*	71000000	*	STKV	Stack value
*	71100000	*	STKFV	Stack parameter value
*	71200000	*	STKAA	Stack array address
*	71300000	*	STKA	Stack variable address
*	71400000	*	STKFA	Stack parameter address
*	71500000	*	GOTO	GOTO statement

COMPILER GENERATED OPCODES - ARITHMETIC

	OP CODE	S6	DESCRIPTION	
	71600000	*	BE	block end
***	71700000	*	TEXT	Text for WRITE
*	72000000	*	BB	block begin
*	72100000	*	PB	program begin
*	72200000	*	PE	procedure end
*	72300000	*	PS	argument parameter is scalar
*	72400000	*	PA	argument parameter is array
*	72500000	*	PFP	argument parameter is parameter
*	72600000	*	PFE	FUNCT procedure end
*	72700000	*	EXPEX	EXEC parameter on STACK top
**	73000000	*	ONDIF	ON DIFEOF
	73100000	*	RDIF	RESET FDIF
***	73200000	*	INDX GOTO	indexed GOTO statement
**	74000000	*	IFJ	IF false, jump
**	75000000	*	JMP	unconditional jump
**	75100000	*	ONFJ	ON fail jump
**	76000000	*	TRAFZ	Transfer to subroutine (no parameter)
**	77000000	*	TRAF	Transfer to function, subroutine
***	77400000	*	EXEC	Transfer to ALLINK program
***	77600000	*	REXEC	execute Micro program
	77777776	*	PAGE	Disc access

*Carry block/variable number
**Carry OBJ address
***Number unique for core

APPENDIX L

VOLTAGE AND CURRENT RANGE DEFINITIONS

TABLE L-1. NORMAL FORCE AND MEASURE RANGES

MODULE	STATEMENT	PROGRAMMABLE VALUE/RESOLUTION				
		RANGE 0	RANGE 1	RANGE 2**	RANGE 3**	RANGE 4
PMU	Force Voltage		0 to 1.023V/1 mV	0 to ±10.23V/10mV	0 to ±40.92V/40mV	0 to ±102.3V/100mV
PMU	Set PMU ForceV		0 to 1.023V/1 mV	0 to ±10.23V/10mV	0 to ±40.92V/40mV	0 to ±102.3V/100mV
PMU	Force Current	0 to +1.023uA/1nA	0 to +102.3 uA/. 1uA	0 to ±10.23mA/10uA	0 to ±102.3mA/. 1mA	
PMU	Set PMU ForceI	0 to ±1.023uA/1nA	0 to ± 102.3 uA/. 1uA	0 to ±10.23mA/10uA	0 to ±102.3mA/. 1mA	
PMU	Set PMU Sense Voltage		0 to ±1.023V/1mV	0 to ±10.23V/10mV	0 to ±40.92V/40mV	0 to ±102.3V/100mV
PMU	Set PMU Sense Current	0 to ±1.023uA/1nA	0 to ±102.3 uA/. 1uA	0 to ± 10.23mA/10uA	0 to ±102.3mA/. 1mA	
DPS	Force VF			0 to +10.23V/10mV	0 to +40.92V/40mV	
DPS	Force IF			0 to ±102.3mA/. 1mA	0 to ±1.023A/1mA	
DPS	Enable Trip			0 to ±102.3mA/. 1mA	0 to ±1.023A/1mA	
DPS	Enable TripV			0 to ±10.23V/10mV	0 to ±40.92V/40mV	
RVS	Set (S0/S1)			+6.0V to -10.23V/10mV	+6.0V to -30.72/40mV*	
RVS	Force E			+6.0V to -10.23V/10mV	+6.0V to -30.72/40mV*	

TABLE L-2. FORCE AND MEASURE RANGES WITH 2V/2mV OPTION

MODULE	STATEMENT	PROGRAMMABLE VALUE/RESOLUTION				
		RANGE 0	RANGE 1	RANGE 2**	RANGE 3**	RANGE 4
PMU	Force Voltage		0 to 2.046V/2 mV	0 to ±10.23V/10mV	0 to ±40.92V/40mV	0 to ±102.3V/100mV
PMU	Set PMU ForceV		10 to 2.046V/2 mV	0 to ±10.23V/10mV	0 to ±40.92V/40mV	0 to ±102.3V/100mV
PMU	Force Current	0 to +1.023uA/1nA	0 to +102.3uA/. 1uA	0 to ±10.23mA/10uA	0 to ±102.3mA/. 1mA	
PMU	Set PMU ForceI	0 to ±1.023uA/1nA	0 to ± 102.3uA/.1uA	0 to ±10.23mA/10uA	0 to ±102.3mA/. 1mA	
PMU	Set PMU Sense Voltage		0 to ±2.046V/2mV	0 to ±10.23V/10mV	0 to ±40.92V/40mV	0 to ±102.3V/100mV
PMU	Set PMU Sense Current	0 to ±1.023uA/1nA	0 to ±102.3uA/. 1uA	0 to ± 10.23mA/10uA	0 to ±102.3mA/. 1mA	
DPS	Force VF		0 to ± 2.046V/2mV	0 to ±10.23V/10mV	0 to ±40.92V/40mV	
DPS	Force IF			0 to ±102.3mA/. 1mA	0 to ±1.023A/1mA	
DPS	Enable Trip			0 to ±102.3mA/. 1mA	0 to ±1.023A/1mA	
DPS	Enable TripV			0 to ±10.23V/10mV	0 to ±40.92V/40mV	
RVS	Set (S0/S1)		0 to 2.046V/2mV	+6.0V to -10.23V/10mV	+6.0V to -30.72V/40mV*	
RVS	Force E			+6.0V to -10.23V/10mV	+6.0V to 30.72V/40mV*	

NOTES

* Max limit is -30.72V for a 5 MHz Pin Electronics Board, -16.00V for a 10 MHz Pin Electronics Board.

** Current ranges depend on the hardware module: 102.3 milliamps is full scale in range 2 for power supplies, but is full scale in range 3 for the PMU.

UP DATED BY NEXT PAGE

APPENDIX M

STATEMENT LIST, REGISTERS WRITTEN, CODE TYPE, AND TIME DELAY

Statement and Options	Registers Written	DMA/ITNTPR	TD*
AT	MCS	D/I (503)	0
BRANCH UNLESS/TO/RESET	PG-MIL	D	
CGEN TGn	TG1,TG2,TG3	D	0
CONN CLK	PPA	D	2
CONN DPS1/DPS2/DPS3/TCOM	PPA	D	2
CPMU PIN	PA	D/I (612)	2
DISABLE TRIPS	SR	I (646)	0
ENABLE/DISABLE DCT0/DCT1	none	I (631,632)	0
ENABLE/DISABLE DOUBLE STROBE	SAMB	I (521)	0
ENABLE/DISABLE LATCHES	SR	I (640/641)	0
ENABLE/DISABLE RELAY	PA	D	2
ENABLE/DISABLE SPLIT/RTO/ MUX/IMASK	SAMD	D	
ENABLE DA/DB	F-RANK1		0
ENABLE ILO/IHI/VLO/VHI	none	I (632, 635)	0
ENABLE MA/MB	F-RANK 2		0
ENABLE PPM	SAMA	D	0
ENABLE TEST (no SET START)	S (W)	D	0
(no SET MAJOR)	L	D	
	MCS, SAMA	D	
	SAMA, IF, LRAX, IF2(AB)	I (521)	
EXTA	SAMB	I	
LOOP	SAMC	I	
MATCH	SR (if time out fail)	I	
AMATCH	SAMD, SR (if time out fail)		
ENABLE TRIPI1/TRIPI2/TRIPI3	SR, DPT1/2/3/, DPS1/2/3	I (601-603)	6
ENABLE TRIPV1/TRIPV2/TRIPV3	SR, DPT1/2/3, DPS1/2/3	I (613-615	6
FORCE CURRENT/VOLTAGE	PPS	D/I (605)	4
FORCE DELAY	TD (special)	I (642)	9
FORCE E0/E1/.../EC1	E0/E1/.../EC1	I (620-623)	7
FORCE IF1/IF2/IF3	SR, DPT1/2/3, DPS1/2/3	I (647-651)	6
FORCE PMU	PPS	D/I (605)	4
FORCE RESET	DPS1/2/3, DPT1/2/3,PPS,	I (640)	6

* See page M-4 for time delay generator.

STATEMENT LIST, REGISTERS WRITTEN, CODE TYPE, AND TIME DELAY (Cont'd.)

Statement and Options	Registers Written	DMA/ITNTPR	TD*
	RVS's, all registers on hardware reset line, MR, DA, DB, MA, MB, SR, SAMA, EIR, TD These registers are restored: bits 1, 2, 7, SAMB, all bits except 4 and 5 MR, bit 0 SR, bit 11 SAMC, SPM mode bit in SAMD, and IND, INC, EIR		
FORCE VF1/VF2/VF3	SR, DPT1/DPT2/DPT3, DPS1/DPS2/DPS3	I (625-627)	6
FORCE WAIT	None	I (642)	10
label:	IND	I (645)	0
LCGEN TG1/TG2/TG3	F	D	0
LSET IX/STROBE/RZ/INVERT/ DA/DB/MA/MB	F	D	0
LSUBR	F	D	0
MEASURE NODE	SAMA, PA, PSL, DCT, SR on failure	I (616)	
MEASURE PIN (no ENABLE DCT1/DCT0)	DCT, SAMA	D	8
(ENABLE DCT1/ DCT0)	SAMA, PSL if AUTO specified, SR on failure	I (616)	
MEASURE PIN # 1,2,6,10	PA, PPS, SAMA, DCT SR on failure	I (616)	
3,4	PA,SAMA,DCT,SR on failure		
5,7,9	PA,PPS,SAMA,DCT,SR on failure		
MEASURE VARIABLE	SAMA, SR on failure	I (616)	0
PGEN LOAD	PG-PCNTR	D	
RD/WR ONE/ZERO/CHECK/ NCHECK	PG-MIL	D	
SET APERIOD	SAMD,TR	I (507)	0
SET ATG4 DELAY	SAMD,PDV,PD,PWV,PW	I (523)	0
WIDTH	SAMD,PWV,PW		
SET CHAIN TWO/FOUR	SAMB,CH	I (521),D	0
OFF	SAMB	I (521)	
SET CLAMP	PSL (Called SPSL in Analysis)	D/I (624)	2
SET CRO	LRAX,CRO(AB)	D	
SET DA	DA	D	2
SET DB	DB	D	2
SET DCT	PSL,DCT	D/I (652)	8
SET DELAY	TD	I (611)	0
SET F	F	D	0

STATEMENT LIST, REGISTERS WRITTEN, CODE TYPE, AND TIME DELAY (Cont'd.)

Statement and Options		Registers Written	DMA/ITNTPR	TD*
SET FC		F	D	0
SET FI		SAMA,F	I (520)	0
SET IFAIL		IF,SAMC	I (502)	0
	COUNT	IF,LRAX,IF2 (AB),SAMC		
SET INVERT		INVERT (W)	D	0
SET IOM3	pin list	SAMD,SAMB,CH	I (501),D	0
	OFF	SAMD,SAMB	I (521)	
SET IOMODE	pin list	SAMB,CH	I (521),D	0
	OFF	SAMB	I (521)	
SET LOGIC		SR	I (640,641)	0
SET MA		MA	D	0
SET MAJOR	loop count	N	D/I (506)	0
	last address	L	D/I (504)	
SET MB		MB	D	0
SET MINOR	loop-count	M	D/I (506)	0
	start & end	J,K	D/I (505)	
SET PAGE		SAMB,SAMA,S,N,MCS, J,K.L,M,TR,F,DA,DB, MA,MB,IND,SAMD	I (501)	0
	SPM	SAMD		
SET PERIOD		TR	I (507)	0
SET PGEN1		PG-PS,PG-SIZE, PG-XSIZE	D	
SET PGENA/PGENC/PGENCN/ PGEND/PGENDN		PG-PS	D	
SET PMU	SENSE	PSL	I (617)	5
	FORCEV	PPS		4
	FORCEI	PSL		4
SET PPM		SAMD	I (524)	0
SET Q		LRAX, QL, Q	I (522)	0
SET R		R	D	3
SET RZ		RZ	D	0
SET S1/S0/SA1/SA0		S1/S0/SA1/SA0	I (630,654)	7
SET S		S	D	1
SET SI		SAMA,S	I (520)	0
SET START		S	D/I (502)	0
SET STROBE		ST	D	0
SET TG	DELAY	PDV,PD,PWV,PW	I (510-517)	0
	WIDTH	PWV,PW		
SET TEST #		PPS,PA	I (655)	
SET XOR		LRAX,XOR (AB)	D	
WRITE (register)		(register)	I (605)	0
XCON PIN		PPA	D	2
XCON VF1/VF2/VF3		PA	I (612)	6
XPMU PIN		PA	D	2

Time Delay Generated

TD	Description
0	no delay
1	0.28 millisecond
2	0.56 millisecond
3	1.75 millisecond
4	Programmed DC Time Delay or 0.56 millisecond with no current range change or 4 millisecond (±1 millisecond) with current range change, whichever is greater.
5	0.56 millisecond with no current range change or 4 millisecond with current range change.
6	Programmed DC Time Delay or 5.37 milliseconds, whichever is greater.
7	Approximately 300 microseconds per volt of change or 0.56 millisecond, whichever is greater.
8	56 microseconds
9	Programmed DC Time Delay
10	The time required for the tester to become not busy.

GLOSSARY OF TERMS AND ACRONYMS

Array	An ordered arrangement or pattern of values which are grouped together positionally with respect to some variable identifier.
Block	A group of program statements between the BLOCK and END statements compiled independently, or a subroutine or function.
Chaining	Expansion of local memory pattern depth by "chaining" or interleaving data from selected local memory channels into adjacent channels. This allows certain pins with a 1K local memory to have effective pattern depths of 2K or 4K. Likewise, certain pins with a 2K local memory can have a pattern depth of 4K.
CPU	Central processing unit
CR	Card reader
DIF	Disc input file
DOF	Disc output file
DMA	Direct memory access
DOPSY	Disc operating system
DUT	Device under test
Expression	A grouping of one or more numbers, variables, and functions combined with arithmetic or Boolean operators and parenthesis so as to represent a quantity or an operation.
FACTOR	Fairchild Algorithmic Compiler Tester Oriented.
FDIF	FACTOR disc input file.
FDOF	FACTOR disc output file.

Function	Parametered calls used to obtain values through a standardized set of operations.
Integer	A whole number including zero.
Label (local memory type)	A symbolic identifier (terminated by an @ symbol), preceeding a "SET F binary pattern" statement, for the local memory location the binary pattern is loaded into. This allows symbolic references to local memory addresses in subsequent FACTOR statements.
Label (non-local memory type)	A symbolic identifier (terminated by a colon) for a FACTOR statement which allows that statement to be referenced in a branching type statement symbolically, rather than by statement number (which is unknown at compilation time).
Load	A local memory load is defined as a sequence of SET F statements interspersed only with ENABLE MA/MB DA/DB statements. A local memory load always commences at location 0 in local memory, unless specifically altered by an AT statement. The load is terminated by some instruction other than the SET F and the ENABLE MA/MB DA/DB. Subsequent SET F statements cause the compiler to assume that a new load is beginning.
Location count	The compiler keeps a location counter during the sequence of instructions forming the load. When the load is terminated the location count is the test end address for the ENABLE TEST, unless a different end address is specified by a SET MAJOR statement.
Long register	Binary registers used to set up local memory and other hardware conditions for functional testing. The long registers are: DA, DB, MA, MB, S, R, F, INVERT, I, STROBE, RZ, and XOR
LP	Line printer
Major loop count (N)	The number of times the major loop is to be executed as defined in the SET MAJOR statement. The loop count equals one if no SET MAJOR statement is entered.

Major loop end address (L) or test end address	The last memory location to be executed unless a fail occurs first, interrupting testing. This is defined by the last local memory word loaded in the previous load, or the one defined in the SET MAJOR statement.
Minor loop count (M)	The number of times the minor loop is to be executed. Note that in the ENABLE TEST CONTINUOUS mode the minor loop continues to execute even when the loop count is one.
Minor loop end address (K)	The last address in the minor loop before execution controls returns to the minor loop start address.
Minor loop start address (J)	The local memory address for the start of the minor loop.
MTR	Magnetic tape read.
MTW	Magnetic tape write.
PID	Primary input device
PMU	Precision measuring unit
POD	Primary output device
Record	An arbitrary amount of data read from or written into an I/O device. For a TTY or VKT a record is the amount of data from one carriage return until the next. For a line printer a record is one line.
Scalar value	A signed numeric quantity having magnitude but no direction, nonarrayed, non Boolean quantity.
System Global	A variable that is retained from one execution of a program to the next.
Test start address (S)	The local memory address at which testing begins. This is always location 0 unless altered by a SET START statement.
TOPSY	Tester operating system.
TRASCII	Truncated ASCII, six bit code instead of ASCII 8 bit code.

Variable	Any quantity which is referred to by a name rather than by an explicit value. Can be a single variable or an arrayed (one dimensional) variable.
WS	Working Storage (area on disc not currently storing files).

INDEX

MANUAL REGISTRATION FORM
AND COMMENT SHEET

FILL OUT AND MAIL TO RECEIVE UPDATES AND SUPPLEMENTS AUTOMATICALLY AS THEY ARE PRINTED.

FROM:

NAME _____

TITLE _____

BUSINESS ADDRESS _____

_____

_____

BUSINESS PHONE _____

MANUAL PART NO. 67095738

REVISION NO. _____

DATE OF PUBLICATION March 1977

COMMENTS:

(Please describe errors, suggested additions or deletions, and reference part number, page number, paragraph number, or drawing number.

ERRORS, OR CHANGE SUGGESTED, ON PAGE(S)

_____

_____

_____

_____

_____

_____

_____

_____

_____

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE STAPLE

Fold Along Dotted Line

FIRST CLASS
PERMIT NO. 5699
San Jose,
California

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

Postage Will Be Paid By

FAIRCHILD SYSTEMS TECHNOLOGY
1725 TECHNOLOGY DRIVE
SAN JOSE, CA 95110

ATTN: SYSTEMS TECHNICAL PUBLICATION DEPT.

Fold Along Dotted Line

STAPLE STAPLE