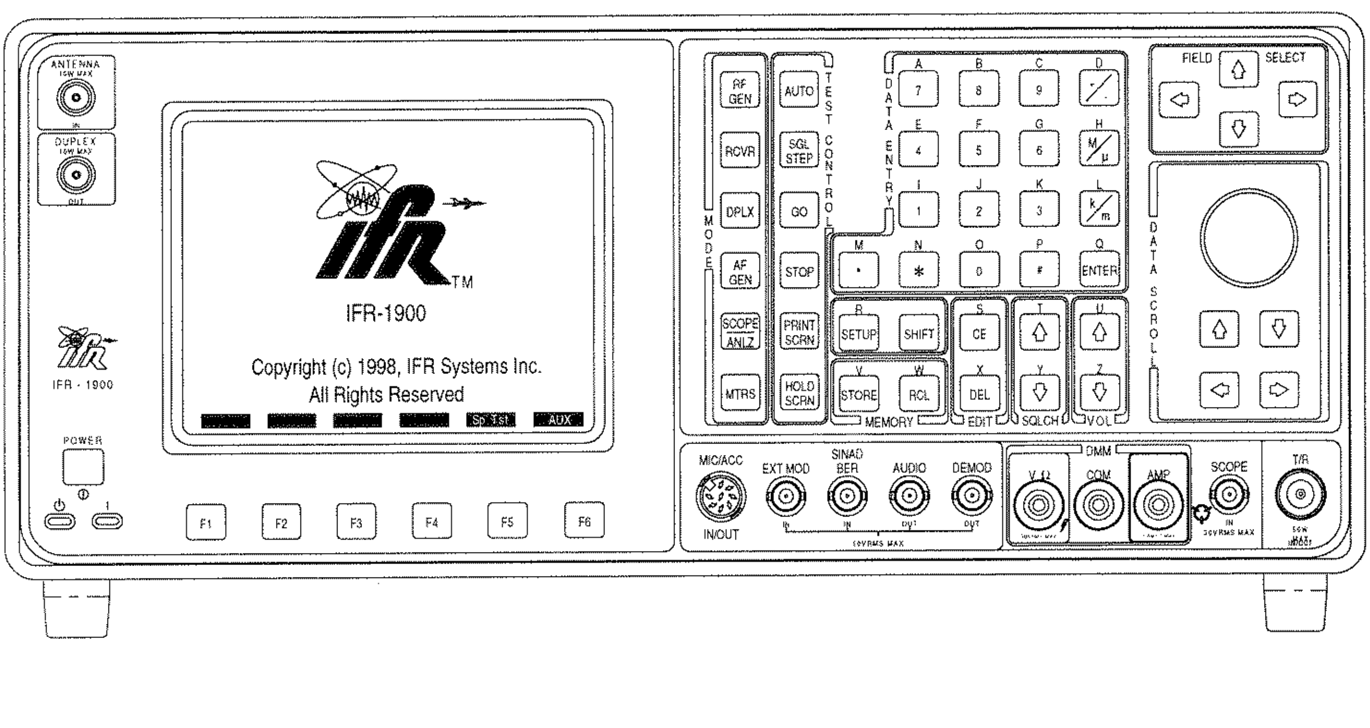
TMAC USERS MANUAL

VOLUME I
HOST
(Includes General TMAC)

SYSTEMS, INC.

IFR-1900 CSA

COMMUNICATIONS SERVICE MONITOR
DUAL MODE / TRI-BAND CELLULAR SYSTEM ANALYZER



10200 West York Street / Wichita, Kansas 67215 U.S.A. / (316) 522-4981 / FAX 316-524-2623

1002-3400-600
HOST Ver - 1.03 and on
Sp Tst Ver - 1.04 and on

PUBLISHED BY
IFR SYSTEMS INC.
Wichita, Kansas

LIST OF EFFECTIVE PAGES

The manual pages listed below that are affected by a current change or revision, are so identified by a revision number.

Date of Issue for original and changed pages are:

Original .... 0 .... January 1998

TOTAL NUMBER OF PAGES IN THIS MANUAL IS 1242 CONSISTING OF THE FOLLOWING

Pg. No.	Rev. No.	Pg. No.	Rev. No.
VOLUME I		**VOLUME II**	
Title	0	Title	0
Copyright	0	Copyright	0
A	0	i through ix	0
B Blank	0	x Blank	0
i through xviii	0	7-1	0
1-1	0	7-2 Blank	0
1-2 Blank	0	8-1 through 8-73	0
2-1 through 2-34	0	8-74 Blank	0
3-1 through 3-105	0	9-1 through 9-405	0
3-106 Blank	0	9-406 Blank	0
4-1 through 4-14	0	10-1 through 10-52	0
5-1 through 5-48	0	11-1 through 11-77	0
6-1 through 6-173	0	11-78 Blank	0
6-174 Blank	0	12-1 through 12-163	0
Index-1 through Index-18	0	12-164 Blank	0
		A-1	0
		A-2 Blank	0
		B-1	0
		B-2 Blank	0
		C-1 through C-7	0
		C-8 Blank	0
		D-1	0
		D-2 Blank	0
		E-1 through E-3	0
		E-4 Blank	0
		Index-1 through Index-18	0

PREFACE

SCOPE

This Two Volume Manual contains instructions for remotely operating the IFR-1900 CSA Communication Service Monitor and Tri-Band/Dual Mode Cellular System Analyzer.
The instruction level is relatively basic and presupposes no previous experience on the part of the operator with remote operation of a communication service monitor or cellular system analyzer. A basic understanding of communication electronics and cellular system formats is helpful. It is strongly recommended that the operator be thoroughly familiar with this manual as well as the Operation Manuals of the Test Set before attempting to operate the Test Set, remotely.

ORGANIZATION

The IFR-1900 CSA TMAC Users Manual is composed of the following volumes and sections:

VOLUME I - IFR-1900 CSA GENERAL AND HOST SPECIFIC TMAC

SECTION 1 - INTRODUCTION

Provides an introduction to the TMAC language.

SECTION 2 - TMAC (TEST MACRO LANGUAGE) OVERVIEW

Describes and explains the various features of the TMAC language, focusing primarily on General TMAC.

SECTION 3 - GENERAL TMAC COMMANDS

Lists all General TMAC Commands alphabetically and provides a detailed definition of each command.

SECTION 4 - CREATING AND LOADING TMAC PROGRAMS

Provides step-by-step procedures and examples for creating and loading TMAC programs.

SECTION 5 - HOST SPECIFIC TMAC QUICK REFERENCE LIST

Briefly lists the IFR-1900 CSA HOST TMAC commands in alphabetical order.

SECTION 6 - HOST SPECIFIC TMAC COMMANDS

Lists and details the Specific TMAC commands for the IFR-1900 CSA HOST. Commands are arranged by Operation Mode for convenience.

VOLUME II - IFR-1900 CSA SPECIAL TEST SPECIFIC TMAC

SECTION 7 - INTRODUCTION TO VOLUME II

Provides an organization of Volume II and definition of the configuration of the Test Set configuration.

SECTION 8 - SPECIAL TEST SPECIFIC TMAC QUICK REFERENCE LIST

Briefly lists the IFR-1900 CSA Special Test Specific TMAC commands in alphabetical order.

SECTION 9 - SPECIAL TEST SPECIFIC TMAC COMMANDS

Lists and details the Specific TMAC commands for the IFR-1900 CSA Special Test. Commands are arranged by Operation Mode for convenience.

SECTION 10 - SPECIAL TEST PROGRAM EXAMPLES

Provides functional Special Test program examples.

SECTION 11 - IS-136 COMMAND REFERENCE

Provides tables showing the relationship between IS-136 Layer 3 Messages and associated IFR-1900 CSA Special Test TMAC commands.

SECTION 12 - SPECIAL TEST KEY WORD INDEX

Provides a permuted index of all of the Special Test commands in the IFR-1900 CSA TMAC Users Manual. **Bold** words in the center column are the particular key words being indexed. Each full command is indexed by each word in the command.

NOMENCLATURE

The IFR-1900 CSA Test Set consists of the following:

FUNCTION	NAME
Communication Service Monitor	HOST
Tri-Band/Dual Mode Cellular System Analyzer	SPECIAL TEST or Sp Tst

The Special Test (Tri-Band/Dual Mode Cellular System Analyzer) utilizes the test equipment contained in the Communication Service Monitor portion of the IFR-1900 CSA, thus the Communication Service Monitor acts as HOST to the Special Test.

For remote communications and uploading of variables and TMAC programs via RS-232, two separate Rear Panel RS-232 Connectors are utilized on the IFR-1900 CSA. The HOST utilizes the RS-232 Connector labeled **HOST**, and the Sp Tst employs the connector labeled **OPT**.

TABLE OF CONTENTS

SECTION 1 - INTRODUCTION

SECTION 2 -TMAC (TEST MACRO LANGUAGE) OVERVIEW

SECTION 3 - GENERAL TMAC COMMANDS

SECTION 4 - CREATING AND UPLOADING TMAC PROGRAMS

SECTION 5- HOST SPECIFIC TMAC QUICK REFERENCE LIST

SECTION 6 - HOST SPECIFIC TMAC COMMANDS

SECTION 7 - INTRODUCTION TO VOLUME II

SECTION 8 - SPECIAL TEST SPECIFIC TMAC QUICK REFERENCE LIST

SECTION 9 - SPECIAL TEST SPECIFIC TMAC COMMANDS

SECTION 10 - SPECIAL TEST PROGRAM EXAMPLES

APPENDICES

INDEX

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1 - INTRODUCTION

TMAC (Test Macro Language) was developed by IFR Systems Inc. to operate IFR test equipment in a remote configuration and allow internal versatile software to be created. TMAC is written so not just single measurements but entire batteries of tests may be performed. Combining this capability with the test functions available with the IFR-1900 CSA Test Set gives the user a great deal of flexibility in testing communication devices in an ATE environment. Additionally this language also provides a format to store and perform user defined test sequences. TMAC is based on the IEEE-488.2 format and conforms to the SCPI Standard. Some commands and operators, created before and not conforming to the SCPI Standard, are so noted.

TMAC is written on several layers of complexity. The first layer consists of the Machine Specific TMAC Commands. These commands are what actually set up and control the Test Set and take the measurements. The Specific TMAC commands for the IFR-1900 CSA encompass the complete operation of the Test Set. Every function of the IFR-1900 CSA can be performed remotely with the exception of power up of the Test Set.

The second level of TMAC is the framework that is used to combine the machine specific commands into a coherent and fluid process. Decision making commands allow different procedures to be performed under differing circumstances. These decisions may be based on complex expressions as well as simple boolean answers being returned from the Test Set. Looping commands allow the continual monitoring of a parameter while performing tasks. Math and logic function commands together with bit manipulating commands allow the use and evaluation of complex functions. TMAC also provides a variety of data structures to assist in developing test procedures. Graphic commands allow the development of user-defined screens and menus to aid the user in performing test sequences. The Status System provided allows the monitoring of the Status Registers called for in the IEEE-488.2 Standard.

The third layer and one of the main strengths of TMAC is the capability to define macros with procedures built of machine specific commands as organized by the framework commands. Macros allow step by step test procedures to be developed, stored and locally or remotely implemented. Using macros is easy. Entering the name of the macro along with any variable parameters, executes the macro. Macros can be executed within other macros, allowing complicated procedures to be divided into smaller tasks. Macros can be initiated remotely, from the Front Panel Keyboard, or automatically upon power up (see Program Commands in Section 6) of the Test Set.

The final layer of TMAC is the ability to Multitask Macros. The Multitasking feature of TMAC allows several macros to alternate command execution at once giving the appearance of being executed simultaneously. Multitasking in TMAC allows the user to activate a task, pass to the next task, put a task to sleep, wake a sleeping task and stop a task.

SECTION 2 - TMAC (TEST MACRO LANGUAGE) OVERVIEW

2-1 SYNTAX

When working with TMAC, punctuation marks are used to establish a format of how commands and parameters are entered. The syntax needed to use TMAC is detailed below.

- Command Forms

Many commands can be entered using a short form or a long form. In this manual, the short form is shown in upper case, while the remainder of the long form is shown in lower case. Upper and lower case letters are used only in this manual to differentiate between the long and short form of commands. TMAC executes any valid command whether in upper and lower case letters or a combination of upper and lower case letters.

Given the command, **STATus:QUEStionable:CONDition?**, entries can be any of the following forms:

Example:
```
status:questionable:condition?  or
STATUS:QUESTIONABLE:CONDITION?  or
STAT:QUES:COND?  or
stat:ques:condition?  or
STATUS:quesTIONABLE:cond?
```

The following forms of the command are **not** allowed:

Example:
```
STATU?:QUESTION:CON?  or
status:questionabl:condit?  or
status:QUESTIONABLE:conditio?
```

Each word in each command may have two acceptable forms (full or abbreviated). Entering a command using other than the two acceptable forms causes a syntax error.

- Colon (:)

TMAC commands are arranged in hierarchical levels using the colon to separate the different levels. A colon before the beginning of a command (**:FGEN**) signifies the command starts at the first hierarchical level.

Example: `FGEN:GEN1:SHAPE:SIN`

FGEN precedes all Function Generator commands. The **FGEN:GEN1** precedes all Function Generator commands dealing with Generator 1. **FGEN:GEN1:SHAPE** precedes all Function Generator commands selecting the waveform of Generator 1. **FGEN:GEN1:SHAPE:SIN** selects a sine wave for the waveform of Generator 1 of the Function Generator.

A new line character marks the end of a command and places the next command at the first command level.

- Semicolon (;)

A semicolon signifies the end of a command and starts the next command at the last level of the previous command.

Example: `FGEN:GEN1:FREQ 2000;SHAPE:SIN;:SCREEN:FUNC`

The first command, **FGEN:GEN1:FREQ 2000**, leaves the TMAC Compiler at the **FGEN:GEN1** level. Preceded by a semicolon, the next command, **SHAPE:SIN**, starts from this level. The next semicolon leaves the TMAC Compiler at the **FGEN:GEN1:SHAPE** level. The following colon resets the Compiler to the first hierarchical level to start the next command, **SCREEN:FUNC**.

Example: `DUP:INP:FREQ?;MOD:USER:MOD:DATA;FILTER 30;:SCREEN:DUP`

This command line performs the following commands:

```
DUP:INP:FREQ?
DUP:INP:MOD:USER:MOD:DATA
DUP:INP:MOD:USER:FILTER 30
SCREEN:DUP
```

- Whitespace

At least one space is needed between commands and parameters. Other spaces or lack of them between items do not affect command execution. Spaces inside command words or between command word and ending punctuation (**ME AS:FREQ?** or **MEAS:FREQ ?**) are not allowed.

- Comma (,)

Commas are used between parameters when more than one is listed and to continue printing on same line.

Example: `MACRO 1,2,3,4`

```
PRINT "This gets printed ",      // Comma causes next printing to
PRINT "on one line."             // begin at end of line, not the
                                 // next line down.
```

- Back slash (\)

A back slash allows a command to be continued on the next line. The back slash cannot break a command word.

Example:

```
FGEN:GEN1:\
FREQ 2000
```

This is the same command as **FGEN:GEN1:FREQ 2000.**

Example:

```
FGEN:GE\
N1:FREQ 2000
```

This command fails to execute and produces an error message because command word **GEN1** is broken.

2-2 COMMENTS

Comments provide clarity in macros and are ignored by the TMAC compiler/interpreter. There are two types of comments.

Characters // cause the rest of line to be a comment.

Example: `PRINT COS(freq*SQR(2))    // PRINTS COS OF FREQ × SQUARE ROOT OF 2`

The message `PRINTS COS OF FREQ × SQUARE ROOT OF 2` is not executed by the TMAC compiler/interpreter.

Characters /* and */ start and end the second type of comment. It can contain several lines.

Example:
```
PRINT COS(freq*SQR(2))    /* prints cosine of the quantity RF Generator
     frequency multiplied by the square root of 2 */
```

The TMAC compiler/interpreter ignores everything between the comment characters.

2-3 NUMERIC VARIABLES AND ARRAYS

2-3-1 GENERAL

Variables and arrays are used to hold numeric values and can be local or global. If declared inside a macro, a variable or array is local and is used inside that macro only. If declared outside a macro, a variable or array is global and is used outside or inside any macro (unless a local variable has the same name as the global variable). Locality takes precedence.

The number of variables and arrays allowed is limited by the amount of memory available. A maximum of 998 memory locations are available for global variables. A global variable takes one location, global arrays take a location for each array element and global string variables take 9 locations each. Local variables are also limited by memory space which varies depending on many factors including call nesting (see Macros in 2-9). One to two hundred local variables are usually available with arrays taking the space of a variable for each array element and string variables taking nine times the space needed for a variable.

The first character of a variable or array name must be a letter while the remaining characters can be letters, digits 0 through 9 and the underscore. The maximum length is 31 characters.

Variables A through Z are pre-declared and global and cannot be declared as a string variable. Variables A through Z are also referred to as Free variables.

To declare a variable, use the **VAR** command:

Example: `VAR Freq,Rad,Mod`

Freq, *Rad* and *Mod* are now declared variables.

Variables can be initialized when they are declared by including an equal sign and the value. The value of numeric variables are equal to 0 when declared unless initialized by the user.

Example: `VAR Rad = 4.32876, Mod`

Rad is declared and assigned a value of 4.32876. Mod is declared and contains a value of 0.

Arrays are also declared using the VAR command. The highest index is included in brackets. The index count starts with 0, making *index* one less than the total number of variables in the array.

Example: `VAR BB[9]`

This command declares array BB with 10 variables: BB[0] through BB[9].

Braces are used to initialize arrays. More than one can be initialized at a time.

Examples:
```
VAR BUNCH[10] = {1,2,3,4,5,6,7,8,9,10,11}
VAR AA[2]={1,2,3}, BB[1]={7,4}, CC[3]={24,345.754,2,100}
```

A variable and an array cannot have the same name at the same time. Accessing an array without specifying an index yields erroneous results.

2-3-2 SAVING VARIABLES AND ARRAYS (HOST ONLY)

Refer to Section 9-2 to run HOST commands when operating as the Sp Tst.

Global variable and array values are lost when the Test Set power is turned off. To save variable or array values in non-volatile memory, use the **NVSAV** command. 283 non-volatile memory locations are available. To recall a variable, use the **NVRCL** command:

Each variable uses one memory location and arrays use a memory location for each value. Free variables, local variables and local arrays cannot be saved in non-volatile memory.

Example:
```
VAR DD
NVSAV DD,10
```

DD is now saved in non-volatile memory location 10. To recall DD, use the NVRCL command.

Example: `NVRCL DD,10`

The content of memory location 10 is loaded into variable DD.

When an array is saved using the **NVSAV** command, the whole array is saved in consecutive memory locations.

Example:
```
VAR list[5]
NVSAV list,1
```

The values of list are saved in memory locations 1 through 6 (6 values starting at 1). Arrays are recalled the same as variables.

Example: `NVRCL list,1`

The contents of memory locations 1 through 6 are loaded into list[0] through list[5].

Saving to previously used memory locations overwrites previous entries.

Example:
```
VAR List[4]={2,4,6,8,10} // Declares an array named List.
VAR Group[2]={23,46,69}  // Declares an array named Group.
NVSAV List, 10           // Saves List in memory locations
                         // 10 through 14.
NVSAV Group,12           // Saves Group in memory locations
                         // 12 through 14.
NVRCL List,10            // Recalls memory locations 10 through
                         // 14 and assigns the values to List.
```

Three of the saved values of List were overwritten when the same memory locations were used to save the array Group.

2-4 CONSTANTS AND DATA ARRAYS

Constants are declared using the **CONST** command. Constants are global only and cannot be declared inside a macro. Constants can only be assigned numeric data.

Example:
```
CONST PI, 3.1415, FREQ, 3000
```

An expression can be assigned to a constant. Expressions are evaluated prior to being assigned.

Example:
```
CONST AA,(4+6)/2     // Assigns 5 to AA.
```

After being declared, a constant can be changed only with another **CONST** command.

Example:
```
CONST AA,36
AA=20
/* The second command results in an error message and is not
performed, leaving AA equal to 36.          */
```

Constant data arrays are declared using the **DATA** command.

Example:
```
DATA offsets_1 = {2,3,4,5,6,7,8,9}
```

Like a constant, data arrays can only be assigned numeric data and an expression can be assigned to a constant. Expressions are evaluated prior to being assigned. After being declared, a data array can be changed only by another **DATA** command. Contents of data arrays are accessed using the index enclosed in brackets, where the first element of the data array has an index value of zero.

Example:
```
DATA Points = {11.6, 17.3, 22.4, 27.1}
Y = Points[2]
```

Variable Y is set equal to 22.4.

The **IDATA** command is the same as **DATA** except uses integer values only.

2-5 NUMERIC NOTATION AND FORMATS

2-5-1 NUMERICAL NOTATION

Numerals are expressed as fixed point decimal, scientific notation or as an integer. Numerals are specified as negative with a – character as shown:

Example: `-51`

This denotes a negative 51.

The maximum fixed point decimal numeral that can be entered is ±2147483647. Scientific notation uses the letter e to denote an exponent of 10.

Example: `54e7`

This number signifies 54 multiplied by 10 to the 7th power or 540,000,000.

100 is the maximum power of 10 that can be entered, although the maximum number in scientific form allowed is ±1.797693e308. Numbers with a larger power of 10 than 100 are assigned to a variable by multiplying powers of 10.

Example: `Large = 54e7*1e100*1e100`

This example assigns 54e207 or 54 multiplied by 10 to the 207th power to variable Large.

For execution reasons, the Sp Tst uses single precision floating point numbers (e.g., 1e±38).

2-5-2 NUMERIC FORMATS

To enter data in a different format (base 2, 8 or 16), precede data with one of the following characters:

#B	Binary
#Q	Octal
#H	Hexadecimal

Data format for the returned data for queries is changed to base 2, 8, 10 or 16 using the **FORMat** command. The default format is base 10.

Binary, octal or hexadecimal data received from the Test Set (HOST only) is preceded by #B, #Q or #H notation. Data in scientific notation is unaffected by any of the **FORMat** commands.

2-6 MATHEMATICAL OPERATORS

TMAC provides a variety of mathematical operators and bit manipulating functions to perform mathematical operations. A 32 bit word is used for binary and bitwise operations.

2-6-1 OPERATORS

Table 2-1 lists the mathematical operators used in TMAC.

=	Assignment operator
**	Exponential operator (does not conform to SCPI Standard)
*	Multiplication operator
/	Division operator
+	Addition operator and positive unary operator
-	Subtraction operator and negative unary operator
%	Modulo operator (does not conform to SCPI Standard)
++	Increment operator
--	Decrement operator
~	Bitwise complement
!	Logical negation (NOT) unary operator
¦	Bitwise OR
&	Bitwise AND
^	Bitwise XOR (conflicts with SCPI Standard for Exponential operator)
<<	Shift left operator
>>	Shift right operator

Table 2-1 Mathematical Operators

2-6-2 ORDER OF CALCULATION

When more than one operator is contained in an expression, the operators are calculated in the following order:

- Contents of parenthesis, calculating from inner sets to outer sets of parenthesis.
- Positive (+), negative (-), bitwise complement (~) and logical negation (!).
- Exponentiation (**).
- Multiplication (*), division (/) and modulus (%).
- Addition and subtraction.
- Shift left (<<) and right (>>) operations.
- Bitwise AND (&).
- Bitwise XOR (^).
- Bitwise OR (¦).
- Logical **AND**.
- Logical **OR**.
- Condition? *true*:*false*. (Refer to **IF ELSE** (Shorthand) command in section 3.)

2-7 MATHEMATICAL FUNCTIONS

TMAC provides the following mathematical functions to assist in mathematical calculations:

TRUE,ON	**FALSE,OFF**	**RND**	**RAND**
LOG	**LN**	**ABS**	**FLOOR**
SQR	**EXP**	**SIN**	**COS**
SIGN			

See Section 3 for a description of each command.

2-8 STRING VARIABLES AND FUNCTIONS

A string is a segment of text consisting of ASCII characters terminated by a null character. String variables are declared using the **STRING** command. There is a free string variable, represented by the $ character that is pre-declared and global. ($ cannot be declared a numeric variable.) String variables are either local (declared inside a macro) and are used only inside that macro or are global (declared outside a macro) and are used outside or inside any macro, depending on scope. 128 bytes (1 byte per ASCII character) of system memory is allocated for each string variable declared.

A literal string (actual ASCII text segment included in macro) is enclosed in quotation marks or apostrophes. To signify a quotation mark inside a literal string enclosed with quotation marks, a pair of quotation marks are used. When quotation marks are used to signify a literal string, apostrophes are treated as normal characters. To signify an apostrophe inside a literal string enclosed in apostrophes, a pair of apostrophes are used. When apostrophes are used to signify a literal string, quotation marks are treated as normal characters.

String variables cannot be initialized when declared. After being declared literal strings and/or other declared string variables can be assigned to a string variable.

Example:
```
STRING WARNING,  Note1,  Message_no_2
Note1 =  "This is a string variable"
```

Strings and string variables are combined (concatenated) by using + character as follows:

Example:
```
STRING Note_1, Note_2, Note_3
Note_1 =  "A STRING"
Note_2 =  "COMBINING"
Note_3 =  Note_1 + " " + Note_2 + " EXAMPLE"
```

Note_3 now equals A STRING COMBINING EXAMPLE.

Segments are extracted from strings using brackets in the following manner:

name1 = *name2*[*start*][*end*]

name1 is set equal to the segment in *name2* that begins with the *start* number and ends with the *end* number. Count of string *name2* begins with 0. String *name2* is not affected unless *name1* and *name2* are the same name.

Example:
```
STRING Test
$ = "Test1"
Test = $[4][4]
```

Test now equals 1, the fourth (count begins with 0) character of string $.

Example:
```
STRING MEMO1
MEMO1 = "message example "
MEMO1 = MEMO1[10][15]+MEMO1[0][3]
```

MEMO1 now contains the following string:

ample mess

Word of caution: when accessing data in a string, invalid data beyond the terminating null character is accessible, because no bounds checking is performed.

String Arrays are also declared using the **STRING** command by including the highest index in brackets.

The index count starts with 0, making index one less than the actual number of variables in the array. Like string variables, string arrays cannot be initialized when declared. Only one string array element is assigned a value at a time.

Example: `STRING MESSAGE[5]`

This command declares a string array named message containing six elements.

Example:
```
MESSAGE[0] = "see"
MESSAGE[1] = "spot"
MESSAGE[2] = "run"
```

This example assigns values to the first three elements of string array MESSAGE.

128 bytes (1 byte per ASCII character) of system memory is allocated for each element of a string array declared. (Sp Tst only: A maximum length for each element of the string array may be specified when the string array is declared. See **STRING** command in Section 3 for details.)

Substringing on string arrays allows segment selection from one or more elements in a string array. Adding a *start* and *end* number selects the segment in the *element* of the declared and initialized string array (*arrayname*). Segments are selected using the following format:

arrayname[*element*][*start*][*end*]

Example: `$ = MESSAGE[0][1][2]`

This example assigns the string "ee" to $. ee is the segment in the first element [0], starting at character [1] and going through character [2].

The following string functions allow manipulations of strings:

CHR	**TAB**	**ASC**	**LEN**
STR	**VAL**	**PIXLEN**	**STRPOS**

See Section 3 for a description of each command.

2-9 MACROS

Macros are groups of commands arranged to accomplish one or more tasks and may be considered programs; however, several macros may be grouped together (along with constants and global variables) into a single program. Macros are written and transferred to the Test Set by a Host System (PC with terminal emulating software). Once the macro is transferred to the Test Set, the macro is stored in memory (non-volatile for HOST) until deleted by a ***PMC** or **FORGET** command or power is removed (Sp Tst only). Once in memory, macros can be transferred to Flash Memory (see Section 4).

Macros are defined using the ***DMC** command (complying with IEEE-488.2) and have the form:

*DMC "name", command,... , command

Example: `*DMC "Square",Y=X**2;PRINT "Y holds X squared"`

Once loaded into memory, a macro is executed by entering the macro name. If the above macro is loaded, entering Square executes the macro Square. Macros can be called out and executed from other macros. However, a macro can only be transferred into memory if all macros called out are defined macros, unless the name of any undefined macro is in an **INTERP** command.

Example:
```
*DMC "Test_1",BEGIN
Average
END
```

Example:
```
*DMC "Test_2",BEGIN
INTERP "Total"
END
```

The macro Test_1 can be transferred into memory only if the macro Average is already defined. Macro Test_2 can be loaded into memory because the string Total is evaluated after execution of the **INTERP** command. The macro Total has to be defined prior to Test_2 execution, otherwise an execution error occurs.

Call nesting is the extended process of calling out other macros that call out other macros that call out other macros, etc. Call nesting uses a limited stack to contain all the internal variables and arrays for each macro called out. Reaching the stack limit initiates a memory error.

With TMAC, as opposed to IEEE-488.2, macros are not limited to one line. Using the **BEGIN** and **END** commands lets the macro contain multiple lines, allowing large involved macros, while also providing clarity lacking with one line macros.

Example:
```
*DMC "Square_x",BEGIN
     Y=X**2
END
```

A macro is deleted from memory using the **FORGET** command. All macros and variables declared after the deleted macro was declared, are also deleted (because they may contain references to the deleted macro). The ***PMC** command deletes all macros and declared variables in memory regardless of location. Predefined macros, listed in Appendix A, are permanently in memory and are unaffected by the ***PMC** or **FORGET** commands. The ***LMC?** query (HOST only) returns a list of the macros and defined variables contained in memory (e.g., `PRINT *LMC?`).

The address of a macro (location of macro in memory) is assigned to a variable by using an ampersand in the following manner:

```
X=&name
```

The **EXEC** command executes the macro at the address following the command.

Example:
```
ADD = &Square_x
EXEC ADD
```

The first command loads the address of Square_x into ADD. The **EXEC** command then executes Square_x.

The **EXEC** command can be used to call out undefined macros by simply declaring a variable and using the variable as the address.

The ***WAI** command pauses command execution until all previous operations are complete. ***WAI** is used following **SCREEN** commands and commands involving routing changes or long execution times.

The amount of available memory is the only limit to the number of macros that can be loaded. The **ROOM** command provides the amount of currently available memory space in bytes.

Example: `PRINT ROOM`

Prints the number of available bytes left in memory on the color display (HOST) or on the RS-232 terminal (Sp Tst).

Refer to 6-13 and 6-14 for HOST specific **PROGram** and **MMEM** commands used in executing HOST specific macros from the Front Panel.

Refer to Section 4 for loading Sp Tst macros into memory for Front Panel execution.

2-10 MACRO DECISION POINTS

TMAC allows decision points to be placed inside a macro to allow different courses of action. For instance, if a condition is true, then one course of action is taken. If the condition is false, then another course of action is taken. See **CASE**, **IF**, **IF ELSE** and **IF ELIF** (see Section 3). Generally, a condition has one of the following relational operators:

=	Equal	!=	Not equal
<	Less than	>	Greater than
<=	Less than or equal	>=	Greater than or equal

Example:

```
IF x=6
   PRINT y
ENDIF
```

Condition x equals 6 must be true for the next command (PRINT y) to execute; otherwise, command execution passes to the command following the **IF ENDIF** command, ignoring the print command.

Example:

```
IF x!=z
   PRINT y
ENDIF
```

Condition x != z (x not equal to z) must be true for commands contained in the **IF ENDIF** command to execute.

Conditions may also contain logical operators **AND** and **OR**. These operators connect two conditions. An expression with an **OR** is true if one or both conditions are true. An expression with an **AND** is true only if both conditions connected by the operator are true. Many logical operators are possible in a condition.

Example:
```
IF x=6 OR x!=z
   PRINT y
ENDIF
```

The above condition is true if either x is equal to 6 or if x is not equal to z or if both are true.

Example:
```
IF x=6 AND x!=z
   PRINT y
ENDIF
```

This condition is true if both x is equal to 6 and x is not equal to z.

Example:
```
IF W=3 OR U< 8 AND V=4
   PRINT y
ENDIF
```

This condition is true if either W is equal to 3 or both U is less than 8 and V is equal to 4.

Mathematical operators can be used in conditions.

Example:
```
IF W> 2*X**3
```

The quantity 2*X**3 is calculated before compared to W.

Example:
```
IF T = U&++V
```

In this example, V is incremented and bitwise AND is performed before comparison with T is performed.

Mathematical functions can also be used in conditions.

Example:
```
IF Y = SQR(COS(2*Freq))
```

In this example, the square root of the cosine of the quantity of 2 multiplied by Freq is calculated before the quantity is compared to Y.

Conditional expressions can be used without **IF** or other decision commands. Conditional expressions alone return a 1 if true and a 0 if false.

Example:
```
X=W<2*y**2
```

If W is less than $2y^2$, X is set to 1. If W is greater than $2y^2$, X is set to 0.

Example:
```
X=W=ABS(W)
```

If W is positive, W equals ABS(W) and X is set to 1. If W is negative, W does not equal ABS(W) and X is set to 0.

2-11 VARIABLES AND ARRAYS IN MACROS

Variables and arrays are passed to a macro when executed by entering variables and arrays (parameters) after the name of the macro. Inside the macro, characters $1, $2, $3,...., $9 are placeholders for the passed parameters. Parameters may be passed to macros during execution.

Example:
```
*DMC "Compute",Y=$1**$2;PRINT $1," TO THE POWER OF",$2," IS",Y
```

Entering `Compute 4,3` executes the macro Compute and sets $1 equal to 4 and $2 equal to 3. The displayed result: `4 TO THE BOWER OF     3 IS     64.`

Example:
```
VAR LIST[2]={1,3,5}

*DMC "Average",BEGIN
VAR SUM=0
FOR N=0 TO $1-1
SUM=SUM + $2[N]
NEXT N
PRINT "THE AVERAGE IS ",SUM/$1
END
Average 3,LIST
```

The macro Average reads an array and finds the average of the elements of the array. The length and name of the array is passed upon macro execution. When Average 3, LIST is entered, array LIST is passed using a **FOR** loop. The macro then prints the average of the 3 array values.

Example:
```
Add=&Compute
EXEC Add,4,3
```

A comma is placed before the variables when a macro is executed using the EXEC command. This example executes the macro Compute and has the same result as the first example.

The **INPUT** command allows data entry during macro execution. Data is entered using the Test Set DATA ENTRY Keypad (HOST) or the RS-232 terminal keyboard (Sp Tst). For the HOST, the **INPUT** command causes a blinking cursor to be displayed. The **EDIT:WIDTH** command limits the width of the blinking cursor to *n* number of pixels.

Example:
```
*DMC "Compute",BEGIN
EDIT:COLOR:MENU 4          // Sets Input background color to Red.
EDIT:COLOR:LETTER 11       // Sets entered data color to Cyan.
EDIT:WIDTH 32              // Sets cursor width to 32 pixels.
INPUT X                    // Displays a blinking cursor
                           // and sets X to entered value.
Y=$1**X
PRINT $1,"TO THE POWER OF ",X," IS ",Y
END
```

If Compute 5 is entered to execute the macro and 3 is entered using the DATA ENTRY Keypad (HOST), `5 TO THE POWER OF 3 IS 125` is printed.

2-12 MULTITASKING MACROS

Multitasking allows command execution to alternate between different macros prior to their completion. This allows complex macros to be separated into more manageable macros. Before sharing execution time with other macros, a macro must: 1) Be declared a task using the **TASK** command; and 2) Be put into the schedule queue using the **ACTIVATE** command.

Multitasking requires the first macro in the schedule queue also be the last macro to finish execution.

See Section 3 for a description of each command.

The following commands provide multitasking control:

TASK	**ACTIVATE**	**TPAUSE**	**TSTOP**
SLEEP	**KILL**	**WAKE**	

Example:

```
*DMC "Main",BEGIN
ACTIVATE "First"
ACTIVATE "Second"
A=0
B=0
WHILE A=0 OR B=0
  TPAUSE
WEND
END

*DMC "First",BEGIN
FOR x = 1 to 10
  PRINT "FIRST"
  IF x=3
    A = 1
    TSTOP
  ENDIF
  TPAUSE
NEXT x
A = 1
END

*DMC "Second",BEGIN
n=1
FOR m=1 to 10
  PRINT "SECOND"
  IF n=2
    TPAUSE
    n=0
  ENDIF
    n=n+1
  NEXT m
  B = 1
END

TASK "Second"   // These TASK commands are placed after macros
TASK "First"    // First and Second, because each macro must be
                // defined before being declared a task.
```

As the above list of commands are uploaded into memory (see Section 4), macros Main, First and Second are defined, followed by the two **TASK** commands. The **TASK** command declares a defined macro as a task. If a **TASK** command is encountered that declares an undefined macro, an error occurs.

Executing the macro Main places Main in the schedule queue and, then using the **ACTIVATE** command, loads tasks First and Second into the schedule queue with Main. A **WHILE** loop assures Main executes until First and Second have finished execution. The following sequence occurs:

- Macro execution enters **WHILE** loop, because conditions are met. The **TPAUSE** command is encountered which causes the execution of Main to pause and pass execution on to the next macro in the schedule queue. The next macro in the schedule queue is First.
- First begins execution, x is set to 1 and `FIRST` is printed. The **IF ENDIF** command is skipped ($x \neq 3$), the **TPAUSE** command executes and command execution passes to Second.
- Second begins execution, n and m are set to 1 and `SECOND` is printed. The **IF ENDIF** command is skipped ($n \neq 2$) and n is incremented to 2. m is incremented to 2 and command execution loops to top of the **FOR** loop. `SECOND` is printed and the **TPAUSE** command inside the **IF ENDIF** command executes passing command execution to Main.
- Command execution begins again after the **TPAUSE** command and returns to the beginning of the **WHILE** loop. A and B are compared and the **TPAUSE** command is executed, passing command execution to First.
- In First, command execution begins again after the **TPAUSE** command. x is incremented to 2, command execution loops to the top of the **FOR** loop and `FIRST` is printed. The **IF ENDIF** command is skipped ($x \neq 3$) and the **TPAUSE** command passes command execution to Second.
- In Second, command execution begin again after the **TPAUSE** command. n is set to 0 and then incremented to 1. m is incremented to 3 and command execution loops to the top of the **FOR** loop. `SECOND` is printed, the **IF ENDIF** command is skipped ($n \neq 2$) and n is incremented to 2. m is incremented to 4 and command execution loops to the top of the **FOR** loop. `SECOND` is printed and the **TPAUSE** command inside the **FOR** loop executes passing command execution to Main.
- Command execution begins again after the **TPAUSE** command and returns to the beginning of the **WHILE** loop. A and B are compared and the **TPAUSE** command is executed, passing command execution to First.
- In First, command execution begins again after the **TPAUSE** command. x is incremented to 3, command execution loops to the top of the **FOR** loop and `FIRST` is printed. A is set to 1 and the **TSTOP** command executes taking First out of the schedule queue. Command execution passes to Second.
- In Second, command execution begin again after the **TPAUSE** command. n is set to 0 and then incremented to 1. m is increment to 5, command execution loops to the top of the **FOR** loop and `SECOND` is printed. The **IF ENDIF** command is skipped ($n \neq 2$) and n is incremented to 2. m is incremented to 6, command execution loops to the top of the **FOR** loop and `SECOND` is printed. The **TPAUSE** command is executed, passing command execution to Main.

- Command execution begins again after the **TPAUSE** command and returns to the beginning of the **WHILE** loop. A and B are compared and the **TPAUSE** command is executed, passing command execution to Second (Main and Second are the only remaining macros in the schedule queue).
- In Second, command execution begin again after the **TPAUSE** command. n is set to 0 and then incremented to 1. m is increment to 7, command execution loops to the top of the **FOR** loop and SECOND is printed. The **IF ENDIF** command is skipped (n ≠ 2) and n is incremented to 2. m is incremented to 8, command execution loops to the top of the **FOR** loop and SECOND is printed. The **TPAUSE** command is executed, passing command execution to Main.
- Command execution begins again after the **TPAUSE** command and returns to the beginning of the **WHILE** loop. A and B are compared and the **TPAUSE** command is executed, passing command execution to Second.
- In Second, command execution begin again after the **TPAUSE** command. n is set to 0 and then incremented to 1. m is increment to 9, command execution loops to the top of the **FOR** loop and SECOND is printed. The **IF ENDIF** command is skipped (n ≠ 2) and n is incremented to 2. m is incremented to 10, command execution loops to the top of the **FOR** loop and SECOND is printed. The **TPAUSE** command is executed, passing command execution to Main.
- Command execution begins again after the **TPAUSE** command and returns to the beginning of the **WHILE** loop. A and B are compared and the **TPAUSE** command is executed, passing command execution to Second.
- In Second, command execution begin again after the **TPAUSE** command. n is set to 0 and then incremented to 1. m is increment to 11, command execution leaves the **FOR** loop. B is set to 1, and macro Second finishes execution leaving only Main in schedule queue. Command execution is returned to Main.
- The conditional expression of the **WHILE** loop is now false, causing command execution to leave the **WHILE** loop. Macro Main finishes execution.

2-13 DISPLAY AND SOUND CONTROL

TMAC allows creation of new screen displays and changes to display screen configurations. Different colors can be used. Windows can be created and moved. Pixels, boxes, lines and figures may be drawn and placed anywhere on the color display. Screen page control makes applications such as animation possible. Audio cues can be implemented.

Some commands work in both the HOST or Sp Tst. Other commands work only with one or the other as indicated.

2-13-1 COLORS

Table 2-2 lists the 16 colors, according to selection number, available on the color display.

COLOR	NUMBER	COLOR	NUMBER
Black	0	Dark Gray	8
Dark Blue	1	Blue	9
Dark Green	2	Green	10
Dark Cyan	3	Cyan	11
Dark Red	4	Red	12
Dark Magenta	5	Magenta	13
Brown	6	Yellow	14
Light Gray	7	White	15

Table 2-2 Colors and Color Selection Numbers

COLOR commands change the color selection for the color display except for Soft Function Keys and menus which are changed by **EDIT:COLOR** commands (HOST only).

Colors are entered using selection number or constant name (see Appendix A).

The following commands are used to control the foreground and background colors, the text and background colors of built-in HOST menus and the colors for the Soft Function Keys:

COLOR **COLOR?** **BCOLOR**

The following **EDIT** commands only apply to the HOST.

EDIT:COLOR:MENU **EDIT:COLOR:MENU?**
EDIT:COLOR:LETTER **EDIT:COLOR:LETTER?**
EDIT:COLOR:SOFT:BOX **EDIT:COLOR:SOFT:BOX?**
EDIT:COLOR:SOFT:LETTER **EDIT:COLOR:SOFT:LETTER?**
EDIT:COLOR:SOFT:SELECT **EDIT:COLOR:SOFT:SELECT?**
EDIT:WIDTH

See Section 3 for a description of each command.

2-13-2 COLOR DISPLAY

The display subsystem divides the color display screen into pixels: 640 in the horizontal direction and 350 in the vertical direction. Window and graphic commands use these pixels to determine screen location. Screen locations are specified using a coordinate system with point 0,0 located in the top left corner of the display screen. The positive horizontal component is to the right and the positive vertical component is in the downward direction.

2-13-3 WINDOWS

Windows, used primarily for menus, can be created and moved. There can be up to 15 windows on a screen at one time. Windows are numbered in the order they are opened and must be closed in reverse order if overlapped. The last window opened is the selected window unless selection is changed by a **WSEL** command. Windows can be any size or color and can be moved to any location on the screen. Following are the window commands:

CLS	**SCREEN:USER**	**USER**
WOPEN	**WMOVE**	**WCLOSE**
WSEL	**WINDOW?**	

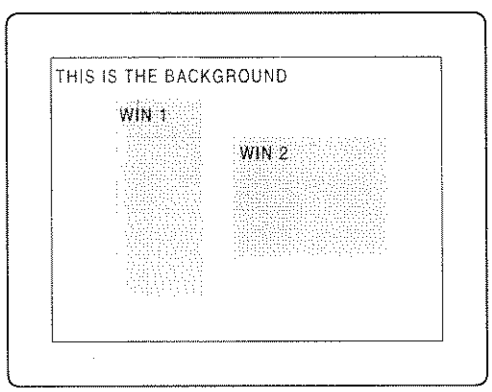
See Section 3 for a description of each command.

Print commands executed while a window is open prints inside the currently selected window. To print outside the windows, select window 0 (background screen).

Example:

```
VAR Win1,Win2
CLS
WOPEN green,100,50,250,300
Win1=WINDOW?
WOPEN red,300,100,550,250
PRINT "WIN 2"
Win2=WINDOW?
WSEL Win1
PRINT "WIN 1"
WSEL 0
PRINT "THIS IS THE BACKGROUND"
```

These commands produce the following display:



8618010

Figure 2-1 Window Example

The display control software forces the window locations and sites to the nearest multiple of eight (byte size). Therefore, the user should determine the location values prior to opening or moving a window.

2-13-4 SOFT FUNCTION KEY DISPLAYS

Soft Function Key definitions and the Soft Function Key frame at the bottom of the display screen are added using the following commands:

KEYPAD:SOFT **KEYPAD:LABel** **KEYPAD:ERASE**

See Section 3 for a description of each command.

Example:
```
CLS
KEYPAD:SOFT
KEYPAD:LAB 3," Test"
```

The following is shown on the Test Set color display:

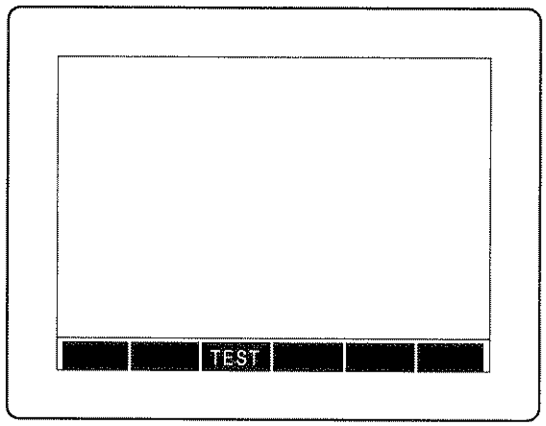


Figure 2-2 Soft Function Key Example

2-13-5 GRAPHICS AND TEXT

The following graphics and text commands create text, pixels, lines, boxes and user-defined shapes for display:

XY	**PIXEL**	**DRAW**	**BOX**
HEIGHT	**XYPRINT**	**PRINT**	**ELLIPSE**[1]
ICON[1]	**CENTER**[2]	**ERASE:TEXT**[2]	**HPRINT**[2]
LJPRINT[2]	**PIXLEN?**[2]	**RJPRINT**[2]	

1. HOST only.
2. Sp Tst only.

For Sp Tst operation, the **PRINT** command prints out the OPT. RS-232 Connector and the **HPRINT** (host print) command prints on the color display of the Test Set.

See Section 3 for a description of each command.

Example:

```
XY 400,100
COLOR blue,black
PIXEL
DRAW 200,100,250,300,red
BOX 0,250,20,350,120,green
BOX 1,280,200,550,280,magenta
```

See Table 2-2 for available colors and color numbers.

This set of commands creates the following graphics displayed in Figure 2-3.

- Blue pixel at point 400,100 and red line from point 200,100 to point 250,300.
- Hollow green box with top left corner at point 250,20 and lower right corner at point 350,120.
- Solid magenta box with top left corner at point 280,200 and lower right corner at point 550,280.

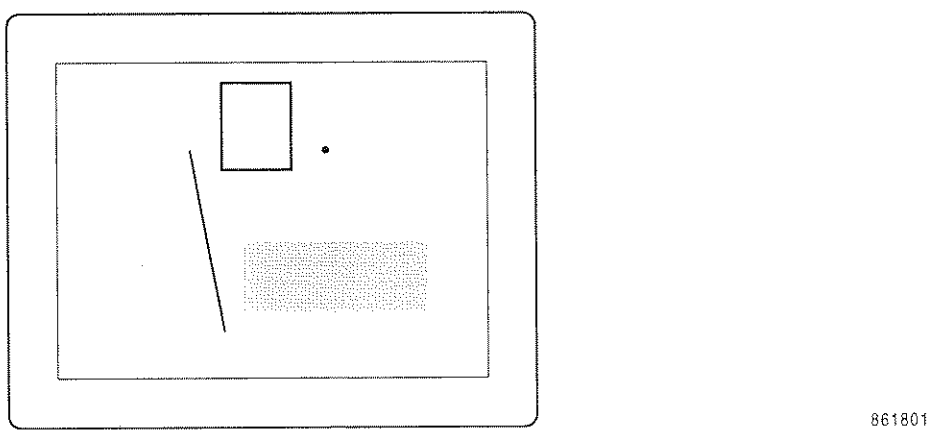
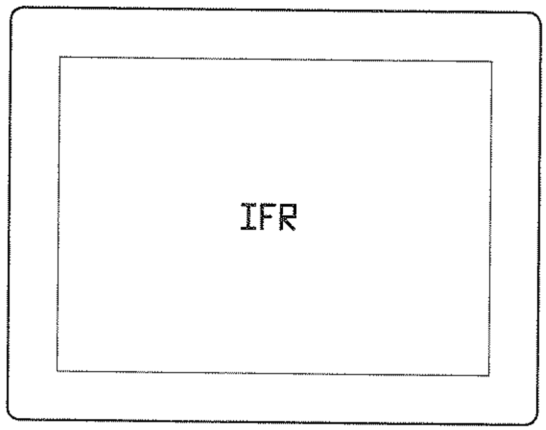


Figure 2-3 Graphics Examples

Example:

```
DATA bits={#hFFFFF3FF,#hFFCFFFFF,#hFFFFF3FF,#hFFCFFFFF,
           #hFFFFF3FF,#hFFCFFFFF,#hFFFFF3FF,#hFFCFFFFF,
           #h00F003C0,#h000F000F,#h00F003C0,#h000F000F,
           #h00F003C0,#h000F000F,#h00F003C0,#h000F000F,
           #h00F003FF,#hFC0FFFFF,#h00F003FF,#hFC0FFFFF,
           #h00F003FF,#hFC0FFFFF,#h00F003FF,#hFC0FFFFF,
           #h00F003C0,#h000F7E00,#h00F003C0,#h000F3F00,
           #h00F003C0,#h000F0F80,#h00F003C0,#h000F07E0,
           #hFFFFF3C0,#h000F01F8,#hFFFFF3C0,#h000F00FC,
           #hFFFFF3C0,#h000F003E,#hFFFFF3C0,#h000F001F}
XY 300,130
ICON 64,20,bits
```

The example creates the following image:



8618006

Figure 2-4 ICON Example

2-13-6 VIDEO PAGE CONTROL (HOST Only)

Two screen displays are available: one active and one static for use as pages to switch back and forth. Two video pages allow the user to construct or make changes to a screen display before actually displaying the new screen. Video Page Control allows the user to specify which video page to currently display or to copy from one to the other.

VIDEOpage:SET **VIDEOpage:COPY**

See Section 3 for a description of each command.

2-13-7 AUDIO TONES

Audio tones are generated using the **SOUND** command (see **SOUND** command in Section 3).

There are two predefined macros, Chirp_1 and Chirp_2, that produce a short series of tones. Chirp_1 and Chirp_2 are detailed in Appendix A.

2-14 SYSTEM COMMANDS

2-14-1 SYSTEM KEY COMMANDS

The System Key commands assign Front Panel Keys to command strings that can execute macros. Command strings are limited to 80 characters. Only 16 keys can be assigned at one time. Appendix B lists Front Panel Keys with keycodes. Appendix A lists Front Panel Keys having predefined constants.

The following are the Key commands:

SYSTem:KEY:DEFine[1]	**SYSTem:KEY:DELete**[1]
SYSTem:KEY[1]	**SYSTem:KEY?**[1]
KEYPAD:CLAIM[1]	**KEYPAD:UNCLAIM**[1]
KEY	**KEY?**

1. HOST only.

See Section 3 for a description of each command.

2-14-2 SYSTEM ERROR COMMAND

The **SYSTem:ERRor** command reports system errors. See Section 3 for a description of this command.

2-14-3 SYSTEM DEFAULTS COMMANDS (HOST Only)

The System Defaults commands restore settings in the HOST to default states.

SYSTem:DEFaults	**SYSTem:DISPlay:DEFaults**
SYSTem:CURsor:DEFaults	

See Section 3 for a description of each command.

2-14-4 SYSTEM PLOT COMMANDS (HOST Only)

The System Plot commands select the GPIB or HOST RS-232 Connector for plotter output.

SYSTem:PLOT:GPIB	**SYSTem:PLOT:SERial**

See Section 3 for a description of each command.

2-14-5 SYSTEM TIME AND DATE COMMANDS

System Time and Date commands allow setting and recording the date and time, in addition to determining elapse time.

SYSTem:DATE?	**SYSTem:DATE**
SYSTem:TIME?	**SYSTem:TIME**
SYSTem:MILLIsec?	**TICKS?**

See Section 3 for a description of each command.

2-14-6 SYSTEM RS-232 CONFIGURE COMMANDS

The following commands, applying only to the HOST, allow the Test Set to control another device using serial RS-232 communication. Commands unique to the device controlled are sent as strings. Responses are received as strings.

SYSTem:PTHRough:SERial **SYSTem:PTHRough:SERial?**
SYSTem:PTHRough:SERial:QUEue? **SYSTem:PTHRough:SERial:KEY?**

The following commands, applying to the HOST and Sp Tst, edit the RS-232 parameters remotely and select the RS-232 Connector for printer output:

SYSTem:COMMunicate:SERial:BAUD
SYSTem:COMMunicate:SERial:PARity
SYSTem:COMMunicate:SERial:BITS
SYSTem:COMMunicate:SERial:SBITs
SYSTem:COMMunicate:SERial:PACE
SYSTem:COMMunicate:SERial:ECHO
SYSTem:COMMunicate:SERial:PRINTer

See Section 3 for a description of each command.

2-15 GPIB OPERATION

The Sp Tst operates as a GPIB Talker/Listener device. The HOST operates as a GPIB Talker/Listener, Controller, Talk Only or Listen Only device. The GPIB Mode of Operation is selected using commands received through the RS-232 Connector, commands from a macro executed within the Test Set or from the Front Panel Auxiliary Functions Menu.

Commands are received and transmitted as strings on GPIB. Convert all commands to strings before transmitting. Precede non-printable characters with the \ character to transmit.

Both the HOST and the Sp Tst can operate on the same GPIB at the same time, using different addresses.

2-15-1 OPERATING TEST SET USING GPIB COMMUNICATION (Sp Tst Only)

The Sp Tst is a GPIB Talker/Listener. The Sp Tst provides a user-defined Status Byte to generate Service Requests. The following commands are used when operating the Sp Tst on the GPIB:

GPIB:ADDRess **GPIB:MASK** **GPIB:SRQ**

See Section 3 for a description of each command.

2-15-2 OPERATING TEST SET USING GPIB COMMUNICATION (HOST Only)

For the Test Set to become a GPIB Talker/Listener, Talk/Listen is selected as the GPIB Mode. This function is performed remotely by executing the following command:

SYSTem:COMMunicate:GPIB:ADDRess

The following suggested procedures operate the HOST Test Set using GPIB communication:

A. Initialization of the Test Set

- Prepare Controller for adding the HOST Test Set to the GPIB.
- Clear the HOST Test Set (see ***CLS** in Section 3).
- Send "*ESE 1" command to Test Set. This command sets the event register of the Standard Event Status Register to 1 to allow reading of the operation complete bit.

B. Sending Commands to Test Set

- Send "*SRE 36" command to the Test Set. This command sets the Service Request enable register to generate an RQS upon the occurrence of an OPC or Error.
- Send desired command with "*OPC" command following.
- Wait for RQS bit to be set (bit 6 of Status Byte) or for a time out.
- Retrieve Status Byte of Test Set and store response to value variable.
- If (value AND 4) is true, send error has occurred.
- If (value AND 4) is false and (value AND 32) is true, command executed.

C. Sending Queries to Test Set

- Send "*SRE 20" command to Test Set. This command sets the Service Request enable register to report the Message Available (MAV) bit and error bit.
- Send query command to Test Set.
- Wait for RQS bit to be set (bit 6 of Status Byte) or for a time out.
- Retrieve Status Byte of Test Set and store response to value variable.
- If (value AND 4) is true, send error has occurred.
- If (value AND 4) is false and (value AND 16) is true, send read command to return results of query.

D. Sending Macros to Test Set

Perform the following procedure until all command lines of macro are sent to Test Set.

- Send command line to Test Set.
- Wait for CMPL (completed I/O) or for a time out.
- If time out, stop sending of macro. Restart sending of macro (see ***PMC** in Section 3).

2-15-3 TEST SET OPERATING AS GPIB CONTROLLER (HOST Only)

For the Test Set to become a GPIB controller, the address of the Test Set must be set to an address that does not conflict with a slave device address (30 is advised) and the GPIB Operation Mode must be set to Controller. The commands used to set these parameters are:

SYSTem:COMMunicate:GPIB:ADDRess
SYSTem:COMMunicate:GPIB:CONTroller

Once the Test Set is made a GPIB Controller, the talker/listener status is returned by repeating the **SYST:COMM:GPIB:ADDR** command. The following commands are used when operating the Test Set as a GPIB Controller:

SYSTem:COMMunicate:GPIB:SLAVe
SYSTem:PTHRough:GPIB
SYSTem:PTHRough:GPIB?
SYSTem:COMMunicate:GPIB:DCL
SYSTem:COMMunicate:GPIB:GET
SYSTem:COMMunicate:GPIB:CMD
SYSTem:COMMunicate:GPIB:SRQ?
SYSTem:COMMunicate:GPIB:SPOLL?

See Section 3 for a description of each command.

2-15-4 TEST SET OPERATING AS TALKER ONLY (HOST Only)

When the Test Set becomes a GPIB Talk Only device, addressing is unnecessary. The Test Set must be the only talking device on the GPIB bus and the remaining devices must be compatible listeners. The following command selects Talk Only as the GPIB Mode of Operation.

SYSTem:COMMunicate:GPIB:TONly

See Section 3 for a description of this command.

To select the Talk/Listen GPIB Mode of Operation again, reset the Test Set address using the **SYST:COMM:GPIB:ADDR** command.

2-15-5 TEST SET OPERATING AS LISTENER ONLY (HOST Only)

When the Test Set becomes a GPIB Listen Only device, the address is set first using the **SYST:COMM:GPIB:ADDR** command. After setting the address, the following command selects Listen Only as the GPIB Mode of Operation.

SYSTem:COMMunicate:GPIB:LONly

See Section 3 for a description of this command.

To select the Talk/Listen GPIB Mode of Operation again, reset the Test Set address using the **SYST:COMM:GPIB:ADDR** command.

2-15-6 TEST SET GPIB PRINT CONTROL (HOST Only)

The following command allows the user to direct the output of the printer to the GPIB Connector:

SYSTem:COMMunicate:GPIB:PRINTer

See Section 3 for a description of this command.

2-15-7 TEST SET FREQUENCY CONTROL (HOST Only)

The following commands provides the user the following remote capabilities:

- To slave the RF Generator, Receiver and Spectrum Analyzer frequency, together. When a user enters a frequency through either one of the three operations, the remaining two operations automatically change to the identical frequency.
- To configure, manipulate or query settings in the Frequency List.

The following TMAC commands provide frequency controls for the HOST in that affect multiple operation modes:

SYSTem:FREQuency:LOCK	**SYSTem:FREQuency:LOCK?**
SYSTem:FREQList:GENerator	**SYSTem:FREQList:GENerator?**
SYSTem:FREQList:OFFset	**SYSTem:FREQList:OFFset?**
SYSTem:FREQList:RECeiver	**SYSTem:FREQList:RECeiver?**
SYSTem:FREQList:SCAN	**SYSTem:FREQList:SCAN?**

See Section 3 for a description of each command.

2-16 STATUS SUBSYSTEM (HOST Only)

TMAC provides an implementation of the Event Status register structure as designated by IEEE-488.2 and SCPI Standards. Standard Event, Operation, Operation Instrument, Questionable, Instrument and Instrument Summary Status Registers and their relationship to one another are shown in Figure 2-6.

2-16-1 STATUS REGISTERS

Each Status Register shown in Figure 2-6 consists of a condition, event and enable register shown in Figure 2-5. The condition register continually receives data coming into the Status Register. When a bit is changed from 0 to 1 for the first time since being cleared, the same bit in the event register is set to 1. The event register is only changed by the first 0 to 1 transitions that occur in the condition register. The condition or event registers are cleared when read.

The 16 bits of the event register and the 16 bits of the enable register undergo a bitwise AND operation. In order for an event register bit to be reported, the corresponding bit in the enable register must be set to 1. The 16 bits resulting from this bitwise AND operation undergo an OR operation with each other. The result is 1 if one or more bits were 1.

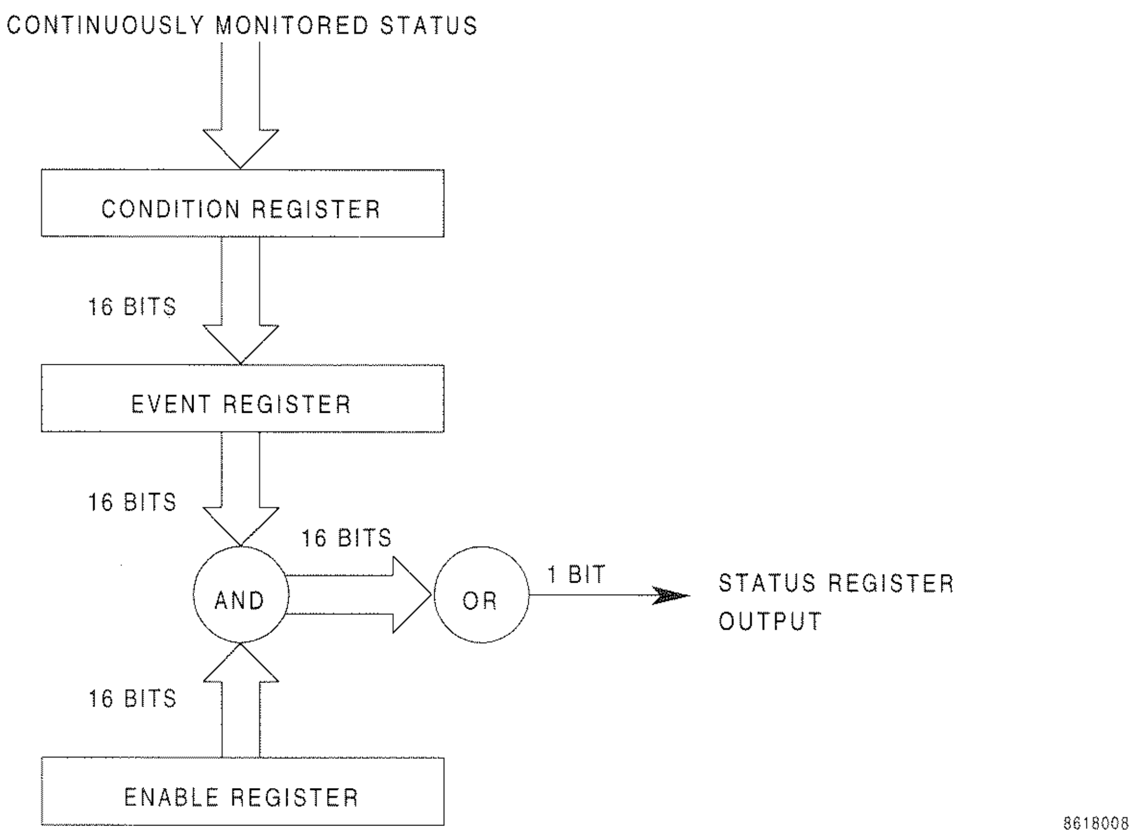


Figure 2-5 HOST Status Register Description

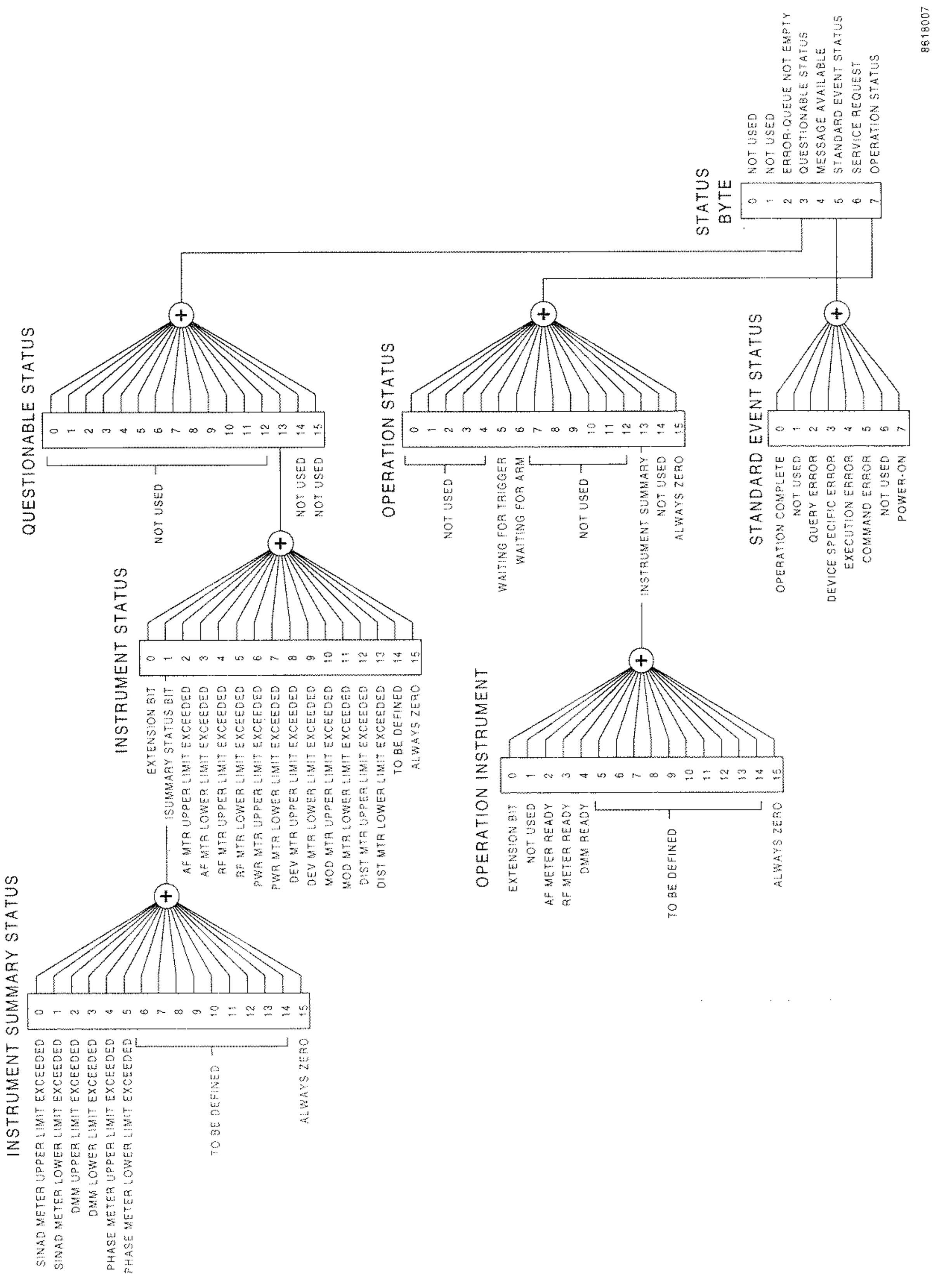


Figure 2-6 HOST Status Register Configuration

STATus:PRESet
Sets all bits in enable registers to an initialized condition. Presets the enable registers of the following status registers to the associated condition:

STATUS REGISTER	CONDITION
Operation Status	OFF (#h0000)
Operation Instrument	OFF (#h0000)
Questionable Status	OFF (#h0000)
Instrument Status	ON (#h7FFF)
Instrument Summary	ON (#h7FFF)

2-16-2 STATUS BYTE

The Status Byte reports an occurrence in any of the Status Registers. and is used to activate a SRQ message and the RQS bit (bit 6 of the Status Byte) (see Figure 2-6). A Status Byte bit determines an SRQ and RQS if the corresponding Service Request enable register bit is set to 1. Bits 0 to 5 and bit 7 of the Status Byte and the same bits of the Service Request enable register undergo a bitwise AND operation. The resulting bits of the AND operation undergo an OR operation. If any of the enabled Status Byte bits are 1, an SRQ is generated and the RQS Status Byte bit (bit 6) is set to 1.

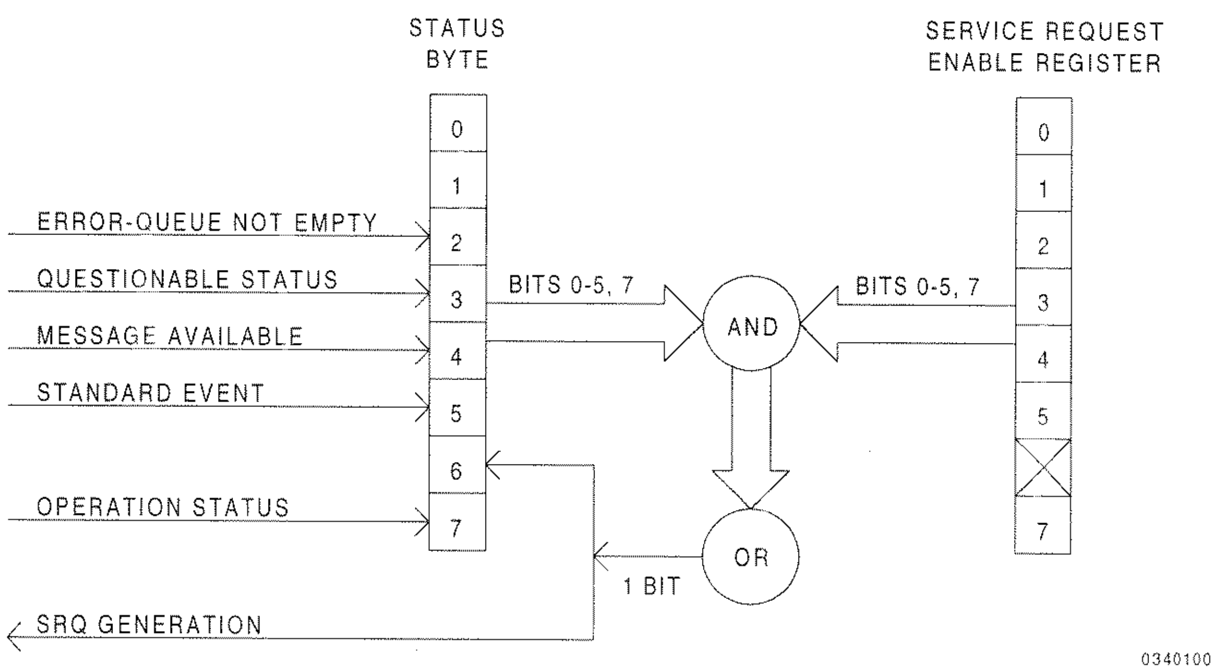


Figure 2-7 HOST RQS Bit and SRQ Generation

The following commands pertain to the Status Byte and Service Request enable register:

***SRE** ***SRE?** ***STB?**

See Section 3 for a description of each command.

2-16-3 STANDARD EVENT STATUS REGISTER

The Standard Event Status Register contains an operation complete bit, several error bits and a power on bit. Figure 2-6 shows the Standard Event Status Register bits. The power on bit is set to 1 following power up of the Test Set. The following commands control this register:

***ESE**	***ESE?**	***ESR?**
***OPC**	***OPC?**	

See Section 3 for a description of each command.

Example:

```
*DMC "ERROR_CHECK",BEGIN    // Define macro named ERROR_CHECK.
*ESE 61                     // Enable Std Event Status Register.
IF (*ESR & 1)= 0            // Check Operation Complete bit.
  IF (*ESR? & 4)=1          // If Standard Event Status Register
    PRINT "QUERY ERROR"     // reports Query Error, print message.
  ELIF (*ESR? & 8)=1        // If Standard Event Status Register
                            // reports Device Specific Error,
    PRINT "DEVICE SPECIFIC ERROR" // print message.
  ELIF (*ESR? & 16)=1       // If Std Event Status Rgtr reports
    PRINT "EXECUTION ERROR"// Execution Error, print message.
  ELIF (*ESR? & 32)=1       // If Standard Event Status Register
    PRINT "COMMAND ERROR"   // shows Command Error, print message.
  ELSE                      // If Std Event Status Rgtr does not
    PRINT "NO ERROR DETECTED"  // report an error type, print msg.
```

2-16-4 OPERATION STATUS REGISTER

The Operation Status Register contains the Waiting For Arm, Waiting For Trigger and Instrument Summary bits. The Operation Instrument Register sets the Instrument Summary bit (see 2-16-5). The following commands control the Operation Status Register:

STATus:OPERation:CONDition?	**STATus:OPERation:ENABLe**
STATus:OPERation:ENABLe?	**STATus:OPERation:EVENt?**

See Section 3 for a description of each command.

2-16-5 OPERATION INSTRUMENT REGISTER

The Operation Instrument Register is used with the **INITiate** and **FETCh** commands. Once an **INITiate** command is completed and the meter is ready to be read, the appropriate meter ready bit is set to 1. This indicates the meter is ready for the **FETCh** command. The following **STATUS** commands control the Operation Instrument Register:

STATus:OPERation:INSTRument:CONDition?
STATus:OPERation:INSTRument:ENABLe
STATus:OPERation:INSTRument:ENABLe?
STATus:OPERation:INSTRument:EVENt?

See Section 3 for a description of each command.

2-16-6 QUESTIONABLE STATUS REGISTER

The Questionable Status Register reports events that could bring Test Set results into question. The Instrument Status Register provides the input. Figure 2-6 shows events reported by both registers. The following **STATUS** commands control the Questionable Status Register:

STATus:QUEStionable:CONDition? **STATus:QUEStionable:ENABLe**
STATus:QUEStionable:ENABLe? **STATus:QUEStionable:EVENt?**

See Section 3 for a description of each command.

2-16-7 INSTRUMENT STATUS REGISTER

The Instrument Status Register reports meter readings exceeding the upper and lower limit of the AF, RF Power, FM Deviation, AM Modulation and Distortion Meters. The results of the Instrument Summary Status Register (an extension of the Instrument Status Register) are also reported. The following **STATUS** commands control the Instrument Status Register.

STATus:QUEStionable:INSTRument:CONDition?
STATus:QUEStionable:INSTRument:ENABLe
STATus:QUEStionable:INSTRument:ENABLe?
STATus:QUEStionable:INSTRument:EVENt?

2-16-8 INSTRUMENT SUMMARY STATUS REGISTER

The Instrument Summary Status Register acts as an extension of the Instrument Status Register. The Instrument Summary Status Register reports the exceeding of the upper and lower limit of the SINAD and Phase Meter and Digital Multimeter. Following are **STATUS** commands controlling the Instrument Summary Status Register.

STATus:QUEStionable:INSTRument:ISUMmary:CONDition?
STATus:QUEStionable:INSTRument:ISUMmary:ENABLe
STATus:QUEStionable:INSTRument:ISUMmary:ENABLe?
STATus:QUEStionable:INSTRument:ISUMmary:EVENt?

The following example performs a DMM reading and checks the Status Byte for the reporting of the DMM Upper or Lower Limit being exceeded. The enable registers of the Instrument Summary, Instrument and Questionable Status Registers are set to pass the information to the Status Byte.

Example:

```
*DMC "DMM_READ",BEGIN                // Define macro named DMM_READ.
STAT:QUES:INSTR:ISUM:ENABLE 12       // Set enable register to read DMM
                                     // Upper and Lower Limit exceeded.
STAT:QUES:INSTR:ENABLE 2             // Set enable register to pass the
                                     // Instrument Summary Status
                                     // Register result.
STAT:QUES:ENABLE 8192                // Set enable register to pass the
                                     // Instrument Status Register result.
*CLS                                 // Clear all condition and event
                                     // registers.
SCREEN:DMM                           // Display DMM Operation Screen.
PPRINT M_DMM?                        // Print DMM reading to Host.
IF (*STB? & 8) != 0                  // If Status Byte reports
                                     // Questionable Status bit as 1,
    PPRINT "limit exceeded"          // print limit exceeded to Host.
  ELSE                               // If Status Byte reports
                                     // Questionable Status bit as 0,
    PPRINT "limit not exceeded"      // print limit not exceeded to Host.
ENDIF                                // End IF statement.
END                                  // End macro DMM_READ.
```

2-17 IEEE 488.2 COMPLIANCE COMMANDS

TMAC contains the following IEEE-488.2 mandated commands:

COMMAND	DESCRIPTION	WHERE USED
*CLS	Clear Status	HOST and Sp Tst
*DDT	Define Device Trigger	HOST
*DMC	Define Macro	HOST and Sp Tst
*EMC	Enable Macro	HOST
*EMC?	Enable Macro query	HOST
*ESE	Standard Event Status Enable	HOST
*ESE?	Standard Event Status Enable query	HOST
*ESR?	Standard Event Status Register query	HOST
*IDN?	Identification query	HOST and Sp Tst
*LMC?	List Macro Query	HOST
*OPC	Operation Complete	HOST
*OPC?	Operation Complete query	HOST
*OPT?	Option identification query	HOST
*PMC	Purge Macro	HOST and Sp Tst
*RCL	Recall	HOST
*RST	Reset	HOST
*SAV	Save	HOST
*SRE	Service Request Enable	HOST
*SRE?	Service Request Enable query	HOST
*STB?	Read Status Byte query	HOST
*TRG	Trigger	HOST
*TST?	Self-Test query	HOST
*WAI	Wait To Continue	HOST and Sp Tst
See Section 3 for a description of each command.		

Table 2-3 IEEE 488.2 Compliance Commands

2-18 GENERAL AND SPECIFIC TMAC COMMANDS

Section 3 discribes in detail TMAC commands that are common to both the HOST and Sp Tst. TMAC commands specific to HOST and Special Test are described in Sections 6 and 9, respectively.

SECTION 3 - GENERAL TMAC COMMANDS

The following pages contain General TMAC commands (commands that apply to IFR-1900CSA Host **and** Special Test [Sp Tst]); listing use, syntax and action. Examples are also provided to aid in the understanding of the commands. Cross references to related commands are also included. Commands are listed in alphabetical order.

+

USE Addition operator or positive unary operator.

SYNTAX *argument1+argument2* or *+argument3*

ELEMENT	DESCRIPTION
argument1	Any constant, variable, number, string or any expression that evaluates to a number or string.
argument2	Same as *argument1*, except must be of the same type used for *argument1*.
argument3	Any constant, variable, number or any expression that evaluates to a number.

ACTION Adds two expressions or indicates the positive value of an expression. (*argumentX* may be any numerical value, variable or any mathematical operation resulting in a numerical value.)

EXAMPLES

```
G=4+F              // Assigns 4 added to the value of F to G.
AA  =  B1 + B2     // Assigns value of B1 added to B2 to AA.
+X                 // The value of X (i.e., no change).
```

-

USE Subtraction operator or negative unary operator.

SYNTAX *argument1-argument2* or *-argument3*

ELEMENT	DESCRIPTION
argument1	Any constant, variable, number, string or any expression that evaluates to a number or string.
argument2	Same as *argument1*, except must be of the same type used for *argument1*.
argument3	Any constant, variable, number or any expression that evaluates to a number.

ACTION Subtracts the value of *argument2* from the value of *argument1* or assigns a negative value to the value of *argument3*.

EXAMPLES

```
H= I-5             // Assigns the resultant value of 5
                   // subtracted from the value of I to H.
V = U-T            // Assigns the resultant value of T
                   // subtracted from the value of U to V.
-B                 // The negative or opposite value of B.
```

~

USE Bitwise complement.

SYNTAX *~argument*

ELEMENT	DESCRIPTION
argument	Any constant, variable, number or any expression that evaluates to a number.

ACTION Inverts the binary representation of the operand using the one's complement form of negative numbers.

SEE ALSO **!, &, ^, ¦, AND, OR**

EXAMPLE

```
X = ~6                    // 0110 (6) is inverted to 1001
                          // (one's complement of -7) and
                          // assigns -7 to X.
```

!

USE Logical negation (NOT) unary operator.

SYNTAX **!***argument*

ELEMENT	DESCRIPTION
argument	Any constant, variable, number or any expression that evaluates to a number.

ACTION Reverses the truth value.

SEE ALSO **~, &, ^, ¦, AND, OR**

EXAMPLE

```
!D                        // The example is true when D equals 0.
```

** (Double Asterisk)

USE Exponential operator (does not conform to SCPI Standard).

SYNTAX argument1**argument2

ELEMENT	DESCRIPTION
argument1	Any constant, variable, number or any expression that evaluates to a number.
argument2	Same as *argument1*.

ACTION Raises value of *argument1* to the power of the value of *argument2*.

EXAMPLES

```
y=4**-2                   // Assigns 4 raised to the -2 power to y.
MM =  NN**4               // Assigns NN raised to the 4th power
                          // to MM.
Freq = C**D               // Assigns C raised to the D power to Freq.
```

* (Single Asterisk)

USE Multiplication operator.

SYNTAX *argument1*argument2*

ELEMENT	DESCRIPTION
argument1	Any constant, variable, number or any expression that evaluates to a number.
argument2	Same as *argument1*.

ACTION Multiplies *argument1* by *argument2*.

SEE ALSO /

EXAMPLES

```
x=4*6                    // Assigns 4 multiplied by 6 to x.
T = 25*S                 // Assigns 25 multiplied by S to T.
JJ = K*LL                // Assigns K multiplied by LL to JJ.
```

/

USE Division operator.

SYNTAX *argument1/argument2*

ELEMENT	DESCRIPTION
argument1	Any constant, variable, number or any expression that evaluates to a number.
argument2	Same as *argument1* except cannot be zero.

ACTION Divides the value of *argument1* by the value of *argument2*.

SEE ALSO *

EXAMPLES

```
Y=8/3.867                // Assigns the quotient of 8 divided
                         // by 3.867 to Y.
Z = x/7                  // Assigns the quotient of x divided
                         // by 7 to Z.
P = Q/R                  // Assigns the quotient of Q divided
                         // by R to P.
```

A command to divide by zero results in an error message.

%

USE Modulo operator (does not conform to SCPI Standard).

SYNTAX *argument1%argument2*

ELEMENT	DESCRIPTION
argument1	Any constant, variable, number or any expression that evaluates to a number.
argument2	Same as *argument1*, except cannot be zero.

ACTION Returns the remainder (an integer) of a division operation. If not integers, the operands are truncated.

EXAMPLES

```
x=7%3           // Assigns 7 modulo 3 to x (x=1).
RR=16%4         // Assigns 16 modulo 4 to RR (RR=0).
N=20.5%3.3      // Assigns 20 modulo 3 to N (N=2).
M=27%L          // Assigns 27 modulo the value of L to M.
```

<<

USE Shift left operator.

SYNTAX *argument1<<argument2*

ELEMENT	DESCRIPTION
argument1	Any constant, variable, number or any expression that evaluates to a positive number.
argument2	Same as *argument1*.

ACTION Shifts the binary representation of the integer value of *argument1* to the left the number of bits specified by the integer value of *argument2*. Emptied bit positions (as result of shifting) are filled with zeros. Shift operators use the two's complement representation for negative numbers.

> Shifts a maximum of 31 bit positions, but *argument2* can be greater than 31. Use the following equation to calculate the number of bit positions shifted to the left: NumShifted = *argument2* % 32.

SEE ALSO >>

EXAMPLES

VAR DIV4	
m = 2	

DIV4 = 75<<m	// Shifts 75 (1001011) left 2 bits to
	// to become 300 (100101100) and is
	// assigned to DIV4. DIV4 = 300_{10}.

X = -19<<3	// Shifts -19
	// (#b11111111111111111111111111101101)
	// left 3 bits to become -152
	// (#b11111111111111111111111101101000)
	// and is assigned to X. X = -152_{10}.

>>

USE Shift right operator.

SYNTAX *argument1>>argument2*

ELEMENT	DESCRIPTION
argument1	Any constant, variable, number or any expression that evaluates to a positive number.
argument2	Same as *argument1*.

ACTION Shifts the binary representation of the integer value of *argument1* to the right the number of bits specified by the integer value of *argument2*. Emptied bit positions (as result of shifting) are filled with zeros. Shift operators use the two's complement representation for negative numbers.

> Shifts a maximum of 31 bit positions, but *argument2* can be greater than 31. Use the following equation to calculate the number of bit positions shifted to the right: NumShifted = *argument2* % 32.

SEE ALSO <<

EXAMPLE

```
Q = 14>>1            // Shifts 14 (1110) right 1 bit to
                     // to become 7 (0111) and is
                     // assigned to Q.  Q = 7₁₀.
```

&

USE Bitwise AND

SYNTAX *argument1&argument2*

ELEMENT	DESCRIPTION
argument1	Any constant, variable, number or any expression that evaluates to a number.
argument2	Same as *argument1*.

ACTION Performs AND operation on each bit of the integer binary representation of two operands. Table 3-1 lists the bit combinations and the AND operation results.

OPERAND 1	OPERAND 2	AND RESULT
0	0	0
0	1	0
1	0	0
1	1	1

Table 3-1 Bitwise AND Operation Results

SEE ALSO !, ~, ^, ¦, **AND**, **OR**

EXAMPLES

```
JJ = 4 & 6        // 0100 (4) is ANDed to 0110 (6).
                  // The resultant value is 0100 (4).
                  // JJ is assigned the value of 4.

ww = 12 & 10      // 1100 (12) is ANDed to 1010 (10).
                  // The resultant value is 1000 (8).
                  // ww is assigned the value of 8.
```

^

USE Bitwise XOR (conflicts with SCPI Standard for Exponential operator).

SYNTAX *argument1^argument2*

ELEMENT	DESCRIPTION
argument1	Any constant, variable, number or any expression that evaluates to a number.
argument2	Same as *argument1*.

ACTION Performs Exclusive-OR (XOR) operation on each bit of the integer binary representation of *argument1* and *argument2.* Table 3-2 lists the bit combinations and the XOR operation results. Operand 1 represents a bit in *argument1* and Operand2 represents the associated bit in *argument2*.

OPERAND 1	OPERAND 2	XOR RESULT
0	0	0
0	1	1
1	0	1
1	1	0

Table 3-2 Bitwise XOR Operation Results

SEE ALSO !, ~, &, ¦

EXAMPLES

```
y = 4 ^ 6         // The left bits of 4 (100) and 6 (110)
                  // are the same (both 1) providing an XOR
                  // result of 0.  The middle bits are
                  // different (0 and 1) providing an XOR
                  // result of 1.  The right bits are the
                  // same (both 0) providing an XOR result
                  // of 0.  Therefore the total result of
                  // the XOR operation is 010 (2); y is
                  // assigned the value of 2.

v = 12 ^ 10       // 12 (1100) and 10 (1010) is exclusive-
                  // OR'ed together, resulting in the value
                  // of 0110 (6) because the end bits are
                  // the same and the middle bits are
                  // different.  v is assigned the value
                  // of 6.
```

¦

USE Bitwise OR operator.

SYNTAX *argument1*¦*argument2*

ELEMENT	DESCRIPTION
argument1	Any constant, variable, number or any expression that evaluates to a number.
argument2	Same as *argument1*.

ACTION Performs OR operation on each bit of the integer binary representation of the two operands. Table 3-3 lists the bit combinations and the OR operation results. Operand 1 represents a bit in *argument1* and Operand2 represents the associated bit in *argument2*.

OPERAND 1	OPERAND 2	OR RESULT
0	0	0
0	1	1
1	0	1
1	1	1

Table 3-3 Bitwise OR Operation Results

SEE ALSO !, ~, &, ^

EXAMPLES

```
y = 4 ¦ 2             // The left and middle bits of 4 (100)
                      // and 2 (010) both result in 1 (11),
                      // when OR'ed together, because at least
                      // one bit in each position is 1.  The
                      // right bits of 4 (100) and 2 (010)
                      // result in 0 (0), when OR'ed together,
                      // because neither bit is 1.  Therefore
                      // the total result of the OR operation
                      // is 110 (6).  y is assigned the value
                      // of 6.

Z = 12 ¦ 10           // 12 (1100) and 10 (1010) are OR'ed
                      // together, resulting in the value 14
                      // (1110), because at least one of the
                      // bits in each position of the two
                      // expressions is a 1 except for the
                      // right position which for both
                      // expression is 0.  Z is assigned the
                      // value of 14.
```

=

USE Assignment or Equal logical operator.

SYNTAX *x = argument1* or *command argument2 = argument3*

ELEMENT	DESCRIPTION
x	Any numerical or string variable.
argument1	Any variable, number, string or any expression that evaluates to the same type as *x*.
command	Any decision point command (**IF**, **ELIF**, **UNTIL**, **WHILE**).
argument2	Any constant, variable, number, string or any expression that evaluates to a number or string.
argument3	Any constant, variable, number, string or any expression that evaluates to the same type as *argument2.*

ACTION Assigns a value to a variable. Variable on the left of the = operator is given the value of the expression on the right if the same type as the variable.

-or-

Provides condition for a decision point command. If conditional expression created with = (logical equal) evaluates to TRUE, then resulting command associated with decision point command is performed.

SEE ALSO **!, AND, OR, !=, <, >, <=, >=, DO UNTIL, IF ..., WHILE**

EXAMPLES

```
x=2                     // Assigns x the value of 2.
Mod = w * 2 + z         // Assigns the resultant value of expression
                        // to the right of = to Mod.
x = x + 2               // Assigns a new value to x that is 2
                        // greater than the previous value.
```

The assignment operator does not denote an equation.

!=

USE Not Equal relational operator.

SYNTAX *argument1*!=*argument2*

ELEMENT	DESCRIPTION
argument1	Any constant, variable, number, string or any expression that evaluates to a number or string.
argument2	Any constant, variable, number, string or any expression that evaluates to the same type as *argument2*.

ACTION Evaluates a Not Equal condition to TRUE or FALSE. Used for decisions in macros.

SEE ALSO **!, AND, OR, =, <, >, <=, >=, DO UNTIL, IF ..., WHILE**

EXAMPLE

```
x=3
if x!=4                  // Evaluates conditional expression (x is
                         // not equal to 4) as TRUE.
  print y                // Performs resulting command, because IF
                         // condition is TRUE.
endif
```

<

USE Less Than relational operator.

SYNTAX *argument1*<*argument2*

ELEMENT	DESCRIPTION
argument1	Any constant, variable, number, string or any expression that evaluates to a number or string.
argument2	Any constant, variable, number, string or any expression that evaluates to the same type as *argument2*.

ACTION Evaluates a Less Than condition to TRUE or FALSE. Used for decisions in macros.

SEE ALSO **!, AND, OR, =, !=, >, <=, >=, DO UNTIL, IF ..., WHILE**

EXAMPLE

```
var h=10, s=6
if h<s+4                 // Evaluates conditional expression (h is
                         // less than s + 4) as FALSE.
  print y                // Does not perform resulting command,
                         // because IF condition is FALSE.
endif
```

>

USE Greater Than relational operator.

SYNTAX *argument1>argument2*

ELEMENT	DESCRIPTION
argument1	Any constant, variable, number, string or any expression that evaluates to a number or string.
argument2	Any constant, variable, number, string or any expression that evaluates to the same type as *argument2*.

ACTION Evaluates a Greater Than condition to TRUE or FALSE. Used for decisions in macros.

SEE ALSO **!, AND, OR**, =, !=, >, <=, >=, **DO UNTIL, IF ..., WHILE**

EXAMPLES

```
var x=3
if x>4                   // Evaluates conditional expression (h is
                         // greater than 4) as FALSE.
  print y                // Does not performs resulting command,
                         // because IF condition is FALSE.
endif
if 5>4                   // Evaluates conditional expression (5 is
                         // greater than 4) as TRUE.
  print y                // Performs resulting command, because IF
                         // condition is TRUE.
endif
```

<=

USE Less Than Or Equal relational operator.

SYNTAX *argument1<=argument2*

ELEMENT	DESCRIPTION
argument1	Any constant, variable, number, string or any expression that evaluates to a number or string.
argument2	Any constant, variable, number, string or any expression that evaluates to the same type as *argument2*.

ACTION Evaluates a Less Than Or Equal condition to TRUE or FALSE. Used for decisions in macros.

SEE ALSO **!, AND, OR**, =, !=, <, >, >=, **DO UNTIL, IF ..., WHILE**

EXAMPLE

```
var t=16
if t*2<=32               // Evaluates conditional expression
                         // (2t is less than or equal to 32) as
                         // TRUE.
  print y                // Performs resulting command, because IF
                         // condition is TRUE.
endif
```

>=

USE Greater Than Or Equal relational operator.

SYNTAX *argument1>=argument2*

ELEMENT	DESCRIPTION
argument1	Any constant, variable, number, string or any expression that evaluates to a number or string.
argument2	Any constant, variable, number, string or any expression that evaluates to the same type as *argument2*.

ACTION Evaluates a Greater Than Or Equal condition to TRUE or FALSE. Used for decisions in macros.

SEE ALSO **!, AND, OR, =, !=, <, >, <=, DO UNTIL, IF ..., WHILE**

EXAMPLE

```
var x=10, t=4
if x-2>=24/t            // Evaluates conditional expression (8 is
                        // greater than or equal to 6) as TRUE.
  print y               // Performs resulting command, because
                        // IF condition is TRUE.
endif
```

++

USE Increment operator.

SYNTAX ++*variable* (increments before) or
variable++ (increments after)

ELEMENT	DESCRIPTION
variable	Any numeric variable.

ACTION Adds 1 to the value of a variable before or after execution of the command in which the operator is included.

SEE ALSO --

EXAMPLE

```
y = 7
x = y++                 // Assigns the value of y (7) to y, then
                        // increments y.  After the execution of
                        // the command, x = 7 and y = 8.
```

--

USE Decrement operator.

SYNTAX --*variable* (decrements before) or
variable-- (decrements after)

ELEMENT	DESCRIPTION
variable	Any numeric variable.

ACTION Subtracts 1 from the value of a variable before or after execution of the command in which the operator is included.

SEE ALSO ++

EXAMPLE

```
y = 7
x = --y                 // Subtracts 1 from y, then assigns this
                        // value to x.  After the second command
                        // is executed, x and y equal 6.
```

*CLS

USE To clear all Status Registers.

SYNTAX ***CLS**

ACTION Clear Status command. Sets all Status Registers including the Status Byte to zero.

SEE ALSO **SYSTem:ERRor?**

EXAMPLES See **STATus:QUEStionable:INSTRument:ISUMmary**.

*DDT (HOST Only)

USE To define what commands are executed by the Group Execute Trigger or ***TRG** command.

SYNTAX ***DDT**

ACTION Define Device Trigger command. Defines what commands are executed by the GET (Group Execute Trigger) command received at the GPIB Connector or the ***TRG** command.

EXAMPLES

```
*DDT SCOPE:TRIGGER:ONE;SCOPE:ARM  // Executes these commands
                                  // when a GET (Group Execute
                                  // Trigger) command is
                                  // received at the GPIB
                                  // Connector or a *TRG
                                  // command is executed.
```

*DMC

USE To define macros and functions.

SYNTAX ***DMC** "*name*", *command*; .... ; *command*

ELEMENT	DESCRIPTION
name	Name of the macro or function defined. The first character of the *name* is a letter while the remaining characters can be letters, digits and the underscore. The length of the *name* can be from 2 to 31 characters.
command	Any valid TMAC command or expression.

ACTION Defines a macro or function with the *name* specified and containing the *commands* listed. After a macro or function is defined, entering the macro or function's *name* executes all command contained within.

Macros utilize several lines using the **BEGIN** and **END** commands. A function is a macro with a **RETURN** command used to return values. The **RETURN** command returns results directly through the active interface connector.

SEE ALSO Macros (2-9), **BEGIN**, **END**, **RETURN**, ***LMC**, **FORGET**, ***PMC**

EXAMPLES

```
*DMC "Sq",SCREEN:USER;:INPUT X;Y=X**2;PPRINT X," squared is ",Y

/* The above *DMC command defines the following commands as
   a macro named Sq.  Upon entering Sq, the INPUT command is
   executed, Y is set to X² and the PPRINT command is
   executed.        */

*DMC "Square",BEGIN  // Accomplishes the same as the Sq macro.
SCREEN:USER          // The BEGIN and END commands allow the
INPUT X              // macro to be written over several lines.
Y=X**2
PPRINT X," squared is ", Y
END

*DMC "SQ",RETURN($1**2)   // Defines a function named SQ.
PRINT SQ 11               // Prints the result of the function
                          // named SQ (121).

*DMC "Av",RETURN(($1+$2+$3)/3)  // Defines a function named Av.
X=Av 13,14,17                   // Executes the function Av.
PPRINT X                        // Returns 14.66667.

*dmc "Sph_Sec_Surface",return(3.1415*$1/2*(4*$2+$3))
print Sph_Sec_Surface 4.4, 2.12, 6.7

/* The above *DMC command defines the function
   (3.1415*$1/2*(4*$2+$3)) as Sph_Sec_Surface.  104.9135 is
   printed on the screen.          */
```

*EMC (HOST Only)

USE To enable/disable macros or return Enable Macro Command status.

SYNTAX ***EMC** *b* or
***EMC?**

ELEMENT	DESCRIPTION
b	Enable ($b = 1$) or disable ($b = 0$).

ACTION Enables/disables execution of macros per value of b or returns the Enable Macro Command status. Returns 1 if macro execution is enabled; 0 otherwise.

SEE ALSO ***DMC**

*ESE (HOST Only)

USE To specify or return the contents of the Standard Event Status Enable Register.

SYNTAX ***ESE** *n* or
***ESE?**

ELEMENT	DESCRIPTION
b	Eight bit value. Range is 0 to 255.

ACTION Specifies (*n*) or returns the contents of the Standard Event Status Enable Register.

SEE ALSO Standard Event Status Register (2-16-3)

*ESR? (HOST Only)

USE To return the contents of the Standard Event Status Register.

SYNTAX ***ESR?**

ACTION Returns and clears the contents of the Standard Event Status Register.

SEE ALSO Status Subsystem (2-16)

EXAMPLES

```
*DMC "ERROR_CHECK",BEGIN
*ESE 61                       // Enable Std Event Status Register.
IF (*ESR & 1)= 0              // Check Operation Complete bit.
   IF (*ESR? & 4)=1           // If Standard Event Status Register
     PRINT "QUERY ERROR"      // reports Query Error, print message.
   ELIF (*ESR? & 8)=1         // If Standard Event Status Register
                              // reports Device Specific Error,

     PRINT "DEVICE SPECIFIC ERROR"      // print message.

   ELIF (*ESR? & 16)=1             // If Std Event Status Rgtr reports
     PRINT "EXECUTION ERROR"       // Execution Error, print message.
   ELIF (*ESR? & 32)=1             // If Standard Event Status
     PRINT "COMMAND ERROR"         // Register shows Command Error,
                                   // print message.
   ELSE                            // If Std Event Status Rgtr does
     PRINT "NO ERROR DETECTED"     // not report an error type, print
                                   // msg.
ENDIF
END
```

*IDN?

USE To return Test Set Identification.

SYNTAX ***IDN?**

ACTION Returns the identification parameters of the Test Set. Includes the manufacturer and model names, serial number and various firmware versions.

EXAMPLE

```
*IDN?
```

<u>An example return from a query to an **HOST** is as follows:</u>

```
IFR SYSTEMS,IFR-1900,4000,1.05/C-03.01S/F-03.02S/M-3.00S
```

The serial number is 4000. The system firmware version is 1.05. The Counter firmware version is 3.01S. The Function Generator firmware version is 3.02S. The Monitor firmware version is 3.00S.

<u>An example return from a query to an **Sp Tst** is as follows:</u>

```
IFR 1900CSA,200,01.05,Aug 20 1997 15:19:53
```

The serial number is 200. The system firmware version is 1.05 with compile date and time.

*LMC? (HOST Only)

USE To return a listing of all currently defined macros.

SYNTAX *LMC?

ACTION Returns a listing of all currently defined macros (to include predefined macros; see Appendix A).

SEE ALSO *EMC

*OPC (HOST Only)

USE To set the Operation Complete bit of the Standard Event Status Register when all pending operations are finished.

SYNTAX *OPC

ACTION Sets the Operation Complete bit of the Standard Event Status Register when all pending operations are finished.

SEE ALSO *WAI

*OPC? (HOST Only)

USE To generate an ASCII 1 in output queue when operation is completed.

SYNTAX *OPC?

ACTION Generates an ASCII 1 in the output queue when operation is complete, activating the Message Available bit in the Status Byte.

SEE ALSO Standard Event Status Register (2-16-3.)

*OPT? (HOST Only)

USE To return the value from Options of the Configuration Report.

SYNTAX *OPT?

ACTION Returns the value from Options of the Configuration Report (see 3-3-10 of the IFR-1900 Operation Manual). Range of return values is 0 to 255.

*PMC

USE To purge all macros and declared variables.

SYNTAX ***PMC**

ACTION Deletes all macros and declared variables except the predefined macros listed in Appendix A.

> HOST only: Global variables and array saved with the **NVSAV** command are not affected.

SEE ALSO Macros (2-9), **FORGET**

*RCL (HOST Only)

USE To restore HOST to routings and settings stored in memory location *n*.

SYNTAX ***RCL** *n*

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number. Range is 0 to 9.

ACTION Recall command. Restores device to the environment (routings and settings) stored in local memory location *n*.

SEE ALSO ***SAV**

*RST (HOST Only)

USE To reset HOST to factory defaults.

SYNTAX ***RST**

ACTION Reset command. Resets device to preset condition (factory defaults).

*SAV (HOST Only)

USE To save current routings and settings in memory location *n*.

SYNTAX ***SAV** *n*

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number. Range is 0 to 9.

ACTION Save command. Saves current environment (routings and settings) in local memory location *n*.

SEE ALSO ***RCL**

*SRE (HOST Only)

USE To specify or return the contents of the Service Request Enable register.

SYNTAX ***SRE** *n* or
***SRE?**

ELEMENT	DESCRIPTION
n	Eight bit value. Range is 0 to 255.

ACTION Service Request Enable command or query. Specifies or returns the contents of the Service Request Enable register.

SEE ALSO Status Byte (2-16-2)

*STB? (HOST Only)

USE To return the contents of the Status Byte register.

SYNTAX ***STB?**

ACTION Status Byte query. Returns the contents of the Status Byte register.

> The Status Byte register is not cleared when read, but maintains the current value until the ***CLS** command is used (***CLS** command also clears the condition and event registers of each Status Register).

SEE ALSO ***CLS**

EXAMPLES See **STATus:QUEStionable:INSTRument:ISUMmary**.

*TRG (HOST Only)

USE To execute trigger command.

SYNTAX ***TRG**

ACTION Trigger command. Executes trigger command as defined by the ***DDT** command.

SEE ALSO ***DDT**, (**SCOPe:TRIGger:SOURce** in Section 6-9)

*TST? (HOST Only)

USE To perform Self Test.

SYNTAX ***TST?**

ACTION Self-Test Query. Performs Self Test and returns 0 if passed; any non-zero number returned indicates Self Test failed.

SEE ALSO Individual Self Test Commands (Appendix C)

*WAI

USE To pause command execution until previous operations are complete.

SYNTAX ***WAI**

ACTION Stops command execution until all pending operations are complete.

> Use ***WAI** after **SCREEN** and **SETUP** commands and other commands involving routing changes or lengthy measurement times.

SEE ALSO **DELAY**

EXAMPLE

```
SCREEN:REC              // Displays the Receiver Screen.
*WAI                    // Stops macro execution until
                        // the screen display is completed.
RECEIVE:INPUT:TR        // Selects T/R Connector for Receiver Input.
*WAI                    // Stops command execution until
                        // the routing change is completed.
```

ABS

USE Absolute value.

SYNTAX **ABS**(*n*)

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number.

ACTION Returns the absolute value of *n*.

SEE ALSO **FLOOR, SIGN**

EXAMPLES

```
y=ABS(-4)               // y is assigned the value of positive 4.

Z=-7
POS=ABS(Z)              // POS is assigned the value of positive 7.
```

ACTIVATE

USE To activate a task for mulitasking.

SYNTAX **ACTIVATE** "*name*"

ELEMENT	DESCRIPTION
name	Name of the task to be loaded into the schedule queue.

ACTION Places a declared task on the schedule queue so the task can be multitasked.

> A macro must be declared a task using the **TASK** command before the macro can be put in the schedule queue using the **ACTIVATE** command. This must be done to each macro prior to being multitasked. Once execution of the macro begins, execution starts from the beginning of the macro.

SEE ALSO Multitasking Macros (2-12), **TASK, TPAUSE, TSTOP**

EXAMPLES See Multitasking examples (2-12).

AND

USE Logical AND.

SYNTAX *cmdA argument1* **AND** *argument2 cmdB* or
cmdA argument1 **AND** *argument2* **AND** *argument3* ... **AND** *conditionX cmdB*

ELEMENT	DESCRIPTION
cmdA	Any decision point command (**IF**, **ELIF**, **UNTIL**, **WHILE**).
argument1	Any expression that evaluates to TRUE, FALSE, 0 (zero) or any non-zero number (0 = FALSE, any non-zero number = TRUE).
argument2 on	Same as *argument1*.
cmdB	Any valid TMAC command.

ACTION Evaluates *argument1* and *argument2* and, if both are TRUE (or evaluated to a non-zero number), then the result of the AND operation is TRUE (FALSE, otherwise). If more than one AND operation in encountered then the AND operations are evaluated from left to right. Successive ANDs evaluate the result of previous AND operations along with individual expressions.

Table 3-4 lists the expression combinations and the AND operation results.

argument1	*argument2*	**AND Result**
False	False	False
False	True	False
True	False	False
True	True	True

Table 3-4 AND Operation Results

SEE ALSO **!, OR, =, !=, <, >, <=, >=, DO UNTIL, IF ..., WHILE**

EXAMPLES

```
var x=6, z=5
if x=6 AND x!=z        // Decision point command utilizing AND
                       // as condition with accompanying
                       // expressions to evaluate.  Condition is
                       // TRUE, because both expressions evaluate
                       // to TRUE.
  print y              // Performs resulting command since condition
                       // is TRUE.
ENDIF
```

ASC

USE ASCII value of character.

SYNTAX **ASC(***string***)**

ELEMENT	DESCRIPTION
string	Any string.

ACTION Returns the ASCII value of the first character of *string*.

SEE ALSO **CHR, LEN, STR, STRPOS, VAL**

EXAMPLES

```
VAR Ascii_value
STRING Char
Char="Example"
Ascii_value = ASC(Char)   // Ascii_value, a numeric variable,
                          // is assigned the ASCII value
                          // of E (67).
```

BCOLOR

USE To change background color.

SYNTAX **BCOLOR** *b*

ELEMENT	DESCRIPTION
b	Color number (see Table 2-2). Range of colors is color numbers 0 to 7.

ACTION Changes the background color to the color *b*.

Can use color name constants defined in Appendix A.

SEE ALSO **COLOR, EDIT:COLOR** commands

BEGIN

USE To allow macros to contain multiple lines.

SYNTAX

***DMC** "*name*", **BEGIN**
sequence
END

ACTION Combines with the **END** command to define the command lines in multiple line macros. The **BEGIN** command is not used with single line macros.

ELEMENT	DESCRIPTION
name	Name of the macro being defined.
sequence	Command sequence performed inside the macro.

SEE ALSO Macros (2-9), ***DMC**, **END**

EXAMPLES

```
*DMC "MPt",VAR Mid;Mid=(Freq1+Freq2)/2;NVSAV Mid,60;PRINT Mid
```

The above macro can be entered as:

```
VAR Mid
*DMC "MPt",BEGIN
Mid=(Freq1+Freq2)/2
NVSAV Mid,60
PRINT Mid
END
```

BOX

USE To create a user defined box on the color display.

SYNTAX **BOX** *f,x1,y1,x2,y2,c*

ELEMENT	DESCRIPTION
f	Creates the box solid or hollow. *f* is 1 for a solid box. *f* is 0 for the box to appear hollow.
x1, y1	Point signifying the top left corner of the box (0,0 to 639,349).
x2, y2	Point signifying the bottom right corner of the box (0,0 to 639,349).
c	Number (or constant name - see Appendix A) specifying the color of the box. See Table 2-2 for the available colors.

ACTION Creates a box using the points specified as the top left and bottom right corners. The box can be created hollow or solid.

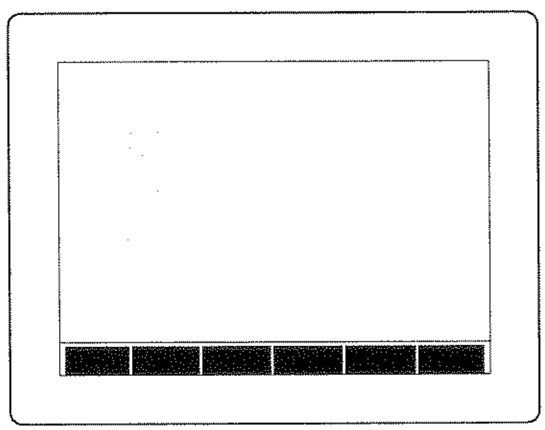
Text cannot be printed in boxes as in windows.

SEE ALSO Graphics (2-13-5)

EXAMPLES

```
BOX 1,150,50,200,250,7
```

The above command displays a light gray box on the color display as follows:



8618021

Figure 3-1 Box Example

CALCulate?

USE To calculate a function and return the result to the Host.

SYNTAX **CALCulate?** *argument*

ELEMENT	DESCRIPTION
argument	Mathematical expression or formula calculated.

ACTION The *argument* is calculated and returned through the active interface connector.

> This command cannot be used inside a macro.

SEE ALSO Mathematical Functions (2-7)

EXAMPLES

```
X=4,Y=3             // Initializes variables.
CALC? X*Y**3        // Calculates the mathematical
                    // expression and returns the result
                    // to the Host.  108 is returned.
```

CASE OF OTHERWISE ENDCASE

USE To provide a predetermined decision point inside a macro.

SYNTAX

CASE *variable*
   OF *value1*: *command*
   OF *value2*: *command*
   .
   .
   OTHERWISE: *command*
ENDCASE

ELEMENT	DESCRIPTION
variable	Expression compared with the values to decide which **OF** command to perform.
value	Expression compared with *variable*. If equal to *variable*, the command following *value* is performed.
command	Command performed if *value* is equal to *variable*.

ACTION Utilizes value of variable to select one of multiple possible preselected commands. There is no limit to the number of **OF** expressions, and the **OTHERWISE** expression is optional. The **OF** values are compared to the **CASE** *variable*. The *command* sequence following the **OF** value that equals the **CASE** *variable* is executed. If none of the **OF** values equal the **CASE** *variable*, the **OTHERWISE** *command*, if present, is executed, then the macro execution passes to the command following the **ENDCASE** command.

SEE ALSO Macro Decision Points (2-10), **IF**, **IF ELSE**, **IF ELIF**

EXAMPLES

```
*DMC "Val"
SCREEN:USER           // Selects the blank user screen.
CASE X
   OF 1: PRINT "X=1"
   OF 2: PRINT "X=2"
   OF 5: PRINT "X=5"
   OTHERWISE:PRINT "X unknown"
ENDCASE

/* X is compared to the OF values 1, 2 and 5.  If X is equal
   to one of the OF values, the command or commands following
   that OF value are executed.  For example: if X=2, the PRINT
   "X=2" command is executed, if X=6, the PRINT "X unknown"
   command is executed.                                    */


*DMC "Menu",BEGIN
SCREEN:USER
VAR Choice
PRINT "Press a DATA ENTRY Key to select a Test"
PRINT "     and press the ENTER Key."
PRINT "1. Test 1"
PRINT "2. Test 2"
PRINT "3. Test 3"
PRINT "Any other selection other than the ones listed"
PRINT "   displays the Receiver Operation Screen."
XY 100,250
INPUT Choice
CASE Choice
   OF 1: PRINT "Test_1 is running"
         DELAY 3000
         CLS
         Test_1
   OF 2: PRINT" Test_2 is running"
         DELAY 3000
         CLS
         Test_2
   OF 3: PRINT" Test_3 is running"
         DELAY 3000
         CLS
         Test_3
   OTHERWISE: PRINT "No Test Performed"
ENDCASE
SCREEN:REC
*WAI                     // Pauses macro execution until screen
                         // command is fully executed.
END
```

(Example is continued on next page.)

(Example is continued from previous page.)

```
/* The macro, "Menu," displays a user-created menu allowing
   the selection of one of three tests.  These tests are
   actually macros Test_1, Test_2 and Test_3 which are
   executed if 1, 2 or 3 is entered in response to the INPUT
   command.  If the entered data is not 1, 2 or 3, the
   OTHERWISE command is performed and "No Test Performed" is
   printed.  After a Test is executed or the PRINT command is
   performed, command execution resumes following the ENDCASE
   command.                                                  */
```

CENTER (Sp Tst Only)

USE To display text centered within a defined display area on the screen of the HOST.

SYNTAX **CENTER** *text*,*x*,*y*,*length*

ELEMENT	DESCRIPTION
text	String of characters and spaces designated within quotation marks.
x	Horizontal starting point in pixels of horizontal display area (0 to 639).
y	Vertical starting point in pixels of horizontal display area (0 to 259).
width	Width in pixels of horizontal display area.

ACTION Displays *text* centered within a horizontal display area. The starting and ending positions are defined by *x*,*y* and *width*.

SEE ALSO **HPRINT**, **LJPRINT**, **PRINT**, **RJPRINT**, **XYPRINT**

EXAMPLES

```
USER                            // Display blank User screen.
CENTER "Test No. 12",0,0,639    // Displays Test No. 12
                                // centered at top of screen.
```

CHR

USE Returns the character equivalent of ASCII number.

SYNTAX **CHR(*n*)**

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number. Range is from 0 to 255.

ACTION Returns the character equivalent of ASCII number *n*.

SEE ALSO **STR, VAL**

EXAMPLES

```
*pmc                         // Clears all macros in memory.
*dmc "ascii_test",begin
user                         // Displays blank User screen.
for n=0 to 255
$=chr(n)                     // $ is assigned ASCII character
                             // associated with the present value
                             // of i.
print $," ",n                // Prints current ASCII character and
                             // and value of n.
for j= 1 to 1000;next j      // Time delay.
next n
end
```

CLS

USE To erase the contents of a window or the color display.

SYNTAX **CLS**

ACTION Clears the currently selected window. To clear the entire color display, select window 0 (the background) using the **WSEL** command.

SEE ALSO **WSEL**

COLOR

USE To specify or return foreground text color.

SYNTAX
COLOR *f* or
COLOR *f,b* or
COLOR?

ELEMENT	DESCRIPTION
f	Foreground text color (see Table 2-2).
b	Optional background contrast color (see Table 2-2).

ACTION Specifies or returns current foreground text color. An optional background contrast color may be included. If excluded, the present background color is assumed.

> The actual color of text printed on the screen of the HOST is determined by the following formula:
>
> **Actual Foreground Text color value** = {[selected foreground text color (*f*) *-xor'ed with-* selected or assumed background contrast color (*b*)] *-xor'ed with-* current background color (see **BCOLOR**)]}. See Table 2-2 for colors and color values.

SEE ALSO **BCOLOR, EDIT:COLOR** commands

EXAMPLES

```
screen:user            // Displays blank user screen (for Sp Tst
                       // use USER).
cls                    // Initializes screen.
bcolor 1               // Selects Dark Blue for background.
color yellow,1         // Using predefined constant, selects
                       // Yellow as foreground text color and
                       // Dark Blue as background contrast color
                       // (could have used dark_blue predefined
                       // constant).
print "This is Yellow!"  // Print Yellow text on Dark_Blue
                         // screen (for Sp Tst use HPRINT).
```

CONST

USE
To define constants.

SYNTAX
CONST *name, argument* or
CONST *name, argument, name, argument, ... ,name, argument*

ELEMENT	DESCRIPTION
name	Name of the constant defined. The first character of the *name* is a letter while the remaining characters can be letters, digits and the underscore. The length of *name* must be from 2 to 31 characters.
argument	Any number or any expression that evaluates to a number.

ACTION
Declares constants and associated values. Constants are global and cannot be declared inside a macro. String data cannot be assigned to a constant. Constant values can only be changed using another **CONST** command.

SEE ALSO
Constants and Data Arrays (2-4), **DATA**, **NVRCL**, **NVSAV**

EXAMPLES
```
CONST PI,3.1415,Wid,X/4.3   // Defines PI as a constant and
                            // gives it a value of 3.1415.
                            // Defines Wid as a constant,
                            // evaluates X/4.3 and assigns this
                            // value to Wid.
```

COS

USE
To return the cosine of an angle.

SYNTAX
COS(*n***)**

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number.

ACTION
Returns the cosine of an angle (*n*) which is measured in radians.

SEE ALSO
SIN

EXAMPLES
```
VAR Cosine_of_radians
Cosine_of_radians=COS(5*3.1415/6)   // Cosine_of_radians now
                                    // holds the cosine of 5π/6
                                    // (≈-0.866).
```

DATA

USE To define constant data arrays.

SYNTAX **DATA** *name={argument, argument, ... ,argument}*

ELEMENT	DESCRIPTION
name	Name of the constant defined. The first character of the *name* is a letter while the remaining characters can be letters, digits and the underscore. The length of the *name* must be from 2 to 31 characters.
argument	Any number or any expression that evaluates to a number.

The first *argument* is assigned to the first index (name[0]) and so on.

ACTION Declares data arrays and associated values. A data array is an array of constants. Data arrays are global and cannot be declared inside a macro. String data cannot be assigned to a data array. Data arrays values can only be changed using another **DATA** command. Data array values are defined by including the order of occurrence in braces (curly brackets). Consequently, a value in the data array is retrieved by specifying the position of the data array value with an index in brackets following the name. This order begins with 0 so that Stat[3] returns the fourth constant in the data array Stat.

SEE ALSO Constants and Data Arrays (2-4), **CONST**, **ICON**, **NVRCL**, **NVSAV**

EXAMPLES

```
DATA STAT={2,34.7,2.874836593,336/4}
X=STAT[1]              // Assigns 34.7 to X.
Z=STAT[0]              // Assigns 2 to Z.
Y=STAT[4]              // Error!  STAT[4] is not defined.
```

DELAY

USE To delay command execution.

SYNTAX **DELAY** *t*

ELEMENT	DESCRIPTION
t	Any constant, variable, number or any expression that evaluates to a number.

ACTION Provides a time delay of length *t* in ms before the next command is executed. Delays are used to allow physical changes enough time to occur before command execution continues. Such changes could be: code generation, routing changes and displaying Operation Screens.

SEE ALSO Macros (2-9), ***WAI**

EXAMPLES

```
SCREEN:GEN             // Displays HOST Generator
                       // Operation screen.
*WAI                   // Waits until all pending operations
                       // are complete.
GEN:DCS:INV 311
DELAY 2000             // Delays the execution of the
                       // following command 2 sec to allow
                       // time for the DCS Code generation.
GEN:DCS:STOP
```

DO UNTIL

USE

To perform a set of commands repeatedly until a specified condition occurs (becomes true).

SYNTAX

DO
sequence
UNTIL *condition*

ELEMENT	DESCRIPTION
condition	Conditional expression evaluated.
sequence	One or more valid commands or expressions.

ACTION

The **DO** loop performs *sequence* **once**, then evaluates *condition*. If *condition* is FALSE, then command execution returns (loops) to the beginning of the **DO** loop to again perform *sequence* and so on. If *condition* is evaluated TRUE, then command execution passes on to the following command, leaving the **DO** loop.

If the *condition* does not eventually become true, the **DO** loop continues to loop until power is cycled.

SEE ALSO

FOR, **WHILE**, For **SYSTEM** Commands, see 2-14.

EXAMPLES

```
*DMC "Count",BEGIN    // Defines a macro named Count.
N=0                   // Assigns 0 to N.
DO                    // Starts DO loop.
   N = N + 1          // Increments N by 1.
   PRINT N            // Prints current value of N.
UNTIL N=2             // Loops back to first command after DO
                      // command until N equals 2.
END                   // End of macro Count.
```

(Examples continued on next page.)

(Examples continued from previous page.)

```
*DMC "Compute",RETURN(FLOOR((50-$1)/4+50))
                              // Defines a function named Compute
                              // with $1 as placeholder to pass a
                              // value into the function.
*DMC "Compute_x",BEGIN        // Defines a macro named Compute_x.
X = 6                         // Assigns 6 to X.
DO                            // Starts DO loop.
   X = Compute X              // Assigns X the result of the
                              // function Compute.  The old value
                              // of X is passed into function.
PRINT X                       // Prints new X.
UNTIL X=50                    // Loops back to first command after
                              // the DO command until X equals 50.
END                           // End of macro Compute_x.

/* The following is a description of the above macro:

   o  X is set to 6, then enters the DO loop.

   o  The Compute function is calculated and sets X equal
      to 61.  X is printed.

   o  The condition (X = 50) is evaluated.  The condition is
      FALSE; command execution loops to first command after
      DO command.

   o  The Compute function is calculated and sets X equal
      to 47.  X is printed.

   o  The condition (X = 50) is evaluated.  The condition is
      FALSE; command execution, again, loops to first command
      after DO command.

   o  The Compute function is calculated and sets X equal
      to 50.  X is printed.

   o  The condition (X = 50) is evaluated.  The condition is
      TRUE; command execution passes on to the command after
      the UNTIL, leaving the DO loop.  Macro execution ends. */


/* The conditional expression does not have to involve a
   variable manipulated inside the loop.                     */

DO
   commands
   .
   .
UNTIL SYSTEM:KEY? > 0         // Continues looping until a Front
                              // Panel Key is pressed.
```

DRAW

USE To create a user-defined line on the color display.

SYNTAX **DRAW** *x1,y1,x2,y2,c*

ELEMENT	DESCRIPTION
x1,y1	Both either a constant, variable, number or any expression that evaluates to a number. Ranges are 0 to 639 and 0 to 349.
x2,y2	Both either a constant, variable, number or any expression that evaluates to a number. Ranges are 0 to 639 and 0 to 349.
c	Any constant, variable, number, string or any expression that evaluates to a number or string. Number specifying color of line. See Table 2-2 for colors and associated values.

ACTION Creates a line of color *c* starting at point *x1,y1* and ending at point *x2,y2.*

SEE ALSO Graphics (2-13-5), **BOX**, **ELLIPSE**, **ICON**

EXAMPLES See Graphic examples (2-13-5).

EDIT:COLOR:LETTER (HOST Only)

USE To change or return the color of text of all built-in menus.

SYNTAX **EDIT:COLOR:LETTER** *c* or
EDIT:COLOR:LETTER?

ELEMENT	DESCRIPTION
c	Any constant, variable, number, string or any expression that evaluates to a number or string. Color or numerical value of menu text color (see Table 2-2).

ACTION Changes or returns the current color of text characters of all built-in menus of the HOST.

SEE ALSO Other **EDIT:COLOR** commands, **EDIT:WIDTH**

EXAMPLES

```
edit:color:letter 15 // Changes the text color of all
                     // built-in menus to White.
```

EDIT:COLOR:MENU (HOST Only)

USE
To change or return background color of all built-in menus.

SYNTAX
EDIT:COLOR:MENU *c* or
EDIT:COLOR:MENU?

ELEMENT	DESCRIPTION
c	Any constant, variable, number, string or any expression that evaluates to a number or string. Color or numerical value of menu background color (see Table 2-2).

ACTION
Changes or returns the current background color of all built-in menus of the HOST.

SEE ALSO
Other **EDIT:COLOR** commands, **EDIT:WIDTH**

EXAMPLES
```
edit:color:menu 6       // Changes the background color of all
                        // built-in menus to Brown.
```

EDIT:COLOR:SOFT:BOX (HOST Only)

USE
To change or return background color of all built-in Soft Function Keys.

SYNTAX
EDIT:COLOR:SOFT:BOX *c* or
EDIT:COLOR:SOFT:BOX?

ELEMENT	DESCRIPTION
c	Any constant, variable, number, string or any expression that evaluates to a number or string. Color or numerical value of Soft Function Key background color (see Table 2-2).

ACTION
Changes or returns current background color of all built-in Soft Function Keys of the HOST.

SEE ALSO
Other **EDIT:COLOR** commands, **EDIT:WIDTH**

EXAMPLES
```
edit:color:soft:box 2     // Changes the background color of
                          // all built-in Soft Function Keys to
                          // Dark Green.
```

EDIT:COLOR:SOFT:LETTER (HOST Only)

USE To change or return the text color of all built-in Soft Function Keys.

SYNTAX **EDIT:COLOR:SOFT:LETTER** *c* or
EDIT:COLOR:SOFT:LETTER?

ELEMENT	DESCRIPTION
c	Any constant, variable, number, string or any expression that evaluates to a number or string. Color or numerical value of Soft Function Key text color (see Table 2-2).

ACTION Changes or returns current color of the text of all built-in Soft Function Keys of the HOST.

SEE ALSO Other **EDIT:COLOR** commands, **EDIT:WIDTH**

EXAMPLES

```
edit:color:soft:letter 15    // Changes the color of the text of
                             // all built-in Soft Function Keys
                             // to White.
```

EDIT:COLOR:SOFT:SELECT (HOST Only)

USE To change or return the text color of all built-in Soft Function Keys when selected.

SYNTAX **EDIT:COLOR:SOFT:SELECT** *c* or
EDIT:COLOR:SOFT:SELECT?

ELEMENT	DESCRIPTION
c	Any constant, variable, number, string or any expression that evaluates to a number or string. Color or numerical value of selected Soft Function Key text color (see Table 2-2).

ACTION Changes or returns current text color of built-in Soft Function Keys when selected.

SEE ALSO Other **EDIT:COLOR** commands, **EDIT:WIDTH**

EXAMPLES

```
edit:color:soft:select 4     // Changes the color of the text
                             // (when selected) of all built-in
                             // Soft Function Keys to Dark Red.
```

EDIT:WIDTH (HOST Only)

USE

To specify the width of the input box that appears with the **INPUT** command.

SYNTAX

EDIT:WIDTH *n*

ELEMENT	DESCRIPTION
n	Any constant, variable, number, or any expression that evaluates to a number.

ACTION

Specifies the width in *n* pixels of the input box that appears on the screen of the HOST when the **INPUT** command is executed.

> Use a factor of 19 for each numerical character (including the decimal point) expected in input box; use a factor of 26 for each alphanumeric (string) character expected in input box.

SEE ALSO

EDIT:COLOR commands, **INPUT**

EXAMPLES

```
edit:width 38          // Changes the width of the input box
                       // on the screen in response to a
                       // INPUT command to accept two
                       // numeric characters.

edit:width 78          // Sizes input box to accept
                       // 3 alphanumeric (string) characters.
```

ELLIPSE

USE

To draw an ellipse

SYNTAX

ELLIPSE *b,x,y,r,a,c*

ELEMENT	DESCRIPTION
b	Any constant, variable, number, or any expression that evaluates to a number. Range is 0 to 1.
x,y	Both either a constant, variable, number or any expression that evaluates to a number. Ranges are 0 to 639 and 0 to 349.
r	Any constant, variable, number, or any expression that evaluates to a number.
a	Any constant, variable, number, or any expression that evaluates to a number. Range is 0.1 to ≈30.
c	Any constant, variable, number, string or any expression that evaluates to a number or string. Color name or associated number (see Table 2-2).

ACTION

Draws an ellipse centered at *x,y* with radius *r* in pixels and aspect ratio *a*. Ellipse is drawn in color selected for *c*. If *b* is 1, the ellipse is solid. If *b* is 0, the ellipse is hollow. An aspect ratio of 0.75 creates a circle. An aspect ratio > 0.75 creates an ellipse that has greater height than width, and an aspect ratio < 0.75 creates an ellipse that has greater width than height. Parameters *x*, *y* and *r* are measured in pixels.

The height and width of any ellipse created by this command are found using the following equations:

WHERE:	WIDTH	HEIGHT
$0.1 \le a \le 1$	2r	8ra/3
$a > 1$	2r/a	8r/3
Due to the pixel alignment, an ellipse which has a height to width ratio of 4:3 (or a height 1/3 greater than the width) has an aspect ratio equal to 1 (a = 1).		

Table 3-5 Ellipse Width to Height Equations

SEE ALSO

Graphics (2-13-5), **BOX, DRAW, ICON**

EXAMPLES

```
ellipse 1,320,175,50,0.375,dark_green  // Creates a solid green
                                       // ellipse that is
                                       // 50 pixels high by
                                       // 100 pixels wide in
                                       // the center of the
                                       // screen.

ellipse 0,320,175,75,1.5,dark_red // Creates a hollow red
                                  // ellipse that is 200 pixels
                                  // high by 100 pixels wide
                                  // in the center of the
                                  // screen.
```

END

USE To allow a macro to contain multiple lines.

SYNTAX ***DMC** "*name*", **BEGIN**
sequence
END

ELEMENT	DESCRIPTION
name	Name of macro being defined.
sequence	Command sequence performed inside macro.

ACTION When used with **BEGIN**, the **END** command defines the end of a multiple line macro.

SEE ALSO Macros (2-9), ***DMC**, **BEGIN**

EXAMPLES

```
*dmc "Print_x",BEGIN       // Defines a macro named Print_x.
INPUT X                    // Receives a value from the
                           // keyboard.
PRINT X                    // Prints the received value.
END                        // End of the macro Print_x.
```

ERASE:TEXT (Sp Tst Only)

USE To erase a defined display area.

SYNTAX **ERASE:TEXT** *x,y,width*

ELEMENT	DESCRIPTION
x,y	Both either a constant, variable, number or any expression that evaluates to a number. Ranges are 0 to 639 and 0 to 349.
width	Any constant, variable, number or any expression that evaluates to a number. Range is 1 to 640

ACTION Erases display area defined by an *x,y* starting point and *width* in pixels.

EXEC

USE To execute a macro at a specified address.

SYNTAX **EXEC** *address* or
EXEC **&***name*

ELEMENT	DESCRIPTION
address	Any constant, variable, number or any expression that evaluates to a number.
name	Name of the macro to be executed.

ACTION Executes macro *name* located at the *address* specified. An **&** (ampersand) before the *name* of the macro, as shown, returns the address of the macro. Macros requiring variables to be passed when the macro's *name* is entered cannot be executed using the **EXEC** command. Unpredictable results occur when using an invalid *address*.

> The **EXEC** command can be used to call out undefined macros by simply declaring a variable and using the variable as the address.

SEE ALSO Macros (2-9)

EXAMPLES

```
ADD=&First              // The address of the macro First
                        // is loaded into the variable ADD.
EXEC ADD                // The macro located at the address
                        // loaded in the variable ADD is executed.

EXEC &First             // Macro First is executed.
```

EXP

USE To calculate the result of e^n

SYNTAX **EXP(***n***)**

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number. Range is from -102.585781 to 88.72283554.

ACTION Returns the result of e^n.

SEE ALSO **LN**, **LOG**

EXAMPLES

```
VAR Expo
x=1.57
Expo=EXP(x)             // e to the 1.57th power (4.806648) is
                        // assigned to Expo.
```

FALSE

USE To produce 0.

SYNTAX **FALSE** or
OFF

ACTION Provides a concise method of specifying a logical 0.

SEE ALSO **!**

EXAMPLES

```
Y=OFF                    // Y is assigned the value of 0.
```

FLOOR

USE To return the truncated value of *n*.

SYNTAX **FLOOR(*n*)**

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number.

ACTION Returns the truncated value of *n*, leaving only the integer portion to the left of the decimal point.

SEE ALSO **ABS, SIGN**

EXAMPLES

```
VAR Round_off
Round_off = FLOOR(5.99)   // Assigns the truncated value
                          // of 5.99 (5) to Round_off.
```

FLUSH

USE To return pending responses.

SYNTAX **FLUSH**

ACTION The **FLUSH** command returns all pending return data upon execution.

> Normally, without the **FLUSH** command, returning data from the queries in a macro is collected in the output queue of the Test Set. The contents of the output queue is returned at the end of any loops present, at the end of the macro or when the queue becomes full (the output queue is 2 kilobytes in size).

SEE ALSO Macros (2-9)

EXAMPLES

```
*DMC "Without_FLUSH",BEGIN // Defines a macro named
                           // Without_FLUSH.
SCREEN:REC                 // Displays the Receiver Operation
                           // Screen.
*WAI                       // Waits for the Receiver Operation
                           // Screen to be completely displayed.
REC:FREQ?                  // Queries the Receiver Frequency.
M_RF?                      // Queries the Frequency Error Meter
                           // reading.
M_PWR?                     // Queries the Power Meter reading.
END                        // Ends macro Without_FLUSH.

/*  The returned data in the above macro without using the
    FLUSH command appears as follows:

       100320,100318,62                                    */

*DMC "With_FLUSH",BEGIN  // Defines a macro named With_FLUSH.
SCREEN:REC               // Displays the Receiver Operation
                         // Screen.
*WAI                     // Waits for the Receiver Operation
                         // Screen to be completely displayed.
REC:FREQ?                // Queries the Receiver Frequency.
FLUSH                    // Returns impending responses.
M_RF?                    // Queries the Frequency Error Meter
                         // reading.
FLUSH                    // Returns impending responses.
M_PWR?                   // Queries the Power Meter reading.
END                      // Ends macro With_FLUSH.

/* The returned data in the above macro _with_ the FLUSH
   commands appears as follows:

       100320
       100318
       62                                                  */
```

FOR NEXT

USE

To perform a set of commands in a loop a given number of times within a macro.

SYNTAX

FOR *variable* = *initial* **TO** *ending* **STEP** *step*
sequence
NEXT *variable*

ELEMENT	DESCRIPTION
variable	Any variable.
initial	Any constant, variable or number or any expression that evaluates to a number.
ending	Any constant, variable or number or any expression that evaluates to a number.
step	Any constant, variable or number or any expression that evaluates to a number.
sequence	Any valid single command or command combination.

ACTION

Performs a command *sequence* the number of times required to increment *variable*, an amount equal to *step*, from an *initial* to an *ending* value.

The following steps are performed:

1. **FOR** command sets *variable* equal to *initial*.
2. Command *sequence* is executed.
3. **NEXT** command increments *variable* by the value *step*.
4. The value of *variable* is evaluated:
   a. If *step* is a positive number **and** the current value of *variable* ≤ *ending*, macro execution returns to Step 2 above.
   b. If *step* is a positive number **and** the current value of *variable* is greater than *ending*, macro execution continues with the command following the **NEXT** command, leaving the **FOR .. NEXT** loop.
   c. If *step* is a negative number **and** the current value of *variable* ≥ *ending*, macro execution returns to Step 2 above.
   d. If *step* is a negative number **and** the current value of *variable* is less than *ending*, macro execution continues with the command following the **NEXT** command, leaving the **FOR .. NEXT** loop.
   e. If *step* increments *variable* such that the difference between *ending* and *variable* is greater than when assigned the value of *initial*, macro execution continues with the command following the **NEXT** command, leaving the **FOR .. NEXT** loop.

> If optional **STEP** and *step* are left off, the default *step* is +1 (even if *initial* and/or *ending* are negative).

> Setting *step* equal to 0, causes a **FOR .. NEXT** loop to loop indefinitely and requires cycling power of the Test Set to stop execution of the loop.

SEE ALSO **DO, WHILE**

EXAMPLES

```
for N=1 to 3
   command1
next N
/* commands following */

/* The following sequence occurs:

   o  N is set equal to 1.

   o  Command1 is executed.

   o  N is incremented by 1 (N=2) (by default, N is incre-
      mented by +1 when step is omitted) and compared to
      ending (3).

   o  Command1 is executed.

   o  N is incremented by 1 (N=3) and compared to ending (3).

   o  Command1 is executed.

   o  N is incremented by 1 (N=4) and compared to ending (3).

   o  Macro execution exits FOR .. NEXT loop and proceeds
      with the commands following the NEXT command.        */

for A=10 to 4 step -3
   print A
next A
print 2*A

/* The following sequence occurs:

   o  A is set equal to 10.

   o  A is printed (10).

   o  -3 is added to A (A = 7) and A is compared to
      ending (4).

   o  A is printed (7).

   o  -3 is added to A (A = 4) and A is compared to
      ending (4).

   o  A is printed (4).

   o  -3 is added to A (A = 1) and A is compared to
      ending (4).

   o  2*A is printed (2) as command execution proceeds below
      the NEXT command.                                      */
```

(Examples continued on next page.)

(Examples continued from previous page.)

```
FOR MM = -1 TO -6 STEP -2
   PRINT MM
NEXT MM
```

```
/* -1, -3 and -5 is printed.  When MM is incremented to -7,
macro execution exits the FOR .. NEXT loop, because -7 is
beyond the ending value.                                  */
```

```
FOR N=1 TO 4 STEP -1
   PRINT N
NEXT N
```

```
/* N is set equal to 1 and printed once.  N is then incre-
mented by -1 and is now equal to 0.  Because the difference
between ending and N (4 - 0 = 4) is greater than when N was
assigned the value of initial (4 - 1 = 3), command execution
proceeds below the NEXT command.                            */
```

FORGET

USE

To delete a macro from memory.

SYNTAX

FORGET *name*

ELEMENT	DESCRIPTION
name	String variable or literal string surrounded by quotation marks (e.g., "macro_1a").

ACTION

Deletes the macro specified by *name*.

> All macros and variables declared after the specified macro was declared are also deleted.

SEE ALSO

***PMC**

EXAMPLES

```
*DMC "FIRST",BEGIN
   PRINT "first macro"
END
*DMC "SECOND",BEGIN
   PRINT "second macro"
END
VAR TO_BE_ERASED              // Variable TO_BE_ERASED is declared.
FORGET "SECOND"               // Macro SECOND and variable
                              // TO_BE_ERASED are deleted from
                              // memory.  Macro FIRST still resides
                              // in memory.
```

FORMat

USE To specify the form of the returned data for queries.

SYNTAX
FORMat BINary or
FORMat OCTal or
FORMat HEXadecimal or
FORMat ASCii (HOST Only) or
FORMat DECimal (Sp Tst Only)

ACTION Specifies form of the returned data for queries. The four forms are Binary (base 2), Octal (base 8), Hexadecimal (base 16) and ASCII (HOST) or Decimal (Sp Tst) (base 10). Default is base 10. For the HOST, all returned data (other than that in ASCII format) is preceded with a prefix signifying the format: #B for Binary, #Q for Octal and #H for Hexadecimal.

> Data in scientific notation is unaffected by these commands.

SEE ALSO Numeric Formats (2-5-2)

EXAMPLES

```
REC:FREQ 103.7 MHZ    // Sets Receiver Frequency to 103.7 MHz.
FORM BIN              // Selects binary format for the
                      // returned data.
REC:FREQ?             // Queries the Receiver Frequency.
                      // #B11001010100010100 is returned.
FORM HEX              // Selects Hexadecimal format for
                      // the returned data.
REC:FREQ?             // Queries the Receiver Frequency.
                      // #H19514 is returned.
```

GPIB:ADDRess (Sp Tst Only)

USE To specify the GPIB address.

SYNTAX **GPIB:ADDRess** *address*

ELEMENT	DESCRIPTION
address	Any constant, variable, integer or any expression that evaluates to a integer. Range is 0 to 31.

ACTION Specifies the GPIB *address*.

SEE ALSO **GPIB:MASK**, **GPIB:SRQ**

EXAMPLES See **GPIB:SRQ**.

GPIB:MASK (Sp Tst Only)

USE To sets the SRQ interrupt mask.

SYNTAX **GPIB:MASK** *n*

ELEMENT	DESCRIPTION
n	Any constant, variable, integer or any expression that evaluates to a integer. 8 bit value with a range of 0 to 255.

ACTION Sets SRQ interrupt mask. The corresponding user defined Status Byte bit is masked unless the mask bit is active (i.e., set to 1).

SEE ALSO **GPIB:ADDRess, GPIB:SRQ**

EXAMPLES See **GPIB:SRQ**.

GPIB:SRQ (Sp Tst Only)

USE To enable a user-defined Status Byte bit to trigger a Service Request.

SYNTAX **GPIB:SRQ** *n*

ELEMENT	DESCRIPTION
n	Any constant, variable, integer or any expression that evaluates to a integer. 8 bit value with a range of 0 to 255.

ACTION Sets a user-defined Status Byte bit to trigger a Service Request. The corresponding mask bit must be active to generate an SRQ.

> The **GPIB:SRQ** command is used in decision point commands (**IF**, **WHEN**, **DO .. UNTIL**, etc.) to trigger a Service Request when a user-defined event occurs (e.g., cellular phone message capture).

SEE ALSO **GPIB:ADDRess, GPIB:MASK**

EXAMPLES

```
GPIB:ADDR 30;MASK 32                  // Set address and unmask
                                      // bit 5.
FOCC:SET;CHAN 333;STAR                // Set up and monitor FOCC
                                      // channel 333.
FOCC:CAPT:MIN "316/522-4981"          // Set to capture specified
                                      // MIN.
IF FOCC:CAPT?=1 GPIB:SRQ 32;ENDIF     // If capture occurs, send SRQ.
```

HEIGHT

USE To change height of text printed on color display.

SYNTAX **HEIGHT** *n*

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number. Range is 1 to 4.

ACTION The height of text printed on the color display after this command is changed to *n* times the standard height (0.175 inches). Text can be printed in four heights: 0.175, 0.35, 0.525 or 0.7 inches.

SEE ALSO Graphics (2-13-5), **PRINT**, **ROTATE**

EXAMPLES

```
HEIGHT 1                        // Sets the text height to 0.175''.
PRINT "standard"                // PRINT command.
PRINT                           // Each PRINT command moves the xy
PRINT                           // position down a line the current
PRINT                           // height set by the HEIGHT command
                                // (currently 0.175'').
HEIGHT 4                        // Sets the text height to 0.7''.
PRINT "large"                   // PRINT command.
HEIGHT 2                        // Sets the text height to 0.35''.
PRINT "double standard"         // PRINT command.
HEIGHT 1                        // Returns the HEIGHT setting to 1.
```

The following is printed on the color display: 1) `standard` in 0.175 inch high text, 2) `large` in 0.7 inch high text and 3) `double standard` in 0.35 inch high text.

Each **PRINT** command without an ending comma moves the xy position to the beginning of the next line using the height setting for the height of the line. The first four **PRINT** commands move the xy position down 0.7" (0.175" for each PRINT command). The **PRINT** "large" command moves the xy position down 0.7 inches. The **PRINT** "double standard" command moves the xy position down 0.35 inches.

The **HEIGHT** setting should be returned to 1 because all printing on the color display uses this setting.

HFLUSH (Sp Tst Only)

USE To quickly send **HOST** commands through the internal SCSI Bus from the Sp Tst to HOST.

SYNTAX **HFLUSH**

ACTION Gives previous **HOST** commands priority by flushing all commands currently in the task queue out the SCSI at that time. (Normally, to avoid time consuming SCSI traffic, **HOST** commands are held in the task queue. When the task queue becomes full, commands are sent across the internal SCSI Bus to the HOST.)

EXAMPLES

```
host "fgen:gen1:modl 5;shape:sin"
host "fgen:gen1:freq 1000;mod:fm
hflush
```

HPRINT (Sp Tst Only)

USE

To print on the display screen.

SYNTAX

HPRINT *argument* or
HPRINT *argument, argument, ... , argument,* or
HPRINT %0*nl, argument, ... ,argument,* or
HPRINT *argument, ... ,%0nl, argument,*

ELEMENT	DESCRIPTION
argument	Expression to be printed. Can be a mathematical expression, a literal string or the contents of variables. Mathematical expressions are calculated and string functions are performed before the results are printed.
%	Indicates that formatting information follows which pertains to the *arguments* that follows. Must be followed by an associated *argument*. Must precede any combination of 0, *n* and *l*.
0	Zero signifies leading zeros are added to fill the field width. 0 is omitted for no leading zeros. When used, must be accompanied by *n*. Applies only to numeric expressions.
n	Specifies field width. *n* is the minimum number of digit spaces (decimal point counts as one). Applies only to numeric expressions. If *l* is D, *n* can be in the form of *b.a*, where *b* designates minimum number of total digit spaces (decimal point counts as one) and *a* designates the maximum number of digits after the decimal point.
l	Specifies the numeric base the data is printed in. Applies only to numeric expressions. Omit for a default of Signed Decimal or enter one of the following: B Binary Q Octal H Hexadecimal U Unsigned decimal D Signed decimal

ACTION

Prints *argument* on the display screen. The **HPRINT** command (**HOST PRINT** command) prints numerical and string values at the current xy location on the color display. Several values are printed on the same line by separating them with commas. If the optional ending comma is used, the next **HPRINT** command continues on the same line; otherwise a new line is started. If an *argument* is a variable, the value of the variable is printed. Mathematical expressions are calculated before printing.

%0nl contains the format settings for numeric expressions and can be placed anywhere in the **HPRINT** command. There can be several format settings in each **HPRINT** command, each one changing the leading zero format, the numeric base and the field width for the expressions that follow. Format settings do not affect the printing of strings.

SEE ALSO **PRINT**, **XYPRINT**, **PPRINT**, **PSCREEN**, **XY**

EXAMPLES

```
HPRINT "CELLULAR PHONE TEST"     // Prints the string text,
                                 // between the quotation
                                 // marks, on the color
                                 // display, starting at the
                                 // current xy location.
```

ICON (HOST Only)

USE To create a user-defined graphic on the color display.

SYNTAX **ICON** *pixels,rows,name*

ELEMENT	DESCRIPTION
pixels	Any constant, variable, number or any expression that evaluates to a number.
rows	Any constant, variable, number or any expression that evaluates to a number.
name	Name of data array containing the bit pattern used for the icon.

ACTION Creates a user-defined graphic image. The icon is created by defining an area of pixels (consisting of *rows* of pixels and *pixels* per row) and a data array specifying with corresponding bits which pixels to turn on and which to turn off (1's to turn on, 0's to turn off). Pixels appear in the current foreground color. The top left corner of the defined area is placed at the current **XY** setting. The data array must consist of 32 bit words.

SEE ALSO **COLOR**, Graphics (2-13-5), **XY**, **DATA**

EXAMPLES See Graphic examples (2-13-5).

IDATA

USE Same as **DATA** except uses integer values only.

SYNTAX **IDATA** *name = {argument, argument, ... ,argument}*

SEE ALSO **DATA**

IF ENDIF

USE

To provide a decision point inside a macro.

SYNTAX

IF *condition sequence;* **ENDIF**
or
IF *condition*
sequence
ENDIF

ELEMENT	DESCRIPTION
condition	Any expression which is TRUE or FALSE when evaluated.
sequence	Any valid TMAC command or, if using the 2nd syntax example above, a series of commands that is performed if *condition* is evaluated as TRUE.

Zero (0) is evaluated by TMAC as FALSE. Any non-zero number is evaluated by TMAC as TRUE.

ACTION

Performs *sequence* if *condition* is evaluated as TRUE. If the condition is FALSE, *sequence* is not executed and execution flow passes to the command following **ENDIF**.

Every **IF** command must be followed by an **ENDIF**. **IF** commands can be nested within other **IF** commands if the **ENDIF** to the inner **IF** command is also inside the outer **IF** command and so on.

SEE ALSO

IF ELSE ENDIF, IF ELIF ELSE ENDIF, CASE

EXAMPLES

```
IF Mod>20 Flag=1;ENDIF      // Sets Flag = 1 if Mod is greater
                            // than 20.  Does nothing if Mod is
                            // not greater than 20.

IF Dev > 3                  // The following IF ENDIF is executed
                            // if Dev is greater than 3.
  IF Freq_Err < 1000
    Freq=200000             // The command Freq=200000 is command
                            // is executed if Freq_Err is less
                            // than 1000 and Dev is greater
                            // than 3.

  ENDIF
ENDIF
```

IF ELIF ELSE ENDIF

USE To provide multiple decision points inside a macro.

SYNTAX

IF *condition*
   sequence
ELIF *condition*
   sequence
ELIF *condition*.
   sequence
   .
   .
ELSE
   sequence
ENDIF

ELEMENT	DESCRIPTION
condition	Any expression which is TRUE or FALSE when evaluated.
sequence	Any valid TMAC command or a series of commands that is performed if *condition* is evaluated as TRUE.

Zero (0) is evaluated by TMAC as FALSE. Any non-zero number is evaluated by TMAC as TRUE.

ACTION Same as **IF ENDIF** command, except the **IF ELIF** command allows multiple *conditions* to be evaluated.

Each **ELIF** (Else If) part allows the checking of a *condition*. If the *condition* is TRUE, the *sequence* immediately following is executed. The optional **ELSE** *sequence* is executed if no **ELIF** *condition* is TRUE. An **ENDIF** command must be included at the end of an **IF ELIF** group of commands. If two **ELIF** commands are TRUE, only the *sequence* following the first TRUE **ELIF** encountered is executed.

SEE ALSO **IF ENDIF, IF ELSE ENDIF, CASE**

EXAMPLES

```
IF X=3
  PRINT "X=3"          // If X=3, this command is executed and
                       // the rest are skipped.
ELIF X=4
  PRINT "X=4"          // If X=4, this command is executed and
                       // the rest are skipped.
ELIF X=5
  PRINT "X=5"          // If X=5, this command is executed and
                       // the rest are skipped.
ENDIF
```

IF ELSE ENDIF

USE

To provide multiple decision points inside a macro.

SYNTAX

IF *conditionA command1*
ELSE *command2*
ENDIF

...or

IF *conditionB*
   sequence1
ELSE
   sequence2
ENDIF

ELEMENT	DESCRIPTION
conditionA	Any expression which is TRUE or FALSE when evaluated.
command1	Any valid TMAC command that is performed if *conditionA* is evaluated as TRUE.
command2	Any valid TMAC command that is performed if *conditionA* is evaluated as FALSE.
conditionB	Any expression which is TRUE or FALSE when evaluated.
sequence1	Any valid TMAC command or a series of commands that is performed if *conditionB* is evaluated as TRUE.
sequence2	Any valid TMAC command or a series of commands that is performed if *conditionB* is evaluated as FALSE.

> Zero (0) is evaluated by TMAC as FALSE. Any non-zero number is evaluated by TMAC as TRUE.

ACTION

Same as **IF ENDIF** command, except the **IF ELSE** command allows two *conditions* to be evaluated.

IF ELSE command allows one of two command sequences to be executed depending on the validity of the *condition*. If the *condition* is TRUE, *command1* or *sequence1* is executed. If the *condition* is FALSE, *command2* or *sequence2* is executed. Every **IF ELSE** command must have an **ENDIF**.

SEE ALSO

IF ENDIF, **IF ELIF ELSE ENDIF**, **CASE**

EXAMPLES

```
IF D>4 PRINT "D HIGH"      // Prints D HIGH if D is greater
                           // than 4; otherwise ...
ELSE PRINT "D LOW"         // D LOW is printed.
ENDIF

IF T=3                  // 1 is added to T if T is equal to 3;
  T=T+1                 // otherwise ...
ELSE
  T=T-1                 //   ... 1 is subtracted from T.
ENDIF
```

IF ELSE (Shorthand)

USE

To utilize an abbreviated version of the **IF ELSE** command.

SYNTAX

command (*condition*? *true*:*false*)

ELEMENT	DESCRIPTION
command	Incomplete command which is performed with one of the arguments (*true* or *false)*.
condition	Any expression which is TRUE or FALSE when evaluated.
true	An argument which when combined with *command* creates a valid TMAC command.
false	An argument which when combined with *command* creates a valid TMAC command.

ACTION

Evaluates *condition* to determine how to finish the incomplete *command.* If *condition* is TRUE, then the argument *true* is used in combination with *command* to create a valid TMAC command. Conversely, if *condition* is FALSE, then the argument *false* is used.

No **ENDIF** is used with this command.

SEE ALSO

IF ENDIF, IF ELSE ENDIF, IF ELIF ELSE ENDIF, CASE

EXAMPLES

```
PRINT (D>4? "D HIGH":"D LOW")    // If D is greater than 4,
                                 // then the complete command
                                 // is: PRINT "D HIGH".  If D
                                 // is not greater than 4, then
                                 // the complete command is:
                                 // "PRINT "D LOW".  The now
                                 // completed command is then
                                 // executed.

T= (T=3? T+1:T-1)          // If the condition T=3 is true, then the
                           // command T=T+1 is executed.  If the
                           // condition is false, then the command
                           // T=T-1 is executed.
```

INPUT

USE

To enter numerical or string data during macro execution.

SYNTAX

INPUT *variable* or
INPUT *string*

ELEMENT	DESCRIPTION
variable	Any valid variable.
string	Any valid string.

ACTION

Stops execution of the macro until numerical or string data is entered and ENTER Key is pressed. The value of the data entered is assigned to *variable* or *string* and command execution continues. For the HOST, an input box is displayed with a width set using the **EDIT:WIDTH** command. Data is entered using the DATA ENTRY Keys and pressing the ENTER Key.

For the Sp Tst, data is entered via the RS-232 terminal.

SEE ALSO

Variables and Arrays in Macros (2-11), Colors (2-13-1), **EDIT:WIDTH**, **VAR**, **STRING**

EXAMPLES

```
*dmc "GET_X",begin
screen:user                  // Displays blank user screen.
*wai                         // Allows time for user screen to be
                             // set up prior to executing next
                             // command.
cls                          // Initializes user screen.
bcolor 0                     // Sets screen background color.
color white,black            // Set on-screen text character color
                             // and background contrast color.
edit:width 76                // Sets input box to 76 pixels wide.
                             // (19 x 4 = 76.  See EDIT:WIDTH.).
print "Enter 65.3"           // Prompts entry again.
keypad:claim                 // Claims keypad for input commands.
input x                      // Opens input box on screen and
                             // pauses macro execution.
print x                      // Prints 65.3 on screen.
keypad:unclaim               // Returns keypad control to HOST.
end
```

INTERP

USE To execute commands that may be defined after the macro containing the commands is defined.

SYNTAX **INTERP** *"string;string;.. ;string"*

ELEMENT	DESCRIPTION
string	String variables, literal strings and string functions.

ACTION The **INTERP** command executes each *string* as a command and is limited to one line.

The **INTERP** command may be run inside or outside a macro.

SEE ALSO **EXEC**

EXAMPLES

```
*dmc "run_tests",begin
interp "test"+str($1)      // Runs macro named "test + place-
                           // holder value" entered with
                           // run_tests command.
end

/* Entering run_tests 4 executes this macro and passes 4
   into the macro.  If test4 macro is not defined
   before defining the run_tests macro, the run_tests macro
   will fail while being defined (typically during upload)
   unless the INTERP command is used. */
```

KEY

USE To wait and return keycode of next key pressed.

SYNTAX **KEY**

ACTION Pauses macro and ...

(*HOST only*) returns keycode (See Appendix B) of the next key pressed on keypad of the HOST.

(*Sp Tst only*) returns the ANSI terminal keycode (usually ASCII) of the next key pressed on the RS-232 terminal keyboard.

Used with a **PRINT**, **PPRINT**, **HPRINT** or **XYPRINT** command.

SEE ALSO **INPUT, KEY?**

EXAMPLES

```
PRINT KEY                 // For HOST:  After a Front Panel Key
                          // is pressed, the keycode is printed on
                          // the color display.
                          // For Sp Tst Option:  After a terminal key
                          // is pressed, the keycode is printed on
                          // the terminal monitor.

*dmc"keycodes",begin
var keyCode=0                // Initializes keycode value.
keypad:claim                 // Claims keyboard (HOST only).
do                           // Starts DO UNTIL loop.
keyCode=key                  // Queries for keycode of key pressed
                             // and assigns keycode to keyCode.
print keyCode                // Prints keycode of last key pressed.
until keyCode=8200           // Returns macro execution to beginning
                             // of loop unless asterisk (*) key
                             // is pressed.
print "done"                 // Prints done when DO UNTIL loop is
                             // exited.
keypad:unclaim               // Releases control of HOST keyboard.
end
```

KEY?

USE To return 1 if key is pressed; 0 otherwise.

SYNTAX **KEY?**

ACTION Returns 1 if key is pressed on keypad of HOST (HOST only) or on keyboard of RS-232 terminal (Sp Tst only) when command is encountered. Returns 0 if no key is pressed.

Used with **IF...**, **WHILE**, **DO UNTIL** commands.

SEE ALSO **INPUT**, **KEY**

EXAMPLES

```
*dmc"key_query",begin
keypad:claim                 // Claims HOST keypad.  Use only
                             // for HOST.
while !key?                  // Loops while no key is pressed.
print "waiting"              // Prints "waiting" for each pass of
                             // loop until a key is pressed.
wend
print "key pressed!"         // Prints message when key is pressed
                             // and WHILE loop is exited.
keypad:unclaim               // Releases HOST keypad (HOST
                             // only).
end
```

KEYPAD:CLAIM (HOST Only)

USE To direct all input from Front Panel Keypad to the TMAC Interpreter.

SYNTAX **KEYPAD:CLAIM**

ACTION Directs all input from the Front Panel Keyboard to the TMAC Interpreter.

Recommended for use with **INPUT**, **KEY** and **KEY?** commands.

SEE ALSO **KEYPAD:UNCLAIM**

EXAMPLES See **KEY** example.

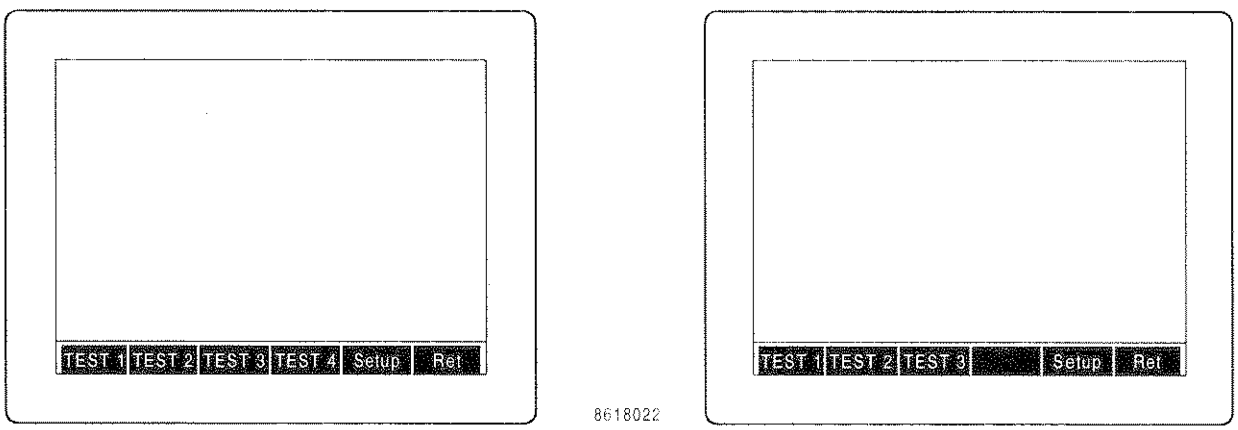
KEYPAD:ERASE (HOST Only)

USE To erase Soft Function Key definitions.

SYNTAX **KEYPAD:ERASE** *n*

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number. Range of *n* is 1 to 6.

ACTION Erases definition for Soft Function Key F*n*.



Before **KEYPAD:ERASE** command is executed

After **KEYPAD:ERASE** command is executed

Figure 3-2 **KEYPAD:ERASE** Example

SEE ALSO Soft Function Key Displays (2-13-4), **KEYPAD:LABel**, **KEYPAD:SOFT**

EXAMPLES

```
KEYPAD:ERASE 4          // Erases definition for Soft
                        // Function Key F4.
```

KEYPAD:LABel

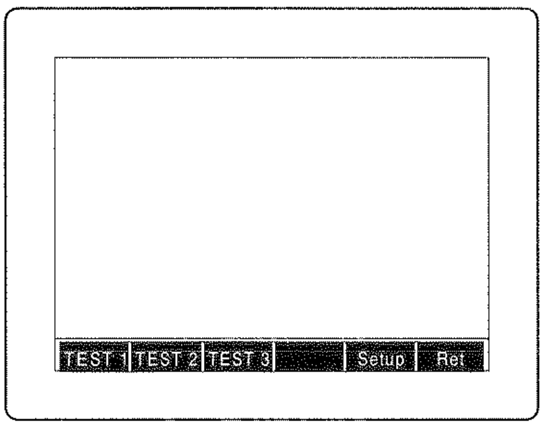
USE To create Soft Function Key definitions.

SYNTAX **KEYPAD:LABel** *n*, *label*

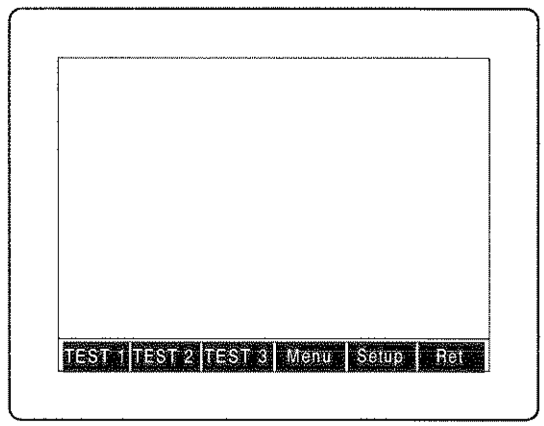
ELEMENT	DESCRIPTION
n	Any constant, variable, integer or any expression that evaluates to a integer. Range of *n* is 1 to 6.
label	Any string variable or literal string surrounded by quotation marks (e.g., "TEST1").

ACTION Creates a new *label* for specified Soft Function Key F*n*.

> Issuing this command with *label* as an empty set of quotation marks ("") is the same as executing the **KEYPAD:ERASE** command.



Before **KEYPAD:LABel** command is executed



After **KEYPAD:LABel** command is executed

Figure 3-3 **KEYPAD:LABel** Example

SEE ALSO Soft Function Key Displays (2-13-4), **KEYPAD:ERASE**, **KEYPAD:SOFT**

EXAMPLES

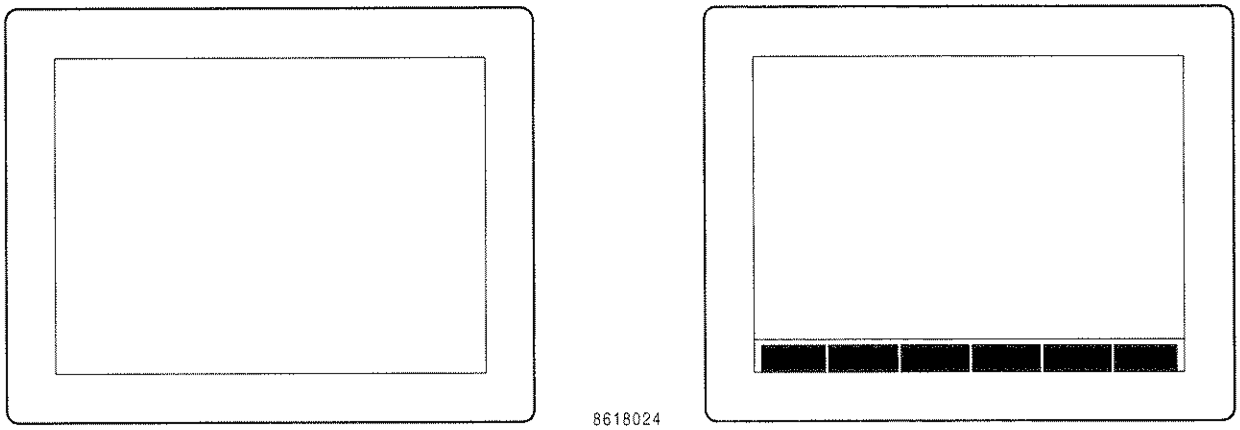
```
KEYPAD:LAB 4, "Menu"
```

KEYPAD:SOFT (HOST Only)

USE To create a Soft Function Key frame.

SYNTAX **KEYPAD:SOFT**

ACTION Creates the Soft Function Key frame at the bottom of the current screen.



Before **KEYPAD:SOFT** command is executed

After **KEYPAD:SOFT** command is executed

Figure 3-4 **KEYPAD:SOFT** Example

SEE ALSO Soft Function Key Displays (2-13-4), **KEYPAD:ERASE**, **KEYPAD:LABel**

EXAMPLES

```
KEYPAD:SOFT

*dmc"panic",begin
screen:user;*wai;cls       // Sets up user screen.
keypad:claim               // Claims keypad for TMAC interpreter.
keypad:soft                // Creates blank Soft Function Key
                           // frame.
keypad:label 6,"panic"     // Labels Soft Function Key F6.
k=key                      // Waits for key press and assigns
                           // keycode to k.
if k=32896                 // Checks to see if key pressed was F6.
print "AIIIIIEEE!!!"       // Provides response to user input
                           // if F6 was pressed.
endif
keypad:unclaim             // Releases keypad.
end
```

KEYPAD:UNCLAIM (HOST Only)

USE To release the Front Panel Keypad for normal use.

SYNTAX **KEYPAD:UNCLAIM**

ACTION Releases the Front Panel Keypad for normal use. Used after claiming the keypad (See **KEYPAD:CLAIM**) for routing all keypad input to the TMAC Interpreter.

SEE ALSO **KEYPAD:CLAIM**

EXAMPLES See **KEY** example.

KILL

USE To delete multi-tasking macro.

SYNTAX **KILL** *name*

ELEMENT	DESCRIPTION
name	Any string variable or literal string surrounded by quotation marks (e.g., "TEST1").

ACTION Removes specified task specified by *name* from the schedule queue and frees memory.

SEE ALSO **TASK, ACTIVATE, TPAUSE, TSTOP, SLEEP, WAKE**

LEN

USE To return the number of characters in a string.

SYNTAX **LEN(***string***)**

ELEMENT	DESCRIPTION
string	String variable or literal string surrounded by quotation marks (e.g., "radio test").

ACTION Returns the length of *string* in number of characters.

SEE ALSO **ASC, STR, STRPOS, VAL**

EXAMPLES

```
string how_long
how_long = "any string"
B = len(how_long)          // Assigns 10 to B.
L = len("another string")  // Assigns 14 to L.
```

LJPRINT (Sp Tst Only)

USE To display numbers or text, left justified, on color display of HOST.

SYNTAX **LJPRINT** *value,x,y,width*

ELEMENT	DESCRIPTION
value	Any constant, variable, string variable, literal string (surrounded by quotation marks), number or any expression that evaluates to a string or number. Strings or string variables and numbers or number variables cannot be mixed.
x,y	Starting point of new display area in pixels (0,0 to 639,349).
width	Any constant, variable or integer or any expression that evaluates to an integer. Expressed in number of pixels

ACTION Erases the display area on the screen of the HOST defined by the *x,y* starting position and *width* in pixels, then prints *value*, left justified, in the defined area.

SEE ALSO **CENTER, HPRINT, PRINT, RJPRINT, XYPRINT**

EXAMPLES

```
$ = "How many lambs did "
ljprint $+"Mary have?",100,100,410      // Prints, left justified
                                        // in erased area display
                                        // area, "How many lambs
                                        // did Mary have?"
ljprint 3*2-5,100,130,20                // Prints 1.
```

LN

USE To return the natural logarithm of *n*.

SYNTAX **LN(***n***)**

ELEMENT	DESCRIPTION
n	Any constant, variable, positive number or any expression that evaluates to a positive number. If n ≤ 0, **LN** command execution results in an error message as this function is not defined for that range of values.

ACTION Returns the natural logarithm (base e) of *n*.

SEE ALSO **EXP, LOG**

EXAMPLES

```
VAR Nat_log
D = 5
Nat_log = LN(D)        // The natural logarithm of 5 (1.609438)
                       // is assigned Nat_log.
```

LOG

USE To return the logarithm of *n* (log *n*).

SYNTAX **LOG(*n*)**

ELEMENT	DESCRIPTION
n	Any constant, variable, positive number or any expression that evaluates to a positive number.

> If n ≤ 0, **LOG** command execution results in an error message as this function is not defined for that range of values.

ACTION Returns the base 10 logarithm of *n*.

SEE ALSO ****, EXP, LN**

EXAMPLES

```
VAR Log_C
C = 5
Log_C = LOG(C)          // The base 10 logarithm of 5 (0.698970004)
                        // is assigned to Log_C.
```

NVRCL (HOST Only)

USE To recall a variable or array stored in non-volatile memory.

SYNTAX **NVRCL** *name,location*

ELEMENT	DESCRIPTION
name	Name of variable or array to be saved.
location	Any constant, variable, integer or any expression that evaluates to an integer. Range 0 to 283.

ACTION Recalls global variables and arrays saved in nonvolatile memory. The recalled value(s) are assigned to any previously declared variable or array. Location of the variable or first element of the array in nonvolatile memory is identified by *location*. Arrays take a memory location for each element.

SEE ALSO Saving Variables and Arrays (2-3-2), **NVSAV**

EXAMPLE See **NVRCL** examples (2-3-2).

NVSAV (HOST Only)

USE To save variables and arrays in non-volatile memory.

SYNTAX **NVSAV** *name,location*

ELEMENT	DESCRIPTION
name	Name of variable or array to be saved.
location	Any constant, variable, integer or any expression that evaluates to an integer. Range 0 to 283.

ACTION Saves global variables or arrays in non-volatile memory. Memory location to store variable or first element of array is identified by *location*.

> Global variables and arrays are not saved when Test Set is turned off and need to be saved in non-volatile memory.

SEE ALSO Saving Variables and Arrays (2-3-2), **NVRCL**

EXAMPLE See **NVSAV** examples (2-3-2).

OFF

See **FALSE**.

ON

See **TRUE**.

OR

USE Logical OR.

SYNTAX *cmdA argument1* **OR** *argument2 cmdB* or
cmdA argument1 **OR** *argument2* **OR** *argument3* ... **OR** *conditionX cmdB*

ELEMENT	DESCRIPTION
cmdA	Any decision point command (**IF**, **ELIF**, **UNTIL**, **WHILE**).
argument1	Any expression that evaluates to TRUE, FALSE, 0 (zero) or any non-zero number (0 = FALSE, any non-zero number = TRUE).
argument2 & on	Same as *argument1*.
cmdB	Any valid TMAC command.

ACTION Evaluates *argument1* and *argument2* and, if either are TRUE (or evaluate to a non-zero number), then the result of the OR operation is TRUE (FALSE, otherwise). If more than one OR operation is encountered then the OR operations are evaluated from left to right. Successive ORs evaluate the result of previous OR operations along with individual expressions.

Table 3-6 lists the expression combinations and the OR operation results.

argument1	*argument2*	**OR Result**
False	False	False
False	True	True
True	False	True
True	True	True

Table 3-6 OR Operation Results

SEE ALSO **!, AND, =, !=, <, >, <=, >=, DO UNTIL, IF ..., WHILE**

EXAMPLES

```
var x=7, z=4
if x=7 OR x<z          // Decision point command utilizing OR
                       // as condition with accompanying
                       // expressions to evaluate.  Condition is
                       // TRUE, because one of the expressions
                       // evaluates to TRUE.
  print y              // Performs resulting command since
                       // condition is TRUE.
ENDIF
```

PAINT

USE To perform a paint fill.

SYNTAX **PAINT** *x,y,fc,bc*

ELEMENT	DESCRIPTION
x,y	Both either a constant, variable, number or any expression that evaluates to a number. Ranges are 0 to 639 and 0 to 349.
fc	Any constant, variable, number or any expression that evaluates to an number. Range is 0 to 15. See Table 2-2 for the available colors and associated numerical values.
bc	Any constant, variable, number or any expression that evaluates to an number. Range is 0 to 15. See Table 2-2 for the available colors and associated numerical values.

ACTION Performs a paint fill of simple or complex shapes using Fill Color (*fc*) specified, starting from Starting Location (*x,y*) and attempts to fill entire screen stopping only at the Boundary Color (*bc*) specified.

SEE ALSO Graphics and Text (2-13-5).

PIXEL

USE To create a user-defined dot on the color display.

SYNTAX **PIXEL**

ACTION Creates a dot in the current foreground color at the current x,y location.

> Changing foreground color or xy location after placing creating a dot on the color display, leaves the dot unaffected.

SEE ALSO **COLOR**, Graphics and Text (2-13-5), **XY**

EXAMPLES

```
XY 300,150
COLOR 6
PIXEL                   // Creates a brown dot at location 300,150.
```

PIXLEN (HOST Only)

USE To return length of a string in pixels.

SYNTAX **PIXLEN(***string***)**

ELEMENT	DESCRIPTION
string	Any string variable or literal string surrounded by quotation marks (e.g., "TELEPHONE TESTS")

ACTION Returns the length of *string* in pixels.

> Useful for centering text on the color display of the HOST.

EXAMPLES

```
STRING Mode             // Declares string Mode.
Mode = "Test Mode 4"    // Initializes Mode.
P = PIXLEN(Mode)        // Assigns pixel length of Mode (160),
                        // which is the length of "Test Mode 4"
                        // in pixels to P.
XY ((640-P)/2)-1,40     // Sets point for horizontal center.
Print Mode              // Prints Mode string.
```

PIXLEN? (Sp Tst Only)

USE To return number of pixels required to display a given value.

SYNTAX **PIXLEN?** *value*

ELEMENT	DESCRIPTION
value	Any constant, variable, number, any expression that evaluates to a number, string variable or literal string surrounded by quotation marks (e.g., "REPEATER TEST").

> Any numerical or string expression is evaluated by TMAC prior to the pixel length being calculated.

ACTION Returns the length in pixels required to display *value* on the screen of the HOST.

SEE ALSO **CENTER, LJPRINT, RJPRINT**

PPRINT (HOST Only)

USE

To print responses out the RS-232 Connector.

SYNTAX

PPRINT *argument* or
PPRINT *argument, argument, ... ,argument* or
PPRINT *argument, argument, ... ,argument,* "\ccc" or
PPRINT %0*nl, argument, argument, ... ,argument* or
PPRINT *argument,* %0*nl, argument, ... ,argument,* "\ccc"

ELEMENTS	DESCRIPTION
argument	Expression to be printed. Can be a mathematical expression, a literal string or the contents of variables. Mathematical expressions are calculated and string functions are performed before the results are printed.
0	Zero signifies leading zeros are added to fill the field width. The 0 is omitted for no leading zeros. Applies only to numeric expressions.
n	Specifies the field width. *n* is the number of digit spaces (decimal point counts as one). Applies only to numeric expressions.
l	Specifies the numeric base the data is printed in. Applies only to numeric expressions. Omit for a default of Signed Decimal or enter one of the following: B Binary Q Octal H Hexadecimal U Unsigned decimal D Signed decimal
ccc	Optional non-printable characters in octal form used to specify the output. -or- n or l Specifies a line feed character. r Specifies a carriage return.

ACTION

Sends all responses that would normally be printed on the color display to the device connected to the RS-232 Connector. (x,y location has no affect on the **PPRINT** command.)

%0*nl* contains the format settings for numeric expressions and can be placed anywhere in the **PPRINT** command. There can be several format settings in each **PPRINT** command, each one changing the leading zero format, the numeric base and the field width for the expressions that follow. Format settings do not affect the printing of strings.

SEE ALSO

PRINT, PSCREEN

EXAMPLES

```
PPRINT "Ring\007\n"   // Prints Ring on the RS-232 device.
                      // The \007 causes the device's bell to
                      // sound (if device has a bell).
                      // The \n provides a line feed command to
                      // the device.
```

PRINT

USE

To print on the display screen (HOST) or to print responses out the OPT. RS-232 Connector (Sp Tst Option).

SYNTAX

PRINT *argument* or
PRINT *argument, argument* , ... ,*argument*, or
PRINT %0*nl*, *argument*, ... ,*argument*, or
PRINT *argument*, ... ,%0*nl*, *argument*,

ELEMENT	DESCRIPTION
argument	Expression to be printed. Can be a mathematical expression, a literal string or the contents of variables. Mathematical expressions are calculated and string functions are performed before the results are printed.
0	Zero signifies leading zeros are added to fill the field width. The 0 is omitted for no leading zeros. Applies only to numeric expressions.
n	Specifies field width. *n* is the minimum number of digit spaces (decimal point counts as one). Applies only to numeric expressions. For Sp Tst, if *l* is D, *n* can be in the form of *b.a*, where *b* designates minimum number of digits before the decimal point and *a* designates the maximum number of digits after the decimal point.
l	Specifies the numeric base the data is printed in. Applies only to numeric expressions. Omit for a default of Signed Decimal or enter one of the following:

B	Binary
Q	Octal
H	Hexadecimal
U	Unsigned decimal
D	Signed decimal

ACTION

Prints numerical and string values out the OPT. RS-232 Connector (Sp Tst) or at the current x,y location on the color display (HOST). Several values are printed on the same line by separating them with commas. If the optional ending comma is used, the next **PRINT** command continues on the same line; otherwise a new line is started. If an *argument* consists of a variable, the value of the variable is printed. Mathematical expressions are calculated before printing.

%0nl contains the format settings for numeric expressions and can be placed anywhere in the **PRINT** command. There can be several format settings in each **PRINT** command, each one changing the leading zero format, the numeric base and the field width for the expressions that follow. Format settings do not affect the printing of strings.

SEE ALSO **HPRINT, XYPRINT, PPRINT, PSCREEN, XY**

EXAMPLES

```
X=44
Y=37.83
$="String"
PRINT %04,X," ",$       // Specifies leading zeros, a field width
                        // of 4 characters and signed decimal (by
                        // default) for any following numerical
                        // outputs.  The following is printed:
                        // 0044 String.

PRINT X,%3," ",Y        // Prints X as is.  Then specifies no
                        // leading zeros, a field width of 3 and
                        // signed decimal (by default) for any
                        // following numerical outputs.  A single
                        // space is printed.  When Y is printed,
                        // the print field is expanded to accommodate
                        // Y.  Because each argument of the PRINT
                        // command (except for Y) is following by a
                        // comma, output is displayed on one line.
                        // The following is printed:  44 37.83

PRINT %4B,X             // Specifies no leading zeros, a field
                        // width of 4 and Binary numeric base for
                        // any following numerical outputs.  The
                        // following is printed:  101100.
                        // The field width is enlarged to
                        // accommodate X.

PRINT %7,$,Y            // Specifies no leading zero, a field
                        // width of 7 and signed decimal (by
                        // default) for any following numerical
                        // outputs.  The following is printed:
                        // String  37.83.  2 blank character
                        // spaces before 3 are provided by the
                        // field width of 7.

PRINT %03Q,X,$,%6,Y     // Specifies leading zeros, a field width
                        // of 3 and Octal numeric base.  Then X
                        // is printed as formatted.  Then String
                        // is printed immediately after because
                        // of the comma between.  Then no leading
                        // zeros, a field width of 6 and signed
                        // decimal numeric base (by default) is
                        // specified.  Then Y is printed.
                        // The complete printed expression is as
                        // follows:  054String 37.83.
```

PSCREEN (HOST Only)

USE To print the display screen to output device.

SYNTAX **PSCREEN**

ACTION Prints the display screen to device connected to the RS-232, GPIB or PRINTER (parallel) Connectors. Output device, connector and print parameters are specified by the Printer submenu under menu item "External I/O" in the Auxiliary Functions Menu (see 3-3-10 of IFR-1900 Operation Manual). Command is entered through RS-232 or GPIB Connector (see 6-2-1 or 6-2-4 of IFR-1900 Operation Manual) or placed in a macro.

> Issuing this command is equivalent to pressing the PRINT SCRN Key on the Front Panel of the Test Set. See Figure 3-1 of IFR-1900 Operation Manual.

SEE ALSO **PRINT**, **PPRINT**

RAND

USE To specify seed for **RND** random generator.

SYNTAX **RAND** *x*

ELEMENT	DESCRIPTION
x	Any constant, variable, number or any expression that evaluates to a number.

ACTION Sets the seed (starting point) for the random generator.

> Using the same seed causes the random generator to produce the same series of pseudo-random numbers.

SEE ALSO **RND**

EXAMPLES

```
*DMC"RND_GEN",BEGIN
RAND 12                    // Sets a starting point for the RND
                           // command.
FOR X=1 TO 10 STEP 1
Z=RND(100)                 // Generates the following numbers:
PRINT Z                    // 47, 77, 31, 27, 39, 100, 57, 46, 41 and 30.
NEXT X
END

/*  Each time RAND is set to 12, the RND(100) command produces
    the same series of random numbers.    */
```

RETURN

USE To return a value from a function or to return macro execution to the previous macro.

SYNTAX **RETURN**

ACTION Stops command execution of the current macro when command is encountered. If command is encountered and the current macro was called from another macro, command execution returns to the macro from which the current macro was called. Command execution resumes at the command immediately following the call of the macro or function. If a variable is specified with the **RETURN** command, the value of the variable is returned.

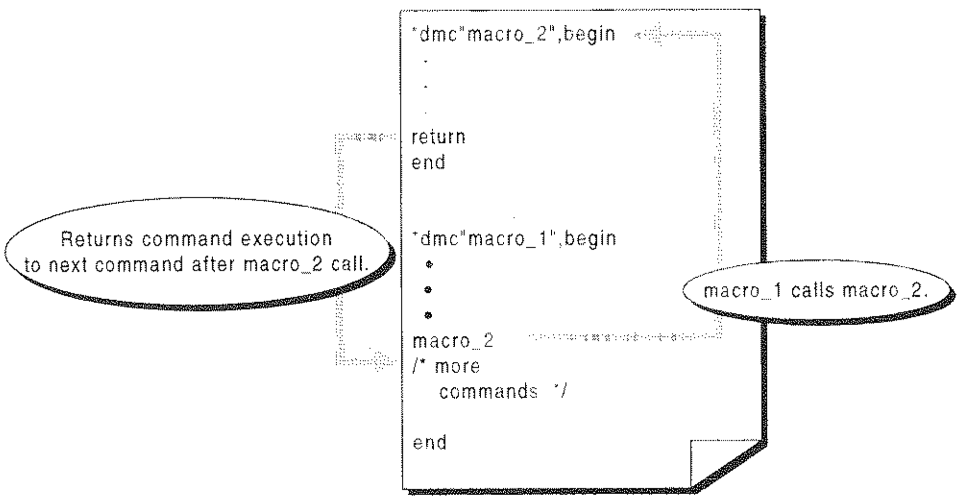


Figure 3-5 RETURN Command

EXAMPLE

```
*DMC "Even",BEGIN              // Macro named "Even" that is called
                               // by the macro named "Test" below.
IF $1 % 2 = 0
   PRINT $1," IS EVEN"
ELSE
   RETURN
ENDIF
END

*DMC "Test",BEGIN              // Macro is called with two arguments:
                               // x raised to the power of y (Test x,y).
Y= $1**$2
PRINT $1,"TO THE POWER OF ",$2," IS ",Y
Even Y
PRINT "TEST IS DONE"
END

/* If passed variable is odd ($1 % 2 equals 1), the RETURN
   command returns command execution back to the macro Test at
   the PRINT "TEST IS DONE" command.   */
```

(Examples continued on next page.)

(Examples continued from previous page.)

```
*DMC "Maximum",BEGIN
VAR Max=0
EDIT:WIDTH 45
FOR X=1 TO 10
INPUT Y
IF Y>Max
   Max=Y
ENDIF
NEXT X
RETURN Max
END

/* The RETURN command returns the value of Max.  The function
   Maximum cannot be executed by simply entering it's name.
   Functions are executed by assigning them to a variable or
   using them in a PRINT command.      */

Z=Maximum              // Function Maximum is executed and the
                       // result is assigned to Z.

PPRINT Maximum         // Function Maximum is executed and the
                       // result is printed on the Host
                       // Terminal (HOST only).
```

RJPRINT (Sp Tst Only)

USE To display numbers or text, right justified, on color display of HOST.

SYNTAX **RJPRINT** *value,x,y,width*

ELEMENT	DESCRIPTION
value	Any constant, variable, string variable, literal string (surrounded by quotation marks), number or any expression that evaluates to a string or number. Strings or string variables and numbers or number variables cannot be mixed.
x,y	Both either a constant, variable, number or any expression that evaluates to a number. Ranges are 0 to 639 and 0 to 349.
width	Any constant, variable or integer or any expression that evaluates to an integer. Expressed in number of pixels

ACTION Erases the display area on the screen of the HOST defined by the *x,y* starting position and *width* in pixels, then prints *value*, right justified, in the defined area.

SEE ALSO **CENTER**, **HPRINT**, **LJPRINT**, **PRINT**, **XYPRINT**

RND

USE To generate a pseudo-random integer between 0 and *n*.

SYNTAX **RND(*n*)**

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number.

ACTION Generates a pseudo-random integer between 0 and *n*. If *n* is not an integer, the value is truncated.

SEE ALSO **RAND**

EXAMPLES

```
WW=RND(100)          // Assigns a pseudo-random number between
                     // 0 and 100 to WW.
```

ROOM

USE To return the amount of memory size available.

SYNTAX **ROOM**

ACTION Returns the available memory size, in bytes, when printed.

SEE ALSO Macros (2-9)

EXAMPLES

```
PRINT ROOM           // Prints the amount of memory available
                     // to the color display (HOST) or out
                     // the OPT. RS-232 (Sp Tst Option).
```

ROTATE (HOST Only)

USE To rotate text, character by character.

SYNTAX **ROTATE *n***

ELEMENT	DESCRIPTION
n	Degrees the character is rotated.

ACTION Specifies the orientation (in 90° intervals) new text is printed, character by character, on the color display. Table 3-7 shows the amount of clockwise rotation for values of *n*.

n	ROTATION
0 to 89	0
90 to 179	90°
180 to 269	180°
270 to 359	270°

Table 3-7 Character Rotation for *n* Values

SCREEN:USER (HOST Only)

USE To create a blank screen in active window.

SYNTAX **SCREEN:USER**

ACTION Creates a blank user screen (with a dark blue background) in the active window.

- Stops any active task.
- Blank screen may be used to create user-developed screens.
- User screen decreases run time because of no readings to update.

SEE ALSO **USER** (for Sp Tst only)

EXAMPLES

```
screen:user             // Displays blank user screen.
*wai                    // Forces macro execution to wait until
                        // user screen is completed.
cls                     // Initializes user screen.
bcolor 0                // Establishes background color (user
                        // should establish background color
                        // of new screen).
```

SIGN

USE To return -1, 0, 1 if *n* is a negative number, zero or a positive number.

SYNTAX **SIGN(*n*)**

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number.

ACTION Returns -1 if $n < 0$, 0 if $n = 0$ or 1 if $n > 0$.

SEE ALSO **ABS**, **FLOOR**

EXAMPLES

```
S=SIGN(-6)              // Assigns -1 to S.
```

SIN

USE To return the sine of angle *n* measured in radians.

SYNTAX **SIN(*n*)**

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number.

ACTION Returns the sine of angle *n* measured in radians.

SEE ALSO **COS**

EXAMPLES

```
VAR Sine_of_radians
Sine_of_radians=SIN(3.1415/3)    // Assigns the sine of
                                 // π/3 (0.86601) to
                                 // Sine_of_radians.
```

SLEEP

USE To temporarily stop multitasking a macro.

SYNTAX **SLEEP** *name*

ELEMENT	DESCRIPTION
name	Name of the macro to remove from schedule queue. String variable or literal string surrounded by quotation marks (e.g., "TEST_4").

ACTION Removes the specified task from the schedule queue. The task can be put back into the schedule queue using the **WAKE** command.

SEE ALSO Multitasking Macros (2-12), **WAKE**

EXAMPLES See Multitasking examples (2-12).

SOUND

USE To create audio tones.

SYNTAX **SOUND** *freq,duration*

ELEMENT	DESCRIPTION
freq	Any constant, variable, positive number or any expression that evaluates to a positive number. Effective range is 125 to 15000.
duration	Any constant, variable, positive number or any expression that evaluates to a positive number.

ACTION Generates a tone of frequency *freq* in Hz for a length of time *duration* in ms and routes the tone to the Test Set Speaker.

EXAMPLES

```
SOUND 600,1000          // Creates a 600 Hz tone for 1 sec.
```

SQR

USE
To return to the positive square root of *n*.

SYNTAX
SQR(*n*)

ELEMENT	DESCRIPTION
n	Any constant, variable, positive number or any expression that evaluates to a positive number.

ACTION
Returns the positive square root of *n*.

> The square root function returns 0 for the square root of negative numbers.

SEE ALSO
EXP, SIGN

EXAMPLES

```
VAR Square_Root
Square_Root = SQR(30)      // Assigns the positive square root
                           // of 30 (5.477226) to Square_Root.
```

STATus:OPERation:INSTRument (HOST Only)

USE
To control or query the Operation Instrument Register.

SYNTAX
STATus:OPERation:INSTRument:CONDition? or
STATus:OPERation:INSTRument:ENABLe *n* or
STATus:OPERation:INSTRument:ENABLe? or
STATus:OPERation:INSTRument:EVENt?

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number (16 bit value). Range is 0 to 65535.

ACTION
Specifies or returns the contents of the following registers within the Operation Instrument Register: Condition register, Enable register and Event register.

> The Operation Instrument Register is used with the **INITiate** and **FETCh** commands. Once an **INITiate** command is completed and the meter is ready to be read, the appropriate meter ready bit is set to 1. This indicates the meter is ready for the **FETCh** command.

SEE ALSO
Status Subsystem (2-16), Figures 2-5 and 2-6

STATus:OPERation (HOST Only)

USE To control or query the Operation Status Register.

SYNTAX **STATus:OPERation:CONDition?** or
STATus:OPERation:ENABLe *n* or
STATus:OPERation:ENABLe? or
STATus:OPERation:EVENt?

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number (16 bit value). Range is 0 to 65535.

ACTION Specifies or returns the contents of the following registers within the Operation Status Register: Condition register, Enable register and Event register.

> The Operation Status Register contains the Waiting For Arm, Waiting For Trigger and Instrument Summary bits. The Operation Instrument Register sets the Instrument Summary bit.

SEE ALSO Status Subsystem (2-16), Figures 2-5 and 2-6, **STATus:OPERation:INSTRument** commands

STATus:PRESet (HOST Only)

USE To set all the bits in the enable registers to an initialized condition.

SYNTAX **STATus:PRESet**

ACTION Sets all bits in enable registers to an initialized condition. Presets the enable registers of the following status registers to the associated condition:

STATUS REGISTER	CONDITION
Operation Status	OFF (#h0000)
Operation Instrument	OFF (#h0000)
Questionable Status	OFF (#h0000)
Instrument Status	ON (#h7FFF)
Instrument Summary	ON (#h7FFF)

Table 3-8 Enable Register Preset Conditions

SEE ALSO Status Subsystem (2-16), Figure 2-6

STATus:QUEStionable:INSTRument:ISUMmary (HOST Only)

USE

To control or query the Instrument Summary Status Register.

SYNTAX

STATus:QUEStionable:INSTRument:ISUMmary:CONDition? or
STATus:QUEStionable:INSTRument:ISUMmary:ENABLe *n* or
STATus:QUEStionable:INSTRument:ISUMmary:ENABLe? or
STATus:QUEStionable:INSTRument:ISUMmary:EVENt?

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number (16 bit value). Range is 0 to 65535.

ACTION

Specifies or returns the contents of the following registers within the Instrument Summary Status Register: Condition register, Enable register and Event register.

The Instrument Summary Status Register acts as an extension of the Instrument Status Register. The Instrument Summary Status Register reports the exceeding of the upper and lower limit of the SINAD and Phase Meter and Digital Multimeter.

SEE ALSO

Status Subsystem (2-16), Figure 2-6

EXAMPLES

```
*DMC "DMM_READ",BEGIN
STAT:QUES:INSTR:ISUM:ENABLE 12  // Sets enable register to
                                // read DMM Upper and Lower
                                // Limit exceeded.
STAT:QUES:INSTR:ENABLE 2      // Sets enable register to pass the
                              // Instrument Summary Status
                              // Register result.
STAT:QUES:ENABLE 8192         // Sets enable register to pass the
                              // Instrument Status Register result.
*CLS                          // Clears all condition and event
                              // registers.
SCREEN:DMM                    // Displays DMM Operation Screen.
PPRINT M_DMM?                 // Prints DMM reading to Host.
IF (*STB? & 8) != 0           // If Status Byte reports
                              // Questionable Status bit as 1,
  PPRINT "limit exceeded"       // print limit exceeded to Host.
ELSE                            // If Status Byte reports
                                // Questionable Status bit as 0,
  PPRINT "limit not exceeded"   // print limit not exceeded to
                                // Host.
ENDIF
END
```

(Examples continued on next page.)

(Examples continued from previous page.)

```
/* The above macro performs a DMM reading and checks the
   Status Byte for the reporting of the DMM Upper or Lower
   Limit being exceeded.  The enable registers of the
   Instrument Summary, Instrument and Questionable Status
   Registers are set to pass the information to the Status
   Byte.      */
```

STATus:QUEStionable:INSTRument (HOST Only)

USE To control or query the Instrument Status Register.

SYNTAX
STATus:QUEStionable:INSTRument:CONDition? or
STATus:QUEStionable:INSTRument:ENABLe *n* or
STATus:QUEStionable:INSTRument:ENABLe? or
STATus:QUEStionable:INSTRument:EVENt?

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number (16 bit value). Range is 0 to 65535.

ACTION Specifies or returns the contents of the following registers within the Instrument Status Register: Condition register, Enable register and Event register.

> The Instrument Status Register reports meter readings exceeding the upper and lower limit of the AF, RF Power, FM Deviation, AM Modulation and Distortion Meters. The results of the Instrument Summary Status Register (an extension of the Instrument Status Register) are also reported.

SEE ALSO Status Subsystem (2-16), Figure 2-6

EXAMPLES See **STATus:QUEStionable:INSTRument:ISUMmary.**

STATus:QUEStionable (HOST Only)

USE

To control or query the Questionable Status Register.

SYNTAX

STATus:QUEStionable:CONDition? or
STATus:QUEStionable:ENABLe *n* or
STATus:QUEStionable:ENABLe? or
STATus:QUEStionable:EVENt?

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number (16 bit value). Range is 0 to 65535.

ACTION

Specifies or returns the contents of the following registers within the Questionable Status Register: Condition register, Enable register and Event register.

> The Questionable Status Register reports events that could bring Test Set results into question. The Instrument Status Register provides the input. Figure 2-6 shows events reported by both registers.

SEE ALSO

Status Subsystem (2-16), Figure 2-6

EXAMPLES

See **STATus:QUEStionable:INSTRument:ISUMmary**.

STOP

USE

For Multitasking. To stop command execution of all macros.

SYNTAX

STOP

ACTION

Stops command execution of all macros regardless of where the **STOP** command is located.

EXAMPLES

```
*DMC "Test",BEGIN
Y= $1**$2
IF $2 < 0
   STOP                  // If the passed variable is negative,
                         // the STOP command is executed and all
                         // macro command execution stops.
ENDIF
PRINT $1,"TO THE POWER OF ",$2," IS ",Y
PRINT "TEST IS DONE"
END
```

SEE ALSO

Multitasking Macros (2-12)

STR

USE To return the string equivalent of a number.

SYNTAX **STR(*n*)**

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number.

ACTION Returns the string equivalent (character representation) of number *n*.

SEE ALSO **ASC, CHR, VAL**

EXAMPLES

```
$ = STR(236)          // Assigns the characters 236 to
                      // string variable $.
```

STRING

USE To declare string variables and arrays.

SYNTAX

STRING *name* or
STRING *name, name, ... ,name* or
STRING *name[index]* or
STRING *name[index], name[index], ... ,name[index]* or
STRING *name, name[index]* or
STRING *name[index][length]* (Sp Tst only)

ELEMENT	DESCRIPTION
name	Name of the string declared. The first character of *name* must be a letter while the remaining characters can be letters, digits and the underscore character. The length of the *name* can be from 2 to 31 characters.
index	Number of array elements minus one.
length	Maximum length of each array element specified by *index* (Sp Tst only).

ACTION Declares string variables and arrays. String variables and arrays cannot be initialized when declared. (Sp Tst only: Specifying maximum length of each element in a string array may be used to save system memory. Not to be confused with substringing; see 2-8.)

> String variables and arrays are either local or global. Local string variables are declared inside a macro and have no meaning outside that macro. Global string variables are declared outside macros and are used inside or outside any macro.

SEE ALSO String Variables and Functions (2-8)

EXAMPLES

```
STRING Mess_1,Mess_2          // Declares string variables named
                              // Mess_1 and Mess_2.
Mess_1="Test "                // Assigns "Test " to variable Mess_1.
Mess_2=Mess_1+"complete"      // Assigns "Test complete" to the
                              // string variable Mess_2.
```

STRPOS

USE To return position of a string within another.

SYNTAX **STRPOS(***string1*,*string2***)**

ELEMENT	DESCRIPTION
string1	Any string variable or literal string surrounded by quotation marks (e.g., "RADIO TESTS NO. 2").
string2	Any string variable or literal string surrounded by quotation marks (e.g., "TESTS NO.").

ACTION Returns the position of *string2* inside *string1*. If *string2* is not found, -1 is returned.

The first position of *string1* is 0.

SEE ALSO **LEN**, **VAL**

EXAMPLES

```
STRING One, Two
One = "preset"
Two = "set"
X = STRPOS(One,Two)    // Assigns 3 to X.  "set" starts with the
                       // 4th character in "preset."
```

SYSTem:COMMunicate:GPIB:ADDRess (HOST Only)

USE To specify Test Set address and select Talk/Listen for GPIB mode.

SYNTAX **SYSTem:COMMunicate:GPIB:ADDRess** *a*

ELEMENT	DESCRIPTION
a	Any constant, variable, integer or any expression that evaluates to an integer. Range is 0 to 31.

ACTION Specifies Test Set address and selects Talk/Listen for GPIB mode.

SEE ALSO GPIB Operation (2-15)

EXAMPLES See **SYSTem:COMMunicate:GPIB:SPOLL?**

SYSTem:COMMunicate:GPIB:CMD (HOST Only)

USE To issue commands through GPIB Connector.

SYNTAX **SYSTem:COMMunicate:GPIB:CMD** *string*

ELEMENT	DESCRIPTION
string	Any string variable or literal string surround by quotation marks.

ACTION When operating the Test Set as a GPIB Controller, issues a command sequence through the GPIB Connector.

SEE ALSO GPIB Operation (2-15)

SYSTem:COMMunicate:GPIB:CONTroller (HOST Only)

USE To make Test Set a GPIB controller.

SYNTAX **SYSTem:COMMunicate:GPIB:CONTroller**

ACTION Selects Controller for the GPIB Operation Mode making the Test Set a GPIB controller.

> For the Test Set to become a GPIB controller, the address of the Test Set must be set to an address that does not conflict with a slave device address and the GPIB Operation Mode must be set to Controller.

SEE ALSO GPIB Operation (2-15), **SYSTem:COMMunicate:GPIB:ADDRess**

EXAMPLES See **SYSTem:COMMunicate:GPIB:SPOLL?**

SYSTem:COMMunicate:GPIB:DCL (HOST Only)

USE To issue a Device Clear message.

SYNTAX **SYSTem:COMMunicate:GPIB:DCL**

ACTION When operating the Test Set as a GPIB Controller, issues a Device Clear message.

SEE ALSO GPIB Operation (2-15)

SYSTem:COMMunicate:GPIB:GET (HOST Only)

USE To issue a Group Execute Trigger.

SYNTAX **SYSTem:COMMunicate:GPIB:GET**

ACTION When operating the Test Set as a GPIB Controller, issues a Group Execute Trigger.

SEE ALSO GPIB Operation (2-15)

SYSTem:COMMunicate:GPIB:LONly (HOST Only)

USE To select Listen Only as the GPIB Operation Mode.

SYNTAX **SYSTem:COMMunicate:GPIB:LONly**

ACTION Selects Listen Only as the GPIB Operation Mode.

> To select the Talk/Listen GPIB Mode of Operation, again, reset the Test Set address using the **SYST:COMM:GPIB:ADDR** command.

> When the Test Set becomes a GPIB Listen Only device, the address is first set using the **SYST:COMM:GPIB:ADDR** command. After setting the address, this command selects Listen Only as the GPIB Mode of Operation.

SEE ALSO GPIB Operation (2-15), **SYSTem:COMMunicate:GPIB:ADDRess**

SYSTem:COMMunicate:GPIB:PRINTer (HOST Only)

USE To direct the output of the printer to the GPIB Connector.

SYNTAX **SYSTem:COMMunicate:GPIB:PRINTer**

ACTION Selects the GPIB Connector for printer output.

SEE ALSO GPIB Operation (2-15)

SYSTem:COMMunicate:GPIB:SLAVe (HOST Only)

USE To specify the destination of **SYSTem:PTHRough:GPIB** commands.

SYNTAX **SYSTem:COMMunicate:GPIB:SLAVe** *address*

ELEMENT	DESCRIPTION
address	Any constant, variable, integer or any expression that evaluates to a integer. Range is 0 to 31.

ACTION When operating the Test Set as a GPIB Controller, specifies the destination of **SYSTem:PTHRough:GPIB** commands to the peripheral device with *address*.

SEE ALSO GPIB Operation (2-15), **SYSTem:PTHRough:GPIB**

EXAMPLES See **SYSTem:COMMunicate:GPIB:SPOLL?**

SYSTem:COMMunicate:GPIB:SPOLL? (HOST Only)

USE To perform a Serial Poll on a device.

SYNTAX **SYSTem:COMMunicate:GPIB:SPOLL?** *address*

ELEMENT	DESCRIPTION
address	Any constant, variable, integer or any expression that evaluates to a integer. Range is 0 to 31.

ACTION When operating the Test Set as a GPIB Controller, performs a Serial Poll on the device with *address* and returns the result. The 8 bit result is device dependent, except bit 6, which is the Service Request bit.

SEE ALSO GPIB Operation (2-15)

EXAMPLES

```
*DMC "Control",BEGIN        // Defines a macro that sets the
SYST:COMM:GPIB:ADDR 30      // Test Set's address to 30 and
SYST:COMM:GPIB:CONT         // makes the Test Set a GPIB
END                         // Controller.
/* The macro Control establishes the Test Set as a GPIB
   Controller.     */

*DMC "Slave",SYST:COMM:GPIB:SLAV $1
/* The macro Slave sets the slave address for a peripheral
   device to $1.     */

*DMC "M1200",SYST:PTHR:GPIB 5,STR($1)+"\n"
/* The macro M1200 passes $1, as a string, through the GPIB
   Connector to the peripheral device with an address of 5.
   The "\n" provides a line feed and marks the end of the
   command passed.          */

*DMC "Read_1200",BEGIN
Control                          // Activates Control macro.
M1200 "DVMAC"                    // Sets 1200 to operate voltmeter.
WHILE SYST:PTHR:SER:QUE? =0      // While no Input from RS-232 Host.
 IF SYST:COMM:GPIB:SRQ? =1       // If there is a Service Request,
   P=SYST:COMM:GPIB:SPOLL? 5     // Poll to see if from 1200.
   IF P&64 != 0                  // If bit 6 is 1, SRQ is from 1200.
     M1200 "DVMRM"               // If SRQ is from 1200, take a
                                 // reading.
   ENDIF
 ENDIF
WEND
END
```

(Examples continued on next page.)

(Examples continued from previous page.)

```
/* The macro Read_1200 assumes a FM/AM-1200S/A (1200) is
   connected to the GPIB Connector of the Test Set and has
   an address of 5.  The following sequence occurs:

   o  Makes the Test Set a controller.

   o  Passes the DVMAC command to the 1200.  Sets the 1200
      for autoranging voltmeter readings.

   o  While there is no Input from the RS-232 Host of the
      Test Set, continually checks for Service Requests from
      the 1200.

   o  Upon each 1200 Service Request, the Test Set passes
      the DVMRM command to the 1200.  This command returns
      a voltmeter reading to the Host of the Test Set.     */
```

SYSTem:COMMunicate:GPIB:SRQ? (HOST Only)

USE To check peripheral devices for a Service Request.

SYNTAX **SYSTem:COMMunicate:GPIB:SRQ?**

ACTION When operating the Test Set as a GPIB Controller, checks peripheral devices for a Service Request. Returns a 1 if there was a Service Request since last check. Once read, returns 0 until next Service Request.

SEE ALSO GPIB Operation (2-15)

EXAMPLES See **SYSTem:COMMunicate:GPIB:SPOLL?**

SYSTem:COMMunicate:GPIB:TONly (HOST Only)

USE To select Talk Only as the GPIB Operation Mode.

SYNTAX **SYSTem:COMMunicate:GPIB:TONly**

ACTION Selects Talk Only as the GPIB Operation Mode.

> To select the Talk/Listen GPIB Mode of Operation, again, reset the Test Set address using the **SYST:COMM:GPIB:ADDR** command.

> When the Test Set becomes a GPIB Talk Only device, addressing is unnecessary. The Test Set must be the only talking device on the GPIB bus and the remaining devices must be compatible listeners.

SEE ALSO GPIB Operation (2-15), **SYSTem:COMMunicate:GPIB:ADDRess**

SYSTem:COMMunicate:SERial

USE

To control the RS-232 Connector or select for printer output.

SYNTAX

SYSTem:COMMunicate:SERial:BAUD *baud_rate* or
SYSTem:COMMunicate:SERial:PARity *parity* or
SYSTem:COMMunicate:SERial:BITS *data_bits* or
SYSTem:COMMunicate:SERial:SBITs *stop_bits* or
SYSTem:COMMunicate:SERial:PACE *handshake* or
SYSTem:COMMunicate:SERial:ECHO *echo* or
SYSTem:COMMunicate:SERial:PRINTer

ELEMENT	DESCRIPTION
baud_rate	Any constant, variable, number or any expression that evaluates to a number. Valid values: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600[1] and 115200[1].
parity	**NONE**, **EVEN**, **ODD**, **MARK** or **SPACE**.
data_bits	Any constant, variable, number or any expression that evaluates to a number. Valid values: 7 or 8.
stop_bits	Any constant, variable, number or any expression that evaluates to a number. Valid values: 1 or 2.
handshake	**XON** (Xon/Xoff), **HW** (Hardware) or **NONE**.
echo	Any constant, variable, number or any expression that evaluates to a number. Valid values: 1 (Enable) or 0 (disable).

1. Sp Tst only.

ACTION

Specifies serial communication parameters for operation through the RS-232 Connector or selects the RS-232 Connector for printer output

EXAMPLES

```
*DMC "Control",BEGIN          // Defines macro to send query to
                              // controlled device & receive result.
STRING Result                 // Defines string to hold query result.
SYST:COMM:SER:BAUD 19200      // Sets RS-232 baud rate to 19200.
SYST:COMM:SER:PAR NONE        // Sets RS-232 parity to none.
SYST:COMM:SER:BITS 8          // Sets RS-232 data bits to 8.
SYST:COMM:SER:SBITS 1         // Sets RS-232 stop bits to 1.
SYST:COMM:SER:PACE XON        // Sets RS-232 handshaking to XON-XOFF.
SYST:COMM:SER:ECHO 1          // Sets RS-232 echo on.
SYST:PTHR:SER "query?"        // Sends string "query?" to device.
WHILE !(SYST:PTHR:SER:QUE?)   // Loops while RS-232 queue is empty.
  TPAUSE                      // Allows Test Set operation while
                              // looping.
WEND                          // End of WHILE loop.
Result = SYST:PTHR:SER?       // Stores received string in
                              // Result.
END
```

SYSTem:CURsor:DEFaults (HOST Only)

USE To restore cursor and Soft Function Keys to default state.

SYNTAX **SYSTem:CURsor:DEFaults**

ACTION Restores the cursor positioning and Soft Function Key level of active screen to default state.

> This command should be followed by a screen command to ensure that screens appear correctly.

SEE ALSO **SYSTem:DEFaults**, **SYSTem:DISPlay:DEFaults**

SYSTem:DATE

USE To set or return system date.

SYNTAX **SYSTem:DATE** *year,month,day* or
SYSTem:DATE?

ELEMENT	DESCRIPTION
year	Any constant, variable, number or any expression that evaluates to a number. Range is 0 to 99.
month	Any constant, variable, number or any expression that evaluates to a number. Range is 1 to 12.
day	Any constant, variable, number or any expression that evaluates to a number. Range is 1 to 31.

ACTION Specifies or returns system date in the form: year, month, day.

> Be careful to observe correct day of month when setting.

SEE ALSO **SYSTem:TIME**

SYSTem:DEFaults (HOST Only)

USE To restore Test Set to default state.

SYNTAX **SYSTem:DEFaults**

ACTION Restores the Test Set to factory default state.

SEE ALSO **SYSTem:CURsor:DEFaults**, **SYSTem:DISPlay:DEFaults**

SYSTem:DISPlay:DEFaults (HOST Only)

USE To restore Test Set to default Color Set.

SYNTAX **SYSTem:DISPlay:DEFaults**

ACTION Restores the Test Set to the default Manufacturer Color Set.

SEE ALSO **SYSTem:CURsor:DEFaults**, **SYSTem:DEFaults**

SYSTem:ERRor?

USE To return number and description of system errors.

SYNTAX **SYSTem:ERRor?**

ACTION Returns earliest error not yet read. The error number and description is returned.

> Once read, the data for the error read is removed from memory. The earliest 16 errors that have not been read are held in memory with the rest being ignored. The 16 stored errors can be cleared from memory using the ***CLS** command.

SEE ALSO ***CLS**

SYSTem:FREQList (HOST Only)

USE To specify or return Frequency List settings.

SYNTAX
SYSTem:FREQList:GENerator *list_num,freq* or
SYSTem:FREQList:GENerator? *list_num* or
SYSTem:FREQList:OFFset *list_num,freq* or
SYSTem:FREQList:OFFset? *list_num* or
SYSTem:FREQList:RECeiver *list_num,freq* or
SYSTem:FREQList:RECeiver? *list_num* or
SYSTem:FREQList:SCAN *list_num,b* or
SYSTem:FREQList:SCAN? *list_num*

ELEMENT	DESCRIPTION
list_num	Any constant, variable, number or any expression that evaluates to a number. Range is 0 to 99.
freq	Any constant, variable, number or any expression that evaluates to a number. Range is 250.0 to 2010000.0.
b	Enable (1) or disable (0).

ACTION Specifies or returns Generator, Receiver, Offset and Scan enable settings. Specifies one of the 100 frequency listings (*list_num*). Frequencies (*freq*) are specified in kHz. The scan status (*b*) of each frequency listing can be turned on or off to be included or removed from frequency list scanning.

SYSTem:FREQuency:LOCK (HOST Only)

USE

To slave the RF Generator, Receiver and Spectrum Analyzer frequency together or to return frequency lock status.

SYNTAX

SYSTem:FREQuency:LOCK *b* or
SYSTem:FREQuency:LOCK?

ELEMENT	DESCRIPTION
b	Any constant, variable, number or any expression that evaluates to a number. Valid values: 1 (Enable) or 0 (disable).

ACTION

Enables/disables or returns the frequency lock state of the Receiver, Generator and Spectrum Analyzer RF lock feature.

- When user enters a frequency in any one of the above three operations modes, the remaining two operation modes automatically change to the identical frequency.
- Works only in Direct Mode.

SYSTem:KEY (HOST Only)

USE

To simulate or return a key pressed or simulated pressed of the Front Panel Keypad.

SYNTAX

SYSTem:KEY *n* or
SYSTem:KEY?

ELEMENT	DESCRIPTION
b	Any constant, variable, number or any expression that evaluates to a number. For valid keycodes see Appendix B.

ACTION

Simulates pressing a Front Panel Key with keycode *n* or returns the keycode of the last key pressed or simulated pressed. If a key has not been pressed, -1 is returned.

SEE ALSO

KEY?, SYSTem:KEY:DEFine

EXAMPLES

```
SYST:KEY 2112          // Simulates pressing RCVR MODE Key.
```

SYSTem:KEY:DEFine (HOST Only)

USE To assign command sequence to a Front Panel Key.

SYNTAX **SYSTem:KEY:DEFine** *n,sequence*

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number. For valid keycodes see Appendix B.
sequence	String variable or literal string surround by quotation marks (e.g., "Test_1; print Max; a=Set_value"). Limited to 80 string characters.

ACTION Assigns the specified command *sequence* string to key with keycode *n*. If the key is pressed, the command *sequence* is executed.

> A maximum of 16 keys can be assigned at any one time. Appendix A lists Front Panel Keys with predefined constants.

SEE ALSO **SYSTem:KEY:DELete**

EXAMPLES

```
SYST:KEY:DEF 8200, "Test_1" // Assigns macro Test_1 to the *
                            // (asterisk) DATA ENTRY Key.
SYST:KEY:DEL 8200           // Deletes the *-Test_1 assignment.
```

SYSTem:KEY:DELete (HOST Only)

USE To cancel a Front Panel keycode command sequence definition.

SYNTAX **SYSTem:KEY:DELete** *n*

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number. For valid keycodes see Appendix B.

ACTION Cancels the command sequence string assigned to keycode *n* with command: **SYSTem:KEY:DEFine**.

SEE ALSO **SYSTem:KEY:DEFine**

SYSTem:MILLIsec?

USE To return current value of the millisecond counter.

SYNTAX **SYSTem:MILLIsec?**

ACTION Returns the current value of the millisecond counter. Range of returned values is 0 to 4294967295 (2^{32} - 1).

Counter returns to zero approximately every 49.7 days.

SEE ALSO **SYSTem:TIME, TICKS?**

SYSTem:PLOT (HOST Only)

USE To select plotter output.

SYNTAX **SYSTem:PLOT:GPIB** or
SYSTem:PLOT:SERial

ACTION Selects GPIB or HOST RS-232 Connector for plotter output.

SEE ALSO **PSCREEN**, **SYSTem:COMMunicate:GPIB:PRINTer**, **SYSTem:COMMunicate:SERial**

SYSTem:PTHRough:GPIB (HOST Only)

USE To send out or receive a *string* through the GPIB Connector.

SYNTAX **SYSTem:PTHRough:GPIB** *string* or
SYSTem:PTHRough:GPIB *address,string* or
SYSTem:PTHRough:GPIB? or
SYSTem:PTHRough:GPIB? *address*

ELEMENT	DESCRIPTION
string	Any string variable or literal string surrounded by quotation marks (e.g., "VOLTS").
address	Any constant, variable, integer or any expression that evaluates to a integer. Range is 0 to 31.

ACTION When operating the Test Set as a GPIB Controller, sends out or waits for and accepts (receives) a *string* through the GPIB Connector. Optional peripheral *address* defaults to the address set by the last **SYSTem:COMMunicate:GPIB:SLAVe** command.

Strings contain commands unique to the controlled device.

SEE ALSO GPIB Operation (2-15), **SYSTem:COMMunicate:GPIB:SLAVe**

EXAMPLES See **SYSTem:COMMunicate:GPIB:SPOLL?**

SYSTem:PTHRough:SERial (HOST Only)

USE To send or receive a string through the RS-232 Connector.

SYNTAX **SYSTem:PTHRough:SERial** *"string"* or
SYSTem:PTHRough:SERial? or
SYSTem:PTHRough:SERial:KEY?

ELEMENT	DESCRIPTION
string	Any string variable or literal string surrounded by quotation marks (e.g., "VOLTS").

ACTION Sends or waits and receives a string (or, in the case of the (**SYSTem:PTHRough:SERial:KEY?** query, one character) through the HOST RS-232 Connector.

> Strings contain commands unique to the controlled device.

SEE ALSO **SYSTem:PTHRough:SERial:QUEue?**, **SYSTem:PTHRough:GPIB**

SYSTem:PTHRough:SERial:QUEue? (HOST Only)

USE To determine is RS-232 queue is empty or contains data.

SYNTAX **SYSTem:PTHRough:SERial:QUEue?**

ACTION Returns a non-zero value if data is in the RS-232 queue, 0 if the RS-232 queue is empty.

SYSTem:TIME

USE To set or return time in 24 hour time format.

SYNTAX **SYSTem:TIME** *hours, minutes, seconds* or
SYSTem:TIME?

ELEMENT	DESCRIPTION
hours	Any constant, variable, number or any expression that evaluates to a number. Range is 0 to 23.
minutes	Any constant, variable, number or any expression that evaluates to a number. Range is 0 to 59.
seconds	Any constant, variable, number or any expression that evaluates to a number. Range is 0 to 59.

ACTION Sets 24 hour time in hours, minutes and seconds or returns 24 hour time in hours, minutes, seconds and milliseconds.

> Although 0.1 milliseconds are displayed when returned, accuracy is only assured to within 16.5 milliseconds.

SEE ALSO **SYSTem:MILLIsec?**, **TICKS?**

EXAMPLES

```
SYST:TIME?
/* Sample time returned:  13,17,32.760 (1:17:32.760 PM)    */
```

TAB

USE To return string spaces (or tabs) to the right.

SYNTAX **TAB(*n*)**

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number.

ACTION Returns a blank string of *n* character spaces in length.

EXAMPLES

```
CLS                          // Starts with blank screen.
PRINT TAB(7),"FREQ = "       // Prints FREQ = starting seven
                             // spaces from the left edge of color
                             // display.
```

TASK

USE For Multitasking.

SYNTAX **TASK** *name*

ELEMENT	DESCRIPTION
name	String variable or literal string surrounded by quotation marks. Name of the macro to be declared a task.

ACTION Declares the specified macro as a task. A macro must be declared a task before it can be placed on the schedule queue for multitasking. The macro must already be loaded in memory.

SEE ALSO Multitasking Macros (2-12), **ACTIVATE**

EXAMPLES See Multitasking example (2-12).

TICKS?

USE To return elapsed time in milliseconds.

SYNTAX **TICKS?**

ACTION Returns the current value of the millisecond counter. Range of returned values is 0 to 4294967295 (2^{32} - 1).

> Used for calculating elapsed time in milliseconds.

SEE ALSO **SYSTem:MILLIsec?**

TPAUSE

USE For multitasking and sharing execution time with other Test Set activities.

SYNTAX **TPAUSE**

ACTION While multitasking, stops command execution and passes command execution to the next task in the schedule queue, if one exists.

SEE ALSO Multitasking Macros (2-12), **TSTOP**

EXAMPLES See Multitasking example (2-12).

TRUE

USE To produce a 1.

SYNTAX **TRUE** or
ON

ACTION Provides a concise method of specifying a logical 1.

SEE ALSO **!**

EXAMPLES

```
X=TRUE                  // Assign 1 to X.
```

TSTOP

USE For Multitasking.

SYNTAX **TSTOP**

ACTION While multitasking, stops command execution and removes the currently running task from the schedule queue and performs a **TPAUSE**. Command execution passes to the next task in the schedule queue.

SEE ALSO Multitasking Macros (2-12), **TPAUSE**

EXAMPLES See Multitasking example (2-12).

USER (Sp Tst Only)

USE To create a blank screen.

SYNTAX **USER**

ACTION Creates a blank user screen in the active window.

Blank screen may be used to create user-developed screens.

SEE ALSO **SCREEN:USER**

EXAMPLES

```
user                      // Displays blank user screen.
*wai                      // Forces macro execution to wait until
                          // user screen is completed.
cls                       // Initializes user screen.
bcolor 0                  // Establishes background color (user
                          // should establish background color
                          // of new screen).
```

VAL

USE To return the numeric value of a string.

SYNTAX **VAL(***string***)**

ELEMENT	DESCRIPTION
string	String variable or literal string surrounded by quotation marks.

ACTION Returns the numeric value of a character representation of a number in *string*.

SEE ALSO **ASC**, **CHR**, **STR**,

EXAMPLES

```
STRING Commd
Commd = "567"             // Assigns the string "567" to Commd.
C = VAL(Commd)            // Assigns the number 567 to C, a numeric
                          // variable.
```

VAR

USE To declare variables and arrays.

SYNTAX
VAR *name* or
VAR *name*, *name*, ... ,*name* or
VAR *name*[*index*] or
VAR *name*[*index*], *name*[*index*], ... ,*name*[*index*] or
VAR *name*, *name*[*index*]

ELEMENT	DESCRIPTION
name	Name of variable or array declared. The first character of the *name* is a letter while the remaining characters can be letters, digits or the underscore. The length of the *name* can be from 2 to 31 characters.
index	Any constant, variable, number or any expression that evaluates to a number. Number of array elements minus 1 (index = # of array elements - 1). Index of an array. Identifies the elements of the array by specifying the order of occurrence. The count of elements starts with 0.

ACTION Variables are declared with a *name*. Arrays are declared with a *name* and an *index*. Variables and arrays can be initialized when declared using an equals (=) character. Using a variable *name* previously used to declare a variable voids the previous variable.

Variables and arrays are either local or global. Local variables and arrays are declared inside a macro and have no meaning outside that macro. Global variables and arrays are declared outside macros and are used inside or outside any macro.

SEE ALSO Numeric Variables and Arrays (2-3), **NVSAV**

EXAMPLES

```
VAR Flag=0,Set_Squ=0.9     // Variables Flag and Set_Squ are
                           // declared and initialized to 0
                           // and 0.9.
VAR Dat[2]={3.2,3.4,3.6}   // Array Data is declared with
                           // 3 elements and each element is
                           // initialized to a value.
```

VIDEOpage:COPY (HOST Only)

USE

To transfer the video image from one video page to another.

SYNTAX

VIDEOpage:COPY *from,to*

ELEMENT	DESCRIPTION
from	Any constant, variable, number or any expression that evaluates to a number. Range is 0 to 1.
to	Any constant, variable, number or any expression that evaluates to a number. Range is 0 to 1.

ACTION

Copies video from the active screen to the static screen or from the static screen to the active screen.

- Two video pages (screens) are available: video page 0 and video page 1.
- This command is used in combination with **VIDEOpage:SET** to build screens in the background, then display new screen, instantly, and for animation applications.

SEE ALSO

VIDEOpage:SET

EXAMPLES

```
video:set 0,1          // Makes default screen 0 to continue to
                       // be displayed and screen 1 to be active
                       // (or be written to) but remain in
                       // background.

/* Build screen 1 by sending print commands, etc., as would
   normally.  However, screen results do not appear.        */

video:copy 1,0         // Copies the video from screen 1 in the
                       // background to screen 0 presently being
                       // displayed.  Changes made to screen 1
                       // is displayed immediately.
```

VIDEOpage:SET (HOST Only)

USE To specify Video Page settings.

SYNTAX **VIDEOpage:SET** *disp,write*

ELEMENT	DESCRIPTION
disp	Any constant, variable, number or any expression that evaluates to a number. Range is 0 to 1.
write	Any constant, variable, number or any expression that evaluates to a number. Range is 0 to 1.

ACTION Specifies which Video Page to display and which Video Page is active (may be written to); the other screen remains static. Two video pages are available: video page 0 and video page 1. See Table 3-9.

disp	*write*	ACTION
0	0	(Default at power up) Screen 0 is displayed; Screen 0 is active. Screen 1 remains static.
0	1	Screen 0 is displayed; Screen 1 is active. Screen 0 remains static.
1	0	Screen 1 is displayed; Screen 0 is active. Screen 1 remains static.
1	1	Screen 1 is displayed; Screen 1 is active. Screen 0 remains static.

Table 3-9 Video Page Settings

Two screen displays are available (screen 0 and screen 1): one active and one static for use as pages to switch back and forth. Two video pages allow user to construct or make changes to a screen display before actually displaying the new screen. Video Page Control allows user to specify which video page to currently display or to copy from one to the other.

SEE ALSO **VIDEOpage:COPY**

WAKE

USE For Multitasking.

SYNTAX **WAKE** *name*

ELEMENT	DESCRIPTION
name	String variable or literal string surrounded by quotation marks. Name of the task put back on the schedule queue.

ACTION Reenters a task (taken off the schedule queue by a **SLEEP** command) into the schedule queue.

SEE ALSO Multitasking macros (2-12), **SLEEP**

EXAMPLES See Multitasking examples (2-12).

WCLOSE

USE To close a window.

SYNTAX **WCLOSE** *n*

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number. Number of the window to close. Windows are numbered in the order they are created. Overlapping windows must be closed in the reverse order they are opened.

ACTION Closes (deletes) a generated window specified by *n*.

SEE ALSO Windows (2-13-3), **WOPEN**, **WINDOW?**

EXAMPLES See Windows examples (2-13-3).

WHILE WEND

USE To perform a set of commands repetitiously while a conditional expression is true.

SYNTAX **WHILE** *condition*
sequence
WEND

ELEMENT	DESCRIPTION
condition	Any expression which is TRUE or FALSE when evaluated.
sequence	Command sequence performed as long as *condition* is true.

ACTION The **WHILE** loop repeatedly executes command *sequence* until condition is evaluated FALSE. If *condition* is FALSE when first evaluated, the command sequence is not executed once like the **DO** loop, because the condition is tested before the command *sequence*.

EXAMPLES

```
N=0                    // Assigns 0 to N.
WHILE N!=2             // Evaluates Condition.
   N=N+1               // Performs command while condition is TRUE.
WEND                   // End of WHILE loop.

/* While performs the following steps:
   o  N is compared to 2 (N=0).
   o  1 is added to N.
   o  N is compared to 2 (N now equals 1).
   o  1 is added to N.
   o  N is compared to 2 (N now equals 2).
   o  N skips over commands inside loop and commands are
      not executed.  Macro execution continues after WEND
      command.                                              */
```

(Examples continued on next page.)

(Examples continued from previous page.)

```
VAR FF
WHILE FF<=LN(2*S)
   FF=FF<<2
   PRINT FF            // The value of FF is shifted left 2 bits
                       // and printed until it is greater than
                       // LN(2*S).
WEND

WHILE SYSTEM:KEY? != 32896 // Continues looping until the
   command                 // Front Panel Key with a keycode
   .                       // of 32896 is pressed.
   .
WEND
```

WINDOW?

USE To return the number of the window currently selected.

SYNTAX **WINDOW?**

ACTION Returns the number of the currently selected window. The selected window is the window affected by the **WMOVE** and **WCLOSE** commands. The last opened window is the selected window unless a **WSEL** command has selected another window.

SEE ALSO Windows (2-13-3), **WSEL**, **WMOVE**, **WCLOSE**

EXAMPLES See Windows examples (2-13-3).

WMOVE

USE To move windows.

SYNTAX **WMOVE** *x*,*y*

ELEMENT	DESCRIPTION
x,*y*	Both either a constant, variable, number or any expression that evaluates to a number. Ranges are 0 to 639 and 0 to 349. New coordinates on the color display of the top left corner of the window to be moved.

ACTION Moves the currently selected window so that the top left corner of the window is located at the specified point.

> Window size and shape are unaffected.

> Moving a window that overlaps another window disrupts the window overlapped.

SEE ALSO Windows (2-13-3), **WSEL**, **WINDOW?**

EXAMPLES See Windows examples (2-13-3).

WOPEN

USE To create windows.

SYNTAX **WOPEN** *c,x1,y1,x2,y2*

ELEMENT	DESCRIPTION
c	Any constant, variable, number or any expression that evaluates to a number. Range is 0 to 15. Number of the color selected for the window. See Table 2-2.
x1,y1	Both either a constant, variable, number or any expression that evaluates to a number. Ranges are 0 to 639 and 0 to 349. Color display coordinates of the top left corner of the window being opened.
x2,y2	Both either a constant, variable, number or any expression that evaluates to a number. Ranges are 0 to 639 and 0 to 349. Color display coordinates of the bottom right corner of the window being opened.

ACTION Creates a window on the color display of color *c* with a top left point of *x1*, *y1* and a bottom right corner of *x2*, *y2*. Height and width of window are forced to 16 pixel increments.

A lack of available memory can cause a window not to open.

SEE ALSO Windows (2-13- 3), **WCLOSE**

EXAMPLES See Windows examples (2-13-3).

WSEL

USE To change which window is currently selected.

SYNTAX **WSEL** *n,hide*

ELEMENT	DESCRIPTION
n	Any constant, variable, number or any expression that evaluates to a number.
hide	Any constant, variable, number or any expression that evaluates to a number. Valid values: 1 (Hide window) or 0 (show window).

ACTION Makes window *n* the currently selected window and hides window if *hide* is set to 1.

- The selected window is the window affected by the WMOVE and WCLOSE command.
- Windows are numbered in the order they are opened.
- A hidden window does not appear on the color display and must be closed and opened again to reappear.

SEE ALSO Windows (2-13-3), **WINDOW?**, **WMOVE**, **WCLOSE**

EXAMPLES See Windows examples (2-13-3).

XY

USE To specify a point on the color display for other display commands to use.

SYNTAX **XY** *horiz,vert*

ELEMENT	DESCRIPTION
horiz	Any constant, variable, number or any expression that evaluates to a number. Range is 0 to 639.
vert	Any constant, variable, number or any expression that evaluates to a number. Range is 0 to 349.

ACTION Specifies a point (one pixel) on the color display to be used by the **PRINT**, **PIXEL**, **HPRINT** and **ICON** commands. Where point 0,0 is the top left corner of the color display, positive *vert* (y coordinate) is the number of pixels in the downward direction and positive *horiz* (x coordinate) is the number of pixels to the right.

The **PRINT** command moves the xy position to a new line using the current **HEIGHT** setting for the height of the new line.

SEE ALSO Color Display (2-13-2), Graphics (2-13-5)

EXAMPLES See Graphics example (2-13-5).

XYPRINT

USE

To print on the display screen.

SYNTAX

XYPRINT *x,y, argument* or
XYPRINT *x,y, argument, argument* , ... ,*argument,* or
XYPRINT *x,y,* %0*nl, argument,* ..... ,*argument, or*
XYPRINT *x,y, argument,* ..... ,%0*nl, argument,*

ELEMENT	DESCRIPTION
x	Any constant, variable, number or any expression that evaluates to a number. Range is 0 to 639.
y	Any constant, variable, number or any expression that evaluates to a number. Range is 0 to 349.
argument	Expression to be printed. Can be a mathematical expression, a literal string or the contents of variables. Mathematical expressions are calculated and string functions are performed before the results are printed.
0	Zero signifies leading zeros are added to fill the field width. The 0 is omitted for no leading zeros. Applies only to numeric expressions.
n	Specifies field width. *n* is the minimum number of digit spaces (decimal point counts as one). Applies only to numeric expressions. If *l* is D, *n* can be in the form of *b.a*, where *b* designates minimum number of digits before the decimal point and *a* designates the maximum number of digits after the decimal point.
l	Specifies the numeric base the data is printed in. Applies only to numeric expressions. Omit for a default of Signed Decimal or enter one of the following:

B	Binary
Q	Octal
H	Hexadecimal
U	Unsigned decimal
D	Signed decimal

ACTION

Combines the **XY** and **HPRINT** commands. Prints numerical and string values at the selected *x,y* location on the color display. Several values are printed on the same line by separating them with commas. If an *argument* is a variable, the value of the variable is printed. Mathematical and string expressions are calculated before printing.

%0*nl* contains the format settings for numeric expressions and can be placed anywhere in the **XYPRINT** command. There can be several format settings in each **XYPRINT** command, each one changing the leading zero format, the numeric base and the field width for the expressions that follow. Format settings do not affect the printing of strings.

SEE ALSO

PRINT, **PPRINT**, **PSCREEN**

EXAMPLES

```
U=44
T=37.83
STRING NOT
NOT="MOBILE PHONE DOES NOT RESPOND"

XYPRINT 5,5,%04,U," ",NOT  // Contains a format setting
                 // specifying leading zeros and a width of
                 // four characters.  Signed decimal is the
                 // numeric base by default.  The command
                 // prints 0044 MOBILE PHONE DOES NOT RESPOND
                 // starting at point 5, 5 location on the
                 // color display.

XYPRINT 319,174,U,%3," ",T // Prints U as is starting in the
                           // middle of the screen.  Then
                           // specifies no leading zeros, a
                           // field width of 3 and signed
                           // decimal (by default) for any
                           // following numerical outputs.
                           // A single space is printed.
                           // When T is printed, the print field
                           // is expanded to accommodate the
                           // value of T.  Because each
                           // argument of the PRINT command
                           // (except for T) is following by a
                           // comma, output is displayed on one
                           // line.  The following is printed:
                           // 44 37.83

XYPRINT 0,0,%4B,U     // Specifies no leading zeros, a field
                      // width of 4 and Binary numeric base for
                      // any following numerical outputs.  The
                      // following is printed in the upper left
                      // corner of the screen:  101100.
                      // The field width is enlarged to
                      // accommodate U.

XYPRINT 30,55,NOT,%7,T    // Specifies no leading zeros, a field
                 // width of 7 and signed decimal (by default)
                 // for any following numerical outputs.
                 // The following is printed starting at xy
                 // location (30,55):
                 // MOBILE PHONE DOES NOT RESPOND  37.83.
                 // 2 blanks character spaces before 3 are
                 // provided by the field width of 7.

XYPRINT 319,174,%3.1,T    // Specifies no leading zeros, a
                          // minimum field width of 3 with a
                          // maximum of 1 digit after the
                          // decimal point.  Displays 37.8
                          // starting in the middle of the
                          // screen.  The field width of 3 was
                          // expanded to fit T, counting the
                          // decimal point.
```

SECTION 4 - CREATING AND UPLOADING TMAC PROGRAMS

4-1 GENERAL

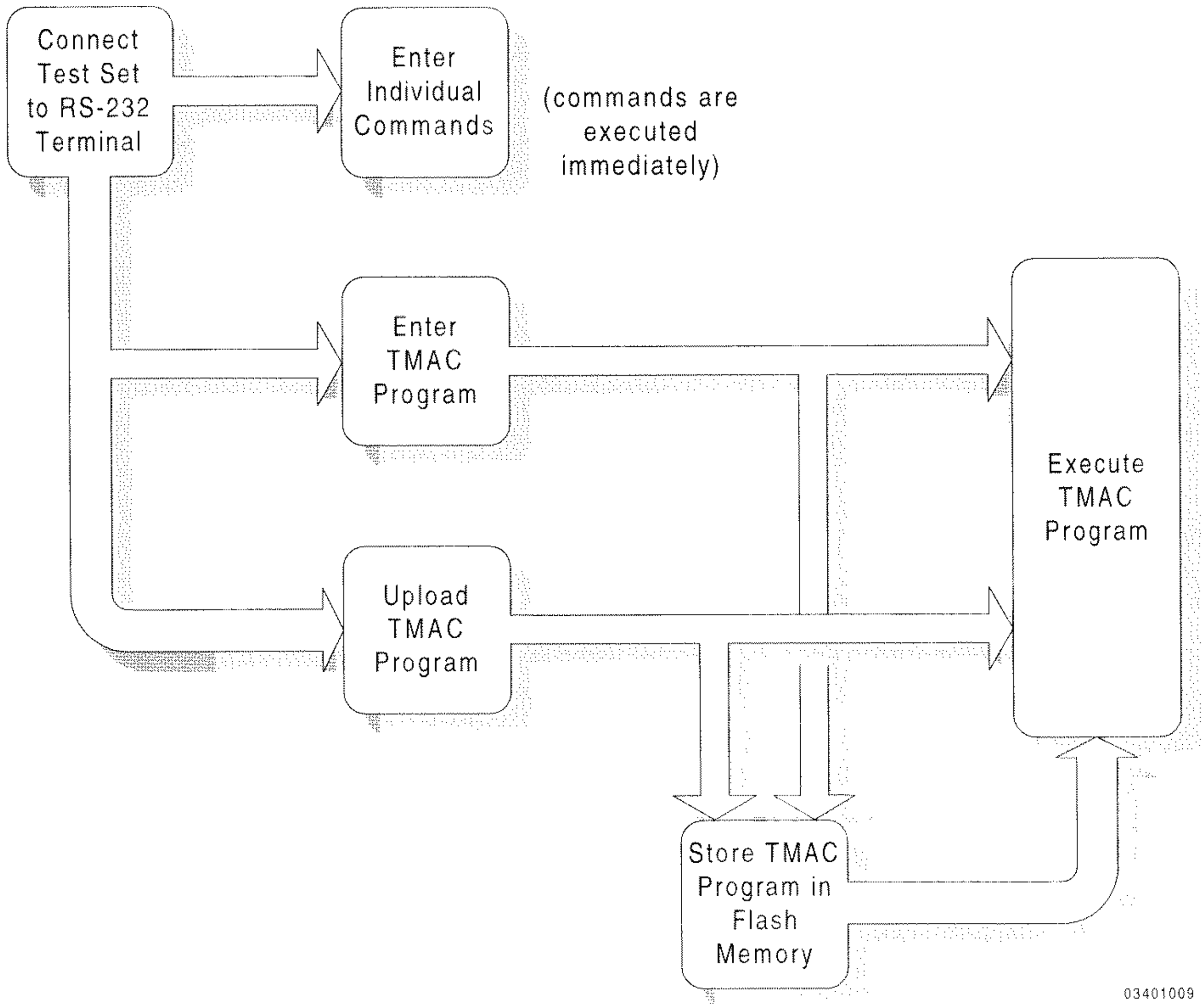


Figure 4-1 Operating IFR-1900 CSA Via RS-232

This section explains the following procedures:

- How to establish communication between an RS-232 Terminal (typically a PC with a terminal emulation program) and the IFR-1900 CSA.
- How to create a TMAC program.
- How to upload a TMAC program into the IFR-1900 CSA working memory (RAM).
- How to execute the TMAC program.
- How to store the uploaded TMAC program into Flash Memory.
- How to execute a TMAC program stored in Flash Memory.

4-2 SETUP

4-2-1 RS-232 TERMINAL EMULATION PROGRAM SETUP

For the purpose of clarity and actual operation, the following procedures use Procomm Plus™ as the RS-232 Terminal Emulation program.

STEP	PROCEDURE

1. Enter Procomm Plus™ from RS-232 terminal.
2. Set Line Settings as follows:

PARAMETER	SETTING
Baud Rate	300 to 115200
Parity	NONE
Data Bits	8
Stop Bits	1
Port	(Comm Connector on PC used for RS-232 to HOST or Sp Tst)

3. Set Terminal Options as follows:

PARAMETER	SETTING
Terminal emulation	ANSI
Duplex	FULL
Soft flow ctrl (Xon/Xoff)	ON
Hard Flow ctrl (RTS/CTS)	OFF
Line wrap	ON
Screen scroll	ON
CR translation	CR
BS translation	NON-DESTRUCTIVE
Break length (millisecs)	N/A
Enquiry (ENQ)	N/A
EGA/VGA true underline	N/A
Terminal width	80
ANSI 7 or 8 bit commands	8 Bit

4. Set ASCII Protocol Options as follows:

PARAMETER	SETTING
Echo locally	NO
Expand blank lines	NO
Expand tabs	NO
Character pacing (millisec)	1
Line pacing (1/10 sec)	1
Pace character	0
Strip 8th bit	NO
ASCII download timeout	N/A
CR translation (upload)	NONE
LF translation (upload)	STRIP
CR translation (download)	N/A
LF translation (download)	N/A

4-2-2 HOST SETUP

STEP PROCEDURE

1. Perform "Remote Operation Using Host System (RS-232)" configuration procedure in Section 6 of the IFR-1900 Operation Manual (1002-3402-200). (Make sure Echo is toggled On.)
2. Verify communication with HOST by pressing Enter key on RS-232 terminal keyboard and verify OK and exclamation mark appears on RS-232 terminal monitor.

4-2-3 SPECIAL TEST SETUP

STEP PROCEDURE

1. Perform "Configuring for RS-232 Operation" configuration procedure described in Appendix D, Remote Operation, in the IFR-1900 CSA Option Operation Manual (1002-3403-200).
2. Verify communication with Special Test by pressing Enter key on RS-232 terminal keyboard and verify question mark appears on RS-232 terminal monitor.

4-3 CREATING A TMAC PROGRAM

STEP PROCEDURE

1. Write TMAC program using RS-232 terminal text editor. Refer to examples in Sections 2, 3 and 4 (e.g., Figure 4-2).
2. Save as text file (program file).

4-4 UPLOADING TMAC PROGRAM

Uploading a TMAC program, places the program into working memory (RAM) for immediate execution or for storing into Flash Memory.

STEP PROCEDURE

1. Enter Procomm Plus™ from RS-232 terminal.
2. Issue *PMC command from RS-232 terminal to purge all existing files (TMAC programs).

The user should include the *PMC command at the beginning of all TMAC programs.

3. Press PgUp key on RS-232 terminal and select ASCII upload protocol.
4. Type in path and file name of text file (TMAC program) written in 4-3, and press Enter key.
5. If failure occurs:
   - Utilize Procomm Plus™ Log function to create separate text file (log file) of Upload function (program file with error messages).
   - Repeat Steps 1 through 4.
   - View log file and correct errors in original program file.
   - Repeat Steps 1 through 4.

4-5 EXECUTING TMAC PROGRAM

After TMAC program is uploaded (or manually entered) into working memory (RAM), perform the following:

STEP PROCEDURE

1. Enter name of macro that is the entry point for TMAC program.
2. Press Enter key of RS-232 Terminal keyboard to execute program.

```
/*  MACRO NAME:  minit1

        SYNTAX:  minit1

       PURPOSE:  Perform a Mobile Station originated call.

         MACRO:  (The following example uses the predefined constants F1
                 and F6 [see Appendix A].)                             */

*dmc "minit1",begin
var done=0
var key_code
do
  css:conf:user
  freq:band 1
  css:chan 334
  css:rflvl -60
  css:start
  css:call:type 0
  host ":keypad:soft"
  color 1,15
  keypad:label 1,"START"
  keypad:label 6,"Ret"
  color 1,11
  center "Mobile Init Call",0,5,640
  do
    key_code=val(host? "syst:key?")
  until (key_code=F6) or (key_code=F1)            // Waits for valid key.
  if key_code=6
    done=1
  else
    keypad:label 1,"STOP"
    keypad:label 6,""                             // Erases F6 label.
    css:call:proc:mobinit
    center "Place Call",0,35,640
    do
      key_code=val(host? "syst:key?")
    until (key_code=F1) or css:recc:status?
    erase:text 0,35,640
    center "Assign Channel",0,35,640
    if(key_code!=F1)
      css:call:proc:assign
      do
        key_code=val(host? "syst:key?)
      until (key_code=F1) or (meas:sat? > 5960)   // Waits for somewhat valid SAT.
      if(key_code!=F1)
        erase:text 0,35,640
        center "Call Successfull",0,35,640
      endif
    endif
    while key_code!=F1
      key_code=val(host? "syst:key?")
    wend
  endif
until done
end
```

Figure 4-2 Example TMAC Program

4-6 STORING UPLOADED TMAC PROGRAM INTO FLASH MEMORY

Storing an uploaded TMAC program into flash memory allows TMAC program to remain in Test Set though power is removed or a ***PMC** (Purge Macro) command is issued.

Use **MMEMory:STORe:MACRo** "*m*","*f*" command to store TMAC program (uploaded in 4-4, *plus any existing program files uploaded since last issue of *PMC or power-up*) into Flash Memory:

- Set *m* to name (See Figure 4-3) of macro to be designated as entry point of uploaded TMAC program to run from Front Panel of the HOST.
- Set *f* to desired Flash File name (i.e., *f* is the name to appear in the Flash File Directory) of TMAC program.

Figure 4-4 shows the relationship between the macro name and the Flash File name displayed in Flash File Directory from the front panel of the HOST.

Refer to 6-15 (HOST) or 9-13 (Sp Tst) for more information on Flash Memory commands.

Examples:

```
mmem:store:macro "minit1","minit1" // Stores file, written in 4-3 and
                                   // and uploaded in 4-4, into Flash
                                   // Memory.

mmem:store:macro "sense_main","s_test"
                                   // Stores file from 4-8 into Flash
                                   // Memory.
```

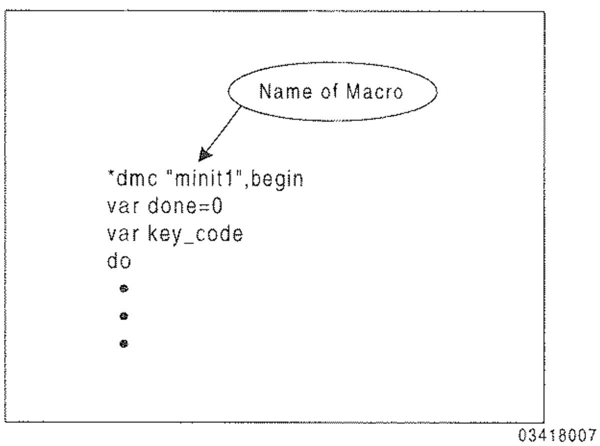


Figure 4-3 Macro Name Identification

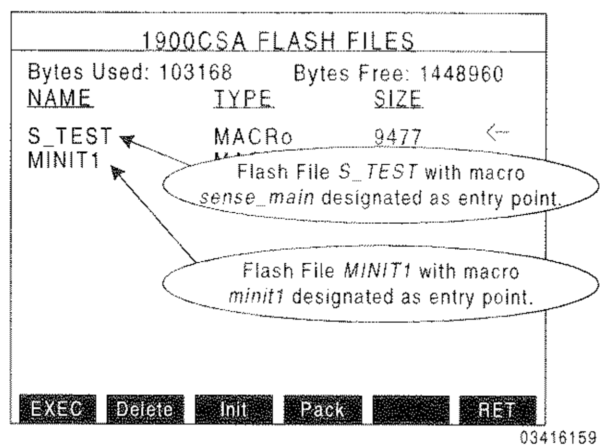


Figure 4-4 Flash File and Macro Name Relationship

4-7 EXECUTING TMAC PROGRAM STORED IN FLASH MEMORY

4-7-1 FROM RS-232 TERMINAL

Issue the following command from RS-232 terminal:

mmem:load:macr "*m*","*f*"

Where: *m* is the name of the macro to be executing, and *f* is the file name of the program file stored in Flash Memory.

If *m* is *, then the macro previously designated by the **mmem:stor:macr** command as the entry point is executed.

4-7-2 FROM HOST FRONT PANEL

A. HOST TMAC Programs

Refer to the description of "User Program" in Section 3-3-10 in the IFR-1900 Operation Manual for further explanation of the following:

STEP	PROCEDURE
1.	Select desired Flash File from Flash Memory File Directory.
2.	Execute selected Flash File (macro).

B. Special Test Programs

Refer to "User Files" in Appendix C in the IFR-1900 CSA Option Operation Manual for a detailed explanation of the following:

STEP	PROCEDURE
1.	Select desired Flash File from 1900CSA Flash Files Directory (see Figure 4-4).
2.	Execute selected Flash File (macro).

4-8 EXAMPLE TMAC PROGRAM WITH MULTIPLE MACROS

The example TMAC program in this section applies only to the Special Test.

Use ***MMEM:STOR:MACR*** *"sense_main","s_test"* to store the following TMAC program file into Flash Memory (after being uploaded).

The following TMAC program uses the predefined constants F1 and F6 (see Appendix A). TMAC program is available on the IFR BBS (316-524-0270).

```
/*    FILE NAME: stest.mac
        PURPOSE: Demonstrate several capabilities of Sp Tst TMAC including:
                 o  Sp Tst TMAC Graphics
                 o  Cell Site Simulation
                 o  Using multiple macros
                 o  Storing macros in Flash Memory
                 o  Running a TMAC program from the Front Panel.
                 o  Using HOST TMAC commands through the HOST command.
                                                                        */

var key_code                  // Declares a global variable.


/*         MACRO: delay1
          SYNTAX: delay1 n
                  (n=delay time in ms.)
         PURPOSE: Provide a delay if the F1 Soft Function Key is not
                  pressed.  */

*dmc "delay1",begin
if (key_code!=F1)
  delay $1
endif
end


/*         MACRO: focc_setup
          SYNTAX: focc_setup
         PURPOSE: Set up the System Parameter Overhead message sent on the
                  Forward Control Channel.  */

*dmc "focc_setup",begin
css:conf:user                 // Sets Sp Tst for Cell Site Simulation.
freq:band 1                   // Sets operating band to U8 (800 MHz).
css:chan 333                  // Sets Forward Control Channel to 333 (879.99 MHz).
css:rflvl -60.0               // Sets output RF Level to -60.0 dBm.
css:focc:pci 1                // Sets Protocol Capability Indicator.
css:focc:rcf 0                // Sets Read Control Filler bit.
css:focc:sid 0                // Sets System Identification Number.
css:focc:n 1                  // Sets Number of Paging Channels.
css:focc:cmax 1               // Sets Number of Access Channels.
css:focc:auth 0               // Sets authentication bit.
css:start
end
```

(*s_test.mac program file continues on following page.*)

```
/*        MACRO:  call_setup
         SYNTAX:  call_setup
        PURPOSE:  Set up parameters used in making a digital call to the
                  Mobile Station (cellular phone).   */

*dmc "call_setup",begin
css:call:type 1                       // Sets call type to digital.
css:call:chan 10                      // Sets Mobile Digital Traffic Channel
                                      // assignment.
css:call:dmac 5                       // Sets Digital Mode Attenuation Code.
css:call:slot 2                       // Sets to Timeslot 2.
css:call:pm 0                         // Sets Privacy Mode bit.
css:call:mem 0                        // Sets Message Encryption Mode to 0.
css:call:ef 0                         // Sets Extended Protocol Forward Channel
                                      // Indicator.
css:fdtc:enable:signal 1              // Enables Signal field.
css:fdtc:signal:pitch 0;cadence 1     // Sets pitch and pattern of Alert tone.
css:fdtc:enable:calling:num 1         // Enables Calling Party Number field.
css:fdtc:calling:type 0               // Sets Calling Party Type.
css:fdtc:calling:plan 0;pi 0;si 0     // Sets Calling Party Numbering Plan
                                      // Identification, Presentation Indicator
                                      // and Screening Indicator.
css:fdtc:enable:dmac 0;ta 1;dtx 0;dic 0
                                      // Disables Digital Mobile Attenuation
                                      // Code, Discontinuous Transmission bit
                                      // and Delay Interval.  Enables Time
                                      // Alignment field.
css:fdtc:ta 2                         // Sets Time Alignment to 2.
end

/*        MACRO:  registration
         SYNTAX:  registration
        PURPOSE:  Force Mobile Station to register.   */

*dmc "registration",begin
center "Registering Mobile",0,35,640
call_setup
do
  css:call:proc:reg
  delay1 1000
  key_code=val(host? "syst:key?")
until (key_code=F1) or css:recc:status?
print "MIN of mobile is ",recc:min?      // Prints information out
print "ESN of mobile is "%h,recc:esn?    // OPT. RS-232 Connector.
end
```

(s_test.mac program file continues on following page.)

```
/*        MACRO:  page1
         SYNTAX:  page1
        PURPOSE:  Page Mobile Station until Page Response is received.   */

*dmc "page1",begin
erase:text 0,35,640
center "Paging Mobile",0,35,640
css:call:proc:page
do
  delay1 1000
  key_code=val(host? "syst:key?")
until (key_code=F1) or css:recc:status?
erase:text 0,35,640
center "Page Response Received",0,35,640
end

/*        MACRO:  assign
         SYNTAX:  assign
        PURPOSE:  Assign Mobile Station to a digital voice channel.  */

*dmc "assign",begin
css:call:proc:assign
delay1 1000                        // Waits for channel change.
erase:text 0,35,640
center "Digital Voice Channel Assigned",0,35,640
end

/*        MACRO:  plc
         SYNTAX:  plc
        PURPOSE:  Send Physical Layer Control message to the Mobile Station
                  until a Physical Layer Control Ack message is received. */

*dmc "plc",begin
do
  css:fdtc:facch:plc
  delay1 500
  key_code=val(host? "syst:key?")
until((key_code=F1) or (rdtc:facch:msg? = "PLC ACK"))
erase:text 0,35,640
center "Physical Layer Control",0,35,640
end
```

(s_test.mac program file continues on following page.)

```
/*        MACRO:  alert
         SYNTAX:  alert
        PURPOSE:  Send Alert message to Mobile Station until Alert with Info
                  Ack is received.  Wait until Connect message is received.
                                                                          */
*dmc "alert",begin
do
  css:fdtc:facch:alert
  delayl 500
  key_code=val(host? "syst:key?")
until((key_code=F1) or (rdtc:facch:amt? = "ALERT"))
erase:text 0,35,640
center "Answer Phone",0,35,640
while !(key_code=F1) and (rdtc:facch:msg? != "CONNECT")
  key_code=val(host? "syst:key?")
wend
erase:text 0,35,640
center "Connect",0,35,640
end
```

(s_test.mac program file continues on following page.)

```
/*        MACRO:  sense
         SYNTAX:  sense
        PURPOSE:  Vary the output RF Level of the Sp Tst until the Mobile
                  Station (phone) reports a BER of 2% to 4%.   */

*dmc "sense",begin
var timeout,ber
i=0
j=0
erase:text 0,35,640
center "Finding Sensitivity of Mobile",0,35,640
do
  css:fdtc:facch:meas                    // Sends Measurement Order until
  delay1 400                             // Measurement Order Ack message is
  key_code=val(host? "syst:key?")        // received.
until (key_code=F1) or (rdtc:facch:msg?="MEAS ACK")
css:rflvl -90                            // Starts with RF Level set at -90 dBm.
delay1 3000
do
  timeout=25
  css:rflvl -90 - i
  erase:text 0,140,640
  xyprint 40,140,"RFLVL: ",%d,-90 - i
  while ((rdtc:sacch:msg? != "CHAN QUAL1") and (--timeout))
    delay1 100
  wend
  ber=rdtc:sacch:ber?
  erase:text 0,160,640
  xyprint 40,160,"BER: ",%d,ber
  if ber=0                               // BER < 0.01%
    i=i+2
  elif ber=1                             // 0.01 < BER < 0.1
    i=i+1
  elif ber=2                             // 0.1 < BER < 0.5
    i=i+.2
  elif ber=3                             // 0.5 < BER < 1.0
    i=i+.1
  elif ber=4                             // 1.0 < BER < 2.0
    i=i+.1
  elif ber=5                             // 2.0 < BER < 4.0
    j=j+1
  elif ber=6                             // 4.0 < BER < 8.0
    i=i-.1
  else                                   // BER > 8.0
    i=i-.2
  endif
  key_code=val(host? "syst:key?")
until j=6 or !(timeout) or key_code=F1   // Continues when BER=5 is found
                                         // 6 times, timeout occurs or
                                         // F1 is pressed.
```

(s_test.mac program file and sense macro continues on following page.)

```
if key_code != F1                                    // Checks for STOP.
  print "Sensitivity of Mobile is ",%4d,-90-i        // Prints information out
  print ""                                           // OPT. RS-232 Connector.
  erase:text 0,35,640
  center "Test Done, Press End or STOP",0,35,640
  while (key_code != F1) and (rdtc:facch:msg? != "RELEASE")
    key_code=val(host? "syst:key?")
    delay1 100
  wend
  erase:text 0,35,640
  center "Test Completed",0,35,640
endif
css:stop
delay 500
end
```

(*s_test.mac program file continues on following page.*)

```
/*        MACRO:  sense_main
         SYNTAX:  sense_main
        PURPOSE:  Provide the main calling routine for the sensitivity test.
                                                                              */
*dmc "sense_main",begin
var done=0
do
  focc_setup
  user
  host ":keypad:soft"
  color 1,15
  keypad:label 1,"START"
  keypad:label 6,"Ret"
  color 1,11
  center "Sensitivity Test",0,5,640
  do
    key_code=val(host? "syst:key?")
  until (key_code=F6) or (key_code=F1)    // Waits for valid key.
  if key_code=F6
    done=1                                // Goes to end if Ret key is pressed.
  else
    keypad:label 1,"STOP"
    keypad:label 6,""                     // Erases F6 label.
    registration
    if(key_code != F1)
      page1
    endif
    if(key_code != F1)
      assign
    endif
    if(key_code != F1)
      plc
    endif
    if(key_code != F1)
      alert
    endif
    if(key_code != F1)
      delay 3000
    endif
    if(key_code != F1)
      sense
    endif
  endif
until done
end
```

(end of s_test.mac program file.)

SECTION 5- HOST SPECIFIC TMAC QUICK REFERENCE LIST

This Quick Reference List is a brief listing of the Specific commands used with the HOST. The Quick Reference List is an aid to the experienced TMAC user. If more detailed information is needed, refer to the specified page.

> Effort has been made to arrange all commands alphabetically; therefore some headings which are more descriptive than the actual commands, may appear out of alphabetical order.

COMMAND	RANGE	PAGE	DESCRIPTION
ACCESSORY COMMANDS			
ACCessory:			
INIT		6-123	Initializes MIC/ACC Connector prior to conducting communication. PTT (Press to Talk) is turned On (low).
IO		6-123	Sends/receives data through MIC/ACC Connector.
RELease		6-123	"Releases" MIC/ACC Connector. PTT is turned Off (high).
STATe?		6-123	Checks presence of MCC/ACC I/O device. 1 = device present. 0 = none.
AF GENERATOR COMMANDS			
See Function Generator Commands.			
AF LEVEL METER COMMAND			
See M_VRMS?			
AF METER COMMANDS			
See M_AF:			
ANALYZER (SPECTRUM ANALYZER) COMMANDS			
ANLZ:			
AVErage [*n*]	1 to 100 (100, Default)	6-79	Selects Average Mode for Analyzer using *n* samples.
CHANnel *n*	1 to 2047, depending on format	6-79	Sets RF Frequency to cellular channel.
CHANnel:			
BAND?		6-80	Returns band for Analyzer Channel Format. Returns one of the following bands for the associated channel format: (NADC) U8, U4 or HY; (ETACS) NOT AVAILABLE; (NAMPS) LOWER, MIDDLE or UPPER.
FORMat:			
AMPS:			
FORward		6-79	Selects AMPS (NADC-U8) Forward channels.
REVerse		6-79	Selects AMPS (NADC-U8) Reverse channels.
ETACS:			
FORward		6-79	Selects ETACS Forward channels.
REVerse		6-79	Selects ETACS Reverse channels.
NADC:			
BAND:*xx*	U8, U4 or HYper	6-79	Selects NADC band.
FORward		6-79	Selects NADC Forward channels.
REVerse		6-79	Select NADC Forward channels.
NAMPS:			
BAND:*x*	Lower, Middle or Upper	6-80	Selects NAMPS Narrow Analog channel designator.
FORward		6-80	Selects NAMPS Forward channels.
REVerse		6-80	Selects NAMPS Reverse channels.
NT400:			
FORward		6-80	Selects NT400 Forward channels.
REVerse		6-80	Selects NT400 Reverse channels.
FORMat?		6-80	Returns Channel Format (NADC:FORWARD, NADC:REVERSE, ETACS:FORWARD, ETACS:REVERSE, NAMPS:FORWARD or NAMPS:REVERSE).

COMMAND	RANGE	PAGE	DESCRIPTION
ANLZ:			
COMPare *n*	1 to 9	6-81	Selects Compare Mode for Analyzer. Trace stored at *n* memory location is compared with Live Trace.
FIND:			
FREQuency?		6-81	Returns frequency of first signal with amplitude larger than Find reference level. Returns 0 if no signal is found.
REFerence *n*	(varies)	6-81	Sets Find reference level in dB.
REFerence?		6-81	Returns Find reference level in dB.
FREQuency *f*	250.0 to 2010000.0	6-81	Sets Analyzer Frequency in kHz.
FREQuency?		6-81	Returns Analyzer Frequency in kHz.
FULL		6-81	Selects full size Analyzer display for RF Generator, Receiver and Duplex Operation Screens.
INPut:			
ANTenna		6-82	Selects ANTENNA IN Connector for Analyzer Input.
ATTenuation *n*	0, 5, 10, 15, 20, 25, 30	6-82	Sets Analyzer Input Attenuation in dB.
ATTenuation:			
LNA		6-82	Sets Analyzer Input Attenuation to LNA.
LNA?		6-82	Returns current state of the Low Noise Amplifier. Returns 1 if LNA is selected; 0 otherwise.
ATTenuation?		6-82	Returns the current value of Input Attenuation.
TR		6-82	Selects T/R Connector for the Analyzer Input.
INPut?		6-82	Returns Analyzer Input setting.
LIVe		6-82	Selects Live Trace mode for the Spectrum Analyzer.
MARKer:			
AOFF		6-83	Disables both Markers.
DELTA:			
AMPLitude?		6-83	Returns amplitude difference between the Trace Marker 1 and Trace Marker 2 crossings in dB.
FREQuency?		6-83	Returns difference between the two Marker positions in MHz.
POINt?		6-83	Returns difference between Marker positions in graticules with 100 graticules equal to the Analyzer display width.
TRACK *b*	1 or 0	6-83	Enables/disables Marker Tracking Feature. Tracking feature keeps Markers a constant distance apart.
MARKER1:			
AMPLitude?		6-84	Returns amplitude of the Trace at the Marker 1 crossing. Range and units depend on the current scale settings (**ANLZ:SCALe** commands).
FREQuency?		6-84	Returns Marker 1 position in kHz (250 to 2010000.0).
POINt *n*	1 to 100	6-84	Sets Marker 1 position to *n* graticules (100 graticules equal to the Analyzer display width).
POINt?		6-84	Returns Marker 1 position in graticules.
STATe *b*	1 or 0	6-84	Enables/disables Marker 1.
STATe?		6-84	Returns current state of Marker 1.
MARKER2:			
AMPLitude?		6-84	Returns amplitude of the Trace at the Marker 2 crossing. Range and units depend on the current scale settings (**ANLZ:SCALe** commands).
FREQuency?		6-84	Returns Marker 2 position in kHz (250 to 2010000.0).
POINt *n*	1 to 100	6-84	Sets Marker 2 position to *n* graticules (100 graticules equal to the Analyzer display width).
POINt?		6-84	Returns Marker 2 position in graticules.
STATe *b*	1 or 0	6-84	Enables/disables Marker 2.
STATe?		6-84	Returns current state of Marker 2.
MODE *type*	DIRect (Direct Mode) or CHANnel (Channel Mode)	6-85	Selects Analyzer RF Mode.
NORMalize		6-85	Normalizes Analyzer Trace to match RF Generator Output.
PEAK		6-85	Selects Peak Hold Feature for Analyzer.

COMMAND	RANGE	PAGE	DESCRIPTION
ANLZ:			
PLOT:			
GRID		6-85	Draws Analyzer grid on attached plotter.
TRACE		6-85	Draws Analyzer trace on attached plotter.
UNITS		6-85	Draws Analyzer units on attached plotter.
QTR		6-85	Selects 1/4 size Analyzer display for RF Generator, Receiver and Duplex Operation Screens.
RCL *n*	1 to 9	6-85	Recalls Analyzer Trace and parameters stored in memory location *n*.
RLEVel *n*	0 to 64	6-85	Sets reference or offset level in dB for Analyzer only in 2 dB scale.
RLEVel?		6-85	Returns reference or offset value in dB for Analyzer in 2 dB scale.
SCALe *n*	2 or 10	6-85	Selects Analyzer Units/Division Factor in dB.
SCALe:			
UNIT:*type*	T/R Analyzer Input: DBM (dBm) or DBW (dBW) ANTENNA Analyzer Input: DBM (dBm), DBMV (dBmV), DBUV (dBμV), DBV (dBV), DBUW (dBμW)	6-86	Sets Analyzer Scale Units. If DBW is selected, the T/R Connector is selected for the Analyzer Input. If DBV, DBMV, DBUV or DBUW is selected, the ANTENNA IN Connector is selected for the Analyzer Input.
UNIT?		6-86	Returns Analyzer Scale Units.
SCALe?		6-86	Returns Analyzer Scale in dB.
SCAN *n*	0 (for zero scan), 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000, 20000, 50000, 100000	6-86	Selects Analyzer Scan Width in kHz.
SCAN?		6-86	Returns Analyzer Scan Width in kHz.
STATE *b*	1 or 0	6-86	Enables/disables Analyzer display in RF Generator, Receiver and Duplex Operation Screens.
STORe *n*	1 to 9	6-86	Stores current Analyzer Trace and environment (routings and settings) in memory location *n*.
TOP?		6-87	Returns top of screen value in current units. Spectrum Analyzer Operation Screen must be displayed.

COMMAND	RANGE	PAGE	DESCRIPTION
For the following **ANLZ:TRACE** commands, the Analyzer display is divided into 400 positions horizontally (0 signifying the left edge of the display, 399 signifying the right edge of the display) and 255 values vertically (0 signifying the bottom of the display, 255 signifying the top of the display).			
ANLZ:			
TRACE:			
DATA *n,offset,value,...,value*	*n* = 1 to 9 *offset* = 0 to 399 value = 0 to 255	6-87	Replaces points of stored Trace *n* with specified vertical values (*value*) starting with horizontal offset (*offset*). Intended for remote GPIB or RS-232 use only; command and data are sent via GPIB or RS-232 from an external application, etc.
DATA? [[[*n*],*offset*],*points*]	*n* = 0 to 9, 0 = Live Trace (0 default). *offset* = 0 to 399 (0 default) *points* = 1 to 400 (400 default)	6-87	Returns the vertical values for each of the *points* specified, starting with *offset* (horizontal offset) of trace *n*. Intended for remote GPIB or RS-232 use only; data is sent out GPIB or RS-232 for use by an external application, etc.
GET *name,n*	*n* = 0 to 9, 0 = Live Trace.	6-88	Assigns vertical values of trace *n* (in graticules) to each element of declared array *name*. First element of array corresponds with first point of trace *n*. If array is less than 400 values in length, the remaining portion of trace is left unassigned to an array. See **ANLZ:TRACE:PUT**.
GET? *n,offset*	*n* = 0 to 9, 0 = Live Trace *offset* = 0 to 399	6-88	Returns vertical value of a point in Trace *n* located at *offset* (horizontal position from the left edge of display).
MAX? [[[*n*],*offset*],*points*]	*n* = 0 to 9, 0 = Live Trace (0 default). *offset* = 0 to 399 (0 default). points = 1 to 400 (400 default)	6-88	Returns the maximum point of Trace *n* within specified number of *points* starting with given *offset*. Result is returned in x,y format with x being the number of positions from the left edge and y being the number of values from the bottom. Trace *n* can be the live Trace or a Trace stored in memory. Intended for remote GPIB or RS-232 use only; data is sent out GPIB or RS-232 for use by an external application, etc.
MIN? [[[*n*],*offset*],*points*]	*n* = 0 to 9, 0 = Live Trace (0 default). *offset* = 0 to 399 (0 default). points = 1 to 400 (400 default)	6-88	Returns the minimum point of Trace *n* within specified number of *points* starting with the given *offset*. Result is returned in x,y format with x being the number of positions from the left edge and y being the number of values from the bottom. Trace *n* can be the Live Trace or a Trace stored in memory. Intended for remote GPIB or RS-232 use only; data is sent via GPIB or RS-232 from an external application, etc.
PUT *name,n*	*n* = 1 to 9	6-89	Assigns values of an array to trace *n*. *name* is array name. *n* signifies stored trace. Each element value represent a vertical value for associated horizontal positions. If array is less than 400 values in length (stored trace length), the array values are assigned to the trace the length of the array starting from horizontal position 0. The remaining portion of the stored trace is left intact. See **ANLZ:TRACE:GET**.

COMMAND	RANGE	PAGE	DESCRIPTION
ANLZ:			
TRACK:			
BWIDth *f*	3, 30, 300 or 3000	6-89	Sets Tracking Generator Bandwidth in kHz.
BWIDth?		6-89	Returns Tracking Generator Bandwidth in kHz.
LEVel *n*	-137.0 to 0.0	6-89	Sets Tracking Generator Level in dBm.
LEVel?		6-89	Returns Tracking Generator Level in dBm.
OUTput:			
DUPlex		6-90	Selects DUPLEX OUT Connector as Tracking Generator Output Connector.
TR		6-90	Selects T/R Connector as Tracking Generator Output Connector.
OUTput?		6-90	Returns current Tracking Generator Output Connector.
RESolution:			
HIGH		6-90	Selects high for Tracking Generator Resolution.
LOW		6-90	Selects low for Tracking Generator Resolution.
MED		6-90	Selects medium for Tracking Generator Resolution.
RESolution?		6-90	Returns HIGH if Tracking Resolution is high, LOW if Tracking Resolution is low or MED if Tracking Resolution is medium.
STATe *b*	1 or 0	6-91	Enables/disables Tracking Generator.
STATe?		6-91	Returns 1 if Tracking Generator is active; 0 if inactive.

AUXILIARY TEST SET (SPECIAL TEST) COMMANDS

AHIT?		6-173	Returns 1 if there is input waiting from the Sp Tst; 0 otherwise.
AUX "*string*"		6-173	Issues commands, as strings, to the Sp Tst.
AUX? "*string?*"		6-173	Issues queries, as strings, to the Sp Tst.

BER (BIT ERROR RATE) METER COMMANDS

See M_BER:

CELLULAR AMPS/NAMPS COMMANDS

Queries for received data return -1 if data is not available or has already been read.

CELL:			
ACTion?		6-126	Returns current Action field value.
B_I?		6-125	Returns current state of Busy/Idle bit.
BAND?		6-130	Returns NADC or NAMPS band. Returns one of the following: U8, U4, HY, L, M or U.
BIS?		6-127	Returns current state of BIS bit.
BOTH		6-125	Selects decoding of both Stream A and B words.
C12?		6-129	Returns current C12 value.
C13?		6-129	Returns current C13 value.
CAPTure:			
MIN "*xxx/xxx-xxxx*"	0-9, # and *	6-131	Specifies MIN capture value.
MIN?		6-131	Returns current MIN Capture value in the format: "xxx/xxx-xxxx."
MODE:*xxx*	MIN, ORDER, BOTH (MIN and ORDER) or OFF (none)	6-131	Specifies Mode on which to Capture.
MODE?		6-131	Returns current capture mode: MIN, ORDER, BOTH or OFF.

COMMAND	RANGE	PAGE	DESCRIPTION
CELL:			
CAPTure:			
ORDer:*xxx*	PAGE, ALERT, RELease, REORDer, SALERT, AUDIT, SNDAddr, INTERCEPT, MAINTenance, POWer, DRETRY, AUTREG, AINTERCEPT, AREORDer, AALERT, VCDES	6-131	Specifies Order on which to capture.
ORDer?		6-131	Returns current Order on which to capture.
CHANnel *n*	1 to 2047, depending on format	6-125	Specifies Channel.
CHANnel?		6-125	Returns current Channel.
CHANPOS1?		6-127	Returns current value of first channel position field from directed retry message.
CHANPOS2?		6-127	Returns current value of second channel position field from directed retry message.
CHANPOS3?		6-127	Returns current value of third channel position field from directed retry message.
CHANPOS4?		6-127	Returns current value of fourth channel position field from directed retry message.
CHANPOS5?		6-127	Returns current value of fifth channel position field from directed retry message.
CHANPOS6?		6-127	Returns current value of sixth channel position field from directed retry message.
CMAC?		6-126	Returns current CMAC value.
CMAX_1?		6-126	Returns current CMAX-1 value.
CPA?		6-127	Returns current state of CPA bit.
DCC?		6-125	Returns current DCC value.
DIGITs?		6-128	Returns current Call address value.
DSCC?		6-128	Returns current DSCC value.
DTX?		6-126	Returns current state of DTX bit.
E?		6-126	Returns current state of E bit.
EF?		6-128	Returns current EF value.
END?		6-126	Returns current state of END bit.
EP?		6-128	Returns current EP value.
ESN?		6-128	Returns current ESN value.
FORMat:			
AMPS		6-129	Sets channel format to AMPS (800 MHz).
NADC:			
BAND:			
HY		6-130	Sets NADC band to HY (Hyperband - 1900 MHz).
U4		6-130	Sets NADC band to U4 (NT400© - 450 MHz).
U8		6-130	Sets NADC band to U8 (AMPS - 800 MHz).
NAMPS:			
BAND:			
Lower		6-130	Sets NAMPS band to Lower.
Middle		6-130	Sets NAMPS band to Middle.
Upper		6-130	Sets NAMPS band to Upper.
NT400		6-129	Sets channel format to NT400 (450 MHz).
PCS		6-129	Sets channel format to PCS (1900 MHz).
FORMat?		6-130	Returns cellular format. Returns one of the following: AMPS, NT400 or PCS.

COMMAND	RANGE	PAGE	DESCRIPTION
CELL:			
GEN:			
FOCC:			
CHANnel *n*	1 to 2047, depending on format	6-132	Specifies Channel.
CHANnel?		6-132	Returns current Channel.
CMAC *n*	0 to 7	6-132	Specifies Control Mobile Attenuation Code in Control-Filler Message.
CMAC?		6-132	Returns current value of Control Mobile Attenuation Code in Control-Filler Message.
CMAX *n*	1 to 128	6-132	Specifies Maximum number of channels in System Parameter Overhead Message.
CMAX?		6-132	Returns current value of Maximum number of channels in System Parameter Overhead Message.
CPA *b*	1 or 0	6-133	Enables/disables Combined Paging/Access bit in System Parameter Overhead Message.
CPA?		6-133	Returns current state of Combined Paging/Access bit in System Parameter Overhead Message.
DCC *n*	0 to 3	6-133	Specifies Digital Color Code in System Parameter Overhead Message.
DCC?		6-133	Returns current value of Digital Color Code in System Parameter Overhead Message.
DTX *n*	0 to 3	6-133	Specifies Discontinuous Transmission in System Parameter Overhead Message.
DTX?		6-133	Returns current value of Discontinuous Transmission in System Parameter Overhead Message.
E *b*	1 or 0	6-133	Enables/disables Extended Address bit in System Parameter Overhead Message.
E?		6-133	Returns current state of Extended Address bit in System Parameter Overhead Message.
EP *b*	1 or 0	6-133	Enables/disables Extended Protocol bit in System Parameter Overhead Message.
EP?		6-133	Returns current state of Extended Protocol bit in System Parameter Overhead Message.
N *n*	1 to 32	6-134	Specifies Number of paging channels in System Parameter Overhead Message.
N?		6-134	Returns current value of Number of paging channels in System Parameter Overhead Message.
RCF *b*	1 or 0	6-134	Enables/disables Read Control Filler bit in System Parameter Overhead Message.
RCF?		6-134	Returns current state of Read Control Filler bit in System Parameter Overhead Message.
REGH *b*	1 or 0	6-134	Enables/disables Registration for Home Stations bit in System Parameter Overhead Message.
REGH?		6-134	Returns current state of Registration for Home Stations bit in System Parameter Overhead Message.
REGR *b*	1 or 0	6-134	Enables/disables Registration for Roaming Mobile Phones bit in System Parameter Overhead Message.
REGR?		6-134	Returns current state of Registration for Roaming Mobile Phones bit in System Parameter Overhead Message.
S *b*	1 or 0	6-134	Enables/disables Serial Number bit in System Parameter Overhead Message.
S?		6-134	Returns current state of Serial Number bit in System Parameter Overhead Message.
SEND		6-132	Begin transmitting System Parameter Overhead Message
SETUP		6-132	Configures Test Set for FOCC Screen without going to screen.

COMMAND	RANGE	PAGE	DESCRIPTION
CELL:			
GEN:			
FOCC:			
SID *n*	0 to 32767	6-135	Specifies System Identification number in System Parameter Overhead Message.
SID?		6-135	Returns current value of System Identification number in System Parameter Overhead Message.
STOP		6-132	Stops transmitting System Parameter Overhead Message
WFOM *b*	1 or 0	6-135	Enables/disables state of Wait for Overhead Message bit in Control-Filler Message.
WFOM?		6-135	Returns current state of Wait for Overhead Message bit in Control-Filler Message.
FVC:			
C12 *b*	1 or 0	6-149	Enables/disables C12 bit in Mobile Station Control Message.
C12?		6-149	Returns current state of C12 bit in Mobile Station Control Message.
C13 *b*	1 or 0	6-149	Enables/disables Mobile Station Control Message.
C13?		6-149	Returns current state of C13 bit in Mobile Station Control Message.
CHAN *n*	1 to 1024	6-149	Specifies voice channel to which call is assigned in Mobile Station Control Message.
CHAN?		6-149	Returns current voice channel to which call is assigned in Mobile Station Control Message.
CHANNEL *n*	1 to 2047, depending on format	6-149	Specifies Channel.
CHANNEL?		6-149	Returns current Channel.
CLI "*string*"	0-9, #, * and N (Null). 32 characters, max.	6-149	Specifies Call Line Identifier string in Mobile Station Control Message.
CLI?		6-149	Returns current Call Line Identifier string in Mobile Station Control Message.
DSCC *n*	0 to 7	6-150	Specifies DSAT Color Code in Mobile Station Control Message.
DSCC?		6-150	Returns current value of DSAT Color Code in Mobile Station Control Message.
LOCAL *n*	0 to 31	6-150	Specifies LOCAL in Mobile Station Control Message.
LOCAL?		6-150	Returns current value of LOCAL in Mobile Station Control Message.
MSL *n*	0 to 31	6-150	Specifies Message Length in Mobile Station Control Message.
MSL?		6-150	Returns current value of Message Length in Mobile Station Control Message.
MST *n*	0 to 225	6-150	Specifies Message Type in Mobile Station Control Message.
MST?		6-150	Returns current value of Message Type in Mobile Station Control Message.
NAMPS:			
BER *n*	0 to 127	6-153	Specifies number of allowable bit errors in Mobile Station Control Message.
BER?		6-153	Returns current number of allowable bit errors in Mobile Station Control Message.
C12 *b*	1 or 0	6-154	Enables/disables C12 bit in Mobile Station Control Message.
C12?		6-154	Returns current state of C12 bit in Mobile Station Control Message.
C13 *b*	1 or 0	6-154	Enables/disables C13 bit in Mobile Station Control Message.
C13?		6-154	Returns current state of C13 bit in Mobile Station Control Message.
CHAN *n*	1 to 1024	6-153	Specifies voice channel to which call is assigned in Mobile Station Control Message (11 least significant bits).
CHAN?		6-153	Returns current voice channel to which call is assigned in Mobile Station Control Message (11 least significant bits).

COMMAND	RANGE	PAGE	DESCRIPTION
CELL:			
GEN:			
FVC:			
NAMPS:			
CHANNEL *n*	0 to 1023	6-153	Specifies Channel.
CHANNEL:*x*	LOWer, MIDdle, UPper	6-153	Specifies Band.
CHANNEL?		6-153	Returns current Channel.
CLI "*string*"	0-9, #, * and N (Null). 32 characters, max.	6-154	Specifies Call Line Identifier string in Mobile Station Control Message.
CLI?		6-154	Returns current Call Line Identifier string in Mobile Station Control Message.
CTYP *b*	1 or 0	6-154	Enables/disables Channel Type indicator in Mobile Station Control Message.
CTYP?		6-154	Returns current state of Channel Type indicator in Mobile Station Control Message.
DSAT *n*	0 to 6	6-155	Specifies Digital Supervisory Audio Tone in Mobile Station Control Message.
DSAT?		6-155	Returns current value of Digital Supervisory Audio Tone in Mobile Station Control Message.
DSCC *n*	0 to 7	6-154	Specifies DSAT Color Code in Mobile Station Control Message.
DSCC?		6-154	Returns current value of DSAT Color Code in Mobile Station Control Message.
LOCAL *n*	0 to 31	6-155	Specifies LOCAL in Mobile Station Control Message.
LOCAL?		6-155	Returns current value of LOCAL in Mobile Station Control Message.
MSL *n*	0 to 31	6-155	Specifies Message Length in Mobile Station Control Message.
MSL?		6-155	Returns current value of Message Length in Mobile Station Control Message.
MST *n*	0 to 255	6-155	Specifies Message Type in Mobile Station Control Message.
MST?		6-155	Returns current value of Message Type in Mobile Station Control Message.
NEXT		6-153	Sends next word of Mobile Station Control Message, if required.
O_E *b*	1 or 0	6-155	Enables/disables Odd/Even bit in Mobile Station Control Message.
O_E?		6-155	Returns current state of Odd/Even bit in Mobile Station Control Message.
ORDER:*xxx*	PAGE, SNDADDR_EVEN, SNDADDR_ODD, S_ALERT, AUDIT, MAINTenance, ALERT, RELease, FADE, SUSP_CALLED_ ADDR, HANDOFF_ CONFIRM, PWRLvl, HANDoff, MRI, EXTENDed	6-156	Specifies Order in Mobile Station Control Message.

COMMAND	RANGE	PAGE	DESCRIPTION
CELL:			
GEN:			
FVC:			
NAMPS:			
PDSCC *n*	0 to 7	6-156	Specifies Present DSAT Color Code in Mobile Station Control Message.
PDSCC?		6-156	Returns current value of Present DSAT Color Code in Mobile Station Control Message.
PWRLVL *n*	0 to 7	6-156	Specifies Power Level.
PWRLVL?		6-156	Returns Power Level.
RSSI *n*	0 to 7	6-156	Specifies Received Signal Strength in Mobile Station Control Message.
RSSI?		6-156	Return current value of Received Signal Strength in Mobile Station Control Message.
SEND		6-153	Sends the Mobile Station Control Message.
SETUP		6-153	Configures Test Set for NAMPS Forward Voice Channel Screen without going to screen.
SHORT_MESSage "*string*"	See IS-88, Appendix A for string details.	6-157	Specifies Short Message string in Mobile Station Control Message.
SHORT_MESSage?		6-157	Returns current Short Message string in Mobile Station Control Message.
VMAC *n*	0 to 7	6-157	Specifies Voice Mobile Attenuation Code in Mobile Station Control Message.
VMAC?		6-157	Returns current value of Voice Mobile Attenuation Code in Mobile Station Control Message.
VOICE_MESSage "*string*"	See IS-88, Appendix A for string details.	6-157	Specifies Voice Mail message string in Mobile Station Control Message.
VOICE_MESSage?		6-157	Returns current Voice Mail message string in Mobile Station Control Message.
VOICE_UNANSWERED "*nn*"	00 to 99	6-157	Specifies number of unanswered messages in Mobile Station Control Message.
VOICE_UNANSWERED?		6-157	Returns current number of unanswered messages in Mobile Station Control Message.
VOICE_URGENT:*x*	ON or OFF	6-157	Turns on or off Urgent message identifier in Mobile Station Control Message.
VOICE_URGENT?		6-157	Returns current state of Urgent message identifier in Mobile Station Control Message.
ORDER:*xxx*	PAGE, SNDADDR, S_ALERT, AUDIT, MAINTenance, ALERT, RELease, PWRLvl, HANDoff, NAMPS_CH_ASGN, EXTENDed	6-150	Specifies ORDER in Mobile Station Control Message.
PSCC *n*	0 to 3	6-151	Specifies Present SAT Color Code in Mobile Station Control Message.
PSCC?		6-151	Returns current value of Present SAT Color Code in Mobile Station Control Message.
PWRLVL *n*	0 to 7	6-151	Specifies Power Level.
PWRLVL?		6-151	Returns Power Level.
REPEAT:*x*	ON or OFF	6-148	Turns Repeat on or off.
REPEAT?		6-148	Returns current state of Repeat.
SCC *n*	0 to 3	6-151	Specifies SAT Color Code in Mobile Station Control Message.
SCC?		6-151	Returns current value of SAT Color Code in Mobile Station Control Message.
SEND		6-148	Sends Mobile Station Control Message.
SETUP		6-148	Configures Test Set for AMPS Forward Voice Channel Screen without going to screen.

COMMAND	RANGE	PAGE	DESCRIPTION
CELL:			
GEN:			
FVC:			
SHORT_MESSage "*string*"	See IS-88, Appendix A for string details.	6-151	Specifies Short Message string in Mobile Station Control Message.
SHORT_MESSage?		6-151	Returns current Short Message string in Mobile Station Control Message.
STOP		6-148	Stops transmission of Mobile Station Control Message.
VMAC *n*	0 to 7	6-151	Specifies Voice Mobile Attenuation Code in Mobile Station Control Message.
VMAC?		6-151	Returns current value of Voice Mobile Attenuation Code in Mobile Station Control Message.
VOICE_MESSage "*string*"	See IS-88, Appendix A for string details.	6-152	Specifies Voice Mail message string in Mobile Station Control Message.
VOICE_MESSage?		6-152	Returns current Voice Mail message string in Mobile Station Control Message.
VOICE_UNANSWERED "*nn*"	00 to 99	6-152	Specifies number of unanswered messages in Mobile Station Control Message.
VOICE_UNANSWERED?		6-152	Returns current number of unanswered messages in Mobile Station Control Message.
VOICE_URGENT:*x*	ON or OFF	6-152	Turns on or off Urgent message identifier in Mobile Station Control Message.
VOICE_URGENT?		6-152	Returns current state of Urgent message identifier in Mobile Station Control Message.
GLACT:			
ACTion:			
ACCess *b*	1 or 0	6-137	Enables/disables Access Attempt in Global Action Overhead Message menu.
ACCess?		6-137	Returns current state of Access Attempt in Global Action Overhead Message menu.
BIS *b*	1 or 0	6-137	Enables/disables Access Type in Global Action Overhead Message menu.
BIS?		6-137	Returns current state of Access Type in Global Action Overhead Message menu.
LOCAL1 *b*	1 or 0	6-137	Enables/disables Local Control 1 in Global Action Overhead Message menu.
LOCAL1?		6-137	Returns current state of Local Control 1 in Global Action Overhead Message menu.
LOCAL2 *b*		6-137	Enables/disables Local Control 2 in Global Action Overhead Message menu.
LOCAL2?		6-137	Returns current state of Local Control 2 in Global Action Overhead Message menu.
NEWACC *b*	1 or 0	6-136	Enables/disables New Access channel set in Global Action Overhead Message menu.
NEWACC?		6-136	Returns current state of New Access channel set in Global Action Overhead Message menu.
OLC *b*	1 or 0	6-136	Enables/disables Overload Control in Global Action Overhead Message menu.
OLC?		6-136	Returns current state of Overload Control in Global Action Overhead Message menu.
REGINCR *b*	1 or 0	6-136	Enables/disables Registration Increment in Global Action Overhead Message menu.
REGINCR?		6-136	Returns current state of Registration Increment in Global Action Overhead Message menu.
RESCAN *b*	1 or 0	6-136	Enables/disables Rescan in Global Action Overhead Message menu.
RESCAN?		6-136	Returns current state of Rescan in Global Action Overhead Message menu.

COMMAND	RANGE	PAGE	DESCRIPTION
CELL:			
GEN:			
GLACT:			
BIS *b*		6-138	Enables/disables Busy-Idle Status bit in the Global Action Overhead Message.
BIS?		6-138	Returns current state of Busy-Idle Status bit in Global Action Overhead Message.
CHANNEL *n*	1 to 2047, depending on format	6-136	Specifies Channel.
CHANNEL?		6-136	Returns current Channel.
LOCALcntl *n*	0 to 65535	6-138	Specifies Local Control in Global Action Overhead Message.
LOCALcntl?		6-138	Returns current value of Local Control in Global Action Overhead Message.
MAXBusy:			
OTHer *n*	0 to 15	6-138	Specifies Maximum number of Busy occurrences allowed for Other accesses in Global Action Overhead Message.
OTHer?		6-138	Returns current value of Maximum number of Busy occurrences allowed for Other accesses in Global Action Overhead Message.
PGR *n*	0 to 15	6-138	Specifies Maximum number of Busy occurrences allowed for Page Responses in Global Action Overhead Message.
PGR?		6-138	Returns current value of Maximum number of Busy occurrences allowed for Page Responses in Global Action Overhead Message.
MAXSztr:			
OTHer *n*	0 to 15	6-139	Specifies Maximum number of Seizure attempts allowed for Other accesses in Global Action Overhead Message.
OTHer?		6-139	Returns current value of Maximum number of Seizure attempts allowed for Other accesses in Global Action Overhead Message.
PGR *n*	0 to 15	6-139	Specifies Maximum number of Seizure attempts allowed for Page Responses in Global Action Overhead Message.
PGR?		6-139	Returns current value of Maximum number of Seizure attempts allowed for Page Responses in Global Action Overhead Message.
NEWACC *n*	0 to 2047	6-139	Specifies New Access Channel starting point in Global Action Overhead Message.
NEWACC?		6-139	Returns current value of New Access Channel starting point in Global Action Overhead Message.
OLC *n*	0 to 32767	6-139	Specifies Overload Control Class in Global Action Overhead Message.
OLC?		6-139	Returns current value of Overload Control Class in Global Action Overhead Message.
REGINCR *n*	0 to 4095	6-140	Specifies Registration Increment in Global Action Overhead Message.
REGINCR?		6-140	Returns current value of Registration Increment in Global Action Overhead Message.
REPEAT:*x*	ON or OFF	6-135	Turns Repeat on or off for Global Action Overhead Message.
REPEAT?		6-135	Returns current state of Repeat for Global Action Overhead Message.
SEND		6-135	Appends Global Action Overhead Message to System Parameter Overhead Message.
SETUP		6-135	Configures Test Set for Global Action Overhead Message Screen without going to screen.
STOP		6-135	Stops Global Action Overhead Message from being sent with System Parameter Overhead Message.

COMMAND	RANGE	PAGE	DESCRIPTION
CELL:			
GEN:			
MSCM:			
C12 *b*	1 or 0	6-141	Enables/disables C12 bit in Mobile Station Control Message.
C12?		6-141	Returns current state of C12 bit in Mobile Station Control Message.
C13 *b*	1 or 0	6-141	Enables/disables C13 bit in Mobile Station Control Message.
C13?		6-141	Returns current state of C13 bit in Mobile Station Control Message.
CHAN *n*	0 to 1024	6-141	Specifies voice channel to which call is assigned in Mobile Station Control Message.
CHAN?		6-141	Returns current voice channel assignment for call in Mobile Station Control Message.
CHANNEL *n*	1 to 2047, depending on format	6-140	Specifies Channel.
CHANNEL?		6-140	Returns current Channel.
CHANPos *x,y*	1 to 6, 0 to 127	6-141	Sets position of a control channel relative to first access channel in Mobile Station Control Message.
CHANPos? *n*	1 to 6	6-141	Returns current position of a control channel relative to first access channel in Mobile Station Control Message.
CLI "*string*"	0-9, #, * and N (Null). 32 characters, max	6-141	Specifies Call Line Identifier string in Mobile Station Control Message.
CLI?		6-141	Returns current Call Line Identifier string in Mobile Station Control Message.
DSCC *n*	0 to 7	6-142	Specifies DSAT Color Code in Mobile Station Control Message.
DSCC?		6-142	Returns current value of DSAT Color Code in Mobile Station Control Message.
EF *b*	1 or 0	6-142	Enables/disables Extended Protocol Forward Channel Indicator bit in Mobile Station Control Message.
EF?		6-142	Returns current state of Extended Protocol Forward Channel Indicator bit in Mobile Station Control Message.
LOCAL *n*	0 to 31	6-142	Specifies LOCAL in Mobile Station Control Message.
LOCAL?		6-142	Returns current value of LOCAL in Mobile Station Control Message.
MIN "*xxx/xxx-xxxx*"	0-9, # and *	6-142	Specifies Mobile Identification Number in Mobile Station Control Message.
MIN?		6-142	Returns current Mobile Identification Number in Mobile Station Control Message.
MSL *n*	0 to 31	6-142	Specifies Message Length in Mobile Station Control Message.
MSL?		6-142	Returns current value of Message Length in Mobile Station Control Message.
MST *n*	0 to 255	6-143	Specifies Message Type in Mobile Station Control Message.
MST?		6-143	Returns current value of Message Type in Mobile Station Control Message.
ORDER:*xxx*	AUDIT, LC, DIR_RTRY, INTRCPT, RELease, REORDER, VC_DES, EXTENDed	6-143	Specifies Order in Mobile Station Control Message.
ORDQ *n*	0 to 7	6-143	Specifies Order Qualifier in Mobile Station Control Message.
ORDQ?		6-143	Returns current value of Order Qualifier in Mobile Station Control Message.

COMMAND	RANGE	PAGE	DESCRIPTION
CELL:			
GEN:			
MSCM:			
REPEAT:*x*	ON or OFF	6-140	Sets Repeat on or off in Mobile Station Control Message.
REPEAT?		6-140	Returns current state of Repeat in Mobile Station Control Message.
SCC *n*	0 to 3	6-143	Specifies Supervisory Audio Tone Color Code in Mobile Station Control Message.
SCC?		6-143	Returns current value of Supervisory Audio Tone Color Code in Mobile Station Control Message.
SEND		6-140	Sends Mobile Station Control Message.
SETUP		6-140	Configures Test Set for Mobile Station Control Message Screen without going to screen.
SHORT_MESSage "*string*"	See IS-88, Appendix A for string details.	6-143	Specifies Short Message string in Mobile Station Control Message.
SHORT_MESSage?		6-143	Returns current Short Message string in Mobile Station Control Message.
STOP		6-140	Stops transmission of Mobile Station Control Message.
VMAC *n*	0 to 7	6-144	Specifies Voice Mobile Attenuation Code in Mobile Station Control Message.
VMAC?		6-144	Returns current value of Voice Mobile Attenuation Code in Mobile Station Control Message.
VOICE_MESSage "*string*"	See IS-88, Appendix A for string details.	6-144	Specifies Voice Mail message string in Mobile Station Control Message.
VOICE_MESSage?		6-144	Returns current Voice Mail message string in Mobile Station Control Message.
VOICE_UNANSWERED "*nn*"	00 to 99	6-144	Specifies number of unanswered messages in Mobile Station Control Message.
VOICE_UNANSWERED?		6-144	Returns current number of unanswered messages in Mobile Station Control Message.
VOICE_URGENT:*x*	ON or OFF	6-144	Turns on or off Urgent message identifier in Mobile Station Control Message.
VOICE_URGENT?		6-144	Returns current state of Urgent message identifier in Mobile Station Control Message.
RECC:			
CALLED_ADDRess "*string*"	0-9, #, and *. 32 characters, max.	6-145	Specifies Called Address.
CALLED_ADDRess?		6-145	Returns current Called Address.
CHANNEL *n*	0 to 1023	6-145	Specifies Channel.
CHANNEL?		6-145	Returns current Channel.
DCC *n*	0 to 3	6-146	Specifies Digital Color Code.
DCC?		6-146	Returns current value of Digital Color Code.
E *b*	1 or 0	6-145	Enables/disables Extended Address bit.
E?		6-145	Returns current state of Extended Address bit.
EP *b*	1 or 0	6-146	Enables/disables Extended Protocol bit.
EP?		6-146	Returns current state of Extended Protocol bit.
ER *b*	1 or 0	6-146	Enables/disables Extended Protocol Reverse Channel bit.
ER?		6-146	Returns current state of Extended Protocol Reverse Channel bit.
ESN "*string*"	0 to 9. 11 digits, max.	6-146	Specifies Electronic Serial Number.
ESN?		6-146	Returns current Electronic Serial Number.
LOCAL *n*	0 to 31	6-146	Specifies LOCAL.
LOCAL?		6-146	Returns current value of LOCAL.
LT *b*	1 or 0	6-146	Enables/disables Last Try bit.
LT?		6-146	Returns current state of Last Try bit.
MIN "*xxx/xxx-xxxx*"	0-9, # and *	6-147	Specifies Mobile Identification Number.
MIN?		6-147	Returns current Mobile Identification Number.

COMMAND	RANGE	PAGE	DESCRIPTION
CELL:			
GEN:			
RECC:			
MSL *n*	0 to 31	6-147	Specifies Message Length.
MSL?		6-147	Returns current value of Message Length.
MST *n*	0 to 255	6-147	Specifies Extended Protocol Message Type.
MST?		6-147	Returns current value of Extended Protocol Message Type.
ORDER *n*	0 to 31	6-147	Specifies Order.
ORDER?		6-147	Returns current value of Order.
ORDQ *n*	0 to 7	6-147	Specifies Order Qualifier.
ORDQ?		6-147	Returns current value of Order Qualifier.
REPEAT:*x*	ON or OFF	6-145	Turns Repeat on or off.
REPEAT?		6-145	Returns current state of Repeat.
S *b*	1 or 0	6-147	Enables/disables Serial Number bit.
S?		6-147	Returns current state of Serial Number bit.
SCM *n*	0 to 15	6-148	Specifies Station Class Mark.
SCM?		6-148	Returns current value of Station Class Mark.
SEND		6-145	Sends Reverse Control Channel message.
SETUP		6-145	Configures Test Set for Reverse Control Channel Screen without going to screen.
STOP		6-145	Stops transmission of Reverse Control Channel message.
T *b*	1 or 0	6-148	Enables/disables Type of message bit.
T?		6-148	Returns current state of Type of message bit.
RVC:			
CALLED_ADDRess "*string*"	0-9, # and *. 32 characters, max.	6-158	Specifies Called Address string for Called-Address Message.
CALLED_ADDRess?		6-158	Returns current Called Address string for Called-Address Message.
CHANNEL *n*	0 to 1023	6-158	Specifies Channel.
CHANNEL?		6-158	Returns current Channel.
LOCAL *n*	0 to 31	6-158	Specifies LOCAL in Order Confirmation Message.
LOCAL?		6-158	Returns current value of LOCAL in Order Confirmation Message.
MESSAGE:*xxx*	ORDER_ CONFIRMation, CALLED_ADDRess, EXTENDed	6-159	Specifies Message to be transmitted.
MSL *n*	0 to 31	6-159	Specifies Message Length in Extended Protocol Message.
MSL?		6-159	Returns current value of Message Length in Extended Protocol Message.
MST *n*	0 to 255	6-159	Specifies Message Type in Extended Protocol Message.
MST?		6-159	Returns current value of Message Type in Extended Protocol Message.
NAMPS:			
BER *n*	0 to 127	6-160	Specifies number of Bit Errors in MRI (Mobile Reported Interference) Order Message.
BER?		6-160	Returns current number of Bit Errors in MRI (Mobile Reported Interference) Order Message.
CALLED_ADDRess "*string*"	0-9, # and *. 32 characters, max.	6-160	Specifies Called Address for Flash/Called-Address Message string.
CALLED_ADDRess?		6-160	Returns current Called Address for Flash/Called-Address Message string.
CHANNEL *n*	1 to 2047, depending on format	6-160	Specifies Channel.
CHANNEL:*x*	LOWer, MIDdle, UPper	6-160	Specifies Band.
CHANNEL?		6-160	Returns current Channel.

COMMAND	RANGE	PAGE	DESCRIPTION
CELL:			
GEN:			
RVC:			
NAMPS:			
CONFIRMation *b*	1 = Order Confirmation Message 0 = Order	6-161	Enables/disables T bit in Reverse Voice Channel Message to specify that message is a Order Confirmation Message or an Order.
CONFIRMation?		6-161	Returns current state of T bit in Reverse Voice Channel Message to specify that message is a Order Confirmation Message (1) or an Order (0).
DSAT *n*	0 to 6	6-161	Specifies Digital Supervisory Audio Tone.
DSAT?		6-161	Returns current value of Digital Supervisory Audio Tone.
LOCAL *n*	0 to 31	6-161	Specifies LOCAL for Order Message or Order Confirmation Message.
LOCAL?		6-161	Returns current value of LOCAL for Order Message or Order Confirmation Message.
MESSAGE:*xxx*	MRI, ORDER, FLASH	6-161	Specifies message to transmitted.
NEXT		6-160	Sends next word of Mobile Station Control Message, if required.
O_E *b*	1 or 0	6-161	Enables/disables Odd/Even bit.
O_E?		6-161	Returns current state of Odd/Even bit.
ORDER *n*	0 to 31	6-162	Specifies ORDER for order messages.
ORDER?		6-162	Returns current value of ORDER for order messages.
ORDQ *n*	0 to 7	6-161	Specifies Order Qualifier for order messages.
ORDQ?		6-161	Returns current value of Order Qualifier for order messages.
RSSI *n*	0 to 7	6-162	Specifies Received Signal Strength in MRI (Mobile Reported Interference) Order Message.
RSSI?		6-162	Returns current value of Received Signal Strength in MRI (Mobile Reported Interference) Order Message.
SEND		6-160	Sends message.
SETUP		6-160	Configures Test Set for NAMPS Reverse Voice Channel Screen without going to screen.
VMAC *n*	0 to 7	6-162	Specifies Voice Mobile Attenuation Code in Order Messages.
VMAC?		6-162	Returns current value of Voice Mobile Attenuation Code in Order Messages.
ORDER *n*	0 to 31	6-159	Specifies ORDER in Order Confirmation Message.
ORDER?		6-159	Returns current value of ORDER in Order Confirmation Message.
ORDQ *n*	0 to 7	6-159	Specifies Order Qualifier in Order Confirmation Message.
ORDQ?		6-159	Returns current value of Order Qualifier in Order Confirmation Message.
REPEAT:*x*		6-158	Turns Repeat on or off.
REPEAT?		6-158	Returns current state of Repeat.
SEND		6-158	Sends RVC message.
SETUP		6-158	Configures Test Set for AMPS Reverse Voice Channel Screen without going to screen.
STOP		6-158	Stops transmission of RVC message.
LOCALCTRL1?		6-129	Returns current first position field for Local Control value.
LOCALCTRL2?		6-129	Returns current second position field for Local Control value.
MAXbusy:			
OTHer?		6-128	Returns current value of Maximum Busy for Other accesses.
PGR?		6-128	Returns current value of Maximum Busy for page response.
MAXSztr:			
OTHer?		6-128	Returns current value of Maximum Seizure Tries for Other accesses.
PGR?		6-128	Returns current value of Maximum Seizure Tries for page response.

COMMAND	RANGE	PAGE	DESCRIPTION
CELL:			
MIN?		6-125	Returns current MIN value. Returning format is: "xxx/xxx-xxxx."
MSL?		6-129	Returns current MSL value.
MST?		6-129	Returns current MST value.
N_1?		6-126	Returns current N-1 value.
NAWC?		6-126	Returns current NAWC value.
NEWACC?		6-127	Returns current NEWACC value.
OLC?		6-127	Returns current OLC value.
ORDer?		6-125	Returns current ORDER value.
PDSCC?		6-128	Returns current PDSCC value.
PSCC?		6-128	Returns current PSCC value.
RCF?		6-127	Returns current state of RCF bit value.
REGH?		6-126	Returns current state of REGH bit.
REGINCR?		6-127	Returns current REGINCR value.
REGR?		6-126	Returns current state of REGR bit.
S?		6-126	Returns current state of S bit.
SCC?		6-125	Returns current SCC value.
SID?		6-125	Returns current SID value.
VCHAN?		6-126	Returns current Voice Channel number.
VMAC?		6-126	Returns current VMAC value.
WFOM?		6-126	Returns current state of WFOM bit.
WORD?		6-125	Returns current decoding selection: WORDA, WORDB or BOTH.
WORDA		6-125	Selects decoding of Stream A words.
WORDB		6-125	Selects decoding of Stream B words.

DEVIATION METER (PEAK) COMMANDS

See M_DEV:

DEVIATION METER (RMS) COMMANDS

See M_DRMS:

DIGITAL MULTIMETER COMMANDS

See M_DMM:

DISTORTION METER COMMANDS

See M_DIST:

DUPLEX COMMANDS

COMMAND	RANGE	PAGE	DESCRIPTION
DUPlex:			
INPut:			
AGC:			
AUTO		6-41	Sets Automatic Gain Control to automatic.
MANual *n*	0 to 255	6-41	Sets Automatic Gain Control to manual and sets level to *n*.
USER:*xxx*	MEASure, SPeech, DATA, HIGH, TYPE1, TYPE2 or TYPE3	6-41	Sets Automatic Gain Control to User setting.
ANTenna		6-41	Selects ANTENNA IN Connector as Transmitter Input Source. Displays Signal Strength Meter reading on the Duplex Transmitter Operation Screen.
ATTenuation *n*	0, 5, 10, 15, 20, 25, 30	6-41	Sets Duplex Input Attenuation in dB.
ATTenuation:			
LNA		6-41	Sets Duplex Input Attenuation to LNA.
LNA?		6-41	Returns current state of Low Noise Amplifier. Returns 1 if LNA is selected; 0 otherwise.
ATTenuation?		6-41	Returns current value of Input Attenuation.

COMMAND	RANGE	PAGE	DESCRIPTION
DUPlex:			
INPut:			
CHANnel *n*	1 to 2047, depending on format	6-42	Sets Duplex Transmitter Frequency cellular channel *n*.
CHANnel:			
BAND?		6-43	Returns band for Duplex Transmitter Channel Format. Returns one of the following bands for the associated channel format: (NADC) U8, U4 or HY; (ETACS) NOT AVAILABLE; (NAMPS) LOWER, MIDDLE or UPPER.
FORMat:			
AMPS:			
FORward		6-42	Selects AMPS (NADC-U8) Forward channels.
REVerse		6-42	Selects AMPS (NADC-U8) Reverse channels.
ETACS:			
FORward		6-42	Selects ETACS Forward channels.
REVerse		6-42	Selects ETACS Reverse channels.
NADC:			
BAND:*xx*	U8, U4 or HYper	6-43	Selects NADC band.
FORward		6-43	Selects NADC Forward. Utilizes current setting of NADC band.
REVerse		6-43	Selects NADC Reverse. Utilizes current setting of NADC band.
NAMPS:			
BAND:*x*	Lower, Middle or Upper	6-42	Selects NAMPS Narrow Analog channel designator.
FORward		6-42	Selects NAMPS Forward channels.
REVerse		6-42	Selects NAMPS Reverse channels.
NT400:			
FORward		6-43	Selects NT400 Forward channels.
REVerse		6-43	Selects NT400 Reverse channels.
FORMat?		6-43	Returns Duplex Transmitter Channel Format (NADC:FORWARD, NADC:REVERSE, ETACS:FORWARD, ETACS:REVERSE, NAMPS:FORWARD or NAMPS:REVERSE).
FIND:			
FREQuency?		6-44	Returns frequency of first signal with amplitude larger than Find reference level.
REFerence *n*	-110 to -5	6-44	Sets Find reference level in dB.
REFerence?		6-44	Returns Find reference level in dB.
FREQuency *n* [*units*]	250000.0 to 2010000000.0 [Hz] 250.0 to 2010000.0 [kHz] 0.2500 to 2010.0 [MHz] *units* - HZ, KHZ or MHz (default units are KHZ)	6-44	Sets Duplex Transmitter Frequency.
FREQuency?		6-44	Returns RF Generator Frequency in kHz.
METER:			
DEVRms		6-44	Displays Deviation Meter (RMS) when followed by a **SCREEN:DUPlex** command.
DISTortion		6-44	Displays Distortion Meter when followed by a **SCREEN:DUPlex** command.
MODMeter		6-44	Displays Modulation Meter when followed by a **SCREEN:DUPlex** command.
PMRms		6-44	Displays Phase Meter (RMS) when followed by a **SCREEN:DUPlex** command.
SINAD		6-45	Displays SINAD Meter when followed by a **SCREEN:DUPlex** command.

COMMAND	RANGE	PAGE	DESCRIPTION
DUPlex:			
INPut:			
MODE *type*	DIRect, CHANnel, SCAN, LIST, FLScan	6-45	Selects Duplex Transmitter Mode.
MODE?		6-45	Returns current mode (DIRECT, CHANNEL, FREQUENCY SCAN, FREQUENCY LIST or FREQUENCY LIST SCAN).
MODulation:			
AM*n*	1 or 2	6-45	Selects Amplitude Modulation.
BFO		6-45	Selects Beat Frequency Oscillation.
FM*n*	1, 2, 3 or 4	6-45	Selects Frequency Modulation.
PM		6-45	Selects Phase Modulation.
USER:			
FILTer *f*	3, 30 or 300	6-46	Sets User Defined Modulation IF Filter in kHz.
MODulation:*type*	FM, AM, BFO, PM or DATA (FM DATA)	6-46	Selects User Defined Modulation.
POST:			
APASs		6-46	Selects All Pass Post Detection Filter for User Defined Modulation.
BPASs *fl,fh*	0.5 to 20.0, 0.1 to 30.0	6-46	Selects Bandpass Post Detection Filter for User Defined Modulation, and specifies lower cutoff frequency (*fl*) and higher cutoff frequency (*fh*) in kHz.
CWEight		6-46	Selects C-Weighted Post Detection Filter for User Defined Modulation.
HPASs *fl*	0.5 to 20.0	6-46	Selects High-Pass Post Detection Filter for User Defined Modulation, and specifies cutoff frequency in kHz.
LPASs *fh*	0.1 to 30.0	6-46	Selects Low-Pass Post Detection Filter for User Defined Modulation, and specifies cutoff frequency in kHz.
MODulation?		6-47	Returns the current Duplex Transmitter Modulation (FM, AM, BFO, PM, DATA or USER).
SCAN:			
ABORt		6-47	Stops frequency scan or frequency list scan depending on present mode.
CONTinue		6-47	Starts frequency scan or frequency list scan depending on present mode.
FREQList:			
PAUSe *n*	0.0 to 99.9	6-48	Frequency List Scan Pause Time. Specifies time period in seconds for Duplex Transmitter to sit on a frequency if squelch is broken.
PAUSe?		6-48	Returns current value of Frequency List Scan Pause Time.
RATe *n*	0.02 to 99.99	6-48	Frequency List Scan Rate. Specifies time period in seconds for Duplex Transmitter to sit on a frequency unless squelch is broken.
RATe?		6-48	Returns current value of Frequency List Scan Rate.
SQUelch *b*	1 or 0	6-49	Enables/disables Squelch.
SQUelch?		6-49	Returns current state of Squelch.
FREQuency?		6-47	Returns current frequency in kHz being scanned.
INCrement *n*	0.0 to 2010250.0	6-47	(Frequency Scan mode only) Specifies increment in kHz between frequencies to be scanned.
PAUse *n*	0.0 to 99.9	6-47	(Frequency Scan mode only) Pause Time. Specifies time period in seconds for Duplex Transmitter to sit on a frequency if squelch is broken.
PAUse?		6-47	(Frequency Scan mode only) Returns a 1 if scanning is paused (stopped) or 0 if currently scanning.
RATe *n*	0.02 to 99.99	6-48	(Frequency Scan mode only) Scan Rate. Specifies time period in seconds for Duplex Transmitter to sit on a frequency unless squelch is broken.

COMMAND	RANGE	PAGE	DESCRIPTION
DUPlex:			
INPut:			
SCAN:			
STARt *f*	250.0 to 2010000.0	6-48	(Frequency Scan mode only) Starting Frequency value in kHz. Specifies lower limit frequency for scanning.
STOP *f*	250.0 to 2010000.0	6-48	(Frequency Scan mode only) Stop Frequency value in kHz. Specifies upper limit frequency for scanning.
TO:			
AUDio *b*	1 or 0	6-49	Connects/disconnects demodulated Receiver Input to AUDIO OUT Connector.
DEMOD *b*	1 or 0	6-49	Connects/disconnects demodulated Receiver Input to DEMOD OUT Connector.
SPEAKer *b*	1 or 0	6-49	Connects/disconnects demodulated Receiver Input to Test Set Speaker.
TR		6-49	Selects T/R Connector as Duplex Transmitter Input Source. Displays Power Meter on Duplex Transmitter Operation Screen.
VOLume:			
AUTO *b*	1 or 0	6-49	Enables/disables Automatic Volume Control.
AUTO?		6-49	Returns Automatic Volume Control state.
METER:			
DISTortion		6-40	Displays Distortion Meter on Duplex Operation Screen.
MODMeter		6-40	Displays Modulation Meter on Duplex Operation Screen.
OFF		6-40	Disables Modulation, Distortion and SINAD Meters.
SINAD		6-40	Displays SINAD Meter on Duplex Operation Screen.
OUTput:			
AUDio *b*	1 or 0	6-50	Connects/disconnects AF Generator Output to AUDIO OUT Connector.
CHANnel *n*	1 to 2047, depending on format	6-50	Sets Duplex Receiver Frequency to cellular channel *n*.
CHANnel:			
BAND?		6-51	Returns band for Duplex Receiver Channel Format. Returns one of the following bands for the associated channel format: (NADC) U8, U4 or HY; (ETACS) NOT AVAILABLE; (NAMPS) LOWER, MIDDLE or UPPER.
FORMat:			
AMPS:			
FORward		6-50	Selects AMPS (NADC-U8) Forward channels.
REVerse		6-50	Selects AMPS (NADC-U8) Reverse channels.
ETACS:			
FORward		6-50	Selects ETACS Forward channels.
REVerse		6-50	Selects ETACS Reverse channels.
NADC:			
BAND:*xx*	U8, U4 or HYper	6-50	Selects NADC band.
FORward		6-50	Selects NADC Forward. Utilizes current setting of NADC band.
REVerse		6-50	Selects NADC Reverse. Utilizes current setting of NADC band.
NAMPS:			
BAND:*x*	Lower, Middle or Upper	6-51	Selects NAMPS Narrow Analog channel designator.
FORward		6-51	Selects NAMPS Forward channels.
REVerse		6-51	Selects NAMPS Reverse channels.
NT400:			
FORward		6-51	Selects NT400 Forward channels.
REVerse		6-51	Selects NT400 Reverse channels.
FORMat?		6-51	Returns Duplex Receiver Channel Format (NADC:FORWARD, NADC:REVERSE, ETACS:FORWARD, ETACS:REVERSE, NAMPS:FORWARD or NAMPS:REVERSE).

COMMAND	RANGE	PAGE	DESCRIPTION
DUPlex:			
OUTput:			
DEMOD *b*	1 or 0	6-51	Connects/disconnects AF Generator Output to DEMOD OUT Connector.
DUPlex	1 or 0	6-52	Connects/disconnects RF Generator Output to DUPLEX OUT Connector.
FREQuency *n* [*units*]	250000.0 to 2010000000.0 [Hz] 250.0 to 2010000.0 [kHz] 0.2500 to 2010.0 [MHz] *units* - HZ, KHZ or MHz (KHZ default)	6-52	Sets Duplex Receiver Frequency.
FREQuency?		6-52	Returns Duplex Receiver Frequency in kHz.
LEVel:			
DBm *n*	-137.0 to 0.0	6-52	Sets RF Level in dBm.
DBm?		6-52	Returns RF Level in dBm.
METER:			
AF		6-52	Displays AF Meter on Duplex Receiver Operation Screen.
DISTortion		6-52	Displays Distortion Meter on Duplex Receiver Operation Screen.
DMM		6-52	Displays Digital Multimeter on Duplex Receiver Operation Screen.
SINAD		6-52	Displays SINAD Meter on Duplex Receiver Operation Screen.
MODE *type*	DIRect, CHANnel, SCAN, LIST, FLScan	6-53	Selects the Duplex Receiver Mode.
MODE?		6-53	Returns current mode (DIRECT, CHANNEL, FREQUENCY SCAN, FREQUENCY LIST or FREQUENCY LIST SCAN).
OFFSet *f*	-999749.9 to 999749.9	6-53	Sets Offset Frequency in kHz.
OFFSet?		6-53	Returns Offset Frequency in kHz.
SCAN:			
ABORt		6-53	Stops frequency scan or frequency list scan (depending on present mode).
CONTinue		6-53	Starts frequency scan or frequency list scan (depending on present mode).
FREQList:			
RATe *n*	0.02 to 99.99	6-54	Frequency List Scan Rate. Specifies time period in seconds for Duplex Receiver to generate current frequency prior to hopping to next frequency.
FREQuency?		6-53	Returns current frequency in kHz being generated.
INCrement *n*	0.0 to 2010000.0	6-53	(Frequency Scan mode only) Specifies increment in kHz between frequencies to be generated.
PAUse?		6-54	(Frequency Scan mode only) Returns a 1 if scanning is paused (stopped) or 0 if currently scanning.
RATe *n*	0.02 to 99.99	6-54	(Frequency Scan mode only) Scan Rate. Specifies time period in seconds for each generated frequency.
STARt *f*	250.0 to 2010000.0	6-54	(Frequency Scan mode only) Start Frequency. Specifies lower limit frequency in kHz for first generated frequency.
STOP *f*	250.0 to 2010000.0	6-54	(Frequency Scan mode only) Stop Frequency. Specifies upper limit frequency in kHz for last generated frequency.
TR		6-54	Routes Duplex Receiver Output to T/R Connector and disconnects DUPLEX OUT Connector.
RCL *n*	1 to 9	6-40	Recalls Duplex environment (routings and settings) stored in memory location *n*.

COMMAND	RANGE	PAGE	DESCRIPTION
DUPlex:			
SPEAKer:			
SOURce *type*	OFF, FGEN (Function Generator), SINAD (SINAD/BER IN Connector) or EXTMOD (EXT MOD IN Connector)	6-40	Selects Test Set Speaker Input. Command takes effect when RF Generator Operation Screen is updated.
STORe *n*	1 to 9	6-40	Stores current Duplex environment (routings and settings) in memory location *n*.

FETCH COMMANDS

COMMAND	RANGE	PAGE	DESCRIPTION
FETCh:			
AF?		6-172	Returns Audio Frequency Meter reading in kHz. Must be used with **INITiate** command.
DMM?		6-172	Returns Digital Multimeter reading. For ACV and DCV Multimeter Function, reading is returned in V. For ACC and DCC Multimeter Function, reading is returned in A. For Ohm Multimeter Function, reading is returned in kΩ. Must be used with **INITiate** command.
RF?		6-172	Returns Frequency Error Meter reading in kHz. Must be used with **INITiate** command.

FLASH MEMORY COMMANDS

See MMEMory:

FREQUENCY ERROR METER COMMANDS

See M_RF:

FUNCTION GENERATOR COMMANDS

COMMAND	RANGE	PAGE	DESCRIPTION
FGEN:			
BOOST *b*	1 or 0	6-56	Enables/disables Generator #1 Boost.
BOOST?		6-56	Returns current state of Generator #1 Boost.
DATA:			
LEVel *n*	0 to 4095	6-56	Specifies audio level for BER Data Generator.
LEVel?		6-56	Returns current audio level for BER Data Generator.
MODL *n*	AM: 0.0 to 100.0 (%) FM: 0.0 to 25.0 (kHz) PM: 0.0 to 10.0 (radians)	6-56	Sets Data Generator to *n* Modulation level.
MODL?		6-56	Returns Data Modulation level setting for current modulation type.
MODulation:*type*	OFF, AM, FM or PM	6-56	Sets Data Generator Modulation.
MODulation?		6-56	Returns Data Generator Modulation.
STATe *b*	1 or 0	6-57	Enables/disables Data Modulation.
EXT:			
LEVel *n*	0 to 100	6-57	Sets External Modulation Proportional output level to *n*%. Command has no effect if Proportional Mode is off.
LEVel?		6-57	Returns External Modulation Proportional output level setting in %.

COMMAND	RANGE	PAGE	DESCRIPTION
FGEN:			
EXT:			
MODL *n*	AM: 0 to 100 (%) FM: 0.0 to 10.0 (kHz) PM: 0.0 to 10.0 (radians)	6-57	Specifies External Modulation level.
MODL?		6-57	Returns External Modulation level setting.
MODulation:*type*	OFF, AM, FM or PM	6-57	Selects External Modulation.
MODulation?		6-57	Returns External Modulation type (OFF, AM, FM or PM).
STATe *b*	1 or 0	6-58	Enables/disables External Modulation.
FSK *x*	1 selects Generator #1, 0 selects Generator #2	6-58	Shifts output between Generator #1 and Generator #2.
GEN1:			
FREQuency *f*	0.0 to 40000.0	6-59	Sets AF Generator 1 output frequency in Hz.
FREQuency?		6-59	Returns AF Generator 1 output frequency in Hz.
LEVel *n*	0 to 100	6-59	Sets Proportional output level of AF Generator 1 in %. Command has no effect if Proportional Mode off.
LEVel?		6-59	Returns Proportional output level setting of AF Generator 1 in %.
MODL *n*	AM: 0 to 100 (%) FM: 0.0 to 100.0 (kHz) PM: 0.0 to 10.0 (radians)	6-59	Sets Modulation level of AF Generator 1.
MODL?		6-59	Returns Modulation level setting of AF Generator 1.
MODulation:*type*	OFF, AM, FM or PM	6-59	Selects Modulation for AF Generator 1.
MODulation?		6-59	Returns Modulation Type for AF Generator 1.
SHAPE:			
DC *n*	-1, 0 or 1	6-60	Sets wave shape to DC level *n* for AF Generator 1.
PULse:			
DCYCLe 50		6-60	Sets wave shape of AF Generator 1 to Pulse with 50% duty cycle.
SHAPE:*type*	SIN (Sine), SQU (Square Wave), RAMP or TRI (Triangle)	6-59	Selects wave shape for AF Generator 1.
SHAPE?		6-59	Returns wave shape of AF Generator 1.
STATe *b*		6-60	Enables/disables AF Generator 1.
GEN2:			
FREQuency *f*	0.0 to 40000.0	6-59	Sets AF Generator 2 output frequency in Hz.
FREQuency?		6-59	Returns AF Generator 2 output frequency in Hz.
LEVel *n*	0 to 100	6-59	Sets Proportional output level of AF Generator 2 in %. Command has no effect if Proportional Mode off.
LEVel?		6-59	Returns Proportional output level setting of AF Generator 2 in %.
MODL *n*	AM: 0 to 100 (%) FM: 0.0 to 100.0 (kHz) PM - 0.0 to 10.0 (radians)	6-59	Sets Modulation level of AF Generator 2.

COMMAND	RANGE	PAGE	DESCRIPTION
FGEN:			
GEN2:			
MODL?		6-59	Returns Modulation level setting of AF Generator 2.
MODulation:*type*	OFF, AM, FM or PM	6-59	Selects Modulation for AF Generator 2.
MODulation?		6-59	Returns Modulation Type for AF Generator 2.
SHAPE:			
DC *n*	-1, 0 or 1	6-60	Sets wave shape to DC level *n* for AF Generator 2.
PULse:			
DCYCLe 50		6-60	Sets wave shape of AF Generator 2 to Pulse with 50% duty cycle.
SHAPE:*type*	SIN (Sine), SQU (Square Wave), RAMP or TRI (Triangle)	6-59	Selects wave shape for AF Generator 2.
SHAPE?		6-59	Returns wave shape of AF Generator 2.
STATe *b*	1 or 0	6-60	Enables/disables AF Generator 2.
GEN3:			
DIGital *type*	DCS, DCSINV, POCSAG, DSAT or DST	6-61	Sets digital *type*.
ENCode *type*	DTMF, TONE, DIGital or RCC	6-61	Selects signaling format to encode. Must be followed with a **SETUP:GEN** or **SCREEN:GEN** command.
MODL *n*	AM: 0 to 100.0 (%) PM: 0.0 to 10.0 (radians) FM (Tone/RCC): 0.0 to 100.0 (kHz) FM (DTMF): 0.0 to 10.0 (kHz) FM (Digital): 0.0 to 25.0 (kHz)	6-61	Sets AF Generator 3 Modulation level.
MODL?		6-61	Returns AF Generator 3 Modulation level setting.
MODulation:*type*	OFF, AM, FM or PM	6-61	Selects AF Generator 3 Modulation.
MODulation?		6-61	Returns AF Generator 3 Modulation type.
RCC *format*	IMTS, MTS, SYS2805 or TREMote	6-61	Selects Generator RCC signaling format.
MIC:			
LEVel *n*	0 to 100	6-61	Sets Proportional output level of MIC/ACC Connector input in %. Command has no effect if Proportional Mode off.
LEVel?		6-61	Returns Proportional output level setting for MIC/ACC Connector input.
MODL *n*	AM: 0 to 100 (%) FM: 0.0 to 25.0 (kHz) PM: 0.0 to 10.0 (radians)	6-62	Sets MIC/ACC Connector input Modulation level.
MODL?		6-62	Returns MIC/ACC Connector input Modulation level setting.
MODulation:*type*	OFF, AM, FM or PM	6-62	Selects MIC/ACC Connector input Modulation type.
MODulation?		6-62	Returns MIC/ACC Connector input Modulation type.
STATe *b*	1 or 0	6-62	Enables/disables external modulation through MIC/ACC Connector.

COMMAND	RANGE	PAGE	DESCRIPTION
FGEN:			
OUTput:			
AUDio *b*	1 or 0	6-63	Connects/disconnects AF Generator Output to AUDIO OUT Connector.
AUDio?		6-63	Returns 1 if AF Generator Output is routed to the AUDIO OUT Connector, 0 if AUDIO OUT Connector is disconnected.
DEMOD *b*	1 or 0	6-63	Connects/disconnects AF Generator Output to DEMOD OUT Connector.
DEMOD?		6-63	Returns 1 if AF Generator Output is routed to DEMOD OUT Connector, 0 if DEMOD OUT Connector is disconnected.
LEVel *v*	0.0000 to 3.1000	6-63	Sets AF Generator output level in volts.
LEVel?		6-63	Returns AF Generator output level in volts.
SPEAKer *b*	1 or 0	6-63	Connects/disconnects AF Generator to Speaker.
SPEAKer?		6-63	Returns 1 if AF Generator Output is routed to Speaker, 0 if Speaker is disconnected.
PROPortional *b*	1 or 0	6-64	Enables/disables AF Generator Proportional mode.
PROPortional?		6-64	Returns 1 if AF Generator is in Proportional mode; 0 otherwise.
RCL *n*	1 to 9	6-64	Recalls AF Generator environment (routings and settings) stored in memory location *n*.
STORe *n*	1 to 9	6-64	Stores current AF Generator environment (routings and settings) in memory location *n*.
PTT:			
STATe *b*	1 = high (mic keyed), 0 = low	6-62	Sets output on the push to talk pin on the MIC/ACC Connector.

GENERATOR COMMANDS

COMMAND	RANGE	PAGE	DESCRIPTION
GENerator:			
AF		6-6	Displays AF Level Meter on RF Generator Operation Screen when followed by **SCREEN:GENerator** command.
CHANnel *n*	1 to 2047, depending on format	6-6	Sets cellular channel in format selected using the **GEN:CHAN:FORM** command.
CHANnel:			
BAND?		6-7	Returns band for Generator Channel Format (NADC - U8, U4 or HY; ETACS - NOT AVAILABLE; NAMPS - LOWER, MIDDLE or UPPER).
FORMat:			
AMPS:			
FORward		6-6	Selects AMPS (NADC-U8) Forward channels.
REVerse		6-6	Selects AMPS (NADC-U8) Reverse channels.
ETACS:			
FORward		6-6	Selects ETACS Forward channels.
REVerse		6-6	Selects ETACS Reverse channels.
NADC:			
BAND:*xx*	U8, U4 or HYper	6-7	Selects NADC band.
FORward		6-7	Selects NADC Forward. Utilizes current setting of NADC band.
REVerse		6-7	Selects NADC Reverse. Utilizes current setting of NADC band.
NAMPS:			
BAND:*x*	Lower, Middle or Upper	6-6	Selects NAMPS channel designator.
FORward		6-6	Selects NAMPS Forward channels.
REVerse		6-6	Selects NAMPS Reverse channels.
NT400:			
FORward		6-7	Selects NT400 Forward channels.
REVerse		6-7	Selects NT400 Reverse channels.
FORMat?		6-7	Returns Generator Channel Format (NADC:FORWARD, NADC:REVERSE, ETACS:FORWARD, ETACS:REVERSE, NAMPS:FORWARD or NAMPS:REVERSE).

COMMAND	RANGE	PAGE	DESCRIPTION
GENerator:			
DCS:			
INVert *nnn*	000 to 777	6-8	Generates three digit octal Digital Coded Squelch (DCS) code in inverted mode.
NORMal *nnn*	000 to 777	6-8	Generates three digit octal DCS code.
STOP		6-8	Stops generating continuous DCS code.
DIAL "*sequence*"	0 to 9 (16 digits max.)	6-8	Generates *sequence* as 2805 Pulse code.
DIAL:			
FREQuency *f*	0.0 to 40000.0	6-8	Sets 2805 Tone frequency in Hz.
FREQuency?		6-8	Returns the current 2805 Tone frequency in Hz.
DISTortion		6-8	Displays Distortion Meter on RF Generator Operation Screen after screen is updated by **SCREEN:GENerator** command.
DMM		6-8	Displays Digital Multimeter on RF Generator Operation Screen after screen is updated by **SCREEN:GENerator** command.
DSAT *n*	0 to 6	6-9	Generates Digital Supervisory Audio Tone (DSAT) code.
DSAT:			
STOP		6-9	Stops generating continuous DSAT code.
DST *n*	0 to 6	6-9	Generates Digital Signal Tone (DST) code.
DST:			
STOP		6-9	Stops generating continuous DST code.
DTMF "*sequence*",[*mark,space*]	0 to 9 (16 digits max.) 25 to 9999 ms for mark or space. (Default for *mark* time is 74 ms. Default for *space* time is 67 ms.)	6-9	Generates DTMF coded *sequence*.
FREQuency *n* [*units*]	250000.0 to 2010000000.0 [Hz] 250.0 to 2010000.0 [kHz] 0.2500 to 2010.0 [MHz] *units* - HZ, KHZ or MHz (KHZ, default)	6-9	Sets RF Generator Frequency.
FREQuency?		6-9	Returns RF Generator Frequency in kHz (250.0 to 2010000.0).
IMTS "*sequence*"	0 to 9 (16 digits max.)	6-9	Generates DCS IMTS code.
LEVel *DBm n*	-137.0 to 0.0	6-10	Sets RF Generator Level in dBm.
LEVel *DBm?*		6-10	Returns RF Generator Level in dBm.

COMMAND	RANGE	PAGE	DESCRIPTION
GENerator:			
LEVel *n* [*units*]	-137.0 to 0.0 (dBm), 0.031 to 224000.0 (µV), 0.000031 to 224.0 (mV) or 0.000000031 to 0.224 (V) units - DBm, V, MV (mV) or UV (µV) (*units* are optional, defaults to current units.)	6-9	Sets RF Generator Level. Range of *n* is -137.0 to 0.0 dBm or 0.031 µV to 0.224 V. Select DBm, V, MV (mV) or UV (µV) for *units*. *units* is optional with the default being the current units. Specifying *units* does not change unit selection of Test Set.
LEVel:			
UNIT		6-10	Toggles RF Generator level units between dBm and Volts.
UNIT?		6-10	Returns current units for RF Generator Level. Returns DBM or V.
LEVel?		6-10	Returns RF Generator Level in current units.
MODE *type*	DIRect, CHANnel, SCAN, LIST, FLScan	6-10	Selects Generator Mode.
MODE?		6-10	Returns current mode (DIRECT, CHANNEL, FREQUENCY SCAN, FREQUENCY LIST or FREQUENCY LIST SCAN).
MTS "*sequence*"	0 to 9 (16 digits max.)	6-10	Generates DCS MTS code.
OUTput:			
AUDio *b*	Connect = 1, Disconnect = 0	6-11	Routes AF Generator Output to AUDIO OUT Connector.
DEMOD *b*	Connect = 1, Disconnect = 0	6-11	Routes AF Generator Output to DEMOD OUT Connector.
POCSAG:			
ALPHA:			
LOWer *capcode*	0 to 2097151	6-11	Generates lower case Alpha message for *capcode* specified.
NUMeric *capcode*	0 to 2097151	6-11	Generates Alphanumeric message for *capcode* specified.
SPECial *capcode*	0 to 2097151	6-11	Generates an Alpha special message for the *capcode* specified.
UPPer *capcode*	0 to 2097151	6-11	Generates upper case Alpha message for *capcode* specified.
BEEP *n, capcode*	1, 2, 3 or 4; 0 to 2097151	6-11	Generates *n* Tone Beep POCSAG message for *capcode* specified.
NUMeric *capcode*	0 to 2097151	6-11	Generates Numeric message for *capcode* specified.
RATe	high = 1, low = 0	6-12	Sets POCSAG rate.
RATe?		6-12	Returns 1 if POCSAG rate is high; returns 0 if rate is low.
RCL *n*	1 to 9	6-12	Recalls RF Generator environment (routings and settings) stored in memory location *n*.
SCAN:			
ABORt		6-12	Stops frequency scan or frequency list scan depending on present mode.
CONTinue		6-12	Starts frequency scan or frequency list scan depending on present mode.
FREQList:			
RATe *n*	0.02 to 99.99	6-13	Frequency List Scan Rate. Specifies time period in seconds for RF Generator to sit on a frequency.
RATe?		6-13	Returns current value of Frequency List Scan Rate.
FREQuency?		6-12	Returns current frequency in kHz being scanned.
INCrement *n*	0.0 to 2010000.0	6-12	(Frequency Scan mode only) Specifies increment in kHz between frequencies to be scanned.

COMMAND	RANGE	PAGE	DESCRIPTION
GENerator:			
SCAN:			
PAUse?		6-13	(Frequency Scan mode only) Returns a 1 if scanning is paused (stopped) or 0 if currently scanning.
RATe *n*	0.02 to 99.99	6-13	(Frequency Scan mode only) Scan Rate. Specifies time period in seconds for RF Generator to sit on a frequency unless squelch is broken.
STARt *f*	250.0 to 2010000.0	6-13	(Frequency Scan mode only) Starting Frequency value in kHz. Specifies lower limit frequency for scanning.
STOP *f*	250.0 to 2010000.0	6-13	(Frequency Scan mode only) Stop Frequency value in kHz. Specifies upper limit frequency for scanning.
SINAD		6-13	Displays SINAD Meter on RF Generator Operation Screen when followed by a **SCREEN:GENerator** command.
SPEAKer:			
SOURce *type*	OFF, FGEN (Function Generator), SINAD (SINAD/BER IN Connector) or EXTMOD (EXT MOD IN Connector)	6-13	Selects Test Set Speaker Input.
STORe *n*	1 to 9	6-14	Stores current RF Generator environment (routings and settings) in memory location *n*.
TONE "*sequence*"	0 to 9, A, G, R and - (to signify a gap). For USER code: 0 to 9, and A to T	6-14	Generates given *sequence* once Audio code is selected using **GENerator:TONE:TYPE** command. If selected Audio code is USER, characters contained in the *sequence* must be previously defined using the **GENerator:TONE:USER: DEFine** command.
TONE:			
TYPE *code*	CCIR, EEA, EIA (U.S.), ZVEI, DDZVEI, DZVEI, NATEL, EURO, TONE56, CCIRH, CCIRH4, USER	6-14	Selects Audio code to generate.
USER:			
DEFine "*id*",*freq*,*duration*	*id*: 0 to 9 and A to T; *freq*: 0.0 to 9999.9; *duration*: 20.0 to 9999.9	6-14	The *id* character is assigned a *freq* in Hz and a *duration* in ms.
TREMote	2050, 1950, 1850, 1750, 1650, 1550, 1450, 1350, 1250, 1150, 1050	6-14	Generates Tone-remote *sequence* for specified function tone frequency given by *f* in Hz.
TREMote:			
STOP		6-14	Stops Tone remote Guard Tone generated by previous **GENerator:TREMote** command.

GENERIC MEASURE COMMANDS

See Measure (Generic) Commands.

COMMAND	RANGE	PAGE	DESCRIPTION
INITIATE COMMANDS			
INITiate:			
AF		6-172	Prepares Audio Frequency Meter for a **FETCh:** command to take reading.
DMM		6-172	Prepares Audio Frequency Meter for a **FETCh:** command to take reading.
RF		6-172	Prepares Frequency Error Meter for a **FETCh:** command to take a reading.
INSTRUMENT COMMAND			
INSTrument:			
SELect *type*	GENerator, RECeiver, DUPlex, FGEN, SCOPe, ANLZ, M_AF, M_RF, M_PWR, M_DEV, M_MOD, M_DIST, M_SINAD, M_SIG, M_BER, M_DMM, M_PM, M_DRMS, M_PMRMS	6-5	Selects default instrument.
AF METER COMMANDS			
M_AF:			
ALARM *b*	1 or 0	6-92	Enables/disables Alarm.
FILTer:			
APASs		6-92	Selects All-Pass Filter for AF Meter.
APASs?		6-92	Returns 1 if All-Pass Filter is enabled, 0 if disabled.
HPASs:			
FREQuency *f*	0.5 to 20.0	6-92	Sets High-Pass cutoff frequency in kHz.
FREQuency?		6-92	Returns High-Pass cutoff frequency in kHz.
STATe *b*	1 or 0	6-92	Enables/disables High-Pass Filter.
STATe?		6-92	Returns 1 if High-Pass Filter is enabled, 0 if disabled.
LPASs:			
FREQuency *f*	0.1 to 30.0	6-92	Set Low-Pass cutoff frequency in kHz.
FREQuency?		6-92	Returns Low-Pass cutoff frequency in kHz.
STATe *b*		6-93	Enables/disables Low-Pass Filter.
STATe?		6-93	Returns 1 if Low-Pass Filter is enabled, 0 if disabled.
INPut:*type*	XAUDIO (Ext Mod), DEMOD (Demod Audio), FGEN (Func Gen Out), SINAD (SINAD/BER), POWer (RF Power)	6-93	Selects Audio Frequency Meter Input.
LL:			
LEVel *f*	0.0000 to 0.2000) (for an upper range of 0.2) 0.000 to 200.000 (for upper ranges of 2, 20 and 200)	6-93	Sets Lower Limit in kHz.
STATe *b*	1 or 0	6-93	Enables/disables Audio Frequency Meter Lower Limit.
PEAK?		6-93	Returns Audio Frequency Meter Peak reading in Hz (0.0 to 200000.0).
PH *b*	1 or 0	6-93	Enables/disables Peak Hold Feature.

COMMAND	RANGE	PAGE	DESCRIPTION
M_AF:			
RANGe:			
AUTO		6-94	Sets frequency range to Autorange.
UPPer *f*	0.2, 2, 20 or 200	6-94	Sets frequency range in kHz.
RCL *n*	1 to 9	6-94	Recalls Audio Frequency Meter environment (routings and settings) stored in memory location *n*.
RESolution *n*	0.1 (Gate Time of) 10 s) or 1 (Gate Time of 1 s)	6-94	Sets Audio Frequency Meter Resolution (Gate Time) in Hz.
STORe *n*	1 to 9	6-94	Stores current Audio Frequency Meter environment (routings and settings) in memory location *n*.
UL:			
LEVel *f*	0.0000 to 0.2000 (for an upper range of 0.2) 0.000 to 200.000 (for upper ranges of 2, 20 and 200)	6-94	Sets Upper Limit in kHz.
STATe *b*	1 or 0	6-94	Enables/disables Audio Frequency Meter Upper Limit.
M_AF?		6-95	Returns Audio Frequency Meter reading in Hz (0.0 to 200000.0).

BER (BIT ERROR RATE) METER COMMANDS

COMMAND	RANGE	PAGE	DESCRIPTION
M_BER:			
PATtern:			
FIXED		6-113	Selects Fixed pattern for the BER Meter test data.
FIXEDUSER?		6-113	Returns specified USER data pattern.
RANDom		6-113	Selects Random pattern for BER Meter test data.
USER *nn*	*nn* = 8 bit pattern.	6-113	Selects USER data pattern (*nn* could be a variable or expression) for the BER Meter test data. Patterns entered that are not base 10 are preceded with # character and letter signifying the number base: H (Hexadecimal), B (Binary) or Q (Octal). Pattern is displayed in Hexadecimal.
POLarity:			
NEGative		6-113	Selects Negative Polarity for BER Meter.
POSitive		6-113	Selects Positive Polarity for BER Meter.
POLarity?		6-113	Returns selected data Polarity. Returns POSITIVE or NEGATIVE.
RATE *n*	75, 150, 300, 600, 1200, 2400, 4800, 16000	6-113	Sets BER Meter rate.
RATE?		6-113	Returns BER Meter Rate setting.
RCL *n*	1 to 9	6-114	Recalls BER Meter environment (routings and settings) stored in memory location *n*.
SIZE *n*	100 to 100000	6-114	Sets BER Meter block size in bits.
SIZE?		6-114	Returns BER Meter block size setting.
STORe *n*	1 to 9	6-114	Stores current BER Meter environment (routings and settings) in memory location *n*.
TYPE:*xxx*	GENerator, RECeiver, DUPlex or BASEband	6-114	Selects Bit Error Rate Type.
M_BER?		6-114	Returns number of errors for last pass.

COMMAND	RANGE	PAGE	DESCRIPTION
DEVIATION METER (PEAK) COMMANDS			
M_DEV:			
ALARM *b*	1 or 0	6-101	Enables/disables Deviation Meter Alarm.
AVErage *b*	1 or 0	6-101	Enables/disables Deviation Meter Averaging.
LL:			
LEVel *f*	0.00 to 20.00 (for range values of 2, 5, 10 or 20 kHz) 0 to 100 (for all other ranges)	6-101	Sets Deviation Meter Lower Limit in kHz.
STATe *b*	1 or 0	6-101	Enables/disables Deviation Meter Lower Limit.
MODE:			
BOTH		6-101	Selects Both Mode for Deviation Meter, reading positive and negative deviation.
NEGative		6-101	Selects Negative Mode for Deviation Meter, reading negative deviation.
NORMalize		6-101	Selects Normalized Mode for Deviation Meter, reading (positive + negative)/2 deviation.
POSitive		6-101	Selects Positive Mode for Deviation Meter, reading positive deviation.
NEG?		6-102	Returns negative Deviation Meter reading as an absolute value in kHz (0.00 to 100.00).
PEAK:			
NEG?		6-102	Returns negative Deviation Meter Peak reading as an absolute value in kHz (0.00 to 100.00).
POS?		6-102	Returns positive Deviation Meter Peak reading as an absolute value in kHz (0.00 to 100.00).
PH *b*	1 or 0	6-102	Enables/disables Deviation Meter Peak Hold Feature.
POS?		6-102	Returns positive Deviation Meter reading as an absolute value in kHz (0.00 to 100.00).
RANGe:			
AUTO		6-102	Sets Deviation Meter range to Autorange.
UPPer *f*	2, 5, 10, 20, 50 or 100	6-102	Sets Deviation Meter range in kHz.
RCL *n*	1 to 9	6-103	Recalls Deviation Meter environment (routings and settings) stored in memory location *n*.
STORe *n*	1 to 9	6-103	Stores current Deviation Meter environment (routings and settings) in memory location *n*.
UL:			
LEVel *f*	0.00 to 20.00 (for range values of 2, 5, 10 or 20 kHz) 0 to 100 (for all other ranges)	6-103	Sets Deviation Meter Upper Limit in kHz.
STATe *b*	1 or 0	6-103	Enables/disables Deviation Meter Upper Limit.
DISTORTION METER COMMANDS			
M_DIST:			
ALARM *b*	1 or 0	6-106	Enables/disables Distortion Meter Alarm.
AVErage *b*	1 or 0	6-106	Enables/disables Average Feature.
FILTer *f*	600 to 1400	6-106	Sets Notch Filter frequency in Hz.
INPut:*type*	DEMOD (Demod Audio), SINAD (SINAD/BER), XAUDio (Ext Mod) or FGEN (Func Gen)	6-106	Sets Distortion Meter Input.

COMMAND	RANGE	PAGE	DESCRIPTION
M_DIST:			
LL:			
LEVel *n*	0.0 to 20.0	6-106	Sets Lower Limit in %.
STATe *b*	1 or 0	6-106	Enables/disables Distortion Meter Lower Limit.
PEAK?		6-106	Returns Distortion Meter Peak reading as a percentage (0.0 to 20.0).
PH *b*		6-106	Enables/disables Distortion Meter Peak Hold Feature. Peak Hold takes effect only after **SCREEN:DISTortion** command.
RCL *n*	1 to 9	6-107	Recalls Distortion Meter environment (routings and settings) stored at memory location *n*.
SELect:			
CMESsage		6-107	Same as **M_DIST:SELect:CWEight** command.
CWEight		6-107	Selects the C-Weight Filter.
LPASs *f*	100 to 30000	6-107	Selects Low-Pass Filter with cutoff frequency of *f* Hz.
STORe *n*	1 to 9	6-107	Stores current Distortion Meter environment (routings and settings) in memory location *n*.
UL:			
LEVel *n*	0.0 to 20.0	6-107	Sets Upper Limit to in %.
STATe *b*	1 or 0	6-107	Enables/disables Distortion Meter Upper Limit.
M_DIST?		6-108	Returns Distortion Meter reading as a percentage (0.0 to 20.0).

DIGITAL MULTIMETER COMMANDS

COMMAND	RANGE	PAGE	DESCRIPTION
M_DMM:			
ALARM *b*		6-115	Enables/disables Multimeter Alarm.
FUNCtion:			
CURRent:			
AC		6-115	Selects AC Ammeter for Multimeter Function.
DC		6-115	Selects DC Ammeter for Multimeter Function.
RESistance		6-115	Selects Ohmmeter for Multimeter Function.
VOLTage:			
AC		6-115	Selects AC Voltmeter for Multimeter Function.
DC		6-115	Selects DC Voltmeter for Multimeter Function.
FUNCtion?		6-115	Returns active Multimeter Function (CURR:AC, CURR:DC, RES, VOLT:AC or VOLT:DC).
INPut:			
IMPedance *n*	150, 600 or 1e6	6-115	Sets Input Impedance in ohms. Command ignored if not in AC Voltmeter Function.
LL:			
LEVel *n*	Ammeter (dc or ac): 0.00000 to 0.19990 (A) (20 and 200 mA range). 0.000 to 19.990 (A) (2 A and 20 A range). Voltmeter (dc or ac): 0.0000 to 0.1999 (V) (200 mV range). 0.00 to 1000.00 (V) (2, 20, 200, 2000 V range). Ohmmeter: 0.0000 to 0.1999 (kΩ) (200 Ω range). 0.000 to 19990 (kΩ) (2 kΩ to 20 MΩ range).	6-116	Sets Multimeter Lower Limit.
STATe *b*	1 or 0	6-116	Enables/disables Multimeter Lower Limit.
PH *b*	1 or 0	6-116	Enables/disables Multimeter Peak Hold feature.

COMMAND	RANGE	PAGE	DESCRIPTION
M_DMM:			
RANGe:			
AUTO		6-116	Sets Multimeter range to Autorange.
UPPer *n*	Voltmeter (dc or ac): 0.2, 2, 20, 200 or 2000 (V) Ammeter (dc or ac): 0.02, 0.2, 2 or 20 (A) Ohmmeter: 0.2, 2, 20, 200, 2000 or 20000 (kΩ)	6-116	Sets Multimeter range.
RCL *n*	1 to 9	6-117	Recalls Digital Multimeter environment (routings and settings) stored in memory location *n*.
STORe *n*	1 to 9	6-117	Stores current Digital Multimeter environment (routings and settings) in memory location *n*.
UL:			
LEVel *n*	Ammeter (dc or ac): 0.00000 to 0.19990 (A) (20 and 200 mA range). 0.000 to 19.990 (A) (2 A and 20 A range). Voltmeter (dc or ac): 0.0000 to 0.1999 (V) (200 mV range). 0.00 to 1000.00 (V) (2, 20, 200, 2000 V range). Ohmmeter: 0.0000 to 0.1999 (kΩ) (200 Ω range). 0.000 to 19990 (kΩ) (2 kΩ to 20 MΩ range).	6-117	Sets Multimeter Upper Limit.
STATe *b*	1 or 0	6-117	Enables/disables Multimeter Upper Limit.
M_DMM?		6-117	Returns Multimeter reading in current Function units.

DEVIATION METER (RMS) COMMANDS

COMMAND	RANGE	PAGE	DESCRIPTION
M_DRMS:			
ALARM *b*	1 or 0	6-120	Enables/disables Deviation Meter (RMS) Alarm.
AVErage *b*	1 or 0	6-120	Enables/disables Deviation Meter (RMS) Averaging.
LL:			
LEVel *f*	0.00 to 10.00	6-120	Sets Deviation (RMS) Lower Limit in kHz.
STATe *b*	1 or 0	6-120	Enables/disables Deviation (RMS) Meter Lower Limit.
PH *b*	1 or 0	6-120	Enables/disables Deviation Meter (RMS) Peak Hold feature.
RANGe:			
AUTO		6-120	Sets Deviation Meter (RMS) range to Autorange.
UPPer *f*	2, 5 or 10	6-120	Sets Deviation Meter (RMS) range in kHz.
RCL *n*	1 to 9	6-120	Recalls Deviation Meter (RMS) environment (routings and settings) stored in memory location *n*.
STORe *n*	1 to 9	6-120	Stores current Deviation Meter (RMS) environment (routings and settings) in memory location *n*.
UL:			
LEVel *f*	0.00 to 10.00	6-121	Sets Deviation (RMS) Upper Limit in kHz.
STATe *b*	1 or 0	6-121	Enables/disables Deviation Meter (RMS) Upper Limit.
M_DRMS?		6-121	Returns Deviation Meter (RMS) reading in kHz (0.00 to 10.00).

COMMAND	RANGE	PAGE	DESCRIPTION
MODULATION METER COMMANDS			
M_MOD:			
ALARM *b*	1 or 0	6-104	Enables/disables Modulation Meter Alarm.
LL:			
LEVel *n*	0.0 to 100.0	6-104	Sets Modulation Meter Lower Limit in %.
STATe *b*	1 or 0	6-104	Enables/disables Modulation Meter Lower Limit.
PEAK?		6-104	Returns Modulation Meter Peak reading as a percentage (0.0 to 100.0).
PH *b*		6-104	Enables/disables Modulation Meter Peak Hold Feature.
RANGe:			
AUTO		6-104	Sets Modulation Meter range to Autorange.
UPPer *n*	40 or 100	6-104	Sets Modulation Meter range in %.
RCL *n*	1 to 9	6-105	Recalls Modulation Meter environment (routings and settings) stored in memory location *n*.
STORe *n*	1 to 9	6-105	Stores current Modulation Meter environment (routings and settings) in memory location *n*.
UL:			
LEVel *n*	0.0 to 100.0	6-105	Sets Modulation Meter Upper Limit in %.
STATe *b*	1 or 0	6-105	Enables/disables Modulation Meter Upper Limit.
M_MOD?		6-105	Returns Modulation Meter reading as a percentage (0.0 to 100.0).
PHASE METER COMMANDS			
M_PM:			
ALARM *b*	1 or 0	6-118	Enables/disables Phase Meter Alarm.
LL:			
LEVel *n*	0.00 to 10.00	6-118	Sets Phase Meter Lower Limit in radians.
STATe *b*	1 or 0	6-118	Enables/disables Phase Meter Lower Limit.
PH *b*	1 or 0	6-118	Enables/disables Phase Meter Peak Hold Feature.
RANGe:			
AUTO		6-118	Sets Phase Meter range to Autorange.
UPPer *n*	1, 5 or 10	6-118	Sets Phase Meter range in radians.
RCL *n*	1 to 9	6-118	Recalls Phase Meter environment (routings and settings) stored in memory location *n*.
STORe *n*	1 to 9	6-119	Stores current Phase Meter environment (routings and settings) in memory location *n*.
UL:			
LEVel *n*	0.00 to 10.00	6-119	Sets Phase Meter Upper Limit in radians.
STATe *b*	1 or 0	6-119	Enables/disables Phase Meter Upper Limit.
M_PM?		6-119	Returns Phase Meter reading in radians (0.00 to 10.00).
PHASE METER (RMS) COMMANDS			
M_PMRMS:			
ALARM *b*	1 or 0	6-122	Enables/disables Phase Meter (RMS) Alarm.
AVErage *b*	1 or 0	6-122	Enables/disables Averaging mode.
LL:			
LEVel *n*	0.00 to 10.00	6-122	Sets Phase Meter (RMS) Lower Limit in radians.
STATe *b*	1 or 0	6-122	Enables/disables Phase Meter (RMS) Lower Limit.
PH *b*	1 or 0	6-122	Enables/disables Phase Meter (RMS) Peak Hold feature.
RANGe:			
AUTO		6-122	Sets Phase Meter (RMS) range to Autorange.
UPPer *n*	1, 5 or 10	6-122	Sets Phase Meter (RMS) range in radians.
RCL *n*	1 to 9	6-122	Recalls Phase Meter (RMS) environment (routings and settings) stored in memory location *n*.
STORe *n*	1 to 9	6-122	Stores current Phase Meter (RMS) environment (routings and settings) in memory location *n*.
UL:			
LEVel *n*	0.00 to 10.00	6-123	Sets Phase Meter (RMS) Upper Limit in radians.
STATe *b*	1 or 0	6-123	Enables/disables Phase Meter (RMS) Upper Limit.
M_PMRMS?		6-123	Returns Phase Meter (RMS) reading in radians (0.00 to 10.00).

COMMAND	RANGE	PAGE	DESCRIPTION
POWER METER COMMANDS			
M_PWR:			
ALARM *b*	1 or 0	6-98	Enables/disables Power Meter Alarm.
DBM[:STATE] *b*	1 or 0	6-98	Enables/disables dBm digital readout. **:STATE** portion of the command is optional.
DBM[:STATE]?		6-98	Returns state of dBm digital readout. **:STATE** portion of the command is optional.
EXT:			
OFFSet *n*	-99.9 to 99.9	6-98	Sets External Loss/Gain Offset value in dBm.
OFFSet?		6-98	Returns External Loss/Gain Offset value in dBm.
STATe *b*	1 or 0	6-98	Enables/disables External Loss/Gain Offset.
STATe?		6-98	Returns External Loss/Gain Offset state setting.
LL:			
LEVel *n*	0.0000 to 0.5000 (for ranges: 0.02, 0.05, 0.1, 0.2 and 0.5). 0.00 to 200.00 (for all other ranges).	6-98	Sets Power Meter Lower Limit in W.
STATe *b*	1 or 0	6-98	Enables/disables Power Meter Lower Limit.
PEAK?		6-99	Returns Power Meter Peak reading in mW (0.0 to 200000.0).
PH *b*	1 or 0	6-99	Enables/disables Power Meter Peak Hold Feature.
RANGe:			
AUTO		6-99	Sets Power Meter range to Autorange.
UPPer *n*	0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 200	6-99	Sets Power Meter range in W.
RCL *n*	1 to 9	6-99	Recalls Power Meter environment (routings and settings) stored in memory location *n*.
RF:			
ASSUMEd *n*	250.0 to 2010000.0	6-99	Sets Assumed RF Frequency, in kHz, within 1 MHz of signal measured.
ASSUMEd?		6-99	Returns Assumed RF Frequency in kHz.
STORe *n*	1 to 9	6-99	Stores current Power Meter environment (routings and settings) in memory location *n*.
TYPE:			
CW		6-100	Selects Average Power Measurement Type for Power Meter.
PEAK		6-100	Selects Peak Power Measurement Type for Power Meter.
RMS		6-100	Selects RMS Power Measurement Type for Power Meter.
UL:			
LEVel *n*	0.0000 to 0.5000 (for upper ranges: 0.02, 0.05, 0.1, 0.2 and 0.5). 0.00 to 200.00 (for all other ranges)	6-100	Sets Power Meter Upper Limit in W.
STATe *b*	1 or 0	6-100	Enables/disables Power Meter Upper Limit.
M_PWR?		6-100	Returns a Power Meter reading in mW (0.0 to 200000.0).
FREQUENCY ERROR METER COMMANDS			
M_RF:			
ALARM *b*	1 or 0	6-96	Enables/disables Alarm.
LL:			
LEVel *f*	0.0000 to 0.1000 (for an upper range of 0.1). 0.000 to 100.000 (for upper ranges of 1, 10 and 100)	6-96	Sets Lower Limit in kHz.
STATe *b*		6-96	Enables/disables Lower Limit.

COMMAND	RANGE	PAGE	DESCRIPTION
M_RF:			
PEAK?		6-96	Returns Frequency Error Meter Peak reading in Hz (0.0 to 100000.0).
PH *b*		6-96	Enables/disables Peak Hold Feature.
RANGe:			
AUTO		6-96	Sets Frequency Error Meter range to Autorange.
UPPer *f*	0.1, 1, 10 or 100	6-96	Sets Frequency Error Meter range in kHz.
RCL *n*	1 to 9	6-96	Recalls Frequency Error Meter environment (routings and settings) stored in memory location *n*.
RESolution *f*	1 (Gate Time of 1 sec) or 10 (Gate Time of 0.1 sec)	6-97	Sets Frequency Error Resolution (Gate Time) in Hz.
STORe *n*	1 to 9	6-97	Stores current Frequency Error Meter environment (routings and settings) in memory location *n*.
UL:			
LEVel *f*	0.0000 to 0.1000 (for an upper range of 0.1). 0.000 to 100.000 for upper ranges of 1, 10 and 100	6-97	Sets Upper Limit in kHz.
STATe *b*	1 or 0	6-97	Enables/disables Upper Limit.
M_RF?		6-97	Returns Frequency Error Meter reading in Hz (-100000.0 to +100000.0).

SIGNAL STRENGTH METER COMMANDS

COMMAND	RANGE	PAGE	DESCRIPTION
M_SIG:			
PEAK?		6-112	Returns a Signal Strength Meter Peak reading (0 to 100).
PH *b*	1 or 0	6-112	Enables/disables Peak Hold Feature.
RCL *n*	1 to 9	6-112	Recalls Signal Strength Meter environment (routings and settings) stored in memory location *n*.
STORe *n*	1 to 9	6-112	Stores current Signal Strength Meter environment (routings and settings) in memory location *n*.
M_SIG?		6-112	Returns a Signal Strength Meter reading (0 to 100).

SINAD METER COMMANDS

COMMAND	RANGE	PAGE	DESCRIPTION
M_SINAD:			
ALARM *b*	1 or 0	6-109	Enables/disables SINAD Meter Alarm.
AVErage *b*	1 or 0	6-109	Enables/disables Average Feature.
FILTer *f*	600 to 1400	6-109	Sets Notch frequency in Hz.
INPut:*type*	DEMOD) (Demod Audio), SINAD (SINAD/BER), XAUDio (Ext Mod) or FGEN (Func Gen	6-109	Selects SINAD Meter Input.
LL:			
LEVel *n*	3.0 to 40.0	6-109	Sets SINAD Meter Lower Limit in dB.
STATe *b*	1 or 0	6-109	Enables/disables SINAD Meter Lower Limit.
PEAK?		6-109	Returns a SINAD Meter Peak reading in dB (3.0 to 40.0).
PH *b*	1 or 0	6-109	Enables/disables SINAD Meter Peak Hold Feature.
RCL *n*	1 to 9	6-109	Recalls SINAD Meter environment (routings and settings) stored in memory location *n*.
RESolution *n*	0.1 or 0.5	6-110	Sets SINAD Meter readout resolution in dB.
RESolution?		6-110	Returns current resolution setting in dB (0.1 or 0.5).
SELect:			
CMESsage		6-110	Same as **M_SINAD:SELect:CWEight** command.
CWEight		6-110	Selects C-Weight Filter.
LPASs *f*	00 to 30000	6-110	Selects Low-Pass Filter with cutoff frequency of *f* Hz.
STORe *n*	1 to 9	6-110	Stores current SINAD Meter environment (routings and settings) in memory location *n*.

COMMAND	RANGE	PAGE	DESCRIPTION
M_SINAD:			
UL:			
LEVel *n*	3.0 to 40.0	6-110	Sets SINAD Meter Upper Limit in dB.
STATe *b*	1 or 0	6-110	Enables/disables SINAD Meter Upper Limit.
M_SINAD?		6-111	Returns a SINAD Meter reading in dB (3.0 to 40.0).

AF LEVEL METER COMMAND

COMMAND	RANGE	PAGE	DESCRIPTION
M_VRMS?		6-123	Returns Voltage RMS reading of received AF Level (0.00 to 10.00).

MEASURE (GENERIC) COMMANDS

Generic commands use optional values, e to signify expected value and r to signify resolution. Expected values help determine Meter range. If r is omitted, the last resolution value is used.

COMMAND	RANGE	PAGE	DESCRIPTION
MEASure:			
AUDio? [*e,r*]		6-170	Returns AF Meter frequency counter reading in Hz (0.0 to 200000.0).
CURRent:			
AC? [*e*]		6-170	Returns DMM ac current reading in amps (0.00000 to 19.990).
DC? [*e*]		6-170	Returns DMM dc current reading in amps (0.00000 to 19.990).
FREQuency? [*e,r*]		6-170	Returns signal frequency reading in kHz (250.0 to 2010000.0 kHz).
MIC?		6-170	Returns 0 if receiving MIC/ACC Input; 1 otherwise.
PHASe? [*e*]		6-170	Returns Phase Meter reading in radians (0.00 to 10.00).
POWer? [*e*]		6-170	Returns Power Meter reading in mW (0.0 to 100000.0).
RESistance? [*e*]		6-170	Returns DMM resistance reading in kΩ (0.0000 to 19990).
SINAD? [*r*]		6-170	Returns SINAD Meter reading in dB (3.0 to 40.0).
SQUelch?		6-170	Returns 1 if squelch broken; 0 otherwise.
TEMPerature:			
AMBient?		6-171	Returns ambient temperature in °C (0.00000 to 100.00000).
POWer?		6-171	Returns Power Termination temperature in °C (0.00000 to 100.00000).
VOLTage:			
AC? [*e*]		6-171	Returns DMM ac voltage reading in volts (0.0000 to 1000.00).
DC? [*e*]		6-171	Returns DMM dc voltage reading in volts (0.0000 to 1000.00).
SUPply? *n*	-15, 5 or 15	6-171	Returns voltage measurement of *n* power supply in volts (0.00 to 20.00).

MODULATION METER COMMANDS

See M_MOD:

FLASH MEMORY COMMANDS

COMMAND	RANGE	PAGE	DESCRIPTION
MMEMory:			
ATTribute:			
DELete "*f*"		6-166	Sets the File Attribute of the specified Flash Memory file to DELETE. *f* is Flash Memory file name.
HIDe "*f*"		6-166	Sets the File Attribute of the specified Flash Memory file to HIDE. *f* is Flash Memory file name.
INITialize "*f*"		6-166	Sets the File Attribute of the specified Flash Memory file to INIT. *f* is Flash Memory file name.
PACK "*f*"		6-166	Sets the File Attribute of the specified Flash Memory file to PACK. *f* is Flash Memory file name.
ATTribute? "*f*"		6-166	Returns the Attribute (DELETE, HIDE, INIT or PACK) of the specified Flash Memory. *f* is Flash Memory file name.

COMMAND	RANGE	PAGE	DESCRIPTION
MMEMory:			
CATalog:			
ENTRY? *n*	0 to 512	6-166	Returns file entry (file name, file type, file size) for given index. Returns $$$ if past end of directory or --- for deleted file. *n* is line number (index) in Flash Memory File Directory.
FREE?		6-166	Returns available file space, in bytes.
USED?		6-166	Returns file space used, in bytes.
CATalog?		6-166	Returns Flash Memory status. First number returned is memory space used in bytes. Second number returned is memory space available in bytes. Remainder data is returned in sets of 3 consisting of file name, file type and file size for each file stored in Flash Memory.
DELete "*f*"		6-167	Deletes Flash Memory *file*, but does not release memory space until Pack operation is done.
INITialize		6-167	Erases all files stored in Flash Memory.
INITialize?		6-167	Returns 1 if file system has been initialized; otherwise 0.
LOAD:			
CALibration "*f*"		6-167	Loads Calibration Data from Flash Memory into Test Set memory. *f* is Flash Memory file name.
DATA *v*,"*f*"		6-167	Loads variable stored as *f* into Test Set memory with name *v*. *v* is name of variable; *f* is Flash Memory file name.
MACRo "*m*","*f*"		6-167	Loads macros and variables stored as file name *f* from Flash to Test Set memory, executing macro *m*. If *m* is *, designated macro (See **MMEMory:STORe:MACRo**) is executed. If *m* is macro name, that macro is executed. If *m* is omitted (""), no macro is executed.
STATe *n*,"*f*"	0 to 9, flash file name	6-167	Loads Test Set State stored as *f* from Flash Memory into Auxiliary Functions "Store Parameters Menu" as entry *n*. *n* is number of stored state of Test Set (*n* = 0 loads current state).
TRACe:			
ANLZ *n*,"*f*"	0 to 9, flash file name	6-167	Loads Spectrum Analyzer trace stored as *f* into Spectrum Analyzer "Store Parameters Menu" as entry *n*. *n* is number of stored trace (*n* = 0 loads live trace).
SCOPe *n*,"*f*"	0 to 9, flash file name	6-167	Loads Oscilloscope trace stored as *f* into Oscilloscope "Store Parameters Menu" as entry *n*. *n* is number of stored trace (*n* = 0 loads live trace).
PACK		6-168	Packs Flash Memory and frees memory space from deleted files.
STORe:			
CALibration "*f*"		6-168	Stores Test Set Calibration Data into Flash Memory. *f* is Flash Memory file name.
DATA *v*,"*f*"		6-168	Stores variable *v* into Flash Memory as *f*. *v* is name of variable; *f* is Flash Memory file name.
MACRo "*m*","*f*"		6-168	Stores all Test Set macros and variables (except free variables) in Flash Memory as *f* with macro *m* specified as designated macro. *m* is name of designated macro; *f* is Flash Memory file name.
STATe *n*,"*f*"	0 to 9, flash file name	6-168	Stores entry *n* of Auxiliary Functions "Store Parameters Menu" as *f* in Flash Memory. (*n* = 0 stores current state.) *n* is number of stored state of Test Set.
TRACe:			
ANLZ *n*,"*f*"	0 to 9, flash file name	6-168	Stores entry *n* (stored trace) of Spectrum Analyzer "Store Parameters Menu" as *f* in Flash Memory. (*n* = 0 stores live trace.) *n* is number of stored trace.
SCOPe *n*,"*f*"	0 to 9, flash file name	6-168	Stores entry *n* (stored trace) of Oscilloscope "Store Parameters Menu" as *f* in Flash Memory. (*n* = 0 stores live trace.) *n* is number of stored trace.
TYPE? "*f*"		6-168	Returns file type. *f* is Flash Memory file name.

COMMAND	RANGE	PAGE	DESCRIPTION
OSCILLOSCOPE COMMANDS			
See Scope Commands			
PHASE METER COMMANDS			
See M_PM:			
PHASE METER (RMS) COMMANDS			
See M_PMRMS:			
POWER METER COMMANDS			
See M_PWR:			
PROGRAM COMMANDS			
PROGram:			
STARTup:			
DELETE		6-164	Deletes power up designation of power up macro. After command is executed, no power up macro is available until a **PROGram:STARTup:NAME** command is executed. This command does not delete the macro itself.
NAME "*name*"		6-164	Selects macro *name* to execute at power up. Startup macro executes after POWER Switch is pressed and automatic self test is performed.
NAME?		6-164	Returns name of current power up macro designated by a **PROGram:STARTup:NAME** command.
PRESS TO TALK COMMAND			
See Function Generator Commands			
RECEIVER COMMANDS			
Queries for received data, return -1 if data is not available or has already been read.			
RECeiver:			
AGC:			
AUTO		6-19	Sets Automatic Gain Control to automatic setting.
MANual *n*	0 to 255	6-19	Sets Automatic Gain Control to manual setting and sets level to *n*.
USER:*xxx*	MEASure, SPeech, DATA, HIGH, TYPE1, TYPE2 or TYPE3	6-19	Sets Automatic Gain Control to User.
CHANnel *n*	1 to 2047, depending on format	6-19	Sets Receive Frequency to cellular channel *n* in the format selected using the **REC:CHAN:FORM** command.
CHANnel:			
BAND?		6-21	Returns band for Receiver Channel Format. Returns one of the following bands for the associated channel format: (NADC) U8, U4 or HY; (ETACS) NOT AVAILABLE; (NAMPS) LOWER, MIDDLE or UPPER.
FORMat:			
AMPS:			
FORward		6-19	Selects AMPS (NADC-U8) Forward channels.
REVerse		6-19	Selects AMPS (NADC-U8) Reverse channels.
ETACS:			
FORward		6-19	Selects ETACS Forward channels.
REVerse		6-19	Selects ETACS Reverse channels.
NADC:			
BAND:*xx*	U8, U4 or HYper	6-20	Selects NADC band.
FORward		6-20	Selects NADC Forward. Utilizes current setting of NADC band.
REVerse		6-20	Selects NADC Reverse. Utilizes current setting of NADC band.

COMMAND	RANGE	PAGE	DESCRIPTION
RECeiver:			
CHANnel:			
FORMat:			
NAMPS:			
BAND:*x*	Lower, Middle or Upper	6-20	Selects NAMPS Narrow Analog channel designator.
FORward		6-20	Selects NAMPS Forward channels.
REVerse		6-20	Selects NAMPS Reverse channels.
NT400:			
FORward		6-20	Selects NT400 Forward channels.
REVerse		6-20	Selects NT400 Reverse channels.
FORMat?		6-20	Returns Receiver Channel Format (NADC:FORWARD, NADC:REVERSE, ETACS:FORWARD, ETACS:REVERSE, NAMPS:FORWARD or NAMPS:REVERSE).
DCS:			
INVert?		6-21	Returns three octal digits in inverted mode, decoded from Digital Coded Squelch (DCS) code. Returns -1 if none available or if invalid for inverted DCS.
NORMal?		6-21	Returns three octal digits decoded from the DCS code. Returns -1 if none available or if invalid for normal DCS.
STATe *b*	1 or 0	6-21	Enables/disables DCS decoding.
DECode *type*	DTMF, TONE or DIGital	6-21	Sets Receiver decoding *type*.
DEVRms		6-21	Displays Deviation Meter (RMS) reading on Receiver Operation Screen when followed by **SCREEN:RECeiver** command.
DIGital *type*	DCS, DCSINV, POCSAG, DSAT or DST	6-21	Sets digital *type*.
DISTortion		6-22	Displays Distortion Meter reading on Receiver Operation Screen when followed by **SCREEN:RECeiver** command.
DMM		6-22	Displays Digital Multimeter reading on Receiver Operation Screen when followed by **SCREEN:RECeiver** command.
DSAT:			
STATe *b*	1 or 0	6-22	Enables/disables DSAT Decoding Function.
DSAT?		6-22	Returns DSAT reading. Returns -1 if none available or invalid for normal transmission.
DST:			
STATe *b*	1 or 0	6-22	Enables/disables DST Decoding Function.
DST?		6-22	Returns DST reading. Returns -1 if none available or invalid for normal transmission.
DTMF:			
STATe *b*		6-22	Enables/disables DTMF Decoding Function.
DTMF?		6-22	Returns string of decoded digits or -1 if nothing decoded.
FIND:			
FREQuency?		6-23	Returns frequency of first signal with amplitude larger than Find reference level. Returns 0 if no signal is found.
REFerence *n*	-110 to -5	6-23	Sets Find reference level in dBm.
REFerence?		6-23	Returns Find reference level in dBm.
FMZ *n*	1 to 4	6-23	Zeros FM Deviation Meter. Displays Receiver Operation Screen and selects FM*n* Modulation.
FREQuency *n* [*units*]	250000.0 to 2010000000.0 [Hz] 250.0 to 2010000.0 [kHz] 0.2500 to 2010.0 [MHz] *units* - HZ, KHZ or MHz (default units are KHZ)	6-23	Sets Receiver Frequency.
FREQuency?		6-23	Returns Receiver Frequency in kHz (250.0 to 2010000.0).

COMMAND	RANGE	PAGE	DESCRIPTION
RECeiver:			
INPut:			
ANTenna		6-24	Selects ANTENNA IN Connector as Receiver Input Source. Displays Signal Strength Meter reading on Receiver Operation Screen.
ATTenuation *n*	0, 5, 10, 15, 20, 25, 30	6-24	Sets Receiver Input Attenuation in dB.
ATTenuation:			
LNA		6-24	Sets Receiver Input Attenuation to LNA.
LNA?		6-24	Returns current state of Low Noise Amplifier. Returns 1 if LNA is selected; 0 otherwise.
ATTenuation?		6-24	Returns current value of Input Attenuation.
TR		6-24	Selects T/R Connector as Receiver Input Source. Displays Power Meter reading on the Receiver Operation Screen.
MODE *type*	DIRect, CHANnel, SCAN, LIST, FLScan	6-25	Selects Receiver Mode.
MODE?		6-25	Returns current mode (DIRECT, CHANNEL, FREQUENCY SCAN, FREQUENCY LIST or FREQUENCY LIST SCAN).
MODMeter		6-25	Displays Modulation Meter reading on Receiver Operation Screen when followed by **SCREEN:RECeiver** command and AM is selected modulation.
MODulation:			
AM*n*	1 or 2	6-25	Selects Amplitude Modulation.
BFO		6-25	Selects Beat Frequency Oscillation.
FM*n*	1, 2, 3 or 4	6-25	Selects Frequency Modulation.
PM		6-26	Selects Phase Modulation.
USER:			
FILTer *f*	3, 30 or 300	6-26	Sets User Defined Modulation IF Filter in kHz.
FILTer?		6-26	Returns current setting of User Defined Modulation IF Filter in kHz.
MODulation:*type*	FM, AM, BFO, PM or DATA [FM DATA]	6-26	Selects User Defined Modulation.
MODulation?		6-26	Returns User Defined Modulation type.
POST:			
APASs		6-26	Selects All Pass Post Detection Filter for User Defined Modulation.
BPASs *fl,fh*	0.5 to 20, 0.1 to 30	6-26	Selects Bandpass Post Detection Filter for User Defined Modulation, and sets lower cutoff frequency (*fl*) and Higher cutoff frequency (*fh*) in kHz.
BPASs:			
HIGH *f*	0.5 to 20.0	6-26	Sets Lower cutoff frequency of Band-Pass Post Detection Filter in kHz.
HIGH?		6-26	Returns Lower cutoff frequency of Band-Pass Post Detection Filter in kHz.
LOW *f*	0.1 to 30.0	6-27	Sets Upper cutoff frequency of Band-Pass Post Detection Filter in kHz.
LOW?		6-27	Returns Upper cutoff frequency of Band-Pass Post Detection Filter in kHz.
CMESsage		6-27	Same as **RECeiver:MODulation:USER:POST:CWEight** command.
CWEight		6-27	Selects C-Weighted Post Detection Filter (C-message noise weighting curve response) for User Defined Modulation.
CWT		6-27	Same as **RECeiver:MODulation:USER:POST:CWEight** command.
HPASs *fl*	0.5 to 20	6-27	Selects High-Pass Post Detection Filter for User Defined Modulation, and specifies cutoff frequency in kHz.
HPASs?		6-27	Returns cutoff frequency for User Defined Modulation High-Pass Post Detection Filter.

COMMAND	RANGE	PAGE	DESCRIPTION
RECeiver:			
MODulation:			
USER:			
POST:			
LPASs *fh*	0.1 to 30	6-28	Selects Low-Pass Post Detection Filter for User Defined Modulation, and specifies cutoff frequency in kHz.
LPASs?		6-28	Returns cutoff frequency for User Defined Modulation Low-Pass Post Detection Filter.
POST?		6-28	Returns current setting of User Defined Post Detection Filter (APAS, LPAS, HPAS, BPAS or CWE).
MODulation?		6-28	Returns Receiver Modulation (FM, AM, BFO, PM, DATA or USER).
OUTput:			
AUDio *b*	1 or 0	6-28	Connects/disconnects Demodulated Audio to AUDIO OUT Connector.
DEMOD *b*	1 or 0	6-28	Connects/disconnects Demodulated Audio to DEMOD OUT Connector.
SPEAKer *b*	1 or 0	6-28	Connects/disconnects Demodulated Audio to Speaker.
PMRms		6-29	Displays Phase Meter (RMS) reading when followed by **SCREEN:RECeiver** command.
POCSAG:			
CAPcode?		6-29	Returns received capcode or -1 if a capcode is not received.
MESSage?		6-29	Returns message from decoded POCSAG signal or -1 if none available.
RATe *b*	high = 1, low = 0	6-29	Sets POCSAG rate.
RATe?		6-29	Returns 1 if POCSAG rate is high, 0 if rate is low.
STATe *b*	1 or 0	6-29	Enables/disables POCSAG decoding.
TYPE?		6-29	Returns POCSAG Function Type (TONE 1 BEEP, TONE 2 BEEPS, TONE 3 BEEPS, TONE 4 BEEPS, NUMERIC, ALPHANUMERIC, NO MESSAGE).
RCL *n*	1 to 9	6-29	Recalls Receiver environment (routings and settings) stored in memory location *n*.
SCAN:			
ABORt		6-30	Stops frequency scan or frequency list scan depending on present mode.
CONTinue		6-30	Starts frequency scan or frequency list scan depending on present mode.
FREQList:			
PAUSe *n*	0.0 to 99.9	6-31	Frequency List Scan Pause Time. Specifies time period in seconds for Receiver to sit on a frequency if squelch is broken.
PAUSe?		6-31	Returns current value of Frequency List Scan Pause Time.
RATe *n*	0.02 to 99.99	6-31	Frequency List Scan Rate. Specifies time period in seconds for Receiver to sit on a frequency unless squelch is broken.
RATe?		6-31	Returns current value of Frequency List Scan Rate.
SQUelch *b*	1 or 0	6-31	Enables/disables Squelch.
SQUelch?		6-31	Returns current state of Squelch.
FREQuency?		6-30	Returns frequency currently being scanned in kHz.
INCrement *f*	0.0 to 2010000.0	6-30	(Frequency Scan mode only) Sets Receiver Scan increment in kHz.
PAUSe *t*	0.0 to 99.9	6-30	(Frequency Scan mode only) Sets Receiver Pause rate in sec. Receiver Scan pause time is length of time Scan Function pauses at frequency with broken squelch. Scan Function stops permanently on squelch broken frequency if pause set to 0.0.
PAUSe?		6-30	(Frequency Scan mode only) Returns 1 if scanning is paused (stopped); 0 if currently scanning.
RATe *t*	0.00 to 99.99	6-30	Sets Receiver Scan rate in sec. Receiver Scan rate is time each frequency is scanned with squelch unbroken.
STARt *f*	250.0 to 2010000.0	6-30	Sets Receiver Scan starting frequency in kHz.
STOP *f*	250.0 to 2010000.0	6-31	Sets Receiver Scan stopping frequency in kHz.

COMMAND	RANGE	PAGE	DESCRIPTION
RECeiver:			
SINAD		6-32	Displays SINAD Meter reading when followed by a **SCREEN:RECeiver** command.
SQUelch *n*	0.0 to 1.0	6-32	Sets squelch to *n*.
SQUelch?		6-32	Returns squelch level.
STORe *n*	1 to 9	6-32	Stores current Receiver environment (routings and settings) in memory location *n*.
TONE:			
DURation? "*id*"	0 to 9 and A to T	6-32	Returns Duration in ms (20 to 9999) of specified User Defined Audio Tone. *id* is character associated with the defined tone.
FREQuency? "*id*"	0 to 9 and A to T	6-32	Returns frequency setting in Hz of specified User Defined Audio Tone *id*.
INPut *n*	0 - Demod Audio, 1 - SINAD/BER, 2 - Ext Mod	6-32	Selects decode signal input.
STATe *b*	1 or 0	6-33	Enables/disables Audio Tone decoding.
TYPE *xxx*	CCIR, EEA, EIA (U.S.), ZVEI, DDZVEI, DZVEI, NATEL, EURO, TONE56, CCIRH, CCIRH4, USER	6-33	Selects Audio Tone to be decoded.
USER:			
DEFine "*id*",*f*,*d*	0 to 9 and A to T; 0.0 to 9999.9; 20 to 9999	6-32	Defines a User Defined Audio Tone and duration to be received; where *id* is character associated with defined tone, *f* is frequency in Hz of defined tone and *d* is duration of defined tone in ms.
TONE?		6-33	Returns decoded Audio Tone sequence or -1 if not available.
VOLume *n*	0.0 to 1.0	6-33	Sets volume to *n*.
VOLume:			
AUTO *b*	1 or 0	6-33	Enables/disables Automatic Volume Control.
AUTO?		6-33	Returns Automatic Volume Control state. 1 is returned if enabled, 0 if disabled.
VOLume?		6-33	Returns the volume level.

RF GENERATOR COMMANDS

See Generator Commands

SCOPE COMMANDS

COMMAND	RANGE	PAGE	DESCRIPTION
SCOPe:			
ARM		6-66	Arms Oscilloscope. Sets Oscilloscope for one sweep. Command ignored unless Trigger is set to One Shot (see **SCOPe:TRIGger** commands).
AVErage *n*	1 to 100 (default is 100)	6-66	Selects Oscilloscope Average Mode using *n* samples.
COMPare *n*	1 to 9	6-66	Selects Compare Mode for Oscilloscope. Trace stored at *n* memory location is compared to the current Live Trace.
COUPling *type*	AC, DC or GROund	6-66	Selects external coupling type.
FULL		6-66	Selects full size Oscilloscope display for RF Generator, Receiver and Duplex Operation Screens.
HORIZontal *n*	-12 to 12	6-66	Sets Horizontal Time Offset to *n* major divisions. (-12 to -1 are major divisions before the trigger; 1 to 12 are major divisions after the trigger.)

COMMAND	RANGE	PAGE	DESCRIPTION
SCOPe:			
INPut:			
FILTer:			
CWEight:			
STATe *b*	1 or 0	6-66	Enables/disables Internal C-Weight filter.
HPASs:			
FREQuency *f*	0.2 to 100	6-66	Sets Internal High-Pass frequency in kHz.
STATe *b*	1 or 0	6-66	Enables/disables Internal High-Pass filter.
LPASs:			
FREQuency *f*	0.2 to 50	6-67	Sets Internal Low-Pass frequency in kHz.
STATe *b*	1 or 0	6-67	Enables/disables Internal Low-Pass filter.
NOTch:			
FREQuency *f*	0.5 to 1.5	6-67	Sets Notch center frequency in kHz.
STATe *b*	1 or 0	6-67	Enables/disables Internal Notch Filter.
INTernal *type*	IF (Rcvr IF), DEMOD (Demod Audio), POWer (RF Pwr Lvl), SINAD (SINAD/BER), FUNCtion (Func Gen), XAUDIO (Ext Mod)	6-67	Selects Internal Oscilloscope Input.
LEVel *n*	0 to 255	6-67	Specifies Trigger level. (0 corresponds to bottom of Oscilloscope Display; 255 corresponding to top.)
LIVe *b*	1 = **SCREEN: SCOPE** command is performed; 0 = nothing happens.	6-67	Selects Live Trace Mode for Oscilloscope.
MARKer:			
AOFF		6-68	Disables both Markers.
DELTA:			
AMPLitude?		6-68	Returns voltage difference of the Trace Marker 1 and Trace Marker 2 crossings in volts. Valid only for Oscilloscope Inputs AC, DC and GND.
POINt?		6-68	Returns the difference of Marker positions in graticules with 100 graticules equal to the Oscilloscope display width.
TIME?		6-68	Returns the difference of the two Marker positions in ms.
TRACK *b*	1 or 0	6-68	Enables/disables Marker Tracking.
MARKER1:			
AMPLitude?		6-68	Returns voltage of live Trace at Marker 1 crossing in volts. Valid only for Oscilloscope Inputs AC, DC and GND.
POINt *n*	1 to 100	6-68	Sets Marker 1 position in graticules (100 graticules is equal to Oscilloscope display width).
POINt?		6-68	Returns Marker 1 position in graticules.
STATe *b*	1 or 0	6-69	Enables/disables Marker 1.
STATe?		6-69	Returns 1 if Marker 1 is active, 0 if Marker 1 is not active.
TIME?		6-69	Returns Marker 1 position in ms from left edge of Oscilloscope display.
MARKER2:			
AMPLitude?		6-68	Returns voltage of live Trace at Marker 2 crossing in volts. Valid only for Oscilloscope Inputs AC, DC and GND.
POINt *n*	1 to 100	6-68	Sets Marker 2 position in graticules (100 graticules is equal to Oscilloscope display width).
POINt?		6-68	Returns Marker 2 position in graticules.
STATe *b*	1 or 0	6-69	Enables/disables Marker 2.
STATe?		6-69	Returns 1 if Marker 2 is active, 0 if Marker 2 is not active.
TIME?		6-69	Returns Marker 2 position in ms from left edge of Oscilloscope display.

COMMAND	RANGE	PAGE	DESCRIPTION
SCOPe:			
PLOT:			
GRID		6-70	Draws Oscilloscope grid on attached plotter.
TRACE		6-70	Draws Oscilloscope trace on attached plotter.
UNITS		6-70	Draws Oscilloscope units on attached plotter.
QTR		6-70	Selects 1/4 size Oscilloscope display for RF Generator, Receiver and Duplex Operation Screens.
RCL *n*	1 to 9	6-70	Recalls Oscilloscope Trace and environment (routings and settings) stored in memory location *n*.
SCALe *n*	(AC, DC or GND Input) 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000, 20000, 50000, 100000 (mV/div). (Demod Audio Input with FM) 2, 4, 10, 20 (kHz/div) (Func Gen or Ext Mod Input) 500, 1000, 2500 (mV/div)	6-70	Sets Oscilloscope scale.
SCALe?		6-71	Returns Oscilloscope scale in mV/div if input is AC, DC, GND, SINAD/BER, Func Gen or Ext Mod. Returns Oscilloscope scale in kHz/div if input is Demod Audio. Returns Oscilloscope scale in W if input is RF Pwr Lvl.
SOURce EXTernal		6-71	Routes Oscilloscope Input from the SCOPE IN Connector. Selects AC Ext for Oscilloscope Input.
SOURce INTernal		6-71	Disconnects Oscilloscope Input from the SCOPE IN Connector. Selects Demod Audio for Oscilloscope Input.
STATe *b*	1 or 0	6-71	Enables/disables Oscilloscope display in RF Generator, Receiver and Duplex Operation Screens.
STORe *n*	1 to 9	6-71	Stores current Oscilloscope Trace and environment (routings and settings) in memory location *n*.
SWEep *n*	1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000, 20000, 50000, 100000	6-71	Sets Oscilloscope sweep rate in µs/div.
SWEep *n* MS	1, 2, 5, 10, 20, 50, 100	6-71	Sets Oscilloscope sweep rate in ms/div.
SWEep *n* US	1, 2, 5, 10, 20, 50, 100, 200, 500	6-72	Sets Oscilloscope sweep rate in µs/div.
SWEep?		6-72	Returns Oscilloscope sweep rate in µs/div.

COMMAND	RANGE	PAGE	DESCRIPTION

For the following **SCOPe:TRACE** commands, the Oscilloscope display is divided into 400 positions horizontally (0 signifying the left edge of the display, 399 signifying the right edge of the display) and 255 values vertically (0 signifying the bottom of the display, 255 signifying the top of the display).

COMMAND	RANGE	PAGE	DESCRIPTION
SCOPe:			
TRACE:			
DATA *n,offset,value,...,value*	*n* = 1 to 9, *offset* = 0 to 399 value = 0 to 255	6-73	Replaces points of stored Trace *n* with specified vertical values (*value*) starting with horizontal offset (*offset*). Intended for remote GPIB or RS-232 use only; command and data are sent via GPIB or RS-232 from an external application, etc.
DATA? [[[*n*],*offset*],*points*]	*n* = 0 to 9, 0 = Live Trace (0 default). *offset* = 0 to 399 (0 default). *points* = 1 to 400 (400 default)	6-73	Returns the vertical values for each of the *points* specified, starting with *offset* (horizontal offset) of trace *n*. Intended for remote GPIB or RS-232 use only; data is sent out GPIB or RS-232 for use by an external application, etc.
GET *name,n*	*n* = 0 to 9, 0 = Live Trace.	6-73	Assigns vertical values of trace *n* (in graticules) to each element of declared array *name*. First element of array corresponds with first point of trace *n*. If array is less that 400 values in length, the remaining portion of trace is left unassigned to an array. See **SCOPe:TRACE:PUT**.
GET? *n,offset*	*n* = 0 to 9, 0 = Live Trace. *offset* = 0 to 399.	6-73	Returns vertical value of a point in Trace *n* located at *offset* (horizontal position from the left edge of display).
MAX? [[[*n*],*offset*],*points*]	*n* = 0 to 9, 0 = Live Trace (0 default). *offset* = 0 to 399 (0 default). *points* = 1 to 400 (400 default).	6-73	Returns the maximum point of Trace *n* within specified number of *points* starting with given *offset*. Result is returned in x,y format with x being the number of positions from the left edge and y being the number of values from the bottom. Trace *n* can be the live Trace or a Trace stored in memory. Intended for remote GPIB or RS-232 use only; data is sent out GPIB or RS-232 for use by an external application, etc.
MIN? [[[*n*],*offset*],*points*]	*n* = 0 to 9, 0 = Live Trace (0 default). *offset* = 0 to 399 (0 default). *points* = 1 to 400 (400 default).	6-74	Returns the minimum point of Trace *n* within specified number of *points* starting with the given *offset*. Result is returned in x,y format with x being the number of positions from the left edge and y being the number of values from the bottom. Trace *n* can be the Live Trace or a Trace stored in memory. Intended for remote GPIB or RS-232 use only; data is sent via GPIB or RS-232 from an external application, etc.
PUT *name,n*	*n* = 1 to 9	6-74	Assigns values of an array to trace *n*. *name* is array name. *n* signifies stored trace. Each element value represent a vertical value for associated horizontal positions. If array is less than 400 values in length (stored trace length), the array values are assigned to the trace the length of the array starting from horizontal position 0. The remaining portion of the stored trace is left intact. See **SCOPe:TRACE:GET**.

COMMAND	RANGE	PAGE	DESCRIPTION
SCOPe:			
TRIGger:			
AUTO		6-74	Sets Trigger to Auto Sweep.
IMMediate		6-74	Triggers Oscilloscope as soon as command is interpreted or sequenced in a macro.
NORM		6-74	Sets Trigger to Normal Sweep.
ONE		6-75	Sets Trigger to One Shot. Causes oscilloscope to sweep once when trigger is received and after setting arm function (**SCOPe:ARM** command).
SOURce *type*	EXTernal (for external trigger, AC or DC Input only), INTernal (for internal trigger, AC or DC Input only) or BUS (triggered by *TRG or IEEE-488.1 GET [Get Execute Trigger] command).	6-75	Selects source to trigger on.
VERTical *n*	0 to 255	6-75	Sets Vertical Offset voltage to *n* graticules. Setting is not linear. For AC, DC or GND, middle setting is approximately 150 to 170. For other Inputs, middle setting is approximately 160 to 180.

SCREEN COMMANDS

COMMAND	RANGE	PAGE	DESCRIPTION
SCREEN:			
AF		6-2	Displays Audio Frequency Meter Operation Screen.
AFLVL		6-2	Displays AF Level (RMS) Meter Operation Screen.
ANLZ		6-2	Displays Spectrum Analyzer Operation Screen.
BER		6-2	Displays Bit Error Rate Meter Operation Screen.
BUILD		6-2	Displays blank blue screen with green border.
CELL		6-2, 6-124	Displays FOCC Forward Control Channel Screen of the AMPS/NAMPS Cell Site Monitor.
DEViation		6-2	Displays Deviation Meter (Peak) Operation Screen.
DISTortion		6-3	Displays Distortion Meter Operation Screen.
DMM		6-3	Displays Digital Multimeter Operation Screen.
DRMS		6-3	Displays Deviation Meter (RMS) Operation Screen.
DUPlex		6-3	Displays Duplex Operation Screen.
DUPRX		6-3	Displays Duplex Receiver Operation Screen.
DUPTX		6-3	Displays Duplex Transmitter Operation Screen.
FREQuency		6-3	Displays Frequency Error Meter Operation Screen.
FUNC		6-3	Displays AF Generator Operation Screen.
GENCELLSETUP		6-3	Displays Cellular Settings Menu of AMPS/NAMPS Cellular Simulation.
GENCELLular		6-3, 6-124	Displays Main Menu of AMPS/NAMPS Cellular Simulation.
GENerator		6-3	Displays RF Generator Operation Screen.
GENFOCC		6-3, 6-124	Displays Forward Control Channel Screen of the AMPS/ NAMPS Cellular Simulation.
GENFVC		6-3, 6-124	Displays the Forward Voice Channel of AMPS/NAMPS Cellular Simulation.
GENFVCNAMPS		6-4	Displays NAMPS FVC Screen of AMPS/NAMPS Cellular Simulation.
GENGLACT		6-4	Displays Global Action Screen for FOCC of AMPS/NAMPS Cellular Simulation.
GENMSCM		6-4	Displays the Mobile Station Control Message for FOCC of AMPS/NAMPS Cellular Simulation.

COMMAND	RANGE	PAGE	DESCRIPTION
SCREEN:			
GENRECC		6-4, 6-124	Displays Reverse Control Channel Simulation Screen of AMPS/NAMPS Cellular Simulation.
GENRVC		6-4, 6-124	Displays Reverse Voice Channel Simulation Screen of AMPS/NAMPS Cellular Simulation.
GENRVCNAMPS		6-4	Displays NAMPS RVC Screen of AMPS/NAMPS Cellular Simulation.
MODulation		6-4	Displays Modulation Meter Operation Screen.
PM		6-4	Displays Phase Meter Operation Screen.
PMRMS		6-4	Displays Phase Meter (RMS) Operation Screen.
POWer		6-4	Displays Power Meter Operation Screen.
RECeiver		6-4	Displays Receiver Operation Screen.
SCOPe		6-4	Displays Oscilloscope Operation Screen.
SIGnal		6-5	Displays Signal Strength Meter Operation Screen.
SINAD		6-5	Displays SINAD Meter Operation Screen.
USER		6-5	Displays blank User Screen without changing routing.

SETUP COMMANDS

COMMAND	RANGE	PAGE	DESCRIPTION
SETUP:			
AF		6-1	Configures Test Set routing for AF Meter Operation.
AFLVL		6-1	Configures Test Set routing for AF Level (RMS) Meter Operation.
ANLZ		6-1	Configures Test Set routing for Spectrum Analyzer Operation.
DISTortion		6-1	Configures Test Set routing for Distortion Meter Operation.
DUPlex		6-1	Configures Test Set routing for Duplex Operation.
DUPRX		6-1	Configures Test Set routing for Duplex Receiver Operation.
DUPTX		6-1	Configures Test Set routing for Duplex Transmitter Operation.
FUNC		6-1	Configures Test Set routing for AF Generator Operation.
GENerator		6-1	Configures Test Set routing for RF Generator Operation.
MONitor		6-2	Configures Test Set routing for Generator/Monitor Operation.
RECeiver		6-2	Configures Test Set routing for Receiver Operation.
SCOPe		6-2	Configures Test Set routing for Oscilloscope Operation.
SINAD		6-2	Configures Test Set routing for SINAD Meter Operation.

SIGNAL STRENGTH METER COMMANDS

See M_SIG:

SINAD METER COMMANDS

See M_SINAD:

SPECTRUM ANALYZER) COMMANDS

See Analyzer Commands.

SECTION 6 - HOST SPECIFIC TMAC COMMANDS

6-1 GENERAL

This section lists HOST specific commands by Operation Mode. Many commands operate only when the Test Set routing is configured for the operation mode of that command. **SETUP** commands establish routings for selected operation modes (see 6-2). **SCREEN** commands establish routings and display the operation screen for the selected operation mode (see 6-3).

Commands affect the Test Set when executed; however changes appear on the Operation Screens only when the Screen is renewed using a **SCREEN** command.

The short form of the command is shown in uppercase letters and the long form is finished in lower case. When entering commands, it is not necessary to use a particular letter case. The TMAC compiler/interpreter is not case sensitive. Brackets ([]) indicate optional command items.

6-2 SETUP COMMANDS

SETUP commands establish the routings for various Operation Modes without displaying the Operation Screens.

SETUP:

AF
[SETUP:AF]
Configures Test Set routing for AF Meter Operation.

AFLVL
[SETUP:AFLVL]
Configures Test Set routing for AF Level (RMS) Meter Operation.

ANLZ
[SETUP:ANLZ]
Configures Test Set routing for Spectrum Analyzer Operation.

DISTortion
[SETUP:DISTortion]
Configures Test Set routing for Distortion Meter Operation.

DUPlex
[SETUP:DUPlex]
Configures Test Set routing for Duplex Operation.

DUPRX
[SETUP:DUPRX]
Configures Test Set routing for Duplex Receiver Operation.

DUPTX
[SETUP:DUPTX]
Configures Test Set routing for Duplex Transmitter Operation.

FUNC
[SETUP:FUNC]
Configures Test Set routing for AF Generator Operation.

GENerator
[SETUP:GENerator]
Configures Test Set routing for RF Generator Operation.

SETUP:

MONitor
[SETUP:MONitor]
Configures Test Set routing for Generator/Monitor Operation.

RECeiver
[SETUP:RECeiver]
Configures Test Set routing for Receiver Operation.

SCOPe
[SETUP:SCOPe]
Configures Test Set routing for Oscilloscope Operation.

SINAD
[SETUP:SINAD]
Configures Test Set routing for SINAD Meter Operation.

6-3 SCREEN COMMANDS

SCREEN commands renew and display Mode Operation Screen. Many commands operate correctly only when the applicable Mode Operation Screen is displayed.

SCREEN:

AF
[SCREEN:AF]
Displays Audio Frequency Meter Operation Screen.

AFLVL
[SCREEN:AFLVL]
Displays AF Level (RMS) Meter Operation Screen.

ANLZ
[SCREEN:ANLZ]
Displays Spectrum Analyzer Operation Screen.

BER
[SCREEN:BER]
Displays Bit Error Rate Meter Operation Screen.

BUILD
[SCREEN:BUILD]
Displays blank blue screen with green border.

CELL
[SCREEN:CELL]
Displays Forward Control Channel Screen of the AMPS/NAMPS Cell Site Monitor.

DEViation
[SCREEN:DEViation]
Displays Deviation Meter (Peak) Operation Screen.

SCREEN:

DISTortion
[SCREEN:DISTortion]

Displays Distortion Meter Operation Screen.

DMM
[SCREEN:DMM]

Displays Digital Multimeter Operation Screen.

DRMS
[SCREEN:DRMS]

Displays Deviation Meter (RMS) Operation Screen.

DUPlex
[SCREEN:DUPlex]

Displays Duplex Operation Screen.

DUPRX
[SCREEN:DUPRX]

Displays Duplex Receiver Operation Screen.

DUPTX
[SCREEN:DUPTX]

Displays Duplex Transmitter Operation Screen.

FREQuency
[SCREEN:FREQuency]

Displays Frequency Error Meter Operation Screen.

FUNC
[SCREEN:FUNC]

Displays AF Generator Operation Screen.

GENCELLSETUP
[SCREEN:GENCELLSETUP]

Displays the Cellular Settings Menu of the AMPS/NAMPS Cellular Simulation.

GENCELLular
[SCREEN:GENCELLular]

Displays Main Menu of the AMPS/NAMPS Cellular Simulation.

GENerator
[SCREEN:GENerator]

Displays RF Generator Operation Screen.

GENFOCC
[SCREEN:GENFOCC]

Displays the Forward Control Channel of the AMPS/NAMPS Cellular Simulation.

GENFVC
[SCREEN:GENFVC]

Displays the Forward Voice Channel of the AMPS/NAMPS Cellular Simulation.

SCREEN:

GENFVCNAMPS
[SCREEN:GENFVCNAMPS]

Displays the Narrow (NAMPS) Forward Voice Channel Screen of the AMPS/NAMPS Cellular Simulation.

GENGLACT
[SCREEN:GENGLACT]

Displays the Global Action Screen for the Forward Control Channel (FOCC) of the AMPS/NAMPS Cellular Simulation.

GENMSCM
[SCREEN:GENMSCM]

Displays the Mobile Station Control Message for the FOCC of the AMPS/NAMPS Cellular Simulation.

GENRECC
[SCREEN:GENRECC]

Displays Reverse Control Channel Simulation Screen of AMPS/NAMPS Cellular Simulation.

GENRVC
[SCREEN:GENRVC]

Displays Reverse Voice Channel Simulation Screen of the AMPS/NAMPS Cellular Simulation.

GENRVCNAMPS
[SCREEN:GENRVCNAMPS]

Displays the Narrow (NAMPS) Reverse Voice Channel Screen of the AMPS/NAMPS Cellular Simulation.

MODulation
[SCREEN:MODulation]

Displays Modulation Meter Operation Screen.

PM
[SCREEN:PM]

Displays Phase Meter Operation Screen.

PMRMS
[SCREEN:PMRMS]

Displays Phase Meter (RMS) Operation Screen.

POWer
[SCREEN:POWer]

Displays Power Meter Operation Screen.

RECeiver
[SCREEN:RECeiver]

Displays Receiver Operation Screen.

SCOPe
[SCREEN:SCOPe]

Displays Oscilloscope Operation Screen.

SCREEN:

SIGnal

[SCREEN:SIGnal]

Displays Signal Strength Meter Operation Screen.

SINAD

[SCREEN:SINAD]

Displays SINAD Meter Operation Screen.

USER

[SCREEN:USER]

Displays blank User Screen without changing routing. User Screen has no readings to update, decreasing run time when inside a macro. Following this command with other graphic, window and keypad commands creates customized menu screens (see 2-13).

6-4 INSTRUMENT COMMAND

The Instrument command allows the user to select a default "instrument" type (e.g., RF Generator, Receiver, Duplex, etc.). This command allows the user to issue TMAC commands without the "instrument" part (first level) of the command.

INSTrument:SELect *type*

[INSTrument:SELect type]

Selects default instrument; eliminates the necessity of the first part of the TMAC command for succeeding commands to that instrument. Enter one of the following for *type*:

GENerator	RECeiver	DUPlex	FGEN
SCOPe	ANLZ	M_AF	M_RF
M_PWR	M_DEV	M_MOD	M_DIST
M_SINAD	M_SIG	M_BER	M_DMM
M_PM	M_DRMS	M_PMRMS	

Example:

```
INST:REC                  // Selects Receiver as Default Instrument.
REC:FREQ?                 // Returns Receiver freq. setting.
FREQ?                     // Also, returns Receiver freq. setting.
GEN:DIST                  // This command is unaffected, performs
                          // normally.
FREQ?                     // Still returns Receiver freq. setting.
INST:GEN                  // Selects RF Generator as default.
FREQ?                     // Returns RF Generator freq. setting.
```

6-5 RF GENERATOR

6-5-1 RF GENERATOR COMMANDS

GENerator:

AF
[GENerator:AF]
Displays AF Level Meter on the RF Generator Operation Screen when followed by **SCREEN:GENerator** command.

CHANnel *n*
[GENerator:CHANnel n]
Sets Frequency to cellular channel *n* (1 to 2047, depending on format) in the format selected using the **GEN:CHAN:FORM** command. RF Generator Operation Screen displays selected cellular channel when RF Generator is in Channel Mode (**GEN:MODE CHAN** command).

CHANnel:

FORMat:

AMPS:

FORward
[GENerator:CHANnel:FORMat:AMPS:FORward]
Selects AMPS (NADC-U8) Forward channels.

REVerse
[GENerator:CHANnel:FORMat:AMPS:REVerse]
Selects AMPS (NADC-U8) Reverse channels.

ETACS:

FORward
[GENerator:CHANnel:FORMat:ETACS:FORward]
Selects ETACS Forward channels.

REVerse
[GENerator:CHANnel:FORMat:ETACS:REVerse]
Selects ETACS Reverse channels.

NAMPS:

FORward
[GENerator:CHANnel:FORMat:NAMPS:FORward]
Selects NAMPS Forward channels.

REVerse
[GENerator:CHANnel:FORMat:NAMPS:REVerse]
Selects NAMPS Reverse channels.

BAND:*x*
[GENerator:CHANnel:FORMat:NAMPS:BAND:x]
Selects NAMPS Narrow Analog channel designator. Range of *x* is Lower, Middle or Upper.

GENerator:

CHANnel:

FORMat:

NT400:

FORward
[GENerator:CHANnel:FORMat:NT400:FORward]
Selects NT400 Forward channels.

REVerse
[GENerator:CHANnel:FORMat:NT400:REVerse]
Selects NT400 Reverse channels.

NADC:

FORward
[GENerator:CHANnel:FORMat:NADC:FORward]
Selects NADC Forward. Utilizes current setting of NADC band.

REVerse
[GENerator:CHANnel:FORMat:NADC:REVerse]
Selects NADC Reverse. Utilizes current setting of NADC band.

BAND:*x*
[GENerator:CHANnel:FORMat:NADC:BAND:x]
Selects NADC band. Range of x is U8, U4 or HYper.

FORMat?
[GENerator:CHANnel:FORMat?]
Returns Generator Channel Format (NADC:FORWARD, NADC:REVERSE, ETACS:FORWARD, ETACS:REVERSE, NAMPS:FORWARD or NAMPS:REVERSE).

BAND?
[GENerator:CHANnel:BAND?]
Returns band for Generator Channel Format. Returns one of the following bands for the associated channel format:

FORMAT	BAND
NADC	U8, U4 or HY
ETACS	NOT AVAILABLE
NAMPS	LOWER, MIDDLE or UPPER

GENerator:

DCS:

INVert *nnn*
[GENerator:DCS:INVert nnn]
Generates the three digit octal Digital Coded Squelch (DCS) code in inverted mode. Range of *nnn* is 000 to 777.

NORMal *nnn*
[GENerator:DCS:NORMal nnn]
Generates the three digit octal DCS code. Range of *nnn* is 000 to 777. See 6-5-2G.

STOP
[GENerator:DCS:STOP]
Stops generating the continuous DCS code.

Example:
```
GEN:DCS:NORM 411     // Generates a continuous DCS code of 411.
DELAY 200            // Waits for 200 ms before next command.
GEN:DCS:STOP         // Stops generating the DCS code.
```

DIAL *"sequence"*
[GENerator:DIAL "sequence"]
Generates *sequence* with a maximum of 16 digits (0 through 9 allowed) as 2805 Pulse code. See 6-5-2I.

DIAL:

FREQuency *f*
[GENerator:DIAL:FREQuency f]
Sets 2805 Tone frequency to *f* Hz. Range of *f* is 0.0 to 40000.0.

FREQuency?
[GENerator:DIAL:FREQuency?]
Returns the current 2805 Tone frequency in Hz.

Example:
```
GEN:DIAL:FREQ 3000   // Sets 2805 Tone frequency to 3000 Hz.
GEN:DIAL:FREQ?       // Returns 2805 Tone frequency setting.
GEN:DIAL "5552424"   // Generates a 2805 Tone frequency message
                     // of 5552424.
```

GENerator:

DISTortion
[GENerator:DISTortion]
Displays Distortion Meter on the RF Generator Operation Screen after the screen is updated by a **SCREEN:GENerator** command.

DMM
[GENerator:DMM]
Displays Digital Multimeter on the RF Generator Operation Screen after the screen is updated by a **SCREEN:GENerator** command.

GENerator:

DSAT *n*
[GENerator:DSAT n]

Generates the Digital Supervisory Audio Tone (DSAT) code specified by *n*. Range of *n* is 0 to 6.

DSAT:STOP
[GENerator:DSAT:STOP]

Stops generating the continuous DSAT code.

DST *n*
[GENerator:DST n]

Generates the Digital Signal Tone (DST) code specified by *n*. Range of *n* is 0 to 6.

DST:STOP
[GENerator:DST:STOP]

Stops generating the continuous DST code.

DTMF *"sequence",mark,space*
[GENerator:DTMF "sequence",mark,space]

Generates a DTMF coded *sequence*. *sequence* has a maximum of 16 digits. Range of optional *mark* and *space* time is from 25 to 9999 ms. Default for *mark* time is 74 ms. Default for *space* time is 67 ms. See 6-5-2D.

FREQuency *n* [*units*]
[GENerator:FREQuency n [units]]

Sets RF Generator Frequency. Range of *n* is 250.0 kHz to 2010.0000 MHz. Select HZ, KHZ, MHZ for *units*. Default for optional *units* is KHZ.

FREQuency?
[GENerator:FREQuency?]

Returns RF Generator Frequency in kHz (250.0 to 2010000.0).

Example:
```
GEN:FREQ 100 MHz      // Sets RF Generator Frequency to 100 MHz.
GEN:FREQ?             // Queries the RF Generator Frequency.
                      // 100000 is returned (100000 kHz).
```

IMTS *"sequence"*
[GENerator:IMTS "sequence"]

Generates *sequence* as a DCS IMTS code. *sequence* has a maximum of 16 digits (0 through 9 allowed).

LEVel *n* [*units*]
[GENerator:LEVel n [units]]

Sets RF Generator Level. Range of *n* is -137.0 to 0.0 dBm or 0.031 µV to 0.224 V. Select DBm, V, MV (mV) or UV (µV) for *units*. *units* is optional with the default being the current units. Specifying *units* does not change unit selection of Test Set.

GENerator:

LEVel:DBm *n*
[GENerator:LEVel:DBm n]

Sets RF Generator Level in dBm. Range of *n* is -137.0 to 0.0.

LEVel:DBm?
[GENerator:LEVel:DBm?]

Returns RF Generator Level in dBm.

LEVel:UNIT
[GENerator:LEVel:UNIT]

Toggles the RF Generator level units between dBm and Volts.

LEVel:UNIT?
[GENerator:LEVel:UNIT?]

Returns the current units for the RF Generator Level. Returns DBM or V.

LEVel?
[GENerator:LEVel?]

Returns the RF Generator Level in the current units.

Example:
```
GEN:LEV -65 DB        // Sets RF Generator Level to -65 dBm.
GEN:LEV:UNIT?         // Queries the units for the RF Generator
                      // Level.  DBM is returned.
GEN:LEV?              // Queries the RF Generator Level.  -65
                      // is returned.
GEN:LEV -30           // Sets RF Generator Level to -30 dBm (units
                      // default to dBm, the current units).
```

MODE *type*
[GENerator:MODE type]

Selects the Generator Mode. Channel Mode displays cellular channel frequency according to **GEN:CHAN** commands.

MODE	*type*
Direct	DIRect
Channel	CHANnel
Frequency Scan	SCAN
Frequency List	LIST
Frequency List Scan	FLScan

MODE?
[GENerator:MODE?]

Returns current mode (DIRECT, CHANNEL, FREQUENCY SCAN, FREQUENCY LIST or FREQUENCY LIST SCAN).

MTS *"sequence"*
[GENerator:MTS "sequence"]

Generates *sequence* as a DCS MTS code. *sequence* has a maximum of 16 digits (0 through 9 allowed).

GENerator:

OUTput:

AUDio *b*
[GENerator:OUTput:AUDio b]
Routes AF Generator Output to the AUDIO OUT Connector if *b* is 1. Disconnects the AUDIO OUT Connector if *b* is 0.

DEMOD *b*
[GENerator:OUTput:DEMOD b]
Routes AF Generator Output to the DEMOD OUT Connector if *b* is 1. Disconnects the DEMOD OUT Connector if *b* is 0.

POCSAG:

ALPHA:

LOWer *capcode*
[GENerator:POCSAG:ALPHA:LOWer capcode]
Generates a lower case Alpha message for the *capcode* specified. Range of the *capcode* is 0 to 2097151.

NUMeric *capcode*
[GENerator:POCSAG:ALPHA:NUMeric capcode]
Generates an Alphanumeric message for the *capcode* specified. Range of the *capcode* is 0 to 2097151.

Example:
```
GEN:POCSAG:NUMeric 4000     // Generates an Alphanumeric message at a
                            // capcode of 4000 at the current rate.
```

SPECial *capcode*
[GENerator:POCSAG:ALPHA:SPECial capcode]
Generates an Alpha special message for the *capcode* specified. Range of the *capcode* is 0 to 2097151.

UPPer *capcode*
[GENerator:POCSAG:ALPHA:UPPer capcode]
Generates an upper case Alpha message for the *capcode* specified. Range of the *capcode* is 0 to 2097151. See 6-5-2H.

BEEP *n, capcode*
[GENerator:POCSAG:BEEP n, capcode]
Generates *n* Tone Beep POCSAG message for *capcode* specified. Select 1, 2, 3 or 4 for *n*. Range of the *capcode* is 0 to 2097151.

NUMeric *capcode*
[GENerator:POCSAG:NUMeric capcode]
Generates Numeric message for the *capcode* specified. Range of *capcode* is 0 to 2097151.

GENerator:

POCSAG:

RATe

[GENerator:POCSAG:RATe]

Sets POCSAG rate to high if *b* is 1, low if *b* is 0.

RATe?

[GENerator:POCSAG:RATe?]

Returns 1 if POCSAG rate is high; returns 0 if rate is low.

Example:

```
*DMC "POCSAG",BEGIN  // Defines a macro named POCSAG.
GEN:POCSAG:RATE 1    // Sets POCSAG rate to High.
FOR X=1 TO 5         // Starts FOR loop of X to loop 5 times.
GEN:POCSAG:BEEP 2,X+5030   // Generates a POCSAG Tone - 2 beeps
                           // message
                     // using the capcode specified by X.
DELAY 3000           // Delays 3 seconds (for POCSAG generation).
*WAI                 // Wait for the last command to be executed.
NEXT X               // Loops X to the top of the loop until X=5.
GEN:POCSAG:RATE?     // Queries the POCSAG rate.  1 is returned
END                  // Ends the macro named POCSAG.
```

RCL *n*

[GENerator:RCL n]

Recalls RF Generator environment (routings and settings) stored in memory location *n*. Range of *n* is 1 to 9.

SCAN:

Except where noted, the following commands are for RF Generator Frequency Scan or Frequency List Scan modes.

ABORt

[GENerator:SCAN:ABORt]

Stops frequency scan or frequency list scan depending on present mode.

CONTinue

[GENerator:SCAN:CONTinue]

Starts frequency scan or frequency list scan depending on present mode.

FREQuency?

[GENerator:SCAN:FREQuency?]

Returns current frequency in kHz being scanned.

INCrement *n*

[GENerator:SCAN:INCrement n]

(Frequency Scan mode only) Specifies increment in kHz between frequencies to be scanned. Range of *n* is 0.0 to 2010000.0.

GENerator:

SCAN:

PAUse?

[GENerator:SCAN:PAUse?]

(Frequency Scan mode only) Returns a 1 if scanning is paused (stopped) or 0 if currently scanning.

RATe *n*

[GENerator:SCAN:RATe n]

(Frequency Scan mode only) Scan Rate. Specifes time period in seconds for RF Generator to sit on a frequency unless squelch is broken. Range of *n* is 0.02 to 99.99.

STARt *f*

[GENerator:SCAN:STARt f]

(Frequency Scan mode only) Starting Frequency value in kHz. Specifies lower limit frequency for scanning. Range of *f* is 250.0 to 2010000.0.

Start frequency must be a valid value and less than Stop frequency for Scan to operate.

STOP *f*

[GENerator:SCAN:STOP f]

(Frequency Scan mode only) Stop Frequency value in kHz. Specifies upper limit frequency for scanning. Range of *f* is 250.0 to 2010000.0.

FREQList:

The following commands are for RF Generator Frequency List Scan mode only.

RATe *n*

[GENerator:SCAN:FREQList:RATe n]

Frequency List Scan Rate. Specifies time period in seconds for RF Generator to sit on a frequency. Range of *n* is 0.02 to 99.99.

RATe?

[GENerator:SCAN:FREQList:RATe?]

Returns current value of Frequency List Scan Rate.

SINAD

[GENerator:SINAD]

Displays SINAD Meter on the RF Generator Operation Screen when followed by a **SCREEN:GENerator** command.

SPEAKer:SOURce *type*

[GENerator:SPEAKer:SOURce type]

Selects Test Set Speaker Input. Select OFF, FGEN (Function Generator), SINAD (SINAD/BER IN Connector) or EXTMOD (EXT MOD IN Connector) for *type*. This command takes effect when the RF Generator Operation Screen is updated.

Example:
```
GEN:SPEAK:SOUR EXTMOD          // Routes EXT MOD IN Connector to the
                               // Test Set Speaker.
```

GENerator:

STORe *n*
[GENerator:STORe n]

Stores current RF Generator environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

TONE *"sequence"*
[GENerator:TONE "sequence"]

Generates given *sequence* once the Audio code is selected using the **GENerator:TONE:TYPE** command. If selected Audio code is USER, characters contained in the *sequence* must be previously defined using the **GENerator:TONE:USER:DEFine** command. Valid *sequence* characters are the digits 0 through 9 and characters A, G, R and - (to signify a gap). Valid *sequence* characters for the USER code are the digits 0 through 9 and characters A through T.

Example:
```
GEN:TONE:TYPE CCIRH    // Selects CCIRH as RF Generator Audio Code.
GEN:TONE "5553434"     // Generates 5552424 as CCIRH Audio Signal.
```

TONE:

TYPE *code*
[GENerator:TONE:TYPE code]

Selects the Audio code to generate. Enter one of the following for *code*:

CCIR	EEA	EIA (U.S).
ZVEI	DDZVEI	DZVEI
NATEL	EURO	TONE56
CCIRH	CCIRH4	USER

See 6-5-2E and 6-5-2F.

USER:DEFine *"id",freq,duration*
[GENerator:TONE:USER:DEFine "id",freq,duration]

The *id* character is assigned a *freq* in Hz and a *duration* in ms. Range of *freq* is 0.0 to 9999.9. Range of *duration* is 20.0 to 9999.9. Valid characters for the *id* are the digits 0 through 9 and characters A through T. See 6-5-2E and 6-5-2F.

TREMote *f*
[GENerator:TREMote f]

Generates the Tone-remote *sequence* for the specified function tone frequency given by *f* in Hz. Select one of the following:

2050	1950	1850
1750	1650	1550
1450	1350	1250
1150	1050	

TREMote:STOP
[GENerator:TREMote:STOP]

Stops the Tone remote Guard Tone generated by a previous **GENerator:TREMote** command.

Example:
```
GEN:TREM 1350      // Generates a Tone Remote 1350 message.
GEN:TREM:STOP      // Stops the generating of the Tone Remote
                   // Guard Tone.
```

6-5-2 REMOTE RF GENERATOR EXAMPLES

A. GENERATING FM MODULATED RF SIGNALS

The following command sequence generates a 1 MHz RF signal FM modulated (5 kHz deviation) with a 1 kHz sine wave and routes the signal to the DEMOD OUT Connector:

```
SCREEN:GEN                     // Displays RF Generator Operation Screen.
GEN:FREQ 1000000               // Sets RF Generator Frequency to 1 MHz.
GEN:LEVEL 0 DB                 // Sets the RF Generator Level to 0 dBm.
GEN:OUTPUT:DEMOD 1             // Routes the RF Generator output to the
                               // DEMOD OUT Connector.
FGEN:GEN1:STATE 1              // Activates SOURCE 1 (AF Generator 1).
FGEN:GEN1:FREQ 1000            // Sets SOURCE 1 Frequency to 1000 Hz.
FGEN:GEN1:MODULATION:FM        // Sets SOURCE 1 Modulation to FM.
FGEN:GEN1:MODL 5               // Sets SOURCE 1 Deviation to 5 kHz.
FGEN:GEN1:SHAPE:SIN            // Sets SOURCE 1 Wave Shape to a sine wave.
```

B. GENERATING AM MODULATED RF SIGNALS

The following command sequence generates a 10 MHz RF signal, AM modulated (80% modulation), with a 2.5 kHz sine wave and routes the signal to the AUDIO OUT Connector:

```
SCREEN:GEN                     // Displays RF Generator Operation Screen.
GEN:FREQ 10 MHZ                // Sets RF Generator Frequency to 10 MHz.
GEN:LEVEL 0 DB                 // Sets the RF Generator Level to 0.0 dBm.
GEN:OUTPUT:AUDIO 1             // Routes the RF Generator output to the
                               // AUDIO OUT Connector.
FGEN:GEN1:STATE 1              // Activates SOURCE 1 (AF Generator 1).
FGEN:GEN1:FREQ 2500            // Sets SOURCE 1 Frequency to 2500 Hz.
FGEN:GEN1:MODULATION:AM        // Sets SOURCE 1 Modulation to AM.
FGEN:GEN1:MODL 80              // Sets SOURCE 1 Modulation to 80%.
FGEN:GEN1:SHAPE:SIN            // Sets SOURCE 1 Wave Shape to a sine wave.
```

C. GENERATING EXTERNALLY MODULATED RF SIGNALS

The following command sequence generates a RF 15 MHz signal, AM modulated (80% modulation), with an external signal applied to the EXT MOD IN Connector and sends the signal to the AUDIO OUT Connector:

```
SCREEN:GEN                     // Displays RF Generator Operation Screen.
GEN:FREQ 15000                 // Sets RF Generator Frequency to 15 MHz.
GEN:LEV -20                    // Sets the RF Generator Level to -20 dBm.
GEN:OUT:AUD 1                  // Routes the RF Generator Output to the
                               // AUDIO OUT Connector.
FGEN:EXT:STATE 1               // Activates SOURCE EXT (for external
                               // modulation).
FGEN:EXT:MOD:AM                // Sets SOURCE EXT Modulation to AM.
FGEN:EXT:MODL 80               // Sets SOURCE EXT Modulation Level to 80%.
```

D. GENERATING A DTMF CODED SIGNAL

The following command sequence generates a 450 MHz signal FM modulated with a DTMF Code. The modulation level is 4 kHz and the RF Output Level is -60 dBm.

```
SCREEN:GEN                     // Displays RF Generator Operation Screen.
*WAI                           // Waits for previous commands to execute.
GEN:FREQ 450 MHZ               // Sets RF Generator Frequency to 450 MHz.
GEN:LEV -60 DB                 // Sets the RF Generator Level to -60 dBm.
FGEN:GEN3:ENCODE DTMF          // Selects DTMF as the signaling format.
SCREEN:GEN                     // Renews RF Generator Operation Screen.
*WAI                           // Waits for previous commands to execute.
FGEN:GEN3:MOD:FM               // Sets SOURCE 3 Modulation to FM.
FGEN:GEN3:MODL 4               // Sets SOURCE 3 Modulation Level to 4 kHz.
GEN:DTMF "55523*#",80,70       // Generates 55523*# DTMF coded signal with
                               // 80 ms mark time and a 70 ms space time.
```

E. GENERATING AUDIO TWO TONE CODING

The following command sequence generates a 150 MHz signal FM modulated with an Audio Two Tone Code. The modulation level is 4 kHz and the RF Output Level is 0 dBm.

```
SCREEN:GEN                     // Displays RF Generator Operation Screen.
*WAI                           // Waits for previous commands to execute.
GEN:FREQ 150 MHZ               // Sets RF Generator Frequency to 150 MHz.
GEN:LEV 0 DB                   // Sets the RF Generator Level to 0 dBm.
FGEN:GEN3:ENCODE TONE          // Selects Audio as the signaling format.
SCREEN:GEN                     // Renews RF Generator Operation Screen.
*WAI                           // Waits for previous commands to execute.
FGEN:GEN3:MOD:FM               // Sets SOURCE 3 Modulation to FM.
FGEN:GEN3:MODL 4               // Sets SOURCE 3 Modulation Level to 4 kHz.
GEN:TONE:TYPE USER             // Selects User Defined for the Audio Code.
GEN:TONE:USER:DEF "0,0,100"    // Defines 0 Tone as 0 Hz, 100 ms.
GEN:TONE:USER:DEF "1,880,500"  // Defines 1 Tone as 880 Hz, 500 ms.
GEN:TONE:USER:DEF "2,2200,500"// Defines 2 Tone as 2200 Hz, 500 ms.
GEN:TONE 102                   // Generates the User Defined Audio Code
                               // defined in the 3 previous commands.
```

F. GENERATING A 5/6 AUDIO TONE SEQUENCE

The following command sequence generates a 162 MHz signal FM modulated with an Audio 5/6 Tone Code. The modulation level is 4 kHz and the RF Output Level is 0 dBm.

```
SCREEN:GEN                     // Displays RF Generator Operation Screen.
*WAI                           // Waits for previous commands to execute.
GEN:FREQ 162 MHZ               // Sets RF Generator Frequency to 162 MHz.
GEN:LEV 0 DB                   // Sets the RF Generator Level to 0 dBm.
FGEN:GEN3:ENCODE TONE          // Selects Audio as the signaling format.
SCREEN:GEN                     // Renews RF Generator Operation Screen.
*WAI                           // Waits for previous commands to execute.
FGEN:GEN3:MOD:FM               // Sets SOURCE 3 Modulation to FM.
FGEN:GEN3:MODL 4               // Sets SOURCE 3 Modulation Level to 4 kHz.
GEN:TONE:TYPE:USER             // Selects User Defined for the Audio Code.
GEN:TONE:USER:DEF 0,900,150    // Defines 0 Tone at 900 Hz and 150 ms.
GEN:TONE:USER:DEF 1,1100,80    // Defines 1 Tone at 1100 Hz and 80 ms.
GEN:TONE:USER:DEF 2,1200,80    // Defines 2 Tone at 1200 Hz and 80 ms.
GEN:TONE:USER:DEF 3,1300,80    // Defines 3 Tone at 1300 Hz and 80 ms.
GEN:TONE:USER:DEF 4,1400,80    // Defines 4 Tone at 1400 Hz and 80 ms.
GEN:TONE:USER:DEF A,0,40       // Defines A Tone at 0 Hz and 40 ms.
GEN:TONE 0A1234                // Generates the User Defined Audio Code
                               // defined in the 6 previous commands.
```

G. GENERATING DCS CODE

The following command sequence generates a 162.450 MHz signal FM modulated with a DCS Code of 456. The modulation level is 1 kHz and the RF Output Level is 0 dBm.

```
SCREEN:GEN                     // Displays RF Generator Operation Screen.
*WAI                           // Waits for previous commands to execute.
GEN:FREQ 162450                // Sets 162.450 MHz RF Generator Frequency.
GEN:LEV 0 DB                   // Sets the RF Generator Level to 0 dBm.
FGEN:GEN3:ENCODE DIG           // Selects Digital as the signaling format.
SCREEN:GEN                     // Renews RF Generator Operation Screen.
*WAI                           // Waits for previous commands to execute.
FGEN:GEN3:MOD:FM               // Sets SOURCE 3 Modulation to FM.
FGEN:GEN3:MODL 1               // Sets SOURCE 3 Modulation Level to 1 kHz.
GEN:DCS:NORM 456               // Generates a 456 DCS Code.
GEN:DCS:STOP                   // Stops the generating of the DCS Code.
```

H. GENERATING POCSAG CODE

The following command sequence generates a 930 MHz signal FM modulated with a POCSAG Code. The modulation level is 4 kHz and the RF Output Level is 0 dBm.

```
SCREEN:GEN                      // Displays RF Generator Operation Screen.
*WAI                            // Waits for previous commands to execute.
GEN:FREQ 450 MHZ                // Sets RF Generator Frequency to 450 MHz.
GEN:LEV 0 DB                    // Sets the RF Generator Level to 0 dBm.
FGEN:GEN3:ENCODE DIG            // Selects Digital as the signaling format.
SCREEN:GEN                      // Renews RF Generator Operation Screen.
*WAI                            // Waits for previous commands to execute.
FGEN:GEN3:MOD:FM                // Sets SOURCE 3 Modulation to FM.
FGEN:GEN3:MODL 4                // Sets SOURCE 3 Modulation Level to 4 kHz.
GEN:POCSAG:RATE 0               // Sets POCSAG rate to Low.
FOR Y=1 TO 5                    // Starts loop.  Y to be looped 5 times.
   GEN:POCSAG:ALPHA:UPPER Y+5130  // Generates an Alpha upper POCSAG
                                  // message using the specified capcode.
   DELAY 3000                   // Delays 3 seconds (for POCSAG generation).
   *WAI                         // Waits for the last command to be executed.
NEXT Y                          // Loops to top of the loop until Y is 5.
```

I. GENERATING 2805 CODE

The following command sequence generates a 155 MHz signal FM modulated with a 2805 Tone (with frequency reset to 1500 Hz). The modulation level is 4 kHz and the RF Output Level is -60 dBm.

```
SCREEN:GEN                      // Displays RF Generator Operation Screen.
*WAI                            // Waits for previous commands to execute.
GEN:FREQ 155 MHZ                // Sets RF Generator Frequency to 155 MHz.
GEN:LEV -60 DB                  // Sets the RF Generator Level to -60.0 dBm.
FGEN:GEN3:ENCODE RCC            // Selects RCC as the signaling format.
SCREEN:GEN                      // Renews RF Generator Operation Screen.
*WAI                            // Waits for previous commands to execute.
FGEN:GEN3:MOD:FM                // Sets SOURCE 3 Modulation to FM.
FGEN:GEN3:MODL 4                // Sets SOURCE 3 Modulation Level to 4 kHz.
GEN:DIAL:FREQ 1500              // Sets 2805 Tone Frequency to 1500 Hz.
GEN:DIAL 5551234                // Generates a 5551234 2805 Tone signal.
```

6-6 RECEIVER

6-6-1 RECEIVER COMMANDS

RECeiver:

AGC:

AUTO
[RECeiver:AGC:AUTO]
Sets Automatic Gain Control to automatic setting.

MANual *n*
[RECeiver:AGC:MANual n]
Sets Automatic Gain Control to manual setting and sets level to *n*. Range of *n* is 0 to 255.

Example: `REC:AGC:MAN 130        // Sets AGC setting to Manual and sets`
`                                // Manual AGC setting to 130.`

USER:*xxx*
[RECeiver:AGC:USER:xxx]
Sets Automatic Gain Control to User. Select one of the following for *xxx*: MEASure, SPeech, DATA, HIGH, TYPE1, TYPE2 or TYPE3.

Example: `REC:AGC:USER:HIGH    // Sets AGC setting to User Defined High.`

CHANnel *n*
[RECeiver:CHANnel n]
Sets Receive Frequency to cellular channel *n* (1 to 2047, depending on format) in the format selected using the **REC:CHAN:FORM** command. The Receiver Operation Screen displays selected cellular channel when Receiver is in Channel Mode (**REC:MODE CHAN** command).

CHANnel:

FORMat:

AMPS:

FORward
[RECeiver:CHANnel:FORMat:AMPS:FORward]
Selects AMPS (NADC-U8) Forward channels.

REVerse
[RECeiver:CHANnel:FORMat:AMPS:REVerse]
Selects AMPS (NADC-U8) Reverse channels.

ETACS:

FORward
[RECeiver:CHANnel:FORMat:ETACS:FORward]
Selects ETACS Forward channels.

REVerse
[RECeiver:CHANnel:FORMat:ETACS:REVerse]
Selects ETACS Reverse channels.

RECeiver:

CHANNel:

FORMat:

NAMPS:

FORward
[RECeiver:CHANNel:FORMat:NAMPS:FORward]
Selects NAMPS Forward channels.

REVerse
[RECeiver:CHANNel:FORMat:NAMPS:REVerse]
Selects NAMPS Reverse channels.

BAND:*x*
[RECeiver:CHANNel:FORMat:NAMPS:BAND:x]
Selects NAMPS Narrow Analog channel designator. Range of *x* is Lower, Middle or Upper.

NT400:

FORward
[RECeiver:CHANNel:FORMat:NT400:FORward]
Selects NT400 Forward channels.

REVerse
[RECeiver:CHANNel:FORMat:NT400:REVerse]
Selects NT400 Reverse channels.

NADC:

FORward
[RECeiver:CHANNel:FORMat:NADC:FORward]
Selects NADC Forward. Utilizes current setting of NADC band.

REVerse
[RECeiver:CHANNel:FORMat:NADC:REVerse]
Selects NADC Reverse. Utilizes current setting of NADC band.

BAND:*x*
[RECeiver:CHANNel:FORMat:NADC:BAND:x]
Selects NADC band. Range of *x* is U8, U4 or HYper.

FORMat?
[RECeiver:CHANNel:FORMat?]
Returns Receiver Channel Format (NADC:FORWARD, NADC:REVERSE, ETACS:FORWARD, ETACS:REVERSE, NAMPS:FORWARD or NAMPS:REVERSE).

RECeiver:

CHANnel:

BAND?

[RECeiver:CHANnel:BAND?]

Returns band for Receiver Channel Format. Returns one of the following bands for the associated channel format:

FORMAT	BAND
NADC	U8, U4 or HY
ETACS	NOT AVAILABLE
NAMPS	LOWER, MIDDLE or UPPER

DCS:

INVert?

[RECeiver:DCS:INVert?]

Returns the three octal digits in inverted mode, decoded from the Digital Coded Squelch (DCS) code. Returns -1 if none available or if invalid for inverted DCS.

NORMal?

[RECeiver:DCS:NORMal?]

Returns the three octal digits decoded from the DCS code. Returns -1 if none available or if invalid for normal DCS. See 6-6-2E.

STATe *b*

[RECeiver:DCS:STATe b]

Enables DCS decoding if *b* is 1, disables if *b* is 0. **RECeiver:DECode DIGital** and **RECeiver:DIGital DCS** (or **DCSINV**) commands must be initiated prior to enabling DCS decoding. DCS decoding must be disabled after decoding is finished.

Example:

```
REC:DEC DIG          // Sets Receiver for decoding digital data.
REC:DIG DCS          // Prepares Receiver for decoding DCS.
REC:DCS:STAT 1       // Enables DCS decoding.
REC:DCS:NORM?        // Returns decoded Normal DCS digits.
REC:DCS:STAT 0       // Disables DCS decoding.
```

DECode *type*

[RECeiver:DECode type]

Sets Receiver for decoding *type*. The setting for *type* is DTMF, TONE or DIGital.

DEVRms

[RECeiver:DEVRms]

Displays Deviation Meter (RMS) reading on Receiver Operation Screen when followed by **SCREEN:RECeiver** command.

DIGital *type*

[RECeiver:DIGital type]

Sets digital *type.* Used with the **RECeiver:DECode DIGital** command to prepare the Receiver for decoding. The setting for *type* is DCS, DCSINV, POCSAG, DSAT or DST.

REceiver:

DISTortion

[RECeiver:DISTortion]

Displays Distortion Meter reading on Receiver Operation Screen when followed by **SCREEN:RECeiver** command.

DMM

[RECeiver:DMM]

Displays Digital Multimeter reading on Receiver Operation Screen when followed by **SCREEN:RECeiver** command.

DSAT:STATe *b*

[RECeiver:DSAT:STATe b]

Enables DSAT Decoding Function if *b* is 1, disables if *b* is 0. **RECeiver:DECode DIGital** and **RECeiver:DIGital DSAT** commands must be initiated prior to enabling DSAT decoding. DSAT decoding must be disabled after decoding is finished.

DSAT?

[RECeiver:DSAT?]

Returns DSAT reading. Returns -1 if none available or invalid for normal transmission.

DST:STATe *b*

[RECeiver:DST:STATe b]

Enables DST Decoding Function if b is 1, disables if b is 0. **RECeiver:DECode DIGital** and **RECeiver:DIGital DST** commands must be initiated prior to enabling DST decoding. DST decoding must be disabled after decoding is finished.

DST?

[RECeiver:DST?]

Returns DST reading. Returns -1 if none available or invalid for normal transmission.

DTMF:STATe *b*

[RECeiver:DTMF:STATe b]

Enables DTMF Decoding Function if *b* is 1, disables if *b* is 0. See 6-6-2C.

DTMF?

[RECeiver:DTMF?]

Returns string of decoded digits or -1 if nothing decoded.

Example:

```
SCREEN:REC          // Displays Receiver Operation Screen.
REC:DEC DTMF        // Sets Receiver for decoding DTMF signals.
REC:DTMF:STAT 1     // Enables DTMF decoding.
REC:DTMF?           // Returns decoded DTMF digits.
REC:DTMF:STAT 0     // Disables DTMF decoding.
```

RECeiver:

FIND:

FREQuency?

[RECeiver:FIND:FREQuency?]

Returns frequency of first signal with amplitude larger than Find reference level. Returns 0 if no signal is found.

REFerence *n*

[RECeiver:FIND:REFerence n]

Sets Find reference level to *n* dBm. Range of *n* is -110 to -5.

REFerence?

[RECeiver:FIND:REFerence?]

Returns Find reference level in dBm (-110 to -5).

Example:

```
REC:FIND:REF -65        // Sets Find Reference Level to -65 dBm.
REC:FIND:FREQ?          // Returns the first frequency (in kHz)
                        // containing a signal greater than -65 dBm.
REC:FIND:REF?           // Queries the current Find Reference Level
                        // (in dBm).  Returns -65.
```

FMZ *n*

[RECeiver:FMZ n]

Zeros FM Deviation Meter. Displays the Receiver Operation Screen and selects FM*n* Modulation. Range of *n* is from 1 to 4.

FREQuency *n* [*units*]

[RECeiver:FREQuency n [units]]

Sets Receiver Frequency. Range of *n* is 250.0 kHz to 2010.0000 MHz. Select HZ, KHZ or MHZ for *units*. *units* is optional with a default of KHZ.

FREQuency?

[RECeiver:FREQuency?]

Returns Receiver Frequency in kHz (250.0 to 2010000.0).

Example:

```
REC:FREQ 100 MHZ      // Sets Receiver Frequency to 100 MHz.
REC:FREQ?             // Queries the Receiver Frequency.
                      // Returns 100000 (100000 kHz).
```

RECeiver:

INPut:

ANTenna

[RECeiver:INPut:ANTenna]

Selects ANTENNA IN Connector as the Receiver Input Source. Displays Signal Strength Meter reading on the Receiver Operation Screen.

> The following Attenuation commands control the Input Attenuation for the Receiver and selection/deselection of LNA (Low Noise Amplifier). Use **RECeiver:INPut:ATTenuation:LNA** to select LNA and 0 dB of input attenuation. To deselect LNA and select a specific value of attenuation use **RECeiver:INPut:ATTenuation** *n*.

ATTenuation:LNA

[RECeiver:INPut:ATTenuation:LNA]

Sets Receiver Input Attenuation to LNA.

ATTenuation:LNA?

[RECeiver:INPut:ATTenuation:LNA?]

Returns current state of the Low Noise Amplifier. Returns 1 if LNA is selected; 0 otherwise.

> LNA (Low Noise Amplifier) has the same attenuation as 0 dB (see **RECeiver:INPut: ATTenuation** *n*), but LNA has a lower noise figure. LNA is the preferred option for doing off-the-air applications.

ATTenuation *n*

[RECeiver:INPut:ATTenuation n]

Sets Receiver Input Attenuation to *n* dB. Possible values for *n*: 0, 5, 10, 15, 20, 25, 30.

ATTenuation?

[RECeiver:INPut:ATTenuation?]

Returns the current value of Input Attenuation.

TR

[RECeiver:INPut:TR]

Selects T/R Connector as the Receiver Input Source. Displays Power Meter reading on the Receiver Operation Screen.

RECeiver:

MODE type

[RECeiver:MODE type]

Selects the Receiver Mode. Channel Mode displays cellular channel frequency according to **REC:CHAN** commands.

MODE	*type*
Direct	DIRect
Channel	CHANnel
Frequency Scan	SCAN
Frequency List	LIST
Frequency List Scan	FLScan

Example:
```
REC:INP:TR          // Selects TR Connector for Receiver Input.
REC:INP:ATT 20      // Sets Receiver Input Attenuation to 20 dB.
REC:MODE CHAN       // Selects Receiver Channel Mode.
```

MODE?

[RECeiver:MODE?]

Returns current mode (DIRECT, CHANNEL, FREQUENCY SCAN, FREQUENCY LIST or FREQUENCY LIST SCAN).

MODMeter

[RECeiver:MODMeter]

Displays Modulation Meter reading on the Receiver Operation Screen when followed by a **SCREEN:RECeiver** command and AM is selected modulation.

MODulation:

AM*n*

[RECeiver:MODulation:AMn]

Selects Amplitude Modulation. Select 1 or 2 for *n*.

BFO

[RECeiver:MODulation:BFO]

Selects Beat Frequency Oscillation.

FM*n*

[RECeiver:MODulation:FMn]

Selects Frequency Modulation. Select 1, 2, 3 or 4 for *n*.

Example:
```
REC:MOD:FM4         // Selects FM4 as Receiver Modulation.
```

RECeiver:

MODulation:

PM
[RECeiver:MODulation:PM]
Selects Phase Modulation.

USER:

FILTer *f*
[RECeiver:MODulation:USER:FILTer f]
Sets User Defined Modulation IF Filter to *f* kHz. Select 3, 30 or 300 for *f*.

FILTer?
[RECeiver:MODulation:USER:FILTer?]
Returns the current setting of the User Defined Modulation IF Filter in kHz.

MODulation:*type*
[RECeiver:MODulation:USER:MODulation:type]
Selects User Defined Modulation *type* (FM, AM, BFO, PM or DATA [FM DATA]).

MODulation?
[RECeiver:MODulation:USER:MODulation?]
Returns User Defined Modulation type.

POST:

APASs
[RECeiver:MODulation:USER:POST:APASs]
Selects All Pass Post Detection Filter for User Defined Modulation.

BPASs *fl,fh*
[RECeiver:MODulation:USER:POST:BPASs fl,fh]
Selects Bandpass Post Detection Filter for the User Defined Modulation. Lower cutoff frequency is set to *fl* kHz with a range of 0.5 to 20. Higher cutoff frequency is set to *fh* kHz with a range of 0.1 to 30.

BPASs:

HIGH *f*
[RECeiver:MODulation:USER:POST:BPASs:HIGH f]
Sets Lower cutoff frequency of the Band-Pass Post Detection Filter in kHz. Range of f is 0.5 to 20.0.

HIGH?
[RECeiver:MODulation:USER:POST:BPASs:HIGH?]
Returns the Lower cutoff frequency of the Band-Pass Post Detection Filter in kHz.

RECeiver:

MODulation:

USER:

POST:

BPASs:

LOW *f*
[RECeiver:MODulation:USER:POST:BPASs:LOW f]
Sets Upper cutoff frequency of the Band-Pass Post Detection Filter in kHz. Range of f is 0.1 to 30.0.

LOW?
[RECeiver:MODulation:USER:POST:BPASs:LOW?]
Returns the Upper cutoff frequency of the Band-Pass Post Detection Filter in kHz.

CWEight
[RECeiver:MODulation:USER:POST:CWEight]
Selects C-Weighted Post Detection Filter (C-message noise weighting curve response) for the User Defined Modulation. Refer to MIL-STD-188-200.

CMESsage
[RECeiver:MODulation:USER:POST:CMESsage]
Same as **RECeiver:MODulation:USER:POST:CWEight** command.

CWT
[RECeiver:MODulation:USER:POST:CWT]
Same as **RECeiver:MODulation:USER:POST:CWEight** command.

HPASs *fl*
[RECeiver:MODulation:USER:POST:HPASs fl]
Selects High-Pass Post Detection Filter for the User Defined Modulation. Cutoff frequency is set to *fl* kHz with a range of 0.5 to 20.

HPASs?
[RECeiver:MODulation:USER:POST:HPASs?]
Returns the cutoff frequency for the User Defined Modulation High-Pass Post Detection Filter.

RECeiver:

MODulation:

USER:

POST:

LPASs *fh*

[RECeiver:MODulation:USER:POST:LPASs fh]

Selects Low-Pass Post Detection Filter for the User Defined Modulation. Cutoff frequency is set to *fh* kHz with a range of 0.1 to 30.

Example:

```
REC:MOD:USER:FILT 30     // Selects the 30 kHz IF Filter.
REC:MOD:USER:MOD:DATA    // Selects User Defined FM DATA for Receiver
                         // Modulation.
REC:MOD:USER:POST:BPAS 10,20  // Selects a Bandpass Post Detection
                         // Filter with 10 kHz lower cutoff and 20 kHz
                         // higher cutoff frequencies.
REC:MOD?                 // Queries the Receiver Modulation.  USER is
                         // returned.
```

LPASs?

[RECeiver:MODulation:USER:POST:LPASs?]

Returns the cutoff frequency for the User Defined Modulation Low-Pass Post Detection Filter.

POST?

[RECeiver:MODulation:USER:POST?]

Returns the current setting of the User Defined Post Detection Filter. Returns one of the following: APAS, LPAS, HPAS, BPAS or CWE.

MODulation?

[RECeiver:MODulation?]

Returns Receiver Modulation (FM, AM, BFO, PM, DATA or USER).

OUTput:

AUDio *b*

[RECeiver:OUTput:AUDio b]

Routes Demodulated Audio to the AUDIO OUT Connector if *b* is 1. Disconnects the AUDIO OUT Connector if *b* is 0.

DEMOD *b*

[RECeiver:OUTput:DEMOD b]

Routes Demodulated Audio to the DEMOD OUT Connector if *b* is 1. Disconnects the DEMOD OUT Connector if *b* is 0.

SPEAKer *b*

[RECeiver:OUTput:SPEAKer b]

Routes Demodulated Audio to the Speaker if *b* is 1. Disconnects Speaker if *b* is 0.

RECeiver:

PMRms

[RECeiver:PMRms]

Displays Phase Meter (RMS) reading when followed by a **SCREEN:RECeiver** command.

POCSAG:

CAPcode?

[RECeiver:POCSAG:CAPcode?]

Returns the received capcode or -1 if a capcode is not received.

MESSage?

[RECeiver:POCSAG:MESSage?]

Returns message from the decoded POCSAG signal or -1 if none available.

RATe *b*

[RECeiver:POCSAG:RATe b]

Sets POCSAG rate to high if *b* is 1, low if *b* is 0.

RATe?

[RECeiver:POCSAG:RATe?]

Returns 1 if POCSAG rate is high, 0 if rate is low.

STATe *b*

[RECeiver:POCSAG:STATe b]

Enables POCSAG decoding if *b* is 1, disables if *b* is 0. **RECeiver:DECode DIGital** and **RECeiver:DIGital POCSAG** commands must be initiated prior to enabling POCSAG decoding. POCSAG decoding must be disabled after decoding is finished.

Changing screens with POCSAG enabled may cause Test Set to lock up.

TYPE?

[RECeiver:POCSAG:TYPE?]

Returns POCSAG Function Type. One of the following is returned:

TONE 1 BEEP	TONE 2 BEEPS	TONE 3 BEEPS
TONE 4 BEEPS	NUMERIC	ALPHANUMERIC
NO MESSAGE		

Example:

```
REC:DEC DIG          // Sets Receiver for decoding digital data.
REC:DIG POCSAG       // Prepares Receiver for decoding POCSAG.
REC:POCSAG:STAT 1    // Enables POCSAG decoding.
REC:POCSAG:RATE 1    // Selects High rate to decode.
REC:POCSAG:MESS?     // Queries for the decoded message.
REC:POCSAG:CAP?      // Queries for the decoded capcode.
REC:POCSAG:TYPE?     // Queries for the decoded message type.
REC:POCSAG:RATE?     // Queries for the current rate setting.
                     // 1 is returned (for High).
REC:POCSAG:STAT 0    // Disables POCSAG decoding.
```

RCL *n*

[RECeiver:RCL n]

Recalls Receiver environment (routings and settings) stored in memory location *n*. Range of *n* is 1 to 9.

RECeiver:

SCAN:

Except where note, the following commands are for Receiver Frequency Scan or Frequency List Scan modes.

ABORt
[RECeiver:SCAN:ABORt]
Stops frequency scan or frequency list scan depending on present mode.

CONTinue
[RECeiver:SCAN:CONTinue]
Starts frequency scan or frequency list scan depending on present mode.

FREQuency?
[RECeiver:SCAN:FREQuency?]
Returns frequency currently being scanned in kHz.

INCrement *f*
[RECeiver:SCAN:INCrement f]
(Frequency Scan mode only.) Sets Receiver Scan increment to *f* kHz. Range of *f* is 0.0 to 2010000.0.

PAUSe *t*
[RECeiver:SCAN:PAUSe t]
(Frequency Scan mode only.) Sets Receiver Pause rate to *t* sec. Range of *t* is 0.0 to 99.9. Receiver Scan pause time is length of time Scan Function pauses at frequency with broken squelch. Scan Function stops permanently on squelch broken frequency if pause set to 0.0.

PAUSe?
[RECeiver:SCAN:PAUSe?]
(Frequency Scan mode only.) Returns a 1 if scanning is paused (stopped) or 0 if currently scanning.

RATe *t*
[RECeiver:SCAN:RATe t]
Sets Receiver Scan rate to *t* sec. Range of *t* is 0.00 to 99.99. Receiver Scan rate is the time each frequency is scanned with squelch unbroken.

STARt *f*
[RECeiver:SCAN:STARt f]
Sets Receiver Scan starting frequency to *f* kHz. Range of *f* is 250.0 to 2010000.0.

RECeiver:

SCAN:

STOP *f*

[RECeiver:SCAN:STOP f]

Sets Receiver Scan stopping frequency to *f* kHz. Range of *f* is 250.0 to 2010000.0.

Example:
```
REC:SCAN:STAR 1000    // Sets Receiver Scan starting frequency to
                      // 1 MHz.
REC:SCAN:STOP 100000  // Sets Receiver Scan stopping frequency to
                      // 100 MHz.
REC:SCAN:INC 250      // Sets Receiver Scan increment to 250 kHz.
REC:SCAN:RAT 1.5      // Sets Receiver Scan rate to 1.5 sec.
REC:SCAN:PAUS 10      // Sets Receiver Scan pause time to 10 sec.
REC:SCAN:CONT         // Starts Receiver Scan.
REC:SCAN:ABOR         // Stops Receiver Scan.
```

FREQList:

The following commands are for Receiver Frequency List Scan mode only.

PAUSe *n*

[RECeiver:SCAN:FREQList:PAUSe n]

Frequency List Scan Pause Time. Specifies time period in seconds for Receiver to sit on a frequency if squelch is broken. Range of *n* is 0.0 to 99.9.

If 0.0 is specified, Receiver sits on frequency as long as squelch is broken.

PAUSe?

[RECeiver:SCAN:FREQList:PAUSe?]

Returns current value of Frequency List Scan Pause Time.

RATe *n*

[RECeiver:SCAN:FREQList:RATe n]

Frequency List Scan Rate. Specifies time period in seconds for Receiver to sit on a frequency unless squelch is broken. Range of *n* is 0.02 to 99.99.

RATe?

[RECeiver:SCAN:FREQList:RATe?]

Returns current value of Frequency List Scan Rate.

SQUelch *b*

[RECeiver:SCAN:FREQList:SQUelch b]

Enables (*b* = 1) or disables (*b* = 0) Squelch.

SQUelch?

[RECeiver:SCAN:FREQList:SQUelch?]

Returns current state of Squelch.

RECeiver:

SINAD
[RECeiver:SINAD]

Displays SINAD Meter reading when followed by a **SCREEN:RECeiver** command.

SQUelch *n*
[RECeiver:SQUelch n]

Sets squelch to *n*. Range of *n* is 0.0 to 1.0.

SQUelch?
[RECeiver:SQUelch?]

Returns squelch level (0.0 to 1.0).

Example:
```
SQU .3          // Sets Squelch level to 3/10 of total
                // allowable setting.
SQU?            // Queries Squelch level.  .3 is returned.
```

STORe *n*
[RECeiver:STORe n]

Stores current Receiver environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

TONE:

USER:DEFine *"id",f,d*
[RECeiver:TONE:USER:DEFine "id",f,d]

Defines a User Defined Audio Tone and duration to be received; where *id* is the character associated with the defined tone, *f* is the freqency in Hz of the defined tone and *d* is the duration of the defined tone in ms. Range of *id* is 0 to 9 and A to T, range of *f* is 0.0 to 9999.9 and range of *d* is 20 to 9999.

Same as **GENerator:TONE:USER:DEFine** command. Both commands act upon the same User Defined Audio Tone database.

DURation? *"id"*
[RECeiver:TONE:DURation? "id"]

Returns the Duration in ms (20 to 9999) of the specified User Defined Audio Tone. *id* is the character associated with the defined tone. Range of *id* is 0 to 9 and A to T.

FREQuency? *"id"*
[RECeiver:TONE:FREQuency? "id"]

Returns the frequency setting in Hz of the specified User Defined Audio Tone *id*. Range of *id* is 0 to 9 and A to T.

INPut *n*
[RECeiver:TONE:INPut n]

Selects the decode signal input as defined below:

n	INPUT
0	Demod Audio
1	SINAD/BER
2	Ext Mod

RECeiver:

TONE:

STATe *b*
[RECeiver:TONE:STATe b]

Enables Audio Tone decoding if *b* is 1, disables if *b* is 0. **RECeiver:DECode TONE** command must be initiated prior to enabling Audio Tone decoding. Audio Tone decoding must be disabled after decoding is finished.

TYPE *xxx*
[RECeiver:TONE:TYPE xxx]

Selects Audio Tone to be decoded. Choose one of the following for *xxx*:

CCIR	EEA	EIA
ZVEI	DDZVEI	DZVEI
NATEL	EURO	TONE56
CCIRH	CCIRH4	USER

See 6-6-2D and F.

TONE?
[RECeiver:TONE?]

Returns the decoded Audio Tone sequence or -1 if not available.

Example:
```
REC:DEC TONE          // Sets Receiver for decoding Audio Tones.
REC:TONE:STAT 1       // Enables Audio Tone decoding.
REC:TONE:TYPE CCIR    // Selects CCIR as the Audio Tone type.
REC:TONE?             // Returns the decoded Audio Tone sequence.
REC:TONE:STAT 0       // Disables Audio Tone decoding.
```

VOLume *n*
[RECeiver:VOLume n]

Sets volume to *n*. Range of *n* is 0.0 to 1.0.

VOLume?
[RECeiver:VOLume?]

Returns the volume level (0.0 to 1.0).

VOLume:

AUTO *b*
[RECeiver:VOLume:AUTO b]

Enables Automatic Volume Control if *b* is 1, disables if *b* is 0.

AUTO?
[RECeiver:VOLume:AUTO?]

Returns Automatic Volume Control state. 1 is returned if enabled, 0 if disabled.

6-6-2 REMOTE RECEIVER EXAMPLES

A. RECEIVING FM SIGNALS

The following command sequence receives a 96 MHz FM signal through the ANTENNA IN Connector and outputs the audio signal through the Test Set Speaker.

```
*DMC "REC_FM",BEGIN           // Define a macro named REC_FM.
SCREEN:REC                    // Display the Receiver Operation Screen.
REC:FREQ 96000                // Set Receiver Frequency to 96 MHz.
REC:MODULATION:FM1            // Select FM1 for Receiver Modulation.
REC:INPUT:ANTENNA             // Select the ANTENNA IN Connector for
                              // Receiver Input.
REC:INPUT:ATTENUATION 0       // Select 0 dB for the Receiver Input
                              // Attenuation Level.
N=1,X=0                       // Set variables to initial values.
WHILE X=0                     // Start WHILE loop to loop as long as
                              // squelch is unbroken.
   REC:SQU N                  // Set squelch 0.02 lower than last setting.
   DELAY 25                   // Allow time for squelch to settle.
   N=N-0.02                   // Decrement variable N (use to set squelch).
   X=:MEAS:SQU?               // Set X to 1 when squelch breaks (see 6-16).
WEND                          // End of WHILE loop.  X loops to the top as
                              // long as X=0 (squelch unbroken).
REC:OUTPUT:SPEAKER 1          // Routes Receiver Output to the Test Set
                              // Speaker.
END                           // End of macro REC_FM.
```

B. RECEIVING AM SIGNALS

The following command sequence receives a 1240 kHz AM signal through the ANTENNA IN Connector and outputs the audio signal through the DEMOD OUT Connector.

```
*DMC "REC_AM",BEGIN           // Define a macro named REC_AM.
SCREEN:REC                    // Display the Receiver Operation Screen.
REC:FREQ 1240                 // Set Receiver Frequency to 96 MHz.
REC:MODULATION:AM1            // Select AM1 for Receiver Modulation.
REC:INPUT:ANTENNA             // Select the ANTENNA IN Connector for
                              // Receiver Input.
REC:INPUT:ATTENUATION 0       // Select 0 dB for the Receiver Input
                              // Attenuation Level.
N=1,X=0                       // Set variables initially.
WHILE X=0                     // Start WHILE loop to loop as long as
                              // squelch is unbroken.
   REC:SQU N                  // Set squelch 0.02 lower than last
setting.
   DELAY 25                   // Allow time for squelch to settle.
   N=N-0.02                   // Decrement variable N (use to set squelch).
   X=:MEAS:SQU?               // Set X to 1 when squelch breaks (see 6-16).
WEND                          // End of WHILE loop.  X loops to the top as
                              // long as X=0 (squelch unbroken).
REC:OUTPUT:DEMOD 1            // Route Receiver Output to the DEMOD OUT
                              // Connector.
END                           // End of macro REC_AM.
```

C. DECODING DTMF CODED SIGNALS

The following command sequence receives a 450 MHz FM modulated DTMF signal and routes the demodulated signal to the AUDIO OUT Connector and the Test Set Speaker.

```
*DMC "REC_DTMF",BEGIN            // Define a macro named REC_DTMF.
SCREEN:REC                       // Display Receiver Operation Screen.
REC:FREQ 450000                  // Set Receiver Frequency to 450 MHz.
REC:MODULATION:FM1               // Select FM1 for Receiver Modulation.
REC:INPUT:TR                     // Select T/R Connector for Receiver Input.
REC:INPUT:ATTENUATION 0          // Select 0 dB for Receiver Input
                                 // Attenuation Level.
N=1,X=0                          // Set variables initially.
WHILE X=0                        // Start WHILE loop to loop as long as
                                 // squelch is unbroken.
   REC:SQU N                     // Set squelch 0.02 lower than last setting.
   DELAY 25                      // Allow time for squelch to settle.
   N=N-0.02                      // Decrement N (use to set squelch).
   X=:MEAS:SQU?                  // Set X to 1 when squelch breaks (see 6-16).
WEND                             // End of WHILE loop.  X loops to the top as
                                 // long as X=0 (squelch unbroken).
REC:OUTPUT:AUDIO 1               // Route the demodulated signal to the
                                 // AUDIO OUT Connector.
REC:OUTPUT:SPEAKER 1             // Route Receiver Output to the Test Set
                                 // Speaker.
REC:DEC DTMF                     // Set Receiver for decoding DTMF signals.
REC:DTMF:STAT 1                  // Enable DTMF decoding.
REC:DTMF?                        // Return DTMF decoded digits.
END                              // End of macro REC_DTMF.
```

D. DECODING CCIR CODED SIGNALS

The following command sequence receives a 450 MHz FM modulated Audio signal and routes the demodulated signal to the AUDIO OUT Connector and the Test Set Speaker.

```
*DMC "REC_CCIR",BEGIN            // Define a macro named REC_CCIR.
SCREEN:REC                       // Display the Receiver Operation Screen.
REC:FREQ 450000                  // Set Receiver Frequency to 450 MHz.
REC:MODULATION:FM1               // Select FM1 for Receiver Modulation.
REC:INPUT:ANT                    // Select the ANTENNA IN Connector for the
                                 // Receiver Input.
REC:INPUT:ATTENUATION 0          // Select 0 dB for the Receiver Input
                                 // Attenuation Level.
N=1,X=0                          // Set variables initially.
WHILE X=0                        // Start WHILE loop to loop as long as
                                 // squelch is unbroken.
   REC:SQU N                     // Set squelch 0.02 lower than last setting.
   DELAY 25                      // Allow time for squelch to settle.
   N=N-0.02                      // Decrement N (use to set squelch).
   X=:MEAS:SQU?                  // Set X to 1 when squelch breaks (see 6-16).
WEND                             // End of WHILE loop.  X loops to the top as
                                 // long as X=0 (squelch unbroken).

/* (macro continues on next page) */
```

```
REC:OUTPUT:AUDIO 1              // Routes the demodulated signal to the
                                // AUDIO OUT Connector.
REC:OUTPUT:SPEAKER 1            // Routes Receiver Output to the Test Set
                                // Speaker.
REC:TONE:TYPE CCIR              // Selects CCIR for the Audio Code Type.
REC:DEC TONE                    // Sets Receiver for decoding Audio Tones.
REC:TONE:STAT 1                 // Enables Audio decoding.
REC:TONE?                       // Returns the decoded Audio Tone sequence.
REC:TONE:STAT 0                 // Disables Audio decoding.
END                             // End of macro REC_CCIR.
```

E. DECODING DCS CODED SIGNALS

The following command sequence receives a 450 MHz FM modulated DCS signal and decodes the signal until Soft Function Key F1 is pressed. The decoded DCS digits are printed to the Host.

```
*DMC "REC_DCS",BEGIN            // Define a macro named REC_DCS.
VAR CODE                        // Defines variable to hold decoded DCS.
SCREEN:REC                      // Displays the Receiver Operation Screen.
REC:FREQ 450000                 // Sets Receiver Frequency to 450 MHz.
REC:MODULATION:FM1              // Selects FM1 for Receiver Modulation.
REC:INPUT:ANT                   // Selects the ANTENNA IN Connector for the
                                // Receiver Input.
REC:INPUT:ATTENUATION 0         // Selects 0 dB for the Receiver Input
                                // Attenuation Level.
N=1,X=0                         // Set variables initially.
WHILE X=0                       // Start WHILE loop to loop as long as
                                // squelch is unbroken.
   REC:SQU N                    // Set squelch 0.02 lower than last setting.
   DELAY 25                     // Allows time for squelch to settle.
   N=N-0.02                     // Decrement N (used to set squelch).
   X=:MEAS:SQU?                 // Set X to 1 when squelch breaks (see 6-16).
WEND                            // End of WHILE loop.  X loops to the top as
                                // long as X=0 (squelch unbroken).
KEYPAD:CLAIM                    // Directs all Keyboard Input to TMAC so
                                // Operation Screen is not changed.
REC:DEC DIG                     // Sets Receiver for decoding digital data.
REC:DIG DCS                     // Prepares Receiver for decoding DCS.
REC:DCS:STAT 1                  // Enable DCS decoding.
WHILE (SYSTEM:KEY? != F1)       // Loop until F1 Key is pressed.
  TPAUSE                        // Allow TMAC to share processor time with
                                // Test Set (see 2-12).
  CODE=REC:DCS:NORM?)           // Decode a Normal DCS digits, if received.
  IF (CODE != "-1")             // If DCS digits are decoded,
    PPRINT CODE                 // print them to the Host.
  ENDIF                         // End of IF statement.
WEND                            // End of While loop.
REC:DCS:STAT 0                  // Disable DCS decoding.
KEYPAD:UNCLAIM                  // Release Keyboard for normal use.
END                             // End of macro REC_DCS.
```

F. DECODING AUDIO USER DEFINED CODED SIGNALS

The following command sequence receives a 450 MHz FM modulated Audio signal, decodes the signal using the RF Generator Audio User Defined Tones and routes the demodulated signal to the AUDIO OUT Connector and the Test Set Speaker.

```
*DMC "REC_AUDIO_USER"            // Define a macro named REC_AUDIO_USER.

                                 // DEFINE THE TONES TO BE DECODED

SCREEN:GEN                       // Displays RF Generator Operation Screen.
GEN:TONE:TYPE:USER               // Selects User Defined for the Audio Code.
GEN:TONE:USER:DEF 0,900,150      // Defines 0 Tone at 900 Hz and 150 ms.
GEN:TONE:USER:DEF 1,1100,80      // Defines 1 Tone at 1100 Hz and 80 ms.
GEN:TONE:USER:DEF 2,1200,80      // Defines 2 Tone at 1200 Hz and 80 ms.
GEN:TONE:USER:DEF 3,1300,80      // Defines 3 Tone at 1300 Hz and 80 ms.
GEN:TONE:USER:DEF 4,1400,80      // Defines 4 Tone at 1400 Hz and 80 ms.
GEN:TONE:USER:DEF A,0,40         // Defines A Tone at 0 Hz and 40 ms.
                                 //
                                 // DECODE THE DEFINED TONES
                                 //
SCREEN:REC                       // Displays the Receiver Operation Screen.
REC:FREQ 450000                  // Sets Receiver Frequency to 450 MHz.
REC:MODULATION:FM1               // Selects FM1 for Receiver Modulation.
REC:INPUT:ANT                    // Selects the ANTENNA IN Connector for the
                                 // Receiver Input.
REC:INPUT:ATTENUATION 0          // Selects 0 dB for the Receiver Input
                                 // Attenuation Level.
N=1,X=0                          // Set variables initially.
WHILE X=0                        // Start WHILE loop to loop as long as
                                 // squelch is unbroken.
   REC:SQU N                     // Set squelch 0.02 lower than last setting.
   DELAY 25                      // Allows time for squelch to settle.
   N=N-0.02                      // Decrement variable N (used to set
                                 // squelch).
   X=:MEAS:SQU?                  // Set X to 1 when squelch breaks (see 6-16).
WEND                             // End of WHILE loop.  X loops to the top as
                                 // long as X=0 (squelch unbroken).
REC:OUTPUT:AUDIO 1               // Routes the demodulated signal to the
                                 // AUDIO OUT Connector.
REC:OUTPUT:SPEAKER 1             // Routes Receiver Output to the Test Set
                                 // Speaker.
REC:TONE:TYPE USER               // Selects User Defined for the Audio Code
                                 // Type.
REC:DEC TONE                     // Sets Receiver for decoding Audio Tones.
REC:TONE:STAT 1                  // Enables Audio decoding.
REC:TONE?                        // Returns the decoded Audio Tone sequence.
REC:TONE:STAT 0                  // Disables Audio decoding.
END                              // End of macro REC_AUDIO_USER.
```

G. DECODING POCSAG CODED SIGNALS

The following command sequence receives a 450 MHz FM modulated POCSAG signal, decodes the signal and prints the POCSAG messages to the Host until Soft Function Key F1 is pressed.

```
*DMC "REC_POCSAG",BEGIN          // Define a macro named REC_POCSAG.
STRING CODE                      // Define a string variable to hold decoded
                                 // POCSAG message.
SCREEN:REC                       // Displays the Receiver Operation Screen.
REC:FREQ 450000                  // Sets Receiver Frequency to 450 MHz.
REC:MODULATION:FM1               // Selects FM1 for the Receiver Modulation.
REC:INPUT:ANT                    // Selects the ANTENNA IN Connector for the
                                 // Receiver Input.
REC:INPUT:ATTENUATION 0          // Selects 0 dB for the Receiver Input
                                 // Attenuation Level.
N=1,X=0                          // Set variables initially.
WHILE X=0                        // Start WHILE loop to loop as long as
                                 // squelch is unbroken.
   REC:SQU N                     // Set squelch 0.02 lower than last setting.
   DELAY 25                      // Allows time for squelch to settle.
   N=N-0.02                      // Decrement variable N (used to set
                                 // squelch).
   X=:MEAS:SQU?                  // Set X to 1 when squelch breaks (see 6-16).
WEND                             // End of WHILE loop. X loops to the top as
                                 // long as X=0 (squelch unbroken).
REC:OUTPUT:SPEAKER 1             // Routes Receiver Output to the Test Set
                                 // Speaker.
REC:POCSAG:RAT 1                 // Sets POCSAG rate to be decoded to High.
KEYPAD:CLAIM                     // Directs all Keyboard Input to TMAC so
                                 // Operation Screen is not changed.
REC:DEC DIG                      // Sets Receiver for decoding digital data.
REC:DIG POCSAG                   // Prepares Receiver for decoding POCSAG.
REC:POCSAG:STAT 1                // Enable POCSAG decoding.
WHILE (SYSTEM:KEY? != F1)        // Loop until F1 Key is pressed.
  TPAUSE                         // Allow TMAC to share processor time with
                                 // Test Set (see 2-12).
  CODE=STR(REC:POCSAG:MESS?)     // Decode a POCSAG message, if received.
  IF (CODE != "-1")              // If a POCSAG message was decoded,
    PPRINT CODE                  // print message to the Host.
  ENDIF                          // End of IF statement.
WEND                             // End of While loop.
REC:POCSAG:STAT 0                // Disable POCSAG decoding.
KEYPAD:UNCLAIM                   // Release Keyboard for normal use.
END                              // End of macro REC_POCSAG.
```

H. RECEIVER SETUP FOR CELLULAR OPERATION

The following command sequence configures the Receiver Mode for receiving Cellular signals through the ANTENNA IN Connector.

```
SCREEN:REC                      // Displays the Receiver Operation Screen.
REC:MOD:USER:MOD:DATA           // Selects User Defined FM Data for the
                                // Receiver Modulation.
REC:MOD:USER:FILT 30            // Selects the 30 kHz IF Filter.
REC:MOD:USER:POST:LPAS 15       // Selects Low-Pass Post Detection Filter
                                // with a cutoff frequency of 15 kHz.
REC:AGC:USER:HIGH               // Selects User Defined High Speed for the
                                // AGC setting.
REC:INPUT:ANTENNA               // Selects the Antenna for Receiver Input.
M_DEV:RANG:UPP 10               // Sets Deviation Meter Range to 10 kHz.
M_AF:RES 1                      // Sets AF Meter Gate Time to .1 sec (1 Hz).
```

6-7 DUPLEX

6-7-1 DUPLEX GENERAL COMMANDS

DUPlex:

METER:

DISTortion
[DUPlex:METER:DISTortion]
Displays Distortion Meter on Duplex Operation Screen.

MODMeter
[DUPlex:METER:MODMeter]
Displays Modulation Meter on Duplex Operation Screen.

OFF
[DUPlex:METER:OFF]
Disables Modulation, Distortion and SINAD Meters.

SINAD
[DUPlex:METER:SINAD]
Displays SINAD Meter on Duplex Operation Screen.

SPEAKer:SOURce *type*
[DUPlex:SPEAKer:SOURce type]
Selects Test Set Speaker Input. Select OFF, FGEN (Function Generator), SINAD (SINAD/BER IN Connector) or EXTMOD (EXT MOD IN Connector) for *type*. Command takes effect when RF Generator Operation Screen is updated.

STORe *n*
[DUPlex:STORe n]
Stores current Duplex environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

RCL *n*
[DUPlex:RCL n]
Recalls Duplex environment (routings and settings) stored in memory location *n*. Range of *n* is 1 to 9.

6-7-2 DUPLEX TRANSMITTER COMMANDS

DUPlex:

INPut:

AGC:

AUTO
[DUPlex:INPut:AGC:AUTO]
Sets Automatic Gain Control to automatic setting.

MANual *n*
[DUPlex:INPut:AGC:MANual n]
Sets Automatic Gain Control to manual setting and sets level to *n*. Range of *n* is 0 to 255.

USER:*XXX*
[DUPlex:INPut:AGC:USER:xxx]
Sets Automatic Gain Control to User setting. Select one of the following for *xxx*: MEASure, SPeech, DATA, HIGH, TYPE1, TYPE2 or TYPE3.

Example: `DUP:INP:AGC:USER:DATA // Sets AGC setting to User Defined Data.`

ANTenna
[DUPlex:INPut:ANTenna]
Selects ANTENNA IN Connector as Transmitter Input Source. Displays Signal Strength Meter reading on the Duplex Transmitter Operation Screen.

The following Attenuation commands control the Input Attenuation for the Duplex Transmitter and selection/deselection of LNA (Low Noise Amplifier). Use **DUPlex:INPut: ATTenuation:LNA** to select LNA and 0 dB of input attenuation. To deselect LNA and select a specific value of attenuation, use **DUPlex:INPut:ATTenuation**.

ATTenuation:LNA
[DUPlex:INPut:ATTenuation:LNA]
Sets Duplex Input Attenuation to LNA.

ATTenuation:LNA?
[DUPlex:INPut:ATTenuation:LNA?]
Returns current state of the Low Noise Amplifier. Returns 1 if LNA is selected; 0 otherwise.

LNA (Low Noise Amplifier) has the same attenuation as 0 dB (see **DUPlex:INPut: ATTenuation** *n*), but LNA has a lower noise figure. LNA is the preferred option for performing off-the-air applications.

ATTenuation *n*
[DUPlex:INPut:ATTenuation n]
Sets Duplex Input Attenuation to *n* dB. Possible values for *n*: 0, 5, 10, 15, 20, 25, 30.

ATTenuation?
[DUPlex:INPut:ATTenuation?]
Returns the current value of Input Attenuation.

DUPlex:

INPut:

CHANnel *n*
[DUPlex:INPut:CHANnel n]
Sets Duplex Transmitter Frequency to cellular channel *n* (1 to 2047, depending on format) in the format selected using the **DUPlex:INPut:CHANnel:FORMat** command. The Duplex Transmitter Operation Screen displays selected cellular channel when Duplex Transmitter is in Channel Mode (**DUPlex:INPut:MODE CHAN** command).

CHANnel:

FORMat:

AMPS:

FORward
[DUPlex:INPut:CHANnel:FORMat:AMPS:FORward]
Selects AMPS (NADC-U8) Forward channels.

REVerse
[DUPlex:INPut:CHANnel:FORMat:AMPS:REVerse]
Selects AMPS (NADC-U8) Reverse channels.

ETACS:

FORward
[DUPlex:INPut:CHANnel:FORMat:ETACS:FORward]
Selects ETACS Forward channels.

REVerse
[DUPlex:INPut:CHANnel:FORMat:ETACS:REVerse]
Selects ETACS Reverse channels.

NAMPS:

FORward
[DUPlex:INPut:CHANnel:FORMat:NAMPS:FORward]
Selects NAMPS Forward channels.

REVerse
[DUPlex:INPut:CHANnel:FORMat:NAMPS:REVerse]
Selects NAMPS Reverse channels.

BAND:*x*
[DUPlex:INPut:CHANnel:FORMat:NAMPS:BAND:x]
Selects NAMPS Narrow Analog channel designator. Range of *x* is Lower, Middle or Upper.

DUPlex:

INPut:

CHANnel:

FORMat:

NT400:

FORward
[DUPlex:INPut:CHANnel:FORMat:NT400:FORward]
Selects NT400 Forward channels.

REVerse
[DUPlex:INPut:CHANnel:FORMat:NT400:REVerse]
Selects NT400 Reverse channels.

NADC:

FORward
[DUPlex:INPut:CHANnel:FORMat:NADC:FORward]
Selects NADC Forward. Utilizes current setting of NADC band.

REVerse
[DUPlex:INPut:CHANnel:FORMat:NADC:REVerse]
Selects NADC Reverse. Utilizes current setting of NADC band.

BAND:*x*
[DUPlex:INPut:CHANnel:FORMat:NADC:BAND:x]
Selects NADC band. Range of *x* is U8, U4 or HYper.

BAND?
[DUPlex:INPut:CHANnel:BAND?]
Returns band for Duplex Transmitter Channel Format. Returns one of the following bands for the associated channel format:

FORMAT	BAND
NADC	U8, U4 or HY
ETACS	NOT AVAILABLE
NAMPS	LOWER, MIDDLE or UPPER

FORMat?
[DUPlex:INPut:CHANnel:FORMat?]
Returns Duplex Transmitter Channel Format (NADC:FORWARD, NADC:REVERSE, ETACS:FORWARD, ETACS:REVERSE, NAMPS:FORWARD or NAMPS:REVERSE).

DUPlex:

INPut:

FIND:

FREQuency?
[DUPlex:INPut:FIND:FREQuency?]
Returns frequency of first signal with amplitude larger than the Find reference level.

REFerence *n*
[DUPlex:INPut:FIND:REFerence n]
Sets Find reference level to *n* dB.

REFerence?
[DUPlex:INPut:FIND:REFerence?]
Returns Find reference level in dB.

Example:
```
DUP:INP:FIND:REF -55       // Sets Find Reference Level to -55 dBm.
DUP:INP:FIND:FREQ?         // Returns the lowest frequency (in kHz)
                           // containing a signal greater than -55 dBm.
DUP:INP:FIND:REF?          // Queries the current Find Reference Level
                           // (in dBm).  -55 is returned.
```

FREQuency *n [units]*
[DUPlex:INPut:FREQuency n [units]]
Sets Duplex Transmitter Frequency. Range of *n* is 250.0 kHz to 2010.0000 MHz.
Select HZ, KHZ, MHZ for *units*. *units* is optional with a default of KHZ.

FREQuency?
[DUPlex:INPut:FREQuency?]
Returns RF Generator Frequency in kHz.

Example:
```
DUP:INP:FREQ 145 MHZ  // Set RF Generator Frequency to 145 MHz.
DUP:INP:FREQ?         // Query the RF Generator Frequency.
                      // 145000 is returned (145000 kHz).
```

METER:

DEVRms
[DUPlex:INPut:METER:DEVRms]
Displays Deviation Meter (RMS) when followed by a **SCREEN:DUPlex** command.

DISTortion
[DUPlex:INPut:METER:DISTortion]
Displays Distortion Meter when followed by a **SCREEN:DUPlex** command.

MODMeter
[DUPlex:INPut:METER:MODMeter]
Displays Modulation Meter when followed by a **SCREEN:DUPlex** command.

PMRms
[DUPlex:INPut:METER:PMRms]
Displays Phase Meter (RMS) when followed by a **SCREEN:DUPlex** command.

DUPlex:

INPut:

METER:

SINAD
[DUPlex:INPut:METER:SINAD]
Displays SINAD Meter when followed by a **SCREEN:DUPlex** command.

MODE *type*
[DUPlex:INPut:MODE type]
Selects the Duplex Transmitter Mode. Channel Mode displays cellular channel frequency according to **DUP:INP:CHAN** commands.

MODE	*type*
Direct	DIRect
Channel	CHANnel
Frequency Scan	SCAN
Frequency List	LIST
Frequency List Scan	FLScan

MODE?
[DUPlex:INPut:MODE?]
Returns current mode (DIRECT, CHANNEL, FREQUENCY SCAN, FREQUENCY LIST or FREQUENCY LIST SCAN).

MODulation:

AM*n*
[DUPlex:INPut:MODulation:AMn]
Selects Amplitude Modulation. Select 1 or 2 for *n*.

Example:
```
DUP:INP:MOD:AM1   // Selects AM1 for Duplex Transmitter
                  // Modulation.
```

BFO
[DUPlex:INPut:MODulation:BFO]
Selects Beat Frequency Oscillation.

FM*n*
[DUPlex:INPut:MODulation:FMn]
Selects Frequency Modulation. Select 1, 2, 3 or 4 for *n*.

PM
[DUPlex:INPut:MODulation:PM]
Selects Phase Modulation.

DUPlex:

INPut:

MODulation:

USER:

FILTer *f*
[DUPlex:INPut:MODulation:USER:FILTer f]
Sets User Defined Modulation IF Filter to *f* kHz. Select 3, 30 or 300 for *f*.

MODulation:*type*
[DUPlex:INPut:MODulation:USER:MODulation:type]
Selects User Defined Modulation. Enter one of the following for *type*: FM, AM, BFO, PM or DATA (FM DATA).

POST:

APASs
[DUPlex:INPut:MODulation:USER:POST:APASs]
Selects All Pass Post Detection Filter for User Defined Modulation.

BPASs *fl,fh*
[DUPlex:INPut:MODulation:USER:POST:BPASs fl,fh]
Selects Bandpass Post Detection Filter for User Defined Modulation. Lower cutoff frequency is set to *fl* kHz with range of 0.5 to 20. Higher cutoff frequency is set to *fh* kHz with a range of 0.1 to 30.

CWEight
[DUPlex:INPut:MODulation:USER:POST:CWEight]
Selects C-Weighted Post Detection Filter (C-message noise weighting curve response) for the User Defined Modulation. Refer to MIL-STD-188-200.

HPASs *fl*
[DUPlex:INPut:MODulation:USER:POST:HPASs fl]
Selects High-Pass Post Detection Filter for User Defined Modulation. Cutoff frequency is set to *fl* kHz with a range of 0.5 to 20.0.

LPASs *fh*
[DUPlex:INPut:MODulation:USER:POST:LPASs fh]
Selects Low-Pass Post Detection Filter for User Defined Modulation. Cutoff frequency is set to *fh* kHz with a range of 0.1 to 30.0.

DUPlex:

INPut:

MODulation?

[DUPlex:INPut:MODulation?]

Returns the current Duplex Transmitter Modulation (FM, AM, BFO, PM, DATA or USER).

Example:

```
DUP:INP:MOD:USER:FILT 3        // Selects the 3 kHz IF Filter.
DUP:INP:MOD:USER:FM            // Selects User Defined FM for the
                               // Duplex Transmitter Modulation.
DUP:INP:MOD:USER:POST:LPAS 15  // Selects a Low-Pass Post Detection
                               // Filter with 15 kHz cutoff
                               // frequency.
DUP:INP:MOD:USER:MOD?          // Queries the Duplex Transmitter
                               // Modulation.  User is returned.
```

SCAN:

Except where otherwise noted, the following commands are for Duplex Transmitter Frequency Scan or Frequency List Scan modes.

ABORt

[DUPlex:INPut:SCAN:ABORt]

Stops frequency scan or frequency list scan depending on present mode.

CONTinue

[DUPlex:INPut:SCAN:CONTinue]

Starts frequency scan or frequency list scan depending on present mode.

FREQuency?

[DUPlex:INPut:SCAN:FREQuency?]

Returns current frequency in kHz being scanned.

INCrement *n*

[DUPlex:INPut:SCAN:INCrement n]

(Frequency Scan mode only.) Specifies increment in kHz between frequencies to be scanned. Range of *n* is 0.0 to 2010250.0 kHz.

PAUse *n*

[DUPlex:INPut:SCAN:PAUse n]

(Frequency Scan mode only.) Pause Time. Specifies time period in seconds for Duplex Transmitter to sit on a frequency if squelch is broken. Range of *n* is 0.0 to 99.9.

If 0.0 is selected, the Duplex Transmitter sits on current frequency as long as squelch is broken.

PAUse?

[DUPlex:INPut:SCAN:PAUse?]

(Frequency Scan mode only.) Returns a 1 if scanning is paused (stopped) or 0 if currently scanning.

DUPlex:

INPut:

SCAN:

RATe *n*

[DUPlex:INPut:SCAN:RATe n]

(Frequency Scan mode only) Scan Rate. Specifes time period in seconds for Duplex Transmitter to sit on a frequency unless squelch is broken. Range of *n* is 0.02 to 99.99.

STARt *f*

[DUPlex:INPut:SCAN:STARt f]

(Frequency Scan mode only) Starting Frequency value in kHz. Specifies lower limit frequency for scanning. Range of *f* is 250.0 to 2010000.0.

Start frequency must be a valid value and less than Stop frequency for Scan to operate.

STOP *f*

[DUPlex:INPut:SCAN:STOP f]

(Frequency Scan mode only) Stop Frequency value in kHz. Specifies upper limit frequency for scanning. Range of *f* is 250.0 to 2010000.0.

FREQList:

The following commands are for Duplex Transmitter Frequency List Scan mode only.

PAUSe *n*

[DUPlex:INPut:SCAN:FREQList:PAUSe n]

Frequency List Scan Pause Time. Specifies time period in seconds for Duplex Transmitter to sit on a frequency if squelch is broken. Range of *n* is 0.0 to 99.9.

If 0.0 is specified, Duplex Transmitter sits on frequency as long as squelch is broken.

PAUSe?

[DUPlex:INPut:SCAN:FREQList:PAUSe?]

Returns current value of Frequency List Scan Pause Time.

RATe *n*

[DUPlex:INPut:SCAN:FREQList:RATe n]

Frequency List Scan Rate. Specifies time period in seconds for Duplex Transmitter to sit on a frequency unless squelch is broken. Range of *n* is 0.02 to 99.99.

RATe?

[DUPlex:INPut:SCAN:FREQList:RATe?]

Returns current value of Frequency List Scan Rate.

DUPlex:

INPut:

SCAN:

FREQList:

SQUelch *b*
[DUPlex:INPut:SCAN:FREQList:SQUelch b]
Enables (*b* = 1) or disables (*b* = 0) Squelch.

SQUelch?
[DUPlex:INPut:SCAN:FREQList:SQUelch?]
Returns current state of Squelch.

TO:

AUDio *b*
[DUPlex:INPut:TO:AUDio b]
Routes demodulated Receiver Input to the AUDIO OUT Connector if *b* is 1. Disconnects the AUDIO OUT Connector if *b* is 0.

DEMOD *b*
[DUPlex:INPut:TO:DEMOD b]
Routes demodulated Receiver Input to the DEMOD OUT Connector if *b* is 1. Disconnects the DEMOD OUT Connector if *b* is 0.

SPEAKer *b*
[DUPlex:INPut:TO:SPEAKer b]
Routes demodulated Receiver Input to the Test Set Speaker if *b* is 1. Disconnects the Test Set Speaker if *b* is 0.

TR
[DUPlex:INPut:TR]
Selects T/R Connector as Duplex Transmitter Input Source. Displays Power Meter on the Duplex Transmitter Operation Screen.

VOLume:

AUTO *b*
[DUPlex:INPut:VOLume:AUTO b]
Enables Automatic Volume Control if *b* is 1, disables if *b* is 0.

AUTO?
[DUPlex:INPut:VOLume:AUTO?]
Returns Automatic Volume Control state. 1 is returned if enabled, 0 if disabled.

6-7-3 DUPLEX RECEIVER COMMANDS

DUPlex:

OUTput:

AUDio *b*
[DUPlex:OUTput:AUDio b]
Routes AF Generator Output to the AUDIO OUT Connector if *b* is 1. Disconnects the AUDIO OUT Connector if *b* is 0.

CHANnel *n*
[DUPlex:OUTput:CHANnel n]
Sets Duplex Receiver Frequency to cellular channel *n* (1 to 2047, depending on format) in the format selected using the **DUPlex:OUTut:CHANnel:FORMat** command. The Duplex Receiver Operation Screen displays selected cellular channel when Duplex Receiver is in Channel Mode (**DUPlex:OUTput:MODE CHAN** command).

CHANnel:

FORMat:

AMPS:

FORward
[DUPlex:OUTput:CHANnel:FORMat:AMPS:FORward]
Selects AMPS (NADC-U8) Forward channels.

REVerse
[DUPlex:OUTput:CHANnel:FORMat:AMPS:REVerse]
Selects AMPS (NADC-U8) Reverse channels.

ETACS:

FORward
[DUPlex:OUTput:CHANnel:FORMat:ETACS:FORward]
Selects ETACS Forward channels.

REVerse
[DUPlex:OUTput:CHANnel:FORMat:ETACS:REVerse]
Selects ETACS Reverse channels.

NADC:

FORward
[DUPlex:OUTput:CHANnel:FORMat:NADC:FORward]
Selects NADC Forward. Utilizes current setting of NADC band.

REVerse
[DUPlex:OUTput:CHANnel:FORMat:NADC:REVerse]
Selects NADC Reverse. Utilizes current setting of NADC band.

BAND:*x*
[DUPlex:OUTput:CHANnel:FORMat:NADC:BAND:x]
Selects NADC band. Range of *x* is U8, U4 or HYper.

DUPlex:

OUTput:

CHANnel:

FORMat:

NAMPS:

FORward
[DUPlex:OUTput:CHANnel:FORMat:NAMPS:FORward]
Selects NAMPS Forward channels.

REVerse
[DUPlex:OUTput:CHANnel:FORMat:NAMPS:REVerse]
Selects NAMPS Reverse channels.

BAND:*x*
[DUPlex:OUTput:CHANnel:FORMat:NAMPS:BAND:x]
Selects NAMPS Narrow Analog channel designator. Range of *x* is Lower, Middle or Upper.

NT400:

FORward
[DUPlex:OUTput:CHANnel:FORMat:NT400:FORward]
Selects NT400 Forward channels.

REVerse
[DUPlex:OUTput:CHANnel:FORMat:NT400:REVerse]
Selects NT400 Reverse channels.

BAND?
[DUPlex:OUTput:CHANnel:BAND?]
Returns band for Duplex Receiver Channel Format. Returns one of the following bands for the associated channel format:

FORMAT	BAND
NADC	U8, U4 or HY
ETACS	NOT AVAILABLE
NAMPS	LOWER, MIDDLE or UPPER

FORMat?
[DUPlex:OUTput:CHANnel:FORMat?]
Returns Duplex Receiver Channel Format (NADC:FORWARD, NADC:REVERSE, ETACS:FORWARD, ETACS:REVERSE, NAMPS:FORWARD or NAMPS:REVERSE).

DEMOD *b*
[DUPlex:OUTput:DEMOD b]
Routes AF Generator Output to the DEMOD OUT Connector if *b* is 1. Disconnects the DEMOD OUT Connector if *b* is 0.

DUPlex:

OUTput:

DUPlex

[DUPlex:OUTput:DUPlex]

Routes RF Generator Output to the DUPLEX OUT Connector and disconnects the T/R Connector.

FREQuency *n* [*units*]

[DUPlex:OUTput:FREQuency n [units]]

Sets Duplex Receiver Frequency. Range of *n* is 250.0 kHz to 2010.0000 MHz. Select HZ, KHZ, MHZ for *units. units* is optional with a default of KHZ.

FREQuency?

[DUPlex:OUTput:FREQuency?]

Returns Duplex Receiver Frequency in kHz.

Example:

```
DUP:OUT:FREQ 900     // Set RF Generator Frequency to 900 kHz.
DUP:OUT:FREQ?        // Query the RF Generator Frequency.
                     // 900 is returned (900 kHz).
```

LEVel:

DBm *n*

[DUPlex:OUTput:LEVel:DBm n]

Sets RF Level to *n* dBm. Range of *n* is 0.0 to -137.0

DBm?

[DUPlex:OUTput:LEVel:DBm?]

Returns RF Level in dBm.

Example:

```
DUP:OUT:LEV:DB -40    // Sets Duplex Receiver Output Level to
                      // -40 dBm.
DUP:OUT:DB?           // Queries Duplex Receiver Output Level.
                      // -40 is returned.
```

METER:

AF

[DUPlex:OUTput:METER:AF]

Displays AF Meter on the Duplex Receiver Operation Screen.

DISTortion

[DUPlex:OUTput:METER:DISTortion]

Displays Distortion Meter on the Duplex Receiver Operation Screen.

DMM

[DUPlex:OUTput:METER:DMM]

Displays Digital Multimeter on the Duplex Receiver Operation Screen.

SINAD

[DUPlex:OUTput:METER:SINAD]

Displays SINAD Meter on the Duplex Receiver Operation Screen.

DUPlex:

OUTput:

MODE *type*

[DUPlex:OUTput:MODE type]

Selects the Duplex Receiver Mode. Channel Mode displays cellular channel frequency according to **DUP:OUT:CHAN** commands.

MODE	*type*
Direct	DIRect
Channel	CHANnel
Frequency Scan	SCAN
Frequency List	LIST
Frequency List Scan	FLScan

MODE?

[DUPlex:OUTput:MODE?]

Returns current mode (DIRECT, CHANNEL, FREQUENCY SCAN, FREQUENCY LIST or FREQUENCY LIST SCAN).

OFFSet *f*

[DUPlex:OUTput:OFFSet f]

Sets Offset Frequency to *f* kHz. Range of *f* is -999749.9 to 999749.9.

OFFSet?

[DUPlex:OUTput:OFFSet?]

Returns Offset Frequency in kHz.

SCAN:

Except where noted, the following commands are for Duplex Receiver Frequency Scan or Frequency List Scan modes.

ABORt

[DUPlex:OUTput:SCAN:ABORt]

Stops frequency scan or frequency list scan (depending on present mode).

CONTinue

[DUPlex:OUTput:SCAN:CONTinue]

Starts frequency scan or frequency list scan (depending on present mode).

FREQuency?

[DUPlex:OUTput:SCAN:FREQuency?]

Returns current frequency in kHz being generated.

INCrement *n*

[DUPlex:OUTput:SCAN:INCrement n]

(Frequency Scan mode only.) Specifies increment in kHz between frequencies to be generated. Range of *n* is 0.0 to 2010000.0.

DUPlex:

OUTput:

SCAN:

PAUse?
[DUPlex:OUTput:SCAN:PAUse?]
(Frequency Scan mode only) Returns a 1 if scanning is paused (stopped) or 0 if currently scanning.

RATe *n*
[DUPlex:OUTput:SCAN:RATe n]
(Frequency Scan mode only) Scan Rate. Specifes time period in seconds for each generated frequency. Range of *n* is 0.02 to 99.99.

STARt *f*
[DUPlex:OUTput:SCAN:STARt f]
(Frequency Scan mode only) Start Frequency. Specifies lower limit frequency in kHz for first generated frequency. Range of *f* is 250.0 to 2010000.0.

STOP *f*
[DUPlex:OUTput:SCAN:STOP f]
(Frequency Scan mode only) Stop Frequency. Specifies upper limit frequency in kHz or last generated frequency. Range of *f* is 250.0 to 2010000.0.

FREQList:

The following applies to Duplex Receiver Frequency List Scan mode only.

RATe *n*
[DUPlex:OUTput:SCAN:FREQList:RATe n]
Frequency List Scan Rate. Specifies time period in seconds for Duplex Receiver to generate the current frequency prior to hopping to next frequency. Range of *n* is 0.02 to 99.99.

TR
[DUPlex:OUTput:TR]
Routes Duplex Receiver Output to the T/R Connector and disconnects DUPLEX OUT Connector.

6-7-4 REMOTE DUPLEX EXAMPLES

A. GENERATING AND RECEIVING FM SIGNALS

The following sequence of commands generates a 160 MHz FM modulated signal routed to the DUPLEX OUT Connector and receives a 165 MHz FM modulated signal through the ANTENNA IN Connector:

```
SCREEN:DUPLEX              // Displays the Duplex Operation Screen
DUP:OUT:FREQ 160000        // Sets 160 MHz Duplex Receiver Frequency.
DUP:OUT:DUP                // Routes the DUPLEX OUT Connector for
                           // the Duplex Receiver Output.
DUP:OUT:LEV:DB -30         // Sets Duplex Receiver Output Level to
                           // -30 dBm.
FGEN:GEN1:STATE 1          // Activates SOURCE 1 (AF Generator 1).
FGEN:GEN1:FREQ 2000        // Sets SOURCE 1 AF frequency to 2000 Hz.
FGEN:GEN1:MOD:FM           // Sets SOURCE 1 Modulation to FM.
FGEN:GEN1:MODL 5           // Sets SOURCE 1 Modulation Level to 5 kHz.
FGEN:GEN1:SHAPE:SIN        // Sets SOURCE 1 Wave Shape to a sine wave.
DUP:INP:FREQ 165000        // Sets Duplex Transmitter Frequency to
                           // 165 MHz.
DUP:INP:ANT                // Selects ANTENNA IN Connector for Duplex
                           // Transmitter Input.
DUP:INP:ATT 0              // Sets Duplex Transmitter Input Attenuation
                           // Level to 0 dB.
DUP:INP:MOD:FM1            // Selects FM1 for the Duplex Transmitter
                           // Modulation type.
```

B. GENERATING AND RECEIVING AM SIGNALS

The following sequence of commands generates a 1.4 MHz AM signal (50% modulation) modulated with Input from the MIC/ACC IN/OUT Connector and receives a 1 MHz AM modulated signal through the ANTENNA IN Connector:

```
SCREEN:DUPLEX              // Displays the Duplex Operation Screen
DUP:OUT:FREQ 140000        // Sets 140 MHz Duplex Receiver Frequency.
DUP:OUT:DUP                // Selects the DUPLEX OUT Connector for
                           // the Duplex Receiver Output.
DUP:OUT:LEV:DB -30         // Sets Duplex Receiver Output Level to
                           // -30 dBm.
FGEN:MIC:STATE 1           // Activates SOURCE 1 (AF Generator 1).
FGEN:MIC:MOD:AM1           // Sets SOURCE 1 Modulation to AM1.
FGEN:MIC:MODL 50           // Sets SOURCE 1 Modulation Level to 50%.
PTT:STATE 1                // Enables MIC/ACC Connector.
DUP:INP:FREQ 1 MHZ         // Sets Duplex Transmitter Frequency to
                           // 1 MHz.
DUP:INP:ANT                // Selects ANTENNA IN Connector for Duplex
                           // Transmitter Input.
DUP:INP:ATT 0              // Sets Duplex Transmitter Input Attenuation
                           // Level to 0 dB.
DUP:INP:MOD:AM1            // Selects AM1 for the Duplex Transmitter
                           // Modulation type.
```

6-8 AF GENERATOR

6-8-1 AF GENERATOR COMMANDS

FGEN:

BOOST *b*
[FGEN:BOOST b]
Enables (*b* = 1) or disables (*b* = 0) Generator #1 Boost.

Synchronizes Generator #2 with Generator #1 to allow AUDIO output level to reach 4.0 Vrms.

BOOST?
[FGEN:BOOST?]
Returns current state of Generator #1 Boost.

DATA:
(Data Generator is used for BER Meter tests.)

LEVel *n*
[FGEN:DATA:LEVel n]
Specifies audio level for BER Data Generator. Range of *n* is 0 to 4095.

0 corresponds to 0 volts, and 4095 corresponds to 5 volts.

LEVel?
[FGEN:DATA:LEVel?]
Returns current audio level for BER Data Generator.

MODulation:*type*
[FGEN:DATA:MODulation:type]
Sets Data Generator Modulation to *type*. Select OFF, AM, FM or PM.

MODulation?
[FGEN:DATA:MODulation?]
Returns Data Generator Modulation.

MODL *n*
[FGEN:DATA:MODL n]
Sets Data Generator to *n* Modulation level. Range of *n* for AM is 0.0 to 100.0 (%), FM is 0.0 to 25.0 (kHz) and PM is 0.0 to 10.0 (radians).

MODL?
[FGEN:DATA:MODL?]
Returns Data Modulation level setting.

FGEN:

DATA:

STATe *b*

[FGEN:DATA:STATe b]

Sets Data Modulation on if *b* is 1 or off if *b* is 0.

Example:

```
FGEN:DATA:MOD:FM       // Sets Data Generator Modulation to FM.
FGEN:DATA:MODL 4       // Sets Data Generator Modulation Level to
                       // 4 kHz.
FGEN:DATA:STAT 1       // Enables Data Generator.
FGEN:DATA:MOD?         // Returns Data Generator Modulation (FM).
FGEN:DATA:MODL?        // Returns Data Generator Modulation Level.
                       // 4 (kHz) is returned.
```

EXT:

(External modulation is input through EXT MOD IN Connector.)

LEVel *n*

[FGEN:EXT:LEVel n]

Sets External Modulation Proportional output level to *n*%. Range of *n* is 0 to 100. Sources are adjusted so that proportional output levels of sources combine to equal Function Generator Output Level.

Functions only if Proportional Mode is On.

LEVel?

[FGEN:EXT:LEVel?]

Returns External Modulation Proportional output level setting in %.

MODL *n*

[FGEN:EXT:MODL n]

Sets External Modulation level to *n*. Range of *n* for AM is 0 to 100 (%), FM is 0.0 to 10.0 (kHz) and PM is 0.0 to 10.0 (radians).

MODL?

[FGEN:EXT:MODL?]

Returns External Modulation level setting.

MODulation:*type*

[FGEN:EXT:MODulation:type]

Sets External Modulation to *type*. Select OFF, AM, FM or PM.

Setting an AF Generator to FM and the other AF Generator to PM, simultaneously, may cause ambiguous readings.

MODulation?

[FGEN:EXT:MODulation?]

Returns External Modulation type (OFF, AM, FM or PM).

FGEN:

EXT:

STATe *b*

[FGEN:EXT:STATe b]

Sets External Modulation on if *b* is 1 or off if *b* is 0.

Example:

```
FGEN:EXT:MOD:AM       // Sets External (EXT MOD) Modulation Type
                      // to AM.
FGEN:EXT:MODL 50      // Sets External Modulation Level to 50%.
FGEN:PROP 1           // Selects Proportional Mode for the AF
                      // Generator.
FGEN:EXT:LEV 30       // Sets External Output Level to 30%.
FGEN:EXT:STAT 1       // Enables External (EXT MOD) Modulation.
FGEN:EXT:MOD?         // Queries External (EXT MOD) Modulation
                      // Type.  AM is returned.
FGEN:EXT:MODL?        // Queries External (EXT MOD) Modulation
                      // Level.  50% is returned.
FGEN:EXT:LEV?         // Queries External (EXT MOD) Output Level.
                      // 30 is returned.
```

FSK *x*

[FGEN:FSK x]

Shifts output (See **FGEN:OUTput:**) between Generator #1 and Generator #2. *x* = 1 selects Generator #1, *x* = 0 selects Generator #2.

Commands for Generator 1 and Generator 2 are identical and are listed as **FGEN:GEN***x*. Parameter *x* is specified as 1 or 2 when the command is entered.

GEN*x*:

FREQuency *f*
[FGEN:GENx:FREQuency f]
Sets AF Generator to *f* Hz. Range of *f* is 0.0 to 40000.0.

FREQuency?
[FGEN:GENx:FREQuency?]
Returns AF Generator in Hz.

LEVel *n*
[FGEN:GENx:LEVel n]
Sets Proportional output level of AF Generator to *n*%. Range of *n* is 0 to 100. Sources are adjusted so that proportional output levels of sources combine to equal Function Generator Output Level.

Functions only if Proportional Mode is On.

LEVel?
[FGEN:GENx:LEVel?]
Returns Proportional output level setting of AF Generator in %.

MODL *n*
[FGEN:GENx:MODL n]
Sets Modulation level to *n*. Range of *n* for AM is 0 to 100 (%), FM is 0.0 to 100.0 (kHz) and PM is 0.0 to 10.0 (radians).

MODL?
[FGEN:GENx:MODL?]
Returns AF Generator Modulation level setting.

MODulation:*type*
[FGEN:GENx:MODulation:type]
Sets AF Generator Modulation to *type*. Select OFF, AM, FM or PM for *type*.

Setting an AF Generator to FM and the other AF Generator to PM, simultaneously, may cause ambiguous readings.

MODulation?
[FGEN:GENx:MODulation?]
Returns AF Generator Modulation Type.

SHAPE:*type*
[FGEN:GENx:SHAPE:type]
Sets AF Generator wave shape to *type*. Select from SIN (Sine), SQU (Square Wave), RAMP or TRI (Triangle).

SHAPE?
[FGEN:GENx:SHAPE?]
Returns AF Generator wave shape.

FGEN:

GENx:

SHAPE:

DC *n*
[FGEN:GENx:SHAPE:DC n]
Sets wave shape to DC at *n* level. Select -1, 0 or 1 for *n*.

PULse:DCYCLe 50
[FGEN:GENx:SHAPE:PULse:DCYCLe 50]
Sets AF Generator wave shape to Pulse with 50% duty cycle.

STATe *b*
[FGEN:GENx:STATe b]
AF Generator is enabled if *b* is 1, disabled if *b* is 0.

Example:

```
                     // AF Generator setup as RF Generator
                     // Modulation Source (AF Generator 1 and 2).
FGEN:GEN1:FREQ 1000  // Sets AF Generator 1 frequency to 1000 Hz.
FGEN:GEN1:MOD:FM     // Selects FM for AF Generator 1 Modulation
                     // Type.
FGEN:GEN1:MODL 5     // Sets AF Generator 1 Modulation Level to
                     // 5 kHz.
FGEN:GEN1:SHAPE:SIN  // Selects a sine wave for AF Generator 1
                     // Wave Form.
FGEN:GEN1:STAT 1     // Enables AF Generator 1.
FGEN:GEN2:FREQ 2500  // Sets AF Generator 2 frequency to 2500 Hz.
FGEN:GEN2:MOD:AM     // Selects AM for AF Generator 2 Modulation
                     // Type.
FGEN:GEN2:MODL 25    // Sets AF Generator 2 Modulation Level to
                     // 25%.
FGEN:GEN2:SHAPE:SIN  // Selects a sine wave for AF Generator 2
                     // Wave Form.
FGEN:GEN2:STAT 1     // Enables AF Generator 2.
```

Example:

```
                     // AF Generator setup as AF Signal Generator
FGEN:GEN1:FREQ 1000  // Sets AF Generator 1 frequency to 1000 Hz.
FGEN:OUT:PROP 1      // Enables AF Generator Proportional Mode.
FGEN:GEN1:LEV 50     // Sets AF Generator 1 Proportion Level to
                     // 50%.
FGEN:GEN1:SHAPE:SIN  // Selects a sine wave for AF Generator 1
                     // Wave Form.
FGEN:GEN1:STAT 1     // Enables AF Generator 1.
FGEN:GEN2:FREQ 2500  // Sets AF Generator 2 frequency to 2500 Hz.
FGEN:GEN2:LEV 100    // Sets AF Generator 2 Proportion Level to
                     // 100%.
FGEN:GEN2:SHAPE:SIN  // Selects a sine wave for AF Generator 2
                     // Wave Form.
FGEN:GEN2:STAT 1     // Enables AF Generator 2.
```

FGEN:

GEN3:

DIGital *type*
[FGEN:GEN3:DIGital type]
Sets digital *type*. Used with the **FGEN:ENCode DIGital** command to prepare Generator 3 encoding. The setting for *type* is DCS, DCSINV, POCSAG, DSAT or DST.

ENCode *type*
[FGEN:GEN3:ENCode type]
Selects a signalling format to encode. Select DTMF, TONE, DIGital or RCC for *type*. Must be followed with a **SETUP:GEN** or **SCREEN:GEN** command.

MODL *n*
[FGEN:GEN3:MODL n]
Sets AF Generator 3 Modulation level. Range of *n* for AM is 0 to 100.0 (%), PM is 0.0 to 10.0 (radians), FM is 0.0 to 100.0 (kHz)-Tone or RCC signalling format, FM is 0.0 to 10.0 (kHz)-DTMF signalling format or FM is 0.0 to 25.0 (kHz)-Digital signalling format.

MODL?
[FGEN:GEN3:MODL?]
Returns AF Generator 3 Modulation level setting.

MODulation:*type*
[FGEN:GEN3:MODulation:type]
Sets AF Generator 3 Modulation to *type*. Select OFF, AM, FM or PM.

Setting an AF Generator to FM and the other AF Generator to PM, simultaneously, may cause ambiguous readings.

MODulation?
[FGEN:GEN3:MODulation?]
Returns AF Generator 3 Modulation type.

Example:
```
FGEN:GEN3:MOD:FM // Selects FM for AF Generator 3 Modulation type.
FGEN:GEN3:MODL 4 // Sets AF Generator 3 Modulation Level to 4 kHz.
```

RCC *format*
[FGEN:GEN3:RCC format]
Sets Generator RCC signaling format. Select IMTS, MTS, SYS2805 or TREMote for *format*.

MIC:
(External audio modulation is input through MIC/ACC Connector.)

LEVel *n*
[FGEN:MIC:LEVel n]
Sets Proportional output level of MIC/ACC Connector input to *n*%. Range of *n* is 0 to 100. Sources are adjusted so that proportional output levels of sources combine to equal Function Generator Output Level. Functions only if Proportional Mode is On.

LEVel?
[FGEN:MIC:LEVel?]
Returns Proportional output level setting for MIC/ACC Connector input.

FGEN:

MIC:

MODL *n*

[FGEN:MIC:MODL n]

Sets MIC/ACC Connector input Modulation level to *n*. Range of *n* for AM is 0 to 100 (%), FM is 0.0 to 25.0 (kHz) and PM is 0.0 to 10.0 (radians).

MODL?

[FGEN:MIC:MODL?]

Returns MIC/ACC Connector input Modulation level setting.

MODulation:*type*

[FGEN:MIC:MODulation:type]

Sets MIC/ACC Connector input Modulation to *type*. Select OFF, AM, FM or PM.

Setting an AF Generator to FM and the other AF Generator to PM, simultaneously, may cause ambiguous readings.

MODulation?

[FGEN:MIC:MODulation?]

Returns MIC/ACC Connector input Modulation type.

STATe *b*

[FGEN:MIC:STATe b]

Enables external modulation through MIC/ACC Connector if *b* is 1, disables if *b* is 0.

PTT:STATe *b*

[PTT:STATe b]

Sets the push to talk pin on the MIC/ACC Connector high (keys the mic) if *b* is 1, sets the push to talk pin low if *b* is 0.

Example:

```
FGEN:MIC:MOD:AM       // Sets External (MIC/ACC) Modulation Type
                      // to AM.
FGEN:MIC:MODL 25      // Sets External (MIC/ACC) Modulation Level
                      // to 25%.
FGEN:OUT:PROP 1       // Selects Proportional Mode for the AF
                      // Generator.
FGEN:MIC:LEV 50       // Sets External (MIC/ACC) Output Level
                      // to 50%
FGEN:MIC:STAT 1       // Enables External (MIC/ACC) Modulation.
FGEN:MIC:MOD?         // Queries External (MIC/ACC) Modulation
                      // Type.  AM is returned.
FGEN:MIC:MODL?        // Queries External (MIC/ACC) Modulation
                      // Level.  25% is returned.
FGEN:MIC:LEV?         // Queries External (MIC/ACC) Output Level.
                      // 50 is returned.
```

FGEN:

OUTput:

AUDio *b*
[FGEN:OUTput:AUDio b]

Routes AF Generator Output to the AUDIO OUT Connector if *b* is 1, disconnects AUDIO OUT Connector from Output if *b* is 0.

AUDio?
[FGEN:OUTput:AUDio?]

Returns 1 if AF Generator Output is routed to the AUDIO OUT Connector, 0 if AUDIO OUT Connector is disconnected.

DEMOD *b*
[FGEN:OUTput:DEMOD b]

Routes AF Generator Output to DEMOD OUT Connector if *b* is 1, disconnects DEMOD OUT Connector from Output if *b* is 0.

DEMOD?
[FGEN:OUTput:DEMOD?]

Returns 1 if AF Generator Output is routed to DEMOD OUT Connector, 0 if DEMOD OUT Connector is disconnected.

LEVel *v*
[FGEN:OUTput:LEVel v]

Sets AF Generator output level to *v* volts. Range for *v* is 0.0000 to 3.1000.

LEVel?
[FGEN:OUTput:LEVel?]

Returns AF Generator output level in volts.

SPEAKer *b*
[FGEN:OUTput:SPEAKer b]

Routes AF Generator to Speaker if *b* is 1, disconnects Speaker from Output if *b* is 0.

SPEAKer?
[FGEN:OUTput:SPEAKer?]

Returns 1 if AF Generator Output is routed to Speaker, 0 if Speaker is disconnected.

FGEN:

PROPortional *b*

[FGEN:PROPortional b]

Selects Proportional mode for AF Generator if *b* is 1. Takes AF Generator out of Proportional mode if *b* is 0.

PROPortional?

[FGEN:PROPortional?]

Returns 1 if AF Generator is in Proportional mode; 0 otherwise.

Example:

```
FGEN:GEN1:FREQ 500    // Sets AF Generator 1 frequency to 500 Hz.
FGEN:OUT:PROP 1       // Enables Proportional Mode of AF Generator.
FGEN:GEN1:LEV 100     // Sets AF Generator 1 Proportion Level to
                      // 100%.
FGEN:GEN1:SHAPE:RAMP  // Selects ramp for the AF Generator 1
                      // Wave Form.
FGEN:GEN1:STAT 1      // Enables AF Generator 1.
FGEN:GEN2:FREQ 2000   // Sets AF Generator 2 frequency to 2000 Hz.
FGEN:GEN2:LEV 100     // Sets AF Generator 2 Proportion Level to
                      // 100%.
FGEN:GEN2:SHAPE:SIN   // Selects a sine wave for AF Generator 2
                      // Wave Form.
FGEN:GEN2:STAT 1      // Enables AF Generator 2.
FGEN:OUT:LEV 1.5      // Sets AF Generator Output Level to 1.5 V.
FGEN:OUT:AUDIO 1      // Routes AF Generator Output to AUDIO OUT
                      // Connector.
```

RCL *n*

[FGEN:RCL n]

Recalls AF Generator environment (routings and settings) stored in memory location *n*. Range of *n* is 1 to 9.

STORe *n*

[FGEN:STORe n]

Stores current AF Generator environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

6-8-2 REMOTE AF GENERATOR EXAMPLES

The following command sequence generates a 1 kHz sine wave, with 1 V level, and routes the Output to the DEMOD OUT Connector:

```
SCREEN:FUNC                 // Displays the AF Generator Operation Screen.
FGEN:GEN1:STATE 1           // Enables AF Generator 1.
FGEN:GEN1:FREQ 1000         // Sets AF Generator 1 frequency to 1000 Hz.
FGEN:GEN1:SHAPE:SIN         // Selects a sine wave for the AF Generator 1
                            // wave form.
FGEN:LEVEL 1                // Sets AF Generator 1 Output Level to 1 V.
FGEN:OUTPUT:DEMOD 1         // Routes AF Generator Output to the DEMOD OUT
                            // Connector.
```

The following command sequence generates the sum of a 1 kHz and 2.5 kHz sine wave in equal proportions and routes this Output to the AUDIO OUT Connector:

```
SCREEN:FUNC                 // Displays the AF Generator Operation Screen.
FGEN:PROPORTIONAL 1         // Selects Proportional Mode for the AF Generator.
FGEN:OUTPUT:LEVEL 1         // Sets AF Generator 1 Output Level to 1 V.
FGEN:GEN1:STATE 1           // Enables AF Generator 1.
FGEN:GEN1:FREQ 1000         // Sets AF Generator 1 frequency to 1000 Hz.
FGEN:GEN1:SHAPE:SIN         // Selects a sine wave for the AF Generator 1
                            // wave form.
FGEN:GEN1:LEVEL 100         // Sets AF Generator 1 Proportional Level to 100%.
FGEN:GEN2:STATE 1           // Enables AF Generator 2.
FGEN:GEN2:FREQ 2500         // Sets AF Generator 2 frequency to 2500 Hz.
FGEN:GEN2:SHAPE:SIN         // Selects a sine wave for the AF Generator 2
                            // wave form.
GEN:GEN2:LEVEL 100          // Sets AF Generator 2 Proportional Level to 100%.
FGEN:OUTPUT:AUDIO 1         // Routes AF Generator Output to the AUDIO OUT
                            // Connector.
```

6-9 OSCILLOSCOPE

6-9-1 OSCILLOSCOPE COMMANDS

SCOPe:

ARM
[SCOPe:ARM]
Arms Oscilloscope. Sets Oscilloscope for one sweep. Command ignored unless Trigger is set to One Shot (see **SCOPe:TRIGger** commands).

AVErage *n*
[SCOPe:AVErage n]
Selects Oscilloscope Average Mode using *n* samples. Range of *n* is 1 to 100 (default is 100).

COMPare *n*
[SCOPe:COMPare n]
Selects Compare Mode for Oscilloscope. Trace stored at *n* memory location is compared to the current Live Trace.

Example:
```
SCOP:COMP 2   // Displays the Trace stored in memory location 2
              // along with the current live Trace.  All
              // Oscilloscope settings affect the live Trace only.
```

COUPling *type*
[SCOPe:COUPling type]
Selects *type* for External coupling. Select AC, DC or GROund.

FULL
[SCOPe:FULL]
Selects a full size Oscilloscope display for the RF Generator, Receiver and Duplex Operation Screens.

HORIZontal *n*
[SCOPe:HORIZontal n]
Sets Horizontal Time Offset to *n* major divisions. *n* is an integer from -12 to 12. -12 to -1 are major divisions before the trigger. 1 to 12 are major divisions after the trigger.

INPut:

FILTer:

CWEight:STATe *b*
[SCOPe:INPut:FILTer:CWEight:STATe b]
Enables Internal C-Weight filter if *b* is 1, disables if *b* is 0.

HPASs:

FREQuency *f*
[SCOPe:INPut:FILTer:HPASs:FREQuency f]
Sets Internal High-Pass frequency to *f* kHz. Range of *f* is 0.2 to 100.

STATe *b*
[SCOPe:INPut:FILTer:HPASs:STATe b]
Enables Internal High-Pass filter if *b* is 1, disables if *b* is 0.

SCOPe:

INPut:

FILTer:

LPASs:

FREQuency *f*

[SCOPe:INPut:FILTer:LPASs:FREQuency f]

Sets Internal Low-Pass frequency to *f* kHz. Range of *f* is 0.2 to 50.

STATe *b*

[SCOPe:INPut:FILTer:LPASs:STATe b]

Enables Internal Low-Pass filter if *b* is 1, disables if *b* is 0.

NOTch:

FREQuency *f*

[SCOPe:INPut:FILTer:NOTch:FREQuency f]

Sets Notch center frequency to *f* kHz. Range of *f* is 0.5 to 1.5.

STATe *b*

[SCOPe:INPut:FILTer:NOTch:STATe b]

Enables Internal Notch Filter if *b* is 1, disables if *b* is 0.

Example:

```
SCOP:INP:FILT:NOT:FREQ 1.5  // Sets Oscilloscope Input Notch Filter
                            // center frequency to 1.5 kHz.
SCOP:INP:FILT:NOT:STAT 1    // Enables Internal Notch Filter.
```

INTernal *type*

[SCOPe:INTernal type]

Selects *type* as Internal Oscilloscope Input. Select from: IF (Rcvr IF), DEMOD (Demod Audio), POWer (RF Pwr Lvl), SINAD (SINAD/BER), FUNCtion (Func Gen) or XAUDIO (Ext Mod).

LEVel *n*

[SCOPe:LEVel n]

Sets Trigger level to *n*. Range of *n* is 0 to 255 with 0 corresponding to the bottom of the Oscilloscope Display and 255 corresponding to the top.

LIVe *b*

[SCOPe:LIVe b]

Selects Live Trace Mode for the Oscilloscope. If *b* is 1, a **SCREEN:SCOPe** command is performed. If *b* is 0, no **SCREEN:SCOPe** command is performed. Default of optional *b* is 1.

SCOPe:

MARKer:

AOFF
[SCOPe:MARKer:AOFF]
Disables both Markers.

DELTA:

AMPLitude?
[SCOPe:MARKer:DELTA:AMPLitude?]
Returns voltage difference of the Trace Marker 1 and Trace Marker 2 crossings in volts. Valid only for Oscilloscope Inputs AC, DC and GND.

POINt?
[SCOPe:MARKer:DELTA:POINt?]
Returns the difference of Marker positions in graticules with 100 graticules equal to the Oscilloscope display width.

TIME?
[SCOPe:MARKer:DELTA:TIME?]
Returns the difference of the two Marker positions in ms.

TRACK *b*
[SCOPe:MARKer:TRACK b]
Enables Marker Tracking if *b* is 1, disables Marker Tracking if *b* is 0. Tracking feature keeps Markers a constant distance apart.

Commands for Marker 1 and Marker 2 are identical and are listed as **SCOPe:MARKER***x* commands. Parameter *x* is specified as 1 or 2 when the command is entered.

SCOPe:

MARKER*x*:

AMPLitude?
[SCOPe:MARKERx:AMPLitude?]
Returns voltage of live Trace at Marker *x* crossing in volts. Valid only for Oscilloscope Inputs AC, DC and GND.

POINt *n*
[SCOPe:MARKERx:POINt n]
Sets Marker *x* position to *n* graticules. Range of *n* is 1 to 100 with 100 graticules equal to Oscilloscope display width.

POINt?
[SCOPe:MARKERx:POINt?]
Returns Marker *x* position in graticules.

SCOPe:

MARKER*x*:

STATe *b*

[SCOPe:MARKERx:STATe b]

Enables Marker *x* if *b* is 1, disables Marker *x* if *b* is 0.

Both Markers are displayed when one or both are active.

STATe?

[SCOPe:MARKERx:STATe?]

Returns 1 if Marker *x* is active, 0 if Marker *x* is not active.

TIME?

[SCOPe:MARKERx:TIME?]

Returns Marker *x* position in ms from left edge of Oscilloscope display.

Example:

```
SCOP:MARKER1:STAT 1  // Enables Marker 1.
SCOP:MARKER2:STAT 1  // Enables Marker 2.
SCOP:MARKER1:POIN 20 // Positions Marker 1 two major divisions
                     // from left edge of display screen.
SCOP:MARKER2:POIN 40 // Positions Marker 2 four major divisions
                     // from left edge of display screen.
SCOP:MARK:TRACK 1    // Enables Marker Tracking.
SCOP:MARKER1:POIN 30 // Positions Marker 1 three divisions from
                     // left edge of display screen
                     // (and Marker 2 five divisions from
                     // left edge of display screen).
SCOP:MARK:DELTA:AMPL? // Queries amplitude difference of Trace
                     // Marker crossings in volts.
SCOP:MARK:DELTA:POIN? // Queries position difference of markers
                     // in graticules.
SCOP:MARK:DELTA:TIME? // Queries position difference of markers
                     // in ms.
SCOP:MARK:TRACK 0    // Disables Marker Tracking.
SCOP:MARKER2:AMPL?   // Queries the amplitude of the Trace at
                     // Marker 2 position.
SCOP:MARKER2:POIN?   // Queries Marker 2 position in graticules.
                     // 50 is returned.
SCOP:MARKER2:TIME?   // Queries Marker 2 position in ms.
SCOP:MARK:AOFF       // Disables both Markers.
```

SCOPe:

PLOT:

> The **SCOPe:PLOT:***xxx* commands send plotter data of the Oscilloscope grid, trace or units out the HOST RS-232 or GPIB Connector. **SYSTem:PLOT:GPIB** or **SYSTem:PLOT:SERial** command must be sent to change plotter output connector.

GRID
[SCOPe:PLOT:GRID]

Draws Oscilloscope grid on attached plotter.

TRACE
[SCOPe:PLOT:TRACE]

Draws Oscilloscope trace on attached plotter.

UNITS
[SCOPe:PLOT:UNITS]

Draws Oscilloscope units on attached plotter.

QTR
[SCOPe:QTR]

Selects a 1/4 size Oscilloscope display for the RF Generator, Receiver and Duplex Operation Screens.

RCL *n*
[SCOPe:RCL n]

Recalls Oscilloscope Trace and environment (routings and settings) stored in memory location *n*. Range of n is 1 to 9.

> Recalled Trace reflects the stored Trace parameters. Changing current settings does not alter the recalled Trace.

Example:
```
SCOP:RCL 4          // Recalls Oscilloscope trace and environment
                    // stored in memory location 4.
SCOP:SCAL 500       // Change current scale to 500 mV/div.
                    // Recalled Trace does not change position.
```

SCALe *n*
[SCOPe:SCALe n]

Sets Oscilloscope scale to *n* mV/div for AC, DC or GND Input. Select from:

1	2	5
10	20	50
100	200	500
1000	2000	5000
10000	20000	50000
100000		

Sets Oscilloscope scale to *n* kHz/div for Demod Audio Input with FM. Select from:

2	4	10
20		

Sets Oscilloscope scale to *n* mV/div for Func Gen or Ext Mod Input. Select from:

500	1000	2500

SCOPe:

SCALe?

[SCOPe:SCALe?]

Returns Oscilloscope scale in mV/div if input is AC, DC, GND, SINAD/BER, Func Gen or Ext Mod. Returns Oscilloscope scale in kHz/div if input is Demod Audio. Returns Oscilloscope scale in W if input is RF Pwr Lvl.

SOURce EXTernal

[SCOPe:SOURce EXTernal]

Routes Oscilloscope Input from the SCOPE IN Connector. Selects AC Ext for Oscilloscope Input.

SOURce INTernal

[SCOPe:SOURce INTernal]

Disconnects Oscilloscope Input from the SCOPE IN Connector. Selects Demod Audio for Oscilloscope Input.

STATe *b*

[SCOPe:STATe b]

Displays an Oscilloscope display in the RF Generator, Receiver and Duplex Operation Screens if *b* is 1. Oscilloscope display is not displayed if *b* is 0.

STORe *n*

[SCOPe:STORe n]

Stores current Oscilloscope Trace and environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

> The **SCOPe:SWEep** commands may be represented by the command: **SCOPe:SWEep** *n* [*units*]. For clarification, the commands are described separately.

SWEep *n*

[SCOPe:SWEep n]

Sets Oscilloscope sweep rate to *n* µs/div. Select from:

1	2	5
10	20	50
100	200	500
1000	2000	5000
10000	20000	50000
100000		

SWEep *n* MS

[SCOPe:SWEep n MS]

Sets Oscilloscope sweep rate to *n* ms/div. Select from:

1	2	5
10	20	50
100		

SCOPe:

SWEep *n* US

[SCOPe:SWEep n US]

Sets Oscilloscope sweep rate to *n* µs/div. Select from:

1	2	5
10	20	50
100	200	500

SWEep?

[SCOPe:SWEep?]

Returns Oscilloscope sweep rate in µs/div.

Example:
```
SCOP:INT FUNC        // Selects AF Generator as the Oscilloscope
                     // Input.
SCOP:LIV             // Selects Live for Oscilloscope Mode.
SCOP:TRIG:AUTO       // Selects Auto for Oscilloscope Trigger
                     // Mode.
```

For the following **SCOPe:TRACE** commands, the Oscilloscope display is divided into 400 positions horizontally (0 signifying the left edge of the display, 399 signifying the right edge of the display) and 256 values vertically (0 signifying the bottom of the display, 255 signifying the top of the display):

SCOPe:

TRACE:

DATA *n,offset,value,value,...,value*
[SCOPe:TRACE:DATA n,offset,value,value,...,value]
Replaces points of a stored Trace *n* with specified vertical values (*value*) starting with horizontal offset *(offset)*. *offset* is given in number of positions from the left edge of the display. Range of *n* is 1 to 9. Multiple *values* are separated by commas. One *value* is displayed for each succeeding horizontal position of the Trace starting with *offset*. Intended for remote GPIB or RS-232 use only.

DATA? [[[*n*],*offset*],*points*]
[SCOPe:TRACE:DATA? [[[n],offset],points]]
Sends the specified number of data points of Trace *n* to the Host. Trace *n* can be the live Trace or a Trace stored in memory. The affected *points* start with the optional *offset* with an *offset* default of 0. Range of *offset* is 0 to 399. Range of *n* is 0 to 9 with 0 signifying the Live Trace (0 default). Range of the number of *points* is 1 to 400 with a default of 400. Intended for remote GPIB or RS-232 use only.

GET *name,n*
[SCOPe:TRACE:GET name,n]
Assigns values of trace *n* (in graticules) to a declared array. Parameter *name* is the name of the declared array. Range of *n* is 0 to 9, 0 signifying the live trace and 1 to 9 signifying stored traces. If array is less than 400 values in length (stored trace length), the array values are assigned for the length of the array, leaving the rest of the stored trace intact.

GET? *n,offset*
[SCOPe:TRACE:GET? n,offset]
Returns value of a point in Trace *n* located the *offset* number of positions from the left edge of the Display. Trace *n* can be the live Trace or a Trace stored in memory. Range of *n* is 0 to 9 with 0 signifying the Live Trace. Range of *offset* is 0 to 399.

MAX? [[[*n*],*offset*],*points*]
[SCOPe:TRACE:MAX? [[[n],offset],points]]
Returns the maximum point of Trace *n* within the specified number of *points* starting with the given *offset*. Result is returned in x,y format with x being the number of positions from the left edge and y being the number of values from the bottom. Trace *n* can be the live Trace or a Trace stored in memory. Range of *n* is 0 to 9 with 0 signifying the Live Trace (0 default). Range of the optional *offset* is 0 to 399 with a default of 0. Range of optional number of *points* is 1 to 400 with a default of 400. Intended for remote GPIB or RS-232 use only.

SCOPe:

TRACE:

MIN? [[[*n*],*offset*],*points*]
[SCOPe:TRACE:MIN? [[[n],offset],points]]

Returns the minimum point of Trace *n* within the specified number of *points* starting with the given *offset*. Result is returned in x,y format with x being the number of positions from the left edge and y being the number of values from the bottom. Trace *n* can be the Live Trace or a Trace stored in memory. Range of *n* is 0 to 9 with 0 signifying the Live Trace (0 default). Range of the optional *offset* is 0 to 399 with a default of 0. Range of optional number of *points* is 1 to 400 with a default of 400. Intended for remote GPIB or RS-232 use only.

PUT *name*,*n*
[SCOPe:TRACE:PUT name,n]

Assigns values of an array to trace *n* (in graticules). Parameter *name* is the array name. Range of *n* is 1 to 9, signifying stored traces. If array is less than 400 values in length (stored trace length), the array values are assigned for the length of the array, leaving the rest of the stored trace intact.

Example:
```
VAR POINTS[199]
SCOP:TRACE:DATA 3,40,128    // Changes a single Trace value,
                            // located one division from the
                            // left edge, to the midway point
                            // (on the axis).
SCOP:TRACE:GET? 0,200       // Queries the value of the live
                            // Trace at the point 200 (center
                            // of oscilloscope display).
POINTS=SCOP:TRACE:DATA? 3,200,200 // Returns the Trace values
                                  // of the right half of the
                                  // Trace stored in memory
                                  // location 3.
SCOP:TRACE:MAX? 0,200       // Queries the maximum Trace value
                            // of the left half of the live
                            // Trace.
SCOP:TRACE:MIN? 0,40,80     // Queries the minimum Trace value
                            // of the live Trace for the second
                            // and third divisions.
```

TRIGger:

AUTO
[SCOPe:TRIGger:AUTO]

Sets Trigger to Auto Sweep.

IMMediate
[SCOPe:TRIGger:IMMediate]

Triggers Oscilloscope as soon as command is interpreted or sequenced in a macro.

NORM
[SCOPe:TRIGger:NORM]

Sets Trigger to Normal Sweep.

SCOPe:

TRIGger:

ONE
[SCOPe:TRIGger:ONE]

Sets Trigger to One Shot. Causes oscilloscope to sweep once when trigger is received and after setting arm function (**SCOPe:ARM** command).

SOURce *type*
[SCOPe:TRIGger:SOURce type]

Selects source to trigger on. Select EXTernal (for external trigger, AC or DC Input only), INTernal (for internal trigger, AC or DC Input only) or BUS (triggered by *TRG or IEEE-488.1 GET [Get Execute Trigger] command).

VERTical *n*
[SCOPe:VERTical n]

Sets Vertical Offset voltage to *n* graticules. Range of n is 0 to 255. Setting is not linear. For AC, DC or GND, middle setting is approximately 150 to 170. For other Inputs, middle setting is approximately 160 to 180.

6-9-2 REMOTE OSCILLOSCOPE EXAMPLES

The following example is a macro that centers a trace in the middle of the color display. If the trace centered has AC, DC or GND Input, GND Input should be selected before executing this macro. If the trace centered has other Input, RF Pwr Lvl Input should be selected before executing this macro. The Oscilloscope Operation Screen must be displayed for this macro to operate.

```
*DMC "Center",BEGIN             // Defines macro names Center.
VAR HOLD,CENTER_LINE,X          // Declares variables needed.
CENTER_LINE = 0                 // Initializes Center_line variable.
FOR X = 0 TO 255                // Loops for every possible vertical trace
                                // position.
  SCOPE:VERT X                  // Moves trace to vertical position x.
  DELAY 50                      // Pause allows SCOPE:VERT command to finish.
  HOLD = SCOP:TRACE:GET? 0,200              // Trace's actual vertical
                                            // position is stored in HOLD.
  IF (HOLD>=(128-3)) AND (HOLD<=(128+3)) // If HOLD is close to center
                                            // of color display,
    CENTER_LINE = X                         // store x value into
                                            // CENTER_LINE.
  ENDIF                         // End of IF statement.
NEXT X                          // End of FOR loop.
SCOPE:VERT CENTER_LINE          // Trace is moved to point previously found
                                // that centers trace.
END                             // End of macro Center.
```

The following example is a macro that finds the maximum of the live trace starting from a given offset. Range of the offset is 0 to 400. This macro returns the horizontal position of the maximum point. If there are multiple maximum points, the first one is returned. With the **RETURN** command included, this macro is executed as a function (ex. X = FINDMAX 0).

```
*DMC "FINDMAX",BEGIN            // Define macro named FINDMAX
VAR NEW,OLD=0,OLD_X=0           // Declare and initialize variables: NEW hold
                                // latest amplitude reading, OLD holds
                                // highest amplitude reading yet found and
                                // OLD_X holds position of OLD.
FOR X=$1 TO 400                 // Loop from offset given to end of color display.
  NEW=SCOPE:TRACE:GET? 0,X      // Read 1 point and store in NEW.
  IF NEW > OLD+3                // If current amplitude is greater than last
                                // max,
    OLD = NEW                   // store new max into old max and,
    OLD_X = X                   // store new max location into old max
                                // location.
  ENDIF                         // End of IF statement.
NEXT X                          // End of looping.
RETURN OLD_X                    // Return location of maximum value found.
END                             // End of FINDMAX macro.
```

The following command sequence compares a stored Trace (1 kHz sine wave) with a live Trace (2 kHz square wave). This example assumes the previous macro "Center" already resides in Test Set memory.

```
                              // AF Generator is setup to create first
                              // signal.

FGEN:OUTPUT:LEVEL 1           // Sets AF Generator Output Level to 1 V.
FGEN:GEN1:STATE 1             // Enables AF Generator 1.
FGEN:GEN1:FREQ 1000           // Sets AF Generator 1 frequency to 1000 Hz.
FGEN:GEN1:SHAPE:SIN           // Selects a sine wave for AF Generator 1
                              // Wave Form.

                              // Oscilloscope parameters are set and first
                              // Trace is stored.

SCREEN:SCOP                   // Displays Oscilloscope Operation Screen.
SCOPE:SOURCE INTERNAL         // Sets Oscilloscope to be triggered
                              // internally.
SCOPE:TRIGGER:NORM            // Selects Normal as Oscilloscope Trigger
                              // Mode.
                              // Sets Trigger Level to the midpoint.
SCOPE:LIVE                    // Sets Oscilloscope Mode to live.
SCOPE:INTERNAL POWER          // Sets Oscilloscope Input to RF Pwr Lvl.
Center                        // Executes macro Center which centers trace
                              // on display screen (see example above).
SCOPE:INTERNAL FUNC           // Selects AF Generator as Oscilloscope
                              // Input.
SCOPE:SCALE 1000              // Sets Scale to 1 volt/div.
SCOPE:SWEEP 200               // Sets Sweep rate to 200 µV/div.
SCOPE:STORE 1                 // Stores Trace in memory location 1.

                              // Second signal is created.

FGEN:GEN1:FREQ 2000           // Changes AF Generator 1 frequency to 2 kHz.
FGEN:GEN1:SHAPE:SQU           // Changes AF Generator 1 Wave Shape to a
                              // square wave.

                              // The first and second signals are
                              // compared.

SCOPE:COMPARE 1               // Selects Compare mode for Oscilloscope.
                              // Trace stored in memory location 1 is
                              // displayed on the Oscilloscope display
                              // with the current live Trace.
```

The following command sequence displays the sum of a 1 kHz and 2.5 kHz sine wave on the Oscilloscope display and returns the period of the signals. This example assumes the macro FINDMAX (see second example) is already loaded into Test Set memory.

```
                                          // Setup AF Generator 1 and 2 to generate
                                          // the signal to be measured.
FGEN:PROPORTIONAL 1                       // Enable AF Generator Proportional Mode.
FGEN:OUTPUT:LEVEL 1                       // Set AF Generator Output Level to 1 V.
FGEN:GEN1:STATE 1                         // Enable AF Generator 1.
FGEN:GEN1:FREQ 1000                       // Set AF Generator 1 Frequency to 1000 Hz.
FGEN:GEN1:SHAPE:SIN                       // Select a sine wave for the AF
                                          // Generator 1 Wave Form.
FGEN:GEN1:LEVEL 100                       // Set AF Generator 1 Proportional Level
                                          // to 100%.
FGEN:GEN2:STATE 1                         // Enable AF Generator 2.
FGEN:GEN2:FREQ 2500                       // Set AF Generator 2 Frequency to 2500 Hz.
FGEN:GEN2:SHAPE:SIN                       // Select a sine wave for the AF
                                          // Generator 1 Wave Form.
FGEN:GEN2:LEVEL 100                       // Set AF Generator 2 Proportional Level
                                          // to 100%.

                                          // MEAS_PER macro measures the period of
                                          // the AF Generator signal.
*DMC "MEAS_PER",BEGIN                     // Define macro named MEAS_PER.
VAR MAX1,MAX2                             // Declare variables: MAX1 holds location
                                          // of first maximum, MAX2 holds location
                                          // of second maximum.
SCREEN:SCOP                               // Display the Oscilloscope Screen.
SCOPE:SOURCE INTERNAL                     // Select Internal Oscilloscope Input.
SCOPE:INTERNAL FUNC                       // Select the AF Generator as the
                                          // Oscilloscope Input.
SCOPE:LIVE                                // Select Live Mode for the Oscilloscope.
SCOPE:SCALE 500                           // Set Oscilloscope Scale to 0.5 V/div.
SCOPE:SWEEP 500                           // Set Oscilloscope Sweep Rate to
                                          // 500 µs/div.
SCOPE:TRIGGER:ONE                         // Select One Shot Trigger Mode.  Hold
                                          // trace still.
DELAY 600                                 // Allow time to set for arming function.
SCOPE:ARM                                 // Display and hold trace when triggered.
MAX1 = FINDMAX 0                          // MAX! is set to first maximum using
                                          // macro of second example.
MAX2 = FINDMAX (MAX1+5)                   // MAX2 is set to second maximum using
                                          // macro of second example.
SCOPE:MARKER1:STATE 1                     // Enables Marker 1.
SCOPE:MARKER1:POINT FLOOR(MAX1/4)         // Position Marker 1 on the first maximum.
SCOPE:MARKER2:STATE 1                     // Enables Marker 2.
SCOPE:MARKER2:POINT FLOOR(MAX2/4)         // Positions Marker 2 on the second
                                          // maximum.
T=SCOPE:MARKER:DELTA:TIME?                // Query the difference in Marker
                                          // positions in ms.
PPRINT "PERIOD IS ",1000/T                // Figure and print period to Host.
END                                       // End macro MEAS_PER.
```

6-10 SPECTRUM ANALYZER

6-10-1 SPECTRUM ANALYZER COMMANDS

ANLZ:

AVErage [*n*]
[ANLZ:AVErage [n]]
Selects Average Mode for Analyzer using *n* samples. Range of *n* is 1 to 100. Default is 100.

CHANnel *n*
[ANLZ:CHANnel n]
Sets RF Frequency to cellular channel *n* (1 to 2047, depending on format) in the format selected using the **ANLZ:CHANnel:FORMat** command. The Spectrum Analyzer Operation Screen displays selected cellular channel when Analyzer is in Channel Mode (**ANLZ:MODE CHAN** command).

CHANnel:

FORMat:

AMPS:

FORward
[ANLZ:CHANnel:FORMat:AMPS:FORward]
Selects AMPS (NADC-U8) Forward channels.

REVerse
[ANLZ:CHANnel:FORMat:AMPS:REVerse]
Selects AMPS (NADC-U8) Reverse channels.

ETACS:

FORward
[ANLZ:CHANnel:FORMat:ETACS:FORward]
Selects ETACS Forward channels.

REVerse
[ANLZ:CHANnel:FORMat:ETACS:REVerse]
Selects ETACS Reverse channels.

NADC:

FORward
[ANLZ:CHANnel:FORMat:NADC:FORward]
Selects NADC Forward channels.

REVerse
[ANLZ:CHANnel:FORMat:NADC:REVerse]
Select NADC Forward channels.

BAND:*xx*
[ANLZ:CHANnel:FORMat:NADC:BAND:x]
Selects NADC band. Range of x is U8, U4 or HYper.

ANLZ:

CHANnel:

FORMat:

NAMPS:

FORward
[ANLZ:CHANnel:FORMat:NAMPS:FORward]
Selects NAMPS Forward channels.

REVerse
[ANLZ:CHANnel:FORMat:NAMPS:REVerse]
Selects NAMPS Reverse channels.

BAND:*x*
[ANLZ:CHANnel:FORMat:NAMPS:BAND:x]
Selects NAMPS Narrow Analog channel designator. Range of *x* is Lower, Middle or Upper.

NT400:

FORward
[ANLZ:CHANnel:FORMat:NT400:FORward]
Selects NT400 Forward channels.

REVerse
[ANLZ:CHANnel:FORMat:NT400:REVerse]
Selects NT400 Reverse channels.

BAND?
[ANLZ:CHANnel:BAND?]
Returns band for Analyzer Channel Format. Returns one of the following bands for the associated channel format:

FORMAT	BAND
NADC	U8, U4 or HY
ETACS	NOT AVAILABLE
NAMPS	LOWER, MIDDLE or UPPER

FORMat?
[ANLZ:CHANnel:FORMat?]
Returns Channel Format (NADC:FORWARD, NADC:REVERSE, ETACS:FORWARD, ETACS:REVERSE, NAMPS:FORWARD or NAMPS:REVERSE).

ANLZ:

COMPare *n*
[ANLZ:COMPare n]

Selects Compare Mode for Analyzer. Trace stored at *n* memory location is compared with Live Trace. Range of *n* is 1 to 9.

Example:
```
ANLZ:COMP 1            // Selects Compare mode for Analyzer.
                       // Trace stored in memory location 1 is
                       // displayed on the Analyzer display
                       // with the current live Trace.
```

FIND:

FREQuency?
[ANLZ:FIND:FREQuency?]

Returns frequency of first signal with amplitude larger than Find reference level. Returns 0 if no signal is found.

REFerence *n*
[ANLZ:FIND:REFerence n]

Sets Find reference level to *n* dB.

REFerence?
[ANLZ:FIND:REFerence?]

Returns Find reference level in dB.

Example:
```
ANLZ:FIND:REF -65      // Sets Find Reference Level to -65 dBm.
ANLZ:FIND:FREQ?        // Returns the lowest frequency (in kHz)
                       // containing a signal greater than -65 dBm.
ANLZ:FIND:REF?         // Queries the current Find Reference Level
                       // (in dBm).  -65 is returned.
```

FREQuency *f*
[ANLZ:FREQuency f]

Sets Analyzer Frequency to *f* kHz. Range for *f* is 250.0 to 2010000.0.

FREQuency?
[ANLZ:FREQuency?]

Returns Analyzer Frequency in kHz.

Example:
```
ANLZ:FREQ 135 MHZ      // Set Analyzer Frequency to 135 MHz.
ANLZ:FREQ?             // Query the Analyzer Frequency.
                       // 135000 is returned (135000 kHz).
```

FULL
[ANLZ:FULL]

Selects a full size Analyzer display for the RF Generator, Receiver and Duplex Operation Screens.

ANLZ:

INPut:

ANTenna

[ANLZ:INPut:ANTenna]

Selects ANTENNA IN Connector for the Analyzer Input.

> The following Attenuation commands control the Input Attenuation for the Spectrum Analyzer and selection/deselection of LNA (Low Noise Amplifier). Use **ANLZ:INPut: ATTenuation:LNA** to select LNA and 0 dB of input attenuation. To deselect LNA and select a specific value of attenuation, use **ANLZ:INPut:ATTenuation**.

ATTenuation:LNA

[ANLZ:INPut:ATTenuation:LNA]

Sets Analyzer Input Attenuation to LNA.

ATTenuation:LNA?

[ANLZ:INPut:ATTenuation:LNA?]

Returns current state of the Low Noise Amplifier. Returns 1 if LNA is selected; 0 otherwise.

> LNA (Low Noise Amplifier) has the same attenuation as 0 dB (see **ANLZ:INPut: ATTenuation** *n*), but LNA has a lower noise figure. LNA is the preferred option for performing off-the-air applications.

ATTenuation *n*

[ANLZ:INPut:ATTenuation n]

Sets Analyzer Input Attenuation to *n* dB. Possible values for *n*: 0, 5, 10, 15, 20, 25, 30.

ATTenuation?

[ANLZ:INPut:ATTenuation?]

Returns the current value of Input Attenuation.

TR

[ANLZ:INPut:TR]

Selects T/R Connector for the Analyzer Input.

INPut?

[ANLZ:INPut?]

Returns Analyzer Input setting.

Example:

```
ANLZ:INP:TR        // Selects T/R Connector for Analyzer Input.
ANLZ:INP:ATT 20    // Sets Analyzer Input Attenuation to 20 dB.
ANLZ:INP?          // Queries Analyzer Input Connector.  TR is
                   // returned.
ANLZ:INP:ATT?      // Queries Analyzer Input Attenuation.
                   // 20 is returned.
```

LIVe

[ANLZ:LIVe]

Selects Live Trace mode for the Spectrum Analyzer.

ANLZ:

MARKer:

AOFF

[ANLZ:MARKer:AOFF]

Disables both Markers.

DELTA:

AMPLitude?

[ANLZ:MARKer:DELTA:AMPLitude?]

Returns amplitude difference between the Trace Marker 1 and Trace Marker 2 crossings in dB.

FREQuency?

[ANLZ:MARKer:DELTA:FREQuency?]

Returns the difference between the two Marker positions in MHz.

POINt?

[ANLZ:MARKer:DELTA:POINt?]

Returns the difference between the Marker positions in graticules with 100 graticules equal to the Analyzer display width.

TRACK *b*

[ANLZ:MARKer:TRACK b]

Enables Marker Tracking Feature if *b* is 1, disables Marker Tracking Feature if *b* is 0. Tracking feature keeps Markers a constant distance apart.

Commands for Marker 1 and Marker 2 are identical and are listed as **ANLZ:MARKER**_x commands._ Parameter _x_ is specified as 1 or 2 when the command is entered.

ANLZ:

MARKER_x:_

AMPLitude?

[ANLZ:MARKERx:AMPLitude?]

Returns amplitude of the Trace at the Marker _x_ crossing. Range and units depend on the current scale settings (**ANLZ:SCALe** commands).

FREQuency?

[ANLZ:MARKERx:FREQuency?]

Returns Marker _x_ position in kHz (250 to 2010000.0).

POINt _n_

[ANLZ:MARKERx:POINt n]

Sets Marker _x_ position to _n_ graticules. Range of _n_ is 1 to 100 with 100 graticules equal to the Analyzer display width.

POINt?

[ANLZ:MARKERx:POINt?]

Returns Marker _x_ position in graticules.

STATe _b_

[ANLZ:MARKERx:STATe b]

Enables Marker x if _b_ is 1, disables Marker x if _b_ is 0.

Example:
```
ANLZ:MARKER1:STAT 1   // Enables Marker 1.
ANLZ:MARKER2:STAT 2   // Enables Marker 2.
ANLZ:MARKER1:POIN 10  // Positions Marker 1 one divisions from the
                      // left edge of the color display.
ANLZ:MARKER2:POIN 30  // Positions Marker 2 three divisions from
                      // the left edge of the color display.
ANLZ:MARK:TRACK       // Enables Marker Tracking.
ANLZ:MARKER1:POIN 60  // Positions Marker 1 six divisions from
                      // the left edge of the color display (and
                      // Marker 2 eight divisions from the
                      // left edge of the color display).
ANLZ:MARK:DELTA:AMPL? // Queries amplitude difference of Trace
                      // Marker crossings in volts.
ANLZ:MARK:DELTA:POIN? // Queries position difference of markers in
                      // graticules.
ANLZ:MARK:DELTA:FREQ? // Queries position difference of markers
                      // in kHz.
ANLZ:MARK:TRACK 1     // Disables Marker Tracking.
ANLZ:MARKER2:AMPL?    // Queries the amplitude of the Trace at the
                      // Marker 2 position.
ANLZ:MARKER2:POIN?    // Queries Marker 2 position in graticules.
                      // 80 is returned.
ANLZ:MARKER2:FREQ?    // Queries Marker 2 position in kHz.
ANLZ:MARK:AOFF        // Disables both Markers.
```

STATe?

[ANLZ:MARKERx:STATe?]

Returns current state of Marker _x_.

ANLZ:

MODE *type*
[ANLZ:MODE type]

Selects Analyzer RF Mode (DIRect [Direct Mode] or CHANnel [Channel Mode]). Channel Mode displays cellular channel frequency according to **ANLZ:CHANnel** commands.

NORMalize
[ANLZ:NORMalize]

Normalizes the Analyzer Trace to match the RF Generator Output.

PEAK
[ANLZ:PEAK]

Selects Peak Hold Feature for the Analyzer.

PLOT:

The **ANLZ:PLOT:***xxx* commands send plotter data of the Analyzer grid, trace or units out the HOST RS-232 or GPIB Connector. **SYSTem:PLOT:GPIB** or **SYSTem:PLOT:SERial** command must be sent to change plotter output connector.

GRID
[ANLZ:PLOT:GRID]

Draws Analyzer grid on attached plotter.

TRACE
[ANLZ:PLOT:TRACE]

Draws Analyzer trace on attached plotter.

UNITS
[ANLZ:PLOT:UNITS]

Draws Analyzer units on attached plotter.

QTR
[ANLZ:QTR]

Selects a 1/4 size Analyzer display for the RF Generator, Receiver and Duplex Operation Screens.

RCL *n*
[ANLZ:RCL n]

Recalls Analyzer Trace and parameters stored in memory location *n*. Range of *n* is 1 to 9.

RLEVel *n*
[ANLZ:RLEVel n]

Sets reference or offset level in dB for the Analyzer only in 2 dB scale. Range of *n* is 0 to 64.

RLEVel?
[ANLZ:RLEVel?]

Returns reference or offset value in dB for the Analyzer in 2 dB scale.

SCALe *n*
[ANLZ:SCALe n]

Sets Analyzer Units/Division Factor to *n* dB. Select 2 or 10.

ANLZ:

SCALe:

UNIT:*type*
[ANLZ:SCALe:UNIT:type]
Sets Analyzer Scale Units to *type*. For T/R Analyzer Input, select DBM (dBm) or DBW (dBW). For ANTENNA Analyzer Input, select:

DBM (dBm)	DBMV (dBmV)	DBUV (dBμV)
DBV (dBV)	DBUW (dBμW)	

If DBW is selected, the T/R Connector is selected for the Analyzer Input. If DBV, DBMV, DBUV or DBUW is selected, the ANTENNA IN Connector is selected for the Analyzer Input.

UNIT?
[ANLZ:SCALe:UNIT?]
Returns Analyzer Scale Units.

SCALe?
[ANLZ:SCALe?]
Returns Analyzer Scale in dB.

SCAN *n*
[ANLZ:SCAN n]
Sets Analyzer Scan Width to *n* kHz. Select 0 for zero scan or one of the following:

1	2	5
10	20	50
100	200	500
1000	2000	5000
10000	20000	50000
100000		

SCAN?
[ANLZ:SCAN?]
Returns Analyzer Scan Width in kHz.

STATE *b*
[ANLZ:STATE b]
Displays Analyzer display in the RF Generator, Receiver and Duplex Operation Screens if *b* is 1. Analyzer display is not displayed if *b* is 0.

STORe *n*
[ANLZ:STORe n]
Stores the current Analyzer Trace and environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

ANLZ:

TOP?

[ANLZ:TOP?]

Returns the top of screen value in current units. Spectrum Analyzer Operation Screen must be displayed.

Example:
```
ANLZ:SCAL 2            // Selects 2 dB/div for Analyzer
                       // Units/Division Factor.
ANLZ:SCAN 100          // Sets Analyzer Scan Width to 100 kHz.
ANLZ:SCAL:UNIT:DBM     // Selects dBm for Analyzer Scale Units.
ANLZ:TOP?              // Queries the amplitude value for the top
                       // of the Analyzer display (in the current
                       // units).
ANLZ:SCAN?             // Queries Analyzer Scan Width in kHz.  100
                       // is returned.
ANLZ:SCAL:UNIT?        // Queries Analyzer Scale Units.
```

For the following **ANLZ:TRACE** commands, the Analyzer display is divided into 400 positions horizontally (0 signifying the left edge of the display, 399 signifying the right edge of the display) and 255 values vertically (0 signifying the bottom of the display, 255 signifying the top of the display):

ANLZ:

TRACE:

DATA *n,offset,value,value,...,value*

[ANLZ:TRACE:DATA n,offset,value,value,...,value]

Replaces points of a stored Trace *n* with specified *values* starting with *offset. Offset* is given in number of positions from the left edge of the display. Range of *n* is 1 to 9. Multiple *values* are separated by commas. Only one *value* can be displayed for each horizontal position of the Trace. Intended for remote GPIB or RS-232 use only.

DATA? [[[*n*],*offset*],*points*]

[ANLZ:TRACE:DATA? [[[n],offset],points]]

Sends the specified number of data points of Trace *n* to the Host. Optional Trace *n* can be the live Trace or a Trace stored in memory. The affected *points* start with the optional *offset* with an *offset* default of 0. Range of *offset* is 0 to 399. Range of *n* is 0 to 9 with 0 signifying the Live Trace (0 default). Range of the number of *points* is 1 to 400 with a default of 400. Intended for remote GPIB or RS-232 use only.

ANLZ:

TRACE:

GET *name,n*
[ANLZ:TRACE:GET name,n]
Assigns values of trace *n* (in graticules) to a declared array. Parameter *name* is the name of the declared array. Range of *n* is 0 to 9, 0 signifying the live trace and 1 to 9 signifying stored traces. If array is less than 400 values in length (stored trace length), the array values are assigned for the length of the array, the remaining portion of trace is left unassigned to an array.

GET? *n,offset*
[ANLZ:TRACE:GET? n,offset]
Returns value of a point in Trace *n* located the *offset* number of positions from the left edge of the Display. Trace *n* can be the live Trace or a Trace stored in memory. Range of *n* is 0 to 9 with 0 signifying the Live Trace. Range of the *offset* is 0 to 399.

MAX? *n,offset,points*
[ANLZ:TRACE:MAX? n,offset,points]
Returns the maximum point of Trace *n* within the specified number of *points* starting with the given *offset*. Result is returned in x,y format with x being the number of positions from the left edge and y being the number of values from the bottom. Trace *n* can be the live Trace or a Trace stored in memory. Range of *n* is 0 to 9 with 0 signifying the Live Trace. Range of the optional *offset* is 0 to 399 with a default of 0. Range of optional number of *points* is 1 to 400 with a default of 400. Intended for remote GPIB or RS-232 use only.

MIN? *n,offset,points*
[ANLZ:TRACE:MIN? n,offset,points]
Returns the minimum point of Trace *n* within the specified number of *points* starting with the given *offset*. Result is returned in x,y format with x being the number of positions from the left edge and y being the number of values from the bottom. Trace *n* can be the live Trace or a Trace stored in memory. Range of *n* is 0 to 9 with 0 signifying the Live Trace. Range of the optional *offset* is 0 to 399 with a default of 0. Range of optional number of *points* is 1 to 400 with a default of 400 Intended for remote GPIB and RS-232 use only.

ANLZ:

TRACK:

PUT *name,n*
[ANLZ:TRACE:PUT name,n]
Assigns values of an array to trace *n* (in graticules). Parameter *name* is the array name. Range of *n* is 1 to 9, signifying stored traces. If array is less than 400 values in length (stored trace length), the array values are assigned for the length of the array, leaving the rest of the stored trace intact.

Example:
```
ANLZ:TRACE:DATA 2,80,128     // Changes a single Trace value,
                             // located 2 divisions from the left
                             // edge, to the midway point (on the
                             // axis).
ANLZ:TRACE:GET? 0,240        // Queries the value of the live
                             // Trace at the point 240 (six
                             // divisions from the left edge of
                             // the display).
ANLZ:TRACE:DATA? 3,200,200   // Returns the Trace values of the
                             // right half of the Trace stored
                             // in memory location 3 (200 values).
ANLZ:TRACE:MAX? 0,200        // Queries the maximum Trace value of
                             // the left half of the live Trace.
ANLZ:TRACE:MIN? 0,40,80      // Queries the minimum Trace value
                             // of the live Trace for the second
                             // and third divisions.
```

BWIDth *f*
[ANLZ:TRACK:BWIDth f]
Sets Tracking Generator Bandwidth to *f* kHz. Select 3, 30, 300 or 3000.

BWIDth?
[ANLZ:TRACK:BWIDth?]
Returns Tracking Generator Bandwidth in kHz.

LEVel *n*
[ANLZ:TRACK:LEVel n]
Sets Tracking Generator Level to *n* dBm. Range of *n* is 0.0 to -137.0.

LEVel?
[ANLZ:TRACK:LEVel?]
Returns Tracking Generator Level in dBm.

ANLZ:

TRACK:

OUTput:

TR
[ANLZ:TRACK:OUTput:TR]
Selects T/R Connector as Tracking Generator Output Connector.

DUPlex
[ANLZ:TRACK:OUTput:DUPlex]
Selects DUPLEX OUT Connector as Tracking Generator Output Connector.

OUTput?
[ANLZ:TRACK:OUTput?]
Returns current Tracking Generator Output Connector.

RESolution:

HIGH
[ANLZ:TRACK:RESolution:HIGH]
Selects high for Tracking Generator Resolution.

LOW
[ANLZ:TRACK:RESolution:LOW]
Selects low for Tracking Generator Resolution.

MED
[ANLZ:TRACK:RESolution:MED]
Selects medium for Tracking Generator Resolution.

RESolution?
[ANLZ:TRACK:RESolution?]
Returns HIGH if Tracking Resolution is high, LOW if Tracking Resolution is low or MED if Tracking Resolution is medium.

ANLZ:

TRACK:

STATe *b*

[ANLZ:TRACK:STATe b]

Enables Tracking Generator if *b* is 1, disables Tracking Generator if *b* is 0.

STATe?

[ANLZ:TRACK:STATe?]

Returns 1 if Tracking Generator is active; 0 if inactive.

Example:

```
ANLZ:TRACK:LEV -10   // Sets Analyzer Tracking Generator Level
                     // to -10 dBm.
ANLZ:TRACK:RES:HIGH  // Sets Analyzer Tracking Generator
                     // Resolution to high.
ANLZ:TRACK:BWID 30   // Sets Analyzer Bandwidth to 30 kHz.
ANLZ:TRACK:STAT 1    // Enables the Analyzer Tracking Generator.
ANLZ:TRACK:LEV?      // Queries the Analyzer Tracking Generator
                     // Level.  -10 is returned.
ANLZ:TRACK:RES?      // Queries the Analyzer Tracking Generator
                     // Resolution.  HIGH is returned.
ANLZ:TRACK:BWID?     // Queries the Analyzer Bandwidth in kHz.
                     // 30 is returned.
ANLZ:TRACK:STAT?     // Queries the current status of the
                     // Analyzer Tracking Generator.  1 is
                     // returned.
```

6-10-2 REMOTE SPECTRUM ANALYZER EXAMPLE

The following command sequence measures a 96 MHz signal received at the ANTENNA IN Connector and returns the amplitude in dB:

```
SCREEN:ANLZ                 // Displays the Analyzer Operation Screen.
ANLZ:SCAN 20                // Sets the Analyzer Scan Width to 20 kHz.
ANLZ:FREQ 96000             // Sets Analyzer RF Frequency to 96 MHz.
ANLZ:LIVE                   // Selects the Live Analyzer Mode.
ANLZ:SCALE 10               // Selects the 10 Units/Division Factor.
ANLZ:SCALE:UNIT:DBM         // Selects dBm for the Analyzer Scale Units.
ANLZ:INPUT:ANTENNA          // Selects the ANTENNA IN Connector for the
                            // Analyzer Input.
ANLZ:INPUT:ATTENUATION 0    // Sets Analyzer Input Attenuation to 0 dB.
ANLZ:MARKER1:STATE 1        // Enables Marker 1.
ANLZ:MARKER1:POINT 50       // Positions Marker 1 at the Analyzer RF
                            // Frequency.
ANLZ:MARKER1:AMPLITUDE?     // Queries the amplitude of the Trace at the
                            // Marker 1 position.
```

6-11 METER COMMANDS

6-11-1 AF METER COMMANDS

M_AF:

ALARM *b*
[M_AF:ALARM b]
Enables Alarm if *b* is 1, disables if *b* is 0. Enabled Alarm sounds when Upper or Lower Limit is surpassed.

FILTer:

APASs
[M_AF:FILTer:APASs]
Selects the All-Pass Filter for the AF Meter.

APASs?
[M_AF:FILTer:APASs?]
Returns 1 if All-Pass Filter is enabled, 0 if disabled.

HPASs:

FREQuency *f*
[M_AF:FILTer:HPASs:FREQuency f]
Sets High-Pass cutoff frequency to *f* kHz. Range of *f* is 0.5 to 20.0.

FREQuency?
[M_AF:FILTer:HPASs:FREQuency?]
Returns High-Pass cutoff frequency in kHz.

STATe *b*
[M_AF:FILTer:HPASs:STATe b]
Enables High-Pass Filter if *b* is 1, disables if *b* is 0.

STATe?
[M_AF:FILTer:HPASs:STATe?]
Returns 1 if High-Pass Filter is enabled, 0 if disabled.

LPASs:

FREQuency *f*
[M_AF:FILTer:LPASs:FREQuency f]
Set Low-Pass cutoff frequency to *f* kHz. Range of *f* is 0.1 to 30.0.

FREQuency?
[M_AF:FILTer:LPASs:FREQuency?]
Returns Low-Pass cutoff frequency in kHz.

M_AF:

FILTer:

LPASs:

STATe *b*
[M_AF:FILTer:LPASs:STATe b]
Enables Low-Pass Filter if *b* is 1, disables if *b* is 0.

STATe?
[M_AF:FILTer:LPASs:STATe?]
Returns 1 if Low-Pass Filter is enabled, 0 if disabled.

Example:
```
M_AF:FILT:LPAS:FREQ 5      // Sets Low-Pass cutoff frequency to 5 kHz.
M_AF:FILT:LPAS:STAT 1      // Enables Low-Pass Filter.
M_AF:FILT:LPAS:FREQ?       // Queries the Low-Pass cutoff frequency in
                           // kHz.  5 is returned.
M_AF:FILT:LPAS:STAT?       // Queries the Low-Pass Filter status.
                           // 1 is returned.
```

INPut:*type*
[M_AF:INPut:type]
Selects the Audio Frequency Meter Input. Select from the following for *type*:
XAUDIO (Ext Mod), DEMOD (Demod Audio), FGEN (Func Gen Out), SINAD (SINAD/BER) or POWer (RF Power).

LL:

LEVel *f*
[M_AF:LL:LEVel f]
Sets Lower Limit to *f* kHz. Range of *f* is 0.0000 to 0.2000 for an upper range of 0.2. Range of *f* is 0.000 to 200.000 for upper ranges of 2, 20 and 200.

STATe *b*
[M_AF:LL:STATe b]
Enables Audio Frequency Meter Lower Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding the Lower Limit sets bit 3 of the Instrument Status Register to one (activating bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Lower Limit and Alarm (**M_AF:ALARM 1**) are enabled, exceeding the Lower Limit activates an audio alarm.

PEAK?
[M_AF:PEAK?]
Returns Audio Frequency Meter Peak reading in Hz (0.0 to 200000.0).

PH *b*
[M_AF:PH b]
Enables Peak Hold Feature when *b* is 1, disables Peak Hold Feature when *b* is 0.

M_AF:

RANGe:

AUTO
[M_AF:RANGe:AUTO]
Sets frequency range to Autorange.

UPPer *f*
[M_AF:RANGe:UPPer f]
Sets frequency range to *f* kHz. Select 0.2, 2, 20 or 200.

Example:
```
M_AF:INP:SINAD       // Selects SINAD/BER IN Connector for AF
                     // Meter Input.
M_AF:RANG:UPP 20     // Sets AF Meter Range to 20 kHz.
M_AF:PH 1            // Enables AF Meter Peak Hold.
M_AF:LL:LEV 55.5     // Sets a Lower Limit of 55.5 kHz.
M_AF:LL:STAT 1       // Enables Lower Limit.
M_AF:ALARM 1         // Enables Alarm.
```

RCL *n*
[M_AF:RCL n]
Recalls Audio Frequency Meter environment (routings and settings) stored in memory location *n*. Range of *n* is 1 to 9.

RESolution *n*
[M_AF:RESolution n]
Sets Audio Frequency Meter Resolution (Gate Time) to *n* Hz. Select 0.1 (Gate Time of 10 s) or 1 (Gate Time of 1 s).

STORe *n*
[M_AF:STORe n]
Stores current Audio Frequency Meter environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

UL:

LEVel *f*
[M_AF:UL:LEVel f]
Sets Upper Limit to *f* kHz. Range of *f* is 0.0000 to 0.2000 for an upper range of 0.2. Range of *f* is 0.000 to 200.000 for upper ranges of 2, 20 and 200.

STATe *b*
[M_AF:UL:STATe b]
Enables Audio Frequency Meter Upper Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding the Upper Limit sets bit 2 of the Instrument Status Register to one (activating bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Upper Limit and Alarm (**M_AF:ALARM 1**) are enabled, exceeding the Upper Limit activates an audio alarm.

M_AF?

[M_AF?]

Returns an Audio Frequency Meter reading in Hz (0.0 to 200000.0). See also **MEASure:AUDio?** query (6-16).

Reading is invalid if squelch is unbroken for some internal routings.

Example:

```
SCREEN:AF              // Displays AF Meter Operation Screen.
M_AF:RANG:AUTO         // Sets AF Meter Range to Autorange.
M_AF:UL:LEV 100        // Sets AF Meter Upper Limit to 100 kHz.
M_AF:UL:STAT 1         // Enables Upper Limit.
M_AF:LL:LEV 25         // Sets AF Meter Lower Limit to 25 kHz.
M_AF:LL:STAT 1         // Enables Lower Limit.
M_AF:INP:XAUDIO        // Selects EXT MOD IN Connector for AF Meter
                       // Input.
M_AF:FILT:LPAS:FREQ 10    // Sets Low-Pass Filter cutoff frequency to
                          // 10 kHz.
M_AF:FILT:LPAS:STAT 1     // Enables Low-Pass Filter.
M_AF:ALARM 1           // Enables Alarm.
M_AF:PH 1              // Enables Peak Hold feature.
M_AF:PEAK?             // Queries AF Meter Peak reading.
M_AF?                  // Queries AF Meter reading.
```

6-11-2 FREQUENCY ERROR METER COMMANDS

M_RF:

ALARM *b*
[M_RF:ALARM b]
Enables Alarm if *b* is 1, disables if *b* is 0. Enabled Alarm sounds when Upper or Lower Limit is surpassed.

LL:

LEVel *f*
[M_RF:LL:LEVel f]
Sets Lower Limit to *f* kHz. Range of *f* is 0.0000 to 0.1000 for an upper range of 0.1. Range of *f* is 0.000 to 100.000 for upper ranges of 1, 10 and 100.

STATe *b*
[M_RF:LL:STATe b]
Enables Lower Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding the Lower Limit sets bit 5 of the Instrument Status Register to one (activating bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Lower Limit and Alarm (**M_RF:ALARM 1**) are enabled, exceeding the Lower Limit activates an audio alarm.

PEAK?
[M_RF:PEAK?]
Returns Frequency Error Meter Peak reading in Hz (0.0 to 100000.0).

PH *b*
[M_RF:PH b]
Enables Peak Hold Feature if *b* is 1, disables Peak Hold Feature if *b* is 0.

RANGe:

AUTO
[M_RF:RANGe:AUTO]
Sets Frequency Error Meter range to Autorange.

UPPer *f*
[M_RF:RANGe:UPPer f]
Sets Frequency Error Meter range to *f* kHz. Select 0.1, 1, 10 or 100.

Example:
```
M_RF:RANG:UPP 10     // Sets Frequency Error Meter Range to
                     // 10 kHz.
M_RF:PH 1            // Enables Frequency Error Meter Peak Hold.
M_RF:LL:LEV 20.5     // Sets a Lower Limit of 20.5 kHz.
M_RF:LL:STAT 1       // Enables Lower Limit.
M_RF:ALARM 1         // Enables Alarm.
```

RCL *n*
[M_RF:RCL n]
Recalls Frequency Error Meter environment (routings and settings) stored in memory location *n*. Range of *n* is 1 to 9.

M_RF:

RESolution *f*

[M_RF:RESolution f]

Sets Frequency Error Resolution (Gate Time) to *f* Hz. Select 1 (Gate Time of 1 sec) or 10 (Gate Time of 0.1 sec).

STORe *n*

[M_RF:STORe n]

Stores current Frequency Error Meter environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

UL:

LEVel *f*

[M_RF:UL:LEVel f]

Sets Upper Limit to *f* kHz. Range of *f* is 0.0000 to 0.1000 for an upper range of 0.1. Range of *f* is 0.000 to 100.000 for upper ranges of 1, 10 and 100.

STATe *b*

[M_RF:UL:STATe b]

Enables Upper Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding the Upper Limit sets bit 4 of the Instrument Status Register to one (activating bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Upper Limit and Alarm (**M_RF:ALARM 1**) are enabled, exceeding the Upper Limit activates an audio alarm.

M_RF?

[M_RF?]

Returns Frequency Error Meter reading in Hz (-100000.0 to +100000.0). Actual signal frequency can be calculated by converting the Receiver Frequency setting (**RECeiver:FREQuency?** query) from kHz to Hz and adding the Frequency Error Meter reading. For an actual measured frequency reading, see **MEASure:FREQuency?** query (6-16).

Reading is invalid if squelch is unbroken for some internal routings.

Example:

```
SCREEN:FREQ          // Displays Frequency Error Meter Operation
                     // Screen.
M_RF:RANG:AUTO       // Sets Frequency Error Meter Range to Autorange.
M_RF:UL:LEV 100      // Sets Frequency Error Meter Upper Limit to
                     // 100 kHz.
M_RF:UL:STAT 1       // Enables Upper Limit.
M_RF:LL:LEV 25       // Sets Frequency Error Meter Lower Limit to
                     // 25 kHz.
M_RF:LL:STAT 1       // Enables Lower Limit.
M_RF:ALARM 1         // Enables Alarm.
M_RF:PH 1            // Enables Peak Hold feature.
M_RF:PEAK?           // Queries Frequency Error Meter Peak reading.
M_RF?                // Queries Frequency Error Meter reading.
```

6-11-3 POWER METER COMMANDS

M_PWR:

ALARM *b*
[M_PWR:ALARM b]
Enables Power Meter Alarm if *b* is 1, disables if *b* is 0. Enabled Power Meter Alarm sounds when Upper or Lower Limit is surpassed.

DBM[:STATE] *b*
[M_PWR:DBM[:STATE] b]
Enables (*b* = 1) or disables (*b* = 0) the dBm digital readout. The **:STATE** portion of the command is optional.

DBM[:STATE]?
[M_PWR:DBM[:STATE]?]
Returns the state of the dBm digital readout. The **:STATE** portion of the command is optional.

EXT:

STATe *b*
[M_PWR:EXT:STATe b]
Enables External Loss/Gain Offset if *b* is 1, disables if *b* is 0. External Loss/Gain Offset compensates Power Meter readings for external gains or losses.

STATe?
[M_PWR:EXT:STATe?]
Returns External Loss/Gain Offset state setting.

OFFSet *n*
[M_PWR:EXT:OFFSet n]
Sets External Loss/Gain Offset value to *n* dBm. Range of *n* is -99.9 to 99.9. Positive values lower Power Meter readings and compensate for external gains. Negative values raise Power Meter readings and compensate for external losses.

OFFSet?
[M_PWR:EXT:OFFSet?]
Returns External Loss/Gain Offset value in dBm.

LL:

LEVel *n*
[M_PWR:LL:LEVel n]
Sets Power Meter Lower Limit to *n* W. Range of *n* is 0.0000 to 0.5000 for ranges: 0.02, 0.05, 0.1, 0.2 and 0.5. Range of *n* is 0.00 to 200.00 otherwise.

STATe *b*
[M_PWR:LL:STATe b]
Enables Power Meter Lower Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding the Lower Limit sets bit 7 of the Instrument Status Register to one (activating bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Lower Limit and Alarm (**M_PWR:ALARM 1**) are enabled, exceeding the Lower Limit activates an audio alarm.

M_PWR:

PEAK?

[M_PWR:PEAK?]

Returns Power Meter Peak reading in mW (0.0 to 200000.0).

PH *b*

[M_PWR:PH b]

Enables Power Meter Peak Hold Feature if *b* is 1, disables if *b* is 0.

RANGe:

AUTO

[M_PWR:RANGe:AUTO]

Sets Power Meter range to Autorange.

UPPer *n*

[M_PWR:RANGe:UPPer n]

Sets Power Meter range to *n* W. Select from:

0.02	0.05	0.1
0.2	0.5	1
2	5	10
20	50	100
200		

RCL *n*

[M_PWR:RCL n]

Recalls Power Meter environment (routings and settings) stored in memory location *n*. Range of *n* is 1 to 9.

Example:

```
M_PWR:RANG:UPP 2  // Sets Power Meter Range to 2 W.
M_PWR:PH 1        // Enables Power Meter Peak Hold.
M_PWR:LL:LEV 0.75 // Sets a Lower Limit of 0.75 W.
M_PWR:LL:STAT 1   // Enables Lower Limit.
M_PWR:ALARM 1     // Enables Alarm.
```

RF:

ASSUMEd *n*

[M_PWR:RF:ASSUMEd n]

Sets Assumed RF Frequency, in kHz, within 1 MHz of signal measured. Range of *n* is 250.0 to 2010000.0 kHz.

ASSUMEd?

[M_PWR:RF:ASSUMEd?]

Returns Assumed RF Frequency.

STORe *n*

[M_PWR:STORe n]

Stores current Power Meter environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

M_PWR:

TYPE:

CW
[M_PWR:TYPE:CW]
Selects Average Power Measurement Type for Power Meter.

PEAK
[M_PWR:TYPE:PEAK]
Selects Peak Power Measurement Type for Power Meter.

RMS
[M_PWR:TYPE:RMS]
Selects RMS Power Measurement Type for Power Meter.

UL:

LEVel *n*
[M_PWR:UL:LEVel n]
Sets Power Meter Upper Limit to *n* W. Range of *n* is 0.0000 to 0.5000 for upper ranges: 0.02, 0.05, 0.1, 0.2 and 0.5. Range of *n* is 0.00 to 200.00 otherwise.

STATe *b*
[M_PWR:UL:STATe b]
Enables Power Meter Upper Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding the Upper Limit sets bit 6 of the Instrument Status Register to one (activating bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Upper Limit and Alarm (**M_PWR:ALARM 1**) are enabled, exceeding the Upper Limit activates an audio alarm.

M_PWR?
[M_PWR?]
Returns a Power Meter reading in mW (0.0 to 200000.0). See also **MEASure:POWer?** query (6-16).

Example:
```
SCREEN:POW          // Displays Power Meter Operation Screen.
M_PWR:RANG:AUTO     // Sets Power Meter Range to Autorange.
M_PWR:UL:LEV 3      // Sets Power Meter Upper Limit to 3 W.
M_PWR:UL:STAT 1     // Enables Upper Limit.
M_PWR:LL:LEV .5     // Sets Power Meter Lower Limit to 0.5 W.
M_PWR:LL:STAT 1     // Enables Lower Limit.
M_PWR:ALARM 1       // Enables Alarm.
M_PWR:PH 1          // Enables Peak Hold feature.
M_PWR:PEAK?         // Queries Power Meter Peak reading.
M_PWR?              // Queries Power Meter reading.
```

6-11-4 DEVIATION METER (PEAK) COMMANDS

M_DEV:

ALARM *b*
[M_DEV:ALARM b]
Enables Deviation Meter Alarm if *b* is 1, disables if *b* is 0. Enabled Alarm sounds when Upper or Lower Limit is surpassed.

AVErage *b*
[M_DEV:AVErage b]
Enables Deviation Meter Averaging if *b* is 1, disables if *b* is 0.

LL:

LEVel *f*
[M_DEV:LL:LEVel f]
Sets the Deviation Meter Lower Limit to *f* kHz. *f* can vary from 0.00 to 20.00 for range values of 2, 5, 10 or 20 kHz. f varies from 0 to 100 otherwise.

STATe *b*
[M_DEV:LL:STATe b]
Enables Deviation Meter Lower Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding the Lower Limit sets bit 9 of the Instrument Status Register to one (activating bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Lower Limit and Alarm (**M_DEV:ALARM 1**) are enabled, exceeding the Lower Limit activates an audio alarm.

MODE:

BOTH
[M_DEV:MODE:BOTH]
Selects Both Mode for Deviation Meter, reading positive and negative deviation. Use **M_DEV:POS?** and **M_DEV:NEG?** to query meter readings.

NEGative
[M_DEV:MODE:NEGative]
Selects Negative Mode for Deviation Meter, reading negative deviation. Use **M_DEV:NEG?** to query meter readings.

NORMalize
[M_DEV:MODE:NORMalize]
Selects Normalized Mode for Deviation Meter, reading (positive + negative)/2 deviation. Use **M_DEV:POS?** to query meter readings.

POSitive
[M_DEV:MODE:POSitive]
Selects Positive Mode for Deviation Meter, reading positive deviation. Use **M_DEV:POS?** to query meter readings.

Deviation Meter Screen must be updated for a Deviation Meter Mode change to take effect.

```
Example: M_DEV:LL:LEV 10        // Sets a Lower Limit of 10 kHz.
         M_DEV:LL:STAT 1        // Enables Lower Limit.
         M_DEV:ALARM 1          // Enables Alarm.
         M_DEV:MODE:POS         // Selects the Positive Mode of the
                                // Deviation (Peak) Meter.
         SCREEN:DEV             // Updates the Deviation (Peak) Meter
                                // Operation Screen.
```

M_DEV:

MODE:NEG?

[M_DEV:NEG?]

Returns a negative Deviation Meter reading as an absolute value in kHz (0.00 to 100.00).

PEAK:

NEG?

[M_DEV:PEAK:NEG?]

Returns a negative Deviation Meter Peak reading as an absolute value in kHz (0.00 to 100.00).

POS?

[M_DEV:PEAK:POS?]

Returns a positive Deviation Meter Peak reading as an absolute value in kHz (0.00 to 100.00).

PH *b*

[M_DEV:PH b]

Enables the Deviation Meter Peak Hold Feature if *b* is 1, disables if *b* is 0.

POS?

[M_DEV:POS?]

Returns a positive Deviation Meter reading as an absolute value in kHz (0.00 to 100.00).

RANGe:

AUTO

[M_DEV:RANGe:AUTO]

Sets Deviation Meter range to Autorange.

UPPer *f*

[M_DEV:RANGe:UPPer f]

Sets Deviation Meter range to *f* kHz. Settings for *f* are: 2, 5, 10, 20, 50 or 100.

```
Example: M_DEV:MODE:BOTH        // Selects Both Mode.
         SCREEN:DEV             // Updates the Deviation (Peak) Meter
                                // Operation Screen.
         M_DEV:RANG:UPP 10      // Sets the Range to 10 kHz.
         M_DEV:PH 1             // Enables Peak Hold feature.
```

M_DEV:

RCL *n*

[M_DEV:RCL n]

Recalls Deviation Meter environment (routings and settings) stored in memory location *n*. Range of *n* is 1 to 9.

STORe *n*

[M_DEV:STORe n]

Stores current Deviation Meter environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

UL:

LEVel *f*

[M_DEV:UL:LEVel f]

Sets Deviation Meter Upper Limit to *f* kHz. *f* can vary from 0.00 to 20.00 for range values of 2, 5, 10 or 20 kHz. *f* varies from 0 to 100 otherwise.

STATe *b*

[M_DEV:UL:STATe b]

Enables Deviation Meter Upper Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding the Upper Limit sets bit 8 of the Instrument Status Register to one (activating bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Upper Limit and Alarm (**M_DEV:ALARM 1**) are enabled, exceeding the Upper Limit activates an audio alarm.

Example:
```
SCREEN:DEV             // Displays Deviation (Peak) Meter
                       // Operation Screen.
M_DEV:RANG:AUTO        // Sets Range to Autorange.
M_DEV:UL:LEV 30        // Sets Upper Limit to 30 kHz.
M_DEV:UL:STAT 1        // Enables Upper Limit.
M_DEV:LL:LEV 5         // Sets Lower Limit to 5 kHz.
M_DEV:LL:STAT 1        // Enables Lower Limit.
M_DEV:ALARM 1          // Enables Alarm.
M_DEV:PH 1             // Enables Peak Hold feature.
M_DEV:AVE 1            // Enables Deviation (Peak) Meter averaging.
M_DEV:MODE:NEG         // Selects the Negative Mode.
SCREEN:DEV             // Updates the Deviation (Peak) Meter
                       // Operation Screen.
M_DEV:NEG?             // Queries Negative Deviation (Peak) Meter
                       // reading.
```

6-11-5 MODULATION METER COMMANDS

The Modulation Meter measures AM.

M_MOD:

ALARM *b*
[M_MOD:ALARM b]
Enables Modulation Meter Alarm if *b* is 1, disables if *b* is 0. Enabled Alarm sounds when Upper or Lower Limit is surpassed.

LL:

LEVel *n*
[M_MOD:LL:LEVel n]
Sets Modulation Meter Lower Limit to *n*%. Range of *n* is 0.0 to 100.0.

STATe *b*
[M_MOD:LL:STATe b]
Enables Modulation Meter Lower Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding the Lower Limit sets bit 11 of the Instrument Status Register to one (activating bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Lower Limit and Alarm (**M_MOD:ALARM 1**) are enabled, exceeding the Lower Limit activates an audio alarm.

PEAK?
[M_MOD:PEAK?]
Returns Modulation Meter Peak reading as a percentage (0.0 to 100.0).

PH *b*
[M_MOD:PH b]
Enables Modulation Meter Peak Hold Feature if *b* is 1, disables if *b* is 0.

RANGe:

AUTO
[M_MOD:RANGe:AUTO]
Sets Modulation Meter range to Autorange.

UPPer *n*
[M_MOD:RANGe:UPPer n]
Sets Modulation Meter range to *n*%. Select 40 or 100.

Example:
```
M_MOD:RANG:UPP 100    // Sets AM Modulation Meter Range to 100%.
M_MOD:PH 1            // Enables AM Modulation Meter Peak Hold.
M_MOD:LL:LEV 20.5     // Sets a Lower Limit of 20.5%.
M_MOD:LL:STAT 1       // Enables Lower Limit.
M_MOD:ALARM 1         // Enables Alarm.
```

M_MOD:

RCL *n*

[M_MOD:RCL n]

Recalls Modulation Meter environment (routings and settings) stored in memory location *n*. Range of *n* is 1 to 9.

STORe *n*

[M_MOD:STORe n]

Stores current Modulation Meter environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

UL:

LEVel *n*

[M_MOD:UL:LEVel n]

Sets Modulation Meter Upper Limit to *n*%. Range of *n* is 0.0 to 100.0.

STATe *b*

[M_MOD:UL:STATe b]

Enables Modulation Meter Upper Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding the Upper Limit sets bit 10 of the Instrument Status Register to one (activating bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Upper Limit and Alarm (**M_MOD:ALARM 1**) are enabled, exceeding the Upper Limit activates an audio alarm.

M_MOD?

[M_MOD?]

Returns a Modulation Meter reading as a percentage (0.0 to 100.0).

Example:

```
SCREEN:MOD          // Displays Modulation Meter Operation Screen.
M_MOD:RANG:AUTO     // Sets Modulation Meter Range to Autorange.
M_MOD:UL:LEV 30     // Sets Modulation Meter Upper Limit to 30%.
M_MOD:UL:STAT 1     // Enables Upper Limit.
M_MOD:LL:LEV 5      // Sets AM Modulation Meter Lower Limit to 5%.
M_MOD:LL:STAT 1     // Enables Lower Limit.
M_MOD:ALARM 1       // Enables Alarm.
M_MOD:PH 1          // Enables Peak Hold feature.
M_MOD:PEAK?         // Queries Modulation Meter Peak reading.
M_MOD?              // Queries Modulation Meter reading.
```

6-11-6 DISTORTION METER COMMANDS

M_DIST:

ALARM *b*
[M_DIST:ALARM b]
Enables Distortion Meter Alarm if *b* is 1, disables if *b* is 0. Enabled Alarm sounds when Upper or Lower Limit is surpassed.

AVErage *b*
[M_DIST:AVErage b]
Enables Average Feature if *b* is 1, disables if *b* is 0.

FILTer *f*
[M_DIST:FILTer f]
Sets Notch Filter frequency to *f* Hz. Range of *f* is 600 to 1400.

INPut:*type*
[M_DIST:INPut:type]
Sets Distortion Meter Input to *type*. Select DEMOD (Demod Audio), SINAD (SINAD/BER), XAUDio (Ext Mod) or FGEN (Func Gen).

LL:

LEVel *n*
[M_DIST:LL:LEVel n]
Sets Lower Limit to *n*%. Range of *n* is 0.0 to 20.0.

STATe *b*
[M_DIST:LL:STATe b]
Enables Distortion Meter Lower Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding the Lower Limit sets bit 13 of the Instrument Status Register to one (activating bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Lower Limit and Alarm (**M_DIST:ALARM 1**) are enabled, exceeding the Lower Limit activates an audio alarm.

Example:
```
M_DIST:INP:XAUD       // Selects EXT MOD IN Connector for
                      // Distortion Meter Input.
M_DIST:FILT 770       // Sets Notch Filter frequency to 770 Hz.
M_DIST:PH 1           // Enables Distortion Meter Peak Hold.
M_DIST:LL:LEV 5.3     // Sets a Lower Limit of 5.3 kHz.
M_DIST:LL:STAT 1      // Enables Lower Limit.
M_DIST:ALARM 1        // Enables Alarm.
```

PEAK?
[M_DIST:PEAK?]
Returns a Distortion Meter Peak reading as a percentage (0.0 to 20.0).

PH *b*
[M_DIST:PH b]
Enables Distortion Meter Peak Hold Feature if *b* is 1, disables if *b* is 0. Peak Hold takes effect only after **SCREEN:DISTortion** command.

M_DIST:

RCL n
[M_DIST:RCL n]

Recalls Distortion Meter environment (routings and settings) stored at memory location *n*. Range of *n* is 1 to 9.

STORe *n*
[M_DIST:STORe n]

Stores current Distortion Meter environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

SELect:

CWEight
[M_DIST:SELect:CWEight]

Selects the C-Weight Filter.

CMESsage
[M_DIST:SELect:CMESsage]

Same as **M_DIST:SELect:CWEight** command.

LPASs *f*
[M_DIST:SELect:LPASs f]

Selects a Low-Pass Filter with cutoff frequency of *f* Hz. Range of *f* is 100 to 30000.

Example:
```
M_DIST:SELECT:LPAS 15000  // Selects a Low-Pass Filter with a
                          // 15 kHz cutoff frequency.
```

UL:

LEVel *n*
[M_DIST:UL:LEVel n]

Sets Upper Limit to *n*%. Range of *n* is 0.0 to 20.0.

STATe *b*
[M_DIST:UL:STATe b]

Enables Distortion Meter Upper Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding the Upper Limit sets bit 12 of the Instrument Status Register to one (activating bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Upper Limit and Alarm (**M_DIST:ALARM 1**) are enabled, exceeding the Upper Limit activates an audio alarm.

M_DIST?

[M_DIST?]

Returns a Distortion Meter reading as a percentage (0.0 to 20.0).

Example:

```
SCREEN:DIST              // Displays Distortion Meter Operation Screen.
M_DIST:INP:DEMOD         // Selects Demodulated Audio as the Distortion
                         // Meter Input.
M_DIST:FILT 1000         // Sets Notch Filter frequency to 1 kHz.
M_DIST:UL:LEV 15         // Sets Distortion Meter Upper Limit to 15%.
M_DIST:UL:STAT 1         // Enables Upper Limit.
M_DIST:LL:LEV 4          // Sets Distortion Meter Lower Limit to 4%.
M_DIST:LL:STAT 1         // Enables Lower Limit.
M_DIST:ALARM 1           // Enables Alarm.
M_DIST:SELECT:LPAS 20000 // Selects a Low-Pass Filter with a 20 kHz
                            // cutoff frequency.
M_DIST:AVE 1             // Enables Averaging.
M_DIST:PH 1              // Enables Peak Hold feature.
M_DIST:PEAK?             // Queries Distortion Meter Peak reading.
M_DIST?                  // Queries Distortion Meter reading.
```

6-11-7 SINAD METER COMMANDS

M_SINAD:

ALARM *b*
[M_SINAD:ALARM b]
Enables SINAD Meter Alarm if *b* is 1, disables if *b* is 0. Enabled Alarm sounds when Upper or Lower Limit is surpassed.

AVErage *b*
[M_SINAD:AVErage b]
Enables Average Feature if *b* is 1, disables if *b* is 0.

FILTer *f*
[M_SINAD:FILTer f]
Sets Notch frequency to *f* Hz. Range of *f* is 600 to 1400.

INPut:*type*
[M_SINAD:INPut:type]
Selects *type* as the SINAD Meter Input. Select DEMOD (Demod Audio), SINAD (SINAD/BER), XAUDio (Ext Mod) or FGEN (Func Gen).

LL:

LEVel *n*
[M_SINAD:LL:LEVel n]
Sets SINAD Meter Lower Limit to *n* dB. Range of *n* is 3.0 to 40.0.

STATe *b*
[M_SINAD:LL:STATe b]
Enables SINAD Meter Lower Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding Lower Limit sets bit 1 of the Instrument Summary Status Register to one (activating bit 1 of the Instrument Status Register, bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Lower Limit and Alarm (**M_SINAD:ALARM 1**) are enabled, exceeding the Lower Limit activates an audio alarm.

Example:
```
M_SINAD:INP:XAUD       // Selects EXT MOD IN Connector for
                       // Distortion Meter Input.
M-SINAD:FILT 850       // Sets Notch Filter frequency to 850 Hz.
M_SINAD:PH 1           // Enables Distortion Meter Peak Hold.
M_SINAD:LL:LEV 4       // Sets a Lower Limit of 4 dB.
M_SINAD:LL:STAT 1      // Enables Lower Limit.
```

PEAK?
[M_SINAD:PEAK?]
Returns a SINAD Meter Peak reading in dB (3.0 to 40.0).

PH *b*
[M_SINAD:PH b]
Enables SINAD Meter Peak Hold Feature if *b* is 1, disables if *b* is 0.

RCL *n*
[M_SINAD:RCL n]
Recalls SINAD Meter environment (routings and settings) stored in memory location *n*. Range of *n* is 1 to 9.

M_SINAD:

RESolution *n*

[M_SINAD:RESolution n]

Sets SINAD Meter readout resolution to *n* dB. *n* is 0.1 or 0.5.

RESolution?

[M_SINAD:RESolution?]

Returns current resolution setting in dB (0.1 or 0.5).

SELect:

CWEight

[M_SINAD:SELect:CWEight]

Selects C-Weight Filter.

CMESsage

[M_SINAD:SELect:CMESsage]

Same as **M_SINAD:SELect:CWEight** command.

LPASs *f*

[M_SINAD:SELect:LPASs f]

Selects Low-Pass Filter with cutoff frequency of *f* Hz. Range of *f* is 100 to 30000.

STORe *n*

[M_SINAD:STORe n]

Stores current SINAD Meter environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

UL:

LEVel *n*

[M_SINAD:UL:LEVel n]

Sets SINAD Meter Upper Limit to *n* dB. Range of *n* is 3.0 to 40.0.

STATe *b*

[M_SINAD:UL:STATe b]

Enables SINAD Meter Upper Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding Upper Limit sets bit 0 of the Instrument Summary Status Register to one (activating bit 1 of the Instrument Status Register, bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Upper Limit and Alarm (**M_SINAD:ALARM 1**) are enabled, exceeding the Upper Limit activates an audio alarm.

M_SINAD?

[M_SINAD?]

Returns a SINAD Meter reading in dB (3.0 to 40.0).

Example:

```
SCREEN:SINAD            // Displays SINAD Meter Operation Screen.
M_SINAD:INP:DEMOD       // Selects Demodulated Audio as the
                        // SINAD Meter.
M_SINAD:FILT 1100       // Sets Notch Filter frequency to 1100 Hz.
M_SINAD:UL:LEV 15       // Sets SINAD Meter Upper Limit to 15 dB.
M_SINAD:UL:STAT 1       // Enables Upper Limit.
M_SINAD:LL:LEV 4        // Sets SINAD Meter Lower Limit to 4 dB.
M_SINAD:LL:STAT 1       // Enables Lower Limit.
M_SINAD:SELECT:LPAS 15000// Selects a Low-Pass Filter with a 15 kHz
                           // cutoff frequency.
M_SINAD:AVE 1           // Enables Averaging.
M_SINAD:PH 1            // Enables Peak Hold feature.
M_SINAD:PEAK?           // Queries SINAD Meter Peak reading.
M_SINAD?                // Queries SINAD Meter reading.
```

6-11-8 SIGNAL STRENGTH METER COMMANDS

M_SIG:

PEAK?
[M_SIG:PEAK?]
Returns a Signal Strength Meter Peak reading (0 to 100).

PH *b*
[M_SIG:PH b]
Enables Peak Hold Feature if *b* is 1, disables if *b* is 0.

RCL *n*
[M_SIG:RCL n]
Recalls Signal Strength Meter environment (routings and settings) stored in memory location *n*. Range of *n* is 1 to 9.

STORe *n*
[M_SIG:STORe n]
Stores current Signal Strength Meter environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

M_SIG?
[M_SIG?]
Returns a Signal Strength Meter reading (0 to 100).

Example:
```
M-SIG:PH 1          // Enables Peak Hold feature.
M_SIG:PEAK?         // Queries Signal Strength Meter Peak reading.
M_SIG?              // Queries Signal Strength Meter reading.
```

6-11-9 BIT ERROR RATE (BER) METER COMMANDS

M_BER:

PATtern:

FIXED
[M_BER:PATtern:FIXED]
Selects Fixed pattern for the BER Meter test data.

RANDom
[M_BER:PATtern:RANDom]
Selects Random pattern for BER Meter test data.

USER *nn*
[M_BER:PATtern:USER nn]
Selects a USER data pattern (could be a variable or expression) for the BER Meter test data using the specified 8 bit pattern *nn*. Patterns entered that are not base 10 are preceded with # character and letter signifying the number base: H (Hexadecimal), B (Binary) or Q (Octal). Pattern is displayed in Hexadecimal.

Example:
```
M_BER:PAT:USER #Q216   // Selects a User Pattern of 8E hexadecimal
                       // (216 Octal) for the BER Meter test data.
X=#HC0                 // Assigns variable to a usable pattern.
M_BER:PAT:USER X+3     // Selects a User Pattern of C3 hexadecimal.
```

USER?
[M_BER:PATtern:FIXEDUSER?]
Returns the specified USER data pattern.

POLarity:

NEGative
[M_BER:POLarity:NEGative]
Selects Negative Polarity for the BER Meter.

POSitive
[M_BER:POLarity:POSitive]
Selects Positive Polarity for the BER Meter.

POLarity?
[M_BER:POLarity?]
Returns the selected data Polarity. Returns POSITIVE or NEGATIVE.

RATE *n*
[M_BER:RATE n]
Sets BER Meter rate to *n*. Select from:

75	150	300
600	1200	2400
4800	16000	

RATE?
[M_BER:RATE?]
Returns the BER Meter Rate setting.

M_BER:

RCL *n*

[M_BER:RCL n]

Recalls the BER Meter environment (routings and settings) stored in memory location *n*. Range of *n* is 1 to 9.

SIZE *n*

[M_BER:SIZE n]

Sets BER Meter block size in bits to *n*. Range of *n* is 100 to 100000.

Example:
```
M_BER:POL:NEG       // Selects Negative Polarity for the BER Meter.
M_BER:RATE 1200     // Sets the data rate to 1200 bps.
M_BER:SIZE 1000     // Sets the data pattern size to 1000 bits.
```

SIZE?

[M_BER:SIZE?]

Returns the BER Meter block size setting.

STORe *n*

[M_BER:STORe n]

Stores current BER Meter environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

TYPE:*xxx*

[M_BER:TYPE:xxx]

Selects *xxx* for the Bit Error Rate Type. Select GENerator, RECeiver, DUPlex or BASEband.

M_BER?

[M_BER?]

Returns the number of errors for the last pass.

Example:
```
M_BER:TYPE:DUP        // Selects Duplex for BER Meter Type.
M_BER:RATE 2400       // Sets BER Meter data rate to 2400 bps.
M_BER:SIZE 10000      // Sets BER Meter Pattern Size to 10,000 bits.
M_BER:POL:POS         // Sets BER Meter Polarity to Positive.
M_BER:PAT:RAND        // Selects the Random Pattern Type for the BER
                      // Meter test.
M_BER?                // Queries a bit error rate reading.
```

6-11-10 DIGITAL MULTIMETER COMMANDS

M_DMM:

ALARM *b*
[M_DMM:ALARM b]
Enables Multimeter Alarm if *b* is 1, disables if *b* is 0. Enabled Alarm sounds when Upper or Lower Limit is surpassed.

FUNCtion:

CURRent:

AC
[M_DMM:FUNCtion:CURRent:AC]
Selects the AC Ammeter for the Multimeter Function.

DC
[M_DMM:FUNCtion:CURRent:DC]
Selects the DC Ammeter for the Multimeter Function.

RESistance
[M_DMM:FUNCtion:RESistance]
Selects the Ohmmeter for the Multimeter Function.

VOLTage:

AC
[M_DMM:FUNCtion:VOLTage:AC]
Selects the AC Voltmeter for the Multimeter Function.

DC
[M_DMM:FUNCtion:VOLTage:DC]
Selects the DC Voltmeter for the Multimeter Function.

FUNCtion?
[M_DMM:FUNCtion?]
Returns active Multimeter Function (CURR:AC, CURR:DC, RES, VOLT:AC or VOLT:DC).

INPut:IMPedance *n*
[M_DMM:INPut:IMPedance n]
Sets Input Impedance to *n* ohms. Select 150, 600 or 1e6. Command valid only in AC Voltmeter Function.

Example:
```
M_DMM:FUNC:CURR:DC      // Selects DC Ammeter for the Multimeter
                        // Function.
M_DMM:FUNC?             // Queries Multimeter Function.  DCC is
                        // returned.
M_DMM:FUNC:VOLT:AC      // Selects AC Voltmeter for the Multimeter
                        // Function.
M_DMM:INP:IMP 1E6       // Sets Input Impedance at 1 MΩ.
```

M_DMM:

LL:

LEVel *n*
[M_DMM:LL:LEVel n]
Sets the Multimeters Lower Limit to *n*. Table 6-1 lists units and ranges for *n*.

FUNCTION	ACC, DCC		ACV, DCV		OHMMETER	
Range	20 and 200 mA	2 A and 20 A	200 mV	2, 20, 200, 2000 V	200 Ω	2 kΩ to 20 MΩ
n Units	A		V		kΩ	
Range of *n*	0.00000 to 0.19990	0.000 to 19.990	0.0000 to 0.1999	0.00 to 1000.00	0.0000 to 0.1999	0.000 to 19990

Table 6-1 DMM Upper and Lower Limit Ranges and Units

STATe *b*
[M_DMM:LL:STATe b]
Enables the Multimeters Lower Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding Lower Limit sets bit 3 of the Instrument Summary Status Register to one (activating bit 1 of the Instrument Status Register, bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Lower Limit and Alarm (**M_DMM:ALARM 1**) are enabled, exceeding the Lower Limit activates an audio alarm.

PH *b*
[M_DMM:PH b]
Enables Multimeter Peak Hold feature if *b* is 1, disables if *b* is 0.

RANGe:

AUTO
[M_DMM:RANGe:AUTO]
Sets the Multimeter range to Autorange.

UPPer *n*
[M_DMM:RANGe:UPPer n]
For AC or DC Voltmeter, sets range to *n* volts. Settings for *n* are: 0.2, 2, 20, 200 or 2000.

For AC or DC Ammeter, sets range to *n* amps. Settings for *n* are: 0.02, 0.2, 2 or 20.

For Ohmmeter, sets range to *n* kΩ. Settings for *n* are: 0.2, 2, 20, 200, 2000 or 20000.

> M_DMM:RANGe:AUTO and M_DMM:RANGe:UPPer commands must be followed by a SCREEN:DMM command.

Example:
```
M_DMM:FUNC:CURR:AC     // Selects AC Ammeter for the Multimeter
                       // Function.
M_DMM:UPP 2            // Sets Ammeter range to 2 A.
M_DMM:FUNC:VOLT:AC     // Selects AC Voltmeter for the Multimeter
                       // Function.
M_DMM:RANG:UPP .2      // Sets Ohmmeter range to 200 Ω.
```

M_DMM:

RCL *n*
[M_DMM:RCL n]
Recalls Digital Multimeter environment (routings and settings) stored in memory location *n*. Range of *n* is 1 to 9.

STORe *n*
[M_DMM:STORe n]
Stores current Digital Multimeter environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

UL:

LEVel *n*
[M_DMM:UL:LEVel n]
Sets Multimeters Upper Limit to *n*. Table 6-1 lists units and ranges for *n*.

STATe *b*
[M_DMM:UL:STATe b]
Enables Multimeters Upper Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding Upper Limit sets bit 2 of the Instrument Summary Status Register to one (activating bit 1 of the Instrument Status Register, bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Upper Limit and Alarm (**M_DMM:ALARM 1**) are enabled, exceeding the Upper Limit activates an audio alarm.

M_DMM?
[M_DMM?]
Returns a Multimeter reading in the current Function units according to Table 6-1. See also **MEASure:** commands (6-16).

Example:
```
M_DMM:FUNC:CURR:DC    // Selects DC Ammeter for the Multimeter
                      // Function.
M_DMM:RANG:LEV 0.2    // Sets Ammeter range to 200 mA.
M_DMM:PH 1            // Enables Peak Hold feature.
M_DMM:ALARM 1         // Enables Alarm.
M_DMM:UL:LEV 150      // Sets Upper Limit to 150 mA.
M_DMM:UL:STAT 1       // Enables Upper Limit.
M_DMM:LL:LEV 30       // Sets Lower Limit to 30 mA.
M_DMM:LL:STAT 1       // Enables Lower Limit.
```

6-11-11 PHASE METER COMMANDS

M_PM:

ALARM *b*

[M_PM:ALARM b]

Enables Phase Meter Alarm if *b* is 1, disables if *b* is 0. Enabled Alarm sounds when a limit is surpassed.

LL:

LEVel *n*

[M_PM:LL:LEVel n]

Sets the Phase Meter Lower Limit to *n* radians. Range of *n* is 0.00 to 10.00.

STATe *b*

[M_PM:LL:STATe b]

Enables the Phase Meter Lower Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding Lower Limit sets bit 5 of the Instrument Summary Status Register to one (activating bit 1 of the Instrument Status Register, bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Lower Limit and Alarm (**M_PM:ALARM 1**) are enabled, exceeding the Lower Limit activates an audio alarm.

PH *b*

[M_PM:PH b]

Enables the Phase Meter Peak Hold Feature if *b* is 1, disables if *b* is 0.

RANGe:

AUTO

[M_PM:RANGe:AUTO]

Sets Phase Meter range to Autorange.

UPPer *n*

[M_PM:RANGe:UPPer n]

Sets Phase Meter range to *n* radians. Select 1, 5 or 10.

Example:
```
M_PM:RANG:UPP 5       // Sets Phase Meter Range to 5 radians.
M_PM:PH 1             // Enables Phase Meter Peak Hold.
M_PM:LL:LEV 1.65      // Sets a Lower Limit of 1.65 radians.
M_PM:LL:STAT 1        // Enables Lower Limit.
M_PM:ALARM 1          // Enables Alarm.
```

RCL *n*

[M_PM:RCL n]

Recalls Phase Meter environment (routings and settings) stored in memory location *n*. Range of *n* is 1 to 9.

M_PM:

STORe *n*

[M_PM:STORe n]

Stores current Phase Meter environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

UL:

LEVel *n*

[M_PM:UL:LEVel n]

Sets Phase Meter Upper Limit to *n* radians. Range of *n* is 0.00 to 10.00.

STATe *b*

[M_PM:UL:STATe b]

Enables Phase Meter Upper Limit if *b* is 1, disables if *b* is 0. When enabled, exceeding Upper Limit sets bit 4 of the Instrument Summary Status Register to one (activating bit 1 of the Instrument Status Register, bit 13 of the Questionable Status Register and bit 3 of the Status Byte) (see 2-16). When the Upper Limit and Alarm (**M_PM:ALARM 1**) are enabled, exceeding the Upper Limit activates an audio alarm.

M_PM?

[M_PM?]

Returns a Phase Meter reading in radians (0.00 to 10.00).

Example:

```
SCREEN:PM          // Displays Phase Meter Operation Screen.
M_PM:RANG:AUTO     // Sets Phase Meter Range to Autorange.
M_PM:UL:LEV 7      // Sets Phase Meter Upper Limit to 7 radians.
M_PM:UL:STAT 1     // Enables Upper Limit.
M_PM:LL:LEV .5     // Sets Phase Meter Lower Limit to 0.5 radians.
M_PM:LL:STAT 1     // Enables Lower Limit.
M_PM:ALARM 1       // Enables Alarm.
M_PM:PH 1          // Enables Peak Hold feature.
M_PM:PEAK?         // Queries Phase Meter Peak reading.
M_PM?              // Queries Phase Meter reading.
```

6-11-12 DEVIATION METER (RMS) COMMANDS

M_DRMS:

ALARM *b*
[M_DRMS:ALARM b]
Enables Deviation Meter (RMS) Alarm if *b* is 1, disables if *b* is 0. Enabled Alarm sounds when a limit is surpassed.

AVErage *b*
[M_DRMS:AVErage b]
Enables Deviation Meter (RMS) Averaging if *b* is 1, disables if *b* is 0.

LL:

LEVel *f*
[M_DRMS:LL:LEVel f]
Sets Deviation (RMS) Lower Limit to *f* kHz. Range of *f* is 0.00 to 10.00.

STATe *b*
[M_DRMS:LL:STATe b]
Enables Deviation (RMS) Meter Lower Limit if *b* is 1, disables if *b* is 0. When the Lower Limit and Alarm (**M_DRMS:ALARM 1**) are enabled, exceeding the Lower Limit activates an audio alarm.

PH *b*
[M_DRMS:PH b]
Enables Deviation Meter (RMS) Peak Hold feature if *b* is 1, disables if *b* is 0.

RANGe:

AUTO
[M_DRMS:RANGe:AUTO]
Sets Deviation Meter (RMS) range to Autorange.

UPPer *f*
[M_DRMS:RANGe:UPPer f]
Sets Deviation Meter (RMS) range to *f* kHz. Select 2, 5 or 10.

Example:
```
M_DRMS:RANG:UPP 10     // Sets Deviation Meter (RMS) Range to
                       // 10 kHz.
M_DRMS:PH 1            // Enables Deviation Meter (RMS) Peak Hold.
M_DRMS:LL:LEV 1.5      // Sets a Lower Limit of 1.5 kHz.
M_DRMS:LL:STAT 1       // Enables Lower Limit.
M_DRMS:ALARM 1         // Enables Alarm.
```

RCL *n*
[M_DRMS:RCL n]
Recalls Deviation Meter (RMS) environment (routings and settings) stored in memory location *n*. Range of *n* is 1 to 9.

STORe *n*
[M_DRMS:STORe n]
Stores current Deviation Meter (RMS) environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

M_DRMS:

UL:

LEVel *f*

[M_DRMS:UL:LEVel f]

Sets Deviation (RMS) Upper Limit to *f* kHz. Range of *f* is 0.00 to 10.00.

STATe *b*

[M_DRMS:UL:STATe b]

Enables Deviation Meter (RMS) Upper Limit if *b* is 1, disables if *b* is 0. When the Upper Limit and Alarm (**M_DRMS:ALARM 1**) are enabled, exceeding the Upper Limit activates an audio alarm.

M_DRMS?

[M_DRMS?]

Returns a Deviation Meter (RMS) reading in kHz (0.00 to 10.00).

Example:

```
SCREEN:DRMS          // Displays Deviation Meter (RMS) Operation
                     // Screen.
M_DRMS:RANG:AUTO     // Sets Deviation Meter (RMS) Range to Autorange.
M_DRMS:UL:LEV 8.4    // Sets Deviation Meter (RMS) Upper Limit
                     // to 8.4 kHz.
M_DRMS:UL:STAT 1     // Enables Upper Limit.
M_DRMS:LL:LEV 2      // Sets Deviation Meter (RMS) Lower Limit
                     // to 2 kHz.
M_DRMS:LL:STAT 1     // Enables Lower Limit.
M_DRMS:ALARM 1       // Enables Alarm.
M_DRMS:PH 1          // Enables Peak Hold feature.
M_DRMS:AVE 1         // Enables Averaging.
M_DRMS?              // Queries Deviation Meter (RMS) reading.
```

6-11-13 PHASE METER (RMS) COMMANDS

M_PMRMS:

ALARM *b*
[M_PMRMS:ALARM b]
Enables Phase Meter (RMS) Alarm if *b* is 1, disables Alarm if *b* is 0. Enabled Alarm sounds when a limit is surpassed.

AVErage *b*
[M_PMRMS:AVErage b]
Enables (*b* = 1) or disables (*b* = 0) the Averaging mode.

LL:

LEVel *n*
[M_PMRMS:LL:LEVel n]
Sets Phase Meter (RMS) Lower Limit to *n* radians. Range of *n* is 0.00 to 10.00.

STATe *b*
[M_PMRMS:LL:STATe b]
Enables Phase Meter (RMS) Lower Limit if *b* is 1, disables Lower Limit if *b* is 0. When the Lower Limit and Alarm (**M_PMRMS:ALARM 1**) are enabled, exceeding the Lower Limit activates an audio alarm.

PH *b*
[M_PMRMS:PH b]
Enables Phase Meter (RMS) Peak Hold feature if *b* is 1, disables Peak Hold feature if *b* is 0.

RANGe:

AUTO
[M_PMRMS:RANGe:AUTO]
Sets Phase Meter (RMS) range to Autorange.

UPPer *n*
[M_PMRMS:RANGe:UPPer n]
Sets Phase Meter (RMS) range to *n* radians. Select 1, 5 or 10.

RCL *n*
[M_PMRMS:RCL n]
Recalls Phase Meter (RMS) environment (routings and settings) stored in memory location *n*. Range of *n* is 1 to 9.

STORe *n*
[M_PMRMS:STORe n]
Stores current Phase Meter (RMS) environment (routings and settings) in memory location *n*. Range of *n* is 1 to 9.

Example:
```
M_PMRMS:RANG:AUTO // Sets Phase Meter (RMS) range to Autorange.
M_PMRMS:PH 1      // Enables Phase Meter (RMS) Peak Hold.
M_PMRMS:STOR 3    // Stores state of Phase Meter (RMS) to
                  // Phase Meter (RMS) memory location 3.
```

M_PMRMS:

UL:

LEVel *n*
[M_PMRMS:UL:LEVel n]
Sets Phase Meter (RMS) Upper Limit to *n* radians. Range of *n* is 0.00 to 10.00.

STATe *b*
[M_PMRMS:UL:STATe b]
Enables Phase Meter (RMS) Upper Limit if *b* is 1, disables Upper Limit if *b* is 0. When the Upper Limit and Alarm (**M_PMRMS:ALARM 1**) are enabled, exceeding the Upper Limit activates an audio alarm.

M_PMRMS?
[M_PMRMS?]
Returns a Phase Meter (RMS) reading in radians (0.00 to 10.00).

Example:
```
SCREEN:PMRMS          // Displays Phase Meter (RMS) Operation Screen.
M_PMRMS:UPP 5         // Sets Phase Meter (RMS) Range to 5 radians.
M_PMRMS:UL:LEV 4.6    // Sets Phase Meter (RMS) Upper Limit to
                      // 4.6 radians.
M_PMRMS:UL:STAT 1     // Enables Phase Meter (RMS) Upper Limit.
M_PMRMS?              // Queries Phase Meter (RMS) reading.
```

6-11-14 AF LEVEL METER COMMAND

M_VRMS?
[M_VRMS?]
Returns a Voltage RMS reading of the received AF Level (0.00 to 10.00).

6-12 ACCESSORY COMMANDS

The Accessory commands are used to remotely control the MIC/ACC Connector (see Figure 3-1 in the IFR-1900 Operation Manual).

ACCessory:

INIT
[ACCessory:INIT]
Initializes MIC/ACC Connector prior to conducting communication. PTT (Press to Talk) is turned On (low). See **PTT:STATe**.

IO
[ACCessory:IO]
Sends/receives data through the MIC/ACC Connector.

RELease
[ACCessory:RELease]
"Releases" MIC/ACC Connector. PTT is turned Off (high).

STATe?
[ACCessory:STATe?]
Checks presence of MCC/ACC I/O device. 0 = none; 1 = device present.

6-13 CELLULAR AMPS/NAMPS

The AMPS/NAMPS[1] Cell Site Monitor and Cellular Simulation is used to test Analog AMPS and NAMPS Cell Sites and Mobile Phones.

The following commands remotely direct the Test Set to display some of the main screens that make up the Cell Site Monitor and Cellular Simulation.

SCREEN:

The following commands remotely direct the Test Set to display some of the main screens that comprise the Cell Site Monitor and Cellular Simulation.

CELL
[SCREEN:CELL]
Displays Cell Site Monitor Forward Control Channel Screen.

GENCELLular
[SCREEN:GENCELLular]
Displays Cellular Simulation Main Menu.

GENFOCC
[SCREEN:GENFOCC]
Displays Cell Site Simulation Forward Control Channel Screen.

GENRECC
[SCREEN:GENRECC]
Displays Mobile Simulation Reverse Control Channel Screen.

GENFVC
[SCREEN:GENFVC]
Displays Cell Site Simulation Forward Voice Channel Screen.

GENRVC
[SCREEN:GENRVC]
Displays Mobile Simulation Reverse Voice Channel Screen.

1. AMPS and NAMPS protocols are used; however, frequencies are extended to include all of North American Digital Cellular (NADC) frequencies. NADC frequencies consist of the following: 800 MHz (AMPS), 450 MHz (NT400©) and 1900 MHz (Hyperband).

6-13-1 AMPS/NAMPS CELL SITE MONITOR COMMANDS

CELL:

WORDA
[CELL:WORDA]
Selects decoding of Stream A words.

WORDB
[CELL:WORDB]
Selects decoding of Stream B words.

BOTH
[CELL:BOTH]
Selects decoding of both Stream A and B words.

WORD?
[CELL:WORD?]
Returns current decoding selection: WORDA, WORDB or BOTH.

CHANnel *n*
[CELL:CHANnel n]
Specifies Channel. Range of *n* is 1 to 2047, depending on format

CHANnel?
[CELL:CHANnel?]
Returns current Channel.

B_I?
[CELL:B_I?]
Returns current state of Busy/Idle bit or -1 if not available.

DCC?
[CELL:DCC?]
Returns current DCC value or -1 if not available.

SCC?
[CELL:SCC?]
Returns current SCC value or -1 if not available.

SID?
[CELL:SID?]
Returns current SID value or -1 if not available.

MIN?
[CELL:MIN?]
Returns current MIN value or -1 if not available. Returning format is: "xxx/xxx-xxxx."

ORDer?
[CELL:ORDer?]
Returns current ORDER value or -1 if not available.

VMAC?
[CELL:VMAC?]
Returns current VMAC value or -1 if not available.

VCHAN?
[CELL:VCHAN?]
Returns current Voice Channel number or -1 if not available.

CMAX_1?
[CELL:CMAX_1?]
Returns current CMAX-1 value or -1 if not available.

N_1?
[CELL:N_1?]
Returns current N-1 value or -1 if not available.

CMAC?
[CELL:CMAC?]
Returns current CMAC value or -1 if not available.

END?
[CELL:END?]
Returns current state of END bit or -1 if not available.

WFOM?
[CELL:WFOM?]
Returns current state of WFOM bit or -1 if not available.

ACTion?
[CELL:ACTion?]
Returns current Action field value or -1 if not available.

NAWC?
[CELL:NAWC?]
Returns current NAWC value or -1 if not available.

S?
[CELL:S?]
Returns current state of S bit or -1 if not available.

E?
[CELL:E?]
Returns current state of E bit or -1 if not available.

REGH?
[CELL:REGH?]
Returns current state of REGH bit or -1 if not available.

REGR?
[CELL:REGR?]
Returns current state of REGR bit or -1 if not available.

DTX?
[CELL:DTX?]
Returns current state of DTX bit or -1 if not available.

CELL:

RCF?

[CELL:RCF?]

Returns current state of RCF bit value or -1 if not available.

CPA?

[CELL:CPA?]

Returns current state of CPA bit or -1 if not available.

OLC?

[CELL:OLC?]

Returns current OLC value or -1 if not available.

BIS?

[CELL:BIS?]

Returns current state of BIS bit or -1 if not available.

REGINCR?

[CELL:REGINCR?]

Returns current REGINCR value or -1 if not available.

CHANPOS1?

[CELL:CHANPOS1?]

Returns current value of first channel position field from the directed retry message or -1 if not available.

CHANPOS2?

[CELL:CHANPOS2?]

Returns current value of the second channel position field from the directed retry message or -1 if not available.

CHANPOS3?

[CELL:CHANPOS3?]

Returns current value of the third channel position field from the directed retry message or -1 if not available.

CHANPOS4?

[CELL:CHANPOS4?]

Returns current value of the fourth channel position field from the directed retry message or -1 if not available.

CHANPOS5?

[CELL:CHANPOS5?]

Returns current value of the fifth channel position field from the directed retry message or -1 if not available.

CHANPOS6?

[CELL:CHANPOS6?]

Returns current value of the sixth channel position field from the directed retry message or -1 if not available.

NEWACC?

[CELL:NEWACC?]

Returns current NEWACC value or -1 if not available.

CELL:

MAXbusy:

PGR?

[CELL:MAXbusy:PGR?]

Returns current value of Maximum Busy for page response or -1 if not available.

OTHer?

[CELL:MAXbusy:OTHer?]

Returns current value of Maximum Busy for Other accesses or -1 if not available.

MAXSztr:

PGR?

[CELL:MAXSztr:PGR?]

Returns current value of Maximum Seizure Tries for page response or -1 if not available.

OTHer?

[CELL:MAXSztr:OTHer?]

Returns current value of Maximum Seizure Tries for Other accesses or -1 if not available.

PSCC?

[CELL:PSCC?]

Returns current PSCC value or -1 if not available.

ESN?

[CELL:ESN?]

Returns current ESN value or -1 if not available.

DIGITs?

[CELL:DIGITs?]

Returns current Call address value or -1 if not available.

PDSCC?

[CELL:PDSCC?]

Returns current PDSCC value or -1 if not available.

EP?

[CELL:EP?]

Returns current EP value or -1 if not available.

EF?

[CELL:EF?]

Returns current EF value or -1 if not available.

DSCC?

[CELL:DSCC?]

Returns current DSCC value or -1 if not available.

CELL:

LOCALCTRL1?

[CELL:LOCALCTRL1?]

Returns current first position field for the Local Control value or -1 if not available.

LOCALCTRL2?

[CELL:LOCALCTRL2?]

Returns current second position field for the Local Control value or -1 if not available.

C12?

[CELL:C12?]

Returns current C12 value or -1 if not available.

C13?

[CELL:C13?]

Returns current C13 value or -1 if not available.

MST?

[CELL:MST?]

Returns current MST value or -1 if not available.

MSL?

[CELL:MSL?]

Returns current MSL value or -1 if not available.

FORMat:

AMPS

[CELL:FORMat:AMPS]

Sets channel format to AMPS (800 MHz).

NT400

[CELL:FORMat:NT400]

Sets channel format to NT400 (450 MHz).

PCS

[CELL:FORMat:PCS]

Sets channel format to PCS (1900 MHz).

CELL:

FORMat:

NADC:

North American Digital Cellular. NADC frequencies consist of the following: 800 MHz (AMPS), 450 MHz (NT400©) and 1900 MHz (Hyperband).

BAND:

U8
[CELL:FORMat:NADC:BAND:U8]
Sets NADC band to U8 (AMPS - 800 MHz).

U4
[CELL:FORMat:NADC:BAND:U4]
Sets NADC band to U4 (NT400© - 450 MHz).

HY
[CELL:FORMat:NADC:BAND:HY]
Sets NADC band to HY (Hyperband - 1900 MHz).

NAMPS:

BAND:

Lower
[CELL:FORMat:NAMPS:BAND:Lower]
Sets NAMPS band to Lower.

Middle
[CELL:FORMat:NAMPS:BAND:Middle]
Sets NAMPS band to Middle.

Upper
[CELL:FORMat:NAMPS:BAND:Upper]
Sets NAMPS band to Upper.

BAND?
[CELL:BAND?]
Returns NADC or NAMPS band. Returns one of the following: U8, U4, HY, L, M or U.

FORMat?
[CELL:FORMat?]
Returns cellular format. Returns one of the following: AMPS, NT400 or PCS.

CELL:

CAPTure:

MODE:*x*
[CELL:CAPTure:MODE:x]
Specifies Mode on which to Capture: MIN, ORDER, BOTH (MIN and ORDER) or OFF (none).

MODE?
[CELL:CAPTure:MODE?]
Returns current capture mode: MIN, ORDER, BOTH or OFF

MIN *"xxx/xxx-xxxx"*
[CELL:CAPTure:MIN "xxx/xxx-xxxx"]
Specifies MIN capture value. Range of *x* is 0-9, # and *.

MIN?
[CELL:CAPTure:MIN?]
Returns current MIN Capture value in the format: "xxx/xxx-xxxx."

ORDer:*x*
[CELL:CAPTure:ORDer:x]
Specifies Order on which to capture. The following orders are valid for *x*:

PAGE	ALERT	RELease
REORDer	SALERT	AUDIT
SNDAddr	INTERCEPT	MAINTenance
POWer	DRETRY	AUTREG
AINTERCEPT	AREORDer	AALERT
VCDES		

ORDer?
[CELL:CAPTure:ORDer?]
Returns current Order on which to capture.

6-13-2 AMPS/NAMPS CELLULAR SIMULATION COMMANDS

A. FORWARD CONTROL CHANNEL

CELL:

GEN:

FOCC:

SETUP
[CELL:GEN:FOCC:SETUP]
Configures the Test Set for the Forward Control Channel Screen without displaying screen.

SEND
[CELL:GEN:FOCC:SEND]
Begin transmitting the System Parameter Overhead Message

STOP
[CELL:GEN:FOCC:STOP]
Stops transmitting the System Parameter Overhead Message

CHANnel *n*
[CELL:GEN:FOCC:CHANnel n]
Specifies Channel. Range of *n* is 1 to 2047, depending on format.

CHANnel?
[CELL:GEN:FOCC:CHANnel?]
Returns current Channel.

CMAC *n*
[CELL:GEN:FOCC:CMAC n]
Specifies value of Control Mobile Attenuation Code in the Control-Filler Message. Range of *n* is 0 to 7.

CMAC?
[CELL:GEN:FOCC:CMAC?]
Returns current value of Control Mobile Attenuation Code in the Control-Filler Message.

CMAX *n*
[CELL:GEN:FOCC:CMAX n]
Specifies value of Maximum number of channels in System Parameter Overhead Message. Range of *n* is 1 to 128.

CMAX?
[CELL:GEN:FOCC:CMAX?]
Returns current value of Maximum number of channels in System Parameter Overhead Message.

CELL:

GEN:

FOCC:

CPA *b*
[CELL:GEN:FOCC:CPA b]
Enables (*b* = 1) or disables (*b* = 0) Combined Paging/Access bit in System Parameter Overhead Message.

CPA?
[CELL:GEN:FOCC:CPA?]
Returns current state of Combined Paging/Access bit in System Parameter Overhead Message.

DCC *n*
[CELL:GEN:FOCC:DCC n]
Specifies value of Digital Color Code in System Parameter Overhead Message. Range of *n* is 0 to 3.

DCC?
[CELL:GEN:FOCC:DCC?]
Returns current value of Digital Color Code in System Parameter Overhead Message.

DTX *n*
[CELL:GEN:FOCC:DTX n]
Specifies value of Discontinuous Transmission in System Parameter Overhead Message. Range of *n* is 0 to 3

DTX?
[CELL:GEN:FOCC:DTX?]
Returns current value of Discontinuous Transmission in System Parameter Overhead Message.

E *b*
[CELL:GEN:FOCC:E b]
Enables (*b* = 1) or disables (*b* = 0) Extended Address bit in System Parameter Overhead Message.

E?
[CELL:GEN:FOCC:E?]
Returns current state of Extended Address bit in System Parameter Overhead Message.

EP *b*
[CELL:GEN:FOCC:EP b]
Enables (*b* = 1) or disables (*b* = 0) Extended Protocol bit in System Parameter Overhead Message.

EP?
[CELL:GEN:FOCC:EP?]
Returns current state of Extended Protocol bit in System Parameter Overhead Message.

CELL:

GEN:

FOCC:

N *n*
[CELL:GEN:FOCC:N n]
Specifies value of Number of paging channels in System Parameter Overhead Message. Range of *n* is 1 to 32.

N?
[CELL:GEN:FOCC:N?]
Returns current value of Number of paging channels in System Parameter Overhead Message.

RCF *b*
[CELL:GEN:FOCC:RCF b]
Enables (*b* = 1) or disables (*b* = 0) Read Control Filler bit in System Parameter Overhead Message.

RCF?
[CELL:GEN:FOCC:RCF?]
Returns current state of Read Control Filler bit in System Parameter Overhead Message.

REGH *b*
[CELL:GEN:FOCC:REGH b]
Enables (*b* = 1) or disables (*b* = 0) Registration for Home Stations bit in System Parameter Overhead Message.

REGH?
[CELL:GEN:FOCC:REGH?]
Returns current state of Registration for Home Stations bit in System Parameter Overhead Message.

REGR *b*
[CELL:GEN:FOCC:REGR b]
Enables (*b* = 1) or disables (*b* = 0) Registration for Roaming Mobile Phones bit in System Parameter Overhead Message.

REGR?
[CELL:GEN:FOCC:REGR?]
Returns current state of Registration for Roaming Mobile Phones bit in System Parameter Overhead Message.

S *b*
[CELL:GEN:FOCC:S b]
Enables (*b* = 1) or disables (*b* = 0) Serial Number bit in System Parameter Overhead Message.

S?
[CELL:GEN:FOCC:S?]
Returns current state of Serial Number bit in System Parameter Overhead Message.

CELL:

GEN:

FOCC:

SID *n*
[CELL:GEN:FOCC:SID n]
Specifies value of System Identification number in System Parameter Overhead Message. Range of *n* is 0 to 32767.

SID?
[CELL:GEN:FOCC:SID?]
Returns current value of System Identification number in System Parameter Overhead Message.

WFOM *b*
[CELL:GEN:FOCC:WFOM b]
Enables (*b* = 1) or disables (*b* = 0) Wait for Overhead Message bit in the Control-Filler Message.

WFOM?
[CELL:GEN:FOCC:WFOM?]
Returns current state of Wait for Overhead Message bit in the Control-Filler Message.

GLACT:

SETUP
[CELL:GEN:GLACT:SETUP]
Configures the Test Set for the Global Action Overhead Message Screen without displaying screen.

SEND
[CELL:GEN:GLACT:SEND]
Appends the Global Action Overhead Message to the System Parameter Overhead Message.

REPEAT:*x*
[CELL:GEN:GLACT:REPEAT:x]
Turns Repeat on (*x* = ON) or off (*x* = OFF) for the Global Action Overhead Message.

REPEAT?
[CELL:GEN:GLACT:REPEAT?]
Returns current state of Repeat for the Global Action Overhead Message.

STOP
[CELL:GEN:GLACT:STOP]
Stops the Global Action Overhead Message from being sent with the System Parameter Overhead Message.

CELL:

GEN:

GLACT:

CHANNEL *n*
[CELL:GEN:GLACT:CHANNEL n]
Specifies Channel. Range of *n* is 1 to 2047, depending on format.

CHANNEL?
[CELL:GEN:GLACT:CHANNEL?]
Returns current Channel.

ACTion:

RESCAN *b*
[CELL:GEN:GLACT:ACTion:RESCAN b]
Turns on (*b* = 1) or off (*b* = 0) Rescan in the Global Action Overhead Message menu.

RESCAN?
[CELL:GEN:GLACT:ACTion:RESCAN?]
Returns current state of Rescan in the Global Action Overhead Message menu.

REGINCR *b*
[CELL:GEN:GLACT:ACTion:REGINCR b]
Turns on (*b* = 1) or off (*b* = 0) Registration Increment in the Global Action Overhead Message menu.

REGINCR?
[CELL:GEN:GLACT:ACTion:REGINCR?]
Returns current state of Registration Increment in the Global Action Overhead Message menu.

NEWACC *b*
[CELL:GEN:GLACT:ACTion:NEWACC b]
Turns on (*b* = 1) or off (*b* = 0) New Access channel set in the Global Action Overhead Message menu.

NEWACC?
[CELL:GEN:GLACT:ACTion:NEWACC?]
Returns current state of New Access channel set in the Global Action Overhead Message menu.

OLC *b*
[CELL:GEN:GLACT:ACTion:OLC b]
Turns on (*b* = 1) or off (*b* = 0) Overload Control in the Global Action Overhead Message menu.

OLC?
[CELL:GEN:GLACT:ACTion:OLC?]
Returns current state of Overload Control in the Global Action Overhead Message menu.

CELL:

GEN:

GLACT:

ACTion:

BIS *b*
[CELL:GEN:GLACT:ACTion:BIS b]
Turns on (b = 1) or off (b = 0) Access Type in the Global Action Overhead Message menu.

BIS?
[CELL:GEN:GLACT:ACTion:BIS?]
Returns current state of Access Type in the Global Action Overhead Message menu.

ACCess *b*
[CELL:GEN:GLACT:ACTion:ACCess b]
Turns on (b = 1) or off (b = 0) Access Attempt in the Global Action Overhead Message menu.

ACCess?
[CELL:GEN:GLACT:ACTion:ACCess?]
Returns current state of Access Attempt in the Global Action Overhead Message menu.

LOCAL1 *b*
[CELL:GEN:GLACT:ACTion:LOCAL1 b]
Turns on (b = 1) or off (b = 0) Local Control 1 in the Global Action Overhead Message menu.

LOCAL1?
[CELL:GEN:GLACT:ACTion:LOCAL1?]
Returns current state of Local Control 1 in the Global Action Overhead Message menu.

LOCAL2 *b*
[CELL:GEN:GLACT:ACTion:LOCAL2 b]
Turns on (b = 1) or off (b = 0) Local Control 2 in the Global Action Overhead Message menu.

LOCAL2?
[CELL:GEN:GLACT:ACTion:LOCAL2?]
Returns current state of Local Control 2 in the Global Action Overhead Message menu.

CELL:

GEN:

GLACT:

BIS *b*
[CELL:GEN:GLACT:BIS b]
Enables (*b* = 1) or disables (*b* = 0) of Busy-Idle Status bit in the Global Action Overhead Message.

BIS?
[CELL:GEN:GLACT:BIS?]
Returns current state of Busy-Idle Status bit in the Global Action Overhead Message.

LOCALcntl *n*
[CELL:GEN:GLACT:LOCALcntl n]
Specifies value of Local Control in the Global Action Overhead Message. Range of *n* is 0 to 65535.

LOCALcntl?
[CELL:GEN:GLACT:LOCALcntl?]
Returns current value of Local Control in the Global Action Overhead Message.

MAXBusy:

OTHer *n*
[CELL:GEN:GLACT:MAXBusy:OTHer n]
Specifies value of Maximum number of Busy occurrences allowed for Other accesses in the Global Action Overhead Message. Range of *n* is 0 to 15.

OTHer?
[CELL:GEN:GLACT:MAXBusy:OTHer?]
Returns current value of Maximum number of Busy occurrences allowed for Other accesses in the Global Action Overhead Message.

PGR *n*
[CELL:GEN:GLACT:MAXBusy:PGR n]
Specifies value of Maximum number of Busy occurrences allowed for Page Responses in the Global Action Overhead Message. Range of *n* is 0 to 15.

PGR?
[CELL:GEN:GLACT:MAXBusy:PGR?]
Returns current value of Maximum number of Busy occurrences allowed for Page Responses in the Global Action Overhead Message.

CELL:

GEN:

GLACT:

MAXSztr:

OTHer *n*
[CELL:GEN:GLACT:MAXSztr:OTHer n]
Specifies value of Maximum number of Seizure attempts allowed for Other accesses in the Global Action Overhead Message. Range of *n* is 0 to 15

OTHer?
[CELL:GEN:GLACT:MAXSztr:OTHer?]
Returns current value of Maximum number of Seizure attempts allowed for Other accesses in the Global Action Overhead Message.

PGR *n*
[CELL:GEN:GLACT:MAXSztr:PGR n]
Specifies value of Maximum number of Seizure attempts allowed for Page Responses in the Global Action Overhead Message. Range of *n* is 0 to 15.

PGR?
[CELL:GEN:GLACT:MAXSztr:PGR?]
Returns current value of Maximum number of Seizure attempts allowed for Page Responses in the Global Action Overhead Message.

NEWACC *n*
[CELL:GEN:GLACT:NEWACC n]
Specifies value of New Access Channel starting point in the Global Action Overhead Message. Range of *n* is 0 to 2047.

NEWACC?
[CELL:GEN:GLACT:NEWACC?]
Returns current value of New Access Channel starting point in the Global Action Overhead Message.

OLC *n*
[CELL:GEN:GLACT:OLC n]
Specifies value of Overload Control Class in the Global Action Overhead Message. Range of *n* is 0 to 32767.

OLC?
[CELL:GEN:GLACT:OLC?]
Returns current value of Overload Control Class in the Global Action Overhead Message.

CELL:

GEN:

GLACT:

REGINCR *n*
[CELL:GEN:GLACT:REGINCR n]
Specifies value of Registration Increment in the Global Action Overhead Message. Range of *n* is 0 to 4095.

REGINCR?
[CELL:GEN:GLACT:REGINCR?]
Returns current value of Registration Increment in the Global Action Overhead Message.

MSCM:

SETUP
[CELL:GEN:MSCM:SETUP]
Configures the Test Set for the Mobile Station Control Message Screen without displaying screen.

SEND
[CELL:GEN:MSCM:SEND]
Sends the Mobile Station Control Message.

REPEAT:*x*
[CELL:GEN:MSCM:REPEAT:x]
Sets Repeat on (*x* = ON) or off (*x* = OFF) in the Mobile Station Control Message.

REPEAT?
[CELL:GEN:MSCM:REPEAT?]
Returns current state of Repeat in the Mobile Station Control Message.

STOP
[CELL:GEN:MSCM:STOP]
Stops transmission of the Mobile Station Control Message.

CHANNEL *n*
[CELL:GEN:MSCM:CHANNEL n]
Specifies Channel. Range of *n* is 1 to 2047, depending on format.

CHANNEL?
[CELL:GEN:MSCM:CHANNEL?]
Returns current Channel.

CELL:

GEN:

MSCM:

CHAN *n*
[CELL:GEN:MSCM:CHAN n]
Specifies voice channel to which call is assigned in the Mobile Station Control Message. Range of *n* is 0 to 1024.

CHAN?
[CELL:GEN:MSCM:CHAN?]
Returns current voice channel assignment for call in the Mobile Station Control Message.

C12 *b*
[CELL:GEN:MSCM:C12 b]
Enables (*b* = 1) or disables (*b* = 0) C12 bit in the Mobile Station Control Message.

C12?
[CELL:GEN:MSCM:C12?]
Returns current state of C12 bit in the Mobile Station Control Message.

C13 *b*
[CELL:GEN:MSCM:C13 b]
Enables (*b* = 1) or disables (*b* = 0) C13 bit in the Mobile Station Control Message.

C13?
[CELL:GEN:MSCM:C13?]
Returns current state of C13 bit in the Mobile Station Control Message.

CHANPos *x,y*
[CELL:GEN:MSCM:CHANPos x,y]
Sets the position of a control channel relative to the first access channel in the Mobile Station Control Message. Range of *x* is 1 to 6; range of *y* is 0 to 127.

CHANPos? *n*
[CELL:GEN:MSCM:CHANPos? n]
Returns the current position of a control channel relative to the first access channel in the Mobile Station Control Message. Range of *n* is 1 to 6.

CLI "*string*"
[CELL:GEN:MSCM:CLI "string"]
Specifies Call Line Identifier string in the Mobile Station Control Message. String may consist of 0-9, #, * and N (Null) with a maximum of 32 characters.

CLI?
[CELL:GEN:MSCM:CLI?]
Returns current Call Line Identifier string in the Mobile Station Control Message.

CELL:

GEN:

MSCM:

DSCC *n*
[CELL:GEN:MSCM:DSCC n]
Specifies value of DSAT Color Code in the Mobile Station Control Message. Range of *n* is 0 to 7.

DSCC?
[CELL:GEN:MSCM:DSCC?]
Returns current value of DSAT Color Code in the Mobile Station Control Message.

EF *b*
[CELL:GEN:MSCM:EF b]
Enables (*b* = 1) or disables (*b* = 0) Extended Protocol Forward Channel Indicator bit in the Mobile Station Control Message.

EF?
[CELL:GEN:MSCM:EF?]
Returns current state of Extended Protocol Forward Channel Indicator bit in the Mobile Station Control Message.

LOCAL *n*
[CELL:GEN:MSCM:LOCAL n]
Specifies value of LOCAL in the Mobile Station Control Message. Range of *n* is 0 to 31.

LOCAL?
[CELL:GEN:MSCM:LOCAL?]
Returns current value of LOCAL in the Mobile Station Control Message.

MIN *"xxx/xxx-xxxx"*
[CELL:GEN:MSCM:MIN "xxx/xxx-xxxx"]
Specifies Mobile Identification Number in the Mobile Station Control Message. Range of *x* is 0-9, # and *.

MIN?
[CELL:GEN:MSCM:MIN?]
Returns current Mobile Identification Number in the Mobile Station Control Message.

MSL *n*
[CELL:GEN:MSCM:MSL n]
Specifies value of Message Length in the Mobile Station Control Message. Range of *n* is 0 to 31.

MSL?
[CELL:GEN:MSCM:MSL?]
Returns current value of Message Length in the Mobile Station Control Message.

CELL:

GEN:

MSCM:

MST *n*
[CELL:GEN:MSCM:MST n]
Specifies value of Message Type in the Mobile Station Control Message. Range of *n* is 0 to 255.

MST?
[CELL:GEN:MSCM:MST?]
Returns current value of Message Type in the Mobile Station Control Message.

ORDQ *n*
[CELL:GEN:MSCM:ORDQ n]
Specifies value of Order Qualifier in the Mobile Station Control Message. Range of *n* is 0 to 7.

ORDQ?
[CELL:GEN:MSCM:ORDQ?]
Returns current value of Order Qualifier in the Mobile Station Control Message.

ORDER:*x*
[CELL:GEN:MSCM:ORDER:x]
Specifies Order in the Mobile Station Control Message. The following orders are valid for *x*:

AUDIT	LC	DIR_RTRY
INTRCPT	RELease	REORDER
VC_DES	EXTENDed	

SCC *n*
[CELL:GEN:MSCM:SCC n]
Specifies value of Supervisory Audio Tone Color Code in the Mobile Station Control Message. Range of *n* is 0 to 3.

SCC?
[CELL:GEN:MSCM:SCC?]
Returns current value of Supervisory Audio Tone Color Code in the Mobile Station Control Message.

SHORT_MESSage *"string"*
[CELL:GEN:MSCM:SHORT_MESSage "string"]
Specifies Short Message string in Mobile Station Control Message. See IS-88, Appendix A for string details.

SHORT_MESSage?
[CELL:GEN:MSCM:SHORT_MESSage?]
Returns current Short Message string in Mobile Station Control Message.

CELL:

GEN:

MSCM:

VMAC *n*
[CELL:GEN:MSCM:VMAC n]
Specifies value of Voice Mobile Attenuation Code in Mobile Station Control Message. Range of *n* is 0 to 7.

VMAC?
[CELL:GEN:MSCM:VMAC?]
Returns current value of Voice Mobile Attenuation Code in Mobile Station Control Message.

VOICE_MESSage "*string*"
[CELL:GEN:MSCM:VOICE_MESSage "string"]
Specifies Voice Mail message string in Mobile Station Control Message. See IS-88, Appendix A for string details.

VOICE_MESSage?
[CELL:GEN:MSCM:VOICE_MESSage?]
Returns current Voice Mail message string in Mobile Station Control Message.

VOICE_UNANSWERED "*nn*"
[CELL:GEN:MSCM:VOICE_UNANSWERED "nn"]
Specifies number of unanswered messages in Mobile Station Control Message. Range of *nn* is 00 to 99.

VOICE_UNANSWERED?
[CELL:GEN:MSCM:VOICE_UNANSWERED?]
Returns current number of unanswered messages in Mobile Station Control Message.

VOICE_URGENT:*x*
[CELL:GEN:MSCM:VOICE_URGENT:x]
Turns on (*x* = ON) or off (*x* = OFF) Urgent message identifier in Mobile Station Control Message.

VOICE_URGENT?
[CELL:GEN:MSCM:VOICE_URGENT?]
Returns current state of Urgent message identifier in Mobile Station Control Message.

B. REVERSE CONTROL CHANNEL

CELL:

GEN:

RECC:

SETUP
[CELL:GEN:RECC:SETUP]
Configures the Test Set for the Reverse Control Channel Screen without displaying screen.

SEND
[CELL:GEN:RECC:SEND]
Sends Reverse Control Channel message.

REPEAT:*x*
[CELL:GEN:RECC:REPEAT:x]
Turns Repeat on (*x* = ON) or off (*x* = OFF).

REPEAT?
[CELL:GEN:RECC:REPEAT?]
Returns current state of Repeat.

STOP
[CELL:GEN:RECC:STOP]
Stops transmission of Reverse Control Channel message.

CHANNEL *n*
[CELL:GEN:RECC:CHANNEL n]
Specifies Channel. Range of *n* is 0 to 1023.

CHANNEL?
[CELL:GEN:RECC:CHANNEL?]
Returns current Channel.

CALLED_ADDRess "*string*"
[CELL:GEN:RECC:CALLED_ADDRess "string"]
Specifies Called Address. String may consist of 0-9, #, and * with a maximum of 32 characters.

CALLED_ADDRess?
[CELL:GEN:RECC:CALLED_ADDRess?]
Returns current Called Address.

E *b*
[CELL:GEN:RECC:E b]
Enables (*b* = 1) or disables (*b* = 0) Extended Address.

E?
[CELL:GEN:RECC:E?]
Returns current state of Extended Address.

CELL:

GEN:

RECC:

EP *b*
[CELL:GEN:RECC:EP b]
Sets state of Extended Protocol bit. (0 or 1).

EP?
[CELL:GEN:RECC:EP?]
Returns current state of Extended Protocol bit.

ER *b*
[CELL:GEN:RECC:ER b]
Enables (*b* = 1) or disables (*b* = 0) Extended Protocol Reverse Channel.

ER?
[CELL:GEN:RECC:ER?]
Returns current state of Extended Protocol Reverse Channel.

DCC *n*
[CELL:GEN:RECC:DCC n]
Specifies value of Digital Color Code. Range of *n* is 0 to 3.

DCC?
[CELL:GEN:RECC:DCC?]
Returns current value of Digital Color Code.

ESN "*string*"
[CELL:GEN:RECC:ESN "string"]
Specifies Electronic Serial Number. String may consist of 0-9 with a maximum of 11 digits.

ESN?
[CELL:GEN:RECC:ESN?]
Returns current Electronic Serial Number.

LOCAL *n*
[CELL:GEN:RECC:LOCAL n]
Specifies value of LOCAL. Range of *n* is 0 to 31.

LOCAL?
[CELL:GEN:RECC:LOCAL?]
Returns current value of LOCAL.

LT *b*
[CELL:GEN:RECC:LT b]
Enables (*b* = 1) or disables (*b* = 0) Last Try.

LT?
[CELL:GEN:RECC:LT?]
Returns current state of Last Try.

CELL:

GEN:

RECC:

MIN *"xxx/xxx-xxxx"*
[CELL:GEN:RECC:MIN "xxx/xxx-xxxx"]
Specifies Mobile Identification Number. String may consist of 0-9, # and *.

MIN?
[CELL:GEN:RECC:MIN?]
Returns current Mobile Identification Number.

MSL *n*
[CELL:GEN:RECC:MSL n]
Specifies value of Message Length. Range of *n* is 0 to 31.

MSL?
[CELL:GEN:RECC:MSL?]
Returns current value of Message Length.

MST *n*
[CELL:GEN:RECC:MST n]
Specifies value of Extended Protocol Message Type. Range of *n* is 0 to 255.

MST?
[CELL:GEN:RECC:MST?]
Returns current value of Extended Protocol Message Type.

ORDER *n*
[CELL:GEN:RECC:ORDER n]
Specifies value of Order. Range of *n* is 0 to 31.

ORDER?
[CELL:GEN:RECC:ORDER?]
Returns current value of Order.

ORDQ *n*
[CELL:GEN:RECC:ORDQ n]
Specifies value of Order Qualifier. Range of *n* is 0 to 7.

ORDQ?
[CELL:GEN:RECC:ORDQ?]
Returns current value of Order Qualifier.

S *b*
[CELL:GEN:RECC:S b]
Enables (*b* = 1) or disables (*b* = 0) Serial Number.

S?
[CELL:GEN:RECC:S?]
Returns current state of Serial Number.

CELL:

GEN:

RECC:

SCM *n*
[CELL:GEN:RECC:SCM n]
Specifies value of Station Class Mark. Range of *n* is 0 to 15.

SCM?
[CELL:GEN:RECC:SCM?]
Returns current value of Station Class Mark.

T *b*
[CELL:GEN:RECC:T b]
Enables (*b* = 1) or disables (*b* = 0) Type of message.

T?
[CELL:GEN:RECC:T?]
Returns current state of Type of message.

C. FORWARD VOICE CHANNEL (AMPS)

CELL:

GEN:

FVC:

SETUP
[CELL:GEN:FVC:SETUP]
Configures the Test Set for the AMPS Forward Voice Channel Screen without displaying screen.

SEND
[CELL:GEN:FVC:SEND]
Sends the Mobile Station Control Message.

REPEAT:*x*
[CELL:GEN:FVC:REPEAT:x]
Turns Repeat on (*x* = ON) or off (*x* = OFF).

REPEAT?
[CELL:GEN:FVC:REPEAT?]
Returns current state of Repeat.

STOP
[CELL:GEN:FVC:STOP]
Stops the transmission of the Mobile Station Control Message.

CELL:

GEN:

FVC:

CHANNEL *n*
[CELL:GEN:FVC:CHANNEL n]
Specifies Channel. Range of *n* is 1 to 2047, depending on format.

CHANNEL?
[CELL:GEN:FVC:CHANNEL?]
Returns current Channel.

CHAN *n*
[CELL:GEN:FVC:CHAN n]
Specifies voice channel to which call is assigned in the Mobile Station Control Message. Range of *n* is 1 to 1024.

CHAN?
[CELL:GEN:FVC:CHAN?]
Returns current voice channel to which call is assigned in the Mobile Station Control Message.

C12 *b*
[CELL:GEN:FVC:C12 b]
Enables (*b* = 1) or disables (*b* = 0) C12 bit in Mobile Station Control Message.

C12?
[CELL:GEN:FVC:C12?]
Returns current state of C12 bit in Mobile Station Control Message.

C13 *b*
[CELL:GEN:FVC:C13 b]
Enables (*b* = 1) or disables (*b* = 0) C13 bit in Mobile Station Control Message.

C13?
[CELL:GEN:FVC:C13?]
Returns current state of C13 bit in Mobile Station Control Message.

CLI "*string*"
[CELL:GEN:FVC:CLI "string"]
Specifies Call Line Identifier string in the Mobile Station Control Message. String may consist of 0-9, #, * and N (Null) with a maximum of 32 characters.

CLI?
[CELL:GEN:FVC:CLI?]
Returns current Call Line Identifier string in the Mobile Station Control Message.

CELL:

GEN:

FVC:

DSCC *n*
[CELL:GEN:FVC:DSCC n]
Specifies value of DSAT Color Code in the Mobile Station Control Message. Range of *n* is 0 to 7.

DSCC?
[CELL:GEN:FVC:DSCC?]
Returns current value of DSAT Color Code in the Mobile Station Control Message.

LOCAL *n*
[CELL:GEN:FVC:LOCAL n]
Specifies value of LOCAL in the Mobile Station Control Message. Range of *n* is 0 to 31.

LOCAL?
[CELL:GEN:FVC:LOCAL?]
Returns current value of LOCAL in the Mobile Station Control Message.

MSL *n*
[CELL:GEN:FVC:MSL n]
Specifies value of Message Length in the Mobile Station Control Message. Range of *n* is 0 to 31.

MSL?
[CELL:GEN:FVC:MSL?]
Returns current value of Message Length in the Mobile Station Control Message.

MST *n*
[CELL:GEN:FVC:MST n]
Specifies value of Message Type in the Mobile Station Control Message. Range of *n* is 0 to 225.

MST?
[CELL:GEN:FVC:MST?]
Returns current value of Message Type in the Mobile Station Control Message.

ORDER:*x*
[CELL:GEN:FVC:ORDER:x]
Specifies ORDER in the Mobile Station Control Message. The following orders are valid for *x*:

PAGE	SNDADDR	S_ALERT
AUDIT	MAINTenance	ALERT
RELease	PWRLvl	HANDoff
NAMPS_CH_ASGN	EXTENDed	

CELL:

GEN:

FVC:

PSCC *n*
[CELL:GEN:FVC:PSCC n]
Specifies value of Present SAT Color Code in the Mobile Station Control Message. Range of *n* is 0 to 3.

PSCC?
[CELL:GEN:FVC:PSCC?]
Returns current value of Present SAT Color Code in the Mobile Station Control Message.

PWRLVL *n*
[CELL:GEN:FVC:PWRLVL n]
Specifies Power Level. Range of *n* is 0 to 7.

PWRLVL?
[CELL:GEN:FVC:PWRLVL?]
Returns Power Level.

SCC *n*
[CELL:GEN:FVC:SCC n]
Specifies value of SAT Color Code in the Mobile Station Control Message. Range of *n* is 0 to 3.

SCC?
[CELL:GEN:FVC:SCC?]
Returns current value of SAT Color Code in the Mobile Station Control Message.

SHORT_MESSage "*string*"
[CELL:GEN:FVC:SHORT_MESSage "string"]
Specifies Short Message string in the Mobile Station Control Message. See IS-88, Appendix A for string details.

SHORT_MESSage?
[CELL:GEN:FVC:SHORT_MESSage?]
Returns current Short Message string in the Mobile Station Control Message.

VMAC *n*
[CELL:GEN:FVC:VMAC n]
Specifies value of Voice Mobile Attenuation Code in the Mobile Station Control Message. Range of *n* is 0 to 7.

VMAC?
[CELL:GEN:FVC:VMAC?]
Returns current value of Voice Mobile Attenuation Code in the Mobile Station Control Message.

CELL:

GEN:

FVC:

VOICE_MESSage "*string*"
[CELL:GEN:FVC:VOICE_MESSage "string"]
Specifies Voice Mail message string in the Mobile Station Control Message. See IS-88, Appendix A for string details.

VOICE_MESSage?
[CELL:GEN:FVC:VOICE_MESSage?]
Returns current Voice Mail message string in the Mobile Station Control Message.

VOICE_UNANSWERED "*nn*"
[CELL:GEN:FVC:VOICE_UNANSWERED "nn"]
Specifies number of unanswered messages in Mobile Station Control Message. Range of *nn* is 00 to 99.

VOICE_UNANSWERED?
[CELL:GEN:FVC:VOICE_UNANSWERED?]
Returns current number of unanswered messages in Mobile Station Control Message.

VOICE_URGENT:*x*
[CELL:GEN:FVC:VOICE_URGENT:x]
Turns on (*x* = ON) or off (*x* = OFF) Urgent message identifier in Mobile Station Control Message.

VOICE_URGENT?
[CELL:GEN:FVC:VOICE_URGENT?]
Returns current state of Urgent message identifier in Mobile Station Control Message.

D. FORWARD VOICE CHANNEL (NAMPS)

CELL:

GEN:

FVC:

NAMPS:

SETUP
[CELL:GEN:FVC:NAMPS:SETUP]
Configures the Test Set for the NAMPS Forward Voice Channel Screen without displaying screen.

SEND
[CELL:GEN:FVC:NAMPS:SEND]
Sends the Mobile Station Control Message.

NEXT
[CELL:GEN:FVC:NAMPS:NEXT]
Sends the next word of the Mobile Station Control Message, if required.

CHANNEL *n*
[CELL:GEN:FVC:NAMPS:CHANNEL n]
Specifies Channel. Range of *n* is 0 to 1023.

CHANNEL:*x*
[CELL:GEN:FVC:NAMPS:CHANNEL:x]
Specifies Band. The following bands are valid for *x*: LOWer, MIDdle, UPper.

CHANNEL?
[CELL:GEN:FVC:NAMPS:CHANNEL?]
Returns current Channel.

CHAN *n*
[CELL:GEN:FVC:NAMPS:CHAN n]
Specifies voice channel to which call is assigned in the Mobile Station Control Message (11 least significant bits). Range of *n* is 1 to 1024.

CHAN?
[CELL:GEN:FVC:NAMPS:CHAN?]
Returns current voice channel to which call is assigned in the Mobile Station Control Message (11 least significant bits).

BER *n*
[CELL:GEN:FVC:NAMPS:BER n]
Specifies number of allowable bit errors in Mobile Station Control Message. Range of *n* is 0 to 127.

BER?
[CELL:GEN:FVC:NAMPS:BER?]
Returns current number of allowable bit errors in Mobile Station Control Message.

CELL:

GEN:

FVC:

NAMPS:

C13 *b*
[CELL:GEN:FVC:NAMPS:C13 b]
Enables (*b* = 1) or disables (*b* = 0) C13 bit in Mobile Station Control Message.

C13?
[CELL:GEN:FVC:NAMPS:C13?]
Returns current state of C13 bit in Mobile Station Control Message.

C12 *b*
[CELL:GEN:FVC:NAMPS:C12 b]
Enables (*b* = 1) or disables (*b* = 0) C12 bit in Mobile Station Control Message.

C12?
[CELL:GEN:FVC:NAMPS:C12?]
Returns current state of C12 bit in Mobile Station Control Message.

DSCC *n*
[CELL:GEN:FVC:NAMPS:DSCC n]
Specifies value of DSAT Color Code in Mobile Station Control Message. Range of *n* is 0 to 7.

DSCC?
[CELL:GEN:FVC:NAMPS:DSCC?]
Returns current value of DSAT Color Code in Mobile Station Control Message.

CLI "*string*"
[CELL:GEN:FVC:NAMPS:CLI "string"]
Specifies Call Line Identifier string in the Mobile Station Control Message. String may consist of 0-9, #, * and N (Null) with a maximum of 32 characters.

CLI?
[CELL:GEN:FVC:NAMPS:CLI?]
Returns current Call Line Identifier string in the Mobile Station Control Message.

CTYP *b*
[CELL:GEN:FVC:NAMPS:CTYP b]
Enables (*b* = 1) or disables (*b* = 0) Channel Type indicator in the Mobile Station Control Message.

CTYP?
[CELL:GEN:FVC:NAMPS:CTYP?]
Returns current state of Channel Type indicator in the Mobile Station Control Message.

CELL:

GEN:

FVC:

NAMPS:

DSAT *n*
[CELL:GEN:FVC:NAMPS:DSAT n]
Specifies value of Digital Supervisory Audio Tone in the Mobile Station Control Message. Range of *n* is 0 to 6.

DSAT?
[CELL:GEN:FVC:NAMPS:DSAT?]
Returns current value of Digital Supervisory Audio Tone in the Mobile Station Control Message.

LOCAL *n*
[CELL:GEN:FVC:NAMPS:LOCAL n]
Specifies value of LOCAL in the Mobile Station Control Message. Range of *n* is 0 to 31.

LOCAL?
[CELL:GEN:FVC:NAMPS:LOCAL?]
Returns current value of LOCAL in the Mobile Station Control Message.

MSL *n*
[CELL:GEN:FVC:NAMPS:MSL n]
Specifies value of Message Length in the Mobile Station Control Message. Range of *n* is 0 to 31.

MSL?
[CELL:GEN:FVC:NAMPS:MSL?]
Returns current value of Message Length in the Mobile Station Control Message.

MST *n*
[CELL:GEN:FVC:NAMPS:MST n]
Specifies value of Message Type in the Mobile Station Control Message. Range of *n* is 0 to 255.

MST?
[CELL:GEN:FVC:NAMPS:MST?]
Returns current value of Message Type in the Mobile Station Control Message.

O_E *b*
[CELL:GEN:FVC:NAMPS:O_E b]
Enables (*b* = 1) or disables (*b* = 0) Odd/Even in the Mobile Station Control Message.

O_E?
[CELL:GEN:FVC:NAMPS:O_E?]
Returns current state of Odd/Even in the Mobile Station Control Message.

CELL:

GEN:

FVC:

NAMPS:

ORDER:*x*
[CELL:GEN:FVC:NAMPS:ORDER:x]
Specifies Order in the Mobile Station Control Message. The following orders are valid for *x*:

PAGE	SNDADDR_EVEN	SNDADDR_ODD
S_ALERT	AUDIT	MAINTenance
ALERT	RELease	FADE
SUSP_CALLED_ ADDR	HANDOFF_ CONFIRM	PWRLvl
HANDoff	MRI	EXTENDed

PDSCC *n*
[CELL:GEN:FVC:NAMPS:PDSCC n]
Specifies value of Present DSAT Color Code in the Mobile Station Control Message. Range of *n* is 0 to 7.

PDSCC?
[CELL:GEN:FVC:NAMPS:PDSCC?]
Returns current value of Present DSAT Color Code in the Mobile Station Control Message.

PWRLVL *n*
[CELL:GEN:FVC:NAMPS:PWRLVL n]
Specifies Power Level. Range of *n* is 0 to 7.

PWRLVL?
[CELL:GEN:FVC:NAMPS:PWRLVL?]
Returns Power Level.

RSSI *n*
[CELL:GEN:FVC:NAMPS:RSSI n]
Specifies value of Received Signal Strength in the Mobile Station Control Message. Range of *n* is 0 to 7.

RSSI?
[CELL:GEN:FVC:NAMPS:RSSI?]
Return current value of Received Signal Strength in the Mobile Station Control Message.

CELL:

GEN:

FVC:

NAMPS:

SHORT_MESSage "*string*"
[CELL:GEN:FVC:NAMPS:SHORT_MESSage "string"]
Specifies Short Message string in the Mobile Station Control Message. See IS-88, Appendix A for string details.

SHORT_MESSage?
[CELL:GEN:FVC:NAMPS:SHORT_MESSage?]
Returns current Short Message string in the Mobile Station Control Message.

VMAC *n*
[CELL:GEN:FVC:NAMPS:VMAC n]
Specifies value of Voice Mobile Attenuation Code in the Mobile Station Control Message. Range of *n* is 0 to 7.

VMAC?
[CELL:GEN:FVC:NAMPS:VMAC?]
Returns current value of Voice Mobile Attenuation Code in the Mobile Station Control Message.

VOICE_MESSage "*string*"
[CELL:GEN:FVC:NAMPS:VOICE_MESSage "string"]
Specifies Voice Mail message string in the Mobile Station Control Message. See IS-88, Appendix A for string details.

VOICE_MESSage?
[CELL:GEN:FVC:NAMPS:VOICE_MESSage?]
Returns current Voice Mail message string in the Mobile Station Control Message.

VOICE_UNANSWERED "*nn*"
[CELL:GEN:FVC:NAMPS:VOICE_UNANSWERED "nn"]
Specifies number of unanswered messages in Mobile Station Control Message. Range of *nn* is 00 to 99.

VOICE_UNANSWERED?
[CELL:GEN:FVC:NAMPS:VOICE_UNANSWERED?]
Returns current number of unanswered messages in Mobile Station Control Message.

VOICE_URGENT:*x*
[CELL:GEN:FVC:NAMPS:VOICE_URGENT:x]
Turns on (*x* = ON) or off (*x* = OFF) Urgent message identifier in Mobile Station Control Message.

VOICE_URGENT?
[CELL:GEN:FVC:NAMPS:VOICE_URGENT?]
Returns current state of Urgent message identifier in Mobile Station Control Message.

E. REVERSE VOICE CHANNEL (AMPS)

CELL:

GEN:

RVC:

SETUP
[CELL:GEN:RVC:SETUP]
Configures the Test Set for the AMPS Reverse Voice Channel Screen without displaying screen.

SEND
[CELL:GEN:RVC:SEND]
Sends an RVC message.

REPEAT:*x*
[CELL:GEN:RVC:REPEAT:x]
Turns Repeat on (*x* = ON) or off (*x* = OFF).

REPEAT?
[CELL:GEN:RVC:REPEAT?]
Returns current state of Repeat.

STOP
[CELL:GEN:RVC:STOP]
Stops transmission of any RVC message.

CHANNEL *n*
[CELL:GEN:RVC:CHANNEL n]
Specifies Channel. Range of *n* is 0 to 1023.

CHANNEL?
[CELL:GEN:RVC:CHANNEL?]
Returns current Channel.

CALLED_ADDRess "*string*"
[CELL:GEN:RVC:CALLED_ADDRess "string"]
Specifies Called Address string for the Called-Address Message. String consists of 0-9, # and * with a maximum of 32 characters.

CALLED_ADDRess?
[CELL:GEN:RVC:CALLED_ADDRess?]
Returns current Called Address string for the Called-Address Message.

LOCAL *n*
[CELL:GEN:RVC:LOCAL n]
Specifies value of LOCAL in Order Confirmation Message. Range of *n* is 0 to 31.

LOCAL?
[CELL:GEN:RVC:LOCAL?]
Returns current value of LOCAL in Order Confirmation Message.

CELL:

GEN:

RVC:

MESSAGE:*x*
[CELL:GEN:RVC:MESSAGE:x]
Specifies Message to be transmitted. The following messages are valid for *x*: ORDER_CONFIRMation, CALLED_ADDRess, EXTENDed.

MSL *n*
[CELL:GEN:RVC:MSL n]
Specifies value of Message Length in Extended Protocol Message. Range of *n* is 0 to 31.

MSL?
[CELL:GEN:RVC:MSL?]
Returns current value of Message Length in Extended Protocol Message.

MST *n*
[CELL:GEN:RVC:MST n]
Specifies Message Type in Extended Protocol Message. Range of *n* is 0 to 255.

MST?
[CELL:GEN:RVC:MST?]
Returns current value of Message Type in Extended Protocol Message.

ORDER *n*
[CELL:GEN:RVC:ORDER n]
Specifies value of ORDER in Order Confirmation Message. Range of *n* is 0 to 31.

ORDER?
[CELL:GEN:RVC:ORDER?]
Returns current value of ORDER in Order Confirmation Message.

ORDQ *n*
[CELL:GEN:RVC:ORDQ n]
Specifies value of Order Qualifier in Order Confirmation Message. Range of *n* is 0 to 7.

ORDQ?
[CELL:GEN:RVC:ORDQ?]
Returns current value of Order Qualifier in Order Confirmation Message.

F. REVERSE VOICE CHANNEL (NAMPS)

CELL:

GEN:

RVC:

NAMPS:

SETUP
[CELL:GEN:RVC:NAMPS:SETUP]
Configures the Test Set for the NAMPS Reverse Voice Channel Screen without displaying screen.

SEND
[CELL:GEN:RVC:NAMPS:SEND]
Sends message.

NEXT
[CELL:GEN:RVC:NAMPS:NEXT]
Sends the next word of the Mobile Station Control Message, if required.

CHANNEL *n*
[CELL:GEN:RVC:NAMPS:CHANNEL n]
Specifies Channel. Range of *n* is 1 to 2047, depending on format.

CHANNEL:*x*
[CELL:GEN:RVC:NAMPS:CHANNEL:x]
Specifies Band. The following bands are valid for *x*: LOWer, MIDdle, UPper.

CHANNEL?
[CELL:GEN:RVC:NAMPS:CHANNEL?]
Returns current Channel.

BER *n*
[CELL:GEN:RVC:NAMPS:BER n]
Specifies number of Bit Errors in the MRI (Mobile Reported Interference) Order Message. Range of *n* is 0 to 127.

BER?
[CELL:GEN:RVC:NAMPS:BER?]
Returns current number of Bit Errors in the MRI (Mobile Reported Interference) Order Message.

CALLED_ADDRess "*string*"
[CELL:GEN:RVC:NAMPS:CALLED_ADDRess "string"]
Specifies Called Address for Flash/Called-Address Message string. String may consist of 0-9, # and * with a maximum of 32 characters.

CALLED_ADDRess?
[CELL:GEN:RVC:NAMPS:CALLED_ADDRess?]
Returns current Called Address for Flash/Called-Address Message string.

CELL:

GEN:

RVC:

NAMPS:

CONFIRMation *b*
[CELL:GEN:RVC:NAMPS:CONFIRMation b]
Sets state of T bit in Reverse Voice Channel Message to specify that message is a Order Confirmation Message (*b* = 1) or an Order (*b* = 0).

CONFIRMation?
[CELL:GEN:RVC:NAMPS:CONFIRMation?]
Returns current state of T bit in Reverse Voice Channel Message to specify that message is a Order Confirmation Message (1) or an Order (0).

DSAT *n*
[CELL:GEN:RVC:NAMPS:DSAT n]
Specifies value of Digital Supervisory Audio Tone. Range of *n* is 0 to 6.

DSAT?
[CELL:GEN:RVC:NAMPS:DSAT?]
Returns current value of Digital Supervisory Audio Tone.

LOCAL *n*
[CELL:GEN:RVC:NAMPS:LOCAL n]
Specifies value of LOCAL for Order Message or Order Confirmation Message. Range of *n* is 0 to 31.

LOCAL?
[CELL:GEN:RVC:NAMPS:LOCAL?]
Returns current value of LOCAL for Order Message or Order Confirmation Message

MESSAGE:*x*
[CELL:GEN:RVC:NAMPS:MESSAGE:x]
Specifies message to transmitted. The following messages are valid for *x*: MRI, ORDER, FLASH.

O_E *b*
[CELL:GEN:RVC:NAMPS:O_E b]
Sets state of Odd/Even bit. (0 or 1).

O_E?
[CELL:GEN:RVC:NAMPS:O_E?]
Returns current state of Odd/Even bit.

ORDQ *n*
[CELL:GEN:RVC:NAMPS:ORDQ n]
Specifies value of Order Qualifier for order messages. Range of *n* is 0 to 7.

ORDQ?
[CELL:GEN:RVC:NAMPS:ORDQ?]
Returns current value of Order Qualifier for order messages,

CELL:

GEN:

RVC:

NAMPS:

ORDER *n*
[CELL:GEN:RVC:NAMPS:ORDER n]
Specifies value of ORDER for order messages. Range of *n* is 0 to 31.

ORDER?
[CELL:GEN:RVC:NAMPS:ORDER?]
Returns current value of ORDER for order messages.

RSSI *n*
[CELL:GEN:RVC:NAMPS:RSSI n]
Specifies value of Received Signal Strength in MRI (Mobile Reported Interference) Order Message. Range of *n* is 0 to 7.

RSSI?
[CELL:GEN:RVC:NAMPS:RSSI?]
Returns current value of Received Signal Strength in MRI (Mobile Reported Interference) Order Message.

VMAC *n*
[CELL:GEN:RVC:NAMPS:VMAC n]
Specifies value of Voice Mobile Attenuation Code in Order Messages. Range of *n* is 0 to 7.

VMAC?
[CELL:GEN:RVC:NAMPS:VMAC?]
Returns current value of Voice Mobile Attenuation Code in Order Messages.

6-13-3 REMOTE AMPS/NAMPS CELL SITE MONITOR EXAMPLE

The following macro performs the Receiver Setup for AMPS/NAMPS Cell Site Monitor Operation:

```
*DMC "CELL_SETUP",BEGIN       // Define a macro named CELL_SETUP.
SCREEN:REC                    // Displays the Receiver Operation Screen.
*WAI                          // Waits for command to finish executing.
REC:MOD:USER:MOD DATA         // Selects User Defined FM Data for Receiver
                              // Modulation.
REC:MOD:USER:FILT 30          // Selects a Low-Pass IF Filter with a 30 kHz
                              // cutoff frequency.
REC:MOD:USER:POST:LPAS 15     // Selects a Low-Pass Post Detection Filter
                              // with a 15 kHz cutoff frequency.
REC:AGC:USER:HIGH             // Selects User Defined High Speed for AGC.
M_DEV:RANG:UPP 10             // Sets Deviation Meter Range to 10 kHz.
M_AF:RES 1                    // Sets AF Meter Gate Time to 0.1 sec (1 Hz).
END                           // End macro CELL_SETUP.
```

The following macro queries cellular readings once a min number is received:

```
*DMC "CELL_TEST",BEGIN        // Define macro named Cell_Read.
STRING MIN,ORDER              // Declare strings needed.
VAR SCC,DCC,SID,VMAC          // Declare variables needed.
VAR CHN,CMX,N1,CMAC           // Declare variables needed.
SCREEN:CELL                   // Display AMPS/NAMPS Cell Site Monitor Screen.
*WAI                          // Wait for command execution to finish.
CELL:CHAN 327                 // Select Control Channel 327
DO                            // Loop while no Min is received.
  $=STR(CELL:MIN?)            // Query for a MIN.
  TPAUSE                      // Pauses macro to allow Test Set processing.
UNTIL $ != "-1"               // End of loop.
MIN=$                         // Set MIN string equal to received MIN.
ORDER=STR(CELL:ORDER?)        // Query Order.
SCC=CELL:SCC?                 // Query SCC value.
DCC=CELL:DCC?                 // Query DCC value.
SID=CELL:SID?                 // Query SID value.
VMAC=CELL:VMAC?               // Query VMAC value.
CHN=CELL:VCHAN?               // Query Voice Channel.
CMX=CELL:CMAX_1?              // Query CMAX value.
N1=CELL:N_1?                  // Query N-1 value.
CMAC=CELL:CMAC?               // Query CMAC value.
PPRINT STR(MIN)+","+STR(ORDER)        // Send MIN and Order to RS-232 Connector.
PPRINT SCC,DCC,SID,VMAC       // Send readings to RS-232 Connector.
PPRINT CHN,CMX,N1,CMAC        // Send readings to RS-232 Connector.
END                           // End of macro CELL_TEST.
```

PROGram:STARTup:

NAME "*name*"
[PROGram:STARTup:NAME "name"]
Selects the macro *name* to execute at power up. The startup macro executes after the POWER Switch is pressed and the automatic self test is performed.

The startup macro can be avoided by continually pressing STOP TEST CONTROL Key during the 2 beeps of the 1-2-4 beep sequence at power up.

NAME?
[PROGram:STARTup:NAME?]
Returns the name of the current power up macro designated by a **PROGram:STARTup:NAME** command.

DELETE
[PROGram:STARTup:DELETE]
Deletes the power up designation of the power up macro. After this command is executed, there is no power up macro until a **PROGram:STARTup:NAME** command is executed. This command does not delete the macro itself.

Example:
```
PROG:START:NAME "Main_Menu"   // Selects macro Main_Menu as the power
                              // up macro.  This macro now executes
                              // when the Test Set is turned on.
PROG:START:NAME?              // Queries the name of the current power
                              // up macro.  Main_Menu is returned.
PROG:START:DELETE             // Deletes the power up designation.
```

6-15 FLASH MEMORY FILE DIRECTORY OPERATION

The Flash Memory File Directory allows storage of various files including Calibration Data Sets and allows user to select a macro to be executed from the front panel of the HOST. Operate Flash Memory File Directory using following procedure:

STEP PROCEDURE

1. Press MTRS MODE Key. Press "AUX" Soft Function Key F6 to display Auxiliary Functions Menu. Press 7 DATA ENTRY Key to display File Directory Screen.

File Directory Free: 2551040

Name	Type	Size	Date
SET11	M	23342	06/23/92
SET10	M	23342	06/23/92
STATE1	S	2304	06/23/92
CAL1	B	440	06/23/92
TRACE1	T	406	06/23/92
VALUE01	B	13	06/23/92
STRING_12	A	139	06/23/92

Delete Pack Init Exec Ret

8617099

2. To load a macro from Flash memory into Test Set memory and then execute the macro, perform the following two steps: Move cursor to macro. Press "Exec" Soft Function Key F5.

3. To load a Calibration Data Set or a stored Test Set State (Test Set settings at time of store), move cursor to file. Press "Load" Soft Function Key F5.

4. To delete a file, move cursor to file. Press "Delete" Soft Function Key F2.

   Flash Memory space is released only when a Pack operation is performed.

5. To perform Pack operation, press "Pack" Soft Function Key F3. Pack releases memory space taken by deleted files.

   Do not power off Test Set during Pack operation as files may be lost. Do not pack with a WINDOW open.

6. To Initialize Flash Memory, press "Init" Soft Function Key F4. Initializing clears Flash Memory and all files are lost.

7. To return to Auxiliary Functions Menu, press "Ret" Soft Function Key F6.

Files are stored in Flash Memory using remote commands only. Spectrum Analyzer and Oscilloscope Traces and variables are loaded into Test Set using remote commands.

The following remote commands are used to operate Flash Memory File Directory.

MMEMory:

ATTribute:

DELete *"f"*
[MMEMory:ATTribute:DELete "f"]
Sets the File Attribute of the specified Flash Memory file to DELETE. *f* is Flash Memory file name.

HIDe *"f"*
[MMEMory:ATTribute:HIDe "f"]
Sets the File Attribute of the specified Flash Memory file to HIDE. *f* is Flash Memory file name.

INITialize *"f"*
[MMEMory:ATTribute:INITialize "f"]
Sets the File Attribute of the specified Flash Memory file to INIT. *f* is Flash Memory file name.

PACK *"f"*
[MMEMory:ATTribute:PACK "f"]
Sets the File Attribute of the specified Flash Memory file to PACK. *f* is Flash Memory file name.

ATTribute? *"f"*
[MMEMory:ATTribute? "f"]
Returns the Attribute (DELETE, HIDE, INIT or PACK) of the specified Flash Memory. *f* is Flash Memory file name.

CATalog?
[MMEMory:CATalog?]
Returns Flash Memory status. First number returned is memory space used in bytes. Second number returned is memory space available in bytes. Remainder data is returned in sets of 3 consisting of file name, file type and file size for each file stored in Flash Memory.

CATalog:

ENTRY? *n*
[MMEMory:CATalog:ENTRY? n]
Returns file entry (file name, file type, file size) for given index. Returns $$$ if past end of directory or --- for deleted file. *n* is line number (index) in Flash Memory File Directory. Range of *n* is 0 to 512.

USED?
[MMEMory:CATalog:USED?]
Returns file space used, in bytes.

FREE?
[MMEMory:CATalog:FREE?]
Returns available file space, in bytes.

MMEMory:

DELete *"f"*
[MMEMory:DELete "f"]
Deletes file but does not release memory space until Pack operation is done. *f* is Flash Memory file name.

INITialize
[MMEMory:INITialize]
Erases all files stored in Flash Memory.

INITialize?
[MMEMory:INITialize?]
Returns 1 if file system has been initialized; otherwise 0.

LOAD:

MACRo *"m","f"*
[MMEMory:LOAD:MACRo "m","f"]
Loads macros and variables stored as the file name from Flash Memory into Test Set memory. If *m* is *, designated macro is executed. If *m* is macro name, that macro is executed. If *m* is omitted (""), no macro is executed. *m* is name of designated macro; *f* is Flash Memory file name.

STATe *n,"f"*
[MMEMory:LOAD:STATe n,"f"]
Loads Test Set State stored as *f* from Flash Memory into Auxiliary Functions "Store Parameters Menu" as entry *n*. (*n* = 0 loads current state.) *n* is number of stored state of Test Set; *f* is Flash Memory file name. Range of *n* is 0 to 9.

TRACe:

SCOPe *n,"f"*
[MMEMory:LOAD:TRACe:SCOPe n,"f"]
Loads Oscilloscope trace stored as *f* into Oscilloscope "Store Parameters Menu" as entry *n*. (*n* = 0 loads live trace.) *n* is number of stored trace; *f* is Flash Memory file name. Range of *n* is 0 to 9.

ANLZ *n,"f"*
[MMEMory:LOAD:TRACe:ANLZ n,"f"]
Loads Spectrum Analyzer trace stored as *f* into Spectrum Analyzer "Store Parameters Menu" as entry *n*. (*n* = 0 loads live trace.) *n* is number of stored trace; *f* is Flash Memory file name. Range of *n* is 0 to 9.

DATA *v,"f"*
[MMEMory:LOAD:DATA v,"f"]
Loads variable stored as *f* into Test Set memory with name *v*. *v* is name of variable; *f* is Flash Memory file name.

CALibration *"f"*
[MMEMory:LOAD:CALibration "f"]
Loads Calibration Data from Flash Memory into Test Set memory. *f* is Flash Memory file name.

MMEMory:

PACK
[MMEMory:PACK]
Packs Flash Memory and frees memory space from deleted files.

Do not power off Test Set during Pack function as files may be lost. Do not Pack with a WINDOW open.

STORe:

MACRo *"m","f"*
[MMEMory:STORe:MACRo "m","f"]
Stores all Test Set macros and variables (except free variables) in Flash Memory as *f* with macro specified as designated macro. *m* is name of designated macro; *f* is Flash Memory file name.

STATe *n,"f"*
[MMEMory:STORe:STATe n,"f"]
Stores entry *n* of Auxiliary Functions "Store Parameters Menu" as *f* in Flash Memory. (*n* = 0 stores current state.) *n* is number of stored state of Test Set. Range of *n* is 0 to 9.

TRACe:

SCOPe *n,"f"*
[MMEMory:STORe:TRACe:SCOPe n,"f"]
Stores entry *n* (stored trace) of Oscilloscope "Store Parameters Menu" as *f* in Flash Memory. (*n* = 0 stores live trace.) *n* is number of stored trace; *f* is Flash Memory file name. Range of *n* is 0 to 9.

ANLZ *n,"f"*
[MMEMory:STORe:TRACe:ANLZ n,"f"]
Stores entry *n* (stored trace) of Spectrum Analyzer "Store Parameters Menu" as *f* in Flash Memory. (*n* = 0 stores live trace.) *n* is number of stored trace; *f* is Flash Memory file name. Range of *n* is 0 to 9.

DATA *v,"f"*
[MMEMory:STORe:DATA v,"f"]
Stores variable *v* into Flash Memory as *f*. *v* is name of variable; *f* is Flash Memory file name.

CALibration *"f"*
[MMEMory:STORe:CALibration "f"]
Stores Test Set Calibration Data into Flash Memory. *f* is Flash Memory file name.

TYPE? *"f"*
[MMEMory:TYPE? "f"]
Returns file type. Returns null string if file does not exist. *f* is Flash Memory file name.

Error messages are returned to HOST when an error occurs. Refer to Table 6-2 for description of error messages.

ERROR NUMBER	ERROR DEFINITION	DESCRIPTION
220	Parameter Error	Incorrect number of parameters were entered with command.
224	Illegal Parameter Value	A parameter entered was not appropriate for command.
225	Out of Memory	Insufficient memory space to perform command.
250	Flash Storage Error	Indicates Flash Memory could not be erased or data could not be stored in Flash Memory.
253	Corrupt Media	Indicates Flash Memory not properly initialized. Initialize Flash Memory.
254	Media Full	Indicates insufficient Flash Memory space to perform command.
255	Directory Full	Indicates command not performed because 512 file names have been used.
256	File Name Not Found	Specified file not stored in Flash Memory.
257	File Name Error	Indicates command attempted to create file name already stored or file name syntax incorrect.

Table 6-2 Flash Memory Error Messages

6-16 GENERIC MEASURE COMMANDS

Generic commands use optional values, *e* to signify expected value and *r* to signify resolution. Expected values help determine Meter range. If *r* is omitted, the last resolution value is used.

MEASure:

AUDio? [*e*,*r*]
[MEASure:AUDio? [e,r]]
Returns an AF Meter frequency counter reading in Hz (0.0 to 200000.0).

CURRent:

AC? [*e*]
[MEASure:CURRent:AC? [e]]
Returns a DMM ac current reading in amps (0.00000 to 19.990).

DC? [*e*]
[MEASure:CURRent:DC? [e]]
Returns a DMM dc current reading in amps (0.00000 to 19.990).

FREQuency? [*e*,*r*]
[MEASure:FREQuency? [e,r]]
Returns signal frequency reading in kHz (250.0 to 2010000.0 kHz).

MIC?
[MEASure:MIC?]
Returns a 0 if receiving MIC/ACC Input or 1 otherwise.

PHASe? [*e*]
[MEASure:PHASe? [e]]
Returns a Phase Meter reading in radians (0.00 to 10.00).

POWer? [*e*]
[MEASure:POWer? [e]]
Returns a Power Meter reading in mW (0.0 to 100000.0).

RESistance? [*e*]
[MEASure:RESistance? [e]]
Returns a DMM resistance reading in kΩ (0.0000 to 19990).

SINAD? [*r*]
[MEASure:SINAD? [r]]
Returns a SINAD Meter reading in dB (3.0 to 40.0). Select .1 or .5 as optional *r* in dB.

SQUelch?
[MEASure:SQUelch?]
Returns a 1 if squelch broken, 0 if squelch is unbroken.

MEASure:

TEMPerature:

AMBient?

[MEASure:TEMPerature:AMBient?]

Returns the ambient temperature in °C (0.00000 to 100.00000).

POWer?

[MEASure:TEMPerature:POWer?]

Returns the Power Termination temperature in °C (0.00000 to 100.00000).

VOLTage:

AC? [*e*]

[MEASure:VOLTage:AC? [e]]

Returns a DMM ac voltage reading in volts (0.0000 to 1000.00).

DC? [*e*]

[MEASure:VOLTage:DC? [e]]

Returns a DMM dc voltage reading in volts (0.0000 to 1000.00).

SUPply? *n*

[MEASure:VOLTage:SUPply? n]

Returns a voltage measurement of *n* power supply in volts (0.00 to 20.00). Select -15, 5 or 15 for *n*.

6-17 INITIATE AND FETCH COMMANDS

INITiate and **FETCh** commands break the measure process into two commands. **INITiate** commands prepare the meter and **FETCh** commands return the reading. The event register of the Operation Instrument Status Register indicates when an **INITiate** command is completed (see example). Figure 2-6 displays the Operation Instrument Status bits.

INITiate:

AF
[INITiate:AF]
Prepares the Audio Frequency Meter for a **FETCh:** command to take a reading.

DMM
[INITiate:DMM]
Prepares the Audio Frequency Meter for a **FETCh:** command to take a reading.

RF
[INITiate:RF]
Prepares the Frequency Error Meter for a **FETCh:** command to take a reading.

FETCh:

AF?
[FETCh:AF?]
Returns Audio Frequency Meter reading in kHz. Must be used with **INITiate** command.

DMM?
[FETCh:DMM?]
Returns Digital Multimeter reading. For ACV and DCV Multimeter Function, reading is returned in V. For ACC and DCC Multimeter Function, reading is returned in A. For Ohm Multimeter Function, reading is returned in kΩ. Must be used with **INITiate** command.

RF?
[FETCh:RF?]
Returns Frequency Error Meter reading in kHz. Must be used with **INITiate** command.

The following example reads the Operation Instrument Register until the RF Meter is ready to be read.

Example:

```
*DMC "RF_Read",BEGIN         // Define macro named RF_Read.
N=0                          // Set variable equal to zero.
SCREEN:FREQ                  // Display Frequency Error Meter Operation
                             // Screen.
SCREEN:USER                  // Display User Screen.
*CLS                         // Clear all condition and event registers.
INIT:RF                      // Initialize for a RF Meter reading.
WHILE (N&8)=0                // Loop until Frequency Error Meter bit of
  N=STAT:OPER:INSTR:EVENT?   // Operation Instrument event register is
                             // set to 1.
WEND
PRINT FETCH:RF?              // Print Frequency Error Meter reading.
END                          // End of macro RF_Read.
```

6-18 AUXILIARY (SPECIAL TEST) COMMANDS

Auxiliary (Sp Tst) commands are used to communicate with the Sp Tst through the internal SCSI bus. Because of the vast command possibilities, the HOST does not check the Auxiliary commands for correct string syntax.

AHIT?
[AHIT?]
Returns a 1 if there is input waiting from the Sp Tst.

AUX *"string"*
[AUX "string"]
Issues commands, as strings, to the Sp Tst.

Example:
```
AUX "DELAY 200"        // The DELAY 200 command is passed to and
                       // executed by the Sp Tst.
```

AUX? *"string?"*
[AUX? "string?"]
Issues queries, as strings, to the Sp Tst.

Example:
```
AUX? "IDN?"            // The IDN? query is passed to the Sp Tst.
                       // The Sp Tst executes the query and
                       // returns the result.
```

INDEX

C (cont)

D

E

I

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

I (cont)

K

L

M

M (cont)

N

O

P

P (cont)

Q

R

R (cont)

S

S (cont)

S (cont)

T

U

V

W

X