COMMUNICATIONS SERVICE MONITOR

2944B, 2945B and 2948B

Programming Manual

Document part no. 46892/683

Communications Service Monitors

2944B, 2945B and 2948B

PROGRAMMING MANUAL

Document part no. 46892/683 (PDF version)

Based on Issue 5 of the printed manual.

28 January 2008

About this manual

This manual explains how to write programs for the Service Monitors 2944B, 2945B or 2948B. The programs can provide remote control for all functions of the Service Monitors as well as automatic testing for cellular and trunking systems. The commands contained in this manual relate to main software version 5.xx or later and systems software version 5.xx. Note that some commands are not available to certain models of Service Monitor.

Intended audience

Persons engaged on work relating to the automatic testing of RF communications equipment. It is assumed that the reader will be familiar with telecommunication terms used in trunking, cellular and avionics radio systems.

Structure

Chapters 1, 2 and 3

These chapters cover all aspects of programming in MI-BASIC.

Chapters 4, 5 and 6

These chapters cover all aspects of programming to remotely control the Service Monitor over the GPIB or RS232 interfaces.

Associated publications

Refer to the appropriate operating manual:

46882/682	for 2945B
46882/692	for 2948B
46882/744	for 2944B

for an up-to-date list of associated publications.

Contents

Chapter 1
GETTING STARTED

Contents

List of figures

What is MI-BASIC?

MI-BASIC is a programming language devised for use in test equipment manufactured by Aeroflex. The programs are run by an MI-BASIC interpreter which resides in the equipment. The software of the Service Monitor contains test procedures specifically written for radio testing; some are specific to Cellular or Trunked radio requirements, while the remainder are general parametric tests. These can all be accessed by MI-BASIC test programs.

A number of MI-BASIC programs are included in the software of the Service Monitor. These have been structured to meet the requirements for automatic tests on many cellular and trunked mobile radio systems.

User Defined Test Programs, written by the user, can be downloaded to the Service Monitor.

The User Defined Programs, and those which are resident in the Service Monitor, can be run manually from the front panel or by remote control.

MI-BASIC and the Service Monitor

MI-BASIC adds a controller facility to the Service Monitor. The facilities which it provides include:

- Program control
- Reading the keyboard
- Manipulation of variables
- Printing to either the display and/or results store
- Control of external hardware via the accessory port or RS232 port
- Parametric measurements
- Execution of internal tests

Writing programs

The syntax for MI-BASIC is outlined in Chapter 2 of this manual. The statements that are recognized by the interpreter are listed in alphabetical order, with definitions, correct syntax and examples of use. MI-BASIC programs can be written using any text editor.

The recommended layout for a program is as follows:

- **LABELS** and **REM** statements should be at the left margin. Other program lines should generally start one tab in from the left margin (see below).
- Where **IF/THEN** statements are used, the program segment following the **THEN** part of the statement should be on the next line and one tab in from the start of the **IF** statement.
- Subroutines should be placed after the **END** statement.
- Text can be entered in upper, lower or mixed case.

An example of a program is given below.

This program displays a menu on the screen of the Service Monitor and requests that the user selects one of the options by using the data keys. When a valid key has been pressed, the program branches to the appropriate subroutine and an acknowledgement is displayed.

If an invalid key is pressed an error message is displayed.

After a subroutine has been executed, the program loops back to the menu and then restarts.

Program example

```
LABEL menu
    CLS
    CLRSTORE
    LPRINT "Select a program number using data keys"
    LPRINT "1. Program 1"
    LPRINT "2. Program 2"
    LPRINT "3. Program 3"
    LPRINT "4. Program 4"

LABEL start
    GETKEY char1
    IF char1 = 0 THEN
      GOTO stop
    ENDIF
    IF char1 < 81 | char1 > 84 THEN
      CLS
      CLRSTORE
      LPRINT "Invalid program number..."
      LPRINT "Please try again..."
      WAIT 1500
      GOTO menu
    ENDIF

    prog_num = char1 - 80

    CLRSTORE
    IF prog_num = 1 THEN
      GOSUB prog1
    ELSEIF prog_num = 2 THEN
      GOSUB prog2
    ELSEIF prog_num = 3 THEN
      GOSUB prog3
    ELSEIF prog_num = 4 THEN
      GOSUB prog4
```

```
    ENDIF
    WAIT 3000
    GOTO menu
LABEL stop
    CLRSTORE
    END

REM Subroutines go after the END statement
LABEL prog1
    LPRINT "This is program 1"
    RETURN

LABEL prog2
    LPRINT "This is program 2"
    RETURN

LABEL prog3
    LPRINT "This is program 3"
    RETURN

LABEL prog4
    LPRINT "This is program 4"
    RETURN
```

Downloading programs, PC to Service Monitor

To download a program to the Service Monitor from a PC or other programming device, the two units must be connected via an RS232 or GPIB link. The program is then downloaded using remote commands.

Note: Before a program is downloaded, the system appropriate to the program must have been selected on the Service Monitor.

After a user-defined program has been downloaded, select USER DEFINED TEST from the TEST PROGRAM MENU. Run the test to ensure that no errors exist in the program.

The Service Monitor has capability for storing one User-Defined program only, downloading a new program clears the memory area used to store them.

The remote commands required to download a program are:

CTRL <A>

PROG:LEARN ACTIVE

PROG:LEARNLINE

PROG:LEARN INACTIVE

CTRL <D>

A brief description of these commands follows.

PROG:LEARN ACTIVE

This command must be sent first; it clears the memory area used to store the user program, clears any relevant flags regardless of the state of the current system, and indicates to the Service Monitor that a user program is about to be downloaded.

On receipt of this command, the AUTOTEST screen is displayed and **** LEARN MODE **** appears on the screen. If a **PROG:LEARNLINE** command is sent when LEARN is INACTIVE, an error is returned.

The status of **PROG:LEARN** can be found by using the **PROG:LEARN?** query command. Either 'ACTIVE' or 'INACTIVE' will be returned to the controlling device.

PROG:LEARNLINE

This command downloads the program line by line. Each line of the program is enclosed within single quotes, and prefixed by **PROG:LEARNLINE**. Using the example program on the previous page, the first few lines of the program are sent as:

```
PROG:LEARNLINE 'LABEL menu'
PROG:LEARNLINE '  CLS'
PROG:LEARNLINE '  CLRSTORE'
```

This is repeated for every line of the program.

PROG:LEARN INACTIVE

This command is sent following the last line of the program to terminate the download process.

Step-by-step procedure for downloading

1. Write the program on a PC or other programming device, using a suitable text editor program. Ensure that your MI-BASIC program conforms to the statements and syntax described in Chapter 2.
2. Connect the Service Monitor to the PC or other controlling device through a GPIB or RS232 link.
3. Set the Service Monitor to operate over the appropriate remote system. Press [HELP SETUP], *[Setup], [Setup Page 2]*, then *[Remote Control]* to select either RS232 Remote or GPIB Remote as appropriate. Use the *[return]* key to return to the previously selected display.
4. Select the relevant system on the Service Monitor, (PMR in this example) from the SYSTEM SELECTION MENU by using the key sequence, [SYSTEM], *[PMR]*. The display shown in Fig. 1-1 appears.
5. Download the program. Send <control> A, then PROG:LEARN ACTIVE, followed by the program, line by line, enclosed within single quotes and prefixed by PROG:LEARNLINE (see the example on page 1-5). The Service Monitor will automatically go into Auto mode, remote operation. Terminate the download by sending PROG:LEARN INACTIVE, then <control> D.
6. When the program has been downloaded, press the [HELP/SETUP] key to return to local operation.
7. The program can now be run. This can be manually from the Service Monitor front panel, or by remote control.
8. Press the *[PROGRAM]* key, followed by the *[user defined]* key to select the downloaded program; the display will be as shown in Fig. 1-2.
9. **Note:** Pressing *[user defined]* has no effect unless a user-defined program is present in the Service Monitor.
10. To run from the Service Monitor:

    Press the *[start]* soft key.
11. To run from remote, send the command:

    PROG:RUNSTATE RUN

    Any syntax errors in the program will show when it is run for the first time. Syntax errors are shown in the results store and the *[start]* soft key (top right) is highlighted and labeled error. Any errors in the program must be corrected and then download again, replacing the errored program.

    The run status of the instrument can be checked from the remote device by using the **PROG:RUNSTATE?** query command. This will return either STOP, RUN, or PAUSE.

The example program is shown here as it is downloaded.

```
PROG:LEARN ACTIVE
PROG:LEARNLINE ‘LABEL menu’
PROG:LEARNLINE ‘       CLS’
PROG:LEARNLINE ‘       CLRSTORE’
PROG:LEARNLINE ‘       LPRINT "Select a program number using data
keys"‘
PROG:LEARNLINE ‘       LPRINT "1. Program 1”’
PROG:LEARNLINE ‘       LPRINT "2. Program 2”’
PROG:LEARNLINE ‘       LPRINT "3. Program 3”’
PROG:LEARNLINE ‘       LPRINT "4. Program 4”’
PROG:LEARNLINE ‘’
PROG:LEARNLINE ‘LABEL start’
PROG:LEARNLINE ‘       GETKEY char1’
PROG:LEARNLINE ‘       IF char1 = 0 THEN’
PROG:LEARNLINE ‘              GOTO stop’
PROG:LEARNLINE ‘       ENDIF’
PROG:LEARNLINE ‘       IF char1 < 81 | char1 > 84 THEN’
PROG:LEARNLINE ‘              CLS’
PROG:LEARNLINE ‘              CLRSTORE’
PROG:LEARNLINE ‘              LPRINT "Invalid program number..."‘
PROG:LEARNLINE ‘              LPRINT "Please try again..."‘
PROG:LEARNLINE ‘              WAIT 1500’
PROG:LEARNLINE ‘              GOTO menu’
PROG:LEARNLINE ‘       ENDIF’
PROG:LEARNLINE ‘’
PROG:LEARNLINE ‘       prog_num = char1 - 80’
PROG:LEARNLINE ‘’
PROG:LEARNLINE ‘       CLRSTORE’
PROG:LEARNLINE ‘       IF prog_num = 1 THEN’
PROG:LEARNLINE ‘              GOSUB prog1’
PROG:LEARNLINE ‘       ELSEIF prog_num = 2 THEN’
PROG:LEARNLINE ‘              GOSUB prog2’
PROG:LEARNLINE ‘       ELSEIF prog_num = 3 THEN’
PROG:LEARNLINE ‘              GOSUB prog3’
PROG:LEARNLINE ‘       ELSEIF prog_num = 4 THEN’
PROG:LEARNLINE ‘              GOSUB prog4’
PROG:LEARNLINE ‘       ENDIF’
PROG:LEARNLINE ‘       WAIT 3000’
PROG:LEARNLINE ‘       GOTO menu’
PROG:LEARNLINE ‘’
PROG:LEARNLINE ‘LABEL stop’
PROG:LEARNLINE ‘       CLRSTORE’
PROG:LEARNLINE ‘       END’
PROG:LEARNLINE ‘’
PROG:LEARNLINE ‘REM Subroutines go after the END statement’
PROG:LEARNLINE ‘LABEL prog1’
PROG:LEARNLINE ‘       LPRINT “This is program 1”’
PROG:LEARNLINE ‘       RETURN’
PROG:LEARNLINE ‘’
PROG:LEARNLINE ‘LABEL prog2’
PROG:LEARNLINE ‘       LPRINT “This is program 2”’
PROG:LEARNLINE ‘       RETURN’
PROG:LEARNLINE ‘’
PROG:LEARNLINE ‘LABEL prog3’
PROG:LEARNLINE ‘       LPRINT “This is program 3”’
PROG:LEARNLINE ‘       RETURN’
PROG:LEARNLINE ‘’
PROG:LEARNLINE ‘LABEL prog4’
PROG:LEARNLINE ‘       LPRINT “This is program 4”’
PROG:LEARNLINE ‘       RETURN’
PROG:LEARN INACTIVE
```

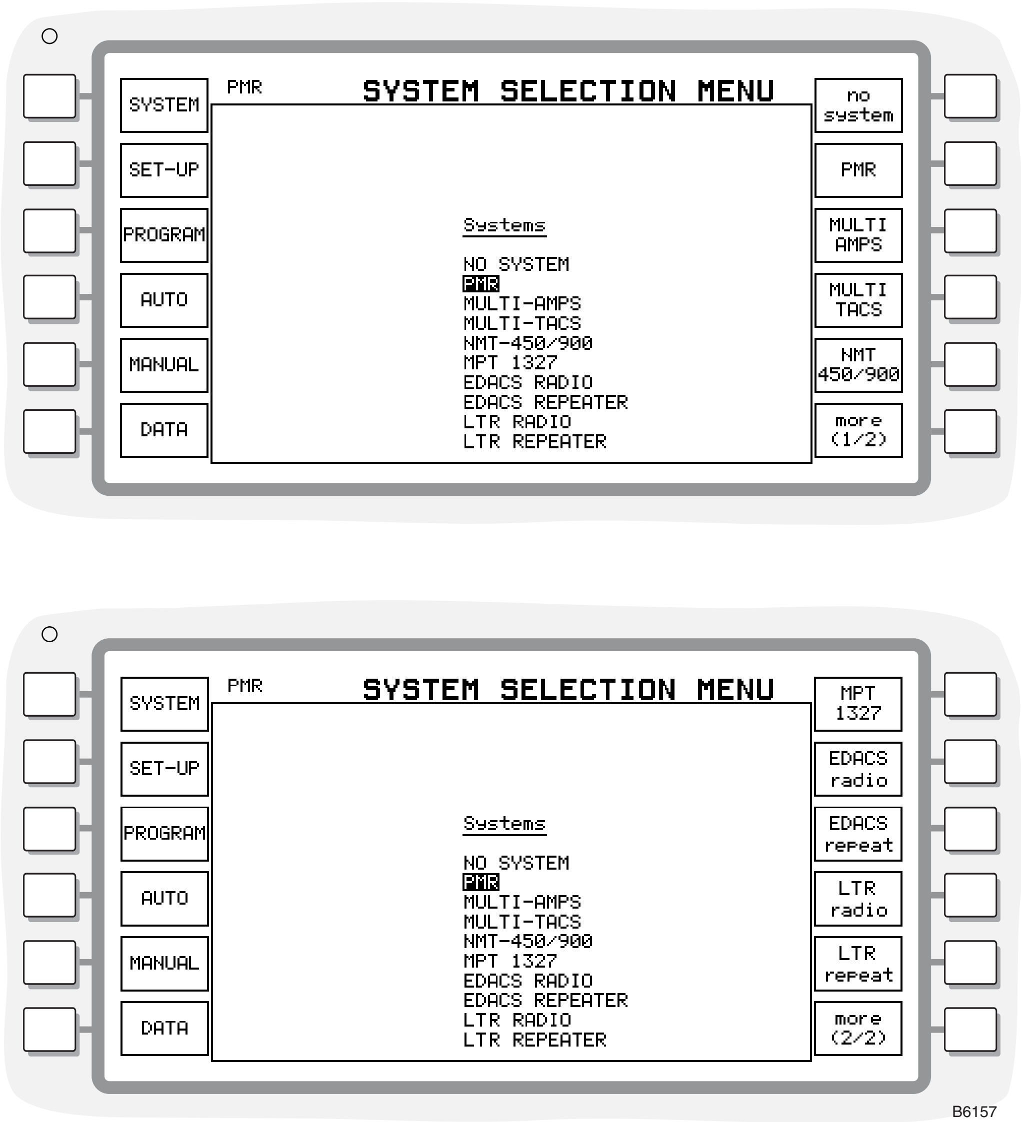


Fig. 1-1 System selection menu

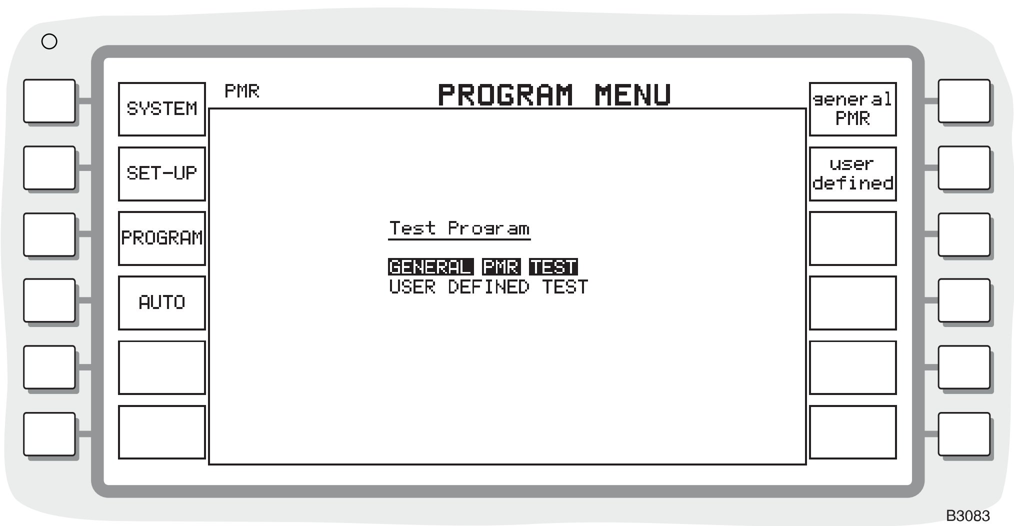


Fig. 1-2 Program menu

Chapter 2
MI-BASIC REFERENCE GUIDE

Contents

List of statements

List of figures

List of tables

Syntax information

<num>

Description: A number. This can be in the range $1.797\times10^{+308}$ to 2.225×10^{-308}. Only decimal notation is supported. See also numeric operators.

Examples: Valid numeric entries are:-

```
6, 12, 0.34, .56, 123.45e-12, 1.0e003
```

<num var>

Description: A numeric variable. This is a name for a number that varies. Maximum variable name length is 24 characters. See also numeric operators.

Examples: In these examples, `fred` and `tom` are <num var>s.

```
fred = 12
```

```
tom = 1.07e2
```

<num expr>

Description: A numeric expression. This is a mathematical expression that has <num> and / or <num var> for its terms, which are related by numerical operators. In its simplest form, a numeric expression may be a number or a numeric variable. See also numeric operators.
In this example, `harry`, `fred` and `tom` are <num var>s, `3` is a <num>, and `+` and `*` are numerical operators. The whole line constitutes a <num expr>.

Example:

```
harry = (fred + 3) * tom
```

<str>

Description: A string. This is a line of ASCII characters and is denoted by being enclosed within quotes - `" "`. See also string operators.

Examples:

```
"Hello world"
"It's raining again!"
```

<str var>

Description: A string variable. This is a name for a string that varies. Maximum variable name length is 24 characters. See also string operators.
A string variable is distinct from a numeric variable in that it always ends in a `$` sign. See also string operators.

Examples:

```
hello$ = "Hello world"
again$ = "It's raining again!"
```

<str expr>

Description: A string expression. This is an expression that has <str> and / or <str var> for its terms, which are related by string operators. See also string operators.
In the example below, `hello$`, `again$`, `first$`, `world$`, and `raining$` are <str var>s, `"again"` and `"world"` are <str>s, `+` and `–` are string operators. Each of the three lines is a <str expr>.

Example:

```
Let hello$ = "Hello world" and
    again$ = "It's raining again!"
If  first$ = again$ - " again" then
    first$  = "It's raining!"  (extraction)
If  world$ = hello$ - "world" then
    world$ = "Hello "    (extraction)
If  raining$ = world$ + first$ then
    raining$ = "Hello It's raining!" (concatenation)
```

CAPITAL LETTERS

Description: A keyword. **BOLD CAPITAL LETTERS** denote a keyword.

Example: **DEFAULT**

<num *n*>
<num var *n*>
<str *n*>
<str var *n*>

Description: These names are used in commands requiring more than one item to indicate the type and number of possible values.
If *n* is missing, it implies one item.
In the example below, the command **NUMRESULTS** must be followed by a <num var>. A further seven <num var>s are optional.

Syntax: **NUMRESULTS** <num var1> [,...,<num var8>]

Example: `NUMRESULTS result, act, tgt`

<name>

Description: Identifier for a keyword, it is used in program control. Maximum length is 24 characters. Invalid <name>s are keywords, numbers, string variables, strings.

Syntax: **LABEL** <name>

Example: `LABEL FRED`

<params>

Description: The *Test Parameters* are defined in Chapter 3 and cross referenced to relevant *System* or *Systems.*

[ ]

Description: An option. Anything enclosed within square brackets is an optional part of the statement.
This example shows that the keyword **GETKEY** can be used by itself or it can have a <num var> after it.

Syntax: **GETKEY** [<num var>]

Examples:

```
GETKEY
GETKEY escape
```

| |

Description: A choice. Parallel lines represent a choice, one of the terms within the parallel lines must be chosen, unless the parallel lines are enclosed within square brackets.
In this example, after the keyword **SOFTKEY**, you must choose either a <num> or a <num var>; after that you must also choose from NORMAL, BRIGHT, DELETE or CLEAR. There is then a choice between using <str1> or <str1> <str2>, but this is an optional part of the statement.

Syntax:

SOFTKEY	<num> <num var>	,	NORMAL BRIGHT DELETE CLEAR	[<str1> <str1><str2>]

Examples:

```
SOFTKEY 2, NORMAL "FRED"
SOFTKEY 7, BRIGHT "test" "start"
SOFTKEY 33, DELETE
fred = 34
SOFTKEY fred, CLEAR
```

...

Description: Repeat. Previous item(s) may be repeated as necessary.

In this example, the keyword **STRRESULTS** must be followed by a <str var>. A further seven <str var>s are optional.

Syntax: **STRRESULTS** <str var1> [, ... ,<str var8>]

Examples:

```
STRRESULTS res1$
STRRESULTS title$, stat$, comment1$
```

<statement>

Description: A statement. An MI-BASIC expression which includes keyword and relevant syntax.
In this example, the keyword **GETKEY** is used with a <num var> called `choice`. When a key is pressed, its decimal value is put into the <num var> `choice`.

Example:

```
GETKEY  choice
```

<program segment>

Description: A program segment. Two or more <statement>s.
In the example, **GETKEY** stops the program until a key is pressed; the decimal value of the key is returned into `choice`. If the value of the key pressed is 2, then the program will print `choice = 2` on the screen, otherwise the program will continue.

Example:

```
GETKEY choice
IF choice = 2 THEN
    LPRINT "choice = 2"
ENDIF
```

<program>

Description: A program. A program is one or more statements that are executed in sequence.
This example program fills the screen with the * character. The screen is filled one character at a time, column by column.

Example:

```
start_column = 0
    start_row = 0
    column = start_column
    row = start_row
LABEL start
    PRINT AT column,row; "*"
    row = row +1

    IF row > 26 THEN
      column = column + 1
      row = start_row
    ENDIF

    IF column > 45 THEN
      GOTO stop
    ENDIF

GOTO start

LABEL stop
END
```

Operators

Parenthesis

Expressions can contain a high degree of complexity. Parentheses are allowed to any level and should be used whenever there is a possibility of ambiguity.

Precedence

The table below lists the operators in groups of equal precedence, the highest priority group first.

Symbol	*Description*
!	NOT
( )	Brackets
+	Plus, [Unary]
–	Minus, [Unary]
LOG	Natural logarithm
LOG10	Base 10 logarithm
EXP	Natural exponent
EXP10	Base 10 exponent
ABS	Absolute value
/	Divide
*****	Multiply
MOD	Modulus or remainder
+	Plus, [Binary]
–	Minus, [Binary]
<<	Left shift
>>	Right shift
>=	Greater than or equal to
>	Greater than
<=	Less than or equal to
<	Less than
=	Equal to
<>	Not equal to
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
AND	AND
XOR	XOR
OR	OR

Numeric operators

Arithmetic operators (binary)

Operator	*Description*
+	Plus
–	Minus
/	Divide
*****	Multiply
MOD	Modulus or remainder

Example: The function of MOD is to give the remainder of the first value divided by the second value.

If `result1 = 25 MOD 10` then
`result1 = 5`

If `result2 = 25 MOD 5` then
`result2 = 0`

If `result3 = 351 MOD 251` then
`result3 = 100`

Arithmetic operators (unary)

Operator	*Description*
+	Plus.
–	Minus.
LOG	Natural logarithm.
LOG10	Base 10 logarithm.
EXP	Natural exponent
EXP10	Base 10 exponent
ABS	Absolute value

Conditional operators

Operator	*Description*
=	Equal to
< >	Not equal to
> =	Greater than or equal to
>	Greater than
< =	Less than or equal to
<	Less than

Bitwise operators (binary)

~

Description: Negate (Bitwise NOT, unary)

Example: If `result = ~(-101)` then
`result = 100`

Explanation:

$$-101_{10} = 1\ 1001\ 1011_2$$
$$\text{invert} = 0\ 0110\ 0100_2$$
$$0\ 0110\ 0100_2 = 64_{16}$$
$$64_{16} = 100_{10}$$
$$\therefore \text{result} = 100$$

Bitwise operators (unary)

Operator

<<

Description: Left shift

Example: If `result = 25 << 2` then `result = 100`

Explanation:

$$
\begin{aligned}
25_{10} &= 0001\ 1001_2 \\
0001\ 1001_2 << 2 &= 0110\ 0100_2 \\
.\qquad 0110\ 0100_2 &= 100_{10} \\
\therefore \text{result} &= 100
\end{aligned}
$$

>>

Description: Right shift

Example: If `result = 400 >> 2` then `result = 100`

Explanation:

$$
\begin{aligned}
400_{10} &= 1\ 1001\ 0000_2 \\
1\ 1001\ 0000_2 >> 2 &= 0110\ 0100_2 \\
.\qquad 0110\ 0100_2 &= 100_{10} \\
\therefore \text{result} &= 100
\end{aligned}
$$

&

Description: Bitwise AND

Example: If `result = 255 & 100` then `result = 100`

Explanation:

$$
\begin{aligned}
255_{10} &= 1111\ 1111_2 \\
100_{10} &= 0110\ 0100_2 \\
1111\ 1111\ \&\ 0110\ 0100_2 &= 0110\ 0100_2 \\
0110\ 0100_2 &= 100_{10} \\
\therefore \text{result} &= 100
\end{aligned}
$$

|

Description: Bitwise OR

Example: If `result = 96 | 4` then `result = 100`

Explanation:

$$
\begin{aligned}
96_{10} &= 0110\ 0000_2 \\
4_{10} &= 0000\ 0100_2 \\
0110\ 0000 \mid 0000\ 0100_2 &= 0110\ 0100_2 \\
0110\ 0100_2 &= 100_{10} \\
\therefore \text{result} &= 100
\end{aligned}
$$

^

Description: Bitwise XOR

Example: If `result = 255 ^ 155` then `result = 100`

Explanation:

$$
\begin{aligned}
255_{10} &= 1111\ 1111_2 \\
155_{10} &= 1001\ 1011_2 \\
1111\ 1111_2\ \wedge\ 1001\ 1011_2 &= 0110\ 0100_2 \\
0110\ 0100_2 &= 100_{10} \\
\therefore \text{result} &= 100
\end{aligned}
$$

Operator

AND

Description: AND

Example: If `result = 255 AND 100` then `result = 1`

Explanation:

255	=	non-zero
non-zero	=	true
true	=	1
100	=	non-zero
non-zero	=	true
true	=	1
1 AND 1	=	1
∴ result	=	1

OR

Description: OR

Example: If `result = 96 OR 4` then `result = 1`

Explanation:

96	=	non-zero
non-zero	=	true
true	=	1
4	=	non-zero
non-zero	=	true
true	=	1
1 OR 1	=	1
∴ result	=	1

XOR

Description: XOR

Example: If `result = 96 XOR 4` then `result = 0`

Explanation:

96	=	non-zero
non-zero	=	true
true	=	1
4	=	non-zero
non-zero	=	true
true	=	1
1 XOR 1	=	0
∴ result	=	0

Logical operators (unary)

Operator

!

Description: NOT

Example: If `result = !(0) * 100` then
`result = 100`

Explanation:

$$!(0) = 1$$
$$1 * 100 = 100$$
$$\therefore \text{result} = 100$$

String operators

Operator

+

Description: Plus (concatenate)

Example:

Let `hello$ = "Hello"`
and `world$ = " world"`

If `result$ = hello$ + world$`
then`Lprint result$`
will print the string `Hello world`

–

Description: Minus (search and remove)

Example:

Let `helloworld$ = "Hello world"`
and `world$ = " world"`

If `result$ = helloworld$ - world$`
then`Lprint result$`
will print the string `Hello`

Statements

ACCESSORY

Function: To set the optional accessory port's logic levels high or low.

Syntax: **ACCESSORY** | LOGIC0 / LOGIC1 / LOGIC2 / LOGIC3 | | HIGH / LOW |

Remarks: LOGIC0, LOGIC1, LOGIC2 and LOGIC3 are control lines for operating ancillary equipment. They are brought to pins on the ACCESSORY socket of the optional parallel printer port. (See table below). This option is fitted to the rear panel of the instrument.

Example: This statement sets the accessory port's LOGIC1 level high.

```
ACCESSORY LOGIC1 HIGH
```

Pin Connections

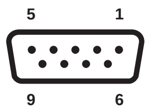


Fig. 2-1 Rear accessory port socket connections (as seen facing panel)

The table below is included for information only. No inference is made as to the uses to which this connector can be used for connecting any equipment other than approved devices manufactured by Aeroflex. The functions of the plug contacts are as shown below.

Table 2-1 Rear accessory port connections

Contact	Function
1	+5 V
2	Logic line 3 or logic contact 3(a) †
3	Logic line 2 or logic contact 2(a) †
4	Logic line 1 or logic contact 1(a) †
5	Logic line 0 or logic contact 0(a) †
6	logic contact 3(b)
7	Logic contact 2(b)
8	Logic contact 1(b)
9	Logic contact 0(b)

Note †

Programmable in SYSTEM MODE.

Open drain drive pulled up to +5 V with 4.7 kΩ. Max sink current 10 mA for V_{OL} = 1 V.

AFGENn FREQ

Please refer to the note on the usage of this command on page 2-66

Function: To set the designated AF generator's frequency.

Syntax: **AFGEN*n* FREQ** | <num> / <num var> / <num expr> | [| Hz / kHz |]

If the unit is omitted, the default unit is Hz.

Remarks: The *n* represents which generator is to be set, where *n* = 1 or 2. The frequency is set using either a number, a numeric variable or a numeric expression. The unit is optional and is set using Hz or kHz. The frequency range for each generator is 10 Hz to 25 kHz.

This command is equivalent to using the *[Audio GEN]*, *[Gen 1/Gen 2]*, [FREQ], and DATA entry keys.

Example: All of these examples assume that the AF generators are ON. This can be done from MI-BASIC using the following statement, where *n* = 1 or 2:-

```
AFGENn STATUS ON
```

Also, the level of the designated generator must be non-zero and the sum of all three generator levels must be less than 5 V.

1. This statement uses a number for the frequency without specifying the optional units and sets AF generator 1 frequency to 100 Hz.

```
AFGEN1 FREQ 100
```

2. This program segment sets AF generator 2 frequency to 5 kHz by using a numeric variable for the frequency and specifying the unit to be kHz.

```
value = 5
AFGEN2 FREQ value kHz
```

3. This program segment sets AF generator 2 frequency to 6 kHz by using a numeric expression for the frequency and specifying the unit to be kHz. This command may be part of a loop where the frequency is incremented by 1 kHz each time the loop is passed through.

```
value = 5
AFGEN2 FREQ value + 1 kHz
```

Note: The Service Monitor has two generators, each of which can be configured as an AF generator or as a modulation generator. The following combinations are possible:-

AFGEN 1 and MODGEN 2
AFGEN 2 and MODGEN 1
AFGEN1 and AFGEN 2
MODGEN1 and MODGEN 2

The act of turning a generator ON will allocate that generator.

AFGEN*n* LEVEL

Please refer to the note on the usage of this command on page 2-66

Function: To set the designated AF generator's level.

Syntax: **AFGEN*n* LEVEL** | <num> / <num var> / <num expr> | [UVOLT / MVOLT / VOLT]

The default unit is V.

Remarks: The *n* represents which generator is to be set, where *n* = 1 or 2. The level is set using either a number, a numeric variable or a numeric expression. The level range for each generator is 0 to 4.095 V.

This command is equivalent to using the *[Audio GEN]*, *[Gen 1/Gen 2]*, [LEVEL], and DATA entry keys..

Example: All of these examples assume that the AF generators are ON. This can be done from MI-BASIC using the following statement, where *n* = 1 or 2:-

```
AFGENn STATUS ON
```

1. This statement uses a number for the level and sets the AF generator 1 level to 1 V.

```
AFGEN1 LEVEL 1
```

2. This program segment uses a numeric variable for the level and sets the AF generator 2 level to 0.5 V.

```
value = 0.5
AFGEN2 LEVEL value
```

3. This program segment uses a numeric expression for the level and sets the AF generator 2 level to 3.5 V. The statement that contains the AFGEN command may be part of a loop where the level is incremented by 0.5 V each time the loop is passed through.

```
value = 3
AFGEN2 LEVEL value + 0.5
```

Note: The Service Monitor has two generators, each of which can be configured as an AF generator or as a modulation generator. The following combinations are possible:-

AFGEN 1 and MODGEN 2
AFGEN 2 and MODGEN 1
AFGEN1 and AFGEN 2
MODGEN1 and MODGEN 2

The act of turning a generator ON will allocate that generator.

AFGEN*n* STATUS

Please refer to the note on the usage of this command on page 2-66

Function: To switch the designated AF generator ON or OFF.

Syntax: **AFGEN*n* STATUS** | ON
OFF |

Remarks: The *n* represents which generator is to be set, where *n* = 1 or 2. This command is equivalent to using the [ON OFF] key.

Example:

1. This statement switches AF generator 1 ON.

```
AFGEN1 STATUS ON
```

2. This statement switches AF generator 2 OFF.

```
AFGEN2 STATUS OFF
```

Note: The Service Monitor has two generators, each of which can be configured as an AF generator or as a modulation generator. The following combinations are possible:-

AFGEN 1 and MODGEN 2
AFGEN 2 and MODGEN 1
AFGEN1 and AFGEN 2
MODGEN1 and MODGEN 2

The act of turning a generator ON will allocate that generator.

AFGEN*n*TS FREQ

Please refer to the note on the usage of this command on page 2-66

Function: To set the designated AF generator's frequency.

Syntax: **AFGEN*n*TS FREQ** | <num> / <num var> / <num expr> | [Hz / kHz]

If the unit is omitted, the default unit is Hz.

Remarks: The *n* represents which generator is to be set, where *n* = 1 or 2. The frequency is set using either a number, a numeric variable or a numeric expression. The unit is optional and is set using Hz or kHz. The frequency range for each generator is 10 Hz to 25 kHz.

This command is equivalent to using the *[Audio GEN]*, *[Gen 1/Gen 2]*, [FREQ], and DATA entry keys.

Example: All of these examples assume that the AF generators are ON. This can be done from MI-BASIC using the following statement, where *n* = 1 or 2:-

```
AFGENnTS STATUS ON
```

Also, the level of the designated generator must be non-zero and the sum of all three generator levels must be less than 5 V.

1. This statement uses a number for the frequency without specifying the optional units and sets AF generator 1 frequency to 100 Hz.

```
AFGEN1TS FREQ 100
```

2. This program segment sets AF generator 2 frequency to 5 kHz by using a numeric variable for the frequency and specifying the unit to be kHz.

```
value = 5
AFGEN2TS FREQ value kHz
```

3. This program segment sets AF generator 2 frequency to 6 kHz by using a numeric expression for the frequency and specifying the unit to be kHz. This command may be part of a loop where the frequency is incremented by 1 kHz each time the loop is passed through.

```
value = 5
AFGEN2TS FREQ value + 1 kHz
```

Note: The Service Monitor has two generators, each of which can be configured as an AF generator or as a modulation generator. The following combinations are possible:-

AFGEN 1 and MODGEN 2
AFGEN 2 and MODGEN 1
AFGEN1 and AFGEN 2
MODGEN1 and MODGEN 2

The act of turning a generator ON will allocate that generator.

AFGEN*n*TS LEVEL

Please refer to the note on the usage of this command on page 2-66

Function: To set the designated AF generator's level.

Syntax: **AFGEN*n*TS LEVEL** | <num> / <num var> / <num expr> | [UVOLT / MVOLT / VOLT]

The default unit is V.

Remarks: The *n* represents which generator is to be set, where *n* = 1 or 2. The level is set using either a number, a numeric variable or a numeric expression. The level range for each generator is 0 to 4.095 V.

This command is equivalent to using the *[Audio GEN]*, *[Gen 1/Gen 2]*, [LEVEL], and DATA entry keys..

Example: All of these examples assume that the AF generators are ON. This can be done from MI-BASIC using the following statement, where *n* = 1 or 2:-

```
AFGENnTS STATUS ON
```

1. This statement uses a number for the level and sets the AF generator 1 level to 1 V.

```
AFGEN1TS LEVEL 1
```

2. This program segment uses a numeric variable for the level and sets the AF generator 2 level to 0.5 V.

```
value = 0.5
AFGEN2TS LEVEL value
```

3. This program segment uses a numeric expression for the level and sets the AF generator 2 level to 3.5 V. The statement that contains the AFGEN command may be part of a loop where the level is incremented by 0.5 V each time the loop is passed through.

```
value = 3
AFGEN2TS LEVEL value + 0.5
```

Note: The Service Monitor has two generators, each of which can be configured as an AF generator or as a modulation generator. The following combinations are possible:-

AFGEN 1 and MODGEN 2
AFGEN 2 and MODGEN 1
AFGEN1 and AFGEN 2
MODGEN1 and MODGEN 2

The act of turning a generator ON will allocate that generator.

AFGEN*n*TS STATUS

Please refer to the note on the usage of this command on page 2-66

Function: To switch the designated AF generator ON or OFF.

Syntax: **AFGEN*n*TS STATUS** | ON | OFF |

Remarks: The *n* represents which generator is to be set, where *n* = 1 or 2. This command is equivalent to using the [ON OFF] key.

Example:

1. This statement switches AF generator 1 ON.

```
AFGEN1TS STATUS ON
```

2. This statement switches AF generator 2 OFF.

```
AFGEN2TS STATUS OFF
```

Note: The Service Monitor has two generators, each of which can be configured as an AF generator or as a modulation generator. The following combinations are possible:-

AFGEN 1 and MODGEN 2
AFGEN 2 and MODGEN 1
AFGEN1 and AFGEN 2
MODGEN1 and MODGEN 2

The act of turning a generator ON will allocate that generator.

AFGEN*n*TS SHAPE

Please refer to the note on the usage of this command on page 2-66

Function: To change the designated AF generator's waveshape.

Syntax: **AFGEN*n*TS SHAPE** | SINE / SQUARE |

Remarks: The *n* represents which generator is to be set, where *n* = 1 or 2. This command is equivalent to using the [ON OFF] key.

Example:

1. This statement sets AF generator 1 to generate a sinewave.

```
AFGEN1TS SHAPE SINE
```

2. This statement sets AF generator 2 to generate a squarewave.

```
AFGEN2TS SHAPE SQUARE
```

Note: The Service Monitor has two generators, each of which can be configured as an AF generator or as a modulation generator. The following combinations are possible:-

AFGEN 1 and MODGEN 2
AFGEN 2 and MODGEN 1
AFGEN1 and AFGEN 2
MODGEN1 and MODGEN 2

The act of turning a generator ON will allocate that generator.

ATOF

Function: To convert a string variable into a numeric variable.

Syntax: **ATOF** (<num var>,<str var>)

Remarks: The **SPRINT** command does the opposite, it converts a numeric variable into a string variable.

Example: This program segment converts a string variable `value$` into a numeric variable `fred` and then prints the result to the screen.
The characters `1234.000000` are printed to the screen.

```
value$ = "1234"
ATOF(fred,value$)
LPRINT fred
```

CLEAR

Function: To clear an area of the screen.

Syntax:

CLEAR(	<num1>	,	<num 2>	,	<num 3>	,	<num 4>	)
	<num var1		<num var2>		<num var3>		<num var4>	
	<num expr1>		<num expr2>		<num expr3>		<num expr4>	

Remarks: The Service Monitor screen size is 400 × 200 pixels, with the origin at the top left corner. The **CLEAR** command clears a rectangular area of the screen. The area to be cleared is defined by coordinates for the top left corner (x1,y1) and for the bottom right corner (x2,y2). Putting the coordinates in the CLEAR statement gives:-

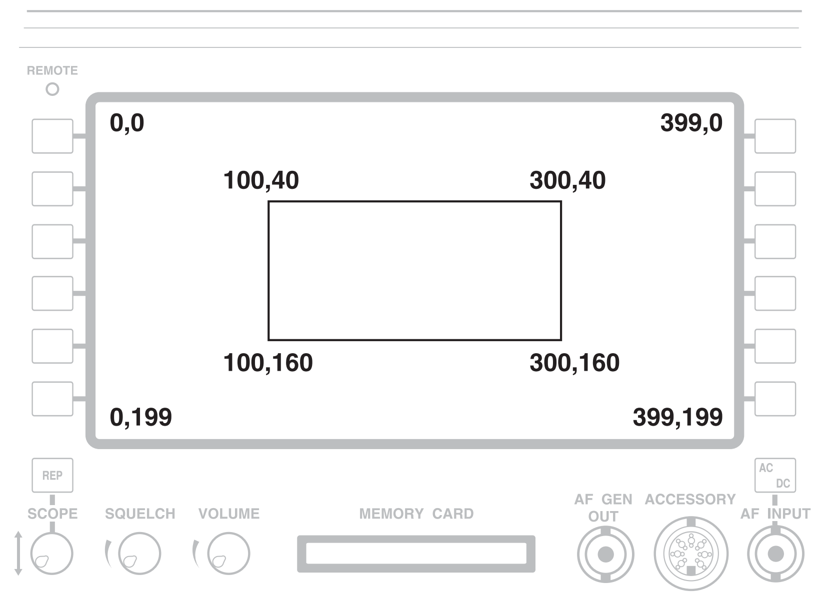
CLEAR(x1,y1,x2,y2)

Example: The following example clears an area of the screen that has corner coordinates as follows:-

Top left corner (100,40)
Top right corner (300,40),
Bottom left corner (100,160)
Bottom right corner (300,160)

This statement uses a combination of numbers, a numeric variable and a numeric expression for the coordinates.

```
coord2 = 40
CLEAR(100,coord2,300,4*coord2)
```



C4194

Fig. 2-2 Co-ordinate reference points on the display

CLRSTORE

Function: To clear the currently displayed results store.

Syntax: **CLRSTORE**

Example: This program segment prints `Hello World!!` into the results store, waits for 3 seconds and then clears the store.

```
LPRINT "Hello World!!"
WAIT 3000
CLRSTORE
```

CLS

Function: To clear the display except for the area allocated to the soft keys.

Syntax: **CLS**

Example: This program segment fills the screen with the * character, waits for 5 seconds and then uses the **CLS** command to clear the middle of the screen.

```
      start_column = -3
      start_row = 0
      column = start_column
      row = start_row
LABEL start
      PRINT AT column,row; "*"
      row = row +1

      IF row > 20 THEN
            column = column + 1
            row = start_row
      ENDIF

      IF column > 46 THEN
            GOTO stop
      ENDIF

      GOTO start

LABEL stop
      WAIT 5000
      CLS
      END
```

DATAFCC

Function: To set-up the Service Monitor to initialise and start Forward Control Channel (FCC) signalling.

Syntax: **DATAFCC**

Remarks: This command may only be required with signalling systems that use a forward control channel, when it should be executed at the beginning of a test.

Example: This program segment is at the beginning of a test, it sets everything up before beginning the test. The **DEFAULT** command clears the pass / fail flags, resets the Service Monitor to its default parameters, then initialises and starts forward control channel signalling. **CLRSTORE** clears the results store.

```
REM start of a test
DEFAULT
DATAFCC
CLRSTORE
```

DEFAULT

Function: To reset the Service Monitor's parameters for the current system variant.

Syntax: **DEFAULT**

Remarks: If a signalling system is in use, the **DEFAULT** command also re-initialises the signalling system and resets the pass and fail count flags.

Example: This program segment is at the beginning of a test; it sets everything up before beginning the test. The **DEFAULT** command clears the pass / fail flags, then initialises and starts forward overhead signalling and **CLRSTORE** clears the results store.

```
REM start of a test
DEFAULT
DATAFCC
CLRSTORE
```

DEMODTYPE

Function: Sets the receiver to expect a particular type of modulation.

Syntax: **DEMODTYPE** | AM
FM |

Remarks:

Example: This statement sets the receiver to demodulate AM signals.

```
DEMODTYPE AM
```

DISPLAY CLEAR

Function: To clear the data display.

Syntax: **DISPLAY CLEAR**

Remarks: This command is not applicable in PMR, it is used to clear the forward / reverse data from the screen (screen titled FORWARD / REVERSE DATA). It is equivalent to pressing the *[clear]* soft key while in that screen.

Example: This program segment is at the beginning of a program. The **DEFAULT** command resets the Service Monitor and clears the pass / fail count; **DISPLAY CLEAR** clears the data display.

```
REM start of a program
DEFAULT
DISPLAY CLEAR
```

DISPLAY PRINTFR

Function: To transfer the expanded forward / reverse data display's data into the results store.

Syntax: **DISPLAY PRINTFR**

Remarks: This command is used during a signalling test such as AMPS or TACS (not PMR) to put the data into the results store.

Example: This program segment executes a **TEST** command; if the test fails, indicated by the fail flag being non zero, the forward / reverse data display's data is transferred into the results store.

Once the display data is in the results store, it can be printed out.

```
TEST PLACECALL
NUMRESULTS error_flag
IF error_flag <> 0 THEN
     DISPLAY PRINTFR
ENDIF
```

DISPLAY PRINTOHD

Function: To transfer the expanded overhead data display's data into the results store.

Syntax: **DISPLAY PRINTOHD**

Remarks: This command is used during a signalling test such as AMPS or TACS (not PMR) to access the data.

Example: This program segment executes a **TEST** command; if the test fails, indicated by the fail flag being non zero, the overhead data display's data is transferred into the results store.

Once the display's data is in the results store, it can be printed out.

```
TEST PLACECALL
NUMRESULTS error_flag
IF error_flag <> 0 THEN
     DISPLAY PRINTOHD
ENDIF
```

END

Function: To terminate the program's execution.

Syntax: **[END]**

Remarks: The **END** command is optional and can be used at any point in a program, it does not necessarily have to be at the end of an input file.

The **END** command can only be used once in a program.

To improve program readability, it is recommended that all subroutines are positioned after the **END** command.

Example: This program has a subroutine to print a message to the screen. The program goes around the loop 10 times; each time it clears the results store and then prints the message in the results store. After 10 loops, the program terminates.

The subroutine is positioned after the **END** command.

```
      count = 0
LABEL start
      count = count + 1
      GOSUB message

      IF count < 10 THEN
            GOTO start
      ENDIF

      END

LABEL message
      CLRSTORE
      LPRINT "Hello"
      LPRINT "Count = ",USING "2.0" count
      RETURN
```

GETDATE

Function: To get the current date from the real time clock.

Syntax: **GETDATE** <str var>

Remarks: If the real time clock option is not fitted, a blank string "   " will be returned.

Example:

```
GETDATE date$
LPRINT "Today is "; date$
```

GETINSTID

Function: To get the instrument serial number.

Syntax: **GETINSTID** <str var>

Remarks:

Example:

```
GETINSTID instid$
LPRINT "Test performed on instrument "; instid$
```

GETKEY

Function: To pause program execution until a key is pressed and optionally return the decimal value of the key pressed.

Syntax: **GETKEY** [<num var>]

Remarks: The program execution is halted until any key is pressed. If <num var> is specified, the decimal value of the pressed key is returned into the <num var>. Tables 2-2 and 2-3, at the end of this chapter, cross-refer the keys to their decimal values. Figures 2-3 and 2-4, also at the end of this chapter, show the key locations with decimal values overwritten.

Example: This program segment draws up 3 soft keys labelled 1, 2 and 3, and prints a request for one of them to be selected. The **GETKEY** command is used with a <num var> called `choice`. The decimal value of the soft key pressed is returned into `choice`, and the value of `choice` determines which message is printed.

If a soft key other than 32, 33 or 34 is pressed, the loop is repeated, and the user is requested by a message printed to the screen to select soft key 1, 2 or 3 again. Once 1, 2 or 3 has been selected, and the relevant message printed to the screen, the program continues.

```
LABEL choose
      SOFTKEY 32,NORMAL "1"
      SOFTKEY 33,NORMAL "2"
      SOFTKEY 34,NORMAL "3"
      LPRINT "Please select 1,2 or 3"
      GETKEY choice

      IF choice = 32 THEN
            LPRINT "Softkey pressed was softkey 1"
      ELSEIF choice = 33 THEN
            LPRINT "Softkey pressed was softkey 2"
      ELSEIF choice = 34 THEN
            LPRINT "Softkey pressed was softkey 3"
      ELSE
            GOTO choose
      ENDIF
```

GETPAUSE

Function: To pause program execution.

Syntax: **GETPAUSE** [| <num> |]
| <num var> |
| <num expr> |

Remarks: There are different pause modes that can be set from either the AUTORUN CONTROL menu, or from remote control. They are:

ALWAYS
MANUAL ONLY (or OFF)
ON_FAILURE

ALWAYS	The program execution is paused every time the **GETPAUSE** command occurs.
MANUAL ONLY	The program execution is only paused if the pause key has been pressed, otherwise the **GETPAUSE** command is ignored.
ON FAILURE	When the **GETPAUSE** command follows a test and the test fails, the program is paused.

If the optional <num> / <num var> / <num expr> is used, it must be set to 1 or 0. If it is set to 1, the program pauses at the **GETPAUSE** command regardless of which pause mode is set.
If the <num> / <num var> / <num expr> = 0, the **GETPAUSE** command is ignored regardless of which pause mode is set.

Example: In the following example, **GETPAUSE** is used by itself and with an optional numeric variable.

Pause mode MANUAL ONLY (OFF). When this pause mode is set, if the pause key is pressed at the beginning of this program segment, the program pauses at the first **GETPAUSE** (after printing `hello...`). If the pause key is not pressed, the program ignores the first **GETPAUSE**. The program stops at the second **GETPAUSE** (after printing `1...`), as the numeric variable is set to 1. The third **GETPAUSE** is ignored, as the numeric variable is set to 0.

Pause mode ALWAYS. When this pause mode is set, the program segment pauses at the first **GETPAUSE** and the second **GETPAUSE**. It ignores the third **GETPAUSE**, as the numeric variable is set to 0.

```
LPRINT "hello..."
GETPAUSE
LPRINT
LPRINT "1..."
fred = 1
GETPAUSE fred
LPRINT "2..."
fred = 0
GETPAUSE fred
LPRINT "3..."
```

GETRXFREQ

Function: To return the current frequency of the RF generator/radio’s receiver.

Syntax: **GETRXFREQ** <num var>

Example: The following program segment reads the current RF receiver frequency.

```
GETRXREQ rx_freq
LPRINT rx_freq
```

GETTIME

Function: To get the current time from the real time clock.

Syntax: **GETTIME** <str var>

Remarks: If the real time clock option is not fitted, a blank string "   " will be returned.

Example:

```
GETTIME time$
LPRINT "Current time is "; time$
```

GETTXFREQ

Function: To return the current instrument receiver frequency/mobile transmitter frequency.

Syntax: **GETTXFREQ** <num var>

Example: The following program segment reads the current RF generater frequency.

```
GETTXFREQ tx_freq
LPRINT tx_freq
```

GOSUB

Function: To jump to a subroutine with the same identifier and return to the next line when execution of the subroutine has been completed.

Syntax: **GOSUB** <name>

Remarks: To improve program readability, it is recommended that the subroutines are placed after the **END** command. The maximum length for <name> is 24 characters. All subroutine <name>s should be unique; <name>s cannot be keywords, numbers, string variables or strings.

Example: This program has a subroutine to print a message on the screen. The program goes around the loop 10 times; each time it clears the results store and then prints the message in the results store. After 10 loops, the program terminates.

The subroutine is positioned after the **END** command.

```
      count = 0
LABEL start
      count = count + 1
      GOSUB message

      IF count < 10 THEN
            GOTO start
      ENDIF

      END

LABEL message
      CLRSTORE
      LPRINT "Hello"
      LPRINT "Count = ",USING "2.0" count
      RETURN
```

GOTO

Function: To jump to a **LABEL** with the same identifier.

Syntax: **GOTO** <name>

Remarks: Maximum length for <name> is 24 characters. All <names> should be unique; <name>s cannot be keywords, strings, string variables or numbers.

Example: This program has a subroutine to print a message on the screen. The program goes around the loop 10 times, using the **GOTO** start statement to go back to the beginning of the loop. Each time around the loop, it clears the results store and then prints the message in the results store. After 10 loops, the program terminates.

The subroutine is positioned after the **END** command.

```
      count = 0
LABEL start
      count = count + 1
      GOSUB message

      IF count < 10 THEN
            GOTO start
      ENDIF

      END

LABEL message
      CLRSTORE
      LPRINT "Hello"
      LPRINT "Count = ",USING "2.0" count
      RETURN
```

IF, THEN, ELSEIF, ELSE, ENDIF

Function: To provide control of program flow.

Syntax:

IF<statement>**THEN** <statement> **ELSEIF** <statement>**THEN** <statement> ... [**ENDIF**]
<prog seg> <prog seg>

ELSE <statement>
<prog seg>

Remarks: An extended **IF** can use multiple lines of statements but has to be terminated with an **ENDIF**. Where an **IF.......THEN** statement is on one line, an **ENDIF** statement must not be used as it will cause a syntax error.

Example: This program asks the user to select a number from 1 to 3 using the data keys.

The program uses an **IF**, **THEN**, **ENDIF** command to check whether the number returned was valid; if it was, the program continues, if not, the program prints a message saying `Number out of range` and then returns to the start.

The decimal values of the numeric DATA keys are from 80 to 89, therefore an offset of 80 is used.

The program uses a **IF**, **THEN**, **ELSEIF**, **ELSE**, **ENDIF** command to determine which message to print on the screen.

```
      offset = 80
LABEL start
      CLRSTORE
      LPRINT "Please select a number from 1 to 3"
      LPRINT "using the data keys."
      GETKEY choice

      IF choice < 81 | choice > 83 THEN
            LPRINT "Number out of range"
            LPRINT "Please try again"
            WAIT 3000
            GOTO start
      ENDIF

      selected = choice - offset

      IF selected = 1 THEN
            GOSUB message1
      ELSEIF selected = 2 THEN
            GOSUB message2
      ELSE
            GOSUB message3
      ENDIF

      END

LABEL message1
      LPRINT "You have selected number 1"
      RETURN
LABEL message2
      LPRINT "You have selected number 2"
      RETURN
LABEL message3
      LPRINT "You have selected number 3"
      RETURN
```

LABEL

Function: To mark a position in a program to jump to from a **GOTO** or a **GOSUB**.

Syntax: **LABEL** <name>

Remarks: Maximum length for <name> is 24 characters; invalid <name>s are numbers, strings, string variables or keywords.

Example: In this program segment, the keyboard buffer is checked and equates a key press to the stored value. If no keys have been pressed, keypress = –1 and the program loops back to the **LABEL** start. If a key has been pressed, the program continues.

```
LABEL start
     LASTKEY keypress
     IF keypress = -1 THEN
      GOTO start
     ENDIF
```

LASTKEY

Function: To get the value of the last key pressed.

Syntax: **LASTKEY** <num var>

Remarks: This is a method of reading the key values without stopping the program. As a key is pressed, its value is stored in a numeric variable which is accessed by this keyword. This means that the value of the last key pressed can be read. Once this is called, the numeric variable is reset to –1 until the next key is pressed. Tables 2-2 and 2-3, at the end of this chapter, cross-refer the keys to their decimal values. Figures 2-3 and 2-4, also at the end of this chapter, show the key locations with decimal values overwritten.

Example: In this program segment, the keyboard buffer is checked and equates a key press to the value in the keyboard buffer. If no keys have been pressed, keypress = –1 and the program loops back to the start. If a key has been pressed, the decimal value of that key is printed to the screen and the program loops back to the start.

```
LABEL start
     CLRSTORE
     LPRINT "Please press a key..."
     WAIT 3000
     LASTKEY keypress
     value = keypress
     WAIT 5000
     IF keypress = -1 THEN
      GOTO start
     ELSE
      LPRINT "Key pressed was " value
     ENDIF
WAIT 3000
GOTO start
```

LET

Function: To assign a numeric expression to a number or numeric variable or numeric expression or to equate a string variable to a string or string variable or string expression.

Syntax: [**LET**] | <num var> = | <num>
<num var>
<num expr> |
| <str var> = | <str>
<str var>
<str expr> |

Remarks: **LET** is an optional keyword.

Example: The following pairs of code produce identical results:-

```
fred = 6
LET fred = 6

fred$ = "hello"
LET fred$ = "hello"
```

LPRINT

Function: To print to the current results store.

Syntax: **LPRINT [<option> , <option> , ... [;]]**

<option> = [**AT** | <num> | <num var> | <num exp>] | <num> | <num var> | <num exp> | <str> | <str var> | **USING**"<num format>" | <num> | <num var> | <num expr> | | **USING**"<str format>" | <str> | <str var> |

<num format> = "[| + | – |] [0] <num1> **.** <num2> "

<str format> = "[–] <num3>"

Remarks: If **AT** is omitted, printing will start at the beginning of the line. If **AT** is used, this specifies at which column to start printing.

The keyword **USING** is a field formatter. All output is right-justified by default; using the optional – sign will cause output to be left-justified.

For numeric format, the optional + causes the output to be signed. Output is normally padded with spaces, the optional 0 will pad the output with zeros. <num1> is the overall field width and <num2> is the number of decimal places.

For string format, <num3> is the field width.

The optional **;** at the end of a line allows subsequent **LPRINT**s to print on the same line. Using just **LPRINT** will print a blank line.

Example: The first **LPRINT** statement prints 00123 to the results store. The following statement prints `-123.46` right-justified in a field width of 10. The next statement prints `hello` right-justified in a field width of 10. The next two statements print `hello` left-justified in a field of 10, followed on the same line by `world`. The last two statements print `hello` and `world` on the same line with and without a space between them.

```
value1 = 123.456
value2 = -123.456
hello$ = "hello"
world$ = "world"
column = 4
LPRINT USING "05.0" value1
LPRINT AT column; USING "+10.2" value2
LPRINT USING "10" hello$
LPRINT USING "-10" hello$;
LPRINT world$
LPRINT hello$, world$
LPRINT hello$ world$
```

The output from the above program segment is:-

```
00123
   –123.46
     hello
hello     world
hello world
helloworld
```

MEASURE

Function: To read back the current value of a measurement at any time during the program.

Syntax: **MEASURE** | AMDEPTH | <num var>
AFFREQ
AFLEVEL
FMDEVN
MODFREQ
RXDISTN
RXSINAD
RXSN
TXDISTN
TXFREQ
TXLEVEL
TXOFFSET
TXSINAD
TXSN

Remarks: <num var> is the numeric variable that the measurement is returned into.
Ensure **RECEIVER AUTOTUNE** status is ON when **MEASURE TX FREQ** is used.

Measurement type	**Units**
AMDEPTH	%
AFFREQ	Hz
AFLEVEL	V
FMDEVN	Hz
MODFREQ	Hz
RXDISTN	%
RXSINAD	dB
RXSN	dB
TXDISTN	%
TXFREQ	Hz
TXLEVEL	W
TXOFFSET	Hz
TXSINAD	dB
TXSN	dB

Example: In the following example, the AM depth is measured. The numeric variable that the AM depth is returned into is called `fred`.

```
MEASURE AMDEPTH fred
```

MODGEN*n* FREQ

Please refer to the note on the usage of this command on page 2-66

Function: To set the designated modulation generator's frequency.

Syntax: **MODGEN*n* FREQ** | <num> / <num var> / <num expr> | [| Hz / kHz |]

If the unit is omitted, the default unit is Hz.

Remarks: The *n* represents which generator is to be set, where *n* = 1 or 2. The frequency is set using either a number, a numeric variable or a numeric expression. The unit is optional and is set using Hz or kHz. The frequency range for each generator is 10 Hz to 25 kHz.

This command is equivalent to using the MOD GEN, FREQ and DATA entry keys

Example: All of these examples assume that the modulation generators are ON. This can be done from MI-BASIC using the following statement, where *n* = 1 or 2:-

```
MODGENn STATUS ON
```

The level of the designated generator must be non-zero

The correct RF port must be selected.

1. This statement uses a number for the frequency without specifying the optional units and sets the frequency of mod gen 1 to 100 Hz

```
MODGEN1 FREQ 100
```

2. This program segment sets the frequency of mod gen 2 to 5 kHz by using a numeric variable for the frequency and specifying the unit to be kHz.

```
value = 5
MODGEN2 FREQ value kHz
```

3. This program segment sets the frequency of mod generator 2 to 6 kHz by using a numeric expression for the frequency and specifying the unit to be kHz. This statement may be part of a loop where the frequency is incremented by 1 kHz each time the loop is passed through.

```
value = 5
MODGEN2 FREQ value + 1 kHz
```

Note: The Service Monitor has two generators, each of which can be configured as an AF generator or as a modulation generator. The following combinations are possible:-

AFGEN 1 and MODGEN 2
AFGEN 2 and MODGEN 1
AFGEN1 and AFGEN 2
MODGEN1 and MODGEN 2

The act of turning a generator ON will allocate that generator.

MODGEN*n* LEVEL

Please refer to the note on the usage of this command on page 2-66

Function: To set the designated modulation generator's level.

Syntax: **MODGEN*n* LEVEL** | <num> <num var> <num expr> | [| Hz kHz % |]

Hz and kHz are used if mod type is FM. If FM is selected, the default unit is Hz.
% is used if mod type is AM.

Remarks: The *n* represents which generator is to be set, where *n* = 1, 2 or 3. The level is set using either a number, a numeric variable or a numeric expression. The level must be non-zero. The correct mod type must be selected; if it is not, the level unit will be set to whatever the current mod type is.

The level range is 0 Hz to 75 kHz for the FM generators and 0 % to 100 % for the AM generators. However, the sum of the levels for each switched on generator must not exceed the above limits,
i.e. If MODGEN1 LEVEL is 60 kHz, MODGEN2 LEVEL must not exceed 15 kHz.

This command is equivalent to using the MOD GEN, LEVEL and DATA entry keys.

Example: All of these examples assume that the mod generators are ON. This can be done from MI-BASIC using the following statement, where *n* = 1 or 2:-

```
MODGENn STATUS ON command.
```

These examples assume that the correct mod type has been selected. This can be done from MI-BASIC using one of the following statements:-

```
MODTYPE FM
MODTYPE AM
```

1. This statement uses a number for the level and sets mod gen 1 level to 100 Hz.

```
MODGEN1 LEVEL 100
```

2. This program segment uses a numeric variable for the level and sets mod gen 2 level to 500 Hz.

```
value = 0.5
MODGEN2 LEVEL value kHz
```

3. This program segment uses a numeric expression for the level and sets mod gen 2 level to 3.5 kHz. The statement that contains the MODGEN command may be part of a loop where the level is incremented by 500 Hz each time the loop is passed through.

```
value = 3
MODGEN2 LEVEL value + 0.5 kHz
```

Note: The Service Monitor has two generators, each of which can be configured as an AF generator or as a modulation generator. The following combinations are possible:-

AFGEN 1 and MODGEN 2
AFGEN 2 and MODGEN 1
AFGEN1 and AFGEN 2
MODGEN1 and MODGEN 2

The act of turning a generator ON will allocate that generator.

MODGEN*n* STATUS

Please refer to the note on the usage of this command on page 2-66

Function: To switch the designated modulation generator ON or OFF.

Syntax: **MODGEN*n* STATUS** | ON / OFF |

Remarks: The *n* represents which generator is to be set, where *n* = 1 or 2. This command is equivalent to using the [ON OFF] key.

Example:

1. This statement switches mod gen 1 ON.

```
MODGEN1 STATUS ON
```

2. This statement switches mod gen 2 OFF.

```
MODGEN2 STATUS OFF
```

Note: The Service Monitor has two generators, each of which can be configured as an AF generator or as a modulation generator. The following combinations are possible:-

AFGEN 1 and MODGEN 2
AFGEN 2 and MODGEN 1
AFGEN1 and AFGEN 2
MODGEN1 and MODGEN 2

The act of turning a generator ON will allocate that generator.

MODGEN*n*TS FREQ

Please refer to the note on the usage of this command on page 2-66

Function: To set the designated modulation generator's frequency.

Syntax:

```
MODGENnTS FREQ  |<num>        | ⎡| Hz   |⎤
                |<num var>    | ⎣| kHz  |⎦
                |<num expr>   |
```

If the unit is omitted, the default unit is Hz.

Remarks: The *n* represents which generator is to be set, where *n* = 1 or 2. The frequency is set using either a number, a numeric variable or a numeric expression. The unit is optional and is set using Hz or kHz. The frequency range for each generator is 10 Hz to 25 kHz.

This command is equivalent to using the MOD GEN, FREQ and DATA entry keys

Example: All of these examples assume that the modulation generators are ON. This can be done from MI-BASIC using the following statement, where *n* = 1 or 2:-

```
MODGENnTS STATUS ON
```

The level of the designated generator must be non-zero

The correct RF port must be selected.

1. This statement uses a number for the frequency without specifying the optional units and sets the frequency of mod gen 1 to 100 Hz

```
MODGEN1TS FREQ 100
```

2. This program segment sets the frequency of mod gen 2 to 5 kHz by using a numeric variable for the frequency and specifying the unit to be kHz.

```
value = 5
MODGEN2TS FREQ value kHz
```

3. This program segment sets the frequency of mod generator 2 to 6 kHz by using a numeric expression for the frequency and specifying the unit to be kHz. This statement may be part of a loop where the frequency is incremented by 1 kHz each time the loop is passed through.

```
value = 5
MODGEN2TS FREQ value + 1 kHz
```

Note: The Service Monitor has two generators, each of which can be configured as an AF generator or as a modulation generator. The following combinations are possible:-

AFGEN 1 and MODGEN 2
AFGEN 2 and MODGEN 1
AFGEN1 and AFGEN 2
MODGEN1 and MODGEN 2

The act of turning a generator ON will allocate that generator.

MODGEN*n*TS LEVEL

Please refer to the note on the usage of this command on page 2-66

Function: To set the designated modulation generator's level.

Syntax: **MODGEN*n*TS LEVEL**

MODGEN*n*TS LEVEL	<num> <num var> <num expr>	[Hz kHz %]

Hz and kHz are used if mod type is FM. If FM is selected, the default unit is Hz.
% is used if mod type is AM.

Remarks: The *n* represents which generator is to be set, where *n* = 1, 2 or 3. The level is set using either a number, a numeric variable or a numeric expression. The level must be non-zero. The correct mod type must be selected; if it is not, the level unit will be set to whatever the current mod type is.

The level range is 0 Hz to 75 kHz for the FM generators and 0 % to 100 % for the AM generators. However, the sum of the levels for each switched on generator must not exceed the above limits,
i.e. If MODGEN1TS LEVEL is 60 kHz, MODGEN2TS LEVEL must not exceed 15 kHz.

This command is equivalent to using the MOD GEN, LEVEL and DATA entry keys.

Example: All of these examples assume that the mod generators are ON. This can be done from MI-BASIC using the following statement, where *n* = 1 or 2:-

```
MODGENnTS STATUS ON command.
```

These examples assume that the correct mod type has been selected. This can be done from MI-BASIC using one of the following statements:-

```
MODTYPE FM
MODTYPE AM
```

1. This statement uses a number for the level and sets mod gen 1 level to 100 Hz.

```
MODGEN1TS LEVEL 100
```

2. This program segment uses a numeric variable for the level and sets mod gen 2 level to 500 Hz.

```
value = 0.5
MODGEN2TS LEVEL value kHz
```

3. This program segment uses a numeric expression for the level and sets mod gen 2 level to 3.5 kHz. The statement that contains the MODGEN command may be part of a loop where the level is incremented by 500 Hz each time the loop is passed through.

```
value = 3
MODGEN2TS LEVEL value + 0.5 kHz
```

Note: The Service Monitor has two generators, each of which can be configured as an AF generator or as a modulation generator. The following combinations are possible:-

AFGEN 1 and MODGEN 2
AFGEN 2 and MODGEN 1
AFGEN1 and AFGEN 2
MODGEN1 and MODGEN 2

The act of turning a generator ON will allocate that generator.

MODGEN*n*TS STATUS

Please refer to the note on the usage of this command on page 2-66

Function: To switch the designated modulation generator ON or OFF.

Syntax: **MODGEN*n*TS STATUS** | ON
OFF |

Remarks: The *n* represents which generator is to be set, where *n* = 1 or 2. This command is equivalent to using the [ON OFF] key.

Example:

1. This statement switches mod gen 1 ON.

```
MODGEN1TS STATUS ON
```

2. This statement switches mod gen 2 OFF.

```
MODGEN2TS STATUS OFF
```

Note: The Service Monitor has two generators, each of which can be configured as an AF generator or as a modulation generator. The following combinations are possible:-

AFGEN 1 and MODGEN 2
AFGEN 2 and MODGEN 1
AFGEN1 and AFGEN 2
MODGEN1 and MODGEN 2

The act of turning a generator ON will allocate that generator.

MODGEN*n*TS SHAPE

Please refer to the note on the usage of this command on page 2-66

Function: To change the designated modulation generator’s waveshape.

Syntax: **MODGEN*n*TS SHAPE** | SINE
SQUARE |

Remarks: The *n* represents which generator is to be set, where *n* = 1 or 2. This command is equivalent to using the [ON OFF] key.

Example:

1. This statement sets mod gen 1 to generate a sinewave.

```
MODGEN1TS SHAPE SINE
```

2. This statement sets mod gen 2 to generate a squarewave.

```
MODGEN2TS SHAPE SQUARE
```

Note: The Service Monitor has two generators, each of which can be configured as an AF generator or as a modulation generator. The following combinations are possible:-

AFGEN 1 and MODGEN 2
AFGEN 2 and MODGEN 1
AFGEN1 and AFGEN 2
MODGEN1 and MODGEN 2

The act of turning a generator ON will allocate that generator.

MODTYPE

Function: To set the type of modulation generator to FM (frequency modulation) or AM (amplitude modulation).

Syntax: **MODTYPE** | AM
FM |

Remarks: Only one type of modulation can be selected at one time, so the **MODTYPE** statement affects the modulation sourced from mod gen 1 and mod gen 2.

Example: This statement sets the mod type to be FM.

```
MODTYPE FM
```

NUMRESULTS

Function: To extract numerical results from the last test run.

Syntax: **NUMRESULTS** <num var1> [,...,<num var8>]

Remarks: There are 8 numeric fields. Numeric values that are returned in these fields are in the same units as those in which they were specified.

(See the **STRRESULTS** command for string results.)

The fields are in the following order:-

1. PASSED
2. ACTUAL
3. TARGET
4. LOWER
5. UPPER
6. AUX1
7. AUX2
8. AUX3

PASSED: The state of the error flag. The error flag is 0 if passed and non-zero if the test failed. See the relevant system Operating Manual Supplement for failure codes.

ACTUAL: The actual reading

TARGET: The target value

LOWER: The lower limit

UPPER: The upper limit

AUX 1, 2 & 3: Auxiliary fields

The fields returned are dependent upon the previous **TEST**.

Example: In the following examples, the <num var>s that represent the fields returned by **NUMRESULTS** are given the same name as the field descriptions above; this does not have to be the case, the names of the fields can be anything as long as they are <num var>s.

1. In the following program segment, the AF frequency is tested. The required frequency is 3.4 kHz and the allowable error is 20%. **NUMRESULTS** is being used to extract the state of the error flag (did the test pass or fail?), the actual frequency (it should be 3.4 kHz ± 20%), and the target frequency, which is 3.4 kHz. These results are then printed to the results store using the **LPRINT** command.

```
TEST AFFREQ REF 3.4,ERROR 20
NUMRESULTS error_flag,actual,target
LPRINT "error flag is " error_flag
LPRINT "actual is     " actual
LPRINT "target is     " target
```

2. In the following program segment, the AF level is tested. The reference level is 2 V, and the error allowed is 10%. NUMRESULTS is used to extract the error flag and the target level; this means that a dummy field has to be used, as the fields that can be returned follow an order and only the first and third fields are required.

```
TEST AFLEVEL REF 2,ERROR 10
NUMRESULTS error_flag,dummy,target
LPRINT "error flag is " error_flag
LPRINT "target is     " target
```

PRINT AT

Function: To print to the screen at specific coordinates.

Syntax:

```
PRINT AT | <num1>     | , | <num 2>    | ; <option>
         | <num var1> |   | <num var2> |

<option> = | <num>                                      |
           | <num var>                                  |
           | <num exp>                                  |
           | <str>                                      |
           | <str var>                                  |
           | USING"<num format>"  | <num>      |        |
           |                      | <num var>  |        |
           |                      | <num expr> |        |
           |                                            |
           | USING"<str format>"  | <str>      |        |
           |                      | <str var>  |        |

<num format> = " [ | + | ] [ 0 ]  <num1> . <num2> "
                   | – |

<str format> = "[ – ]<num3>"
```

Remarks: The coordinates are specified as column, row. There are 58 columns numbered from –6 to 51, and 30 rows, 0 to 29.

The keyword **USING** is used as a field formatter. All output is normally right-justified; using the optional – sign will cause output to be left-justified. For numeric format, the optional + causes the output to be signed. Output is normally padded with spaces, the optional 0 will pad the output with zeros. <num1> is the overall field width and <num2> is the number of decimal places.

For string format, <num3> is the field width.

Example: This program segment uses the **PRINT AT** command to demonstrate that characters can be printed anywhere on the screen.

```
CLS
PRINT AT 10,10; "YOU CA    E SCREEN"
PRINT AT 15,11; "N    H"
PRINT AT 20,12; "T NO TN"
PRINT AT 15,13; "P          I"
PRINT AT 15,14; "RINT A     O"
PRINT AT 20,15; "T ANY P"
```

This program segment uses the **PRINT AT USING** command to demonstrate the field formatting.

```
val1 = 123.456
val2 = 678.90
ten = 10
CLS
PRINT AT ten, 10; USING "4.1" val1
PRINT AT ten, 11; USING "4.2" val1 + val2
```

PRINT AT (continued)

The output from the first program is:-

```
YOU CA    E SCREEN
     N    H
          T NO TN
     P          I
     RINT A     O
          T ANY P
```

The output from the second program is:-

```
123.5
802.36
```

PRINTF AT

Function: To print flashing characters to the screen at specific coordinates.

Syntax: **PRINTF AT** | <num1> / <num var1> | , | <num 2> / <num var2> | ; <option>

<option> = | <num> / <num var> / <num exp> / <str> / <str var> / **USING**"<num format>" | <num> / <num var> / <num expr> | / **USING**"<str format>" | <str> / <str var> | |

<num format> = " [| + / – |] [0] <num1> **.** <num2> "

<str format> = "[–]<num3>"

Remarks: This command prints flashing text, otherwise it is exactly the same as the **PRINT AT** command.

PRINTOFF

Function: To disable printing from tests to the results store.

Syntax: **PRINTOFF**

Remarks: This statement allows the user to customise the tests by formatting the results layout using the **LPRINT** command.

Example: In the following program segment, **PRINTOFF** stops the test printing the results to the results store.

The register status flag is read; if it is 1, **TEST REGISTER** is executed. If **TEST REGISTER** passes, some string results are printed into the results store using the **LPRINT** command. If the test fails, a fail message is printed into the results store using the **LPRINT** command.

In both cases (pass or fail) the format of the results is determined by **LPRINT** commands and not by the test itself.

```
PRINTOFF
handoff_count = 0
fail_count = 0

REM register and output mobile information
READPARAM REGISTER STATUS flag
IF flag = 1 THEN
      TEST REGISTER
      REM if passed output mobile information
      REM else put up error message
      NUMRESULTS error_flag
      IF error_flag = 0 THEN
            STRRESULTS dummy$, min$, dummy$, esn$
            LPRINT min$
            LPRINT
            LPRINT esn$
      ELSE
            LPRINT "**** REGISTRATION FAILED ****"
            fail_count = fail_count + 1
      ENDIF
      GETPAUSE
ENDIF
```

PRINTON

Function: To enable printing from tests to the results store.

Syntax: **PRINTON**

Remarks: This statement enables the default printing of results.

Example: In the following program segment, **PRINTON** re-enables the printing of the results to the results store. The format of the results before the **PRINTON** statement is determined by **LPRINT** commands and not by the test itself.

PRINTON is used to re-enable printing from the test to the results store. **TEST PLACECALL** is executed (the results from this test will be printed to the screen by the test, not the user) and the program continues.

```
PRINTOFF
handoff_count = 0
fail_count = 0

REM register and output mobile information
READPARAM REGISTER STATUS flag
IF flag = 1 THEN
      TEST REGISTER
      REM if passed output mobile information
      REM else put up error message
      NUMRESULTS error_flag
      IF error_flag = 0 THEN
            STRRESULTS dummy$, min$, dummy$, esn$
            LPRINT min$
            LPRINT
            LPRINT esn$
      ELSE
            LPRINT "**** REGISTRATION FAILED ****"
            fail_count = fail_count + 1
            ENDIF
            GETPAUSE
      ENDIF
      LPRINT
      LPRINT "TESTER.............  DATE..........."
      LPRINT
      LPRINT
ENDIF

REM do initial call processing tests
PRINTON
TEST PLACECALL
NUMRESULTS error_flag
```

READPARAM

Function: To read back the setting of a parameter at any time during a program.

Syntax: **READPARAM**

READPARAM	AFFREQ AFLEVEL FMDEVN MODFREQ RFDISTN RFSINAD RFSN RXDISTN RXEXPAND RXSENS RXSINAD RXSN TXCOMPRESS TXDISTN TXFREQ TXLEVEL TXLIMIT TXMODSENS TXNOISE TXSINAD TXSN CONFIGURATION DATAPERFORM DSATDEVN DTMFDECODE HANDOFF HOOKFLASH HSDEVN LANDCLEAR LSDEVN MOBILECLEAR PAGEMOBILE PLACECALL POWERLEVEL PTTOFF PTTON RADIOCALL RADIOCLEAR REGISTER SATDEVN SATFREQ SPOTFREQ STDEVN STDURN STFREQ TESTSETCALL TESTSETCLEAR	<params> <num var>

For the relevant parameters, see Chapter 3. <num var> is the numeric variable that the parameter is returned into. Parameters are returned in %, Watts, Volts, dBVolts, Hz or seconds.

READPARAM (continued)

Parameter read	Status	Numeric value	Parameter read	Status	Numeric value
STATUS	ON	1	RXFILTER	NONE	0
	OFF	0		LP15KHZ	1
ACCPORT	LOGIC0	0		LP300HZ	3
	LOGIC1	1		STDBP	4
	LOGIC2	2		CCITT	5
	LOGIC3	3		CMESS	6
TXFILTER	NONE	0	CHANTYPE	WIDE	0
	LP15KHZ	1		NARROW	1
	LP300HZ	3		ABOVE	2
	STDBP	4		BELOW	3
	CCITT	5		ROTATE	4
	CMESS	6		LAST	5
CALLTYPE	INDIVIDUAL	0			
	GROUP	1			
	EMERGENCY	2			
	ROTATE	3			

Example: In this statement, the status of **TXLIMIT** is read. **STATUS** is the parameter and `test_on` is the <num var> in which **TXLIMIT** status is returned. If the **STATUS** is ON, 1 is returned. If it is OFF, 0 is returned.

```
READPARAM TXLIMIT STATUS test_on
```

RECEIVER AUTOTUNE

Function: To switch the receiver's auto-tune on or off.

Syntax: **RECEIVER AUTOTUNE** | ON | OFF |

Example: This statement switches the receiver's auto-tune off.

```
RECEIVER AUTOTUNE OFF
```

RECEIVER FREQ

Function: To set the frequency to which the receiver is tuned.

Syntax: **RECEIVER FREQ** | <num> | <num var> | <num expr> | [| Hz | kHz | MHz |]

If the unit is omitted, the default unit is Hz.

Example: In the following example, the first line sets the frequency to which the receiver is tuned to the numeric field called `centre_freq`. The second line prints the result.

```
RECEIVER FREQ centre_freq
LPRINT centre_freq
```

REM

Function: To comment a program.

Syntax: **REM** <comments to end of line>

Remarks: This keyword enables commenting of the MI-BASIC program. Anything after the **REM** keyword up to the end of the line is ignored by the program.

Example: The **REM** statement comments the program and is ignored.

```
REM This is a comment.
```

RETURN

Function: To return from a subroutine.

Syntax: **RETURN**

Remarks:

Example: This program has a subroutine to print a message on the screen. The program goes around the loop 10 times, each time it jumps to the **LABEL** message, executes the subroutine, and returns from the subroutine when the **RETURN** keyword is executed. After 10 loops, the program terminates.

The subroutine is positioned after the **END** command.

```
      count = 0
LABEL start
      count = count + 1
      GOSUB message

      IF count < 10 THEN
            GOTO start
      ENDIF

      END

LABEL message
      CLRSTORE
      LPRINT "Hello"
      LPRINT "Count = ",USING "2.0" count
      RETURN
```

RFGEN FREQ

Function: To set the RF generator's frequency.

Syntax: **RFGEN FREQ** | <num> / <num var> / <num expr> | [| Hz / kHz / MHz |]

If the unit is omitted, the default unit is Hz.

Remarks: The frequency is set using either a number, a numeric variable or a numeric expression. The unit is optional and is set using Hz, kHz or MHz. The frequency range for the generator is 400 kHz to 1050 MHz.

This statement is equivalent to using the RF GEN, FREQ and DATA entry keys.

Example: All of these examples assume the RF generator is ON. This can be done from MI-BASIC using the following statement:-

```
RFGEN STATUS ON
```

The level of the generator must be non-zero. The level the generator can be set to will depend on which RF port is selected.

1. This statement uses a number for the frequency without specifying the optional units and sets the RF generator frequency to 400 kHz.

```
RFGEN FREQ 400000
```

2. This program segment sets the RF generator frequency to 500 kHz by using a numeric variable for the frequency and specifying the unit to be kHz.

```
value = 500
RFGEN FREQ value kHz
```

3. This program segment sets the RF generator frequency to 1.5 MHz by using a numeric expression for the frequency and specifying the unit to be MHz. This command may be part of a loop where the frequency is incremented by 1 MHz each time the loop is passed through.

```
value = 0.5
RFGEN FREQ value + 1 MHz
```

RFGEN LEVEL

Function: To set the RF generator's level.

Syntax: **RFGEN LEVEL** | <num> / <num var> / <num expr> | [dBm]

The unit is optional and defaults to dBm.

Remarks: The level is set using either a number, a numeric variable or a numeric expression. The level unit is dBm and optional. The level the generator can be set to will depend on which RF port is selected.

This statement is equivalent to using the RF GEN, LEVEL and DATA entry keys.

Example: All of these examples assume that the RF generator is ON; this can be done from MI-BASIC using the following statement:-

```
RFGEN STATUS ON
```

1. This statement uses a number to set the RF generator level to 5 dBm.

```
RFGEN LEVEL 5 dBm
```

2. This program segment uses a numeric variable to set the RF generator level to –2 dBm.

```
harry = -2
RFGEN LEVEL harry dBm
```

3. This program segment uses a numeric expression to set the RF generator level to –5 dBm.

```
harry = -2
tom = -3
RFGEN LEVEL harry + tom
```

RFGEN STATUS

Function: To switch the RF generator ON or OFF.

Syntax: **RFGEN STATUS** | ON / OFF |

Remarks: This statement is equivalent to using the [ON OFF] key.

Example:

1. This statement switches the RF generator ON.

```
RFGEN STATUS ON
```

2. This statement switches the RF generator OFF.

```
RFGEN STATUS OFF
```

RFPORT

Function: To set up the RF port configuration (BNC / N-type connectors).

Syntax: **RFPORT** | N_DUPLEX
BNC_OUT_N_IN
BNC_OUT_ANT_IN
N_OUT_ANT_IN |

Remarks: This command is equivalent to using the [SELECT] key.

To set up the RF port for simplex operation, set the RF port to a 2-port duplex mode and ignore the other RF port.

Example: This statement configures the RF port to output RF on the BNC socket and input RF on the N-type socket.

```
RFPORT BNC_OUT_N_IN
```

This statement configures the RF port to output and input RF on the N-type socket.

```
RFPORT N_DUPLEX
```

RS232_IN

Function: To read data from the RS232 port.

Syntax: **RS232_IN** <num var>

Remarks: This command reads the ASCII equivalent of the character that is sent. If this value is printed to the screen, the decimal value will be printed.

If the buffer is empty, –1 is returned.

Example: The following program segment receives:-
`hello`, <carriage return>, <line feed>, <line feed>, `world`, <carriage return>, <line feed>.

`h`	=	68_{16} =	ASCII for `h`	= 104_{10}
`e`	=	65_{16} =	ASCII for `e`	= 101_{10}
`l`	=	$6C_{16}$ =	ASCII for `l`	= 108_{10}
`l`	=	$6C_{16}$ =	ASCII for `l`	= 108_{10}
`o`	=	$6F_{16}$ =	ASCII for `o`	= 111_{10}

<carriage return>	=$0D_{16}$	=	ASCII for <carriage return>	= 13_{10}
<line feed>	=$0A_{16}$	=	ASCII for <line feed>	= 10_{10}

`w`	=	77_{16} =	ASCII for `w`	= 119_{10}
`o`	=	$6F_{16}$ =	ASCII for `o`	= 111_{10}
`r`	=	72_{16} =	ASCII for `r`	= 114_{10}
`l`	=	$6C_{16}$ =	ASCII for `l`	= 108_{10}
`d`	=	64_{16} =	ASCII for `d`	= 100_{10}

<carriage return>	=$0D_{16}$	=	ASCII for <carriage return>	= 13_{10}
<line feed>	=$0A_{16}$	=	ASCII for <line feed>	= 10_{10}

```
LABEL start
      RS232_IN fred
      LPRINT fred
      GETPAUSE
      GOTO start
```

The decimal value printed to the screen will be:-

```
104.000000
101.000000
108.000000
108.000000
111.000000
13.000000
10.000000
119.000000
111.000000
114.000000
108.000000
100.000000
13.000000
10.000000
```

RS232_OUT

Function: To send data to the RS232 port.

Syntax: **RS232_OUT** | <num> / <num var> / <num expr> / <str> |

Remarks: Depending on the RS232 port configuration, either 7 bit (0 - 127) or 8 bit (0 - 255) numbers or characters can be sent.

Example: The program segment below will send:-
`A`, <carriage return>, `B`, <carriage return>, `hello world`
to the RS232 port.

```
RS232_OUT 65
fred = 13
RS232_OUT fred
tom = 1
RS232_OUT 65 + tom
RS232_OUT 13
RS232_OUT "hello world"
```

$65_{10} = 41_{16}$ = ASCII for `A`
$13_{10} = D_{16}$ = ASCII for <carriage return>
$66_{10} = 42_{16}$ = ASCII for `B`

RXFILTER

Function: To set the Rx filter.

Syntax: **RXFILTER** | NONE / LP15KHZ / LP300HZ / STDBP / CCITT / CMESS |

Remarks:

LP15KHZ	Low pass 3 dB @ 15 kHz
LP300HZ	Low pass 3 dB @ 300 Hz
STDBP	Standard Band Pass 300 Hz to 3.4 kHz
CCITT	Psophometric Weighting, European
CMESS	Psophometric Weighting, North American

Example: The following statement sets the Rx filter to 300 Hz low pass:-

```
RXFILTER LP300HZ
```

SOFTKEY

Function: To draw and label a soft key at defined positions on the screen.

Syntax: **SOFTKEY** { <num> | <num var> } , { NORMAL | DELETE | CLEAR } [{ <str1> | <str1> <str2> }]

Remarks: The <num> / <num var> is used to define the soft key position. See Fig. 2-3 and Fig. 2-4 on page 2-67 for key numbers and positions.

NORMAL	Normal intensity MI-BASIC soft key
CLEAR	Clears the soft key but not the function
DELETE	Clears the soft key and masks the original function so that another soft key can be drawn in the same position and used for the duration of the program

<str1> / <str1> <str2> provide the optional labelling of the soft key. Where one label is provided, it is centred both vertically and horizontally. Where two labels are provided, they are positioned one above the other and are centred horizontally.

The maximum string length is 7 characters.

Example: The following program segment deletes soft key 3 (*[* ↑ *]* soft key), clears soft key 4 (*[* ↓ *]* soft key), then pauses.

When the program is paused, if soft key 4 is pressed, the soft key re-appears and functions as it should - i.e. changes to the next store - the key has been cleared but the function is not masked. If soft key 3 is pressed, nothing happens - the key has been deleted and its original function masked. When the program that contains this program segment stops, the screen is redrawn, hence the original soft keys are restored.

```
LABEL start
      SOFTKEY 3, DELETE
      WAIT 1000
      SOFTKEY 4, CLEAR
      GETPAUSE
      GOTO start
```

The following program segment uses a numeric variable for the key position and demonstrates the use of the NORMAL keyword.

```
LABEL start
      fred = 3
      SOFTKEY fred,NORMAL "Hello" "There"
      GOTO start
```

SPRINT

Function: To convert a numeric variable into a string variable.

Syntax: **SPRINT** <str var> [<option>] | <num>
<num var>
<num expr> |

<option> = **USING** <num format> | <num>
<num var>
<num expr> |

<num format> = " [+ | –] [0] <num1> **.** <num2> "

Remarks: The **ATOF** command does the opposite, it converts a string variable into a numeric variable.

The keyword **USING** is used as a field formatter. All output is right-justified, using the optional – sign will cause output to be left-justified.

For numeric format, the optional + causes the output to be signed. Output is normally padded with spaces; the optional 0 will pad the output with zeros. <num1> is the overall field width and <num2> is the number of decimal places.

For string format, <num3> is the field width.

Example: The following program segment uses the **SPRINT** keyword with a number, a numeric variable and a numeric expression. It also uses the **USING** keyword to format the fields.

```
value1 = 12.345
value2 = 34.5678
SPRINT first$,98.6543
SPRINT second$ USING "4.2" 98.6543
SPRINT third$,value1
SPRINT fourth$ USING "4.2" value1
SPRINT fifth$ USING "4.2" value1 + value2

LPRINT "first  = " first$
LPRINT "second = " second$
LPRINT "third  = " third$
LPRINT "fourth = " fourth$
LPRINT "fifth  = " fifth$
```

The output from this program segment is:-

```
first  = 98.654300
second = 98.65
third  = 12.345600
fourth = 12.35
fifth  = 46.91
```

STRRESULTS

Function: To read string results from the previous test.

Syntax: **STRRESULTS** <str var1> [,...,<str var8>]

Remarks: All tests produce some infomation which is stored in strings. There are 8 string fields; the first is the title of the test, the rest depend on which test has been executed.

(See the **NUMRESULTS** command for numerical results.)

The fields are in the following order:-

1. TITLE
2. STATUS
3. COMMENT1
4. COMMENT2
5. AUX1
6. AUX2
7. AUX3
8. AUX4

TITLE:	Test title
STATUS:	PASSED or ✶FAIL✶
COMMENT:	Pass or fail summary report
AUX 1, 2, 3 & 4	Additional test information (test-dependent).

Example: In this example, the string variables that represent the fields returned by **STRRESULTS** are given the same name as the field descriptions above; this does not have to be the case, the names of the fields can be anything as long as they are string variables.

In the following program segment, the AF input has been set to 4 kHz. The required frequency is 3.4 kHz and the allowable error is 20%. **STRRESULTS** is used to read the string information in the three fields. These results are then printed to the results store using the **LPRINT** command.

```
TEST AFFREQ REF 3.4,ERROR 20
STRRESULTS title$,status$,comment1$
LPRINT "title is     " title$
LPRINT "status is " status$
LPRINT "comment is " comment$
```

The output from this program is:-

```
AF FREQUENCY         PASSED 4.000 kHz +17.6%
title is     AF FREQUENCY
status 1 is PASSED
comment 1 is 4.000kHz +17.6%
```

TEST

Function: To execute a pre-defined test

Syntax: **TEST**

```
TEST  AFFREQ          [ <params> ]
      AFLEVEL
      FMDEVN
      MODFREQ
      RFDISTN
      RFSINAD
      RFSN
      RXDISTN
      RXEXPAND
      RXSENS
      RXSINAD
      RXSN
      TXCOMPRESS
      TXDISTN
      TXFREQ
      TXLEVEL
      TXLIMIT
      TXMODSENS
      TXNOISE
      TXSINAD
      TXSN

      DATAPERFORM
      DSATDEVN
      DTMFDECODE
      HANDOFF
      HOOKFLASH
      HSDEVN
      LANDCLEAR
      LSDEVN
      MOBILECLEAR
      PAGEMOBILE
      PLACECALL
      POWERLEVEL
      PTTOFF
      PTTON
      RADIOCALL
      RADIOCLEAR
      REGISTER
      SATDEVN
      SATFREQ
      STDEVN
      STDURN
      STFREQ
      TESTSETCALL
      TESTSETCLEAR
```

Remarks: The parameters for each test will depend on which system is used.
For the relevant parameters, refer to Chapter 3.

The order that the parameters are entered in is not important. Where parameters are not specified, the default parameters are used. The default parameters can be changed using the **WRITEPARAM** statement.

TEST HANDOFF. This test will perform a handoff to the channel specified in the "VCHAN" parameter.

TEST POWERLEVEL. This test will set the power level to the level specified in the "VMAC" parameter on AMPS or TACS systems, or the "TRAFFIC POWER" parameter on NMT systems.

Example: In this example, the **TEST RXSENS** test is executed. The modulation level, Rx filter and upper limit are specified. The other parameters for this test, i.e. status, averages and reference SINAD, are not specified, so the default values for these parameters are used.

```
TEST RXSENS MODLEVEL 5KHZ, RXFILTER NONE, UPPER -116dBm
```

In the following examples, the transmit power level of an AMPS or TACS mobile would be set to power level 5, and the transmit power level of an NMT mobile would be set to power level 3.

```
WRITEPARAM "VMAC" 5
TEST POWERLEVEL

WRITEPARAM "TRAFFIC_POWER" 3
TEST POWERLEVEL
```

In the following example, a handoff will be performed to channel 100:-

```
WRITEPARAM "VCHAN" 100
TEST HANDOFF
```

TXFILTER

Function: To set the Tx filter.

Syntax: **TXFILTER** | NONE
LP15KHZ
LP300HZ
STDBP
CCITT
CMESS |

Remarks:

LP15KHZ	Low pass 3 dB @ 15 kHz
LP300HZ	Low pass 3 dB @ 300 Hz
STDBP	Standard Band Pass 300 Hz to 3.4 kHz
CCITT	Psophometric Weighting, European
CMESS	Psophometric Weighting, North American.

Example: The following statement sets the Tx filter to 15 kHz low pass:-

```
TXFILTER LP15KHZ
```

WAIT

Function: To wait the specified number of milliseconds before continuing the program.

Syntax: **WAIT** | <num>
<num var>
<num expr> |

Remarks: The number of milliseconds to wait can be entered as a number, a numeric variable or a numeric expression.

Example: In the following program segment, a number, a numeric variable and a numeric expression are used to specify the number of milliseconds to wait.

```
LABEL start
      CLRSTORE
      LPRINT "Waiting 1 second"
      WAIT 1000
      LPRINT "Waiting 2 seconds"
      delay = 2000
      WAIT delay
      LPRINT "Waiting 3 seconds"
      onesec = 1000
      WAIT delay + onesec
      GOTO start
```

WRITEPARAM

Function: To set a parameter at any time during the program.

Syntax: **WRITEPARAM** AFFREQ | AFLEVEL | FMDEVN | MODFREQ | RFDISTN | RFSINAD | RFSN | RXDISTN | RXEXPAND | RXSENS | RXSINAD | RXSN | TXCOMPRESS | TXDISTN | TXFREQ | TXLEVEL | TXLIMIT | TXMODSENS | TXNOISE | TXSINAD | TXSN | CONFIGURATION | DATAPERFORM | DSATDEVN | DTMFDECODE | HANDOFF | HOOKFLASH | HSDEVN | LANDCLEAR | LSDEVN | MOBILECLEAR | PAGEMOBILE | PLACECALL | POWERLEVEL | PTTOFF | PTTON | RADIOCALL | RADIOCLEAR | REGISTER | SATDEVN | SATFREQ | SPOTFREQ | STDEVN | STDURN | STFREQ | TESTSETCALL | TESTSETCLEAR <params>

WRITEPARAM	AFFREQ AFLEVEL FMDEVN MODFREQ RFDISTN RFSINAD RFSN RXDISTN RXEXPAND RXSENS RXSINAD RXSN TXCOMPRESS TXDISTN TXFREQ TXLEVEL TXLIMIT TXMODSENS TXNOISE TXSINAD TXSN CONFIGURATION DATAPERFORM DSATDEVN DTMFDECODE HANDOFF HOOKFLASH HSDEVN LANDCLEAR LSDEVN MOBILECLEAR PAGEMOBILE PLACECALL POWERLEVEL PTTOFF PTTON RADIOCALL RADIOCLEAR REGISTER SATDEVN SATFREQ SPOTFREQ STDEVN STDURN STFREQ TESTSETCALL TESTSETCLEAR	<params>

Remarks: For the relevant parameters, see 'Test parameters' in Chapter 3.

Example: In this statement, the AF frequency reference is set to 2 kHz.

```
WRITEPARAM AFFREQ REF 2 kHz
```

Instrument key decimal values

For use with **GETKEY** and **LASTKEY** commands.

Table 2-2 Key descriptions vs decimal values

Data keys

0	80
1	81
2	82
3	83
4	84
5	85
6	86
7	87
8	88
9	89
Decimal point	90
Minus	91
Delete	92
MHz, V	96
kHz, mV	97
Hz, µV	98
dB, %	99
dBm	100

Function keys

MEM	57
DISPLAY HOLD	56

Input - output select keys

RF IN/OUT SELECT	114

Mode keys

HELP SET-UP	53
Tx TEST	48
Rx TEST	49
Dx TEST	50
SYSTEM	54
SPEC ANA	51
AF TEST	52

Soft keys, left , 32 to 37

Left, top	32
Left, bottom	37

Soft keys, right, 0 to 5

Right, top	0
Right, bottom	5

Table 2-3 Decimal values vs key descriptions

0 - 5, Right soft keys

0	Top right
5	Bottom right

32 - 37, Left soft keys

32	Top left
37	Bottom left

48 - 54, Mode keys

48	Tx TEST
49	Rx TEST
50	Dx TEST
51	SPEC ANA
52	AF TEST
53	HELP SET-UP
54	SYSTEM

56 - 57, Function keys

56	DISPLAY HOLD
57	MEM

80 ...... 114, Data keys

80	0
81	1
82	2
83	3
84	4
85	5
86	6
87	7
88	8
89	9
90	Decimal point
91	Minus
92	Delete
96	Data entry MHz, V
97	Data entry kHz, mV
98	Data entry Hz, µV
99	Data entry dB, %
100	Data entry dBm
114	RF IN/OUT routeing

AFGENn/TS and MODGENn/TS commands in AF and modulation generators

In B-series 2945 Communication Service Monitors (such as 2945B and 2948B), and in A-series (such as 2945A and 2948) with Cellular software 44533-440 V4.06 or later, the original MI-BASIC AFGENn and MODGENn commands could cause problems whereby, for example, the generator settings shown on the instrument's Tx, Rx, Dx and AF Test screens did not match the settings called for by the programmer. This is caused by the generators controlled by these commands being part of the Cellular Board, while the generators shown on the displays are part of the main instrument (and hence, totally different).

Such problems are overcome through a new set of AF and Modulation Generator commands, AFGENnTS and MODGENnTS, found in the 2945B Series (and 2945A Series fitted with Cellular Software 44533-440 V4.06 or later).

The old AFGENn and MODGENn commands have been retained for backwards compatibility, but it is recommended that new programs are implemented with AFGENnTS and MODGENnTS. Additionally, it is recommended not to mix old and new commands within the same program.

The relevant commands are listed below. Full details start on page 2-17 (AFGENnTS) and page 2-40 (MODGENnTS).

AF Generator commands	
Legacy	**Recommended**
AFGENn FREQ	AFGENnTS FREQ
AFGENn LEVEL	AFGENnTS LEVEL
AFGENn STATUS	AFGENnTS LEVEL
	AFGENnTS SHAPE

Modulation Generator commands	
Legacy	**Recommended**
MODGENn FREQ	MODGENnTS FREQ
MODGENn LEVEL	MODGENnTS FREQ
MODGENn STATUS	MODGENnTS STATUS
	MODGENnTS SHAPE

Notes:

There are two AF/Modulation Generators available. Selection is made by setting n to 1 or 2, e.g. AFGEN2 STATUS ON.

The syntax of the new commands is the same as that for the corresponding old ones.

There is an additional new command for selecting the waveshape SINE or SQUARE.

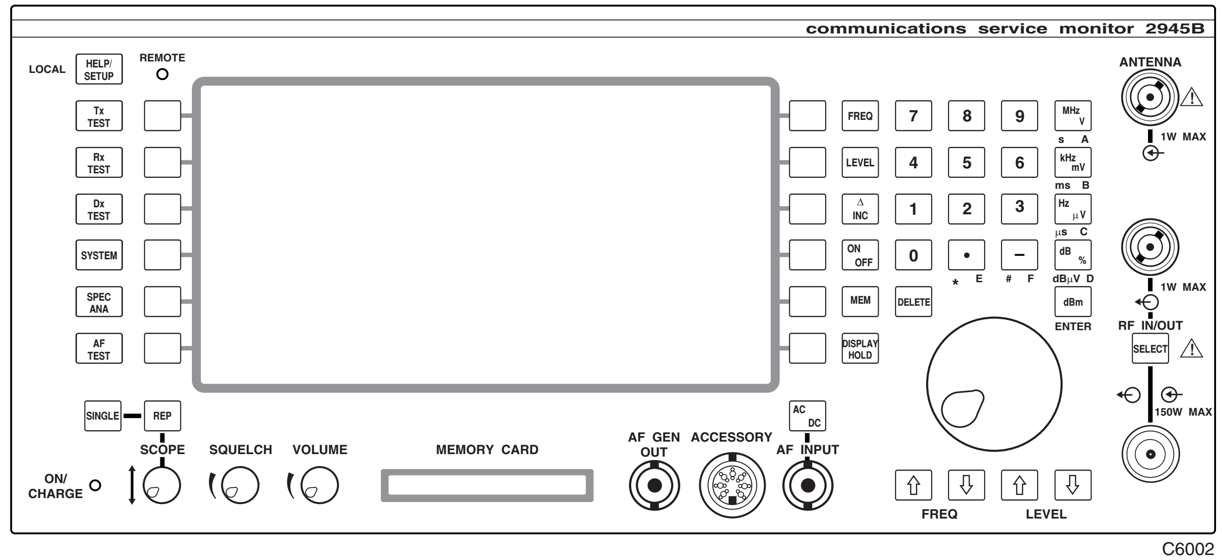


Fig. 2-3 Front panel layout

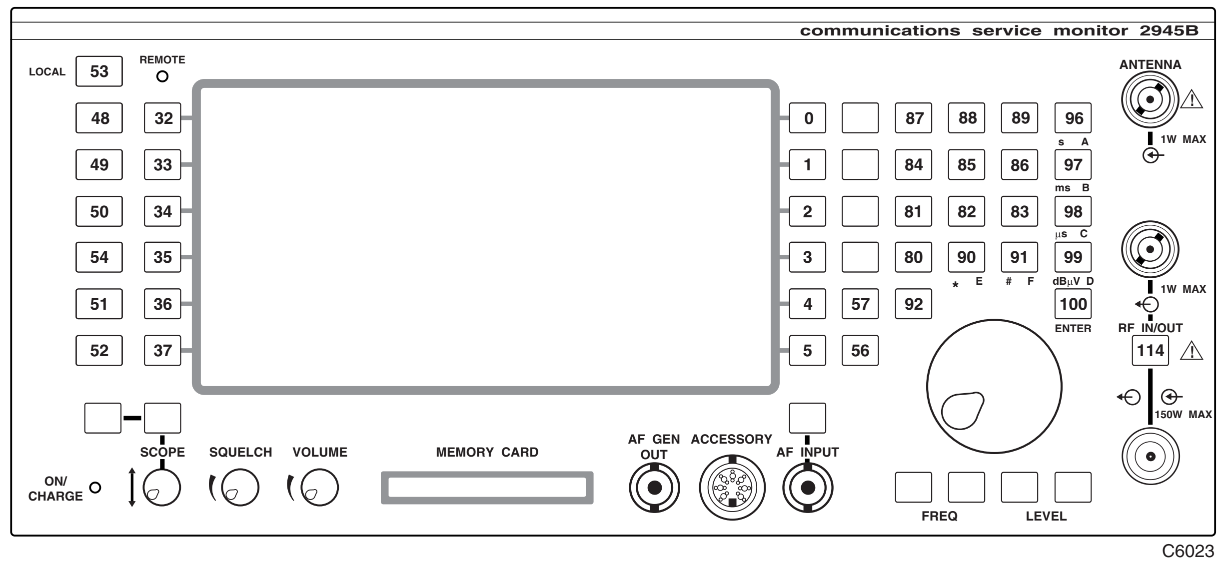


Fig. 2-4 Key code number definition

Chapter 3
TEST PARAMETERS

This Chapter lists and gives more details about the **TEST** commands. These details also apply to the **READPARAM** and **WRITEPARAM** commands if these commands are substituted for **TEST** at every occurrence.

The **TEST/READPARAM/WRITEPARAM** commands are described in Chapter 2 of this manual.

Tables 3-1 and 3-2 serve as a contents page for this chapter and show to which system or systems each test is applicable. The tables show the pre-defined tests that are available within MI-BASIC for inclusion in test programs. The specific system columns show which tests are included in that system's test programs.

The test parameters, with their default settings, are listed in the operating manual supplement for the appropriate system. (See *Autorun parameter settings* in Chapter 1 of the relevant supplement.)

Table 3-1 Parametric tests

COMMANDS	PMR	AMPS	TACS	NMT	MPT 1327	EDACS Radio	EDACS Repeater	LTR Radio	LTR Repeater	See page
AFFREQ										3-3
AFLEVEL										3-3
FMDEVN										3-3
MODFREQ										3-3
RFDISTN							✓		✓	3-4
RFSINAD							✓		✓	3-5
RFSN							✓		✓	3-6
RXDISTN	✓	✓	✓	✓	✓	✓	✓	✓	✓	3-8
RXEXPAND	✓	✓	✓	✓	✓	✓	✓	✓	✓	3-8
RXSENS	✓	✓	✓	✓	✓	✓	✓	✓	✓	3-9
RXSINAD	✓	✓	✓	✓	✓	✓	✓	✓	✓	3-10
RXSN	✓	✓	✓	✓	✓	✓	✓	✓	✓	3-11
TXCOMPRESS	✓	✓	✓	✓	✓	✓	✓	✓	✓	3-12
TXDISTN	✓	✓	✓	✓	✓	✓	✓	✓	✓	3-13
TXFREQ	✓	✓	✓	✓	✓	✓	✓	✓	✓	3-14
TXLEVEL	✓				✓	✓	✓	✓	✓	3-14
TXLIMIT	✓	✓	✓	✓	✓	✓	✓	✓	✓	3-15
TXMODSENS	✓									3-16
TXNOISE	✓	✓	✓	✓	✓	✓	✓	✓	✓	3-16
TXSINAD	✓	✓	✓	✓	✓	✓	✓	✓	✓	3-17
TXSN	✓	✓	✓	✓	✓	✓	✓	✓	✓	3-18

Table 3-2 Cellular and trunking tests

COMMANDS	PMR	AMPS	TACS	NMT	MPT 1327	EDACS Radio	EDACS Repeater	LTR Radio	LTR Repeater	See page
CONFIGURATION	✓									3-19
DATADEVN								✓	✓	3-19
DATAPERFORM		✓	✓			✓	✓			3-19
DSATDEVN		✓	✓							3-20
DTMFDECODE	✓	✓	✓			✓		✓		3-20
HANDOFF		✓	✓	✓	✓					3-21
HOOKFLASH		✓	✓							3-22
HSDEVN							✓			3-22
LANDCLEAR		✓	✓	✓	✓	✓				3-23
LISTENOFF								✓		3-23
LISTENON								✓		3-23
LSDEVN						✓	✓			3-23
MOBILECLEAR		✓	✓	✓	✓	✓				3-24
PAGEMOBILE		✓	✓	✓	✓	✓				3-25
PLACECALL		✓	✓	✓	✓	✓				3-26
POWERLEVEL		✓	✓	✓						3-27
PTTOFF	✓				✓	✓		✓		3-28
PTTON	✓				✓	✓		✓		3-28
RADIOCALL							✓			3-29
RADIOCLEAR							✓			3-29
RADIOPTTOFF									✓	3-30
RADIOPTTON									✓	3-30
REGISTER		✓	✓	✓	✓					3-31
SATDEVN		✓	✓	✓						3-32
SATFREQ		✓	✓	✓						3-32
SPOTFREQ	✓									3-32
STDEVN		✓	✓							3-33
STDURN		✓	✓							3-33
STFREQ		✓	✓							3-33
TESTSETCALL							✓			3-34
TESTSETCLEAR							✓			3-34
TESTSETPTTOFF									✓	3-35
TESTSETPTTON									✓	3-35

Parametric test commands

TEST AFFREQ

Function : AF frequency reference

Syntax : **TEST AFFREQ REF** | <num> / <num var> / <num expr> | [| Hz / kHz / MHz |]

Function : AF frequency error

Syntax : **TEST AFFREQ ERROR** | <num> / <num var> / <num expr> | %

TEST AFLEVEL

Function : AF level reference

Syntax : **TEST AFLEVEL REF** | <num> / <num var> / <num expr> | [| VOLT / MVOLT |]

Function : AF level error

Syntax : **TEST AFLEVEL ERROR** | <num> / <num var> / <num expr> | %

TEST FMDEVN

Function : FM deviation reference

Syntax : **TEST FMDEVN REF** | <num> / <num var> / <num expr> | [| Hz / kHz / MHz |]

Function : FM deviation error

Syntax : **TEST FMDEVN ERROR** | <num> / <num var> / <num expr> | %

TEST MODFREQ

Function : Modulation frequency reference

Syntax : **TEST MODFREQ REF** | <num> / <num var> / <num expr> | [| Hz / kHz / MHz |]

Function : Modulation frequency error

Syntax : **TEST MODFREQ ERROR** | <num> / <num var> / <num expr> | %

TEST RFDISTN

Function : RF distortion status

Syntax : **TEST RFDISTN STATUS** | ON / OFF |

TEST RFDISTN STATUS	ON OFF

Function : RF distortion averages

Syntax : **TEST RFDISTN AVERAGES**	<num> <num var> <num expr>

Function : RF distortion modulation level

Syntax : **TEST RFDISTN MODLEVEL**	<num> <num var> <num expr>	[Hz kHz MHz]

Function : RF distortion Tx filter

Syntax : **TEST RFDISTN TXFILTER**	NONE LP15KHZ LP300HZ STDBP CCITT CMESS

Function : RF distortion upper limit

Syntax : **TEST RFDISTN UPPER**	<num> <num var> <num expr>	%

TEST RFSINAD

Function : RF SINAD status

Syntax : **TEST RFSINAD STATUS** | ON
OFF |

Function : RF SINAD averages

Syntax : **TEST RFSINAD AVERAGES** | <num>
<num var>
<num expr> |

Function : RF SINAD modulation level

Syntax : **TEST RF SINAD MODLEVEL** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

Function : RF SINAD Tx filter

Syntax : **TEST RF SINAD TXFILTER** | NONE
LP15KHZ
LP300HZ
STDBP
CCITT
CMESS |

Function : RF SINAD lower limit

Syntax : **TEST RFSINAD LOWER** | <num>
<num var>
<num expr> | DB

TEST RFSN

Function : RF Signal-to-Noise ratio status

Syntax : **TEST RFSN STATUS** | ON
OFF |

Function : RF Signal-to-Noise ratio averages

Syntax : **TEST RFSN AVERAGES** | <num>
<num var>
<num expr> |

Function : RF Signal-to-Noise ratio modulation level

Syntax : **TEST RFSN MODLEVEL** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

Function : RF Signal-to-Noise ratio Tx filter

Syntax : **TEST RFSN TXFILTER** | NONE
LP15KHZ
LP300HZ
STDBP
CCITT
CMESS |

Function : RF Signal-to-Noise ratio lower limit

Syntax : **TEST RFSN LOWER** | <num>
<num var>
<num expr> | DB

TEST RXDISTN

Function : Rx distortion status

Syntax : **TEST RXDISTN STATUS** | ON
OFF |

Function : Rx distortion averages

Syntax : **TEST RXDISTN AVERAGES** | <num>
<num var>
<num expr> |

Function : Rx distortion RF generator level

Syntax : **TEST RXDISTN RFGENLEVEL** | <num>
<num var>
<num expr> | DBM

Function : Rx distortion modulation level

Syntax : **TEST RXDISTN MODLEVEL** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

Function : Rx distortion Rx filter

Syntax : **TEST RXDISTN RXFILTER** | NONE
LP15KHZ
LP300HZ
STDBP
CCITT
CMESS |

Function : Rx distortion upper limit

Syntax : **TEST RXDISTN UPPER** | <num>
<num var>
<num expr> | %

Function : Rx distortion NB modulation level [NAMPS and NTACS only]

Syntax : **TEST RXDISTN NBMODLEVEL** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

TEST RXEXPAND

Function : Rx Expansion status

Syntax : **TEST RXEXPAND STATUS** | ON / OFF |

Function : Rx Expansion averages

Syntax : **TEST RXEXPAND AVERAGES** | <num> / <num var> / <num expr> |

Function : Rx Expansion modulation level

Syntax : **TEST RXEXPAND MODLEVEL** | <num> / <num var> / <num expr> | [| Hz / kHz / MHz |]

Function : Rx Expansion Rx filter

Syntax : **TEST RXEXPAND RXFILTER** | NONE / LP15KHZ / LP300HZ / STDBP / CCITT / CMESS |

Function : Rx Expansion reference

Syntax : **TEST RXEXPAND REF** | <num> / <num var> / <num expr> |

Function : Rx Expansion error

Syntax : **TEST RXEXPAND ERROR** | <num> / <num var> / <num expr> | %

Function : Rx Expansion NB modulation level [NAMPS only]

Syntax : **TEST RXEXPAND NBMODLEVEL** | <num> / <num var> / <num expr> | [| Hz / kHz / MHz |]

Function : Rx Expansion modulation swing [PMR only]

Syntax : **TEST RXEXPAND SWING** | <num> / <num var> / <num expr> | DB

TEST RXSENS

Function : Rx sensitivity status

Syntax : **TEST RXSENS STATUS** | ON
OFF |

Function : Rx sensitivity averages

Syntax : **TEST RXSENS AVERAGES** | <num>
<num var>
<num expr> |

Function : Rx sensitivity modulation level

Syntax : **TEST RXSENS MODLEVEL** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

Function : Rx sensitivity Rx filter

Syntax : **TEST RXSENS RXFILTER** | NONE
LP15KHZ
LP300HZ
STDBP
CCITT
CMESS |

Function : Rx sensitivity RF upper limit

Syntax : **TEST RXSENS UPPER** | <num>
<num var>
<num expr> | DBM

Function : Rx sensitivity reference SINAD

Syntax : **TEST RXSENS REFSINAD** | <num>
<num var>
<num expr> | DB

Function : Rx sensitivity NB modulation level [NAMPS only]

Syntax : **TEST RXSENS NBMODLEVEL** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

TEST RXSINAD

Function : Rx SINAD status

Syntax : **TEST RXSINAD STATUS** | ON / OFF |

Function : Rx SINAD averages

Syntax : **TEST RXSINAD AVERAGES** | <num> / <num var> / <num expr> |

Function : Rx SINAD RF level

Syntax : **TEST RXSINAD RFGENLEVEL** | <num> / <num var> / <num expr> | DBM

Function : Rx SINAD modulation level

Syntax : **TEST RXSINAD MODLEVEL** | <num> / <num var> / <num expr> | [| Hz / kHz / MHz |]

Function : Rx SINAD Rx filter

Syntax : **TEST RXSINAD RXFILTER** | NONE / LP15KHZ / LP300HZ / STDBP / CCITT / CMESS |

Function : Rx SINAD lower limit

Syntax : **TEST RXSINAD LOWER** | <num> / <num var> / <num expr> | DB

Function : Rx SINAD NB modulation level [NAMPS and NTACS only]

Syntax : **TEST RXSINAD NBMODLEVEL** | <num> / <num var> / <num expr> | [| Hz / kHz / MHz |]

TEST RXSN

Function : Rx Signal-to-Noise ratio status

Syntax : **TEST RXSN STATUS** | ON / OFF |

Function : Rx Signal-to-Noise ratio averages

Syntax : **TEST RXSN AVERAGES** | <num> / <num var> / <num expr> |

Function : Rx Signal-to-Noise ratio RF level

Syntax : **TEST RXSN RFGENLEVEL** | <num> / <num var> / <num expr> | DBM

Function : Rx Signal-to-Noise ratio modulation level

Syntax : **TEST RXSN MODLEVEL** | <num> / <num var> / <num expr> | [| Hz / kHz / MHz |]

Function : Rx Signal-to-Noise ratio Rx filter

Syntax : **TEST RXSN RXFILTER** | NONE / LP15KHZ / LP300HZ / STDBP / CCITT / CMESS |

Function : Rx Signal-to-Noise ratio lower limit

Syntax : **TEST RXSN LOWER** | <num> / <num var> / <num expr> | DB

Function : Rx Signal-to-Noise ratio NB modulation level [NAMPS and NTACS only]

Syntax : **TEST RXSN NBMODLEVEL** | <num> / <num var> / <num expr> | [| Hz / kHz / MHz |]

TEST TXCOMPRESS

Function : Tx Compression status

Syntax :	**TEST TXCOMPRESS STATUS**	ON OFF

Function : Tx Compression averages

Syntax :	**TEST TXCOMPRESS AVERAGES**	<num> <num var> <num expr>

Function : Tx Compression modulation level

Syntax :	**TEST TXCOMPRESS MODLEVEL**	<num> <num var> <num expr>	[Hz kHz MHz]

Function : Tx Compression Tx filter

Syntax :	**TEST TXCOMPRESS TXFILTER**	NONE LP15KHZ LP300HZ STDBP CCITT CMESS

Function : Tx Compression reference

Syntax :	**TEST TXCOMPRESS REF**	<num> <num var> <num expr>

Function : Tx Compression error

Syntax :	**TEST TXCOMPRESS ERROR**	<num> <num var> <num expr>	%

Function : Tx Compression NB modulation level [NAMPS and NTACS only]

Syntax :	**TEST TXCOMPRESS NBMODLEVEL**	<num> <num var> <num expr>	[Hz kHz MHz]

Function : Tx Compression modulation swing. [PMR only]

Syntax :	**TEST TXCOMPRESS SWING**	<num> <num var> <num expr>	DB

TEST TXDISTN

Function : Tx Distortion status

Syntax : **TEST TXDISTN STATUS**
- ON
- OFF

Function : Tx Distortion averages

Syntax : **TEST TXDISTN AVERAGES**
- <num>
- <num var>
- <num expr>

Function : Tx Distortion modulation level

Syntax : **TEST TXDISTN MODLEVEL**
- <num>
- <num var>
- <num expr>

[
- Hz
- kHz
- MHz

]

Function : Tx Distortion Tx filter

Syntax : **TEST TXDISTN TXFILTER**
- NONE
- LP15KHZ
- LP300HZ
- STDBP
- CCITT
- CMESS

Function : Tx Distortion upper limit

Syntax : **TEST TXDISTN UPPER**
- <num>
- <num var>
- <num expr>

%

Function : Tx Distortion NB modulation level [NAMPS and NTACS only]

Syntax : **TEST TXDISTN NBMODLEVEL**
- <num>
- <num var>
- <num expr>

[
- Hz
- kHz
- MHz

]

TEST TXFREQ

Function : Tx Frequency status

Syntax : **TEST TXFREQ STATUS**
ON
OFF

Function : Tx Frequency error tolerance

Syntax : **TEST TXFREQ ERROR**
<num>
<num var>
<num expr>
[Hz
kHz
MHz]

Function : Tx Frequency reference [PMR only]

Syntax : **TEST TXFREQ REF**
<num>
<num var>
<num expr>
[Hz
kHz
MHz]

Function : Tx Frequency NB error tolerance [NAMPS and NTACS only]

Syntax : **TEST TXFREQ NBERROR**
<num>
<num var>
<num expr>
[Hz
kHz
MHz]

TEST TXLEVEL

Function : Tx Level status

Syntax : **TEST TXLEVEL STATUS**
ON
OFF

Function : Tx Level averages

Syntax : **TEST TXLEVEL AVERAGES**
<num>
<num var>
<num expr>

Function : Tx Level upper limit

Syntax : **TEST TXLEVEL UPPER**
<num>
<num var>
<num expr>
[WATT
MWATT]

Function : Tx Level lower limit

Syntax : **TEST TXLEVEL LOWER**
<num>
<num var>
<num expr>
[WATT
MWATT]

TEST TXLIMIT

Function : Tx Limiting status

Syntax : **TEST TXLIMIT STATUS** | ON
OFF |

Function : Tx Limiting averages

Syntax : **TEST TXLIMIT AVERAGES** | <num>
<num var>
<num expr> |

Function : Tx Limiting modulation level

Syntax : **TEST TXLIMIT MODLEVEL** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

Function : Tx Limiting Tx filter

Syntax : **TEST TXLIMIT TXFILTER** | NONE
LP15KHZ
LP300HZ
STDBP
CCITT
CMESS |

Function : Tx Limiting overload factor

Syntax : **TEST TXLIMIT OVERLOAD** | <num>
<num var>
<num expr> | DB

Function : Tx Limiting upper limit

Syntax : **TEST TXLIMIT UPPER** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

Function : Tx Limiting NB modulation level [NAMPS and NTACS only]

Syntax : **TEST TXLIMIT NBMODLEVEL** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

Function : Tx Limiting NB upper limit [NAMPS and NTACS only]

Syntax : **TEST TXLIMIT NBUPPER** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

TEST TXMODSENS

Function : Tx Modulation Sensitivity status

Syntax : **TEST TXMODSENS STATUS** | ON
OFF |

Function : Tx Modulation Sensitivity modulation level

Syntax : **TEST TXMODSENS MODLEVEL** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

Function : Tx Modulation Sensitivity Tx filter

Syntax : **TEST TXMODSENS TXFILTER** | NONE
LP15KHZ
LP300HZ
STDBP
CCITT
CMESS |

Function : Tx Modulation Sensitivity upper limit

Syntax : **TEST TXMODSENS UPPER** | <num>
<num var>
<num expr> | [| UVOLT
MVOLT
VOLT |]

Function : Tx Modulation Sensitivity lower limit

Syntax : **TEST TXMODSENS LOWER** | <num>
<num var>
<num expr> | [| UVOLT
MVOLT
VOLT |]

TEST TXNOISE

Function : Tx Noise status

Syntax : **TEST TXNOISE STATUS** | ON
OFF |

Function : Tx Noise averages

Syntax : **TEST TXNOISE AVERAGES** | <num>
<num var>
<num expr> |

Function : Tx Noise Tx filter

Syntax : **TEST TXNOISE TXFILTER** | NONE
LP15KHZ
LP300HZ
STDBP
CCITT
CMESS |

Function : Tx Noise upper limit

Syntax : **TEST TXNOISE UPPER** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

TEST TXSINAD

Function : Tx SINAD status

Syntax : **TEST TXSINAD STATUS** | ON
OFF |

Function : Tx SINAD averages

Syntax : **TEST TXSINAD AVERAGES** | <num>
<num var>
<num expr> |

Function : Tx SINAD modulation level

Syntax : **TEST TXSINAD MODLEVEL** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

Function : Tx SINAD Tx filter

Syntax : **TEST TXSINAD TXFILTER** | NONE
LP15KHZ
LP300HZ
STDBP
CCITT
CMESS |

Function : Tx SINAD lower limit

Syntax : **TEST TXSINAD LOWER** | <num>
<num var>
<num expr> |

Function : Tx SINAD NB modulation level [NAMPS and NTACS only]

Syntax : **TEST TXSINAD NBMODLEVEL** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

TEST TXSN

Function : Tx Signal-to-Noise ratio status

Syntax : **TEST TXSN STATUS**
ON
OFF

Function : Tx Signal-to-Noise ratio averages

Syntax : **TEST TXSN AVERAGES**
<num>
<num var>
<num expr>

Function : Tx Signal-to-Noise ratio modulation level

Syntax : **TEST TXSN MODLEVEL**
<num>
<num var>
<num expr>
[Hz
kHz
MHz]

Function : Tx Signal-to-Noise ratio Tx filter

Syntax : **TEST TXSN TXFILTER**
NONE
LP15KHZ
LP300HZ
STDBP
CCITT
CMESS

Function : Tx Signal-to-Noise ratio lower limit

Syntax : **TEST TXSN LOWER**
<num>
<num var>
<num expr>
DB

Function : Tx Signal-to-Noise ratio NB modulation level [NAMPS and NTACS only]

Syntax : **TEST TXSN NBMODLEVEL**
<num>
<num var>
<num expr>
[Hz
kHz
MHz]

Cellular and trunking test commands

TEST CONFIGURATION

Note : CONFIGURATION is *not* a test; it is a store for non-specific parameters that is used with the **READPARAM** and **WRITEPARAM** commands (PMR only). It is included in this listing for completeness.

Function :	Switches duplex testing on or off (PMR only)		
Syntax :	**WRITEPARAM CONFIGURATION DUPLEX**	ON OFF	

TEST DATADEVN

Function :	Data deviation status		
Syntax :	**TEST DATADEVN STATUS**	ON OFF	
Function :	Data deviation reference		
Syntax :	**TEST DATADEVN REF**	<num> <num var> <num expr>	[Hz kHz MHz]
Function :	Data deviation error tolerance		
Syntax :	**TEST DATADEVN ERROR**	<num> <num var> <num expr>	%

TEST DATAPERFORM

Function :	Data Performance status		
Syntax :	**TEST DATAPERFORM STATUS**	ON OFF	
Function :	Data Performance RF generator level		
Syntax :	**TEST DATAPERFORM RFGENLEVEL**	<num> <num var> <num expr>	DBM
Function :	Data Performance lower limit		
Syntax :	**TEST DATAPERFORM LOWER**	<num> <num var> <num expr>	%

TEST DSATDEVN

Function : DSAT Deviation status

Syntax : **TEST DSATDEVN STATUS** | ON / OFF |

Function : DSAT Deviation averages

Syntax : **TEST DSATDEVN AVERAGES** | <num> / <num var> / <num expr> |

Function : DSAT Deviation reference

Syntax : **TEST DSATDEVN REF** | <num> / <num var> / <num expr> | [| Hz / kHz / MHz |]

Function : DSAT Deviation error tolerance

Syntax : **TEST DSATDEVN ERROR** | <num> / <num var> / <num expr> | %

TEST DTMFDECODE

Function : DTMF Decode status

Syntax : **TEST DTMFDECODE STATUS** | ON / OFF |

Function : DTMF Decode timeout

Syntax : **TEST DTMFDECODE TIMEOUT** | <num> / <num var> / <num expr> | SEC

Function : DTMF Decode use accessory port

Syntax : **TEST DTMFDECODE USEACCPORT** | ON / OFF |

Function : DTMF Decode set accessory port

Syntax : **TEST DTMFDECODE ACCPORT** | LOGIC0 / LOGIC1 / LOGIC2 / LOGIC3 |

TEST HANDOFF

Function : Any Handoff command

Caution : To select the channel to handoff to, the following line needs to be inserted into the program before any **TEST HANDOFF** command:-

WRITEPARAM "VCHAN"	<num> <num var> <num expr>

Example : The following program segment will do a handoff to channel 581:-

```
WRITEPARAM "VCHAN" 581
TEST HANDOFF
```

Note: *If the channel selected does not exist, the instrument will find the next highest valid channel and handoff to that channel instead.*

To read back the channel the mobile is on, the following line must be executed:-

READPARAM "VCHAN" <num var>

Function : Handoff RF generator level

Syntax :

TEST HANDOFF RFGENLEVEL	<num> <num var> <num expr>	DBM

Function : Handoff timeout [MPT only]

Syntax :

TEST HANDOFF TIMEOUT	<num> <num var> <num expr>	SEC

Function : Handoff channel type [NAMPS only]

Syntax :

TEST HANDOFF CHANTYPE	WIDE NARROW ABOVE BELOW ROTATE LAST

Function : Handoff channel type [NTACS only]

Syntax :

TEST HANDOFF CHANTYPE	WIDE NARROW ROTATE LAST

Function : Handoff mobile power level [NMT only]

Syntax :

TEST HANDOFF MOBILELEVEL	<num> <num var> <num expr>

TEST HOOKFLASH

Function : Hook Flash status

Syntax : **TEST HOOKFLASH STATUS** | ON / OFF |

Function : Hook Flash timeout

Syntax : **TEST HOOKFLASH TIMEOUT** | <num> / <num var> / <num expr> | SEC

Function : Hook Flash use accessory port

Syntax : **TEST HOOKFLASH USEACCPORT** | ON / OFF |

Function : Hook Flash set accessory port

Syntax : **TEST HOOKFLASH ACCPORT** | LOGIC0 / LOGIC1 / LOGIC2 / LOGIC3 |

TEST HSDEVN

Function : High speed data deviation status

Syntax : **TEST HSDEVN STATUS** | ON / OFF |

Function : High speed data deviation reference

Syntax : **TEST HSDEVN REF** | <num> / <num var> / <num expr> | [| Hz / kHz / MHz |]

Function : High speed data deviation error tolerance

Syntax : **TEST HSDEVN ERROR** | <num> / <num var> / <num expr> | %

TEST LANDCLEAR

Function : Clear from Land RF generator level

Syntax : **TEST LANDCLEAR RFGENLEVEL** | <num> / <num var> / <num expr> | DBM

Function : Clear from Land use accessory port

Syntax : **TEST LANDCLEAR USEACCPORT** | ON / OFF |

Function : Clear from Land set accessory port

Syntax : **TEST LANDCLEAR ACCPORT** | LOGIC0 / LOGIC1 / LOGIC2 / LOGIC3 |

TEST LISTENOFF

Function : Listen Off RF generator level

Syntax : **TEST LISTENOFF RFGENLEVEL** | <num> / <num var> / <num expr> | DBM

TEST LISTENON

Function : Listen On RF generator level

Syntax : **TEST LISTENON RFGENLEVEL** | <num> / <num var> / <num expr> | DBM

TEST LSDEVN

Function : Low speed data deviation status

Syntax : **TEST LSDEVN STATUS** | ON / OFF |

Function : Low speed data deviation reference

Syntax : **TEST LSDEVN REF** | <num> / <num var> / <num expr> | [| Hz / kHz / MHz |]

Function : Lowspeed data deviation error tolerance

Syntax : **TEST LSDEVN ERROR** | <num> / <num var> / <num expr> | %

TEST MOBILECLEAR

Function :	Clear From Mobile RF generator level		
Syntax :	**TEST MOBILECLEAR RFGENLEVEL**	<num> <num var> <num expr>	DBM
Function :	Clear From Mobile timeout		
Syntax :	**TEST MOBILECLEAR TIMEOUT**	<num> <num var> <num expr>	SEC
Function :	Clear From Mobile use accessory port		
Syntax :	**TEST MOBILECLEAR USEACCPORT**	ON OFF	
Function :	Clear From Mobile set accessory port		
Syntax :	**TEST MOBILECLEAR ACCPORT**	LOGIC0 LOGIC1 LOGIC2 LOGIC3	

TEST PAGEMOBILE

Function :	Page Mobile RF generator level		
Syntax :	**TEST PAGEMOBILE RFGENLEVEL**	<num> <num var> <num expr>	DBM
Function :	Page Mobile timeout		
Syntax :	**TEST PAGEMOBILE TIMEOUT**	<num> <num var> <num expr>	SEC
Function :	Page Mobile use accessory port		
Syntax :	**TEST PAGEMOBILE USEACCPORT**	ON OFF	
Function :	Page Mobile set accessory port		
Syntax :	**TEST PAGEMOBILE ACCPORT**	LOGIC0 LOGIC1 LOGIC2 LOGIC3	
Function :	Page Mobile channel type [NAMPS only]		
Syntax :	**TEST PAGEMOBILE CHANTYPE**	WIDE NARROW ABOVE BELOW ROTATE LAST	
Function :	Page Mobile channel type [NTACS only]		
	TEST PAGEMOBILE CHANTYPE	WIDE NARROW ROTATE LAST	
Function :	Page Mobile power level [NMT only]		
Syntax :	**TEST PAGEMOBILE MOBILELEVEL**	<num> <num var> <num expr>	
Function :	Page Mobile set call type [EDACS radio only]		
Syntax :	**TEST PAGEMOBILE CALLTYPE**	INDIVIDUAL GROUP EMERGENCY ROTATE	

TEST PLACECALL

Function : Place Call RF generator level

Syntax : **TEST PLACECALL RFGENLEVEL** | <num>
<num var>
<num expr> | DBM

Function : Place Call timeout

Syntax : **TEST PLACECALL TIMEOUT** | <num>
<num var>
<num expr> | SEC

Function : Place Call use accessory port

Syntax : **TEST PLACECALL USEACCPORT** | ON
OFF |

Function : Place Call set accessory port

Syntax : **TEST PLACECALL ACCPORT** | LOGIC0
LOGIC1
LOGIC2
LOGIC3 |

Function : Place Call channel type [NAMPS only]

Syntax : **TEST PLACECALL CHANTYPE** | WIDE
NARROW
ABOVE
BELOW
ROTATE
LAST |

Function : Place Call channel type [NTACS only]

Syntax : **TEST PLACECALL CHANTYPE** | WIDE
NARROW
ROTATE
LAST |

Function : Place Call mobile power level [NMT only]

Syntax : **TEST PLACECALL MOBILELEVEL** | <num>
<num var>
<num expr> |

TEST POWERLEVEL

Function : Any Power Level command

Caution : To select the power level to change to and test, the following line needs to be inserted into the program before any **TEST POWERLEVEL** command:-

WRITEPARAM "VMAC" | <num> | [for AMPS and TACS]
| <num var> |
| <num expr> |

WRITEPARAM "TRAFFIC_POWER" | <num> | [for NMT]
| <num var> |
| <num expr> |

Example : The following program segment will test power level 4 [AMPS and TACS]:-

```
WRITEPARAM "VMAC" 4
TEST POWERLEVEL
```

The following program segment will test power level 2 [NMT]:-

```
WRITEPARAM "TRAFFIC_POWER" 2
TEST POWERLEVEL
```

Function : Power Level status

Syntax : **TEST POWERLEVEL STATUS** | ON
OFF |

Syntax : **TEST POWERLEVEL AVERAGES** | <num>
<num var>
<num expr> |

Function : Power Level upper limit (relative to reference power)

Syntax : **TEST POWERLEVEL RUPPER** | <num>
<num var>
<num expr> | DB

Function : Power Level lower limit (relative to reference power)

Syntax : **TEST POWERLEVEL RLOWER** | <num>
<num var>
<num expr> | DB

Function : Power Level high power upper limit (relative to reference power)

Syntax : **TEST POWERLEVEL HIRUPPER** | <num>
<num var>
<num expr> | DB

Function : Power Level high power lower limit (relative to reference power)

Syntax : **TEST POWERLEVEL HIRLOWER** | <num>
<num var>
<num expr> | DB

TEST PTTOFF

Function : PTT OFF timeout

Syntax : **TEST PTTOFF TIMEOUT** | <num> / <num var> / <num expr> | SEC

Function : PTT OFF use pressels [MPT and PMR only]

Syntax : **TEST PTTOFF USEPRESSEL** | ON / OFF |

Function : PTT OFF reference power level [PMR only]

Syntax : **TEST PTTOFF**

Function : PTT OFF use accessory port[EDACS and LTR radio only]

Syntax : **TEST PTTOFF USEACCPORT** | ON / OFF |

Function : PTT OFF set accessory port [EDACS and LTR radio only]

Syntax : **TEST PTTOFF ACCPORT** | LOGIC0 / LOGIC1 / LOGIC2 / LOGIC3 |

TEST PTTON

Function : PTT ON timeout

Syntax : **TEST PTTON TIMEOUT** | <num> / <num var> / <num expr> | SEC

Function : PTT ON use pressels [MPT and PMR only]

Syntax : **TEST PTTON USEPRESSEL** | ON / OFF |

Function : PTT ON use accessory port

Syntax : **TEST PTTON USEACCPORT** | ON / OFF |

Function : PTT ON set accessory port

Syntax : **TEST PTTON ACCPORT** | LOGIC0 / LOGIC1 / LOGIC2 / LOGIC3 |

Function : PTT ON reference power level [PMR only]

Syntax : **TEST PTTON REF** | <num> / <num var> / <num expr> | W

TEST RADIOCALL

Function : Call from radio RF generator level

Syntax : **TEST RADIOCALL RFGENLEVEL** | <num> / <num var> / <num expr> | DBM

Function : Call from radio timeout

Syntax : **TEST RADIOCALL TIMEOUT** | <num> / <num var> / <num expr> | SEC

Function : Call from radio use accessory port

Syntax : **TEST RADIOCALL USEACCPORT** | ON / OFF |

Function : Call from radio set accessory port

Syntax : **TEST RADIOCALL ACCPORT** | LOGIC0 / LOGIC1 / LOGIC2 / LOGIC3 |

TEST RADIOCLEAR

Function : Clear from radio RF generator level

Syntax : **TEST RADIOCLEAR RFGENLEVEL** | <num> / <num var> / <num expr> | DBM

Function : Clear from radio timeout

Syntax : **TEST RADIOCLEAR TIMEOUT** | <num> / <num var> / <num expr> | SEC

Function : Clear from radio use accessory port

Syntax : **TEST RADIOCLEAR USEACCPORT** | ON / OFF |

Function : Clear from radio set accessory port

Syntax : **TEST RADIOCLEAR ACCPORT** | LOGIC0 / LOGIC1 / LOGIC2 / LOGIC3 |

TEST RADIOPTTOFF

Function :	PTT Off from radio RF generator level		
Syntax :	**TEST RADIOPTTOFF RFGENLEVEL**	<num> <num var> <num expr>	DBM

Function :	PTT Off from radio timeout		
Syntax :	**TEST RADIOPTTOFF TIMEOUT**	<num> <num var> <num expr>	SEC

Function :	PTT Off from radio use accessory port	
Syntax :	**TEST RADIOPTTOFF USEACCPORT**	ON OFF

Function :	PTT Off from radio set accessory port	
Syntax :	**TEST RADIOPTTOFF ACCPORT**	LOGIC0 LOGIC1 LOGIC2 LOGIC3

TEST RADIOPTTON

Function :	PTT On from radio RF generator level		
Syntax :	**TEST RADIOPTTON RFGENLEVEL**	<num> <num var> <num expr>	DBM

Function :	PTT On from radio timeout		
Syntax :	**TEST RADIOPTTON TIMEOUT**	<num> <num var> <num expr>	SEC

Function :	PTT On from radio use accessory port	
Syntax :	**TEST RADIOPTTON USEACCPORT**	ON OFF

Function :	PTT On from radio set accessory port	
Syntax :	**TEST RADIOPTTON ACCPORT**	LOGIC0 LOGIC1 LOGIC2 LOGIC3

TEST RADIOCLEAR

Function : Clear from radio RF generator level

Syntax : **TEST RADIOCLEAR RFGENLEVEL** | <num> / <num var> / <num expr> | DBM

Function : Clear from radio timeout

Syntax : **TEST RADIOCLEAR TIMEOUT** | <num> / <num var> / <num expr> | SEC

Function : Clear from radio use accessory port

Syntax : **TEST RADIOCLEAR USEACCPORT** | ON / OFF |

Function : Clear from radio set accessory port

Syntax : **TEST RADIOCLEAR ACCPORT** | LOGIC0 / LOGIC1 / LOGIC2 / LOGIC3 |

TEST REGISTER

Function : Registration status

Syntax : **TEST REGISTER STATUS** | ON / OFF |

Function : Registration RF generator level

Syntax : **TEST REGISTER RFGENLEVEL** | <num> / <num var> / <num expr> | DBM

Function : Registration retry enabled [AMPS and TACS only]

Syntax : **TEST REGISTER RETRYENABLED** | ON / OFF |

Function : Registration use accessory port

Syntax : **TEST REGISTER ACCPORT** | ON / OFF |

Function : Regisration set accessory port

Syntax : **TEST USEACCPORT** | LOGIC0 / LOGIC1 / LOGIC2 / LOGIC3 |

Function : Regisration mobile power level [NMT only]

Syntax : **TEST REGISTER MOBILELEVEL** | <num> / <num var> / <num expr> |

TEST SATDEVN

Function : SAT Deviation status

Syntax : **TEST SATDEVN STATUS** | ON
OFF |

Function : SAT Deviation reference

Syntax : **TEST SATDEVN REF** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

Function : SAT Deviation error tolerance

Syntax : **TEST SATDEVN ERROR** | <num>
<num var>
<num expr> | %

TEST SATFREQ

Function : SAT Frequency status

Syntax : **TEST SATFREQ STATUS** | ON
OFF |

Function : SAT Frequency error tolerance

Syntax : **TEST SATFREQ ERROR** | <num>
<num var>
<num expr> | %

TEST SPOTFREQ

Note : SPOTFREQ is *not* a test; it is a store for spot frequency parameters that is used with the **READPARAM** and **WRITEPARAM** commands (PMR only). It is included in this listing for completeness.

Function : Spot Frequency status

Syntax : **WRITEPARAM SPOTFREQ STATUS** | ON
OFF |

Function : Spot frequency for Tx channel *n*, where *n* = 1 to 16

Syntax : **WRITEPARAM SPOTFREQ TXCHAN*n*** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

Function : Spot frequency for Rx channel *n*, where *n* = 1 to 16

Syntax : **WRITEPARAM SPOTFREQ RXCHAN*n*** | <num>
<num var>
<num expr> | [| Hz
kHz
MHz |]

TEST STDEVN

Function: ST Deviation status

Syntax: **TEST STDEVN STATUS** | ON / OFF |

Function: ST Deviation reference

Syntax: **TEST STDEVN REF** | <num> / <num var> / <num expr> | [| Hz / kHz / MHz |]

Function: ST Deviation error tolerance

Syntax: **TEST STDEVN ERROR** | <num> / <num var> / <num expr> | %

TEST STDURN

Function: ST Duration status

Syntax: **TEST STDURN STATUS** | ON / OFF |

Function: ST Duration error tolerance

Syntax: **TEST STDURN ERROR** | <num> / <num var> / <num expr> | %

TEST STFREQ

Function: ST Frequency status

Syntax: **TEST STFREQ** | ON / OFF |

Function: ST Frequency error tolerance

Syntax: **TEST STFREQ ERROR** | <num> / <num var> / <num expr> | %

Function: ST Frequency reference

Syntax: **TEST STFREQ REF** | <num> / <num var> / <num expr> | [| Hz / kHz / MHz |]

TEST TESTSETCALL

Function : Call from Aeroflex RF generator level

Syntax : **TEST TESTSETCALL RFGENLEVEL** | <num>
<num var>
<num expr> | DBM

Function : Call from Aeroflex timeout

Syntax : **TEST TESTSETCALL TIMEOUT** | <num>
<num var>
<num expr> | SEC

Function : Call from Aeroflex use accessory port

Syntax : **TEST TESTSETCALL USEACCPORT** | ON
OFF |

Function : Call from Aeroflex set accessory port

Syntax : **TEST TESTSETCALL ACCPORT** | LOGIC0
LOGIC1
LOGIC2
LOGIC3 |

Function : Call from Aeroflex set call type

Syntax : **TEST TESTSETCALL CALLTYPE** | INDIVIDUAL
GROUP
EMERGENCY
ROTATE |

TEST TESTSETCLEAR

Function : Clear from Aeroflex RF generator level

Syntax : **TEST TESTSETCLEAR RFGENLEVEL** | <num>
<num var>
<num expr> | DBM

Function : Clear from Aeroflex timeout

Syntax : **TEST TESTSETCLEAR TIMEOUT** | <num>
<num var>
<num expr> | SEC

Function : Clear from Aeroflex use accessory port

Syntax : **TEST TESTSETCLEAR USEACCPORT** | ON
OFF |

Function : Clear from Aeroflex set accessory port

Syntax : **TEST TESTSETCLEAR ACCPORT** | LOGIC0
LOGIC1
LOGIC2
LOGIC3 |

TEST TESTSETPTTOFF

Function : PTT Off from Aeroflex RF generator level

Syntax :	**TEST TESTSETPTTOFF RFGENLEVEL**	<num> <num var> <num expr>	DBM

Function : PTT Off from Aeroflex timeout

Syntax :	**TEST TESTSETPTTOFF TIMEOUT**	<num> <num var> <num expr>	SEC

TEST TESTSETPTTON

Function : PTT On from Aeroflex RF generator level

Syntax :	**TEST TESTSETPTTON RFGENLEVEL**	<num> <num var> <num expr>	DBM

Function : PTT On from Aeroflex timeout

Syntax :	**TEST TESTSETPTTON TIMEOUT**	<num> <num var> <num expr>	SEC

Chapter 4
INTRODUCTION TO REMOTE CONTROL

Contents

List of figures

Introduction

The Service Monitor can be controlled remotely over either the RS232 interface which is a standard feature, or over the optional GPIB. The command set used is designed to comply with IEEE488.2-1987 which is a specification for GPIB.

Programs to control the Service Monitor remotely over the two interfaces will have much in common, the main differences being the way in which characters are transmitted.

Control characters are used over the RS232 interface to simulate some of the features of the GPIB interface. A list of these, with their respective actions, is given later in this chapter.

IEEE 488.2 conventions

A simple explanation of the structure of how commands and the data they take or return is presented is given here. For more complete information refer to the latest copy of the IEEE488.2 specification.

Command headers/compound headers

Compound headers allow a complex set of commands to be built up from a smaller set of basic elements in a 'tree' structure. The elements of a compound header are separated by a colon (**:**).

The use of compound headers brings a number of advantages. Commands are less cryptic compared with a traditional 'flat' instrument command set.

Example:

AFGEN1:FREQ

:LEVEL

:SHAPE

:STATUS

Although it is possible to use the full compound header starting from the tree root every time,

(e.g. AFGEN1:FREQ 1KHZ;AFGEN1:SHAPE SINE),

sequences of <COMMAND MESSAGE UNITS> and <QUERY MESSAGE UNITS> can often be shortened by taking advantage of the special rules which apply to compound headers.

Having 'descended' the tree, (for example to create the <PROGRAM MESSAGE UNIT> AFGEN1:SHAPE SINE), any other elements at that level may be included in the <PROGRAM MESSAGE> without repeating the entire path through the tree.

Example:

AFGEN1:FREQ 1KHZ;SHAPE SINE

is equivalent to the two <PROGRAM MESSAGES>

AFGEN1:FREQ 1KHZ followed by AFGEN1:SHAPE SINE.

Note the use of the <PROGRAM MESSAGE UNIT SEPARATOR> character ";" between <PROGRAM MESSAGE UNITS>.

Here is another example.

MODSCOPE:TBASE SC_500US;TRIG REPEAT

is equivalent to the two <PROGRAM MESSAGES>:

MODSCOPE:TBASE SC_500US and MODSCOPE:TRIG REPEAT

To return to the top of the tree so that another "branch" may be descended, a colon is used.

Example:

MODGEN1:FREQ 10KHZ;LEVEL 100MV;:MODGEN2:FREQ 3KHZ

Abbreviations

In general, header elements can be abbreviated to the shortest unique string at that level and part of the command tree.

Example:

AFGEN1:F is equivalent to AFGEN1:FREQ

Program data

The following program data functional elements are accepted by the Service Monitor:

<CPD> (also known as <CHARACTER PROGRAM DATA>)

<NRf> (also known as <DECIMAL NUMERIC PROGRAM DATA>)

<STRING PROGRAM DATA>

<ARBITRARY BLOCK PROGRAM DATA>

All these functional elements are defined in IEEE 488.2-1987.

<CPD>

Character program data is used to set a parameter to one of a number of states that are best described by short alphanumeric strings.

Example:

ON

OFF and ON are the possible <CPD> elements to set the status of the RF generator. Note that when setting the parameter, the short form (i.e. OF and ON) may be used.

<NRf>

Flexible numeric representation (also known as <DECIMAL NUMERIC PROGRAM DATA>) covers integer and floating point representations.

Examples:

–466	Integer value.
4.91	Explicitly placed decimal point.
59.5E+2	Mantissa and Exponent representation

The format is known as "flexible" because any of the three representations may be used for any type of numeric parameter.

Example:

Where a parameter requires an integer value in the range 1 to 100, and the user needs to set its value to 42, the following values will be accepted by the Service Monitor.

42	Integer
42.0	Floating point.
4.2E1, 4200E-2	Floating point - Mantissa/exponent.
41.5	Rounded up to 42
42.4	Rounded down to 42

<STRING PROGRAM DATA>

String program data consists of a number of ASCII characters enclosed in quotes. Either a pair of single ('ASCII 39') or double ("ASCII 34") quotes may be used. If the quote character chosen to mark the beginning and end of the string also appears within it, it must be doubled.

Example:

'This string contains the word "Hello"'
will be interpreted as the string:
This string contains the word 'Hello'

<ARBITRARY BLOCK PROGRAM DATA>

This format is used for the transmission of large quantities of 8-bit binary data.

Since it is not intended that the user should ever need to compile data of this type for transmission to the Service Monitor, details of the format are not given here.

Note that data received from the Service Monitor as <INDEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA> is already in a form suitable for transmission back to the Service Monitor as <ARBITRARY BLOCK PROGRAM DATA>.

Also note that since only the indefinite length form is used, the data must be terminated by line feed with EOI asserted. This means that a command requiring <ARBITRARY BLOCK PROGRAM DATA> must be the last <PROGRAM MESSAGE UNIT> of the <PROGRAM MESSAGE>.

Response data

The following response data functional elements are generated by the Service Monitor:

<CRD> (also known as <CHARACTER RESPONSE DATA>)

<NR1>

<NR2>

<NR3>

<STRING RESPONSE DATA>

<INDEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA>

<ARBITRARY ASCII RESPONSE DATA>

<BOOLEAN RESPONSE DATA>

<CRD>

This type of response is returned when reading the value of a parameter which can take a number of discrete states. States are represented by short alphanumeric strings.

Example:

ON

OFF and ON are the possible <CRD> responses if the parameter which determines the status of the RF frequency generator is queried.

Note that when setting the parameter, the short form (i.e. OF and ON) may be used. When the parameter is queried, the long form is always returned.

<NR1>

This type of numeric response is used when returning the value of integer parameters, such as averaging number or number of measurement points.

Examples:

15
+3
-57

<NR2>

This type of numeric response includes an explicitly placed decimal point, but no exponent.

Examples:

17.91
-18.27
+18.83

<NR3>

This type of numeric response includes an explicitly placed decimal point and an exponent.

Examples:

1.756E+2
182.8E–3

<STRING RESPONSE DATA>

This takes a similar form to <STRING PROGRAM DATA> except that the delimiting character is always a double quote, ("ASCII 34").

<INDEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA>

This form of response is used when reading blocks of 8-bit binary data from the Service Monitor. Examples include settings and results store contents.

The format comprises a '#' character followed by a '0' followed by the data, followed by a newline character (ASCII 10). EOI is asserted with the terminating newline character.

Because EOI is always used as a terminator, a <QUERY MESSAGE UNIT> which generates data in this form must be the last <QUERY MESSAGE UNIT> in the <PROGRAM MESSAGE>.

<INDEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA> cannot be used over RS232 remote control.

<ARBITRARY ASCII RESPONSE DATA>

This takes the form of an ASCII string terminated by newline (ASCII 10) with EOI asserted.

Notes on interpreting data returned in this format will be found in the descriptions for the few commands that use it.

Because EOI is always used as a terminator, a <QUERY MESSAGE UNIT> which generates data in this form must be the last <QUERY MESSAGE UNIT> in the <PROGRAM MESSAGE>.

Terminators

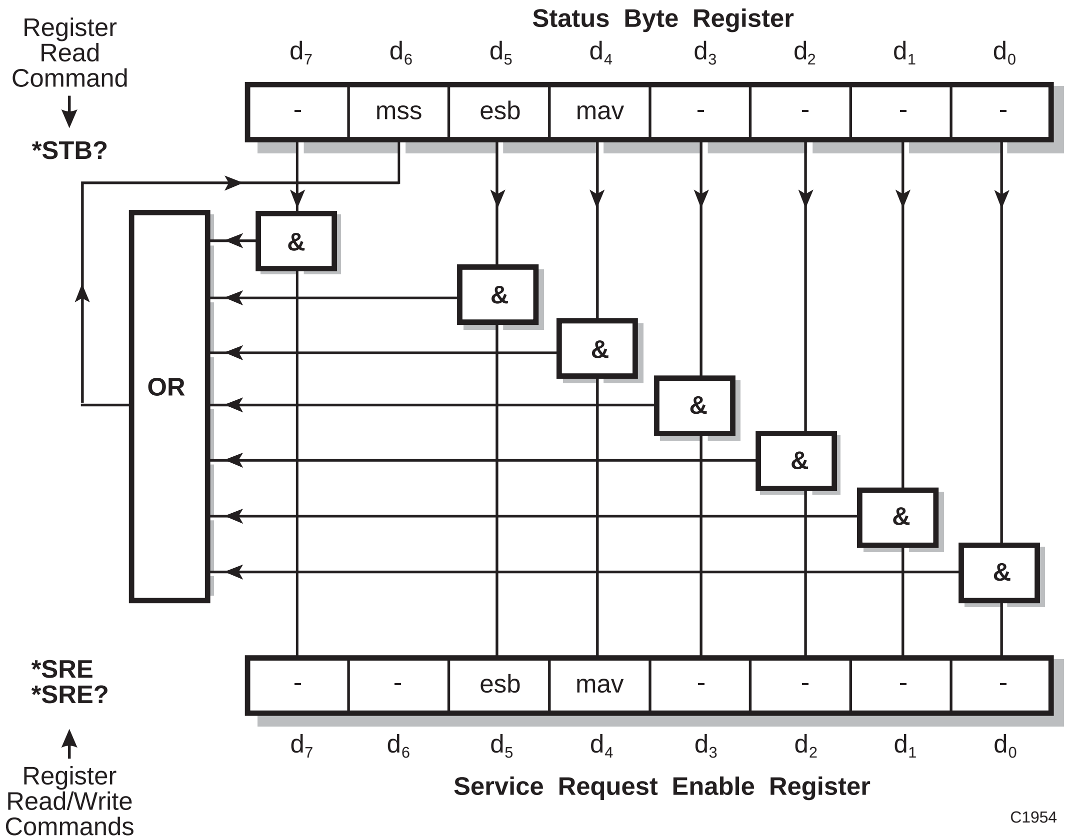
A **<PROGRAM MESSAGE TERMINATOR>** (as defined in IEEE 488.2-1987) can be a newline character (ASCII 10), a newline character with the ^END message asserted at the same time, or an ^END message asserted with the final character of the <PROGRAM MESSAGE>. The terminator may be preceded by any number of "white space" characters - i.e. any single ASCII-encoded byte in the range 0 to 9 and 11 to 32 decimal.

A **<RESPONSE MESSAGE TERMINATOR>** (as defined in IEEE 488.2-1987) is a newline character with the ^END message asserted at the same time.

Many GPIB controllers terminate program messages with a newline character and, by default, accept newline as the response message terminator. When transferring binary data - which may contain embedded newline characters - it is necessary to ensure that the controller uses only ^END messages. Usually this requires the controller's GPIB interface to be set up to generate and detect ^END. Refer to the documentation supplied with the controller.

Status reporting

The Service Monitor has a status reporting structure implemented as per IEEE488.2-1987. The purpose of this is to inform the controller/user program of events or errors as they occur within the Service Monitor. Particular events can be ignored by programming mask registers using the common commands. Refer to Fig. 4-1 *Status byte when read by *STB* and Fig 4-2 *Standard events register (as defined in IEEE 488.2 1987)*.

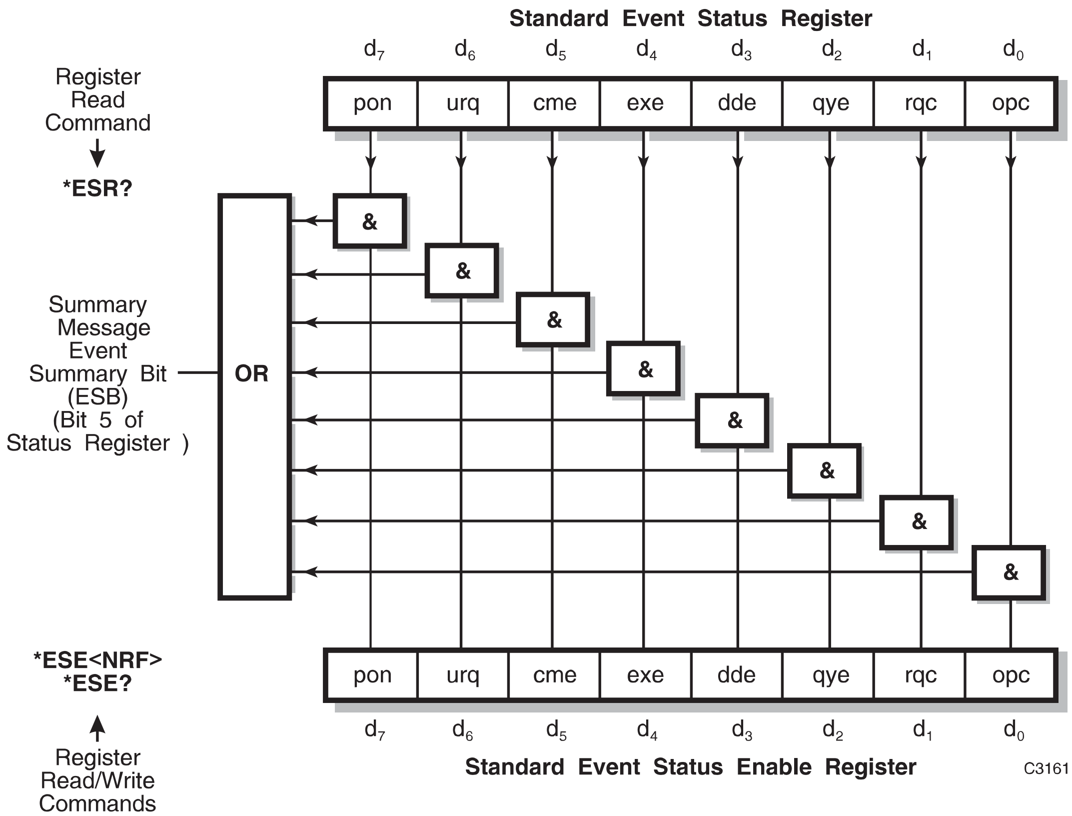


d_0	**Not used**	
d_1	**Not used**	
d_2	**Not used**	
d_3	**Not used**	
d_4	**MAV**	**Message available in output queue**
d_5	**ESB**	**Event Status Register Summary Bit**
d_6	**MSS**	**True when (Status Byte > 0) AND (Enable Reg > 0)**
d_7	**Not used**	

Notes...

When read by Serial Poll (rather than *STB?), d_6 contains RQS (Request Service) as defined in IEEE 488.2.

*Fig. 4-1 Status byte when read by *STB*



d_0	**OPC**	**Operation Complete**
d_1	**RQC**	**Request Control - Not implemented in this instrument**
d_2	**QYE**	**Query Error**
d_3	**DDE**	**Device-Specific Error**
d_4	**EXE**	**Execution Error**
d_5	**CME**	**Command Error**
d_6	**URQ**	**User Request - Not implemented in this instrument**
d_7	**PON**	**Power on**

Fig. 4-2 Standard events register (as defined in IEEE 488.2 1987)

At the top of the reporting structure is the status byte. Corresponding to the status byte is the service request enable register. When the result of masking the status byte with the service request enable register is non zero then the request for service (RQS) bit is set. Over the GPIB this causes an SRQ (service request) at the controller.

By programming the mask registers appropriately the Service Monitor could be set to produce an SRQ upon a particular error so that a recovery routine could be run in the users program. Similarly an SRQ upon a message being ready in the output buffer is a typical use of the status reporting.

Polling the status reporting registers is just as valid a method of getting information on Service Monitor state.

Only three bits of the status byte are used in the Service Monitor.

> Bit 4, the MAV (message available) bit, states that the output buffer is not empty and therefore a message is waiting to be read.
>
> Bit 5 is the ESB or event status bit. It is a summary of the standard event status register, standard event enable register combination.
>
> Bit 6 is the RQS or request service bit.

The status byte can be read in two ways.

Firstly by performing a serial poll. Once a serial poll has been done the RQS bit is cleared until a new event causes it to be set.

Secondly by using the *STB? common command. In this case the RQS bit is replaced by MSS (master summary status) which is set when either the ESB or MAV is set and unmasked.

The event status bit summarizes the contents of the standard event status register. This register contains bits for events or errors occurring over the communications interface. This includes protocol errors, command (syntax) errors and the bit set upon operation complete.

This register can be **destructively** read with the *ESR? common command. Bits 2, 3, 4 and 5, the error bits, are read by the error query commands COMerror?, DEVerror?, Execerror?, and Qerror?. See Table 4-1 Error data

The mask register associated with the standard event status register is the standard event status enable register. Only unmasked set bits in the status register cause the ESB to be set. The *ESE common command allows the event status enable register to be set and cleared.

The *CLS common command can be used to clear any event bits that have been set.

Table 4-1 Error data

<QUERY MESSAGE UNIT>	Error data	Indication
COMmerror?	0	'No Error'
	1	'Illegal * Command'
	2	'Parameter not allowed'
	3	'Unrecognized mnemonic'
	4	'Mnemonic not unique'
	5	'Write not allowed'
	6	'Read not allowed'
	7	'Syntax error'
:DEVerror	0	'No Error'
	1	'Value out of range'
	2	'Wrong mode for measurement'
	3	'Wrong setup for measurement'
	4	'Cannot change item'
	5	'Wrong setup for command'
	6	'Option not fitted'
	7	'Systems test in progress'
	8	'Store empty'
	9	'No memory card present'
	10	'Card not formatted'
	11	'No card interface fitted'
	12	'File not found'
	13	'Not a settings store for recall'
:Execerror	0	'No Error'
	1	'Num option data out of range'
	2	'Excess data'
	3	'Insufficient data'
	4	'Data required'
	5	'Unrecognized text option'
	6	'Alpha text not unique'
	7	'Unrecognized suffix'
	8	'Suffix not allowed'
:Qerror	0	'No Error'
	1	'Interrupted'
	2	'Unterminated'
	3	'Deadlocked'

Message exchange protocol

IEEE488.2-1987 defines a protocol for the exchange of messages between devices. There are three error states that the instrument can enter, if this protocol is broken. The error states and the reason they occur are:

1) **UNTERMINATED** occurs when the controller attempts to read a response without having sent a complete query.
2) **INTERRUPTED** occurs when the controller starts to send a new message before having read the response to a preceding terminated query.
3) **DEADLOCK** happens if the input and output buffers both become full. This can only occur if the controller has sent a long message containing many queries. The Service Monitor has input and output buffers of 256 bytes length. The output buffer is effectively full if there is insufficient room for it to contain the next formatted message.

IEEE 488.1 Operations and states

Device Clear

Device Clear is an operation defined over the GPIB bus. Upon receiving a **Device Clear** the instrument is sent into the remote state, clears both its input and output buffers and resets the remote software to a known state. It does not alter the state of any flags within the status reporting other than the message available.

The main use of **Device Clear** is to reset the communications and is used when there has been any communication problem. It is good practice to send a **Device Clear** at the beginning of a remote program.

Local Lockout

Over **Remote** the controller can set the instrument into **Local Lockout** state. When **Local Lockout** is set the front panel is disabled and the **LOCAL** key will be made ineffective. Local lockout is often used when the instrument is part of an automatic test system and left unattended. In this state the instrument cannot be affected by operation of the front panel. Sending the instrument local over the remote does not release this state. The keyboard can only be re-enabled by releasing **Local Lockout** over the remote interface or by switching the supply off and on.

RS232 Features

Handshaking

Handshaking of communications over the GPIB is automatic but over the RS232 the Service Monitor implements it in two ways. Firstly by using the handshake characters XOFF - stop transmitting - and XON - start - and secondly by using the DTR and DSR lines. While the DSR line is inactive the instrument does not transmit. If the test set wishes the controller to stop transmitting it de-asserts its DTR line.

Control characters

The following list shows the control characters that are used over the RS232 system to simulate certain features of the IEEE 488 interface.

^A (control A 01H) - connect or go to remote

^D (control D 04H) - disconnect or go to local

^T (control T 14H) - device clear

^R (control R 12H) - local lockout

^P (control P 10H) - release local lockout

^Q (control Q 11H) - XON char for software handshake

^S (control S 13H) - XOFF char for software handshake

^X (control X 18H) - serial poll forces transmission of status byte over RS232

Line feed/carriage return

Do not enter a carriage return after a line feed character. The line feed is seen as a valid terminator, and any additional carriage return will be misinterpreted as a character intended for the next message.

Command layout

In the list of commands, each command is set out as follows:

(1) Path from the subsystem root

Example:

:AFGEN1
    :FREQ

(2) Description

Describes the purpose of the command.

(3) Parameters

The first line lists each parameter, stating its <PROGRAM DATA> functional element (as defined in IEEE 488.2-1987).

Subsequent lines explain the meaning of each parameter.

Angle brackets <...> indicate that the enclosed parameter is described in more detail later in the text.

Example:

<CPD> or <NRf>

Status Selection

The first line states that the command takes one parameter. This parameter can be either character program data or a numeric value. The second line, *(Italics)*, describes the parameter.

(4) Allowed suffixes

A list of the suffixes or units allowed for numeric values is provided.

Example:

MHZ,KHZ

This would mean that a frequency could be entered with either MHZ or KHZ units.

(5) Default suffix

If a command takes a numeric parameter which has a unit, then if a value is sent without a suffix it is assumed to be in the units of the default suffix.

Example:

MHZ

A number sent without a suffix for this command is assumed to be in MHz.

(6) Valid data

Commands that respond to specific alpha data in order to set a condition, will also respond to numerical data corresponding to the position of the alpha command in the valid data listing.

Examples:

a. (From :ACcessories:Dpowertype)

Valid data: 0 or CW
1 or PEP

ACCESS:DPOWER 1 or ACCESS:DPOWER PEP
have the same meaning

b. (From :RXFilt)

Valid Data: 0 or LP_50KHZ
1 or LP_15KHZ
2 or STD_BP
3 or LP_300HZ

RXFILT 2 or RXFILT STD_BP
have the same meaning.

(7) Example

An example of the use of the command is provided.

(8) Response

Query responses follow the same format as parameter definitions. The first line shows the response in terms of its IEEE 488.2 functional elements, and below it is given the semantics of the response.

Example:

<NR2>
Frequency (Hz)

(9) Example response

This field gives an example of a typical response from a query. Usually this corresponds to the example field.

Getting started

This section provides an introduction to Remote programming of the Service Monitor, including a worked example.

The remote command set

The first point to notice when controlling the Service Monitor over one of the remote interfaces is that there is not a straightforward mapping between manual front panel operations and their remote equivalents.

Common commands

The IEEE 488.2 common commands all start with a * character. Those which are implemented in the Service Monitor are listed at the start of Chapter 5 of this manual.

The most important command is *RST, which places the instrument in a defined state. It is good practice to send *RST at the start of any remote program.

Preparing the Service Monitor for REMOTE operation

RS232 Serial port

The connections required between the RS232 serial port and the controlling device are described in the operating manual for the particular Service Monitor. (See *Chapter 2, Installation* of the relevant manual under *Remote Control Connections*.)

Entering remote for RS232

The *[Remote Control]* key on setup page 2, allows the user to select which of the remote control systems is active.

RS232 remote control can be selected simultaneously with RS232 selected as the printer option.

Serial port parameters

The RS232 serial port of the Service Monitor is used for connecting to a printer and for the serial remote control. The *[Serial Setup]* key, also on setup page 2, gives access to the display shown in Fig. 4-3 *Serial set up menu*.

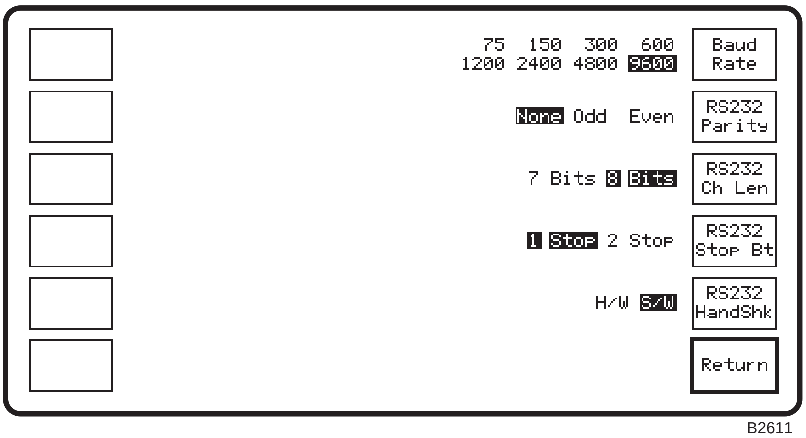


Fig. 4-3 Serial setup menu

The parameters are set by repeated presses of the key, toggling through the available options. Under most operating conditions the default settings are the performance optima. These should be used unless the controller device requires a different setting.

Baud Rate. The default setting is 9600. A slower rate may be required if control is via a modem.

RS232 Parity. Default setting, None (no parity bits).

RS232 Channel Length. Default 8 Bits.

RS232 Stop Bits. Default 1 Stop (one stop bit).

RS232 Handshake. The default is S/W (software handshaking).

GPIB control port

The connections required between the GPIB interface port and the controlling device are described in the operating manual's *Chapter 2, Installation*, under *Remote Control Connections*.

GPIB address

The Service Monitor must be given an address code before it can be used by remote control over the GPIB. This address is entered on setup page 2 by using the *[GPIB Addr]* key. Pressing this key allows the required address number to be entered using the data entry keys. The number must be unique on the system to the instrument and within the range 1 to 30.

Example: simple receiver final test

In this example, Service Monitor remote commands are stated without making any assumptions about the controller and programming language to be used. These commands, of course, will need to be incorporated into the program language statements of the target controller. Here are some examples of how this would be done in practice, using the reset command, *RST. The Service Monitor address is assumed to be 8.

*RST
Command as printed in the example.
PRINT @8:"*RST"
Controller using TBASICR programming language (TransEra Corporation).
OUTPUT 708;"*RST"
Controller using HTBASICTM programming language (TransEra Corporation).

It may sometimes be necessary to send a DEVICE CLEAR command, if the GPIB system fails to respond to *RST or appears to lock up. Examples of this command are as follows:

DEVICE CLEAR	Command as printed in the example.
WRITE GPIB CMD_SDC(8)	Controller using TBASICR.
CLEAR 708	Controller using HTBASICTM

Step 1. Preset the Instrument to a Known State

DEVICE CLEAR

*RST

Preset the instrument.

Step 2. Select the instrument mode for the test

TEST RX

Select Receiver test mode.

Remember that IEEE 488.2 requires a single space character between the command header and its parameter(s).

Step 3. Set RF output port, frequency and level

GENSW GEN_N

Select the N-type output port.

RFGEN:FREQ 470.0

Set the Service Monitor signal generator frequency to 470 MHz.

RFGEN:LEV -110DBM

Set the signal generator level to -110 dBm.

Step 4. Set Mod type, Mod gen level

MODTYPE FM

Generate frequency modulation.

MODGEN2:FMDEVN 6KHZ

Set mod gen 2 deviation to 6 kHz

Step 5. Set distortion measurement type

RXDISTN SINAD

Select the measurement of receiver distortion to be SINAD. A requested measurement of SINAD will cause an error if the correct measurement type is not selected.

Step 6. Turn off instrument measure cycle

MEASCYCL OFF

Measurements within the Service Monitor are taken sequentially in a loop. If the current measurement is valid when requested remotely and this measurement cycle is running then the current value is returned immediately. When the cycle is stopped then a remote measurement request forces a new measurement to be taken.

Step 7. Measure audio level

MEASU:AFLEVEL?

Take a measurement of audio level at the front panel AF input.

Step 8. Measure audio frequency

MEASU:AFFREQ?

Measure the frequency of the audio signal.

Step 9. Measure audio SINAD

MEASU:RXSINAD?

Request a measurement of receiver SINAD (at the AF input).

Step 10. Turn on the measure cycle

MEASC ON

Restore the measure cycle to a running state. Sending the Service Monitor to local control would do this automatically.

Chapter 5
INSTRUMENT COMMANDS

Contents

Common commands

List of common commands

The following common * (star) commands are implemented:

*CLS	5-1	*OPC	5-2	*SRE?	5-4
*ESE	5-1	*OPC?	5-3	*STB?	5-4
*ESE?	5-2	*OPT?	5-3	*TST?	5-4
*ESR?	5-2	*RST	5-3	*WAI	5-4
*IDN?	5-2	*SRE	5-4		

*CLS

Description: Clear Status Command. Clears all the Status Event registers. Does not affect the Enable Registers.
Note: The IEEE 488.2 Device Clear function only affects the GPIB functions. The input and output buffers are cleared and the instrument put into a state to accept new messages. It does not put the instrument functions into a defined state, this is performed by the *RST common command.

Parameters: N/A

Allowed suffices: N/A

Default suffix: N/A

Example: `*CLS`

*ESE

Description: Standard Event Status Enable Command. Sets the Standard Event Enable Register. Range 0 to 255.

Parameters: <NRf>

Allowed suffices: N/A

Default suffix: N/A

Example: `*ESE 255`

*ESE?

Description: Standard Event Status Enable Query. Returns the value of the Standard Event Status Enable Register as NR1. The range of the returned data is 0 to 255.

Parameters: N/A

Response: <NR1>

Example Response: `255`

*ESR?

Description: Event Status Register Query. Returns the value of the Standard Event Status Register as NR1. The range of the returned data is 0 to 255. This is a destructive read, which clears the register and associated summary bits.

Parameters: N/A

Response: <NR1>

Example Response: `8`

*IDN?

Description: Identification Query. Returns an arbitrary ASCII response comprising four data fields in the format: <Manufacturer>,<type number>,<serial number>,<firmware version number>:<option firmware version><EOM>. Option firmware version refers to the analog systems card. If this is not fitted 00.00 will be returned in this field.

Parameters: N/A

Response: <Arbitrary ASCII response data>, <Arbitrary ASCII response data>, <Arbitrary ASCII response data>, <Arbitrary ASCII response data>.

Example Response: `IFR,2945B, 132637-001,04.00:03.00<EOM>`

*OPC

Description: Operation Complete Command. Sets the Operation Complete bit in the Standard Event Status Register when execution of the preceding operation is complete.

Parameters: N/A

Allowed suffices: N/A

Default suffix: N/A

Example: `*OPC`

*OPC?

Description: Operation Complete Query. Returns a '1' when the preceding operation has been completed.

Parameters: N/A

Response: <NR1>

Example Response: `1`

*OPT?

Description: Returns the state of enabled options in 32 data fields

Parameters: N/A

Response: <NR1>,<NR1>,<NR1>,...,<NR1>

Data field 1, state of 600Ω option
Data field 2, state of Analog systems card option
Data field 4, state of Parallel interface option
Data field 5, state of GPIB option
Data field 6, state of Memory card drive option
Data field 8, state of SSB demodulator option
Data field 9, state of occupied bandwidth option
Data field 21, state of Listen filters option
Data field 22, state of POCSAG decode option
Data field 23, state of CCITT filter option
Data field 24, state of CMESS filter option
Data field 25, state of Avionics option
Data field 26, state of tone remote option
Data field 29, state of distortion notch filter option
Data field 30, state of low noise signal generator option
Data field 32, state of internal battery option
All other fields reserved for future use.

Example Response: `1,1,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,`
`0,0,0,0,0,0,0,0,1,1`

*RST

Description: Reset Command. Sets the instrument functions to the factory default power up state.

Parameters: N/A

Allowed suffices: N/A

Default suffix: N/A

Example: `*RST`

*SRE

Description: Service Request Enable Command. Sets the Service Request Enable Register. Range 0 to 255.

Parameters: <NRf>

Allowed suffices: N/A

Default suffix: N/A

Example: `*SRE 32`

*SRE?

Description: Service Request Enable Query. Returns the value of the Service Request Enable Register as NR1. (Elaborate).

Parameters: N/A

Response: <NR1>

Example Response: `32`

*STB?

Description: Read Status Byte Query. Returns the value of the Status Byte.

Parameters: N/A

Response: <NR1>

Example Response: `32`

*TST?

Description: Self Test Query. Returns a '0' if the instrument passed all self tests.

Parameters: N/A

Response: <NR1>

Example Response: `0`

*WAI

Description: Wait to Continue Command. Inhibits execution of an overlapped command until the execution of the preceding operation has been completed.

Parameters: N/A

Allowed suffices: N/A

Default suffix: N/A

Example: `*WAI`

Instrument-specific commands

These commands are listed in alphabetical order. The tones function commands are in their correct location according to their top level command. These are:

DCSTONES	**(DCstones)**
DTMFTONES	**(DTmftones);**
POCSAGTONES	**(POcsagtones);**
SEQTONES	**(SEQtones);**
TONEREM	**(TONERem);**
TONES	

The commands for the Avionics System, which are only available in the Avionics Communication Service Monitor 2948B with Option 25, are included in the listing. The top level commands for this system are:

ILSGEN	**(Ilsgen)**
MKRBCN	**(MKrbcn)**
SELCAL	**(SELcal)**
VORGEN	**(Vorgen)**

List of instrument-specific commands

:ACcessories

Controls the accessories including the directional power head

Not used alone

:ACcessories

:Dpowertype

Description: Controls how the directional power head accessory takes measurement of power. Carrier wave or Peak envelope power.

Parameters: <CPD> or <NRf>
Power measurement selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or CW
1 or PEP

Example: `ACCESS:DPOWER PEP`

:Dpowertype?

Parameters: N/A

Response: <CRD>
Current selection.

Example Response: `PEP`

:ACcessories

:Freqpower

Description: Sets the RF frequency at which the Light Weight Power Head (*Schomandl*) operates.

Parameters: <CPD> or <NRf>
Frequency

Allowed suffices: MHZ, KHZ, HZ.

Default suffix: MHZ

Example: `ACCESS:FREQPOWER 300.0MHZ`

:Freqpower?

Parameters: N/A

Response: <CRD>
Frequency (MHz).

Example Response: `300.000000`

:ACcessories

:LOGIC*n*

Description:	Controls the state of the accessory logic lines on the Parallel printer option where *n* = 0, 1, 2 or 3
Parameters:	<CPD> or <NRf> *Logic line state*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or LOW 1 or HIGH
Example:	`ACCESS:LOGIC2 HIGH`

:LOGIC*n*?

Parameters:	N/A
Response:	<CRD> *Current selection.*
Example Response:	`HIGH`

:ACcessories

:MODE0

Description:	Controls the operation mode of accessory logic line 0 on the Parallel printer option
Parameters:	<CPD> or <NRf> *Logic line mode*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or AS_SETTING 1 or CLOSE_ON_TX
Example:	`ACCESS:MODE0 AS_SETTING`

:MODE0?

Parameters:	N/A
Response:	<CRD> *Current selection.*
Example Response:	`AS_SETTING`

:ACcessories

:MODE1

Description:	Controls the operation mode of accessory logic line 1 on the Parallel printer option
Parameters:	<CPD> or <NRf> *Logic line mode*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or AS_SETTING 1 or CLOSE_ON_SQ
Example:	`ACCESS:MODE1 AS_SETTING`

:MODE1?

Parameters:	N/A
Response:	<CRD> *Current selection.*
Example Response:	`AS_SETTING`

:ACcessories?

Description:	Queries the status of all the accessories. Produces the combined return values of the sub commands of ACCESSORIES. These responses are separated by semi-colons.
Parameters:	N/A
Response:	<CRD>;<CRD>;<CRD>;<CRD>;<CRD>;<CRD>;<CRD>
Example Response:	`PEP;LOW;HIGH;HIGH;LOW;AS_SETTING;AS_SETTING`

:AFGEN*n*

Controls audio generator *n* where *n* = 1 or 2

Not used alone

:AFGEN*n*

:Freq

Description: Sets Audio Generator *n* Frequency where *n* = 1 or 2.

Parameters: <NRf>

Frequency (kHz)

Allowed suffices: KHZ or HZ

Default suffix: KHZ

Example: `:AFGEN1:FREQ 10.000KHZ`

Sets Audio gen 1 frequency to 10 kHz

:Freq?

Parameters: N/A

Response: <NR2>

Frequency in kHz to 0.1 Hz resolution

Example Response: `5.0000`

Frequency currently set to 5 kHz

:AFGEN*n*

:Level

Description: Sets Audio Generator *n* Level where *n* = 1 or 2

Parameters: <NRf>

Allowed suffices: MV, V, DBM

Default suffix: MV

Example: `:AFGEN1:LEVEL 100MV`

Sets Audio gen 1 level to 100 mV

:Level?

Parameters: N/A

Response: <NR2>

Audio level in mV to 0.1 mV resolution

Example Response: `99.0`

:AFGEN*n*

:SHape

Description:	Sets Audio Generator *n* Shape where *n* = 1 or 2
Parameters:	<CPD> or <NRf> *Shape selection*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or SINE 1 or SQUARE
Example:	`AFGEN1:SHAPE SQUARE` Sets audio gen 1 shape to square

:SHape?

Parameters:	N/A
Response:	<CRD> *Current shape*
Example Response:	`SINE` Audio gen shape is currently set to sine

:AFGEN*n*

:STatus

Description:	Sets Audio Generator *n* Status where *n* = 1 or 2
Parameters:	<CPD> or <NRf> *Status selection*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or OFF 1 or ON
Example:	`AFGEN1:STATUS OFF` Sets audio gen 1 off

:STatus?

Parameters:	N/A
Response:	<CRD> *Current selection*
Example Response:	`OFF`

:AFGEN*n*?

Description: Queries the status of Audio Generator *n* where *n* = 1 or 2. Produces the combined return values of the sub commands of AFGEN1 (AFGEN2). These responses are separated by semi-colons.

Parameters: N/A

Response: <NR2>;<NR2>;<CRD>;<CRD>

Example Response: `10.0000;100.0;SINE;OFF`

:AFGENLock

Description: Locks the AF generator levels to the same value by locking AFGEN 2 level to AFGEN 1 level

Parameters: <CPD> or <NRf>
AF generator locking

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or ON

Example: `:AFGENLOCK OFF`
Removes the locking between AFGEN 1 and AFGEN 2

:AFGENLock?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `OFF`

:AFInput

Description: Controls the coupling of the audio input socket

Parameters: <CPD> or <NRf>
Audio input selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or AC
1 or DC

Example: `AFI AC`
Sets audio input coupling to AC

:AFInput?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `AC`

:AUDFilt

:Filter

Description: Sets the Audio Filter in the Audio path

Parameters: <CPD>, <CPD>, <NRf>
BP Filtertype, HPfilt, Lowpass freq or
<CPD>, <NRf>
LP Filtertype, Lowpass freq or
<CPD>, <CPD>
HP Filtertype, HPfilt

Allowed suffices: NRf in Hz, | kHz

Default suffix: kHz

Valid Data:

Filter type	BP or HP or LP
HPfilt	HP_50 or HP_300
LPfilt	frequency

Example:
```
:AUDFilt:Filter BP,HP_50,4.3KHZ
:AUDFilt:Filter LP,15
:AUDFilt:Filter HP,HP_300
```

:Filter?

Parameters: N/A

Response: <CRD>, <CRD>, <NR2>
BP Filtertype, HPfilt, Lowpass freq or
<CRD>, <NR2>
LP Filtertype, Lowpass freq or
<CRD>, <CRD>
HP Filtertype, HPfilt

Example Response:
`BP,HP_50,4.3` (or)
`LP,15` (or)
`HP,HP_300`

:AUDFilt

:Psoph

Description:	Controls the Psophometric filter of the receiver
Parameters:	<CPD> or <NRf>
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or OFF 1 or ON
Example:	`AUDFilt:Psoph OFF`

:Psoph?

Parameters:	N/A
Response:	<CRD> *Psophometric filter status*
Example Response:	`OFF`

:AUDFilt?

Description:	Queries the status of the Audio Filters. These responses are separated by semi-colons
Parameters:	N/A
Response:	*filt_resp*;<CRD> (see AUDFilt: filter for *filt_resp*)
Example Response:	`BP;HP_300,3.400;OFF`

:AUDIoif

Controls the 600 Ohm Audio interface option

Not used alone

:AUDIoif

:Inputimp

Description: Controls the audio input impedance when the 600 Ohm interface is fitted

Parameters: <CPD> or <NRf>
Input impedance selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or HIGH
1 or OHMS600

Example: `AUDIOIF:INPUT HIGH`

:Inputimp?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `HIGH`

:AUDIoif

:Outputimp

Description: Controls the audio output impedance when the 600 Ohm interface is fitted

Parameters: <CPD> or <NRf>
Output impedance selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or LOW
1 or OHMS600

Example: `AUDIOIF:OUT LOW`

:Outputimp?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `LOW`

:AUDIoif

:Pad

Description:	Controls the audio output attenuator when the 600 Ohm interface is fitted
Parameters:	<CPD> or <NRf> *Output attenuator selection*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or OUT 1 or IN
Example:	`AUDIOIF:PAD IN`

:Pad?

Parameters:	N/A
Response:	<CRD> *Current selection*
Example Response:	`IN`

:AUDIoif?

Description:	Queries the entire status of the audio impedance interface by producing the combined return values of the sub commands of AUDIOIF These responses are separated by semi-colons.
Parameters:	N/A
Response:	<CRD>;<CRD>;<CRD>
Example Response:	`HIGH;LOW;OUT`

:AUDScope

Controls the audio oscilloscope — RX and AF test modes

Not used alone

:AUDScope

:Afrange

Description: Controls the vertical range of the audio oscilloscope

Parameters: <CPD> or <NRf>
Vertical range selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data:
0 or SC_10MV
1 or SC_20MV
2 or SC_50MV
3 or SC_100MV
4 or SC_200MV
5 or SC_500MV
6 or SC_1V
7 or SC_2V
8 or SC_5V
9 or SC_10V
10 or SC_20V

Example: `:AUDS:AFR SC_1V`
Sets audio scope range to 1 V per division

:Afrange?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `SC_10V`

:AUDScope

:Persistence

Description: Selects the trace persistence setting of the audio oscilloscope

Parameters: <CPD> or <NRf>
Trace Persistence Setting

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or LOW
2 or MEDIUM
3 or HIGH
4 or INFINITE

Example: `:AUDS:PERSISTENCE:LOW`

Sets the audio oscilloscope trace persistence to low

:Persistence?

Parameters: N/A

Response: <CRD>
Current audio oscilloscope trace persistence setting

Example Response: `LOW`

:AUDScope

:TBase

Description:	Controls the time base of the audio oscilloscope
Parameters:	<CPD> or <NRf> *Time base selection*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or SC_50US 1 or SC_100US 2 or SC_200US 3 or SC_500US 4 or SC_1MS 5 or SC_2MS 6 or SC_5MS 7 or SC_10MS 8 or SC_20MS 9 or SC_50MS 10 or SC_100MS 11 or SC_200MS 12 or SC_500MS 13 or SC_1S 14 or SC_2S 15 or SC_5S
Example:	`AUDS:TB SC_10MS` Sets audio oscilloscope time base to 10 ms per division

:TBase?

Parameters:	N/A
Response:	<CRD> *Current audio oscilloscope time base*
Example Response:	`SC_2MS` Time base is set to 2 ms per div

:AUDScope

:TRig

Description: Controls the trigger of the audio oscilloscope

Parameters: <CPD> or <NRf>
Trigger selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or SINGLE
1 or REPEAT

Example: `:AUDS:TR REPEAT`
Sets audio oscilloscope to repeat

:TRig?

Parameters: N/A

Response: <CRD>
Current trigger selection

Example Response: `REPEAT`
Audio scope trigger is set to repeat

:AUDScope?

Description: Queries the entire status of the audio oscilloscope by producing the combined return values of the sub commands of AUDSCOPE
These responses are separated by semi-colons

Parameters: N/A

Response: <CRD>;<CRD>;<CRD>;<CRD>

Example Response: `SC_1V;LOW;SC_100MS;REPEAT`
Audio scope settings are:
1 V per div.
Low trace persistence
100 ms per div.
Repeat trigger

:Barchart

Controls the ranges of all the barcharts within the instrument

Not used alone

:Barchart

:AFDistn

Description: Controls the range of the audio distortion barchart

Parameters: <CPD> or <NRf>
Range selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or AUTO
1 or AD_10PC
2 or AD_30PC
3 or AD_100PC

Example: `:BARCH:AFD 1`
Sets barchart range to 10 percent

:AFDistn?

Parameters: N/A

Response: <CRD>
Current range

Example Response: `AUTO`

:Barchart

:AFLevel

Description: Controls the range of the audio level barchart

Parameters: <CPD> or <NRf>
Range selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or AUTO
1 or AL_100MV
2 or AL_300MV
3 or AL_1V
4 or AL_3V
5 or AL_10V
6 or AL_30V
7 or AL_100V

Example: `:BARCH:AFL AL_30V`

:AFLevel?

Parameters: N/A

Response: <CRD>
Current range

Example Response: `AL_300MV`

:Barchart

:AFSInad

Parameter: <CPD> or <NRf>
Range selection

Description: Controls the range of the audio SINAD barchart

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or AUTO
1 or ASI_18DB
2 or ASI_30DB
3 or ASI_50DB

Example: `:BARCH:AFSI ASI_18DB`

:AFSInad?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `AUTO`

:Barchart

:AFSN

Description: Controls the range of the audio signal to noise barchart

Parameters: <CPD> or <NRf>
Range selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or AUTO
1 or ASN_30DB
2 or ASN_50DB
3 or ASN_100DB

Example: `:BARCH:AFSN ASN_100DB`

:AFSN?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `ASN_100DB`

:Barchart

:TXAmmod

Description: Controls the range of the amplitude modulation level barchart

Parameters: <CPD> or <NRf>
Range selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or AUTO
1 or AML_20PC
2 or AML_100PC

Example: `BARCH:TXAM AUTO`

:TXAmmod?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `AML_20PC`

:Barchart

:TXDistn

Description: Controls the range of the mod signal distortion barchart

Parameters: <CPD> or <NRF>
Range selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or AUTO
1 or MD_10PC
2 or MD_30PC
3 or MD_100PC

Example: `:BARCH:TXD MD_10PC`

:TXDistn?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `AUTO`

:Barchart

:TXFmmod

Description: Controls the range of the frequency modulation level barchart

Parameters: <CPD> or <NRf>
Range selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data:
0 or AUTO
1 or FML_1KHZ
2 or FML_3KHZ
3 or FML_10KHZ
4 or FML_30KHZ
5 or FML_100KHZ

Example: `BARCH:TXFM FML_100KHZ`

:TXFmmod?

Parameters: N/A

Response: <CRD>

Example Response: `FML_30KHZ`

:Barchart

:TXPower

Description: Controls the range of the transmitter level (power or voltage) barchart

Parameters: <CPD> or <NRf>
Range selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data:

0 or AUTO	16 or PWR_3KV
1 or PWR_100UV	17 or PWR_10MW
2 or PWR_300UV	18 or PWR_30MW
3 or PWR_1MV	19 or PWR_100MW
4 or PWR_3MV	20 or PWR_300MW
5 or PWR_10MV	21 or PWR_1W
6 or PWR_30MV	22 or PWR_3W
7 or PWR_100MV	23 or PWR_10W
8 or PWR_300MV	24 or PWR_30W
9 or PWR_1V	25 or PWR_100W
10 or PWR_3V	26 or PWR_300W
11 or PWR_10V	27 or PWR_1KW
12 or PWR_30V	28 or PWR_3KW
13 or PWR_100V	29 or PWR_10KW
14 or PWR_300V	30 or PWR_30KW
15 or PWR_1KV	31 or PWR_100KW

Example: `BARCH.TXP PWR_1W`

:TXPower?

Parameters: N/A
Current selection

Response: <CRD>

Example Response: `PWR_1W`

:Barchart

:TXSInad

Description: Controls the range of the modulation SINAD level barchart

Parameters: <CPD> or <NRf>
Range selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or AUTO
1 or MSI_18DB
2 or MSI_30DB
3 or MSI_50DB

Example: `BARCH:TXSI 1`

:TXSInad?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `MSI_18DB`

:Barchart

:TXSN

Description: Controls the range of the modulation signal to noise level barchart

Parameters: <CPD> or <NRf>
Range selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or AUTO
1 or MSN_30DB
2 or MSN_50DB
3 or MSN_100DB

Example: `BARCH:TXSN 1`

:TXSN?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `MSN_30DB`

:Barchart?

Description: Produces the combined return values of the sub commands of BARCHART

These responses are separated by semi-colons.

Parameters: N/A

Response: <CRD>;<CRD>;<CRD>;<CRD>;<CRD>;<CRD>;
<CRD>;<CRD>;<CRD>;<CRD>

Example Response:
```
AD_10PC,AL_300MV,ASI_18DB,ASN_100DB,AML_20PC,
MD_10PC,FML_30KHZ,PWR_1W,MSI_18DB,MSN_30DB
```

:COMmerror?

Description: Returns the last command error generated by the remote parser

Parameters: N/A

Response: <NR1>
Last error

Responses: 0 corresponds to 'No Error'
1 'Illegal * Command'
2 'Parameter not allowed'
3 'Unrecognized mnemonic' The command received was not one recognized by the parser
4 'Mnemonic not unique' An abbreviated command mnemonic was received which was too short to uniquely identify one command. e.g.:AFGEN1:S 1
5 'Write not allowed' A command was received which could only be a query and attempted to set some parameter e.g.:COMMERROR 1
6 'Read not allowed' A command was received which could only be an action and tried to query some state or other
7 'Syntax error' Some part of the command did not meet the parser specification

:COPy

Description: Performs a screen dump to a printer. Action only.

Parameters: N/A

Allowed suffices: N/A

Default suffix: N/A

Example: `COPY`

:DCstones

Controls the settings for DCS tones

Not used alone

:DCstones

:AFlevel

Description: Sets Audio DCS Generator Level

Parameters: <NRf>

Level

Allowed suffices: MV, V, DBM

Default suffix: MV

Example: `:DCS:AFLEVEL 100MV`

Sets level to 100 mV

:AFlevel?

Parameters: N/A

Response: <NR2>

Audio level in mV to 0.1 mV resolution

Example Response: `99.0`

:DCstones

:AMdepth

Description: Sets DCS Generator AM Depth

Parameters: <NRf>

Depth

Allowed suffices: PCT

Default suffix: PCT

Example: `DCS:AMD 30PCT`

:AMdepth?

Parameters: N/A

Response: <NR2>

AM depth (%)

Example Response: `30.0`

:DCstones

:Bitrate

Description: Sets DCS bitrate

Parameters: <NRf>
Frequency

Allowed suffices: KHZ,HZ

Default suffix: KHZ

Example: `DCS:BITRATE 134HZ`

:Bitrate?

Parameters: N/A

Response: <NR2>
Bitrate (kHz)

Example Response: `0.134`

:DCstones

:Code

Description: Sets the DCS code

Parameters: <Octal Program Data>
Code

Allowed suffices: N/A

Default suffix: N/A

Example: `DCS:CODE #Q777`

Sets DCS code to 777. (DCS codes are octal. #Q is the 488.2 prefix for octal).

:Code?

Parameters: N/A

Response: <OCTAL PROGRAM DATA>
DCS code

Example Response: `#Q777`

:DCstones

:Fmdevn

Description:	Sets DCS Generator FM Deviation
Parameters:	<NRf> *Deviation*
Allowed suffices:	KHZ, HZ
Default suffix:	HZ
Example:	`DCS:FM 2.4KHZ`

:Fmdevn?

Parameters:	N/A
Response:	<NR2> Deviation (Hz)
Example Response:	`2400`

:DCstones

:REadcode?

Description:	Returns the decoded DCS tones with information about preferred codes. The returned string contains each code and a parameter either N or P, depending on the code being PREFERRED or NON-PREFERRED, up to a maximum of 7 codes. Query only.
Parameters:	N/A
Response:	<ARB ASCII STRING DATA>
Example Response:	`#Q231,N,#Q504,N,#Q631,P,#Q636,N,#Q745,N`

:DCstones

:RXpolarity

Description:	Sets DCS receive polarity
Parameters:	<CPD> or <NRf> *Polarity*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or NORMAL 1 or INVERT
Example:	`DCS:RXPOL INVERT`

:RXpolarity?

Parameters:	N/A
Response:	<CRD> Polarity
Example Response:	`INVERT`

:DCstones

:Status

Description:	Sets DCS status
Parameters:	<CPD> or <NRf> *Status*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or OFF 1 or ON
Example:	`DCS:STAT ON`

:Status?

Parameters:	N/A
Response:	<CRD> Status
Example Response:	`ON`

:DCstones

:Txpolarity

Description:	Sets DCS transmit polarity
Parameters:	<CPD> or <NRf> *Polarity*
Allowed suffices:	N/A
Valid Data:	0 or NORMAL 1 or INVERT
Default suffix:	N/A
Example:	`DCS:TXPOL  NORMAL`

:Txpolarity?

Parameters:	N/A
Response:	<CRD> *Polarity*
Example Response:	`NORMAL`

:DCstones?

Description: Queries the status of DCS tones by producing the combined return values of the sub commands of DCSTONES

These responses are separated by semi-colons.

Parameters: N/A

Response: <NR2>;<NR2>;<NR2>;<OCTAL PROGRAM DATA>; <NR2>;<ARB ASCII STRING DATA>;<CRD>;<CRD>; <CRD>;

Example Response: `99.0;30.0;0.134;#Q777;2400;#Q231,N,#Q504,N,#Q631,P,#Q636,N,#Q745,N;INVERT;ON;NORMAL`

Settings are:

AF level 99 mV
AM depth 30%
Bitrate 134 Hz
Encode code is 777 (Octal)
FM deviation is 2.4 kHz
Decoded Octal codes are 231, 504, 631 (preferred), 636, 745
Rx polarity is inverted
Status is On
Tx polarity is normal.

:DEModtype

Description: Sets the type of demodulation used on the received signal

Parameters: <CPD> or <NRf>
Demod selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or AM
1 or FM
2 or SSB

Example: `DEMOD FM`

:DEModtype?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `FM`

:DEVerror?

Description: Returns the last device error generated by the instrument

Parameters: N/A

Response: <NR1>
Last error

Responses: 0 corresponds to 'No Error'
1 corresponds to 'Value out of range'
Some parameter received with a command was too large or small for the instrument to be able to set.
2 corresponds to ‘Wrong mode for measurement’
3 corresponds to ‘Wrong setup for measurement’
4 corresponds to ‘Cannot change item’
5 corresponds to ‘Wrong setup for command’
6 corresponds to ‘Option not fitted’
7 corresponds to ‘Systems test in progress’
8 corresponds to ‘Store empty’
9 corresponds to ‘No memory card present’
10 corresponds to ‘Card not formatted’
11 corresponds to ‘No card interface fitted’
12 corresponds to ‘File not found’
13 corresponds to ‘Not a settings store for recall’

:DTmftones

Controls the settings for DTMF tones

Not used alone

:DTmftones

:DEcodereset

Description: Resets the DTMF decoder and clears the received sequence. This is an action only.

Parameters: N/A

Allowed suffices: N/A

Default suffix: N/A

Example: `DTMF:DECODERESET`

:DTmftones

:DUration

Description: Sets the DTMF tone duration

Parameters: <NRf>
Duration

Allowed suffices: MS,S

Default suffix: MS

Example: `DTMF:DUR 200`
Sets the tone duration to 200 ms

:DUration?

Parameters: N/A

Response: <NR1>
Tone duration (ms)

Example Response: `200`

:DTmftones

:Freqshift

Description: Sets the DTMF tone frequency shift in percent

Parameters: <NRf>
Frequency shift

Allowed suffices: PCT

Default suffix: PCT

Example: `DTMF:FREQSHIFT 2PCT`
Adjust all DTMF frequencies by plus 2%.

:Freqshift?

Parameters: N/A

Response: <NR2>
Frequency shift (%)

Example Response: `2.0`

:DTmftones

:HIAFlevel

Description: Sets DTMF Generator High Tone Audio Level

Parameters: <NRf>

Level

Allowed suffices: MV, V, DBM

Default suffix: MV

Example: `:DTMF:HIAFLEVEL 100MV`

Sets level to 100mV

:HIAFlevel?

Parameters: N/A

Response: <NR2>

Audio level in mV to 0.1 mV resolution

Example Response: `99.0`

:DTmftones

:HIAMdepth

Description: Sets DTMF Generator High Tone AM Depth

Parameters: <NRf>

Depth (%).

Allowed suffices: PCT

Default suffix: PCT

Example: `DTMF:HIAMD 30PCT`

:HIAMdepth?

Parameters: N/A

Response: <NR2>

AM depth (%) To 0.1%

Example Response: `30.0`

:DTmftones

:HIFmdevn

Description: Sets DTMF Generator High Tone FM Deviation

Parameters: <NRf>

Deviation

Allowed suffices: KHZ, HZ

Default suffix: HZ

Example: `DTMF:HIFM 2.4KHZ`

:HIFmdevn?

Parameters: N/A

Response: <NR1>

Deviation (kHz)

Example Response: `2400`

:DTmftones

:LOAFlevel

Description: Sets DTMF Generator Low Tone Audio Level

Parameters: <NRf>

Level

Allowed suffices: MV, V, DBM

Default suffix: MV

Example: `:DTMF:LOAFLEVEL 100MV`

Sets level to 100 mV

:LOAFlevel?

Parameters: N/A

Response: <NR2>

Audio level in mV to 0.1 mV resolution

Example Response: `99.0`

:DTmftones

:LOAMdepth

Description:	Sets DTMF Generator Low Tone AM Depth
Parameters:	<NRf> *Depth*
Allowed suffices:	PCT
Default suffix:	PCT
Example:	`DTMF:LOAMD 30PCT`

:LOAMdepth?

Parameters:	N/A
Response:	<NR2> *AM depth (%)*
Example Response:	`30.0`

:DTmftones

:LOFmdevn

Description:	Sets DTMF Generator Low Tone FM Deviation
Parameters:	<NRf> *Deviation*
Allowed suffices:	KHZ, HZ
Default suffix:	HZ
Example:	`DTMF:LOFM 2.4KHZ`

:LOFmdevn?

Parameters:	N/A
Response:	<NR2> *Deviation (Hz)*
Example Response:	`2400`

:DTmftones

:Mode

Description: Sets the mode of generation in DTMF

Parameters: <CPD> or <NRf>
Mode

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or STOP
1 or BURST
2 or CONT

Example: `DTMF:MODE BURST`

:Mode?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `STOP`

:DTmftones

:Pause

Description: Sets the DTMF Pause duration

Parameters: <NRf>
Pause duration

Allowed suffices: MS,S

Default suffix: MS

Example: `DTMF:PAUSE 100`
Sets the pause duration to 100 ms

:Pause?

Parameters: N/A

Response: <NR1>
Pause duration (ms)

Example Response: `100`

:DTmftones

:READSequence?

Description: Returns the contents of the DTMF decode register. This is query only.

Parameters: N/A

Response: <STRING RESPONSE DATA>
DTMF received sequence

Example Response: `"123456789ABCD*#"`

:DTmftones

:READTone?

Description: Returns the statistics of the nth tone in the received sequence

Parameters: <NRf>
Tone number

Response: <ARB ASCII DATA>,<NR2>,<NR2>,<NR2>,<NR2>,<NR1>
Tone, Low tone frequency (kHz), Low tone error (%), High tone frequency (kHz), High tone error (%), Tone duration (ms)

Example: `DTMF:READTONE? 5`

Example Response: `5,0.0.7698,0.0,1.3363,0.0,50`

:DTmftones

:Sequence

Description: Sets DTMF encode sequence

Parameters: <STRING PROGRAM DATA>
Sequence

Allowed suffices: N/A

Default suffix: N/A

Example: `DTMF:SEQUENCE "01438742200"`

:Sequence?

Parameters: N/A

Response: <STRING RESPONSE DATA>
Current selection

Example Response: `"01438742200"`

:DTmftones?

Description: Queries the status of DTMF tones by producing the combined return values of the sub commands of DTMFTONES

These responses are separated by semi-colons

Parameters: N/A

Response: <NR1>;<NR2>;<NR2>;<NR2>;<NR1>;<NR2>;<NR2>;<NR1>;
<CRD>;<NR1>;<STRING RESPONSE DATA>;
<STRING RESPONSE DATA>

Example Response:

```
200;2.0;99.0;30.0;2400;99.0;30.0;2400;STOP;10
0;"123456789ABCD*#";"01438742200"
```

DTMF settings are:

Duration is 200 ms
Frequency shift is 2%
AF level for high tone is 99 mV
High tone AM depth is 30%(if AM set)
High tone FM deviation is 2.4 kHz (if FM set)
Low tone AF level is 99 mV
Low tone AM depth is 30% (if AM set)
Low tone FM deviation is 2.4 kHz (if FM set)
DTMF tone mode stopped
Pause duration is 100 ms
the decoded sequence is 123456789ABCD*#
the sequence to encode is 01438742200.

:Execerror?

Description: Returns the type of the last error generated by the execution control routine

Parameters: N/A

Response: <NR1>

Last error

Responses: 0 corresponds to 'No Error'

1 corresponds to 'Num option data out of range'

A command which takes <CPD> or <NRf> in a one of few form has been sent with a number larger than that recognized as the highest possible. e.g. :AUDSCOPE:TRIG 5

2 corresponds to 'Excess data'

More data was received with the command than was expected e.g. :AFGEN1:FR 10.000,15.000

3 corresponds to 'Insufficient data'

The command had fewer data fields than expected

4 corresponds to 'Data required'

No data came with the command when some was definitely required e.g. :AFGEN1:FREQ

5 corresponds to 'Unrecognized text option'

<CPD> was received which did not tally with the allowed character data strings for that data field e.g.:AUDSCOPE:AFRANGE GARBLE

6 corresponds to 'Alpha text not unique'

The abbreviated <CPD> received was too short to be uniquely recognized e.g. :AUDSCOPE:AFRANGE SC_2

7 corresponds to 'Unrecognized suffix'

The suffix received with a particular data field was not one allowed for that command e.g. :AFGEN1:FREQ 10.000DBM

8 corresponds to 'Suffix not allowed'

A suffix was sent with a numeric data field when one was not allowed e.g. :AUDSCOPE:AFRANGE 5KHZ

:Genswitch

Description: Controls the routing of the RF output signal

Parameters: <CPD> or <NRf>
Output selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or GEN_N
1 or GEN_BNC

Example: `GENSW GEN_BNC`

:Genswitch?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `GEN_BNC`

:Ilsgen

Controls the ILS signal generator

Not used alone

:Ilsgen

:DDm

Description: Sets the difference in depth of modulation of the ILS frequencies

Parameters: <NRf>
DDM

Allowed suffices: PCT

Default suffix: PCT

Example: `ILSGEN:DDM 5PCT`

:DDm?

Parameters: N/A

Response: NR2
DDM value (%)

Example Response: `5.0`

:Ilsgen

:DIr

Description: Sets the direction of flight for ILS — left or right for localizer, up or down for glideslope

Parameters: <CPD> or <NRf>
Direction

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or DOWN_RIGHT
1 or UP_LEFT

Example: `ILS:DIR 0`

:DIr?

Parameters: N/A

Response: <CRD>
Direction

Example Response: `DOWN_RIGHT`

:Ilsgen

:Idepth

Description: Sets the modulation depth of the ident

Parameters: <NRf>

Allowed suffices: PCT

Default suffix: PCT

Example: `ILS:IDEPTH 10PCT`

:Idepth?

Parameters: N/A

Response: <NR2>
Ident depth to 0.1%

Example Response: `10.0`

:Ilsgen

:Mode

Description: Sets the mode of operation in ILS between localizer, glideslope and localizer with ident

Parameters: <CPD> or <NRf>
mode

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or LOCALISER
1 or GLIDESLOPE
2 or IDENT

Example: `ILS:MODE GLIDE`

:Mode?

Parameters: N/A

Response: <CRD>
current mode

Example Response: `GLIDESLOPE`

:Ilsgen

:RFFreq

Description: Sets RF frequency (localizer or glideslope, as selected)

Parameters: <NRf>
Frequency

Allowed suffices: MHZ, KHZ, HZ

Default suffix: MHZ

Example: `ILS:RFFREQ 108.1`

:RFFreq?

Parameters: N/A

Response: <NR2>
Frequency (MHz)

Example Response: `108.100000`

:Ilsgen

:RFLevel

Description: Sets the RF output level in ILS mode

Parameters: <NRf>
Level

Allowed suffices: DBM,DBUV,UV,MV

Default suffix: DBM

Example: `ILS:RFLEVEL –80DBM`

:RFLevel?

Parameters: N/A

Response: <NR2>
Level (dBm)

Example Response: `–80.0`

:Ilsgen

:RFOut

Description: Sets the RF output in ILS mode

Parameters: <CPD> or <NRf>
Output selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or GEN_N
1 or GEN_BNC

Example: `ILSGEN:RFO GEN_N`

:RFOut?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `GEN_BNC`

:Ilsgen

:SDm

Description: Sets the specific depth of modulation of the ILS frequencies

Parameters: <NRf>

SDM

Allowed suffices: PCT

Default suffix: PCT

Example: `ILS:SDM 40`

:SDm?

Parameters: N/A

Response: <NR2>

SDM (%) to 0.1%

Example Response: `40.0`

:Ilsgen

:SUppress

Description: Controls which of the ILS tones are suppressed

Parameters: <CPD> or <NRf>

Tone suppression selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or SUPP_NONE
1 or SUPP_90
2 or SUPP_150

Example: `ILS:SUPPRESS 0`

:SUppress?

Parameters: N/A

Response: <CRD>

Current selection

Example Response: `SUPP_NONE`

:Ilsgen?

Description: Produces the combined return values of the sub commands of ILSGEN

These responses are separated by semi-colons

Parameters: N/A

Response: <NR2>;<CRD>;<NR2>;<CRD>;<NR2>;
<NR2>;<CRD>;<NR2>;<CRD>

Example Response:
```
5.0;DOWN_RIGHT;10.0;GLIDESLOPE;
108.100000;-80.0;GEN_BNC;
40.0;SUPP_NONE
```

:MEASCycl

Description: Controls whether or not the measure cycle within the instrument is running

Parameters: <CPD> or <NRf>
Measure cycle status

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or ON

Example: `MEASCYCL OFF`

:MEASCycl?

Parameters: N/A

Response: <CRD>
Current status

Example Response: `OFF`

:MEASUre

:AFFreq?

Description: Returns current measurement of audio frequency in kHz to a resolution of 0.1 Hz

Parameters: N/A

Response: <NR2>
Frequency (kHz)

Example Response: `1.0000`

:MEASUre

:AFLevel?

Description: Returns measured value of audio input level in currently selected units

Parameters: N/A

Response: <NR2>
Level in selected units

Example Response: `101.1`

:MEASUre

:ALevel?

Description: Returns the current value of positive and negative AM Depth. (See also :MEASUre:AMdepth? that returns the current average value.)

Parameters: N/A

Response: <NR2>,<NR2>
Pos depth (%), Neg depth (%)

Example Response: `29.5,33.5`

:MEASUre

:AMdepth?

Description: Returns measured value of transmitter amplitude modulation depth in percent to a resolution of 0.1 percent. (See also :MEASUre:ALevel? that returns the positive and negative depths.)

Parameters: N/A

Response: <NR2>
Depth (%)

Example Response: `31.5`

:MEASUre

:FLevel?

Description: Returns the current value of positive and negative FM Deviation. (See also :MEASUre:FMdevn? that returns the current average value.)

Parameters:

Response: <NR2>,<NR2>
Pos deviation (Hz), Neg deviation (Hz)

Example Response: `25100,24950`

:MEASUre

:FMdevn?

Description: Returns measured value of transmitter deviation in Hz. (See also :MEASUre:FLevel? that returns the positive and negative deviations.)

Parameters: N/A

Response: <NR2>
Deviation (Hz)

Example Response: `25025`

:MEASUre

:FWdpwr?

Description: Returns current value of reading from the directional power accessory in units of dBm

Parameters: N/A

Response: <NR2>

Forward power (dBm)

Example Response: `48.2`

:MEASUre

:HARM2?

Description: Returns current value of reading in units of dBc

Parameters: N/A

Response: <NR2>

Second harmonic level (dBc) to 0.1 dBc

Example Response: `-50.3`

:MEASUre

:HARM3?

Description: Returns current value of reading in units of dBc

Parameters: N/A

Response: <NR2>

Third harmonic level (dBc) to 0.1 dBc

Example Response: `-50.3`

:MEASUre

:HARM4?

Description: Returns current value of reading in units of dBc

Parameters: N/A

Response: <NR2>

Fourth harmonic level (dBc) to 0.1 dBc

Example Response: `-50.3`

:MEASUre

:HARM5?

Description: Returns current value of reading in units of dBc

Parameters: N/A

Response: <NR2>
Fifth harmonic level (dBc) to 0.1 dBc

Example Response: `–50.3`

:MEASUre

:MKr1?

Description: Returns value of level at marker on spectrum analyzer or transient analyzer in units of dBm to a resolution of 0.1 dBm

Parameters: N/A

Response: <NR2>
Level (dBm)

Example Response: `10.1`

:MEASUre

:MOdfreq?

Description: Returns current value of modulation frequency in kHz to a resolution of 0.1 Hz

Parameters: N/A

Response: <NR2>
Frequency (kHz)

Example Response: `0.9999`

:MEASUre

:Occbw?

Description: Returns current value of Occupied Bandwith in kHz to a resolution of 0.1 kHz

Parameters: N/A

Response: <NR2>
Frequency (kHz)

Example Response: `25.6`

:MEASUre

:REvpwr?

Description: Returns current value of reading from the directional power accessory in units of dBm

Parameters: N/A

Response: <NR2>

Reverse power (dBm)

Example Response: `40.5`

:MEASUre

:RXDistn?

Description: Returns measured value of audio input distortion in percent to a resolution of 0.1 percent

Parameters: N/A

Response: <NR2>

Distortion (%) to 0.1%

Example Response: `3.2`

:MEASUre

:RXSInad?

Description: Returns measured value of audio input distortion in dB to a resolution of 0.1 dB

Parameters: N/A

Response: <NR2>

SINAD (dB)

Example Response: `34.4`

:MEASUre

:RXSN?

Description: Returns measured value of audio signal to noise in dB to a resolution of 0.1 dB

Parameters: N/A

Response: <NR2>

S/N (dB) to 0.1 dBm

Example Response: `28.2`

:MEASUre

:Satrace?

Description: Waits for the current Spectrum Analyzer, Occupied Bandwidth, or Transient Analysis sweep to complete, and then returns a set of bytes representing the maximum value, in pixels, of each of the 249 columns that make up the trace. The bytes are returned in order from left to right across the screen, with 0 corresponding to the bottom of the screen, and 160 corresponding to the top of a monochrome display or 192 for a color display.
Note that, depending on the sweep time, it may be required to extend the remote control application's query timeout period.

Parameters: N/A

Response: <DEFINITE LENGTH ARBITRARY RESPONSE DATA><nl>

This takes the form of:

- A "#" character, followed by:
- a single-character digit (n), followed by:
- an n-character integer (m) representing the number of bytes of pixel data that follow, followed by:
- m bytes of pixel data, followed by:
- a <newline> character.

Example Response: `#3249<byte1><byte2>……<byte249><nl>`

:MEASUre

:TXDistn?

Description: Returns current value of transmitter distortion in percent to a resolution of 0.1 %

Parameters: N/A

Response: <NR2>

Distortion (%) to 0.1%

Example Response: `2.0`

:MEASUre

:TXFreq?

Description: Returns current value of reading from RF counter in MHz to a resolution of 1 Hz

Parameters: N/A

Response: <NR2>

Frequency(MHz)

Example Response: `101.537123`

:MEASUre

:TXLevel?

Description: Returns current value of reading from RF power meter in currently selected units

Parameters: N/A

Response: <NR2>

Level (dBm) to 0.1 dBm

Example Response: `31.2`

:MEASUre

:TXOffset?

Description: Returns current reading of offset from the currently set receiver frequency in kHz to a resolution of 1 Hz

Parameters: N/A

Response: <NR2>

Frequency (kHz)

Example Response: `–1.300`

:MEASUre

:TXSInad?

Description: Returns current value of transmitter SINAD in dB to a resolution of 0.1 dB

Parameters: N/A

Response: <NR2>

SINAD (dB) to 0.1 dBm

Example Response: `26.0`

:MEASUre

:TXSN?

Description: Returns current value of transmitter signal-noise ratio in dB to a resolution of 0.1 dB

Parameters: N/A

Response: <NR2>

S/N (dB) to 0.1 dBm

Example Response: `20.1`

:MEASUre

:Vswr?

Description: Returns current value of reading from Directional power accessory

Parameters: N/A

Response: <NR2>

Example Response: `2.11`

:MKrbcn

Controls the marker beacon generator

Not used alone

:MKrbcn

:Depth

Description: Sets the modulation depth in Marker beacon mode

Parameters: <NRf>

depth

Allowed suffices: PCT

Default suffix: PCT

Example: `MKRBCN:DEPTH 30.0`

:Depth?

Parameters: N/A

Response: <NR2>

Depth (%)

Example Response: `30.0`

:MKrbcn

:Freq

Description:	Sets modulation freq in Marker beacon mode
Parameters:	<NRf> *Frequency*
Allowed suffices:	KHZ, HZ
Default suffix:	KHZ
Example:	`MKRBCN:FREQ 1.3`

:Freq?

Parameters:	N/A
Response:	<NR2> *Frequency (kHz)*
Example Response:	`1.3000`

:MKrbcn

:RFFreq

Description:	Sets RF frequency in marker beacon mode
Parameters:	<NRf> *Frequency*
Allowed suffices:	MHZ, KHZ, HZ
Default suffix:	MHZ
Example:	`MKR:RFFREQ 108.0`

:RFFreq?

Parameters:	N/A
Response:	<NR2> *Frequency (MHz)*
Example Response:	`108.000000`

:MKrbcn

:RFLevel

Description:	Sets the RF output level in marker beacon mode
Parameters:	<NRf> *Level*
Allowed suffices:	DBM,DBUV,UV,MV
Default suffix:	DBM
Example:	`MKR:RFLEVEL –80DBM`

:RFLevel?

Parameters:	N/A
Response:	<NR2> *Level (dBm)*
Example Response:	`–80.0`

:MKrbcn

:RFOut

Description:	Sets the RF output in Marker beacon mode
Parameters:	<CPD> or <NRf> *Output selection*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or GEN_N 1 or GEN_BNC
Example:	`MKRBCN:RFOUT 0`

:RFOut?

Parameters:	N/A
Response:	<CRD> *Output selection*
Example Response:	`GEN_N`

:MKrbcn?

Description:	Produces the combined return values of the sub commands of MKRBCN These responses are separated by semi-colons
Parameters:	N/A
Response:	<NR2>;<NR2>;<NR2>;<NR2>;<CRD>
Example Response:	`30.0;1.3000;108.000000;` `–80.0;GEN_N`

:MODFilt

:Filter

Description:	Sets the Audio Filter in the Demod path
Parameters:	<CPD>, <CPD>, <NRf> BP Filtertype, HPfilt, Lowpass freq or <CPD>, <NRf> LP Filtertype, Lowpass freq or <CPD>, <CPD> HP Filtertype, HPfilt
Allowed suffices:	NRf in Hz, | kHz
Default suffix:	kHz
Valid Data:	Filter type — BP or HP or LP HPfilt — HP_50 or HP_300 LPfilt — frequency

Example:	`:MODFilt:Filter BP,HP_50,4.3KHZ`
	`:MODFilt:Filter LP,15`
	`:MODFilt:Filter HP,HP_300`

:Filter?

Parameters:	N/A
Response:	<CRD>, <CRD>, <NR2> BP Filtertype, HPfilt, Lowpass freq or <CRD>, <NR2> LP Filtertype, Lowpass freq or <CRD>, <CRD> HP Filtertype, HPfilt
Example Response:	`BP,HP_50,4.3` (or) `LP,15` (or) `HP,HP_300`

:MODFilt

:Psoph

Description:	Controls the Psophometric filter in the Demod path
Parameters:	<CPD> or <NRf>
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or OFF 1 or ON
Example:	`MODFilt:Psoph OFF`

:Psoph?

Parameters:	N/A
Response:	<CRD> *Psophometric filter status*
Example Response:	`OFF`

:MODFilt?

Description:	Queries the status of the Mod Filters. These responses are separated by semi-colons
Parameters:	N/A
Response:	*filt_resp*;<CRD> (see MODFilt: filter for *filt_resp)*
Example Response:	`BP;HP_300,3.400;OFF`

:MODGEN*n*

Controls modulation generator *n* where *n* = 1 or 2

Not used alone

:MODGEN*n*

:Amdepth

Description: Sets Modulation Generator *n* AM Depth where *n* = 1 or 2

Parameters: <NRf>
depth

Allowed suffices: PCT

Default suffix: PCT

Example: `MODGEN1:AMD 30PCT`

:Amdepth?

Parameters: N/A

Response: <NR2>
AM depth (%)

Example Response: `30.0`

:MODGEN*n*

:FMdevn

Description: Sets Modulation Generator *n* FM Deviation where *n* = 1 or 2

Parameters: <NRf>
Deviation

Allowed suffices: KHZ, HZ

Default suffix: KHZ

Example: `MODGEN1:FM 2.4KHZ`

:FMdevn?

Parameters: N/A

Response: <NR1>
Deviation (Hz)

Example Response: `2400`

:MODGEN*n*

:FReq

Description: Sets modulation generator *n* frequency where *n* = 1 or 2

Parameters: <NRf>
Frequency (kHz)

Allowed suffices: KHZ,HZ

Default suffix: KHZ

Example: `MODGEN1:FR 2KHZ`

:FReq?

Parameters: N/A

Response: <NR2>
Frequency (kHz)

Example Response: `2.0000`

:MODGEN*n*

:Level

Description: Sets modulation generator *n* level where *n* = 1 or 2. This is an action only.

Parameters: <NRf>

Allowed suffices: HZ, KHZ, PCT

Default suffix: HZ (if FM set)
PCT (to 0.1%, if AM set)

Example: `MODGEN1:LEVEL 5KHZ`

:MODGEN*n*

:SHape

Description: Sets Modulation Generator *n* Shape where *n* = 1 or 2

Parameters: <CPD> or <NRf>
shape selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or SINE
1 or SQUARE

Example: `MODGEN1:SHAPE 0`

:SHape?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `SINE`

:MODGEN*n*

:STatus

Description:	Sets Modulation Generator *n* Status where *n* = 1 or 2
Parameters:	<CPD> or <NRf> *Status selection*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or OFF 1 or ON
Example:	`MODGEN1:STAT ON`

:STatus?

Parameters:	N/A
Response:	<CRD> *Current status*
Example Response:	`ON`

:MODGEN*n*?

Description:	Queries the status of Modulation Generator *n* where *n* = 1 or 2. Produces the combined return values of the sub commands of MODGEN1 (MODGEN2). These responses are separated by semi-colons.
Parameters:	N/A
Response:	<NR2>;<NR2>;<NR2>;<CRD>;<CRD>
Example Response:	`30.0;2.400;2.000;SINE;OFF`

:MODGENLock

Description: Locks the modulation generator levels to the same value by locking MODGEN 2 level to MODGEN 1 level

Parameters: <CPD> or <NRf>
Modulation generator locking

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or ON

Example: `MODGENLOCK OFF`

Removes the locking between MODGEN 1 and MODGEN 2

:MODGENLock?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `OFF`

:MODGENX

Controls the external modulation source

Not used alone

:MODGENX

:Amdepth

Description: Sets External Modulation Generator AM Depth

Parameters: <NRf>
AM depth

Allowed suffices: PCT

Default suffix: PCT

Example: `MODGENX:AM 10PCT`

:Amdepth?

Parameters: N/A

Response: <NR2>
AM depth (%)

Example Response: `10.0`

:MODGENX

:Coupling

Description: Sets External Modulation coupling

Parameters: <CPD> or <NRf>
Coupling selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or AC
1 or DC

Example: `MODGENX:COUPLING DC`

:Coupling?

Parameters: N/A

Response: <CRD>
Selected ext mod coupling

Example Response: `AC`

:MODGENX

:Fmdevn

Description: Sets External Modulation Generator FM Deviation

Parameters: <NRf>
Deviation

Allowed suffices: KHZ, HZ

Default suffix: KHZ

Example: `MODGENX:FM 1.0`

:Fmdevn?

Parameters: N/A

Response: <NR2>
Deviation (kHz)

Example Response: `1.000`

:MODGENX

:Level

Description: Sets External Modulation Generator level. This is an action only.

Parameters: <NRf>

Allowed suffices: HZ, KHZ, PCT

Default suffix: HZ (if FM set)
PCT (to 0.1%, if AM set)

Example: `MODGENX:LEVEL 5KHZ`

:MODGENX

:SOurce

Description:	Sets External Modulation source
Parameters:	<CPD> or <NRf> *Source selection*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or EXT_MOD_IP 1 or MICROPHONE
Example:	`MODGENX:SOURCE 0`

:SOurce?

Parameters:	N/A
Response:	<CRD> *Selected source*
Example Response:	`EXT_MOD_IP`

:MODGENX

:STatus

Description:	Sets External Modulation Generator Status
Parameters:	<CPD> or <NRf> *Status selection*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or OFF 1 or ON
Example:	`MODGENX:STAT ON`

:STatus?

Parameters:	N/A
Response:	<CRD> *Current status*
Example Response:	`ON`

:MODGENX?

Description:	Queries the status of the External Modulation Generator. Produces the combined return values of the sub commands of MODGENX. These responses are separated by semi-colons.
Parameters:	N/A
Response:	<NR2>;<CRD>;<NR2>;<CRD>;<CRD>
Example Response:	`10.0;AC;1.000;EXT_MOD_IP;ON`

:MODScope

Controls the modulation oscilloscope — TX test mode

Not used alone

:MODScope

:Amrange

Description: Controls the range of Y Sensitivity of the oscilloscope (Tx AM test mode)

Parameters: <CPD> or <NRf>
Range selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or SC_5PC
1 or SC_10PC
2 or SC_20PC

Example: `MODSC:AMR SC_10PC`

:Amrange?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `SC_5PC`

:MODScope

:Fmrange

Description: Controls the range of Y Sensitivity of the oscilloscope(Tx FM test mode)

Parameters: <CPD> or <NRf>
Range selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or SC_200HZ
1 or SC_500HZ
2 or SC_1KHZ
3 or SC_2KHZ
4 or SC_5KHZ
5 or SC_10KHZ
6 or SC_25KHZ

Example: `MODSC:FMR SC_1KHZ`

:Fmrange?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `SC_1KHZ`

:MODScope

:Persistence

Description: Selects the trace persistence setting of the modulation oscilloscope

Parameters: <CPD> or <NRf>
Trace Persistence Setting

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or LOW
2 or MEDIUM
3 or HIGH
4 or INFINITE

Example: `:MODS:PERSISTENCE:LOW`
Sets the trace persistence for the modulation oscilloscope to low

:Persistence?

Parameters: N/A

Response: <CRD>
Current modulation oscilloscope trace persistence setting

Example Response: `LOW`

:MODScope

:TBase

Description: Controls the timebase of the oscilloscope in Tx test mode

Parameters: <CPD> or <NRf>
Range selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data:
0 or SC_50US
1 or SC_100US
2 or SC_200US
3 or SC_500US
4 or SC_1MS
5 or SC_2MS
6 or SC_5MS
7 or SC_10MS
8 or SC_20MS
9 or SC_50MS
10 or SC_100MS
11 or SC_200MS
12 or SC_500MS
13 or SC_1S
14 or SC_2S
15 or SC_5S

Example: `MODSC:TBASE 4`

:TBase?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `SC_1MS`

:MODScope

:TRig

Description: Controls the trigger of the oscilloscope in TX test mode

Parameters: <CPD> or <NRf>
Trigger selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or SINGLE
1 or REPEAT

Example: `MODSC:TRIG REPEAT`

:TRig?

Parameters: N/A

Response: <CRD>
Trigger selection

Example Response: `REPEAT`

:MODScope?

Description: Queries the entire status of the modulation oscilloscope by producing the combined return values of the sub commands of MODSCOPE

These responses are separated by semi-colons

Parameters: N/A

Response: <CRD>;<CRD>;<CRD>;<CRD>;<CRD>

Example Response: `SC_5PC;SC_1KHZ;LOW;SC_100MS;REPEAT`

Modulation scope settings are:

AM sensitivity 5%
FM sensitivity 1 kHz
Low trace persistence
100 ms per div
Repeat trigger

:MODType

Description: Sets the type of modulation used on the signal generator

Parameters: <CPD> or <NRf>
Modulation type selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or AM
1 or FM

Example: `MODT FM`

:MODType?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `FM`

:Occbw

:Ratio

Description: Sets the measurement ratio for the Occupied Bandwidth facility.

Parameters: <NRf>
Measurement ratio setting

Allowed suffices: PCT

Default suffix: PCT

Valid Data:

Example: `95.0`

:Ratio?

Parameters: N/A

Response: <NR2>
Current ratio setting

Example Response: `95.0`

:POcsagtones

Controls the settings for POCSAG tones

Not used alone

:POcsagtones

:AFlevel

Description: Sets Audio POCSAG Generator Level

Parameters: <NRf>
Level

Allowed suffices: MV, V, DBM

Default suffix: MV

Example: `:POCSAG:AFLEVEL 100MV`
Sets level to 100 mV

:AFlevel?

Parameters: N/A

Response: <NR2>
Audio level in mV to 0.1 mV resolution

Example Response: `99.0`

:POcsagtones

:ALert

Description: Sets the alert message type to be sent to the POCSAG device under test

Parameters: <CPD> or <NRf>
Alert type

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or NUMERIC_ALERT
1 or ALERT_ONLY_TYPE_1
2 or ALERT_ONLY_TYPE_2
3 or TEXT_ALERT

Example: `POCSAG:ALERT TEXT_ALERT`

:ALert?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `TEXT_ALERT`

:POcsagtones

:Bitrate

Description: Sets POCSAG bitrate

Parameters: <NRf>
Frequency

Allowed suffices: KHZ,HZ

Default suffix: KHZ

Example: `POCSAG:BITRATE 2.4KHZ`

:Bitrate?

Parameters: N/A

Response: <NR2>
Bitrate (kHz)

Example Response: `2.4000`

:POcsagtones

:Callpager

Description: Causes the generation of a POCSAG signal sequence with the current settings. This is an action only.

Parameters: N/A

Allowed suffices: N/A

Default suffix: N/A

Example: `POCSAG:CALL`

:POcsagtones

:DECODEAs

Description: Sets whether data received with a POCSAG message is decoded as a Numeric or Alphanumeric message

Parameters: <CPD> or <NRf>
Decode format

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or NUMERIC
1 or ALPHANUMERIC

Example: `POCSAG:DECODEAS ALPHA`

:DECODEAs?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `ALPHANUMERIC`

:POcsagtones

:DECODEOn

Description: Sets the condition for POCSAG decoding to begin

Parameters: <CPD> or <NRf>
Decode condition

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or ALL
1 or RIC
2 or MESSAGE

Example: `POCSAG:DECODEON RIC`

Start POCSAG decoder on receipt of a particular Radio Identity Code

:DECODEOn?

Parameters: N/A

Response: <CRD>
Current Selection

Example Response: `RIC`

:POcsagtones

:DECODEReset

Description: Clears and resets the POCSAG decoder. Action only.

Parameters: N/A

Allowed suffices: N/A

Default suffix: N/A

Example: `POCSAG:DECODERESET`

:POcsagtones

:Fmdevn

Description: Sets POCSAG Generator FM Deviation

Parameters: <NRf>
Deviation

Allowed suffices: KHZ, HZ

Default suffix: HZ

Example: `POCSAG:FM 2.4KHZ`

:Fmdevn?

Parameters: N/A

Response: <NR1>
Deviation (Hz)

Example Response: `2400`

:POcsagtones

:Message

Description: Sets the message sent in POCSAG

Parameters: <CPD> or <NRf>
Message

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or NULL_MESS
1 or NUMERIC_MESS_1
2 or NUMERIC_MESS_2
3 or TEXT_MESS_1
4 or TEXT_MESS_2
5 or TEXT_MESS_3
6 or TEXT_MESS_4

Example: `POCSAG:MESSAGE TEXT_MESS_1`

:Message?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `TEXT_MESS_1`

:POcsagtones

:Polarity

Description: Sets the transmit polarity of the POCSAG message

Parameters: <CPD> or <NRf>
Polarity

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or NORMAL
1 or INVERT

Example: `POCSAG:POLARITY INVERT`

:Polarity?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `INVERT`

:POcsagtones

:READMessage?

Description: Reads back the received POCSAG message. This is a query only.

Parameters: N/A

Response: <STRING RESPONSE DATA>
Received POCSAG message

Example Response: `"16:02 TEST"`

:POcsagtones

:READStats?

Description: Reads back the statistics of the received POCSAG message. This is a query only.

Parameters: N/A

Response: <NR2>,<NR1>,<CRD>,<CRD>,<NR1>,<NR1>
Bit rate (kHz), RIC, Polarity, Alert type, Errored codewords, Fixed codewords.

Example Response: `1.201,360044,NORMAL,TEXT_ALERT,0,0`

:POcsagtones

:RFFreq

Description: Sets the RF generator frequency in POCSAG mode

Parameters: <NRf>
Frequency

Allowed suffices: MHZ,KHZ

Default suffix: MHZ

Example: `POCSAG:RFFREQ 466.075MHZ`

:RFFreq?

Parameters: N/A

Response: <NR2>
Frequency (MHz)

Example Response: `466.075000`

:POcsagtones

:RFLevel

Description: Sets RF output level in POCSAG mode

Parameters: <NRf>
Level

Allowed suffices: DBM

Default suffix: DBM

Example: `POCSAG:RFLEVEL −80`

:RFLevel?

Parameters: N/A

Response: <NR2>
Level (dBm)

Example Response: `−80.0`

:POcsagtones

:RFOut

Description: Sets the RF output port in POCSAG mode

Parameters: <CPD> or <NRf>
Output selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or GEN_N
1 or GEN_BNC

Example: `POCSAG:RFOUT GEN_N`

:RFOut?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `GEN_N`

:POcsagtones

:RIc

Description: Sets the Radio Identity Code in POCSAG mode

Parameters: <NRf>
Radio Identity Code

Allowed suffices: N/A

Default suffix: N/A

Example: `POCSAG:RIC 360044`

:RIc?

Parameters: N/A

Response: <NR1>
Radio Identity Code

Example Response: `360044`

:POcsagtones?

Description: Queries the status of POCSAG tones by producing the combined return values of the sub commands of POCSAGTONES

These responses are separated by semi-colons

Parameters: N/A

Response: <NR2>;<CRD>;<NR2>;<CRD>;<CRD>;<NR1>;<CRD>;<CRD>;<STRING RESPONSE DATA>;<NR2>,<NR1>,<CRD>,<CRD>,<NR1>,<NR1>;<NR2>;<NR2>;<CRD>;<NR1>

Example Response:

```
99.0;TEXT_ALERT;2.400;ALPHANUMERIC;RIC;
2.400;TEXT_MESS_1;INVERT;"16:02 TEST";
1.201, 360044, NORMAL,TEXT_ALERT,0,0;
466.07500;–80.0;GEN_N;360044
```

POCSAG settings are:

AF level 99 mV
Alert message type is test alert; bitrate is 2.4 kHz
Data is decoded as alphanumeric
Decoding begins on recognition of RIC code
FM deviation is 2.4 kHz; message to send is text message 1
Transmit polarity is inverted
Message decoded was 16:02 TEST.

Statistics of received message are:

Bit rate 1.201 kHz
RIC is 360044
Polarity is normal
Alert type is text message
No errored codewords
No fixed codewords.

The RF gen frequency is 466.075 MHz
RF level is –80 dBm
RF output port is the N-type
RIC code is 360044.

:PREemph

Description: Controls whether frequency modulation is routed through the pre-emphasis filter

Parameters: <CPD> or <NRf>
Pre-emphasis selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or ON

Example: `PREEMPH ON`

:PREemph?

Parameters: N/A

Response: <CRD>
Pre-emphasis status

Example Response: `ON`

:Qerror?

Description: Returns the last Queue error generated by the Message Exchange Protocol Enforcer

Parameters: N/A

Response: <NR1>
Last error

Responses: 0 corresponds to 'No Error'
1 corresponds to 'Interrupted'
2 corresponds to 'Unterminated'
3 corresponds to 'Deadlocked'

:RECAll

Description: Makes the instrument perform a recall from a particular store number. Instrument settings stores only.

Parameters: <NRf>
Store number

Allowed suffices: N/A

Default suffix: N/A

Example: `RECALL 1`

:RECEiver

Controls the instrument's receiver

Not used alone

:RECEiver

:Autotune

Description: Controls the autotune function of the receiver

Parameters: <CPD> or <NRf>

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or ON

Example: `RECE:AUTO OFF`

:Autotune?

Parameters: N/A

Response: <CRD>
Autotune status

Example Response: `OFF`

:RECEiver

:Deemph

Description: Controls the de-emphasis filter of the receiver

Parameters: <CPD> or <NRf>

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or ON

Example: `RECE:DEEMPH OFF`

:Deemph?

Parameters: N/A

Response: <CRD>
De-emphasis status

Example Response: `OFF`

:RECEiver

:FERror

Description: Controls the frequency error measurement method of the Rx Test mode

Parameters: <CPD> or <NRf>

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or PPM
1 or FREQ

Example: `RECE:FER PPM`

:FERror?

Parameters: N/A

Response: <CRD>
Frequency error measurement method

Example Response: `PPM`

:RECEiver

:FIlter

Description: Controls the IF bandwidth of receiver

Parameters: <CPD> or <NRf>
Filter selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or FIL_300HZ
1 or FIL_3KHZ
2 or FIL_30KHZ
3 or FIL_300KHZ

Example: `RECE:FILT 1`

:FIlter?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `FIL_30KHZ`

:RECEiver

:FREQ

Description: Sets the frequency of the test set's receiver

Parameters: <NRf>
Frequency

Allowed suffices: MHZ, KHZ, HZ

Default suffix: MHZ

Example: `RECE:FREQ 890.0625MHZ`

:FREQ?

Parameters: N/A

Response: <NR2>
Frequency (MHz)

Example Response: `890.062500`

:RECEiver

:FRESn

Description — Controls the frequency resolution of the RF counter

Parameters: — <CPD> or <NRf>
Resolution selection

Allowed suffices: — N/A

Default suffix: — N/A

Valid Data — 0 or RESN_1HZ
1 or RESN_10HZ
2 or RESN_P1HZ

Example: — `RECE:FRESN 1`

:FRESn?

Parameters: — N/A

Response: — <CRD>
Current selection

Example Response: — `RESN_10HZ`

:RECEiver

:HARMFIlter

Description: — Controls the IF bandwidth of receiver in harmonic analysis mode.

Parameters: — <CPD> or <NRf>
Filter selection

Allowed suffices: — N/A

Default suffix: — N/A

Valid Data: — 0 or FIL_300HZ
1 or FIL_3KHZ
2 or FIL_30KHZ
3 or FIL_300KHZ

Example: — `RECE:HARMFILT 0`

:HARMFIlter?

Parameters: — N/A

Response: — <CRD>
Current selection

Example Response: — `FIL_300HZ`

:RECEiver

:HARMOnics

Description	Controls whether harmonics are measured in TX test mode
Parameters:	<CPD> or <NRf> *Harmonic measurement mode selection*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data	0 or OFF 1 or ON
Example:	`RECE:HARM 1`

:HARMOnics?

Parameters:	N/A
Response:	<CRD> *Current selection*
Example Response:	`ON`

:RECEiver

:Powerbw

Description	Controls whether power measurements are taken with the broadband power meter or with the narrow band meter
Parameters:	<CPD> or <NRf> *Power measurement selection*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data	0 or BROADBAND 1 or INBAND
Example:	`RECE:POWERBW BROAD`

:Powerbw?

Parameters:	N/A
Response:	<CRD> *Current selection*
Example Response:	`BROADBAND`

:RECEiver

:Reflevel

Description: Controls receiver attenuator hold. Indicated values are the maximum receiver input level that should be used

Parameters: <CPD> or <NRf>
Held attenuator level selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data:

0 or REF_AUTO	Auto
1 or REF_AM50_NM24_DBM	Ant –50 dBm, ‘N’ –24 dBm
2 or REF_AM40_NM14_DBM	Ant –40 dBm, ‘N’ –14 dBm
3 or REF_AM30_NM4_DBM	Ant –30 dBm, ‘N’ –4 dBm
4 or REF_AM20_N6_DBM	Ant –20 dBm, ‘N’ 6 dBm
5 or REF_AM10_N16_DBM	Ant –10 dBm, ‘N’ 16 dBm
6 or REF_A0_N26_DBM	Ant 0 dBm, ‘N’ 26 dBm
7 or REF_A10_N36_DBM	Ant 10 dBm, ‘N’ 36 dBm
8 or REF_A20_N46_DBM	Ant 20 dBm, ‘N’ 46 dBm
9 or REF_A30_N56_DBM	Ant 30 dBm, ‘N’ 56 dBm

Example: `RECE:REFLEVEL 4`

:Reflevel?

Parameters: N/A

Response: <CRD> or <NR1>
Current held attenuator level

Example Response: `REF_AM20_N6_DBM`

:RECEiver

:Ssbsens

Description: Controls the receiver sensitivity when in SSB demodulation mode

Parameters: <CPD> or <NRf>
Receiver sensitivity selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data:
0 or LOW
1 or MEDIUM
2 or HIGH

Example: `RECE:SSB LOW`

:Ssbsens?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `LOW`

:RECEiver?

Description: Queries the entire status of the receiver by producing the combined return values of the sub commands of RECEIVER

These responses are separated by semi-colons

Parameters: N/A

Response: <CRD>;<CRD>;<CRD>;<CRD>;<NR2>;<CRD>;<CRD>;
<CRD>;<CRD>;<CRD>

Example Response: `OFF;OFF;FREQ;FIL_30KHZ;890.062500;`

`RESN_1HZ;FIL_300HZ;OFF;BROADBAND;LOW`

:RECSwitch

Description: Controls the routing of the RF input from the transmitter under test

Parameters: <CPD> or <NRf>
Input selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or REC_N
1 or REC_ANT

Example: `RECSW 0`

:RECSwitch?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `REC_N`

:RESponse

Controls some advanced features of the response formatter

Not used alone

:RESponse

:Format

Description: Controls whether or not the response formatter includes CR, LF in the output for better presentation

Parameters: <CPD> or <NRf>
Response format selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or ON
2 or MINIMUM

Example: `RESP:FORM OFF`

:Format?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `OFF`

:RESponse

:Header

Description: Controls whether or not the response formatter returns the command header and if so to what extent

Parameters: <CPD> or <NRf>
Response header selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or MINIMUM
2 or FULL
3 or DEFAULT

Example: `:RESP:HEAD FULL`

:Header?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `:RESPONSE:HEADER FULL`

:RESponse?

Description: Queries the status of the response formatter by producing the combined return values of the sub commands of RESPONSE

These responses are separated by semi-colons

Parameters: N/A

Response: <CRD>;<CRD>

Example Response: `OFF;OFF`

:RFgen

Controls the instrument's RF signal generator

Not used alone

:RFgen

:Freq

Description: Sets the frequency of the RF generator for receiver testing

Parameters: <NRf>

Frequency

Allowed suffices: MHZ,KHZ,HZ

Default suffix: MHZ

Example: `:RFGEN:FREQ 98.8MHZ`

:Freq?

Parameters: N/A

Response: <NR2>

Frequency (MHz)

Example Response: `98.800000`

:RFgen

:Level

Description: Sets the level of the RF generator for receiver testing

Parameters: <NRf>

Level

Allowed suffices: DBM,DBUV,UV,MV

Default suffix: DBM

Example: `RFGEN:LEV –80DBM`

:Level?

Parameters: N/A

Response: <NR2>

Level (dBm)

Example Response: `–80.0`

:RFgen

:Mode

Description: Controls the Attenuator Hold mode of the RF generator for receiver testing

Parameters: <CPD> or <NRf>
Mode selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or Normal
1 or Seamless

Example: `RFGEN:MODE SEAMLESS`

:Mode?

Parameters: N/A

Response: <CRD>
Current mode

Example Response: `SEAMLESS`

:RFgen

:Topseamlevel

Description: Sets the top level of the seamless range of the RF generator for receiver testing

Parameters: <NRf>
Level

Allowed suffices: DBM

Default suffix: DBM

Example: `RFGEN:LEV –20DBM`

:Topseamlevel?

Parameters: N/A

Response: <NR2>
Level (dBm)

Example Response: `–20.0`

:RFgen

:Status

Description: Controls the status of the RF generator for receiver testing

Parameters: <CPD> or <NRf>
Status selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or ON

Example: `RFGEN:STAT ON`

:Status?

Parameters: N/A

Response: <CRD>
Current status

Example Response: `ON`

:RFgen

:Volts

Description: Controls whether the RF generator volts level is EMF or PD

Parameters: <CPD> or <NRf>
Status selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or PD
1 or EMF

Example: `RFGEN:VOLTS PD`

:Volts?

Parameters: N/A

Response: <CRD>
Current status

Example Response: `PD`

:RFgen?

Description: Queries the status of the RF signal generator by producing the combined return values of the sub commands of RFGEN

These responses are separated by semi-colons

Parameters: N/A

Response: <NR2>;<NR2>;<CRD>;<CRD>;<NR2>;<CRD>

Example Response: `98.800000;−80.0;SEAMLESS;ON;−20;PD`

:RXDIsp

Description	Controls the type of data display in RX and AF test modes
Parameters:	<CPD> or <NRf> *Display selection*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data	0 or BARCHARTS 1 or SCOPE 2 or LARGE_SCOPE
Example:	`RXDISP BARCHARTS`

:RXDIsp?

Parameters:	N/A
Response:	<CRD> *Current selection*
Example Response:	`BARCHARTS`

:RXDNotch

Description:	Sets the audio notch frequency
Parameters:	<NRf> *Frequency (kHz)*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	One of the factory-set audio notch frequencies (Option 29) or 1 kHz
Example:	`RXDNOTCH 0.9`

:RXDNotch?

Parameters:	N/A
Response:	<NR2> *Current set audio notch frequency (kHz)*
Example Response:	`0.9`

:RXDType

Description: Controls the audio distortion measurement type

Parameters: <CPD> or <NRf>
Distortion measurement type

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or DISTN
2 or SINAD
3 or SN

Example: `RXDTYPE SINAD`

:RXDType?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `SINAD`

:RXEqtx

Description: Sets the RF generator frequency to the receiver frequency adjusted for the duplex offset. Action only.

Parameters: N/A

Allowed suffices: N/A

Default suffix: N/A

Example: `RXEQTX`

:RXFilt

Description: Controls the audio input filter bandwidth

Note. This command is retained for compatibility with the 2945A command set. Use AUDFILT:FILTER.

Parameters: <CPD> or <NRf>
Filter selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or LP_50KHZ
1 or LP_15KHZ
2 or STD_BP
3 or LP_300HZ
4 or LP_3KHZ
5 or HP_300HZ
6 or PSOPH (CMESS or CCITT as fitted)

Example: `RXFILT 2`

:RXFilt?

Parameters: N/A

Response: <CRD>
Current selection

Note. If the current audio input filter is not one of the standard 2945A filters then a null-string is returned.

Example Response: `STD_BP`

:SELcal

Controls the SELCAL signal generator

Not used alone

:SELcal

:Depth

Description: Sets the modulation depth in SELCAL

Parameters: <NRf>
Depth

Allowed suffices: PCT

Default suffix: PCT

Example: `SELCAL:DEPTH 20.0`

:Depth?

Parameters: N/A

Response: <NR2>
AM depth (%)

Example Response: `20.0`

:SELcal

:Mode

Description: Sets the mode of generation in SELCAL

Parameters: <CPD> or <NRf>
Mode

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or STOP
1 or BURST
2 or CONT

Example: `SELCAL:MODE BURST`

:Mode?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `STOP`

:SELcal

:RFFreq

Description:	Sets RF frequency in SELCAL mode
Parameters:	<NRf> *Frequency*
Allowed suffices:	MHZ, KHZ, HZ
Default suffix:	MHZ
Example:	`SELCAL:RFFREQ 108`

:RFFreq?

Parameters:	N/A
Response:	<NR2> *Frequency (MHz)*
Example Response:	`108.000000`

:SELcal

:RFLevel

Description:	Sets RF output level in SELCAL mode
Parameters:	<NRf> *Level*
Allowed suffices:	DBM,DBUV,UV,MV
Default suffix:	DBM
Example:	`SELCAL:RFLEV –80DBM`

:RFLevel?

Parameters:	N/A
Response:	<NR2> *Level (dBm)*
Example Response:	`–80.0`

:SELcal

:RFOut

Description:	Sets the RF output in SELCAL mode
Parameters:	<CPD> or <NRf> *Output selection*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or GEN_N 1 or GEN_BNC
Example:	`SEL:RFOUT GEN_N`

:RFOut?

Parameters:	N/A
Response:	<CRD> *Current selection*
Example Response:	`GEN_N`

:SELcal

:Sequence

Description:	Sets the calling sequence in SELCAL
Parameters:	<SPD> *Calling sequence*
Allowed suffices:	N/A
Default suffix:	N/A
Example:	`SELCAL:SEQ "ABCD"`

:Sequence?

Parameters:	N/A
Response:	<SPD> *Calling sequence*
Example Response:	`"ABCD"`

SELcal?

Description:	Queries the status of the SELCAL signal generator by producing the combined return values of the sub commands of SELCAL These responses are separated by semi-colons
Parameters:	N/A
Parameters:	N/A
Response:	<NR2>;<CRD>;<NR2>;<NR2>;<CRD>; <STRING RESPONSE DATA>
Example Response:	`20.0;STOP;108.00000;` `–80.0;GEN_N;"ABCD"`

:SEQtones

Controls the settings for SEQUENTIAL tones

Not used alone

:SEQtones

:AFlevel

Description: Sets Sequential Generator Audio Level

Parameters: <NRf>

Level

Allowed suffices: MV, V, DBM

Default suffix: MV

Example: `:SEQ:AFLEVEL 100MV`

Sets level to 100 mV

:AFlevel?

Parameters: N/A

Response: <NR2>

Audio level in mV to 0.1 mV resolution

Example Response: `99.0`

:SEQtones

:AMdepth

Description: Sets Sequential Generator AM Depth

Parameters: <NRf>

Depth

Allowed suffices: PCT

Default suffix: PCT

Example: `SEQ:AMD 30PCT`

:AMdepth?

Parameters: N/A

Response: <NR2>

AM depth (%)

Example Response: `30.0`

:SEQtones

:DECOdereset

Description:	Clears and resets the Sequential tones decoder. Action only.
Parameters:	N/A
Allowed suffices:	N/A
Default suffix:	N/A
Example:	`SEQ:DECODERESET`

:SEQtones

:DECStd

Description:	Sets the sequential tones standard for the decoder
Parameters:	<CPD> or <NRf> *Decode standard*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or CCIR 1 or ZVEI 2 or DZVEI 3 or EEA 4 or EIA 5 or USER1 6 or USER2
Example:	`SEQ:DECSTD EIA`

:DECStd?

Parameters:	N/A
Response:	<CRD> *Decode standard*
Example Response:	`EIA`

:SEQtones

:DUration

Description: Sets the tone duration in sequential tones

Parameters: <NRf>
Duration

Allowed suffices: MS, S

Default suffix: MS

Example: `SEQ:DUR 100`
Sets the sequential tone duration to 100 ms

:DUration?

Parameters: N/A

Response: <NR1>
Duration (ms)

Example Response: `100`

:SEQtones

:ENcstd

Description: Sets the sequential tones standard for the encoder

Parameters: <CPD> or <NRf>
Encode standard

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or CCIR
1 or ZVEI
2 or DZVEI
3 or EEA
4 or EIA
5 or USER1
6 or USER2

Example: `SEQ:ENCSTD EEA`

:ENCstd?

Parameters: N/A

Response: <CRD>
Encode standard

Example Response: `EEA`

:SEQtones

:EXtended

Description: Sets the extended tone duration

Parameters: <NRf>

Extended tone duration

Allowed suffices: MS,S

Default suffix: MS

Example: `SEQ:EXT 750MS`

:EXtended?

Parameters: N/A

Response: <NR1>

Extended tone duration

Example Response: `750`

:SEQtones

:FMdevn

Description: Sets Sequential Generator FM Deviation

Parameters: <NRf>

Deviation

Allowed suffices: KHZ, HZ

Default suffix: KHZ

Example: `SEQ:FM 2.4KHZ`

:FMdevn?

Parameters: N/A

Response: <NR1>

Deviation (Hz)

Example Response: `2400`

:SEQtones

:FReqshift

Description:	Sets the sequential tone frequency shift in percent
Parameters:	<NRf> *Frequency shift*
Allowed suffices:	PCT
Default suffix:	PCT
Example:	`SEQ:FREQSHIFT 2PCT` Adjusts all tone frequencies by plus 2%

:FReqshift?

Parameters:	N/A
Response:	<NR2> *Frequency shift (%)*
Example Response:	`2.0`

:SEQtones

:Mode

Description:	Sets the mode of generation in sequential tones
Parameters:	<CPD> or <NRf> *Mode*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or STOP 1 or BURST 2 or CONT
Example:	`SEQ:MODE BURST`

:Mode?

Parameters:	N/A
Response:	<CRD> *Current selection*
Example Response:	`STOP`

:SEQtones

:READSequence?

Description: Returns the contents of the sequential decode register. This is a query only.

Parameters: N/A

Response: <STRING RESPONSE DATA>
Sequential tones received sequence

Example Response: `"80E0E-80101-80A01"`

:SEQtones

:READTone?

Description: Returns the statistics of the nth tone in the received sequence

Parameters: <NRf>
Tone number

Response: <ARB ASCII data>,<NR2>,<NR2>,<NR1>
Tone, Tone frequency (kHz), Tone error (%), Tone duration (ms)

Example: `SEQ:READTONE? 5`

Example Response: `5,1.250,0.0,50`

:SEQtones

:REVertive

Description: Sets whether or not revertive tones are active

Parameters: <CPD> or <NRf>
Revertive tones status

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or ON

Example: `SEQ:REV ON`

:REVertive?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `ON`

:SEQtones

:SEquence

Description: Sets the sequential tones encode sequence

Parameters: <STRING PROGRAM DATA>
Sequence

Allowed suffices: N/A

Default suffix: N/A

Example: `SEQ:SEQ "80101"`

:SEquence?

Parameters: N/A

Response: <STRING RESPONSE DATA>
Sequence

Example Response: `"80101"`

:SEQtones

:STandard

Description: Sets the sequential tones standard for the encoder or decoder whichever is currently selected

Parameters: <CPD> or <NRf>
Tone standard

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or CCIR
1 or ZVEI
2 or DZVEI
3 or EEA
4 or EIA
5 or USER1
6 or USER2

Example: `SEQ:STANDARD EEA`

:STandard?

Parameters: N/A

Response: <CRD>
Standard

Example Response: `EEA`

:SEQtones

:USER*n*

:Copystandard

Description: Copies one of the default standards to the user *n* tone standard, where *n* is 1 or 2. Action only.

Parameters: <CPD> or <NRf>
Standard

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or CCIR
1 or ZVEI
2 or DZVEI
3 or EEA
4 or EIA

Example: `SEQ:USER1:COPY CCIR`

:SEQtones

:USER*n*

:Duration

Description: Sets the tone duration of the sequential tones user *n* standard where *n* is 1 or 2

Parameters: <NRf>
Duration

Allowed suffices: MS, S

Default suffix: MS

Example: `SEQ:USER1:DUR 100`

Sets the user 1 tone duration to 100 ms

:USER*n*

:Duration?

Parameters: N/A

Response: <NR1>
Duration (ms)

Example Response: `100`

:SEQtones

:USER*n*

:Extended

Description: Sets the extended tone duration of the user *n* tone standard, where *n* is 1 or 2

Parameters: <NRf>

Extended tone duration

Allowed suffices: MS, S

Default suffix: MS

Example: `SEQ:USER1:EXT 750MS`

:USER*n*

:Extended?

Parameters: N/A

Response: <NR1>

Extended tone duration

Example Response: `750`

:SEQtones

:USER*n*

:Freq

Description: Sets the frequency of the nth tone in the user *n* tone standard, where *n* in user *n* is 1 or 2

Parameters: <NRf>,<NRf>

Tone number,Frequency

Allowed suffices: Tone number:<N/A>,Frequency:KHZ,HZ

Default suffix: Tone number:<N/A>,Frequency:KHZ

Example: `SEQ:USER1:FREQ 5,1.300`

Sets frequency of tone 5 to 1.3 kHz

:USER*n*

:Freq?

Parameters: <NRf>

Tone number

Response: <NR2>

Frequency (kHz)

Example: `SEQ:USER1:FREQ? 5`

Example Response: `1.300`

:SEQtones?

Description: Queries the status of SEQUENTIAL tones by producing the combined return values of the sub commands of SEQTONES

These responses are separated by semi-colons

Parameters: N/A

Response: <NR2>;<NR2>;<CRD>;<NR1>;<CRD>;<NR1>;<NR1>;
<NR2>;<CRD>;<STRING RESPONSE DATA>;<CRD>;
<STRING RESPONSE DATA>;<CRD>;<NR1>;<NR1>;
<NR1>;<NR1>

Example Response:

```
99.0;30.0;EEA;100;EEA;750;2.400;2.0;STOP;
"80E0E-80101-80A01";ON;"80101";EEA;100;
750;100;750
```

The sequential tones settings are :

AF level 99 mV; AM depth 30% (if AM set)
Decode standard is EEA
Tone duration is 100 ms
Encode standard is EEA
Extended tone duration is 750 ms
FM deviation is 2.4 kHz (if FM set)
Frequency shift is 2%
Sequential mode is stopped
Decoded sequential sequence is "80E0E-80101-80A01"
Revertive tones are on
Sequence to encode is "80101"
Standard (decode or encode whichever is selected) is EEA
USER1 normal tone duration is 100 ms
USER1 extended tone duration is 750 ms
USER2 normal tone duration is 100 ms
USER2 extended tone duration is 750 ms.

:SETfilt

:Lp*n*

Description: Sets user defined Lowpass Audio/Demod filter frequency where *n* = 1 to 4.
Note. This sets only the user defined value. To set filters use AUDfilt ; Filter or MODFilt : Filter.

Parameters: <NRf>
Frequency (kHz)

Allowed suffices: KHZ or HZ

Default suffix: KHZ

Example: `:SETfilt:LP1 1.000KHZ`
Sets Lowpass Audio Filter 1 MMI soft key to a frequency of 1.000 kHz

:Lp*n*?

Parameters: N/A

Response: <NR2>
Frequency in kHz to 1 Hz resolution

Example Response: `5.000`
Frequency currently set to 5 kHz

:SETfilt

:Bp*n*

Description: Sets user defined Bandpass Audio/Demod filter bandwidth using a highpass filter and one of the user defined lowpass filters where *n* = 1 to 4.
Note. This sets only the user defined value. To set filters use AUDfilt:Filter or MODFilt:Filter.

Parameters: <CPD>, <CPD>
High Pass Filter, *Low Pass Filter*

Valid Data: `HP_50 | HP300`
`LP1 | LP2 | LP3 | LP4`

Example: `:SETfilt:BP1:HP_50,LP1`
Sets Audio bandpass filter 1 to highpass component of 50 Hz and lowpass component to be Lowpass filter 1.

:Bp*n*?

Parameters: N/A

Response: <CRD>, <CRD>

Example Response: `HP_50,LP1`
Audio bandpass filter 1 has highpass component of 50 Hz and lowpass component is Lowpass filter 1.

:SETfilt

:Default

Description: Sets the Lowpass Filters and the Bandpass Filters to factory default values

Parameters: N/A

Allowed suffices:

Default suffix:

Example: `:SETfilt:Default`

:SETfilt?

Description: Queries the status of the user defined audio Demod filters.

These responses are separated by semi-colons

Parameters: N/A

Response: <NR2>;<NR2>;<NR2>;<NR2>;<CRD>,<CRD>;<CRD>,<CRD>;<CRD>,<CRD>;<CRD>,<CRD>

Example Response: `15.000;4.000;3.400;0.300;HP_300,LP3;HP_300,LP2;HP_50,LP1;HP_50,LP3`

:SPecana

Controls the instrument spectrum analyzer

Not used alone

:SPecana

:Bwvideo

Description: Sets the video bandwidth filter

Parameters: <CPD> or <Nrf>

Video BW filter selection

Allowed suffices: N/A

Default suffix: N/A

Valid data 0 or OFF
1 or ON

Example: `SPECANA:BWV 0`

:Bwvideo?

Parameters: N/A

Response: <CRD>

Video BW filter selection

Example Response: `OFF`

:SPecana

:Center

Description: Sets the center frequency of the spectrum analyzer scan

Parameters: <NRf>
Center frequency

Allowed suffices: MHZ,KHZ,HZ

Default suffix: MHZ

Example: `SPECANA:CENT 500MHZ`

:Center?

Parameters: N/A

Response: <NR2>
Frequency (MHz)

Example Response: `500.000000`

:SPecana

:Filter

Description: Controls the resolution bandwidth of the spectrum analyzer

Parameters: <CPD> or <NRf>
Filter selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or AUTO
1 or FIL_300HZ
2 or FIL_3KHZ
3 or FIL_30KHZ
4 or FIL_300KHZ
5 or FIL_3MHZ

Example: `SPEC:FILT 0`

:Filter?

Parameters: N/A

Response: <CRD>
Current selection

Example: `AUTO`

:SPecana

:LLFilt

Description: Controls the audio filter bandwidth when in the look and listen mode

Parameters: <CPD> or <NRf>
Filter selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or LP_15KHZ
1 or STD_BP

Example: `SPEC:LLFILT STD_BP`

:LLFilt?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `STD_BP`

:SPecana

:LLIfbw

(Only available if the 'Demodulation filters' option is fitted)

Description: Controls the IF bandwidth in the Look and Listen mode

Parameters: <CPD> or <NRf>
IF bandwidth selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or FIL_5KHZ
1 or FIL_12P5KHZ
2 or FIL_25KHZ
3 or FIL_50KHZ
4 or FIL_300KHZ
5 or FIL_15KHZ

Example: `SPEC:LLIFBW FIL_25KHZ`

:LLIfbw?

(Only available if the 'Demodulation filters' option is fitted)

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `FIL_25KHZ`

:SPecana

:LLSpan

Description: Controls the span of the spectrum analyzer sweep when in the look and listen mode

Parameters: <CPD> or <NRf>
Span selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or LL_1MHZ
1 or LL_500KHZ
2 or LL_200KHZ
3 or LL_100KHZ

Example: `SPEC:LLSP LL_1MHZ`

:LLSpan?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `LL_1MHZ`

:SPecana

Note: Use :MEASUre:MKr1? to return the signal level at the marker.

:MArker

Description: Controls the status of the spectrum analyzer marker

Parameters: <CPD> or <NRf>
Marker status

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or ON
2 or DELTA

Example: `SPEC:MARK ON`

:MArker?

Parameters: N/A

Response: <CRD>
Marker status

Example Response: `ON`

:SPecana

:MKRFreq

Description: Sets the frequency of the marker on the spectrum analyzer display

Parameters: <NRf>
Marker frequency

Allowed suffices: MHZ,KHZ

Default suffix: MHZ

Example: `SPEC:MKRF 499.8`

:MKRFreq?

Parameters: N/A

Response: <NR2>
Frequency (MHz)

Example Response: `499.800000`

:SPecana

:MKRPeak

Description: Sets the marker to the highest level on the spectrum analyzer display. This is an action only.

Parameters: N/A

Allowed suffices: N/A

Default suffix: N/A

Valid data N/A

Example: `SPEC:MKRP`

:SPecana

:MOde

Description: Controls the operating mode of the spectrum analyzer

Parameters: <CPD> or <NRf>
Spectrum analyzer mode

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or NORMAL
1 or LOOK_LIST

Example: `SPEC:MODE LOOK_LIST`

:MOde?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `LOOK_LIST`

:SPecana

:Peakhold

Description: Displays the highest received signal level at each frequency point on the spectrum analyzer display

Parameters: <CPD>
Peak hold status

Allowed suffices: N/A

Default suffix: N/A

Valid data OFF
ON

Example: `SPEC:PEAKHOLD ON`

:Peakhold?

Parameters: N/A

Response: <CRD>
Peak hold status

Example Response: `ON`

:SPecana

:Reflevel

Description: Sets the reference level (top of screen) of the spectrum analyzer

Parameters: <NRf>
Reference level

Allowed suffices: DBM

Default suffix: DBM

Example: `SPEC:REFLEV 10DBM`

:Reflevel?

Parameters: N/A

Response: <NR2>
Reference level (dBm)

Example Response: `10.0`

:SPecana

:SPan

Description:	Sets the span of the spectrum analyzer sweep
Parameters:	<NRf> *Span*
Allowed suffices:	MHZ,KHZ
Default suffix:	MHZ
Example:	`SPEC:SPAN 100MHZ`

:SPan?

Parameters:	N/A
Response:	<NR2> *Span (MHz)*
Example Response:	`100.000000`

:SPecana

:STArt

Description:	Sets the start frequency of the spectrum analyzer sweep
Parameters:	<NRf> *Start frequency*
Allowed suffices:	MHZ,KHZ
Default suffix:	MHZ
Example:	`SPEC:START 450`

:STArt?

Parameters:	N/A
Response:	<NR2> *Start frequency (MHz)*
Example Response:	`450.000000`

:SPecana

:STOp

Description:	Sets the stop frequency of the spectrum analyzer sweep
Parameters:	<NRf> *Stop frequency*
Allowed suffices:	MHZ,KHZ
Default suffix:	MHZ
Example:	`SPEC:STOP 550MHZ`

:STOp?

Parameters:	N/A
Response:	<NR2> *Stop frequency (MHz)*
Example Response:	`550.000000`

:SPecana

:TGLevel

Description:	Sets the level of the spectrum analyzer tracking generator
Parameters:	<NRf> *Tracking generator level*
Allowed suffices:	DBM
Default suffix:	DBM
Example:	`SPEC:TGLEV 0DBM`

:TGLevel?

Parameters:	N/A
Response:	<NR2> *Current level (dBm)*
Example Response:	`0.0`

:SPecana

:TGMode

Description: Controls the operating mode of the RF gen in spectrum analyzer

Parameters: <CPD> or <NRf>
Spectrum analyzer mode

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or TRACK_GEN
1 or SIG_GEN

Example: `SPEC:TGM SIG_GEN`

:TGMode?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `SIG_GEN`

:SPecana

:TGOffset

Description: Sets the frequency offset of the spectrum analyzer tracking generator

Parameters: <NRf>
Frequency offset

Allowed suffices: MHZ,KHZ

Default suffix: MHZ

Example: `SPEC:TGOFF  –10.7`

:TGOffset?

Parameters: N/A

Response: <NR2>
Offset frequency (MHz)

Example Response: `–10.700000`

:SPecana

:TGStatus

Description: Controls the status of the spectrum analyzer tracking generator

Parameters: <CPD> or <NRf>
Status

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or ON

Example: `SPEC:TGSTAT ON`

:TGStatus?

Parameters: N/A

Response: <CRD>
Current status

Example Response: `ON`

:SPecana

:Vertscale

Description: Controls the vertical scale of the spectrum analyzer display

Parameters: <CPD> or <NRf>
dB per division

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or TEN_DB_PER
1 or TWO_DB_PER

Example: `SPEC:VERTSCALE 0`

:Vertscale?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `TEN_DB_PER`

:SPecana?

Description: Queries the status of the spectrum analyzer by producing the combined return values of the sub commands of SPECANA

These responses are separated by semi-colons

Parameters: N/A

Response: <CRD>;<NR2>;<CRD>;<CRD>;<CRD>;<CRD>;<CRD>;
<NR2>;<CRD>;<NR2>;<NR2>;<NR2>;<NR2>;<NR2>;
<CRD>;<NR2>;<CRD>;<CRD>

Example Response:
```
ON;500.00000;AUTO;STD_BP;LL_1MHZ;OFF;
499.800000;LOOK_LIST;10.0;100.000000;
450.000000;550.000000;0.0;SIG_GEN;
−10.700000;ON;TEN_DB_PER
```

:TEstmode

Description: Controls the basic mode of the communications service monitor

Parameters: <CPD> or <NRf>
Mode

Allowed suffices: N/A

Default suffix: N/A

Valid Data
0 or RX_TEST
1 or TX_TEST
2 or DX_TEST
3 or SYSTEMS
4 or AF_TEST
5 or SPEC_ANA
6 or TONES_MODE
7 or ACC_PWR_MODE
8 or TRANSIENT_MODE
9 or OCC_BW

Example: `TEST SPEC_ANA`

:TEstmode?

Parameters: N/A

Response: <CRD>
Current mode

Example Response: `SPEC_ANA`

:TONEMode

Description	Controls the type of tones in tones mode
Parameters:	<CPD> or <NRf> *Type of tones*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data	0 or SEQ 1 or DTMF 2 or POCSAG 3 or DCS 4 or SELCAL 5 or ILS 6 or MKR 7 or VOR 8 or TONEREM
Example:	`TONEMODE POCSAG`

:TONEMode?

Parameters:	N/A
Response:	<CRD> *Type of tones*
Example Response:	`SEQ`

:TONERem

:FUNCDur

Description: Sets the Function Tone duration in Tones Remote mode

Parameters: <NRf>
Duration

Allowed suffices: MS, S

Default suffix: MS

Example: `TONEREM:FUNCDUR 40`
Sets the Function Tone duration to 40 ms

:FUNCDur?

Parameters: N/A

Response: <NR1>
Duration (ms)

Example Response: `40`

:TONERem

:FUNCFreq

Description: Sets the function Tone frequency in Tones Remote mode

Parameters: <NRf>
Frequency (kHz)

Allowed suffices: KHZ, HZ

Default suffix: KHZ

Example: `:TONERem:FUNCFREQ 1.9500 KHZ`
Sets Function Tone frequency to 1.950 kHz

:FUNCFreq?

Parameters: N/A

Response: <NR2>
Frequency in kHz to 1 Hz resolution

Example Response: `1.9500`
Frequency currently set to 1.950 kHz

:TONERem

:FUNCLev

Description: Sets the function Tone level in Tones Remote mode

Parameters: <NRf>

Level

Allowed suffices: DB

Default suffix: DB

Example: `TONEREM:FUNCLEV 0.0 DB`

Sets Function Tone level to 0.0 dB

:FUNCLev?

Parameters: N/A

Response: <NR2>

Level (dB)

Example Response: `0.0`

:TONERem

:GUARDDur

Description: Sets the Guard Tone duration in Tones Remote mode

Parameters: <NRf>

Duration

Allowed suffices: MS, S

Default suffix: MS

Example: `TONEREM:GUARDDUR 40`

Sets the Guard Tone duration to 40 ms

:GUARDDur?

Parameters: N/A

Response: <NR1>

Duration (ms)

Example Response: `40`

:TONERem

:GUARDFreq

Description:	Sets the Guard Tone frequency in Tones Remote mode
Parameters:	<NRf> *Frequency (kHz)*
Allowed suffices:	KHZ, HZ
Default suffix:	KHZ
Example:	`:TONERem:GUARDFREQ 2.1750 KHZ` Sets Guard Tone frequency to 2.1750 kHz

:GUARDFreq?

Parameters:	N/A
Response:	<NR2> *Frequency in kHz to 1 Hz resolution*
Example Response:	`1.270` Frequency currently set to 1.1750 kHz

:TONERem

:GUARDLev

Description:	Sets the Guard Tone level in Tones Remote mode
Parameters:	<NRf> *Level*
Allowed suffices:	DB
Default suffix:	DB
Example:	`TONEREM:GUARDLEV 10.0 DB` Sets Function Tone level to 10.0 dB

:GUARDLev?

Parameters:	N/A
Response:	<NR2> *Level (dB)*
Example Response:	`10.0`

:TONERem

:Levref

Description: Sets the Reference Level for the tones used in the Tone Remote system

Parameters: <NRf>
Level

Allowed suffices: DB MV

Default suffix: MV

Example: `TONEREM:LEVREF 100.0 MV`
Sets Tone Remove reference level to 100.0 mV

:Levref?

Parameters: N/A

Response: <NR2>
Level mV

Example Response: `100.0`

:TONERem

:MAXDur

Description: Sets the Max Tone duration in Tones Remote mode

Parameters: <NRf>
Duration

Allowed suffices: MS, S

Default suffix: MS

Example: `TONEREM:MAXDUR 120`
Sets the Max Tone duration to 120 ms

:MAXDur?

Parameters: N/A

Response: <NR1>
Duration (ms)

Example Response: `120`

:TONERem

:MAXFreq

Description: Sets the Max Tone frequency in Tones Remote mode

Parameters: <NRf>

Frequency (kHz)

Allowed suffices: KHZ, HZ

Default suffix: KHZ

Example: `:TONERem:MAXFreq 2.150 KHZ`

Sets Max Tone frequency to 2.1750 kHz

:MAXFreq?

Parameters: N/A

Response: <NR2>

Frequency in kHz to 1 Hz resolution

Example Response: `2.1750`

Frequency currently set to 2.1750 kHz

:TONERem

:MAXLev

Description: Sets the Max Tone level in Tones Remote mode

Parameters: <NRf>

Level

Allowed suffices: DB

Default suffix: DB

Example: `TONEREM:MAXLEV 10.0 DB`

Sets Max Tone level to 10.0 dB

:MAXLev?

Parameters: N/A

Response: <NR2>

Level (dB)

Example Response: `10.0`

:TONERem

:Mode

Description:	Sets the tone transmit mode for the Tone Remote system
Parameters:	<CPD> or <NRf> *Mode*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or STOP 1 or BURST 2 or CONT
Example:	`TONEREM:MODE BURST` *Tone Remote output set to Burst*

:Mode?

Parameters:	N/A
Response:	<CRD> *Current selection*
Example Response:	`BURST`

:TONERem?

Description:	Queries the status of Tone Remote by producing the combined return values of the sub commands of TONERem. These responses are separated by semi-colons.
Parameters:	N/A
Response:	<NR1>;<NR2>;<NR2>;<NR1>;<NR2>;<NR2>; <NR2>;<NR1>;<NR2>;<NR2>;<CRD>
Example Response:	`40;1.9500;0.0;120;2.1750;` `–20.0;100.0;120;2.1750;10.0;STOP`

The sequential tones settings are:

Function Tone Duration	40 ms
Function Tone Frequency	1.9500 kHz
Function Tone Level	0.0 dB
Guard Tone Duration	120 ms
Guard Tone Frequency	2.1750 kHz
Guard Tone Level	–20 dB
Level Reference	100.0 mV
Max Tone Duration	120 ms
Max Tone Frequency	2.1750 kHz
Max Tone Level	10.0 dB
Mode	STOP

:TONES

:Afdecodelevel

Description: Sets the maximum audio input level, in rms volts, when receiving audio tones

Parameters: <CPD> or <NRf>
Maximum audio input level

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or AFDECODE_20MV
1 or AFDECODE_40MV
2 or AFDECODE_100MV
3 or AFDECODE_200MV
4 or AFDECODE_400MV
5 or AFDECODE_1V
6 or AFDECODE_2V
7 or AFDECODE_4V
8 or AFDECODE_10V
9 or AFDECODE_20V
10 or AFDECODE_40V

Example: `TONES:AFDECODELEVEL AFDECODE_100MV`
Sets the audio input level to 100 mV rms

:Afdecodelevel?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `AFDECODE_100MV`

:TONES

:Decodepath

Description: Sets the signal routing to the tones decoder

Parameters: <CPD> or <NRf>
Decode path

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or RF
1 or AF

Example: `TONES:DECODE RF`
Selects the demodulated RF signal as input to the tones decoder

:Decodepath?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `RF`

:TONES

:Encodepath

Description: Sets the signal routing from the tones encoder

Parameters: <CPD> or <NRf>
Encode path

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or RF
1 or AF

Example: `TONES:ENCODE RF`
Send the tones modulated on the RF signal

:Encodepath?

Parameters: N/A

Response: <CRD>
Encode path

Example Response: `RF`

:TONES

:Function

Description: Sets the function of the instrument in tones to either encode or decode

Parameters: <CRD> or <NRf>
Function

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or ENCODE
1 or DECODE

Example: `TONES:FUNC ENCODE`

:Function?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `ENCODE`

:TONES

:Reflevel

Description: Sets the receiver reference level when receiving RF tones. These choices are for the Antenna input. The equivalent reference level at the N-type input is 26 dB higher.

Parameters: <CPD> or <NRf>
Ref Level choice

Allowed suffices: N/A

Default suffix: N/A

Valid Data:
0 or REF_M50DBM
1 or REF_M40DBM
2 or REF_M30DBM
3 or REF_M20DBM
4 or REF_ M10DBM
5 or REF_0DBM
6 or REF_10DBM
7 or REF_20DBM
8 or REF_30DBM

Example: `TONES:REF REF_20DBM`

Sets input reference level to 20 dBm at the Antenna or 46 dBm at the N-type

:Reflevel?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `REF_20DBM`

:TONES

:Type

Description: Sets the type of tones currently used by the instrument

Parameters: <CPD> or <NRf>
Type

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or SEQ
1 or DTMF
2 or POCSAG
3 or DCS
4 or SELCAL
5 or ILS
6 or MKR
7 or VOR
8 or TONEREM

Example: `TONES:TYPE SEQ`

:Type?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `SEQ`

:TONES?

Description: Queries the status of TONES by producing the combined return values of the sub commands of TONES

These responses are separated by semi-colons

Parameters: N/A

Response: <CRD>,<CRD>,<CRD>,<CRD>,<CRD>,<CRD>

Example Response: `AFDECODE_100MV;RF;RF;ENCODE;REF_20DBM;SEQ;`

The tones settings are:

AF decode level, 100 mV
Decode path, RF
Encode path, RF
Function set; Encode
RF level set, 20 dBm
Type selected, Sequential.

:TRansient

Controls the instrument RF transient recorder

Not used alone

:TRansient

:Arm

Description: Arms the transient analysis trace. This is an action only.

Parameters: N/A

Allowed suffices: N/A

Default suffix: N/A

Valid Data: N/A

Example: `TRANS:ARM`

:TRansient

Note: Use :MEASUre:MKr1? to return the signal level at the marker.

:MArker

Description: Controls the status of the transient analyzer marker

Parameters: <CPD> or <NRf>

Marker status

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or ON

Example: `TRANS:MARK ON`

:MArker?

Parameters: N/A

Response: <CRD>

Marker status

Example Response: `ON`

:TRansient

:MKrtime

Description: Sets the marker position on the transient analyzer screen relative to the trigger point

Parameters: <NRf>
Marker position in time

Allowed suffices: S,MS,US

Default suffix: US

Example: `TRANS:MKRT –10MS`

:MKrtime?

Parameters: N/A

Response: <NR1>
Time (uS)

Example Response: `500`

:TRansient

:POlarity

Description: Controls the transient analyzer trigger polarity

Parameters: <CPD> or <NRf>
Trigger polarity

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or NEGATIVE
1 or POSITIVE

Example: `TRANS:POL POS`

:POlarity?

Parameters: N/A

Response: <CRD>
Trigger polarity

Example Response: `POSITIVE`

:TRansient

:PRetrig

Description: Controls the percentage of transient analyzer display showing pre-trigger period

Parameters: <CPD> or <NRf>*Pretrigger selection*

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or PRE_0
1 or PRE_25
2 or PRE_50
3 or PRE_75
4 or PRE_100

Example: `TRANS:PRE PRE_25`

:PRetrig?

Parameters: N/A

Response: <CRD>
Percentage pre-trigger display period

Example Response: `PRE_25`

:TRansient

:Reflevel

Description: Sets the reference level (top of screen) of the transient analyzer

Parameters: <NRf>
Reference level

Allowed suffices: DBM

Default suffix: DBM

Example: `TRANS:REFLEV 10DBM`

:Reflevel?

Parameters: N/A

Response: <NR2>
Reference level (dBm)

Example Response: `10.0`

:TRansient

:State?

Description: Returns the current state of the transient analyzer

Parameters: N/A

Response: <CRD>

Responses: ARMED,TRIGGERED or STORED

Example Response: `STORED`

:TRansient

:TBase

Description: Controls the timebase of the transient analyzer

Parameters: <CPD> or <NRf>
Range selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data:
0 or TA_50US
1 or TA_100US
2 or TA_200US
3 or TA_500US
4 or TA_1MS
5 or TA_2MS
6 or TA_5MS
7 or TA_10MS
8 or TA_20MS
9 or TA_50MS
10 or TA_100MS
11 or TA_200MS
12 or TA_500MS
13 or TA_1S
14 or TA_2S
15 or TA_5S

Example: `TRANS:TBASE 4`

:TBase?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `TA_1MS`

:TRansient

:TRglevel

Description: Sets the trigger level (relative to the top of screen) of the transient analyzer

Parameters: <NRf>
Trigger level

Allowed suffices: DB

Default suffix: DB

Example: `TRANS:TRGLEV −10DB`

:TRglevel?

Parameters: N/A

Response: <NR2>
Trigger level (dB)

Example Response: `−10.0`

:TRansient?

Description: Queries the status of the transient analyzer by producing the combined return values of the sub commands of TRANSIENT

These responses are separated by semi-colons

Parameters: N/A

Response: <CRD>;<NR1>;<CRD>;<CRD>;<NR2>;<CRD>;
<CRD>;<NR2>

Example Response: `ON;500;POSITIVE;PRE_25;10.0;STORED;`
`TA_IMS;−10.0`

:TXDIsp

Description Controls the type of data display used in TX test mode

Parameters: <CPD> or <NRf>
Display selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data 0 or BARCHARTS
1 or SCOPE
2 or LARGE_SCOPE

Example: `TXDISP SCOPE`

:TXDIsp?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `SCOPE`

:TXDNotch

Description: Sets the demod notch frequency

Parameters: <NRf>
Frequency (kHz)

Allowed suffices: N/A

Default suffix: N/A

Valid Data: One of the factory-set demod notch frequencies (Option 29) or 1 kHz

Example: `TXDNOTCH 0.9`

:TXDNotch?

Parameters: N/A

Response: <NR2>
Current set demod notch frequency (kHz)

Example Response: `0.9`

:TXDType

Description: Controls the distortion measuring method when in Transmitter test mode

Parameters: <CPD> or <NRf>
Distortion measurement type

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or OFF
1 or DISTN
2 or SINAD
3 or SN

Example: `TXDTYPE SINAD`

:TXDType?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `SINAD`

:TXFilt

Description: Controls the demodulation bandwidth filtering when in the transmitter test mode

Note. This command is retained for compatibility with the 2945A command set. Use MODFILT:FILTER.

Parameters: <CPD> or <NRf>
Filter selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or LP_50KHZ
1 or LP_15KHZ
2 or STD_BP
3 or LP_300HZ
4 or LP_3KHZ
5 or HP_300HZ
6 or PSOPH (CMESS or CCITT as fitted)

Example: `TXFILT 1`

:TXFilt?

Parameters: N/A

Response: <CRD>
Current selection

Note. If the current demod filter is not one of the standard 2945A filters then a null-string is returned.

Example Response: `LP_15KHZ`

:UNitmeas

Controls displayed measurement units

Not used alone

:UNitmeas

:Aflevel

Description: Controls the displayed units for the audio level measurement

Parameters: <CPD> or <NRf>
Units selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or AFL_VOLTS
1 or AFL_DBV
2 or AFL_DBM
3 or AFL_DBR
4 or AFL_WATTS

Example: `UNIT:AFLEV AFL_VOLTS`

:Aflevel?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `AFL_VOLTS`

:UNitmeas

:DBRAFref

Description: Sets the reference level for Audio dB Relative measurements and turns on dBr.

Parameters: <NRf>
Level

Allowed suffices: MV, V, DBM

Default suffix: MV

Example: `:UNitmeas:DBRAFref 200MV`

:DBRAFref?

Parameters: N/A

Response: <NR2>
Level (mV)

Example Response: `200.0`

:UNitmeas

:DBRAMref

Description: Sets the reference level for AM dB Relative modulation measurements and turns on dBr.

Parameters: <NRf>
Level

Allowed suffices: PCT

Default suffix: PCT

Example: `:UNitmeas:DBRAMref 20PCT`

:DBRAMref?

Parameters: N/A

Response: <NR2>
Level (%)

Example Response: `20.0`

:UNitmeas

:DBRFMref

Description: Sets the reference level for FM dB Relative modulation measurements and turns on dBr.

Parameters: <NRf>
Level

Allowed suffices: HZ, KHZ

Default suffix: HZ

Example: `:UNitmeas:DBRFMref 1500HZ`

:DBRFMref?

Parameters: N/A

Response: <NR2>
Level (Hz)

Example Response: `1500`

:UNitmeas

:Extimp

Description: Sets the External Impedance Value used in the calculation of the AF Power for Audio measurements

Parameters: <CPD> or <NRf>
Range selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or IMP_4
1 or IMP_8
2 or IMP_16
3 or IMP_75
4 or IMP_100
5 orIMP_150
6 orIMP_300
7 orIMP_600

Example: `:UNitmeas:Extimp IMP_600`

:Extimp?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `IMP_600`

:UNitmeas

:HOLDAfpk

Description: Controls whether the peak hold facility for audio level measurements is ON or OFF.

Parameters: <CPD>
Audio level peak hold state

Allowed suffices: N/A

Default suffix: N/A

Valid Data: OFF
ON

Example: `UNIT:HOLDAfpk OFF`

:HOLDAfpk?

Parameters: N/A

Response: <CRD>
Audio level peak hold state

Example Response: `OFF`

:UNitmeas

:HOLDModpk

Description: Controls whether the peak hold facility for modulation level measurements is ON or OFF.

Parameters: <CPD>
Mod level peak hold state

Allowed suffices: N/A

Default suffix: N/A

Valid Data: OFF
ON

Example: `UNIT:HOLDModpk OFF`

:HOLDModpk?

Parameters: N/A

Response: <CRD>
Mod level peak hold state

Example Response: `OFF`

:UNitmeas

:MODDbr

Description: Controls whether the measurement units are provided as dBr (ON) or absolute (OFF)

Parameters: <CPD>
dBr measurement mode

Allowed suffices: N/A

Default suffix: N/A

Valid Data: OFF
ON

Example: `UNIT:MODDbr OFF`

:MODDbr?

Parameters: N/A

Response: <CRD>
dBr measurement mode

Example Response: `OFF`

:UNitmeas

:MODLevel

Description: Controls whether FM modulation measurements are made using RMS or peak detectors

Parameters: <CPD> or <NRf>
Measurement mode

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or PEAK
1 or RMS

Example: `UNIT:MODLevel RMS`

:MODLevel?

Parameters: N/A

Response: <CRD>
Measurement mode

Example Response: `RMS`

:UNitmeas

:Rflevel

Description: Controls the displayed units for the RF level measurement

Parameters: <CPD> or <NRf>
Units selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or RFL_DBM
1 or RFL_VOLTS
2 or RFL_WATTS

Example: `UNIT:RFLEVEL RFL_DBM`

:Rflevel?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `RFL_DBM`

:UNitmeas?

Description: Queries the measurement units by producing the combined return values of the sub commands of UNITMEAS

These responses are separated by semi-colons

Parameters: N/A

Response: <CRD>;<NR2>;<NR2>;<NR2>;<CRD>;<CRD>;<CRD>;<CRD>

Example Response: `AFL_VOLTS;200.0;20.0;1500;IMP_600;ON; PEAK;RFL_DBM`

:USeroptions

Controls user selections

Not used alone

:USeroptions

:Extatten

Description: Controls the value of assumed external attenuation

Parameters: <NRf>

External attenuation

Allowed suffices: DB

Default suffix: DB

Example: `USER:EXTATTEN 20DB`

:Extatten?

Parameters: N/A

Response: <NR2>

External attenuation

Example Response: `20.0`

:USeroptions

:Notch?

Parameters: N/A

Response: <NR2>

Frequencies of distortion notch filters Option 29

Example Response: `150.00, 900.00`

:USeroptions

:PRINTPort

Description:	Controls the current printer port selection
Parameters:	<CPD> or <NRf> *Printer port*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or PARALLEL 1 or SERIAL
Example:	`USER:PRINTP SERIAL`

:PRINTPort?

Parameters:	N/A
Response:	<CRD> *Printer port*
Example Response:	`SERIAL`

:USeroptions

:PRINTType

Description:	Controls the current printer type selection
Parameters:	<CPD> or <NRf> *Printer type*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or EPSON80 1 or EPSON100 2 or LASER75 3 or LASER100 4 or LASER150
Example:	`USER:PRINTT LASER75`

:PRINTType?

Parameters:	N/A
Response:	<CRD> *Printer* type
Example Response:	`LASER75`

:USeroptions

:RS232

:Baudrate

Description: Controls the current RS232 baudrate selection

Parameters: <CPD> or <NRf>
Baudrate

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or BAUD_9600
1 or BAUD_4800
2 or BAUD_2400
3 or BAUD_1200
4 or BAUD_600
5 or BAUD_300
6 or BAUD_150
7 or BAUD_75

Example: `USER:RS232:BAUD BAUD_9600`

:RS232

:Baudrate?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `BAUD_9600`

:USeroptions

:RS232

:Charlen

Description: Controls the current RS232 character length

Parameters: <CPD> or <NRf>
Character length

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or BITS_8
1 or BITS_7

Example: `USER:RS232:CHAR BITS_8`

:RS232

:Charlen?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `BITS_8`

:USeroptions

:RS232

:Handshake

Description: Controls the current RS232 handshaking method

Parameters: <CPD> or <NRf>
Handshaking method

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or SOFTWARE
1 or HARDWARE

Example: `USER:RS232:HANDSHAKE SOFTWARE`

:RS232

:Handshake?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `SOFTWARE`

:USeroptions

:RS232

:Parity

Description: Controls the current RS232 parity bits

Parameters: <CPD> or <NRf>
Parity

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or NONE
1 or ODD
2 or EVEN

Example: `USER:RS232:PARITY NONE`

:RS232

:Parity?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `NONE`

:USeroptions

:RS232

:Stopbits

Description: Controls the current RS232 stop bits selection

Parameters: <CPD> or <NRf>
Stop bits

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or BITS_1
1 or BITS_2

Example: `USER:RS232:STOPBITS BITS_1`

:RS232

:Stopbits?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `BITS_1`

:USeroptions

:RS232?

Description: Queries the user selected options by producing the combined return values of the sub commands of USEROPTIONS:RS232

These responses are separated by semi-colons

Parameters: N/A

Response: <CRD>;<CRD>;<CRD>;<CRD>;<CRD>

Example Response: `BAUD_9600;BITS_8;SOFTWARE;NONE;BITS_1`

:USeroptions

:RXDavgs

Description: Controls the number of measurements over which RX distortion, SINAD and SN are averaged

Parameters: <NRf>

Number of averages

Allowed suffices: N/A

Default suffix: N/A

Example: `USER:RXDAV 10`

:RXDavgs?

Parameters: N/A

Response: <NR1>

Number of RX distortion measurements averaged

Example Response: `10`

:USeroptions

:RXEqtxoffset

Description: Controls the duplex offset value

Parameters: <NRf>

frequency

Allowed suffices: MHZ,KHZ

Default suffix: MHZ

Example: `USER:RXEQTX  –10.0`

:RXEqtxoffset?

Parameters: N/A

Response: <NR2>

Offset in MHz

Example Response: `–10.000000`

:USeroptions

:RXNavgs

Description: Controls the number of noise samples in AF noise and distortion measurements.

Parameters: <NRf>

number of samples, in tens

Allowed suffices: N/A

Default suffix: N/A

Example: `USER:RXNAVGS 3` *(30 averages)*

:RXNavgs?

Parameters: N/A

Response: <NR2>

Number of samples

Example Response: `3`

:USeroptions?

Description: Queries the user selected options by producing the combined return values of the sub commands of USEROPTIONS

These responses are separated by semi-colons

Parameters: N/A

Response: <NR2>;<CRD>;<CRD>;<CRD>;<CRD>;<CRD>;<CRD>;<CRD>;<NR1>;<NR2>

Example Response: `20.0;SERIAL;LASER_75;BAUD_9600;BITS_8;SOFTWARE;NONE;BITS_1;10,−10.000000`

:Vorgen

Controls the VOR signal generator

Not used alone

:Vorgen

:BEARIng

Description: Sets the bearing in VOR mode

Parameters: <NRf>
Bearing

Allowed suffices: DEG

Default suffix: DEG

Example: `VOR:BEARING 172.2`

:BEARIng?

Parameters: N/A

Response: <NR2>
Bearing (degrees)

Example Response: `172.2`

:Vorgen

:BEARRef

Description: Sets the bearing reference either to or from the beacon

Parameters: <CPD> or <NRf>
Bearing reference

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or TO
1 or FROM

Example: `VOR:BEARREF FROM`

:BEARRef?

Parameters: N/A

Response: <CRD>
Bearing reference

Example Response: `FROM`

:Vorgen

:Idepth

Description:	Sets the ident signal modulation depth
Parameters:	<NRf> *Depth*
Allowed suffices:	PCT
Default suffix:	PCT
Example:	`VOR:IDEPTH 20.0`

:Idepth?

Parameters:	N/A
Response:	<NR2> *Ident depth (%)*
Example Response:	`20.0`

:Vorgen

:Levlock

Description:	Controls whether the reference carrier and subcarrier levels are locked together
Parameters:	<CPD> or <NRf> *Level lock selection*
Allowed suffices:	N/A
Default suffix:	N/A
Valid Data:	0 or OFF 1 or ON
Example:	`VOR:LEVLOCK ON`

:Levlock?

Parameters:	N/A
Response:	<CRD> *Current selection*
Example Response:	`ON`

:Vorgen

:Mode

Description: Sets the VOR signal generated. Either adjustable bearing normal VOR or constant bearing with ident.

Parameters: <CPD> or <NRf>
Mode

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or VOR
1 or IDENT

Example: `VOR:MODE VOR`

:Mode?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `VOR`

:Vorgen

:REfdepth

Description: Sets the reference modulation depth

Parameters: <NRf>
Ref depth

Allowed suffices: PCT

Default suffix: PCT

Example: `VOR:REFDEPTH 30.0`

:REfdepth?

Parameters: N/A

Response: <NR2>
Ref depth (%)

Example Response: `30.0`

:Vorgen

:RFFreq

Description: Sets the RF frequency in VOR mode

Parameters: <NRf>
Frequency

Allowed suffices: MHz, kHz

Default suffix: MHz

Example: `VOR:RFREQ 107.9`

:RFFreq?

Parameters: N/A

Response: <NR2>
Frequency (MHz)

Example Response: `107.900000`

:Vorgen

:RFLevel

Description: Sets the RF output level in VOR mode

Parameters: <NRf>
Level

Allowed suffices: DBM,DBUV,UV,MV

Default suffix: DBM

Example: `VOR:RFLEVEL –80DBM`

:RFLevel?

Parameters: N/A

Response: <NR2>
Level (dBm)

Example Response: `–80.0`

:Vorgen

:RFOut

Description: Sets the RF output in VOR mode

Parameters: <CPD> or <NRf>
Output selection

Allowed suffices: N/A

Default suffix: N/A

Valid Data: 0 or GEN_N
1 or GEN_BNC

Example: `VOR:RFOUT 0`

:RFOut?

Parameters: N/A

Response: <CRD>
Current selection

Example Response: `GEN_N`

:Vorgen

:Subdepth

Description: Sets the sub-carrier modulation depth

Parameters: <NRf>
Sub-carrier depth

Allowed suffices: PCT

Default suffix: PCT

Example: `VOR:SUBDEPTH 30.0`

:Subdepth?

Parameters: N/A

Response: <NR2>
Sub-carrier depth (%)

Example Response: `30.0`

:Vorgen?

Parameters: N/A

Description: Queries the status of the VOR signal generator by producing the combined return values of the sub commands of VORGEN

These responses are separated by semi-colons

Response: <NR2>;<CRD>;<NR2>;<CRD>;<CRD>;<NR2>;<NR2>;
<NR2>;<CRD>;<NR2>

Example Response: `172.2;FROM;20.0;ON;VOR;30.0;`
`107.900000;−80.0;GEN_N;30.0`

Chapter 6
SYSTEM COMMANDS

Contents

Introduction

This chapter describes System commands, that is, those commands that are specifically intended for use in System test programs.

Command abbreviations

The command syntax provided in this chapter, as in Chapter 5, indicates the minimum number of characters that must be entered for a command to be valid by showing them in uppercase. For example, the command :PROg:SEquence can be entered as just :PRO:SE. Conversely, the complete command, :PROG: SEQUENCE, is equally valid, and more readily understandable. Another consideration is that the addition of new commands in future development could invalidate the currently acceptable minimum number of characters. It is therefore recommended that, where possible, the complete command is used.

Program download and run commands

:PROg

:LEARN

Description : Set the test set into state where an MI-BASIC program can be downloaded

Parameters : <CPD> ACTIVE
INACTIVE

Allowed suffices : Not applicable

Example : `:PROg:LEARN ACTIVE`

:LEARN?

Description : Returns the MI-BASIC program download status of the test set

Response : <CRD> ACTIVE
INACTIVE

Example : `:PROg:LEARN?`

Example response : `PR:LEARN ACTIVE`

:PROg

:LEARNLine

Description : Downloads the next line of an MI-BASIC program

Parameters : <ASCII data>

Allowed suffices : Not applicable

Example : `:PROg:LEARNLine "REM Testing Testing 123"`
`(See Part 1, Chapter 1, for more examples)`

:PROg

:Tsi or :SEquence

See note below

Description : Selects the test program to be run

Parameters : <CPD> CALL_PROC
CALL_RF
BRIEF
COMPREHENSIVE
USER
GEN_PMR
GENERAL
QUICK

Allowed suffices : Not applicable

Examples :
```
:PROg:Tsi CALL_PROC
:PROg:SEquence CALL_PROC
```

:Tsi? or :SEquence?

Description : Returns the currently selected test program

Response : <CRD> CALL_PROC
CALL_RF
BRIEF
COMPREHENSIVE
USER
GEN_PMR
GENERAL
QUICK

Example :
```
:PROg:Tsi?
:PROg:SEquence?
```

Example response :
```
PR:T CALL_PROC
PR:SE CALL_PROC
```

Note: All programs are not applicable to all systems. The table below shows utilization.

PROGRAMS	PMR	AMPS	TACS	NMT	MPT 1327	EDACS Radio	EDACS Repeater	LTR Radio	LTR Repeater
CALL_PROC		✓	✓	✓	✓	✓		✓	
CALL_RF		✓	✓	✓	✓	✓		✓	
BRIEF		✓	✓	✓	✓	✓		✓	
COMPREHENSIVE		✓	✓	✓	✓	✓		✓	
GEN_PMR	✓								
GENERAL							✓		✓
QUICK							✓		✓
USER	✓	✓	✓	✓	✓	✓	✓	✓	✓

:PROg

:PAuse

Description : Sets the test set's Auto Test pause state

Parameters : <CPD> OFF
ON_FAILURE
ALWAYS

Allowed suffices : Not applicable

Example : `:PROg:PAuse OFF`

:PAuse?

Description : Returns the current setting of the test set's Auto Test pause state

Response : <CPR> OFF
ON_FAILURE
ALWAYS

Example : `:PROg:PAuse?`

Example response : `PR:PA OFF`

:PROg

:Runstate

Description : Controls the running of the test set's Auto Test program

Parameters : <CPD> STOP
RUN
PAUSE
CONTINUE

Allowed suffices : Not applicable

Example : `:PROg:Runstate RUN`
Run the program

: Runstate?

Description : Returns the current setting of the Auto Test program control

Response : <CRD> STOP
RUN
PAUSE
CONTINUE

Example : `:PROg:Runstate?`

Example response : `PR:R RUN`

:PROg

:Numresults?

Description : Returns the numerical test results

Response : <CRD> A single ASCII string with each of the 8 numerical results, comma-separated:-

Data field 1, PASSED
Data field 2, ACTUAL
Data field 3, TARGET
Data field 4, LOWER
Data field 5, UPPER
Data field 6, AUX1
Data field 7, AUX2
Data field 8, AUX3

Example : `:PROg:Numresults?`

Example response : `PR:N 0,38.638,0.000,25.000,0.000,0.000, 0.000,0.000`
Response to RxSINAD test where the test passed at 38.6 dB with a minimum pass value of 25 dB

:PROg

:STrresults?

Description : Returns the string test results

Response : <CRD> A single ASCII string with each of the 8 string results, comma-separated:-

Data field 1, TITLE
Data field 2, STATUS
Data field 3, COMMENT1
Data field 4, COMMENT2
Data field 5, AUX1
Data field 6, AUX2
Data field 7, AUX3
Data field 8, AUX4

Example : `:PROg:STrresults?`

Example response : `PR:ST REGISTRATION,PASSED,MIN:000-0-000000,, ESN:15/20/00/26346STD1,CLASS:4,DTX:0, CHANNELS:1320`
Response to a TACS registration test

Test commands

The following :STEst commands allow you to run the System tests remotely. When the command is executed, the screen changes (if necessary) to the AUTORUN screen and the test executes in the normal way. Details of the tests can be found in the Operating Manual Supplement for the particular system in use.

If the test passes, a value of '1' is returned. Otherwise the test fails and a value of '0' is returned. To get the test results, use the :PROg:Numresults? or :PROg:STrresults? command after the test has completed. A list showing other :PROg commands is given earlier in this chapter (see page 6-1)

List of commands

:STEst

Used on all systems

:AFFreq?

Description :	Runs the Audio Frequency measurement test. The test passes if the result is within the allowed % error.
Response :	<NR1> 0 or 1
Example :	`:STEst:AFFreq?`
Example response :	`1`
Channel type :	Traffic, Voice or Working only

:STEst

Used on all systems

:AFLevel?

Description : Runs the AF Level test. The test passes if the result is within the allowed % error.

Response : <NR1> 0 or 1

Example : `:STEst:AFLevel?`

Example response : `1`

Channel type : Traffic, Voice or Working only

:STEst

Used on LTR Repeater and LTR Radio

:DATADevn?

Description : Runs the Data Deviation test. The test passes if the result is better than the allowed % error.

Response : <NR1> 0 or 1

Example : `:STEst:DAtadevn?`

Example response : `1`

Channel type : N/A

:STEst

Used on AMPS, TACS, EDACS Repeater and EDACS Radio

:DATAPerform?

Description : Runs the Data Performance test. The test passes if the result is better than the allowed % error.

Response : <NR1> 0 or 1

Example : `:STEst:DAtaperform?`

Example response : `1`

Channel type : Wide-band Voice only {AMPS/TACS}
Control only {EDACS}

:STEst

Used on NAMPS and NTACS

:DSatdevn?

Description : Runs the Digital SAT Deviation test. The test passes if the result is within the allowed % error.

Response : <NR1> 0 or 1

Example : `:STEst:DSatdevn?`

Example response : `1`

Channel type : Narrow-band Voice only

:STEst

Used on AMPS, NAMPS, TACS, NTACS, PMR, EDACS Radio and LTR Radio

:DTmfdecode?

Description : Runs the DTMF Decode test. The test passes if the required tone sequence is received correctly within the timeout period.

Response : <NR1> 0 or 1

Example : `:STEst:DTmfdecode?`

Example response : `1`

Channel type : Traffic, Voice or Working only

:STEst

Used on all systems

:Fmdevn?

Description : Runs the FM Deviation test. The test passes if the result is within the allowed % error.

Response : <NR1> 0 or 1

Example : `:STEst:Fmdevn?`

Example response : `1`

Channel type : Traffic, Voice or Working only

:STEst

Used on AMPS, NAMPS, TACS, NTACS, MPT1327, and NMT

:HAndoff?

Description :	Runs the Handoff test. The test passes if a handoff to the specified channel is completed within the timeout period.
Parameter :	<NR1> Channel number
Response :	<NR1> 0 or 1
Example :	`:STEst:HAndoff? 100`
Example response :	`1`
Channel type :	Traffic or Voice only

:STEst

Used on AMPS, NAMPS, TACS, and NTACS

:HOokflash?

Description :	Runs the Hookflash test. The test passes if a hookflash message is received within the timeout period.
Response :	<NR1> 0 or 1
Example :	`:STEst:HOokflash?`
Example response :	`1`
Channel type :	Traffic or Voice only

:STEst

Used on EDACS Repeater

:HSdevn?

Description :	Runs the High Speed Data Deviation test. The test passes if the result is in the allowed % error.
Response :	<NR1> 0 or 1
Example :	`:STEst:HSdevn?`
Example response :	`1`
Channel type :	Control only

:STEst

Used on AMPS, NAMPS, TACS, NTACS, MPT1327, NMT and EDACS Radio

:LAndclear?

Description : Runs the Clear From Land test. The test passes if the mobile is cleared down within the timeout period.

Response : <NR1> 0 or 1

Example : `:STEst:LAndclear?`

Example response : `1`

Channel type : Traffic, Voice or Working only

:STEst

Used on LTR Radio

:LISTENON?

Description : Runs the Listen On test. The test passes if the mobile goes into listen mode within the timeout period.

Response : <NR1> 0 or 1

Example : `:STEst:LISTENON?`

Example response : `1`

Channel type : N/A

:STEst

Used on LTR Radio

:LISTENOFf?

Description : Runs the Listen Off test. The test passes if the mobile is cleared down within the timeout period.

Response : <NR1> 0 or 1

Example : `:STEst:LISTENOFf?`

Example response : `1`

Channel type : N/A

:STEst

Used on EDACS Repeater and EDACS Radio

:LSdevn?

Description : Runs the Low Speed Data Deviation Test. The test passes if the result is within the allowed % error.

Response : <NR1> 0 or 1

Example : `:STEst:LSdevn?`

Example response : `1`

Channel type : Working only

:STEst

Used on AMPS, NAMPS, TACS, NTACS, MPT1327, NMT and EDACS Radio

:MOBileclear?

Description : Runs the Clear From Mobile test. The test passes if the mobile clears down within the timeout period.

Response : <NR1> 0 or 1

Example : `:STEst:MOBileclear?`

Example response : `1`

Channel type : Traffic, Voice or Working only

:STEst

Used on all systems

:MODfreq?

Description : Runs the Modulation Frequency test. The test passes if the result is within the allowed % error.

Response : <NR1> 0 or 1

Example : `:STEst:MODfreq?`

Example response : `1`

Channel type : Traffic, Voice or Working only

:STEst

Used on AMPS, NAMPS, TACS, NTACS, MPT1327, NMT, and EDACS Radio

:PAgemobile?

Description : Runs the Page Mobile test. The test passes if the mobile goes into "conversation" within the timeout period.

Response : <NR1> 0 or 1

Example : `:STEst:PAgemobile?`

Example response : `1`

Channel type : Control or Combined CC/TC (NMT only) or Calling (NMT only).

:STEst

Used on AMPS, NAMPS, TACS, NTACS, MPT1327, NMT, and EDACS Radio

:PLacecall?

Description : Runs the Place Call test. The test passes if the mobile goes into "conversation" within the timeout period.

Response : <NR1> 0 or 1

Example : `:STEst:PLacecall?`

Example response : `1`

Channel type : Control *(All systems)*
or Traffic *(NMT only)*
or Combined CC/TC or Access *(NMT)*.

:STEst

Used on AMPS, NAMPS, TACS, NTACS, and NMT

:POwerlevel?

Description : Runs the Power Level test. The test passes if the result is within the allowed % error

Parameter : <NR1> Range dependent on system selected;
0-7 for AMPS and TACS; 0-3 for NMT

Response : <NR1> 0 or 1

Example : `:STEst:POwerlevels? 3`

Example response : `1`

Channel type : Traffic or Voice only

:STEst

Used on MPT1327, PMR , EDACS Radio and LTR Radio

:PTTOFf?

Description : Tests that the mobile's Press To Talk (PTT) button is in the released (OFF) state. The test passes if the mobile stops transmitting within the timeout period.

Parameter : <NR1> 0 or 1

Example : `:STEst:PTTOFf?`

Example response : `1`

Channel Type : Traffic only

:STEst

Used on MPT1327, PMR and EDACS Radio and LTR Radio

:PTTON?

Description : Tests that the mobile's Press To Talk (PTT) button is in the pressed (ON) state. The test passes if the mobile starts transmitting within the timeout period.

Response : <NR1> 0 or 1

Example : `:STEst:PTTON?`

Example response : `1`

Channel type : Traffic only

:STEst

Used on EDACS Repeater

:RADIOCALL?

Description : Runs the Call From Radio test. The test passes if the base station goes to a working channel within the timeout period.

Response : <NR1> 0 or 1

Example : `:STEst:RADIOCALL?`

Example response : `1`

Channel type : Control only

:STEst

Used on EDACS Repeater

:RADIOCLear?

Description :	Runs the Clear From Radio test. The test passes if the base station clears down within the timeout period.
Response :	<NR1> 0 or 1
Example :	`:STEst:RADIOCLear?`
Example response :	`1`
Channel type :	Working only

:STEst

Used on LTR Repeater

:RADIOPTTON?

Description :	Runs the PTT Off From Radio test. The test passes if the repeater goes into conversation within the timeout period.
Response :	<NR1> 0 or 1
Example :	`:STEst:RADIOPTTON?`
Example response :	`1`
Channel type :	N/A

:STEst

Used on LTR Repeater

:RADIOPTTOFf?

Description :	Runs the PTT On From Radio test. The test passes if the repeater goes idle within the timeout period.
Response :	<NR1> 0 or 1
Example :	`:STEst:RADIOPTTOFf?`
Example response :	`1`
Channel type :	N/A

:STEst

Used on EDACS Repeater and LTR Repeater

:RFDistn?

Description : Runs the RF through-path distortion test. The test passes if the result is better than the upper % limit.

Response : <NR1> 0 or 1

Example : `:STEst:RFDistn?`

Example response : `1`

Channel type : Working only

:STEst

Used on EDACS Repeater and LTR Repeater

:RFSInad?

Description : Runs the RF through-path SINAD test. The test passes if the result is better than the lower dB limit.

Response : <NR1> 0 or 1

Example : `:STEst:RFSInad?`

Example response : `1`

Channel type : Working only

:STEst

Used on EDACS Repeater and LTR Repeater

:RFSN?

Description : Runs the RF through-path S/N test. The test passes if the result is better than the lower dB limit.

Response : <NR1> 0 or 1

Example : `:STEst:RFSN?`

Example response : `1`

Channel type : Working only

:STEst

Used on AMPS, NAMPS, TACS, NTACS, MPT1327, and NMT

:REgister?

Description : Will test that the mobile can register. The test passes if the mobile registers within the timeout period.

Response : <NR1> 0 or 1

Example : `:STEst:REgister?`

Example response : `1`

Channel type : Control or Combined CC/TC (NMT only).

:STEst

Used on all systems

:RXDistn?

Description : Runs the Receiver Distortion test. The test passes if the result is better than the allowed % limit.

Response : <NR1> 0 or 1

Example : `:STEst:RXDistn?`

Example response : `1`

Channel type : Traffic, Voice or Working only

:STEst

Used on all systems

:RXExpand?

Description : Runs the Receiver Expansion test. The test passes if the result is within the allowed % error.

Response : <NR1> 0 or 1

Example : `:STEst:RXExpand?`

Example response : `1`

Channel type : Traffic, Voice or Working only

:STEst

Used on all systems

:RXSEns?

Description : Runs the Receiver Sensitivity test. The test passes if the result is better than the RF upper dBm limit.

Response : <NR1> 0 or 1

Example : `:STEst:RXSEns?`

Example response : `1`

Channel type : Traffic, Voice or Working only

:STEst

Used on all systems

:RXSInad?

Description : Runs the Receiver SINAD test. The test passes if the result is better than the lower dB limit.

Response : <NR1> 0 or 1

Example : `:STEst:RXSInad?`

Example response : `1`

Channel type : Traffic, Voice or Working only

:STEst

Used on all systems

:RXSN?

Description : Runs the Receiver S/N test. The test passes if the result is better than the lower dB limit.

Response : <NR1> 0 or 1

Example : `:STEst:RXSN?`

Example response : `1`

Channel type : Traffic, Voice or Working only

:STEst

Used on AMPS, TACS, and NMT

:SATDevn?

Description : Runs the SAT Deviation test. The test passes if the result is within the allowed % error.

Response : <NR1> 0 or 1

Example : `:STEst:SATDevn?`

Example response : `1`

Channel type : Wide-band Voice or Traffic only

:STEst

Used on AMPS, TACS, and NMT

:SATFreq?

Description : Runs the SAT Frequency test. The test passes if the result is within the allowed % error.

Response : <NR1> 0 or 1

Example : `:STEst:SATFreq?`

Example response : `1`

Channel type : Traffic or Voice only

:STEst

Used on AMPS and TACS

:STDEvn?

Description : Runs the ST Deviation test. The test passes if the result is within the allowed % error. This test requires that the test status is ON and the :STEst:PAgemobile test run. The Page Mobile test measures the ST deviation, which is then returned when this test is run. The test status can be set to ON from MI-BASIC or from the front panel

Response : <NR1> 0 or 1

Example : `:STEst:STDEvn?`

Example response : `1`

Channel type : Wide-band Voice only

:STEst

Used on AMPS and TACS

:STDUrn?

Description : Runs the ST Duration test. The test is performed against the most recent valid ST signalling context. For example, if the ST Duration test is requested after a hookflash on TACS, then the test will check that the ST was present for 400 ms.

The test passes if the result is within the allowed % error.

Response : <NR1> 0 or 1

Example : `:STEst:STDUrn?`

Example response : `1`

Channel type : Wide-band Voice only

:STEst

Used on AMPS and TACS

:STFreq?

Description : Runs the ST Frequency test. The test passes if the result is within the allowed % error. This test requires that the test status is ON and the :STEst:PAgemobile test run. The Page Mobile test measures the ST frequency, which is then returned when this test is run.

Response : <NR1> 0 or 1

Example : `:STEst:STFreq?`

Example response : `1`

Channel type : Wide-band Voice only

:STEst

Used on EDACS Repeater

:TESTSETCALL?

Description : Runs the Call From Service Monitor test. The test passes if the base station goes to a working channel within the timeout period.

Response : <NR1> 0 or 1

Example : `:STEst:TESTSETCALL?`

Example response : `1`

Channel type : Control only

:STEst

Used on EDACS Repeater

:TESTSETCLear?

Description : Runs the Clear From Service Monitor test. The test passes if the base station clears the call down within the timeout period.

Response : <NR1> 0 or 1

Example : `:STEst:TESTSETCLear?`

Example response : `1`

Channel type : Working only

:STEst

Used on LTR Repeater

:TESTSETPTTON?

Description : Runs the PTT On from Test Set test. The test passes if the repeater goes into conversation within the timeout period.

Response : <NR1> 0 or 1

Example : `:STEst:TESTSETPTTON?`

Example response : `1`

Channel type : N/A

:STEst

Used on LTR Repeater

:TESTSETPTTOFf?

Description : Runs the PTT Off from Test Set test. The test passes if the repeater goes to idle within the timeout period.

Response : <NR1> 0 or 1

Example : `:STEst:TESTSETPTTOFf?`

Example response : `1`

Channel type : N/A

:STEst

Used on all systems

:TXCompress?

Description :	Runs the Transmitter Compression test. The test passes if the result is within the allowed % error.
Response :	<NR1> 0 or 1
Example :	`:STEst:TXCompress?`
Example response :	`1`
Channel type :	Traffic, Voice or Working only

:STEst

Used on all systems

:TXDistn?

Description :	Runs the Transmitter Distortion test. The test passes if the result is better than the upper % limit.
Response :	<NR1> 0 or 1
Example :	`:STEst:TXDistn?`
Example response :	`1`
Channel type :	Traffic, Voice or Working only

:STEst

Used on all systems

:TXFreq?

Description :	Runs the Transmitter Frequency test. The test passes if the result is within the allowed % error.
Response :	<NR1> 0 or 1
Example :	`:STEst:TXFreq?`
Example response :	`1`
Channel type :	Traffic, Voice or Working only

:STEst

Used on all systems

:TXLEvel?

Description : Runs the Transmitter Level test. The test passes if the result is within the allowed dBm limits.

Response : <NR1> 0 or 1

Example : `:STEst:TXLEvel?`

Example response : `1`

Channel type : Traffic, Voice or Working only

:STEst

Used on all systems

:TXLImit?

Description : Runs the Transmitter Limiting test. The test passes if the result is better than the upper limit.

Response : <NR1> 0 or 1

Example : `:STEst:TXLImit?`

Example response : `1`

Channel type : Traffic, Voice or Working only

:STEst

Used on all systems

:TXModsens?

Description : Runs the Transmitter Modulation Sensitivity test. The test passes if the result is within the allowed levels.

Response : <NR1> 0 or 1

Example : `:STEst:TXModsens?`

Example response : `1`

Channel type : Traffic, Voice or Working only

:STEst

Used on all systems

:TXNoise?

Description : Runs the Transmitter Noise test. The test passes if the result is better than the upper noise limit.

Response : <NR1> 0 or 1

Example : `:STEst:TXNoise?`

Example response : `1`

Channel type : Traffic, Voice or Working only

:STEst

Used on all systems

:TXSInad?

Description : Runs the Transmitter SINAD test. The test passes if the result is better than the lower dB limit.

Response : <NR1> 0 or 1

Example : `:STEst:TXSInad?`

Example response : `1`

Channel type : Traffic, Voice or Working only

:STEst

Used on all systems

:TXSN?

Description : Runs the Transmitter S/N test. The test passes if the result is better than the lower dB limit.

Response : <NR1> 0 or 1

Example : `:STEst:TXSN?`

Example response : `1`

Channel type : Traffic, Voice or Working only

Setup commands

List of commands

:SYSetup

Used on TACS, LTR repeater and LTR radio

:AId

Description : Sets TACS/LTR Area Identification

Parameters : <NR1> Range 0 to 32767 (TACS)
Range 0 to 1 (LTR)

Allowed suffices : Not applicable

Example : `:SYSetup:AId 12345`
Sets area identification number to 12345

:SYSetup

Used on MPT1327

:ALohan

Description : Sets size of ALOHA frame for MPT1327

Parameters : <NR1> Range 1 to 15

Allowed suffices : Not applicable

Example : `:SYSetup:ALohan 5`
Sets the ALOHA frame length to 5 slots

:SYSetup

Used on NMT

:AReanumber

Description : Sets NMT Area Number

Parameters : <NR1> Range 0 to 4

Allowed suffices : Not applicable

Example : `:SYSetup:AReanumber 3`
Sets area number to 3

:SYSetup

Used on AMPS, NAMPS, TACS, NTACS, NMT, MPT1327, EDACS Repeater and EDACS Radio

:CC

Description : Sets the Control Channel for the selected system

Parameters : <NR1> Range 0 to 9999 depending on selected system

Allowed suffices : Not applicable

Example : `:SYSetup:CC 100`
Sets the control channel number to 100

:SYSetup

Used on MPT1327

:CDdc

Description : Sets an MPT1327 mobile radio's Called Disconnect count

Parameters : <NR1> Range 0 to 9

Allowed suffices : Not applicable

Example : `:SYSetup:CDdc 5`
Sets the called disconnect parameter to 5

:SYSetup

Used on NMT

:CGChaccess

Description : Sets an NMT900 system's calling access channel

Parameters : <CPD > ALL, GROUP_A or GROUP_B

Allowed suffices : Not applicable

Example : `:SYSetup:CGChaccess GROUP_B`
Sets the calling access channel to GROUP_B

:SYSetup

Used on MPT1327

:CGDc

Description : Sets an MPT1327 mobile radio's Calling Disconnect count

Parameters : <NR1> Range 0 to 9

Allowed suffices : Not applicable

Example : `:SYSetup:CGDc 5`
Sets the called disconnect parameter to 5

:SYSetup

Used on EDACS Radio and LTR Radio

:CHANnelinc

Description : Sets EDACS/LTR channel increment for the test

Parameters : <NR1> 1 to 24 depending on the selected variant

Allowed suffices : Not applicable

Example : `:SYSetup:CHANnelinc 3`
Sets the channel increment to 3

:SYSetup

Used on TACS

:CHARgerate

Description : Sets a TACS mobile's charge rate

Parameters : <NR1> Range 0 to 2047

Allowed suffices : Not applicable

Example : `:SYSetup:CHARgerate 5`
Sets the charge rate to 5

:SYSetup

Used on MPT1327

:CIdent

Description : Sets an MPT1327 mobile's Calling Identity

Parameters : <NR1> Range 0 to 8191

Allowed suffices : Not applicable

Example : `:SYSetup:CIdent 2000`
Sets the calling identity to 2000

:SYSetup

Used on NMT

:COmpandor

Description : Requests the NMT450i mobile's compandor to be in or out

Parameters : <CPD > OUT or IN

Allowed suffices : Not applicable

Example : `:SYSetup:COmpandor IN`
Sets the NMT450i mobile's compandor to be IN

:SYSetup

Used on MPT1327

:CPrefix

Description : Sets an MPT1327 mobile's Calling Prefix

Parameters : <NR1> Range 0 to 127

Allowed suffices : Not applicable

Example : `:SYSetup:CPrefix 100`
Sets the calling prefix to 100

:SYSetup

Used on AMPS and TACS

:DCc

Description : Sets the systems Digital Colour Code

Parameters : <NR1> Range 0 to 3

Allowed suffices : Not applicable

Example : `:SYSetup:DCc 2`
Sets the digital colour code to 2

:SYSetup

Used on all systems

:DIsplay

Description : Sets the Autorun control display format to summary or full

Parameters : <CPD > SUMMARY or FULL

Allowed suffices : Not applicable

Example : `:SYSetup:DIsplay SUMMARY`
Autorun tests will now be displayed in summary format

:SYSetup

Used on NAMPS and NTACS

:DScc

Description : Sets the systems Digital SAT Colour Code

Parameters : <NR1> Range 0 to 6

Allowed suffices : Not applicable

Example : `:SYSetup:DScc 2`

Sets the digital SAT colour code to 2

:SYSetup

Used on MPT1327

:FOa

Description : Enables or disables full off-air signaling for MPT1327

Parameters : <CPD > DISABLE or ENABLE

Allowed suffices : Not applicable

Example : `:SYSetup:FOa ENABLE`

Enables full off-air signaling for MPT1327 system

:SYSetup

Used on MPT1327, NMT, and PMR

:FTc

Description : Sets the First Traffic Channel used by Auto Test

Parameters : <NR1> Range 0 to 9999 depending on system

Allowed suffices : Not applicable

Example : `:SYSetup:FTc 100`

Sets the first traffic channel to 100

:SYSetup

Used on AMPS, NAMPS, PMR, TACS, and NTACS

:FVc

Description : Sets the First Voice Channel used by Auto Test

Parameters : <NR1> Range 0 to 9999 depending on system

Allowed suffices : Not applicable

Example : `:SYSetup:FVc 100`
Sets the first voice channel to 100

:SYSetup

Used on EDACS Radio and LTR Radio

:FWc

Description : Sets the first working channel used by Auto Test

Parameters : <NR1> Range 1 to 24 depending on selected variant

Allowed suffices : Not applicable

Example : `:SYSetup:FWc 3`
Sets the first working channel to 3

:SYSetup

Used on MPT1327

:GIdent

Description : Sets an MPT1327 mobile's Group Calling Identity

Parameters : <NR1> Range 0 to 8191

Allowed suffices : Not applicable

Example : `:SYSetup:GIdent 100`
Sets the group calling identity to 100

:SYSetup

Used on MPT1327

:GPrefix

Description : Sets an MPT1327 mobile's Group Calling Prefix

Parameters : <NR1> Range 0 to 127

Allowed suffices : Not applicable

Example : `:SYSetup:GPrefix 100`
Sets the group calling prefix to 100

:SYSetup

Used on EDACS Repeater, EDACS Radio, LTR Repeater and LTR Radio

:GRoupid

Description : Sets the group identity (GID) of the Service Monitor and mobile used

Parameters : <NR1> Range 0 to 2047

Allowed suffices : Not applicable

Example : `:SYSetup: GRoupid 100`
Sets the group ID to 100

:SYSetup

Used on AMPS, TACS, NAMPS, NTACS, NMT, PMR, and MPT1327

:HANDOFFInc

Description : Sets the handoff increment for the test

Parameters : <NR1> Range 0 to 9999 depending on selected system

Allowed suffices : Not applicable

Example : `:SYSetup:HANDOFFInc 100`
Sets the handoff increment to 100

:SYSetup

Used on NMT

:HANDOFFScheme

Description : Sets the handoff scheme to be used

Parameters : <CPD> C = “ordinary”, (NMT450, NMT450i, NMT900)
C1 = “improved” (NMT900)
C2 = “short” (NMT450i, NMT900)

Allowed suffices : Not applicable

Example : `:SYSetup:HANDOFFScheme C1`
Sets the handoff scheme to ‘improved’

:SYSetup

Used on NMT

:HCcb

Description : Sets the higher control channel band

Parameters : <NR1> Range 0 to 9999

Allowed suffices : Not applicable

Example : `:SYSetup:HCcb 1000`
Sets the higher scan band to 1000

:SYSetup

Used on LTR

:HR

Description : Sets the home repeater for the selected system

Parameters : <NR1> Range 1 to 20

Allowed suffices : Not applicable

Example : `:SYSetup:HR 14`
Sets the home repeater to 14

:SYSetup

Used on NMT

:HTcb

Description : Sets the higher traffic channel band

Parameters : <NR1> Range 0 to 9999

Allowed suffices : Not applicable

Example : `:SYSetup:HTcb 1000`
Sets the higher traffic channel band to 1000

:SYSetup

Used on AMPS, NAMPS, TACS, and NTACS

:Idstep

Description : Sets the identity step for these systems

Parameters : <NR1> Range 0 to 65535

Allowed suffices : Not applicable

Example : `:SYSetup:Idstep 500`
Sets the identity step to 500

:SYSetup

Used on NMT

:LCcb

Description : Sets the lower control channel band

Parameters : <NR1> Range 0 to 9999

Allowed suffices : Not applicable

Example : `:SYSetup:LCcb 400`
Sets the lower scan band to 400

:SYSetup

Used on NMT, PMR, and MPT1327

:LTC

Description : Sets the last traffic channel

Parameters : <NR1> Range 0 to 9999 depending on selected system

Allowed suffices : Not applicable

Example : `:SYSetup:LTC 300`
Sets the last traffic channel to 300

:SYSetup

Used on NMT

:LTCB

Description : Sets the lower traffic channel band

Parameters : <NR1> Range 0 to 9999

Allowed suffices : Not applicable

Example : `:SYSetup:LTCB 400`
Sets the lower traffic scan band to 400

:SYSetup

Used on AMPS, NAMPS, PMR, TACS, and NTACS

:LVc

Description : Sets the last voice channel

Parameters : <NR1> Range 0 to 9999 depending on selected system

Allowed suffices : Not applicable

Example : `:SYSetup:LVc 300`
Sets the last voice channel to 300

:SYSetup

Used on EDACS Radio and LTR Radio

:LWc

Description : Sets the last working channel used by Auto Test

Parameters : <NR1> Range 1 to 24 depending on selected variant

Allowed suffices : Not applicable

Example : `:SYSetup:LWc 7`
Sets the last working channel to 7

:SYSetup

Used on MPT1327

:MIDent

Description : Sets an MPT1327 mobile's identity

Parameters : <NR1> Range 0 to 8191

Allowed suffices : Not applicable

Example : `:SYSetup:MIDent 100`
Sets the mobile's identity to 100

:SYSetup

Used on AMPS, TACS, and NMT

:MIN

Description : Sets the Mobile Identity Number

Parameters : <CPD> **Note:** format is system-specific, as shown below:-
1. AMPS, JTACS or NTACS = "*ddd-ddd-dddd*"
2. TACS = "*ddd-d-dddddd*"
3. NMT = "*hhhhhhh*"
where *d* = decimal character
and *h* = hexadecimal character

Allowed suffices : Not applicable

Example : `:SYSetup:MIN "123-456-7890"`
Sets the MIN to 123-456-7890

:SYSetup

Used on NMT

:MObiletype

Description : Sets the NMT mobile type

Parameters : <CPD> HMSPORT or ORDINARY

Allowed suffices : Not applicable

Example : `:SYSetup:MObiletype ORDINARY`
Sets the NMT mobile type to be ordinary

:SYSetup

Used on MPT1327

:MPrefix

Description : Sets an MPT1327 mobile's prefix

Parameters : <NR1> Range 0 to 127

Allowed suffices : Not applicable

Example : `:SYSetup:MPrefix 100`
Sets the mobile's prefix to 100

:SYSetup

Used on AMPS, NAMPS, MPT1327, TACS, and NTACS

:Nformat

Description : Sets the base by which the mobile's ESN is displayed

Parameters : <CPD> DECIMAL
HEX
OCTAL (AMPS and TACS only)
STANDARD1
STANDARD2 (AMPS and TACS only)

Allowed suffices : Not applicable

Example : `:SYSetup:Nformat HEX`
The received mobile's ESN will now be displayed in HEX

:SYSetup

Used on all systems

:PAusemode

Description : Sets the pause condition for Autorun testing

Parameters : <CPD> MANUAL_ONLY
ON_FAILURE
ALWAYS

Allowed suffices : Not applicable

Example : `:SYSetup:PAusemode ON_FAILURE`
Autorun will now pause when it detects a test failure

:SYSetup

Used on MPT, EDACS and LTR Radio

:POwertype

Description : Sets the radio power type

Parameters : <CPD> MOBILE
PORTABLE

Allowed suffices : Not applicable

Example : `:SYSetup:POwertype MOBILE`
Radio power type is mobile

:SYSetup

Used on all systems

:PRint

Description : Sets the print condition for Autorun testing

Parameters : <CPD> OFF, ON

Allowed suffices : Not applicable

Example : `:SYSetup:PRint ON`
Autorun will now print each test results on an external printer

:SYSetup

Used on EDACS Repeater and EDACS Radio

:RAdiolid

Description : Sets the Logical ID (LID) for the radio (mobile)

Parameters : <NR1> Range 0 to 16383

Allowed suffices : Not applicable

Example : `:SYSetup: RAdiolid 100`
Sets the Logical ID of the radio to 100

:SYSetup

Used on AMPS, NAMPS, TACS, and NTACS

:REGID

Description : Sets the Registration ID on the test set

Parameters : <NR1> Range 0 to 1048575 depending on selected system

Allowed suffices : Not applicable

Example : `:SYSetup:REGID 4000`
Sets the Registration ID to 4000

:SYSetup

Used on AMPS, NAMPS, TACS, and NTACS

:REGINc

Description : Sets the Registration ID increment

Parameters : <NR1> Range 0 to 4095 depending on selected system

Allowed suffices : Not applicable

Example : `:SYSetup:REGINc 4000`
Sets the registration ID increment to 4000

:SYSetup

Used on LTR Radio

:REPidle

Description : Sets the idle frame repeat rate (in seconds)

Parameters : <NR1> Range 1 to 20

Allowed suffices : Not applicable

Example : `:SYSetup:REPidle 10`
Sets the idle frame repeat rate to 10 s

:SYSetup

Used on NMT

:SAtphi

Description : Sets the SAT number Φ for the NMT system

Parameters : <NR1> Range 0 to 4

Allowed suffices : Not applicable

Example : `:SYSetup:SAtphi 2`
Sets the NMT system SAT to 2

:SYSetup

Used on NMT

:SCAnband

Description : Sets the NMT900 scan band channel type

Parameters : <CPD> BASIC
CCTC
CC
TC

Allowed suffices : Not applicable

Example : `:SYSetup:SCAnband BASIC`
Sets the NMT scan band type to a BASIC channel

:SYSetup

Used on AMPS, and TACS

:SCC

Description : Sets the systems SAT Colour Code

Parameters : <NR1> Range 0 to 2

Allowed suffices : Not applicable

Example : `:SYSetup:SCC 2`
Sets the SAT colour code to 2

:SYSetup

Used on AMPS

:SID

Description : Sets the AMPS system identity

Parameters : <NR1> Range 0 to 32767

Allowed suffices : Not applicable

Example : `:SYSetup:SID 3952`
Sets the system identity to 3952

:SYSetup

Used on NMT

:SIGnallingscheme

Description : Sets the user-defined signaling scheme

Parameters : <CPD> NMT450
NMT450I

Allowed suffices : Not applicable

Example : `:SYSetup:SIGnallingscheme NMT450`
Sets the signaling scheme to NMT450

:SYSetup

Used on NMT

:SISc

Description : Enables or disables NMT System Challenge mode

Parameters : <CPD> DISABLE
ENABLE

Allowed suffices : Not applicable

Example : `:SYSetup:SISc ENABLE`
Enables NMT system challenge mode

:SYSetup

Used on NMT

:TA

Description : Sets the Traffic Area for NMT systems

Parameters : <NR1> Range 0 to FF, Hex

Allowed suffices : Not applicable

Example : `:SYSetup:TA #Ha2`
Sets the traffic area to a2

:SYSetup

Used on NMT

:TC

Description : Sets the Tariff Class for NMT systems

Parameters : <NR1> Range 0 to 255

Allowed suffices : Not applicable

Example : `:SYSetup:TC 100`
Sets the tariff class to 100

:SYSetup

Used on MPT1327

:TCOn

Description : Specifies the Traffic Confirmation Signaling for MPT1327

Parameters : <CPD> AHOY
DISABLE
PRESSEL

Allowed suffices : Not applicable

Example : `:SYSetup:TCOn AHOY`
Sets all traffic channel confirmations (Handoff, Placecall etc.) to require an "AHOY" message from the MPT1327 mobile unit

:SYSetup

Used on EDACS Repeater, EDACS Radio

:TEstsetlid

Description : Sets the Logical Identity (LID) for the Service Monitor

Parameters : <NR1> Range 1 to 16383

Allowed suffices : Not applicable

Example : `:SYSetup: TEstsetlid 100`
Sets the service monitor logical ID to 100

:SYSetup

Used on all systems

:TYpe

Description : Sets System Type

Parameters : <CRD> AMPS, TACS, NMT, PMR, MPT1327, EDACSRADIO, EDACSREPEATER, LTRRADIO, LTRREPEATER or NONE

Allowed suffices : Not applicable

Example : `:SYSetup:TYpe AMPS`
Sets current system to AMPS

:SYSetup

Used on EDACS Repeater, EDACS Radio, LTR Repeater and LTR Radio

:TXpolarity

Description : Sets the polarity of the transmitter

Parameters : <CPD> NORMAL, INVERT or AUTO

Allowed suffices : Not applicable

Example : `:SYSetup: TXpolarity NORMAL`
Set the polarity of the transmitter to be NORMAL

:SYSetup

Used on AMPS, NAMPS, TACS, NTACS, NMT, MPT1327,
EDACS Repeater, EDACS Radio, LTR Repeater and LTR Radio

:VAriant

Description : Sets System Variant

Parameters : <CPD>

EAMPS, NAMPS, USERAMPS

ETACS, TACS2, CTACS1, CTACS2, JTACS, NTACS USERTACS

NMT450, NMT900, BENELUX, NMTF, AUSTRIA, SPAIN, MALAYSIA, SAUDI_B1, SAUDI_B2, INDONESIA, THAILAND, OMAN, TUNISIA, TURKEY, HUNGARY, RUSSIA, CZECH, BULGARIA, SLOVENIA, USERNMT

BAND3, JRC, HONGKONG, AUTONET, MADEIRA, AMT, NL_TRAXYS, NZ_MPT1327, UK_WATER, PH_INDO, REPART, CARRIS, RADIOMOVEL, USERMPT

REPEATER1, REPEATER2, REPEATER3, REPEATER4

RADIO1, RADIO2, RADIO3, RADIO4

LREPEATER1, LREPEATER2, LREPEATER3, LREPEATER4

LRADIO1, LRADIO2, LRADIO3, LRADIO4

Allowed suffices : Not applicable

Example : `:SYSetup:VAriant NAMPS`
Sets system variant to NAMPS

Setup commands for user-defined systems

List of commands

:SYUsersetup

Used on MPT1327

:Blockn

:CHANNELOffset

Description : For a given user-defined channel block *n*, specifies the offset between logical and physical channel numbers

Parameters : <NR1> *n* = 1 to 8
<NR1> Range 0 to 9999 (Decimal)

Allowed suffices : Not applicable

Example : `:SYUsersetup:Blockn:CHANNELOffset 1,100`
Sets the logical to physical channel offset to be 100 for channel block 1

:SYUsersetup

Used on AMPS, NAMPS, TACS, NTACS, MPT1327, and NMT

:Blockn

:CHANNELSpacing

Description : For a given user-defined channel block *n*, specifies the spacing between RF channels

Parameters : <NR1> *n* = 1 to 8
<NRf> Range ±5 kHz to ±99.999 kHz

Allowed suffices : `HZ, KHZ, MHZ`

Example : `:SYUsersetup:Blockn:CHANNELSpacing 1,30kHz`
Sets the channel spacing for user-defined channel block 1 to 30 kHz

:SYUsersetup

Used on AMPS, NAMPS, TACS, NTACS, MPT1327, and NMT

:Blockn

:Duplexoffset

Description : For a given user-defined channel block *n*, specifies the spacing between forward and reverse channel links for a duplex system

Parameters : <NR1> *n* = 1 to 8
<NRf> Range –75.0 MHz to +75.0 MHz

Allowed suffices : `HZ, KHZ, MHZ`

Example : `:SYUsersetup:Blockn:Duplexoffset 1,45MHz`
Sets the duplex offset to 45 MHz for channel block 1

:SYUsersetup

Used on AMPS, NAMPS, TACS, NTACS, MPT1327, and NMT

:Blockn

:Highestchannel

Description : For a given user-defined channel block *n*, specifies the highest channel for the current channel block number

Parameters : <NR1> *n* = 1 to 8
<NR1> Range 0 to 9999

Allowed suffices : Not applicable

Example : `:SYUsersetup:Blockn:Highestchannel 1,1000`
Sets the highest channel number to be 1000 for channel block 1

:SYUsersetup

Used on AMPS, NAMPS, TACS, NTACS, MPT1327, and NMT

:Blockn

:Lowestchannel

Description : For a given user-defined channel block *n*, specifies the lowest channel

Parameters : <NR1> *n* = 1 to 8
<NR1> Range 0 to 9999

Allowed suffices : Not applicable

Example : `:SYUsersetup:Blockn:Lowestchannel 1,10`
Sets the lowest channel number to 10 for channel block 1

:SYUsersetup

Used on AMPS, NAMPS, TACS, NTACS, and NMT

:Blockn

:State

Description : For a given user-defined channel block *n*, enables or disables the block

Parameters : <NR1> *n* = 1 to 8
<CPD> INCLUDED or EXCLUDED

Allowed suffices : Not applicable

Example : `:SYUsersetup:Blockn:State 1,INCLUDED`
Enables channel block 1

:SYUsersetup

Used on AMPS, NAMPS, TACS, NTACS, MPT1327, and NMT

:Blockn

:Txbasefreq

Description : For a given user-defined channel block *n*, specifies the transmitter frequency for the first channel in the block

Parameters : <NR1> *n* = 1 to 8
<NRf> Range 10.0 MHz to 999.9999 MHz

Allowed suffices : `HZ, KHZ, MHZ`

Example : `:SYUsersetup:Blockn:Txbasefreq 1,45MHz`
Sets the transmitter base frequency to 45 MHz for channel block 1

:SYUsersetup

Used on EDACS Repeater, EDACS Radio

:CHannelno

:Duplexoffset

Description : For a given channel n, specifies the offset between transmit and receive frequencies.

Parameters : <NR1> n= 1 to 24

<NRf> Range -75.0 MHz to +75.0 MHz

Allowed suffices : HZ, KHZ, MHZ

Example : `:SYUsersetup: CHannelno:Duplexoffset 1, 45MHz`
Sets the duplex offset for channel 1 to 45 MHz

:SYUsersetup

Used on LTR Repeater, LTR Radio

:CHannelno

:Fccchan

Description : For a given channel n, specifies the FCC channel number.

Parameters : <NR1> n = 1 to 20

Allowed suffices : Not applicable

Example : `:SYUsersetup: CHannelno:Fccchan 1`
Set the FCC channel number to 1

:SYUsersetup

Used on EDACS Repeater, EDACS Radio, LTR Repeater and LTR Radio

:CHannelno

:State

Description : For a given channel n, enables or disables the channel.

Parameters : <NR1> n = 1 to 24 (depending on system)
<CPD> INCLUDED or EXCLUDED

Allowed suffices : Not applicable

Example : `:SYUsersetup: CHannelno:State 1, INCLUDED`
Enables channel 1

:SYUsersetup

Used on EDACS and LTR Repeater, EDACS and LTR Radio

:CHannelno

:Txfreq

Description : For a given channel n, specifies the transmitter frequency.

Parameters : <NR1> n = 1 to 24 (depending on system)
<NRf> Range 10.0 MHz to 999.9999 MHz

Allowed suffices : Hz, kHz, MHz

Example : `:SYUsersetup: CHannelno:Txfreq 1, 845MHz`
Set the transmitter frequency for channel 1 to 845 MHz

:SYUsersetup

Used on NMT

:CONtrolpower

Description : Sets the reverse Control Channel Power for an NMT mobile

Parameters : <NR1> Range 0 to 3

Allowed suffices : Not applicable

Example : `:SYUsersetup:CONtrolpower 3`
Sets the reverse control channel power to 3

:SYUsersetup

Used on NMT

:COUntrycode

Description : Sets the operational Country Code for the user-defined NMT system

Parameters : <NR1> Range 0 to 15

Allowed suffices : Not applicable

Example : `:SYUsersetup:COUntrycode 15`
Sets the country code to 15

:SYUsersetup

Used on AMPS, NAMPS, TACS, NTACS, MPT1327, LTR Radio, LTR Repeater and NMT

:DATADevn

Description : Sets the data deviation level for both control and voice (traffic) channels

Parameters : <NRf> Range 0 to 8 kHz

Allowed suffices : `HZ, KHZ, MHZ`

Example : `:SYUsersetup:DATADevn 5kHz`
Sets the data deviation level to 5 kHz

:SYUsersetup

Used on EDACS Repeater, EDACS Radio

:DATARate

Description : Sets the high speed data rate for the EDACS system

Parameters : <CPD> BAUD_9600 or BAUD_4800

Allowed suffices : Not applicable

Example : `:SYUsersetup:DATARate BAUD_9600`
Sets the data rate to 9600 baud

:SYUsersetup

Used on LTR Repeater, LTR Radio

:Fband

Description : Sets the frequency band for the LTR system

Parameters : <CPD> BAND_800, BAND_900 or BAND_USER

Allowed suffices : Not applicable

Example : `:SYUsersetup:Fband BAnD_900`
Sets the LTR frequency band to 900 MHz

:SYUsersetup

Used on EDACS Repeater, EDACS Radio

:Hsdatadevn

Description : Sets the high speed data deviation

Parameters : <NRf> Range 0 to 8 kHz

Allowed suffices : Hz, kHz, MHz

Example : `:SYUsersetup:Hsdatadevn 3.0kHz`
Sets the high speed data deviation to 3 kHz

:SYUsersetup

Used on EDACS Repeater, EDACS Radio

:Lsdatadevn

Description : Sets the low speed data deviation

Parameters : <NRf> Range 0 to 8 kHz

Allowed suffices : Hz, kHz, MHz

Example : `:SYUsersetup: Lsdatadevn 500Hz`
Sets the low speed data deviation to 500 Hz

:SYUsersetup

Used on EDACS Repeater and LTR Repeater

:Rfport

Description : Sets the RF port selection for the current EDACS/LTR system

Parameters : <CPD> N_OUT_N_IN,
BNC_OUT_N_IN,
BNC_OUT_ANT_IN,
N_OUT_ANT_IN

Allowed suffices : Not applicable

Example : `:SYUsersetup: Rfport BNC_OUT_ANT_IN`
sets RF port selection for BNC out ANTENNA in

:SYUsersetup

Used on AMPS, NAMPS, TACS, NTACS, and NMT

:SAtdevn

Description : Sets the SAT deviation level for voice traffic channels

Parameters : <NRf> Range 0 to 2.5 kHz

Allowed suffices : `HZ, KHZ, MHZ`

Example : `:SYUsersetup:SAtdevn 1.7kHz`
Sets the SAT deviation level to 1.7 kHz

:SYUsersetup

Used on NMT

:SIgnallingscheme

Description : Sets the user-defined signaling scheme

Parameters : <CPD> NMT450, NMT450I, and NMT900

Allowed suffices : Not applicable

Example : `:SYUsersetup:SIgnallingscheme NMT450`
Sets the signaling scheme to NMT450

:SYUsersetup

Used on MPT1327, LTR Radio and LTR Repeater

:SYNC

Description : Sets the user-defined MPT1327 Control Channel Sync pattern or LTR Sync

Parameters : <NR1 > Range 0 to FFFF, Hex

Allowed suffices : Not applicable

Example : `:SYUsersetup:SYNC #HCD17`
Sets the control channel sync pattern to CD17

:SYUsersetup

Used on MPT1327

:SYNT

Description : Sets the user-defined MPT1327 Traffic Channel Sync pattern

Parameters : <NR1> Range 0 to FFFF, Hex

Allowed suffices : Not applicable

Example : `:SYUsersetup:SYNT #HCD17`
Sets the traffic channel sync pattern to CD17

:SYUsersetup

Used on NMT

:TA

Description : Sets the user defined NMT Y1 traffic area.

Parameters : <NR1> Range 0 to F, HEX

Allowed suffices : Not applicable

Example : `:SYUsersetup:TA #H3`

Sets Y1 to 3

:SYUsersetup

Used on all systems except PMR

:TItle

Description : Sets the User Title for the user-defined system

Parameters : <CPD> Range 1 to 10 characters

Allowed suffices : Not applicable

Example : `:SYUsersetup:TItle "FRED"`

Sets the user title to "FRED"

:SYUsersetup

Used on NMT

:TRafficpower

Description : Sets the reverse Traffic Channel Power for an NMT mobile

Parameters : <NR1> Range 0 to 3

Allowed suffices : Not applicable

Example : `:SYUsersetup:TRafficpower 3`

Sets the reverse traffic channel power to 3

AEROFLEX INTERNATIONAL LTD.
SOFTWARE LICENCE AND WARRANTY

This document is an Agreement between the user of this Licensed Software, the Licensee, and Aeroflex International Limited, the Licensor. By opening this Software package or commencing to use the software you accept the terms of this Agreement. If you do not agree to the terms of this Agreement please return the Software package unopened to Aeroflex International Limited or do not use the software.

1. DEFINITIONS

The following expressions will have the meanings set out below for the purposes of this Agreement:

Add-In Application Software	Licensed Software that may be loaded separately from time to time into the Equipment to improve or modify its functionality
Computer Application Software	Licensed Software supplied to run on a standard PC or workstation
Designated Equipment	the single piece of Equipment upon which the licensed software is installed
Downloaded Software	any software downloaded from an Aeroflex web site
Embedded Software	Licensed Software that forms part of the Equipment supplied by Aeroflex and without which the Equipment cannot function
Licence Fee	the consideration ruling at the date of this Agreement for the use of one copy of the Licensed Software on the Designated Equipment
Licensed Software	All and any programs, listings, flow charts and instructions in whole or in part including Add-in, Computer Application, Downloaded and Embedded Software supplied to work with Designated Equipment

2. LICENCE FEE

The Licensee shall pay the Licence Fee to Aeroflex in accordance with the terms of the contract between the Licensee and Aeroflex.

3. TERM

This Agreement shall be effective from the date hereof and shall continue in force until terminated under the provisions of Clause 9.

4. LICENCE

4.1 Unless and until terminated, this Licence confers upon the Licensee the non-transferable and non-exclusive right to use the Licensed Software on the Designated Equipment.

4.2 The Licensee may not use the Licensed Software on other than the Designated Equipment, unless written permission is first obtained from Aeroflex and until the appropriate additional Licence Fee has been paid to Aeroflex.

4.3 The Licensee may not amend or alter the Licensed Software and shall have no right or licence other than that stipulated herein.

4.4 The Licensee may make not more than two copies of the Licensed Software (but not the Authoring and Language Manuals) in machine-readable form for operational security and shall ensure that all such copies include Aeroflex's copyright notice, together with any features which disclose the name of the Licensed Software and the Licensee. Furthermore, the Licensee shall not permit the Licensed Software or any part to be disclosed in any form to any third party and shall maintain the Licensed Software in secure premises to prevent any unauthorised disclosure. The Licensee shall notify Aeroflex immediately if the Licensee has knowledge that any unlicensed party possesses the Licensed Software. The Licensee's obligation to maintain confidentiality shall cease when the Licensed Software and all copies have been destroyed or returned. The copyright in the Licensed Software shall remain with Aeroflex. The Licensee will permit Aeroflex at all reasonable times to audit the use of the Licensed Software.

4.5 The Licensee will not disassemble or reverse engineer the Licensed Software, nor sub-licence, lease, rent or part with possession or otherwise transfer the whole or any part of the Licensed Software.

5 WARRANTY

5.1 Aeroflex certifies that the Licensed Software supplied by Aeroflex will at the time of delivery function substantially in accordance with the applicable Software Product Descriptions, Data Sheets or Product Specifications published by Aeroflex.

5.2 The warranty period (unless an extended warranty for Embedded Software has been purchased) from date of delivery in respect of each type of Licensed Software is:

Embedded Software	12 months
Add-In Application Software	90 days
Computer Application Software	90 days
Downloaded Software	No warranty

5.3 If during the appropriate Warranty Period the Licensed Software does not conform substantially to the Software Product Descriptions, Data Sheets or Product Specifications Aeroflex will provide:

5.3.1 In the case of Embedded Software and at Aeroflex's discretion either a fix for the problem or an effective and efficient work-around.

5.3.2 In the case of Add-In Application Software and Computer Application Software and at Aeroflex's discretion replacement of the software or a fix for the problem or an effective and efficient work-around.

5.4 Aeroflex does not warrant that the operation of any software will be uninterrupted or error free.

6 The above Warranty does not apply to:

6.1 Defects resulting from software not supplied by Aeroflex, from unauthorised modification or misuse or from operation outside of the specification.

6.2 Third party produced Proprietary Software which Aeroflex may deliver with its products, in such case the third party Software Licence Agreement including its warranty terms shall apply.

7 The remedies offered above are sole and exclusive remedies and to the extent permitted by applicable law are in lieu of any implied conditions, guarantees or warranties whatsoever and whether statutory or otherwise as to the software all of which are hereby expressly excluded.

8. INDEMNITY

8.1 Aeroflex shall defend, at its expense, any action brought against the Licensee alleging that the Licensed Software infringes any patent, registered design, trademark or copyright, and shall pay all Licensor's costs and damages finally awarded up to an aggregate equivalent to the Licence fee provided the Licensee shall not have done or permitted to be done anything which may have been or become any such infringement and shall have exercised reasonable care in protecting the same failing which the Licensee shall indemnify Aeroflex against all claims costs and damages incurred and that Aeroflex is given prompt written notice of such claim and given information, reasonable assistance and sole authority to defend or settle such claim on behalf of the Licensee. In the defence or settlement of any such claim, Aeroflex may obtain for the Licensee the right to continue using the Licensed Software or replace it or modify it so that it becomes non-infringing.

8.2 Aeroflex shall not be liable if the alleged infringement:

8.2.1 is based upon the use of the Licensed Software in combination with other software not furnished by Aeroflex, or

8.2.2 is based upon the use of the Licensed Software alone or in combination with other software in equipment not functionally identical to the Designated Equipment, or

8.2.3 arises as a result of Aeroflex having followed a properly authorised design or instruction of the Licensee, or

8.2.4 arises out of the use of the Licensed Software in a country other than the one disclosed to Aeroflex as the intended country of use of the Licensed Software at the commencement of this Agreement.

8.3 Aeroflex shall not be liable to the Licensee for any loss of use or for loss of profits or of contracts arising directly or indirectly out of any such infringement of patent, registered design, trademark or copyright.

9. TERMINATION

9.1 Notwithstanding anything herein to the contrary, this Licence shall forthwith determine if the Licensee:

9.1.1 As an individual has a Receiving Order made against him or is adjudicated bankrupt or compounds with creditors or as a corporate body, compounds with creditors or has a winding-up order made against it or

9.1.2 Parts with possession of the Designated Equipment.

9.2 This Licence may be terminated by notice in writing to the Licensee if the Licensee shall be in breach of any of its obligations hereunder and continue in such breach for a period of 21 days after notice thereof has been served on the Licensee.

9.3 On termination of this Agreement for any reason, Aeroflex may require the Licensee to return to Aeroflex all copies of the Licensed Software in the custody of the Licensee and the Licensee shall, at its own cost and expense, comply with such requirement within 14 days and shall, at the same time, certify to Aeroflex in writing that all copies of the Licensed Software in whatever form have been obliterated from the Designated Equipment.

10. THIRD PARTY LICENCES

The software or part thereof may be the proprietary property of third party licensors. In such an event such third party licensors (as referenced on the package or the Order Acknowledgement) and/or Aeroflex may directly enforce the terms of this Agreement and may terminate the Agreement if the Licensee is in breach of the conditions contained herein.

11. EXPORT REGULATIONS

The Licensee undertakes that where necessary the Licensee will conform with all relevant export regulations imposed by the Governments of the United Kingdom and/or the United State of America.

12. NOTICES

Any notice to be given by the Licensee to Aeroflex shall be addressed to:

Aeroflex International Limited, Longacres House, Six Hills Way, Stevenage, SG1 2AN, UK.

13. LAW AND JURISDICTION

This Agreement shall be governed by the laws of England and shall be subject to the exclusive jurisdiction of the English courts. This agreement constitutes the whole Contract between the parties and may be changed only by memorandum signed by both parties.

CHINA Beijing
Tel: [+86] (10) 6539 1166
Fax: [+86] (10) 6539 1778

CHINA Shanghai
Tel: [+86] (21) 5109 5128
Fax: [+86] (21) 5150 6112

FINLAND
Tel: [+358] (9) 2709 5541
Fax: [+358] (9) 804 2441

FRANCE
Tel: [+33] 1 60 79 96 00
Fax: [+33] 1 60 77 69 22

GERMANY
Tel: [+49] 8131 2926-0
Fax: [+49] 8131 2926-130

HONG KONG
Tel: [+852] 2832 7988
Fax: [+852] 2834 5364

INDIA
Tel: [+91] 80 5115 4501
Fax: [+91] 80 5115 4502

KOREA
Tel: [+82] (2) 3424 2719
Fax: [+82] (2) 3424 8620

SCANDINAVIA
Tel: [+45] 9614 0045
Fax: [+45] 9614 0047

SPAIN
Tel: [+34] (91) 640 11 34
Fax: [+34] (91) 640 06 40

UK Burnham
Tel: [+44] (0) 1628 604455
Fax: [+44] (0) 1628 662017

UK Stevenage
Tel: [+44] (0) 1438 742200
Fax: [+44] (0) 1438 727601
Freephone: 0800 282388

USA
Tel: [+1] (316) 522 4981
Fax: [+1] (316) 522 1360
Toll Free: (800) 835 2352

web **www.aeroflex.com** Email **info-test@aeroflex.com**

November 2005