HP 8921A HP-IB Programmer's Guide

SERIAL NUMBERS

This manual applies directly to instruments with firmware revisions:
A.12.01 and above and documents all MAJOR changes that apply to your instrument.

HP Part No. 08921-90023
Printed in U.S.A. June 1994 (Rev. A)

First Edition

Hewlett-Packard Company
Learning Products Department
24001 E. Mission
Liberty Lake, WA 99019-9599
U.S.A.

In This Book

Chapter 1—*Using HP-IB* describes the general guidelines for using HP-IB and how to prepare the Test Set for HP-IB usage. This chapter includes example programs for controlling the basic functions of the Test Set.

Chapter 2—*Command Guidelines* contains information about syntax names, punctuation, units, and state commands. There is also a programming example.

Chapter 3—*HP-IB Commands* contains syntax diagrams, and syntax for front-panel key operations and other commonly used operations such as Save, and Recall. This chapter also contains syntax for triggering measurements.

Chapter 4—*Advanced Operations* includes information about speeding-up measurements, status reporting, error reporting, and pass control.

Chapter 5—*Memory Cards/Mass Storage* describes the types of mass storage (memory cards, ROM, RAM) and formats (DOS, LIF) used by the Test Set.

Chapter 6—*IBASIC Overview* describes the how the TESTS subsystem used IBASIC. It describes entering and editing programs, program control, writing programs for the TESTS subsystem, and using the Test Set's software development tool.

Contents

3. HP-IB Commands

6. Test Set Instrument BASIC Overview

Figures

Tables

1

Using HP-IB

Getting Started

What is HP-IB?

The Hewlett-Packard Interface Bus (HP-IB) conforms to IEEE 488.2 standards providing bi-directional data transfer between devices. This allows several valuable functions:

- Programs running in the Test Set's IBASIC Controller can control all the Test Set's functions using its internal HP-IB bus. This controller provides a single-instrument automated test system. (The HP 11807 Radio Test Software is used this way.)
- Programs running in the Test Set's IBASIC Controller can control other instruments connected to the bus. (Requires Option 003 — RS-232/HP-IB/Current Measurement)
- A connected controller can remotely control the Test Set. (Requires Option 003 — RS-232/HP-IB/Current Measurement)
- A connected HP-IB printer can be used to print test results and full screen images. (Requires Option 003 — RS-232/HP-IB/Current Measurement)

HP-IB Information Provided in This Manual

What We Do Explain

- How to configure your Test Set for HP-IB operation.
- Command syntax structure.
- How to make an instrument setting over HP-IB.
- How to read-back instrument settings over HP-IB.
- How to make measurements over HP-IB.
- All of the HP-IB command syntax for the Test Set.
- IBASIC program transfer over HP-IB.
- Various advanced functions: increasing measurement speed, status reporting, error reporting, pass control, and so forth.

What We Don't Explain

- HP-IB (IEEE 488.2) theory of operation.[1]
- HP-IB electrical specifications.[1]
- HP-IB connector pin functions.[1]
- IBASIC programming[2] (other than general guidelines related to HP-IB).

[1]Refer to the *Tutorial Description of the Hewlett-Packard Interface Bus* (HP P/N 5952-0156) for detailed information on HP-IB theory and operation.

[2] Refer to the HP Instrument BASIC Programmer's Guide for more information on the IBASIC language.

General HP-IB Guidelines

The following guidelines should be considered before developing HP-IB test procedures:

- Unless your test only needs to use functions available on the **TX TEST** and **RX TEST** screens, avoid using these screens in your automated tests. Going to either of those screens automatically re-configures 6 fields to their last settings on that screen. (See *Interaction Between Screens* in chapter 3 of the User's Guide.) The **AF ANALYZER**, **RF ANALYZER**, and **RF GENERATOR** screens contain most of the functions available on the **RX TEST** and **TX TEST** screens.
- Perform the test manually, recording the test results, then write your test to imitate manual operation. For example, if a function is located on the **RF GENERATOR** screen, use the "DISP RFG" (Display RF Generator) command to go to the **RF Generator** screen before selecting the needed function. If you need to use functions on another screen, use the appropriate DISPlay command to get to that screen before selecting the next function.
- Compare the manual test results to the automated test results. If the test results do not match, be sure the correct TRIGger commands are being sent. (See *Speeding-Up Measurements* in chapter 4.)

Control Annunciators

The letters and symbols at the top right corner of the display indicate these conditions:

- **R** indicates remote operation from an external controller or IBASIC program running in the Test Set.
- **L** indicates the Test Set is listening, and is ready to receive a manual or remote command.
- **T** indicates the Test Set is talking to another HP-IB device.
- **S** indicates a service request has been generated. (See *Status Reporting*.)
- **C** indicates the Test Set is currently an active controller.
- ***** indicates an IBASIC program is running.
- **?** indicates an IBASIC program is waiting for a user response.

Preparing the Test Set For HP-IB Use

1. Attach an HP-IB cable from the rear-panel HP-IB connector to any instruments/controller in the test system.
2. Access the **I/O Configure** screen.
3. Set the `HP-IB Adrs`.
4. Set the HP-IB `Mode`:
   a. `Talk&Listen` should be used unless you are using the Test Set to control connected HP-IB devices.
   b. `Control` configures the Test Set to be a system controller. Use this setting for controlling connected HP-IB devices.
5. Access the **Print Configure** screen.
6. Select printer `Model`.
7. Select the `Printer Port`.
8. If an HP-IB printer is connected, set `Printer Address` to your HP-IB printer's address.
9. If a serial printer is used, refer to the RS-232 connector information in chapter 5 of the Test Set's User's Guide, and the serial communication settings on the **I/O CONFIGURE** screen discussed in the User's Guide, chapter 4.

Multiple Controllers

Only one system controller can be connected to the bus at a time. If two or more system controllers are on the bus, instruments connected to the bus will not operate dependably, if at all.

Passing bus control from one controller to another is discussed in *Passing Control* in chapter 4.

Using the Test Set's IBASIC Controller

Hewlett-Packard Instrument BASIC (IBASIC) programs running in the Test Set's IBASIC controller must use **8xx** as the interface select code for addressed commands to the Test Set. The select code for controlled external instruments does not change.

For example, if you are using the HP-IB address **7**14 for Test Set programs running on a connected external BASIC controller, you must change the address to **8**14 when running the same program from the Test Set's internal IBASIC controller. (Refer to the *HP Instrument BASIC Programmer's Manual.*)

Example:
Using an external controller, HP-IB commands might look like this

```
OUTPUT 714;"*RST"   !This command is sent to the Test Set.
OUTPUT 719;"*RST"   !This command is sent to another instrument.
```

Running internally on the Test Set's IBASIC controller, the same commands would look like this—

```
OUTPUT 814;"*RST"  !The Test Set's bus select code changes to 8.
OUTPUT 719;"*RST"  !The other instrument's address doesn't change.
```

Basic Function Examples

The following simple examples indicate the basic approach to HP-IB operations with the Test Set. The punctuation and command syntax used for these examples are given later in this chapter.

The bus address (714) used in the following BASIC language examples assumes a bus controller interface select code of **7**, and a Test Set HP-IB address of **14**. (All examples assume an external controller is being used.)

Changing a Field Setting

To change a setting over HP-IB:

1. Access the screen containing the setting using a DISPlay command.
2. Make the desired setting as indicated on the syntax diagram for that screen.

The following example makes several instrument setting changes:

```
OUTPUT 714;"DISP RFG"           !Display the RF Generator screen.
OUTPUT 714;"RFG:FREQ 850 MHZ"   !Set the RF Gen Freq to 850 MHz.
OUTPUT 714;"RFG:OUTP 'DUPL'"    !Set the Output Port to Duplex.
OUTPUT 714;"DISP AFAN"          !Display the AF Analyzer screen.
OUTPUT 714;"AFAN:INP 'FM DEMOD'" !Set the AF Anl In to FM Demod.
```

Reading a Field Setting

To read an instrument setting over HP-IB:

1. Access the screen containing the setting using a Display command.
2. Use the Query form of the syntax for that setting. The returned value is in *HP-IB Units*.
3. Enter the value into a variable that can be printed out, or used in calculations or other program routines.

For example, to print the AF Anl In field setting using the BASIC language, enter the commands:

```
OUTPUT 714;"DISP AFAN"   !Display the AF Analyzer screen.
OUTPUT 714;"AFAN:INP?"   !Ask the Test Set for the field's value.
ENTER 714;Af_input$      !Enter the returned value into a variable.
PRINT Af_input$          !Print the value as a quoted string.
```

Note **Returned Numeric Values**

Whenever a measurement or numeric setting is queried, the returned value is always in *HP-IB Units*. Refer to the *Specifying Units of Magnitude* information in chapter 2.

Making a Simple Measurement

The basic method for making a measurement is very similar to reading a field setting. You access the screen displaying the measurement, tell the Test Set the measurement you need, and then enter the returned value into a variable.

For example, if you want to print the `TX Power` field's value, enter the BASIC language commands:

```
OUTPUT 714;"DISP RFAN"            !Display the RF Analyzer screen.
OUTPUT 714;"MEAS:RFR:POW?"          !Ask for the measurement value.
ENTER 714;Tx_power       !Enter the returned value into a variable.
PRINT Tx_power                                    !Print the value.
```

The example above is for a very simple measurement. There are many considerations that must be made when you write your test programs (such as triggering and measurement speed). Refer to the *Speeding-Up Measurements* in chapter 4 and *Triggering Measurements* information given in chapter 3.

2

Command Guidelines

The following topics discuss rules and guidelines you need to know when operating the Test Set using HP-IB.

Syntax Names

All syntax of more than four characters have an alternate abbreviated form using only the upper case letters and number (if used). *(Refer to the syntax diagram for the* **RX TEST** *screen.)* You can use upper and lower case for all commands.

For example, to set the destination of AF Generator 1 to Audio Out, you could use any of the following commands:

```
AFGENERATOR1:DESTINATION 'AUDIO OUT'
            or
afgenerator1:destination 'audio out'
            or
   afg1:dest 'audio out'
            or
   AFG1:DEST 'AUDIO OUT'
```

Command Punctuation

Note

Language Considerations: The punctuation for Test Set HP-IB commands conforms to the IEEE 488.2 standards. It is possible that the programming language you are running on your controller will not accept some of the punctuation used in the BASIC examples. It is therefore necessary that you understand the punctuation and language equivalents required by your language for HP-IB operation.

Using Quotes for String Entries

Quotation marks (' and ") are used to select a non-numeric field setting, such as AM DEMOD for the `AF Anl In` field. The value is entered into the command line as a quoted alphanumeric string.

Quotes are used with all Underlined (toggling) and One-of-many (menu choice) fields. (See *How Do I Change A Field's Setting* in chapter 1 of the User's Guide for field type descriptions.)

For example, if you need to set the **DUPLEX TEST** screen's `Output Port` to Duplex, you would enter the menu choice, 'Dupl'.

```
RFG:OUTP 'Dupl'
     or
RFG:OUTP "Dupl"
```

Using Spaces

When changing a field setting, a space must always precede the setting value in the command (command <space> value), regardless of field type.
RFG:FREQ ^{space} 850MHZ
RFG:ATT ^{space} 'OFF'

Using Colons to Separate Commands

HP-IB command syntax are arranged by a control hierarchy that is analogous to manual operation.

When you want to make an instrument setting using front-panel controls, you access the screen first, select the desired field, and make the appropriate setting. You use HP-IB commands the same way, using the colon (:) to separate the syntax levels.

For example, if you have already accessed the **AF Analyzer** screen using the "DISP AFAN" command, and you want to tell the Test Set to set the `Input Gain` field to 40 dB, you would enter the command:

```
AFAN:INP:GAIN '40 dB'
```

Using Semicolons for Multiple Commands

You can output multiple commands from one program line by separating the commands with a semicolon (;). The semicolon tells the Test Set to back up one level of hierarchy and accept the next command at the same level as the previous command.

For example, on one command line, you can -

1. access the **AF Analyzer** screen,
2. set the AF Analyzer Input to AM Demod, and
3. set Filter 1 to 300 Hz.

```
DISP AFAN;AFAN:INP 'AM DEMOD';FILT1 '300Hz HPF'
```

The semicolon after the "DISP AFAN" command told the Test Set to return to the same level of the command hierarchy as the display command . The semicolon after the INP 'AM DEMOD' command told the Test Set that the next command (FILT1 '300Hz HPF') is on the same command level as the INP 'AM DEMOD' command.

A semicolon followed by a colon (;:) tells the Test Set that the next command is at the top level of the command hierarchy. The example below sets the RF Analyzer's Tune Frequency to 850 MHz, and sets the AF Analyzer's Input to FM Demod.

```
RFAN:FREQ 850 MHZ;:AFAN:INP 'FM DEMOD'
```

Using Question Marks to Query

The question mark (?) is used to query (read-back) an instrument setting or measurement value, and is entered immediately after a command. Queried information must be read into a variable before it can be displayed, printed, or used as a numeric value in your program.

Queried information is returned in the same format used to set the value (a queried numeric entry function returns numeric data; quoted string functions return quoted string information).

For example, the BASIC language commands ...

```
OUTPUT 714;"AFG1:DEST?"     !Query the AFGen1 To field.
ENTER 714;Afg1_to$          !Enter queried value into a variable.
PRINT Afg1_to$              !Print the queried value.
```

... print the string value of the AFGen1 To field; The printed value for this example is either AM, FM, or Audio Out (depending on the current field setting).

Specifying Units of Magnitude

Many numeric settings and measurements in the Test Set have one or more associated units of magnitude (V, mV, μV; Hz, kHz, MHz ...). Using manual operation, the units can be easily changed to display measurements and alter settings in the most convenient format.

HP-IB operation is similar to manual operation in that it lets you set values using any appropriate unit. However, using HP-IB, you can only read-back numeric values in fundamental units.

To be able to write efficient HP-IB programs, you must understand how the Test Set deals with different types of units.

Display Units (DUNits)

Display Units are those shown on the Test Set's front-panel *display*; the full set of units you use to manually control the instrument. For instance; Hz, kHz, MHz, and GHz can all be used to set and display the RF Generator frequency.

Changing Display Units. Use the **DUNits** syntax to change the Display Unit for any measurement or setting. For example, to change the Display Unit for the `TX Power` measurement from **W** to **dBm**, you would enter the following BASIC command:

```
OUTPUT 714;":MEAS:RFR:POW:DUN DBM"
```

DISPLAY UNITS	HP-IB SYNTAX EXAMPLES
GHz	`:MEAS:RFR:FREQ:ABS:DUN GHZ`
MHz	`:MEAS:RFR:FREQ:ABS:DUN MHZ`
kHz	`:MEAS:RFR:FREQ:ABS:DUN KHZ`
Hz	`:MEAS:RFR:FREQ:ABS:DUN HZ`
ppm	`:MEAS:RFR:FREQ:ERR:DUN PPM`
% Δ	`:MEAS:RFR:FREQ:ERR:DUN PCTDIFF`
V	`:MEAS:RFR:POW:DUN V`
mV	`:MEAS:RFR:POW:DUN MV`
μV	`:RFG:AMPL:DUN UV`
dBμV	`:RFG:AMPL:DUN DBUV`
W	`:MEAS:RFR:POW:DUN W`
mW	`:MEAS:RFR:POW:DUN MW`
dBm	`:MEAS:RFR:POW:DUN DBM`
db	`:MEAS:AFR:DISTN:DUN DB`
%	`:MEAS:AFR:DISTN:DUN PCT`
s	`:DEC:FGEN:GATE:DUN S`
ms	`:DEC:FGEN:GATE:DUN MS`

Reading-Back Display Units. Use the **DUNits?** syntax for a measurement or setting to read-back its Display Unit. For example, to see what the Display Unit is for the `TX Power` measurement, you would enter these BASIC commands:

```
OUTPUT 714;"MEAS:RFR:POW:DUNits?" !Read Display Units for TX Power.
ENTER 714;A$      !Enter the returned value into a string variable.
PRINT A$          !Print the units.
```

The returned value will be whatever is shown on the Test Set's front-panel display for this measurement: dBm, V, mV, dBuV, or W. (Note: all returned characters are in upper case. Example: dBuV displayed returns DBUV.)

Guidelines for Display Units.

- Querying measurements or settings over HP-IB always returns numeric values in HP-IB Units, regardless of the current Display Unit.
- You can always use any appropriate Display Unit for a setting or measurement to display the value on the Test Set, regardless of its HP-IB Unit.
- The Display Unit for a numeric field is not affected when changing the field's value over HP-IB, regardless of the unit used to set the field's value.

  For example, if the `AFGen1 Freq` field is set to 25.0000 kHz, and you send the command AFG1:FREQ 10 HZ to change it to 10 Hz, the instrument displays '0.0100 kHz'; not 10 Hz.

HP-IB UNITs

An HP-IB Unit is the fundamental unit used to interpret each setting and measurement when using HP-IB (such as Hz for frequencies, Watts for power, and Volts for voltages). Changing HP-IB units has no affect on the front-panel Display Units.

Changing HP-IB Units. Use the UNITs syntax to change the HP-IB unit. Only the HP-IB units for power-related settings and measurements can be changed.

Table 2-1. HP-IB Units That Can Be Changed

Function	Available HP-IB Units
TX Power measurement	W or dBm
SINAD measurement	PCT or dB
RF Gen. Amplitude setting	W or dBm

For example, to change the HP-IB for the `TX Power` measurement from **W** to **dBm**, you would enter the following BASIC command:

```
OUTPUT 714;"MEAS:RFR:POW:UNIT DBM"
```

Reading-back the HP-IB UNITs over HP-IB returns the new setting, even though the Display Units on the front-panel have not changed for this measurement.

Reading-Back HP-IB Units. Use the **UNITs?** syntax for a measurement or setting to determine its HP-IB unit. For example, to see what the HP-IB Unit is for the `TX Power` measurement, you would enter these BASIC commands:

```
OUTPUT 714;"MEAS:RFR:POW:UNITs?"    !Read HP-IB Units for TX Power.
ENTER 714;A$        !Enter the returned value into a string variable.
PRINT A$            !Print the units.
```

In this example, 'W' would be printed (Watts).

Guidelines for HP-IB Units.

- When changing a numeric setting (such as `AFGen1 Freq`), any non-HP-IB unit must be specified in the HP-IB command, or the HP-IB Unit is assumed by the Test Set.

  For example, if the `RF Gen Freq` field is already set to 950 **MHz**, and you send the command RFG:FREQ 900 the Test Set thinks you are trying to set it to 900 **Hz**, and results in an "Input value out of range" error. Sending the command RFG:FREQ 900 MHZ would set the value to 900 **MHz**.

- The returned numeric value for a measurement is *always* in HP-IB units, regardless of the displayed units. The numeric value is expressed in scientific notation.

  For example, if the `TX Frequency` measurement is shown as 150.000000 MHz on the Test Set's display, the returned value over HP-IB is 1.5000000E+008 (or 1.5×10^8). To convert the returned value to a non-HP-IB unit, you must enter the value into a conversion formula in your program.

Attribute Units (AUNits)

Attribute units are used with the measurement Data Functions: REF SET, METER, LO and HI LIMIT. AUNits provide a way to remotely query and change the units used to define a Data Function's value (such as the Reference level).

For example, Distortion is usually expressed in units of **PCT** (%) (although dB can be used), but when the REF SET function is used, the displayed unit on the front panel and value returned over HP-IB are always in **dB**. By using the AUNits syntax, you can change and read-back the units associated with the *measurement reference*, independent of what the displayed measurement is on the front panel.

Attribute units use the same set of units as HP-IB units, but are only used with the measurement Data Functions.

Changing Attribute Units. Some measurements can use more than one Attribute unit. For example, the audio Distortion measurement allows you to set a Reference in units of **dB** or **PCT** (%).

To change the Attribute unit for a referenced Distortion measurement from % to **dB**, you would enter the following BASIC command:

```
OUTPUT 714;"MEAS:AFR:DISTN:AUN DB"
```

The Display Units shown on the Test Set's front panel are not affected by changing Attribute Units. However, the Reference value and unit returned over HP-IB are in the new Attribute unit you specified using the above command.

Reading-back Attribute Units. Use the **AUNits?** syntax to read-back the Attribute unit for an appropriate measurement.

The following BASIC language example shows how AUNits could be used to read-back a Distortion reference level you have already set:

```
OUTPUT 714;"MEAS:AFR:DISTN:REF?" !Read the numeric value of the
                                 !reference you set for measuring Distortion.
ENTER 714;A$ !Enter the reference value into a variable.
OUTPUT 714;"MEAS:AFR:DISTN:AUN?" !Read the Attribute units used to
                                 !set the reference.
ENTER 714;B$ !Enter the Attribute units into a variable.
PRINT A$;B$  !Print out the variables in the form <VALUE><UNITS>
```

If you had set a reference of 25%, 2.50000000E+001PCT is printed.

Using STATE Commands

STATe commands correspond to using the [ON/OFF] key to turn measurements, Data Functions, and settings on and off.

Use 1 or ON to turn something on; 0 or OFF to turn something off. When queried, the returned value will be either '1' (on) or '0' (off).

For example, to turn off AF Generator 1 FM and the TX Power measurement, and turn on measurement averaging for the FM Deviation measurement, enter the following BASIC commands:

```
OUTPUT 714;"AFG1:FM:STAT OFF"       !Turn off FM source AFG1. *
OUTPUT 714;"MEAS:RFR:POW:STAT 0"      !Turn off TX Power
OUTPUT 714;"MEAS:AFR:FM:REF:STAT ON" !Turn on REF SET for FM Dev.
```

*This assumes the `AFGen1 To` field is set to FM.

State Command Guidelines

- Measurements that are displayed as numbers, or as analog meters using the METER function, can be turned on and off.
- All of the DATA FUNCTIONS can be turned on and off.
- Any function that generates a signal can be turned on and off. This includes the RF Generator and Tracking Generator amplitude, and AF Generators 1 and 2 levels (including the Encoder functions).
- You can not turn off the Oscilloscope's trace when you are displaying the **OSCILLOSCOPE** screen, or the Spectrum Analyzer's trace when you are displaying the **SPECTRUM ANALYZER** screen.

Sample HP-IB Program

This program was written on an HP 9000 Series 200 controller using Rocky Mountain BASIC (RMB). To run this program directly in the Test Set's IBASIC Controller, use exclamation marks (!) to comment-out lines 60, 440, 450, and 460.

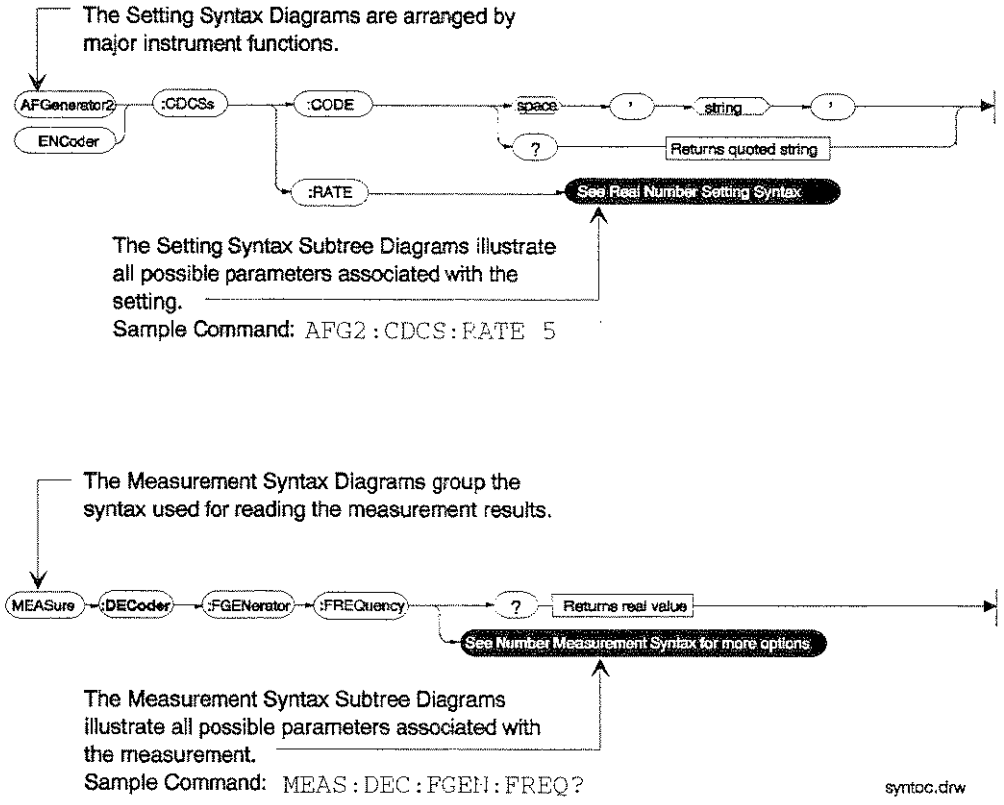
```
10  ! This program creates an FM carrier, measures and displays the
20  ! deviation, and draws the modulation waveform from the oscilloscope
30  ! to your external controller's display.
40  ! The carrier is generated and analyzed through the uncalibrated input
50  ! path (so no external cables are required).
60  GCLEAR                          !Clear any presently-displayed graphics.
70  Bus=7                           ! Using bus select code 7.
80  Dut=100*Bus+14                  ! Test Set HP-IB address 14
90  CLEAR Bus                       ! Good practice to clear the bus
100 CLEAR SCREEN                    ! Clear the CRT
110 OUTPUT Dut;"*RST"               ! Preset the Test Set
120 OUTPUT Dut;"DISP DUPL"          ! Display the DUPLEX TEST screen
130 OUTPUT Dut;"RFG:AMPL -14 DBM"       ! RF Gen Amptd to -14 dBm
140 OUTPUT Dut;"AFAN:INP 'FM Demod'"    ! AF Analyzer input FM Demod
150 OUTPUT Dut;"AFAN:DET 'Pk+-Max'"     ! AF Analyzer det peak +/-Max
160 !
170 ! The following trigger guarantees the instrument will auto-tune and
180 ! auto-range to the input signal before measuring.
190 !
200 OUTPUT Dut;"TRIG"               ! Trigger all active measurements
210 OUTPUT Dut;"MEAS:AFR:FM?"       ! Request an FM deviation measurement
220 ENTER Dut;Dev                   ! To accept the FM measurement
230 PRINT USING "K,D.DDD,K";"Measured FM = ",Dev/1000," kHz peak."
240 DISP "'Continue' when ready..."   ! Set up and use a USER key
250 ON KEY 1 LABEL "Continue",15 GOTO Proceed
260 LOOP                            ! Loop until the key is pressed
270 END LOOP
280 Proceed: OFF KEY                ! Turn off the 'Continue' key
290 DISP ""                         ! Clear the user prompt
300 !
310 ! Measure and plot an oscilloscope trace to see the waveform shape.
320 DIM Trace(0:416)                ! 417 trace points
330 OUTPUT Dut;"DISP OSC"           ! Display the SCOPE screen
340 OUTPUT Dut;"TRIG"               ! Trigger all active measurements
350 OUTPUT Dut;"MEAS:OSC:TRAC?"     ! Request the oscilloscope trace
360 ENTER Dut;Trace(*)              ! Accept the oscilloscope trace
370 !
380 ! CRT is (X,Y)=(0,0) in lower left corner to (399,179) upper right.
390 ! (Each pixel is about 0.02 mm wide by 0.03 mm tall, not square.)
400 ! Scale vertically for 0 kHz dev center-screen and +4 kHz dev top
410 ! of screen.  Leave the next line for external control, or
420 ! comment it out for IBASIC (Test Set stand-alone) control.
430 !
440 PLOTTER IS CRT,"98627A" !Your display may have a different specifier.
450 GRAPHICS ON  !Enable graphics to plot the waveform.
460 WINDOW 0,399,0,179
470 !
480 MOVE 0,89.5+Trace(0)/4000*89.5
490 FOR I=1 TO 416
500 DRAW I/416*399,89.5+Trace(I)/4000*89.5
510 NEXT I
520 END
```

<— —>

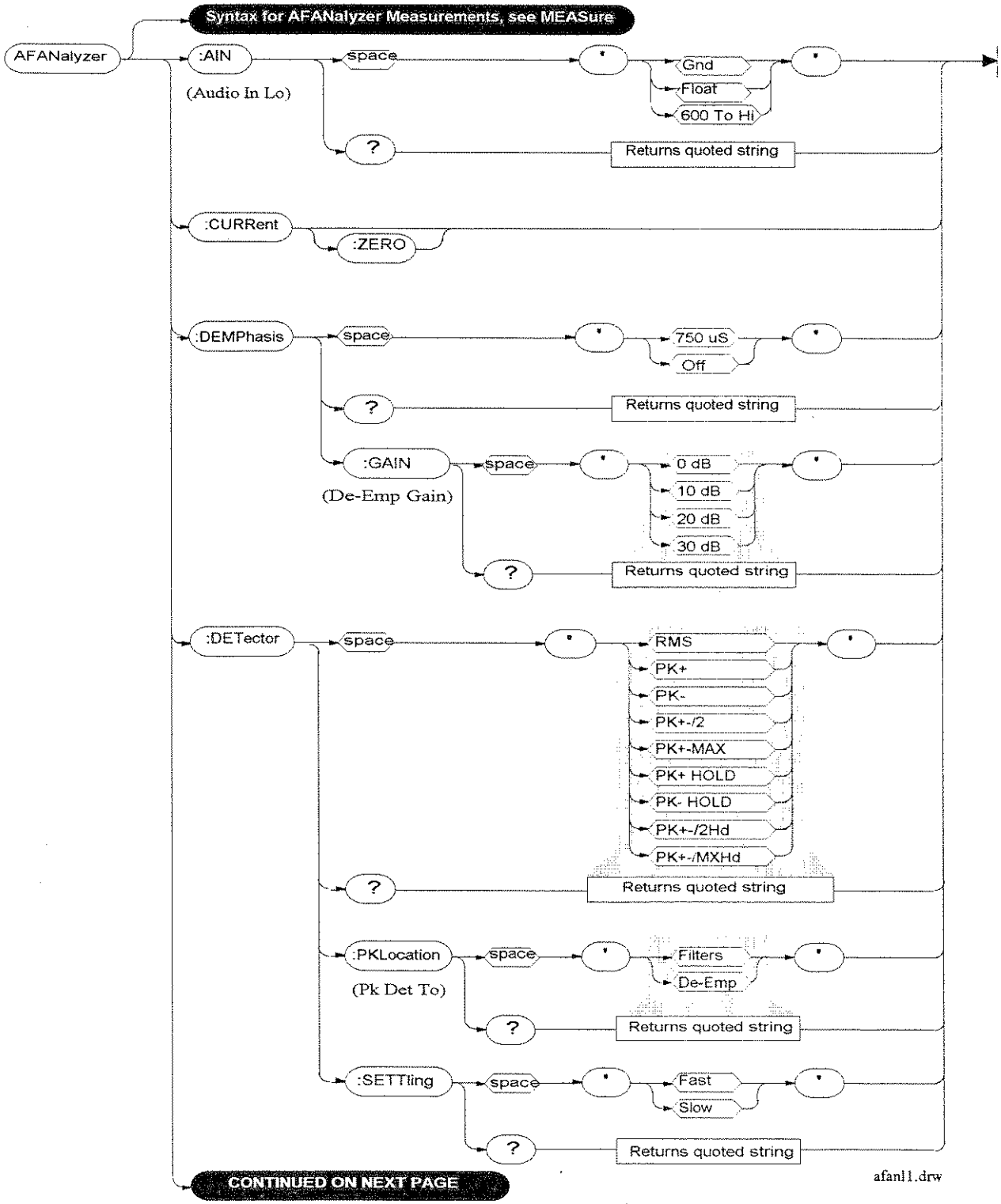
3

HP-IB Commands

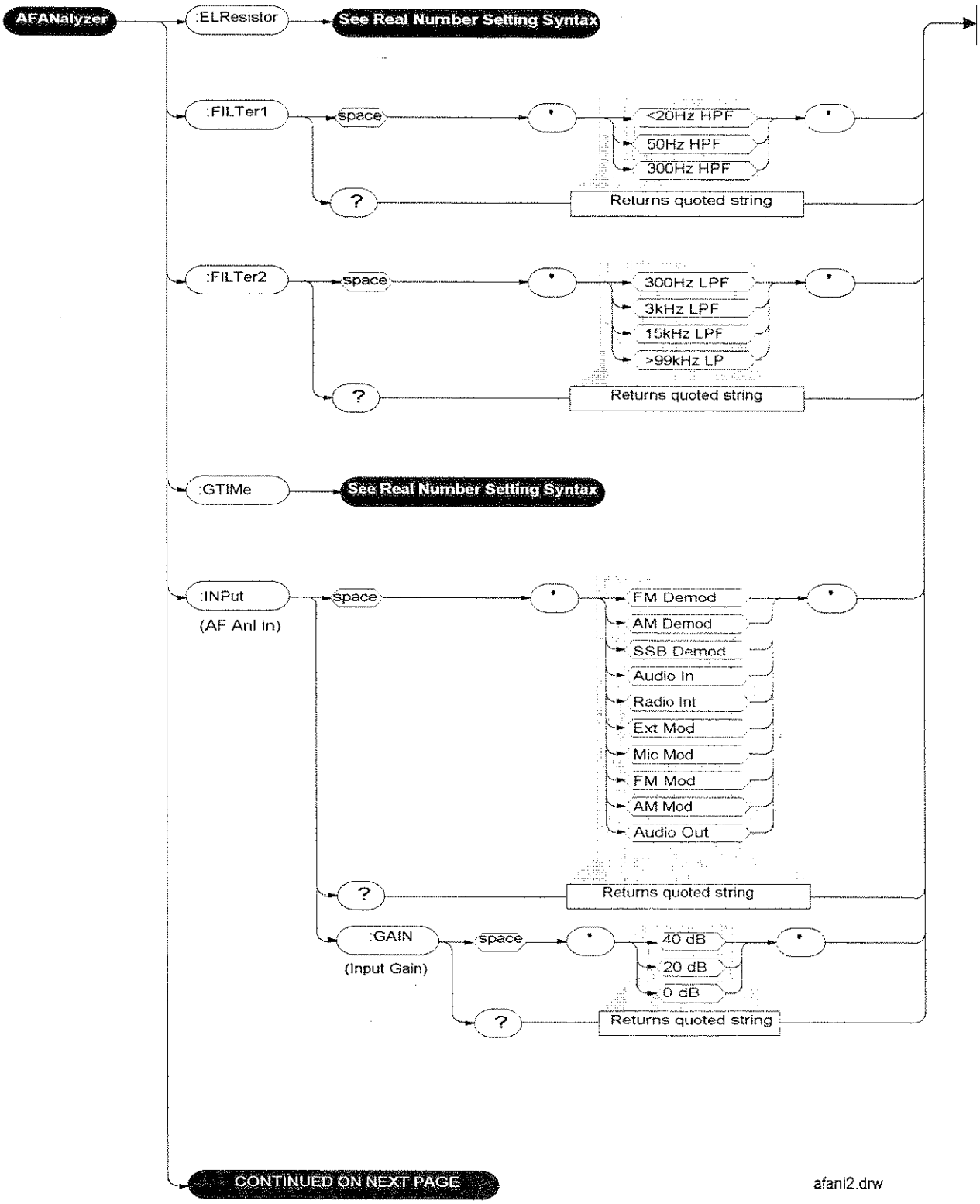
HP-IB Syntax Diagrams



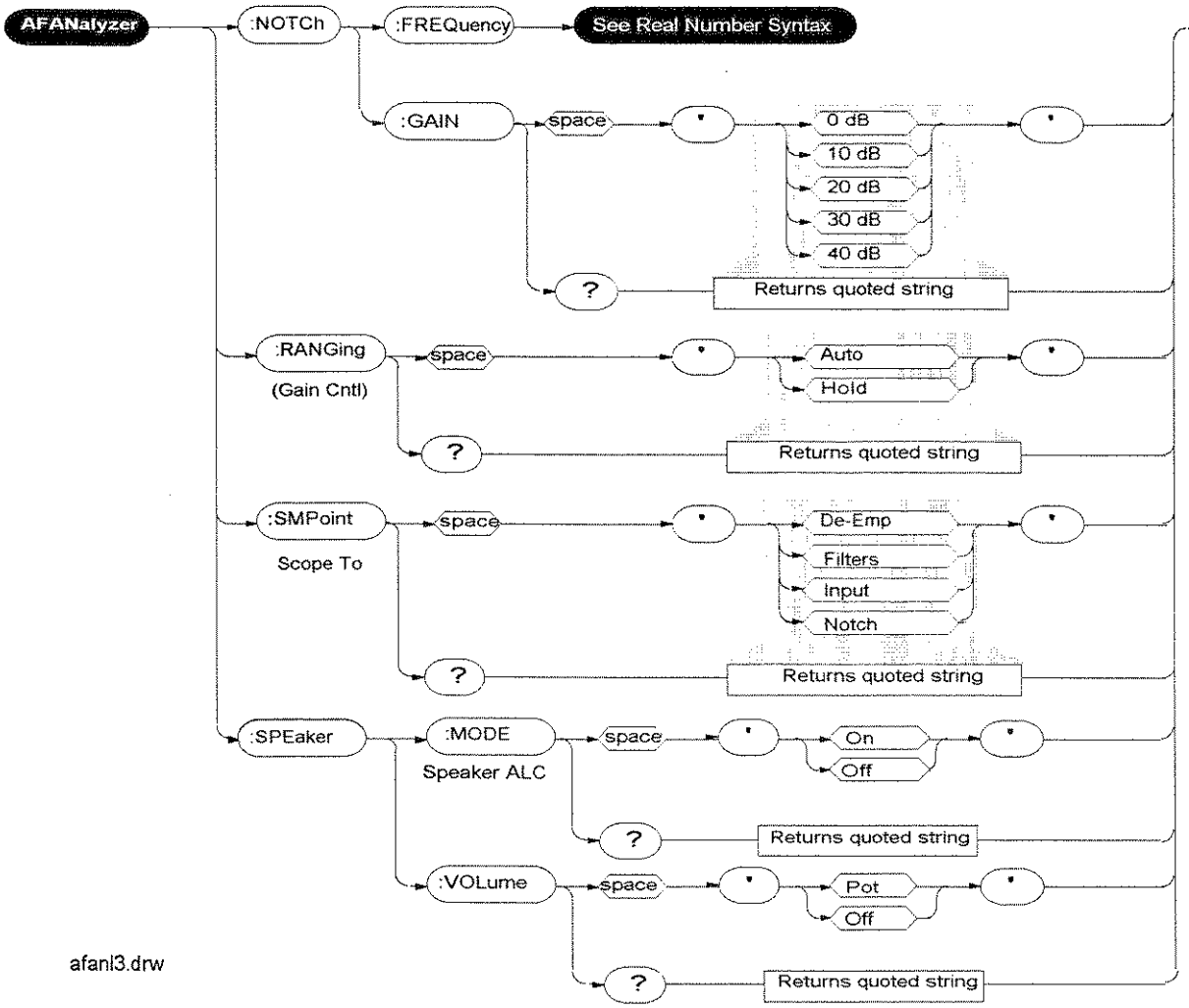
AF Analyzer


Syntax for AFANalyzer Measurements, see MEASure
AFANalyzer
:AIN
(Audio In Lo)
space
'
Gnd
Float
600 To Hi
'
?
Returns quoted string
:CURRent
:ZERO
:DEMPhasis
space
'
750 uS
Off
'
?
Returns quoted string
:GAIN
(De-Emp Gain)
space
'
0 dB
10 dB
20 dB
30 dB
'
?
Returns quoted string
:DETector
space
'
RMS
PK+
PK-
PK+-/2
PK+-MAX
PK+ HOLD
PK- HOLD
PK+-/2Hd
PK+-/MXHd
'
?
Returns quoted string
:PKLocation
(Pk Det To)
space
'
Filters
De-Emp
'
?
Returns quoted string
:SETTling
space
'
Fast
Slow
'
?
Returns quoted string
CONTINUED ON NEXT PAGE
afanl1.drw

AF Analyzer (continued)

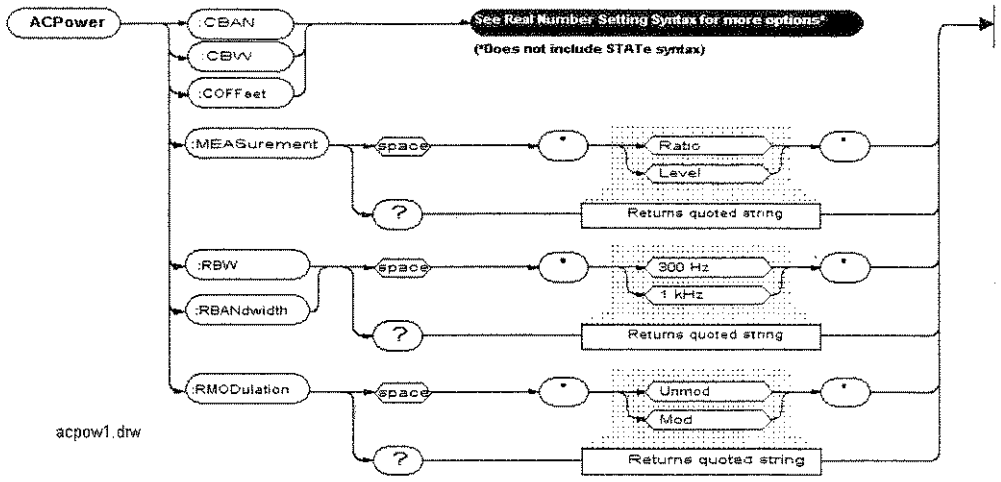


AF Analyzer (continued)

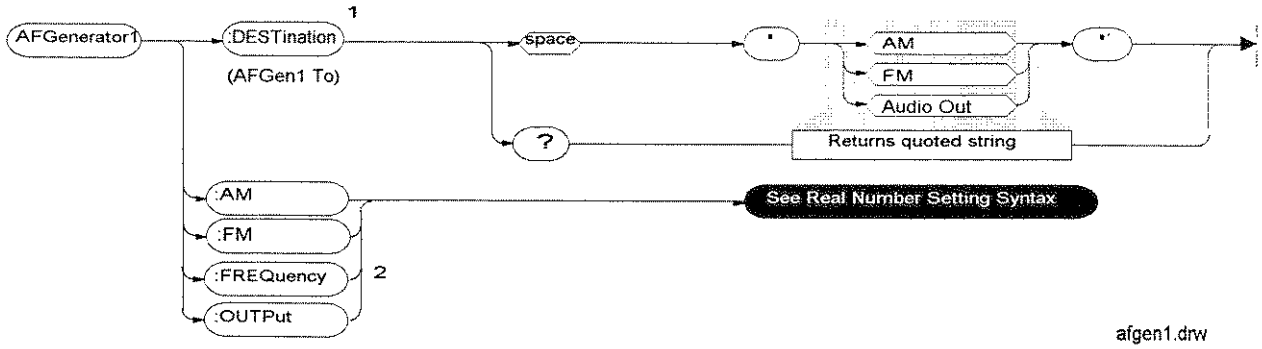


afanl3.drw

Adjacent Channel Power (ACP)



AF Generator 1



[1]In setting AFG 1 & 2, you must first set a destination, then the modulation depth (or Audio Out voltage), then the rate (frequency).

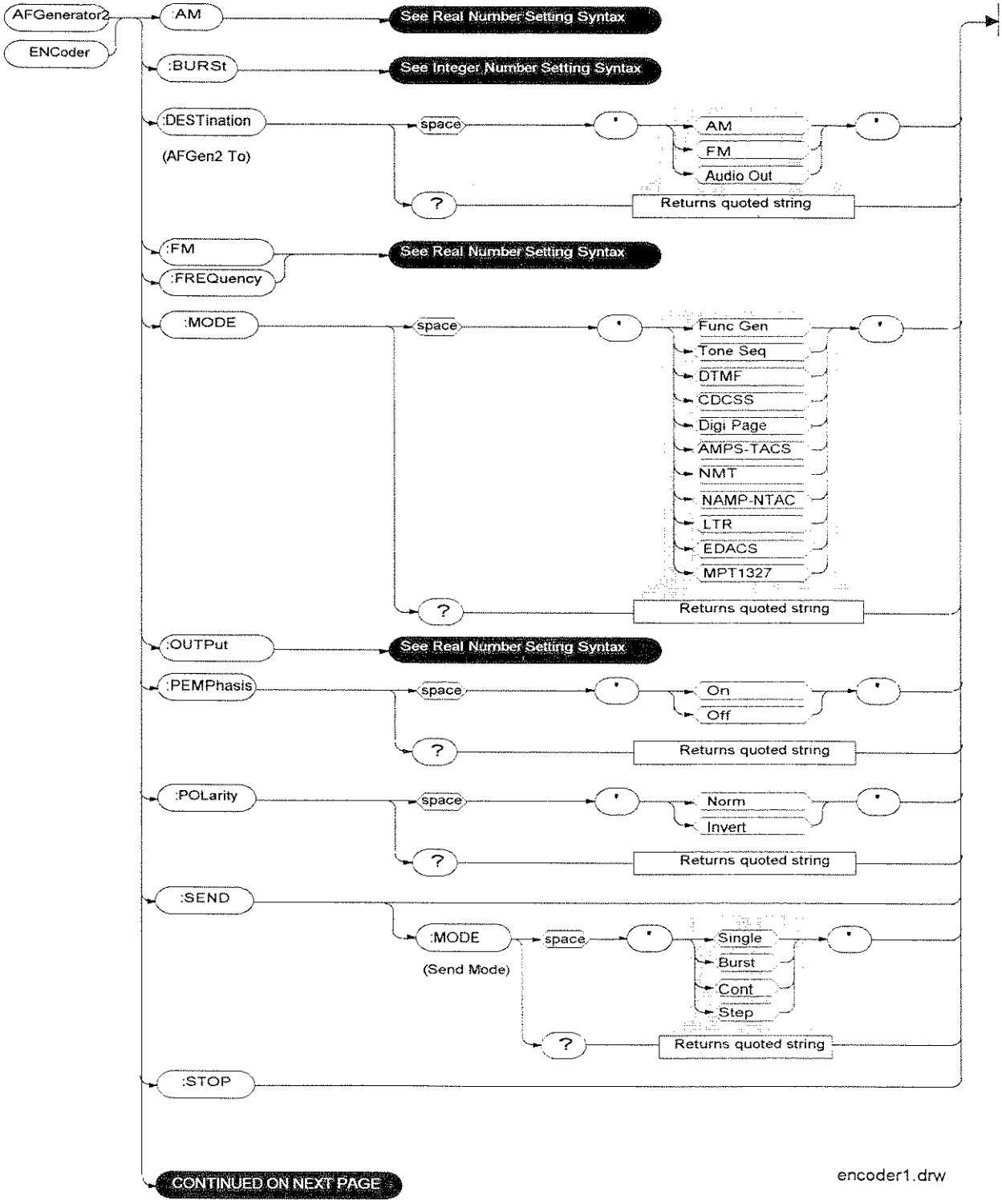
[2]AM, FM, and Output set the amplitude when AM, FM, or Audio Out is first specified as the DESTination. FREQuency is the modulation rate or Audio Out frequency.

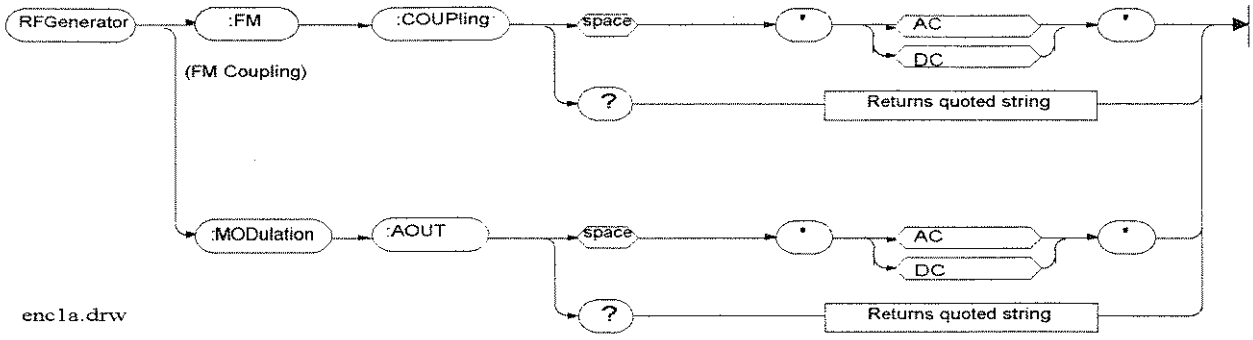
AF Generator 2 Pre-Modulation Filters

One of four pre-modulation filter settings is *automatically* selected for each Encoder Mode to improve performance. They can only be changed using HP-IB commands; however, we recommend you do not change this setting. The syntax to change or query this setting are -

```
AFG2:FILTER:MODE 'On|Off'                                    (select one)
AFG2:FILTER:MODE?                        (query the current mode setting)
AFG2:FILTER 'None|20kHz LPF|250Hz LPF|150Hz LPF'             (select one)
AFG2:FILTER?                           (query the current filter setting)
```

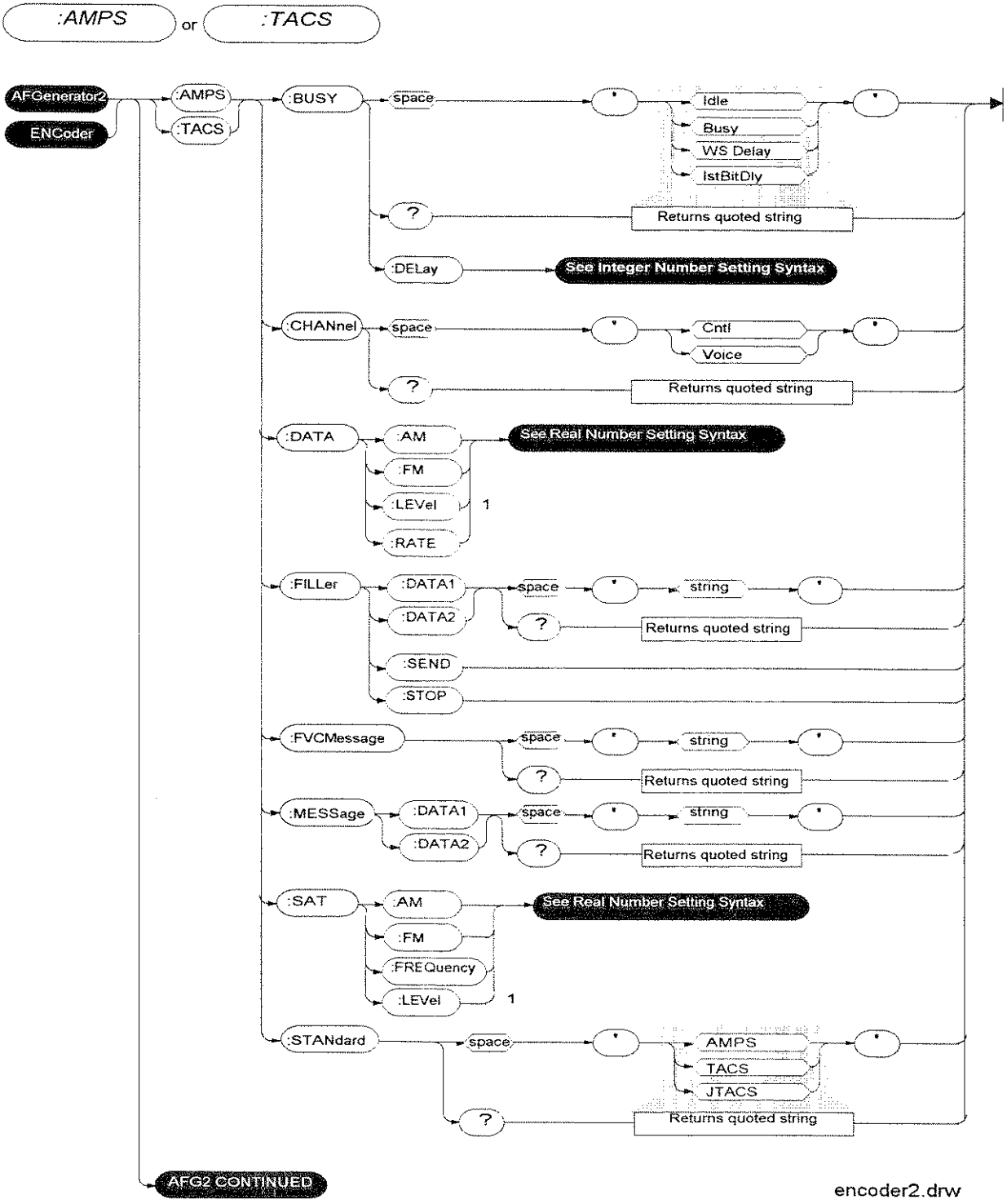
AF Generator 2/Encoder




RFGenerator
:FM
(FM Coupling)
:COUPling
space
'
AC
DC
'
?
Returns quoted string
:MODulation
:AOUT
space
'
AC
DC
'
?
Returns quoted string

enc1a.drw

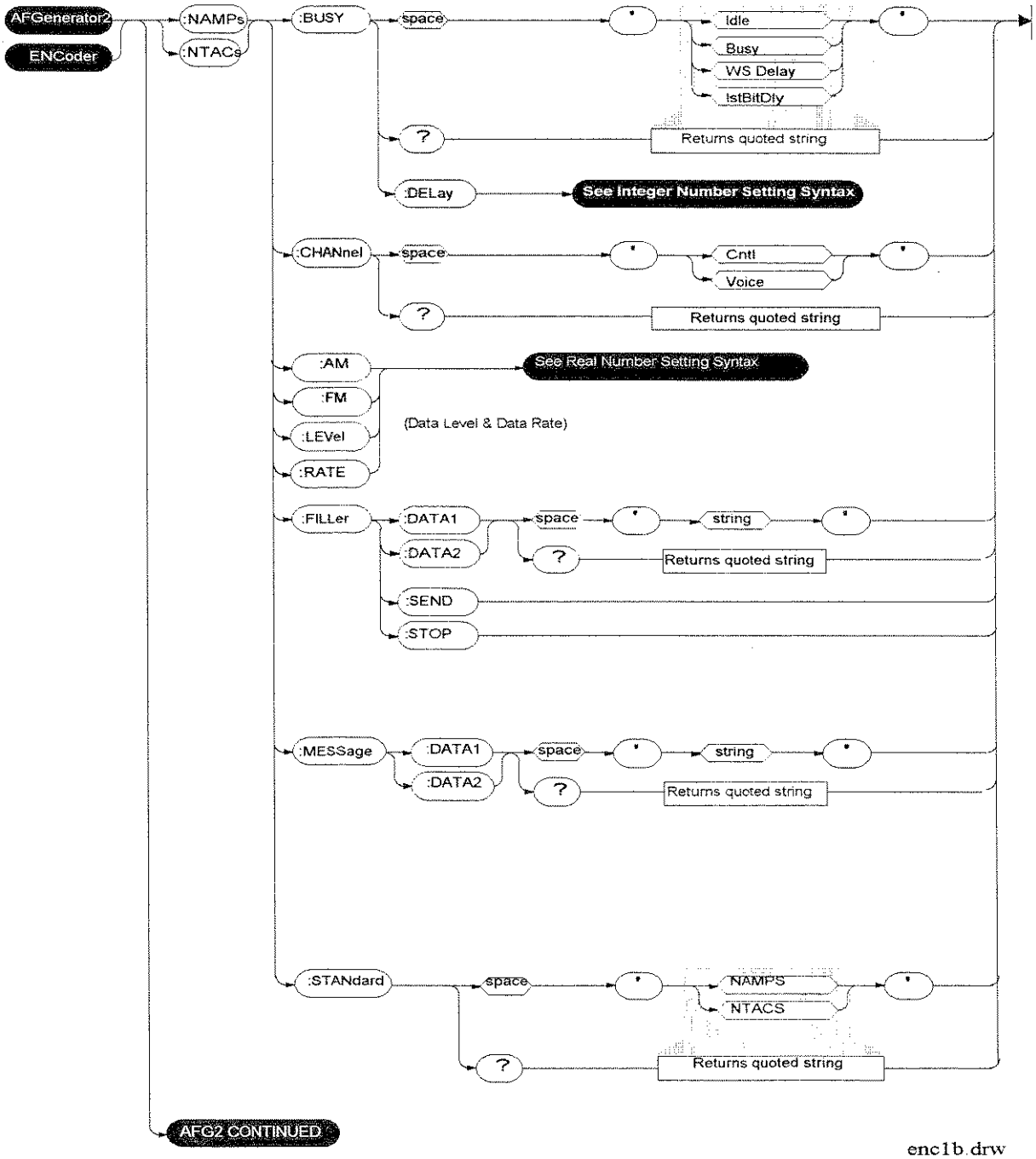
AF Generator 2/Encoder (continued)



[1]AM, FM, and LEVel correspond to the AFGen2 To setting.

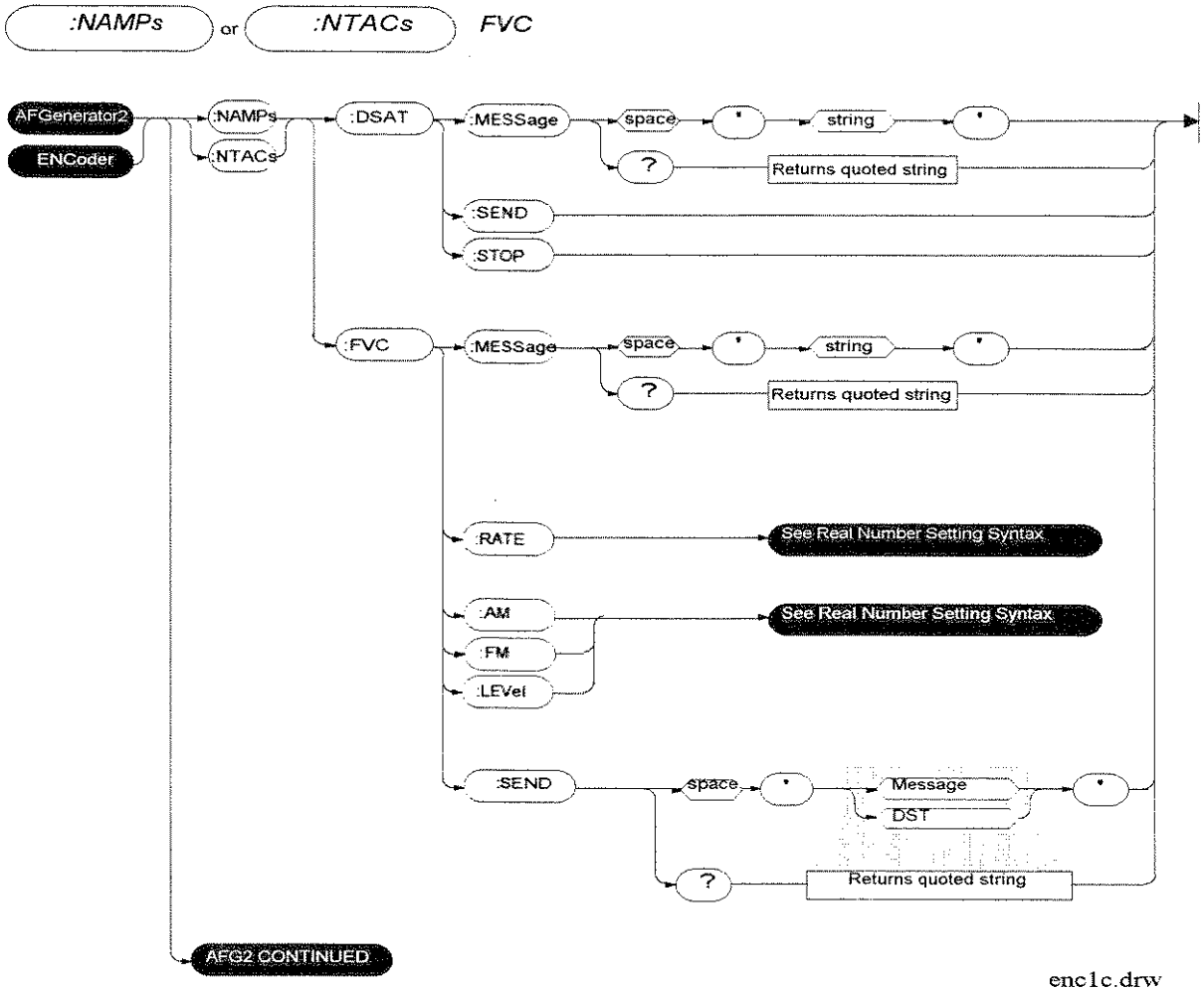
AF Generator 2/Encoder (continued)

:NAMPs or :NTACs FOCC



enc1b.drw

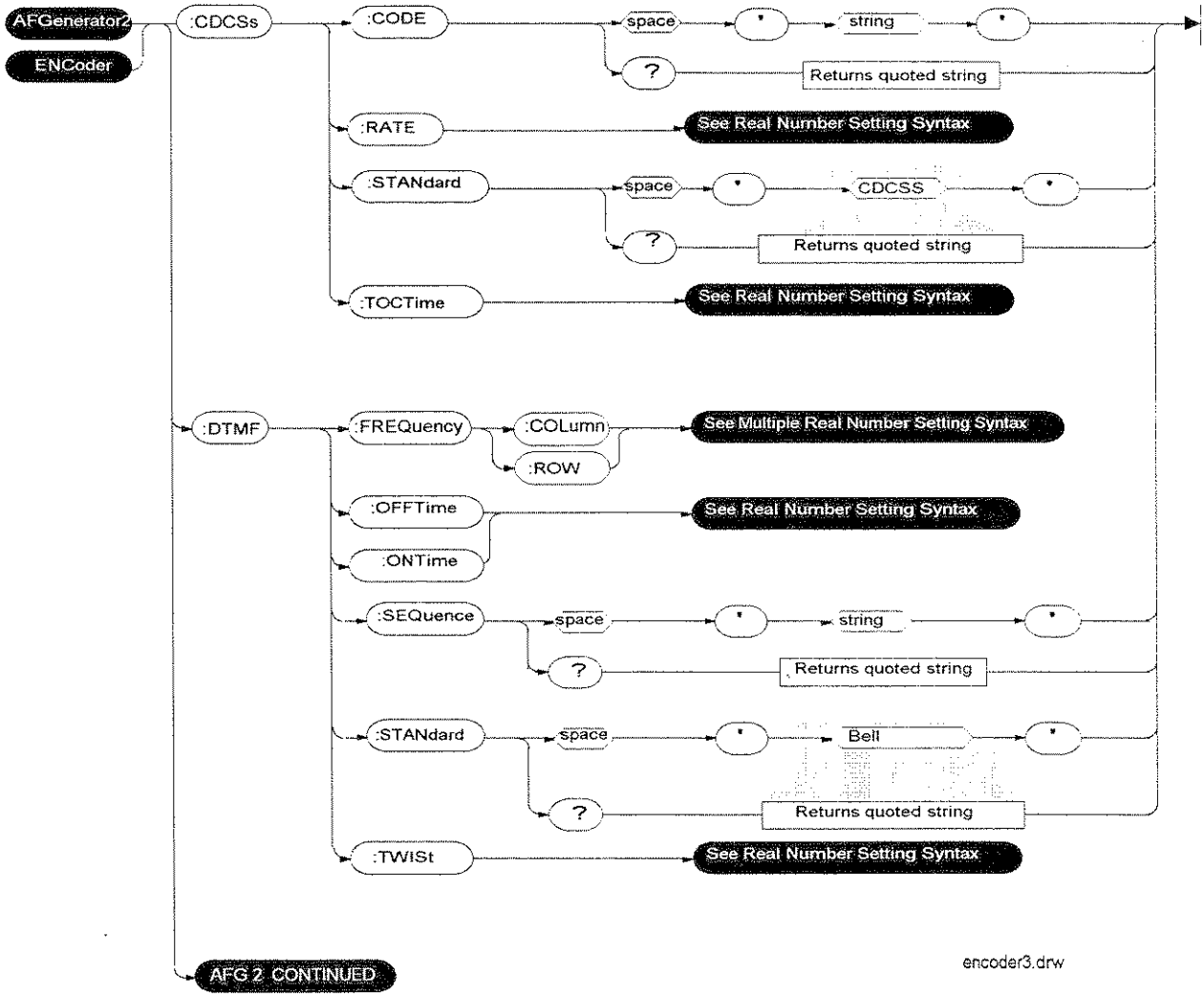
AF Generator 2/Encoder (continued)



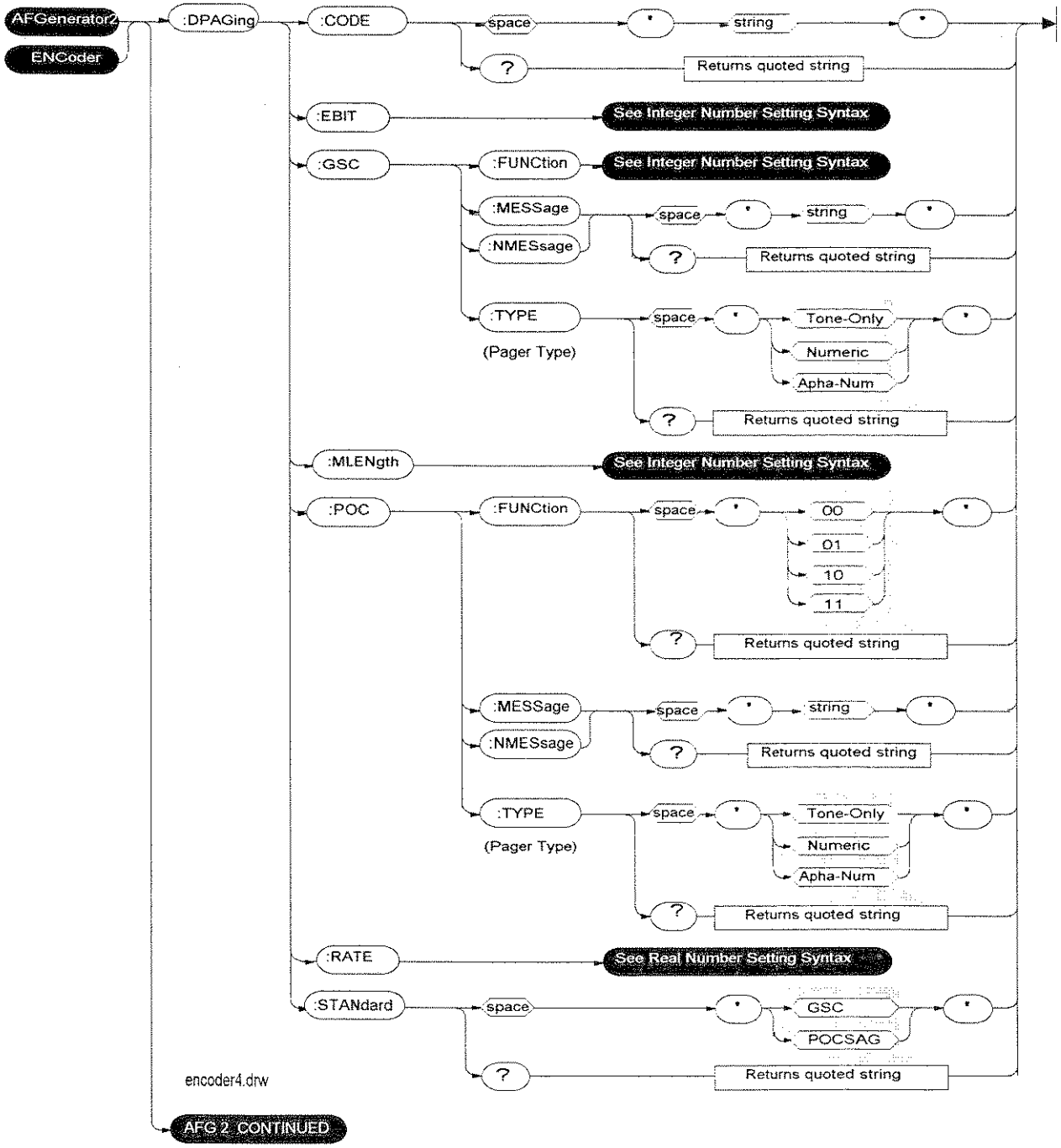
enc1c.drw

AF Generator 2/Encoder (continued)

:CDCSs and :DTMF

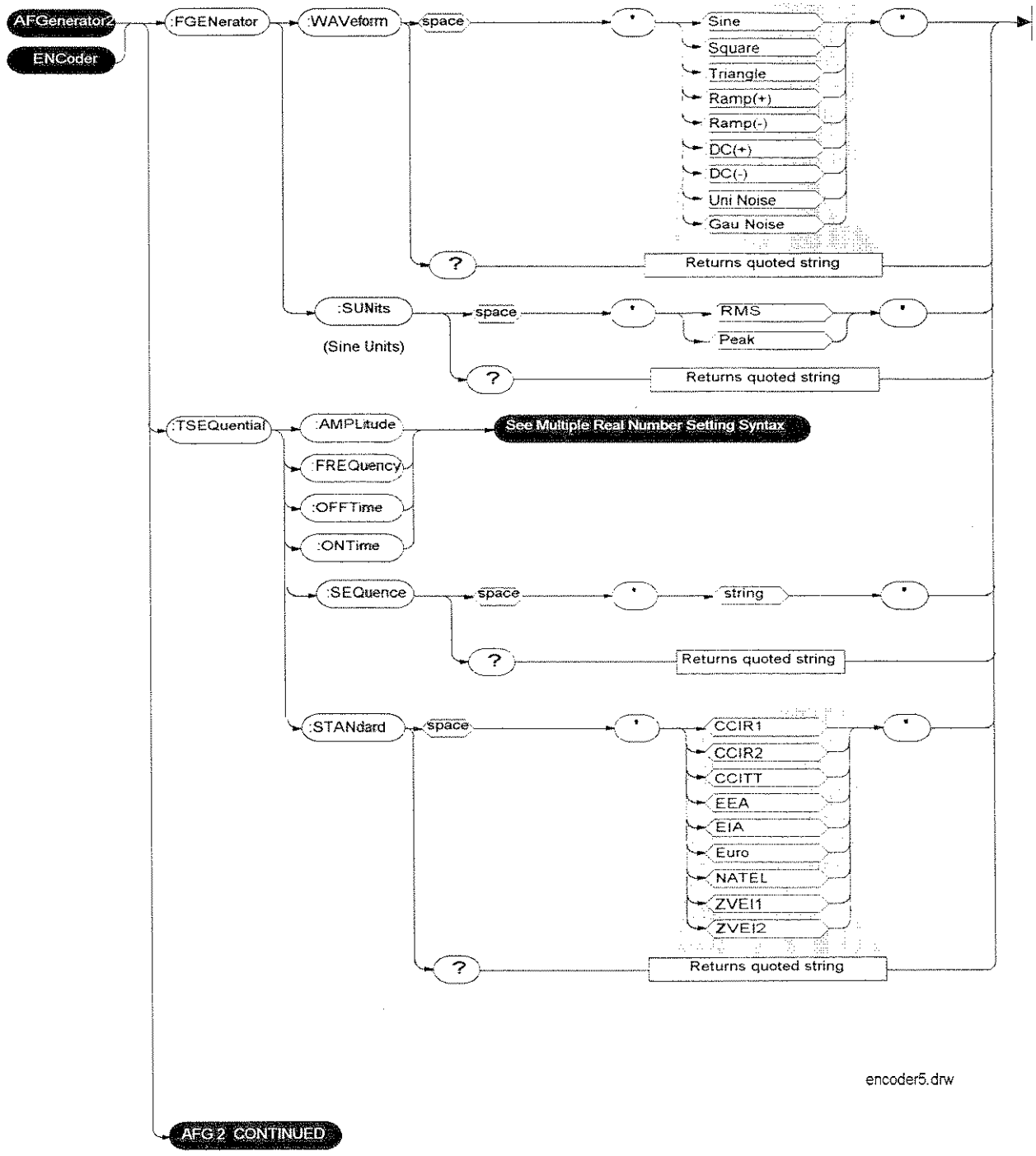


:DPAGing


AFGenerator2
ENCoder
:DPAGing
:CODE
space
'
string
'
?
Returns quoted string
:EBIT
See Integer Number Setting Syntax
:GSC
:FUNCtion
See Integer Number Setting Syntax
:MESSage
:NMESsage
space
'
string
'
?
Returns quoted string
:TYPE
(Pager Type)
space
'
Tone-Only
Numeric
Apha-Num
'
?
Returns quoted string
:MLENgth
See Integer Number Setting Syntax
:POC
:FUNCtion
space
'
00
01
10
11
'
?
Returns quoted string
:MESSage
:NMESsage
space
'
string
'
?
Returns quoted string
:TYPE
(Pager Type)
space
'
Tone-Only
Numeric
Apha-Num
'
?
Returns quoted string
:RATE
See Real Number Setting Syntax
:STANdard
space
'
GSC
POCSAG
'
?
Returns quoted string
encoder4.drw
AFG 2 CONTINUED

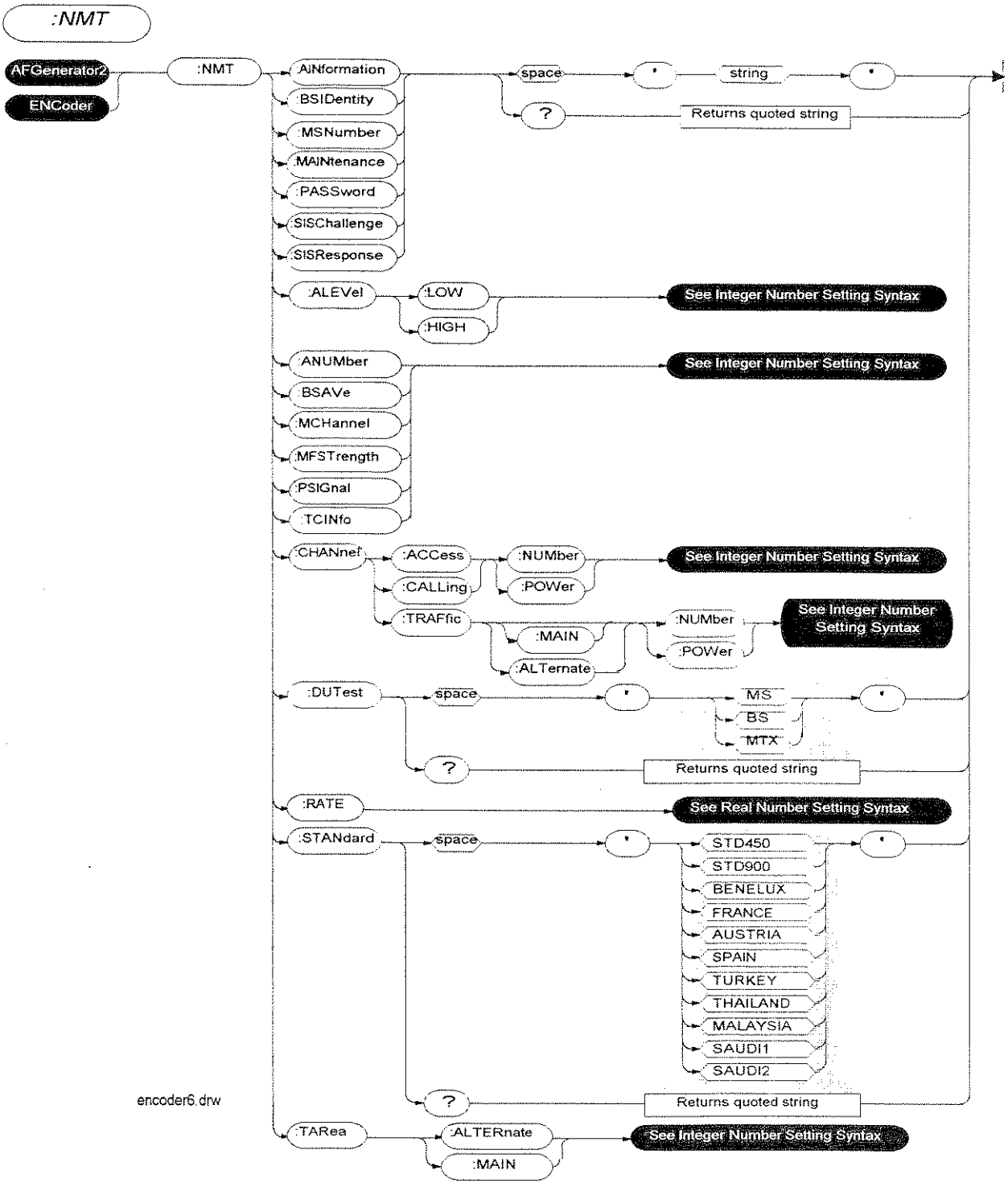
AF Generator 2/Encoder (continued)

:FGENerator and :TSEQuential

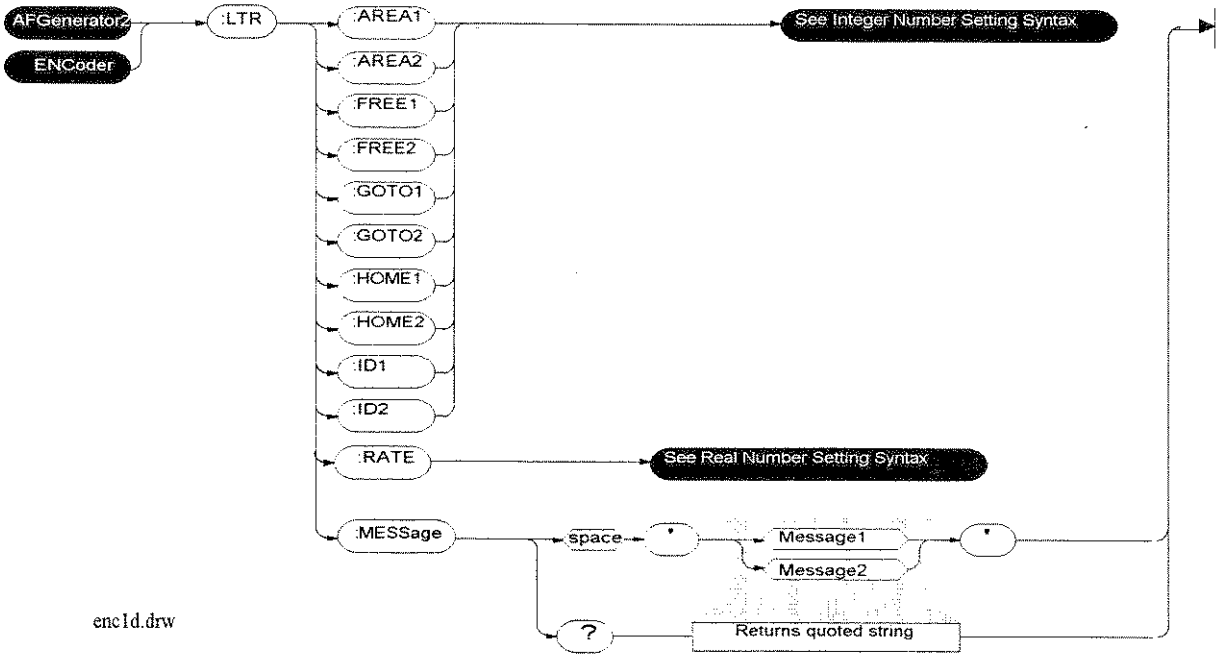


encoder5.drw

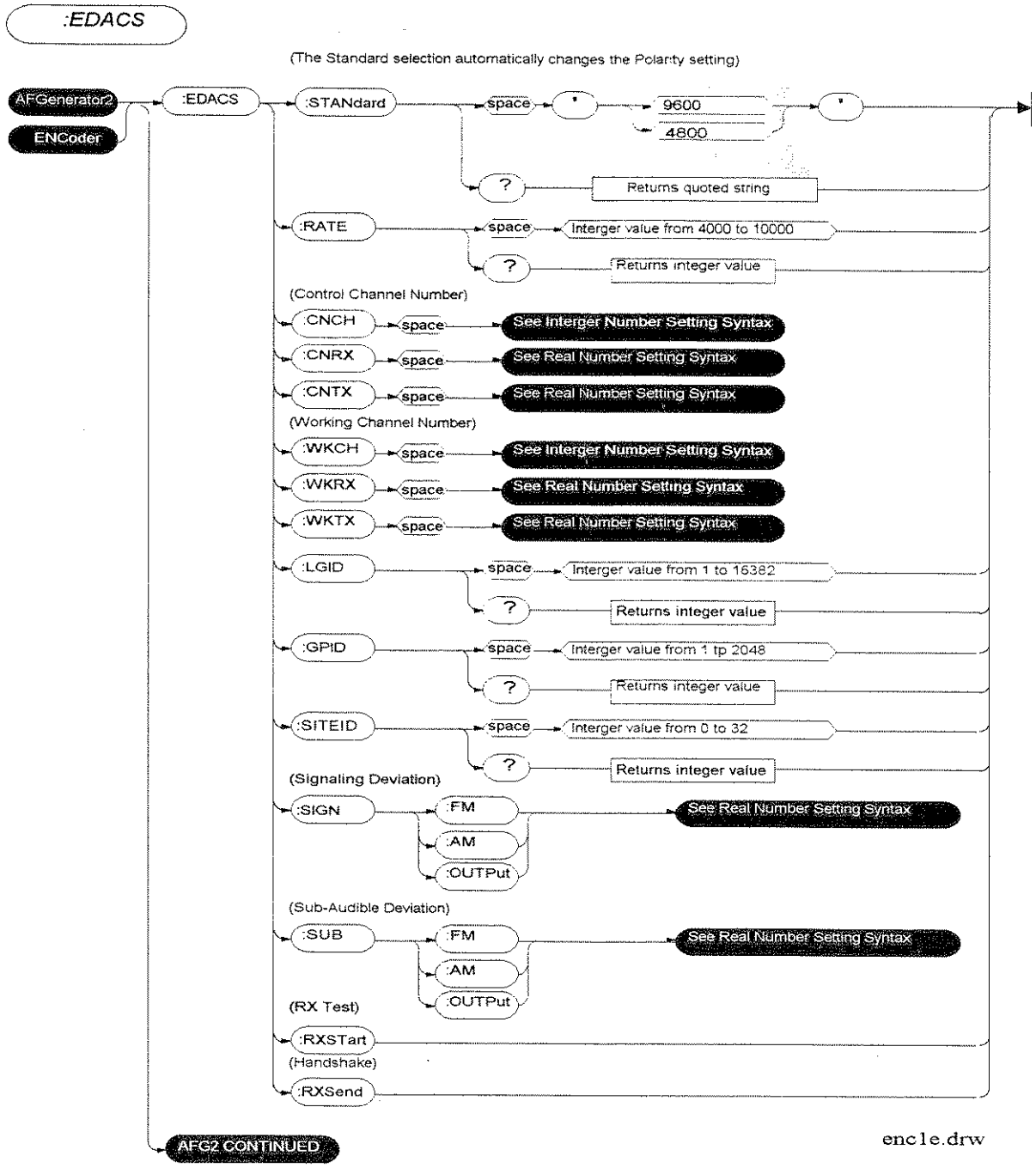
AF Generator 2/Encoder (continued)

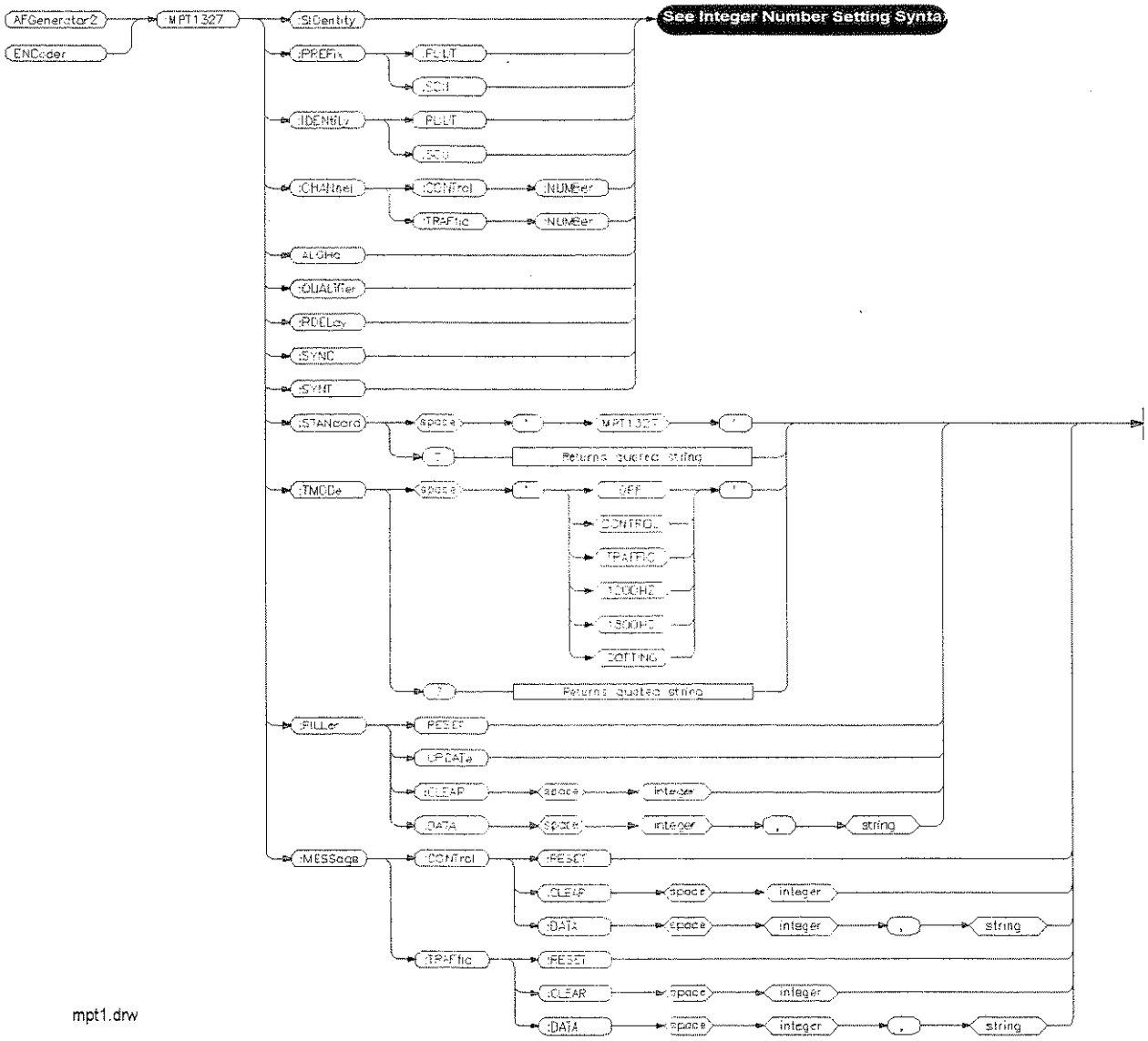


AF Generator 2/Encoder (continued)



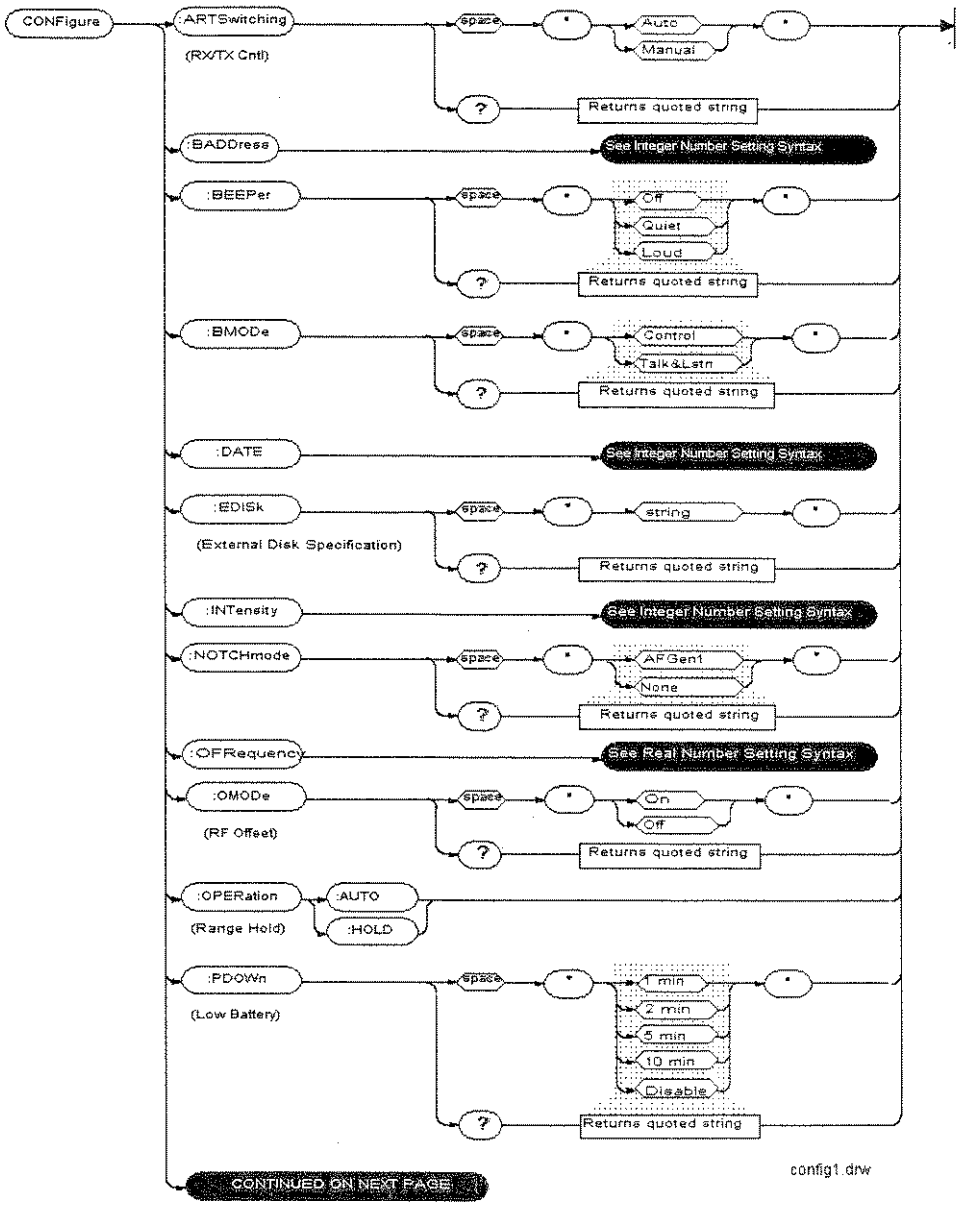
enc1d.drw


:EDACS
(The Standard selection automatically changes the Polarity setting)
AFGenerator2
ENCoder
:EDACS
:STANdard
space
'
9600
4800
'
?
Returns quoted string
:RATE
space
Interger value from 4000 to 10000
?
Returns integer value
(Control Channel Number)
:CNCH
space
See Interger Number Setting Syntax
:CNRX
space
See Real Number Setting Syntax
:CNTX
space
See Real Number Setting Syntax
(Working Channel Number)
:WKCH
space
See Interger Number Setting Syntax
:WKRX
space
See Real Number Setting Syntax
:WKTX
space
See Real Number Setting Syntax
:LGID
space
Interger value from 1 to 16382
?
Returns integer value
:GPID
space
Interger value from 1 tp 2048
?
Returns integer value
:SITEID
space
Interger value from 0 to 32
?
Returns integer value
(Signaling Deviation)
:SIGN
:FM
:AM
:OUTPut
See Real Number Setting Syntax
(Sub-Audible Deviation)
:SUB
:FM
:AM
:OUTPut
See Real Number Setting Syntax
(RX Test)
:RXSTart
(Handshake)
:RXSend
AFG2 CONTINUED
enc1e.drw

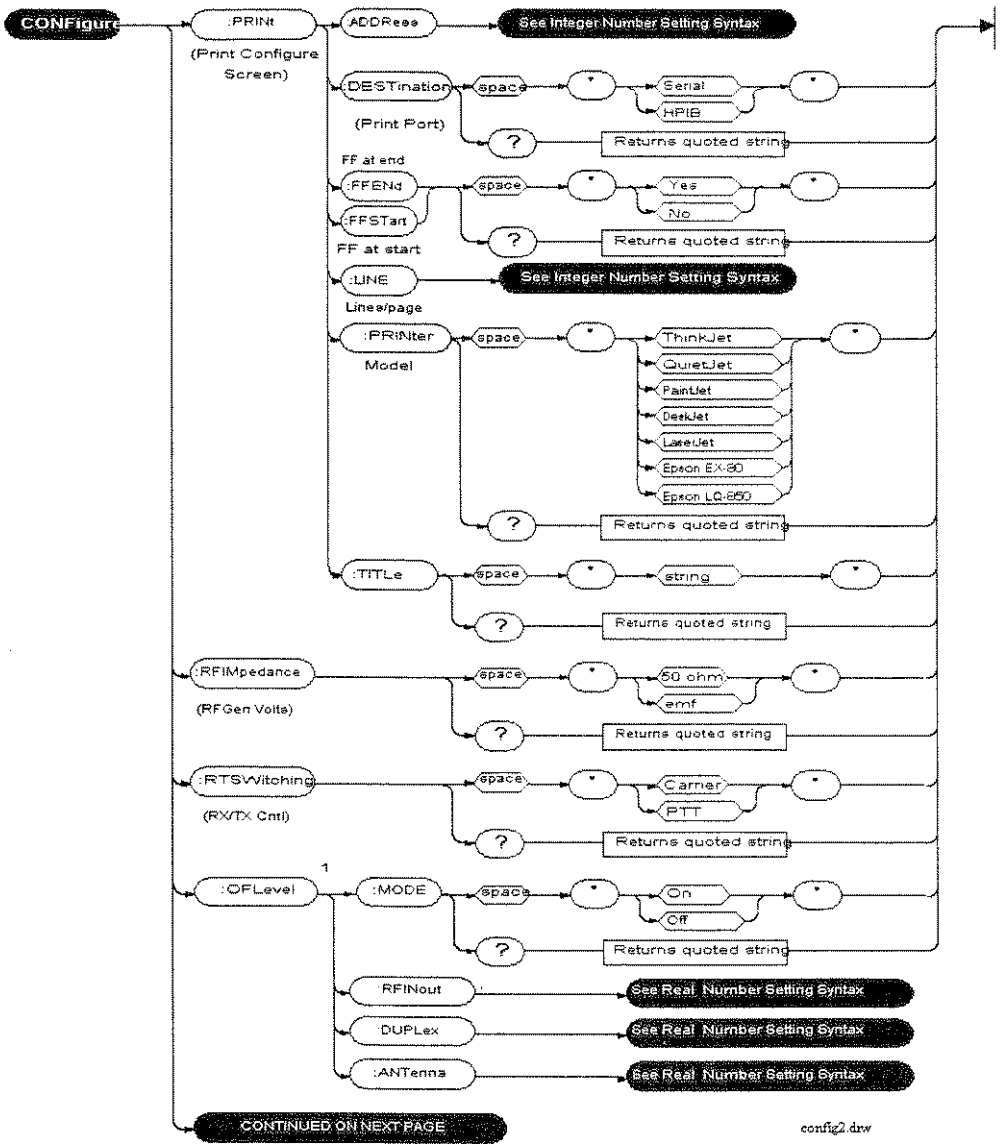

AFGenerator2
ENCoder
MPT1327
:SIDentity
See Integer Number Setting Syntax
:PREFix
:DEFault
:SCAN
:IDENTity
:DEFault
:SCAN
:CHANnel
:CONTrol
:NUMBer
:TRAFfic
:NUMBer
:ALOHa
:QUALifier
:PDELay
:SYNC
:SYNT
:STANdard
space
'
MPT1327
'
?
Returns quoted string
:TMODe
space
'
OFF
CONTROL
TRAFFIC
1200HZ
600HZ
SETTING
'
?
Returns quoted string
:FILLer
:RESET
:UPDATe
:CLEAR
space
integer
:DATA
space
integer
,
string
:MESSage
:CONTrol
:RESET
:CLEAR
space
integer
:DATA
space
integer
,
string
:TRAFfic
:RESET
:CLEAR
space
integer
:DATA
space
integer
,
string

mpt1.drw

Configure, I/O Configure

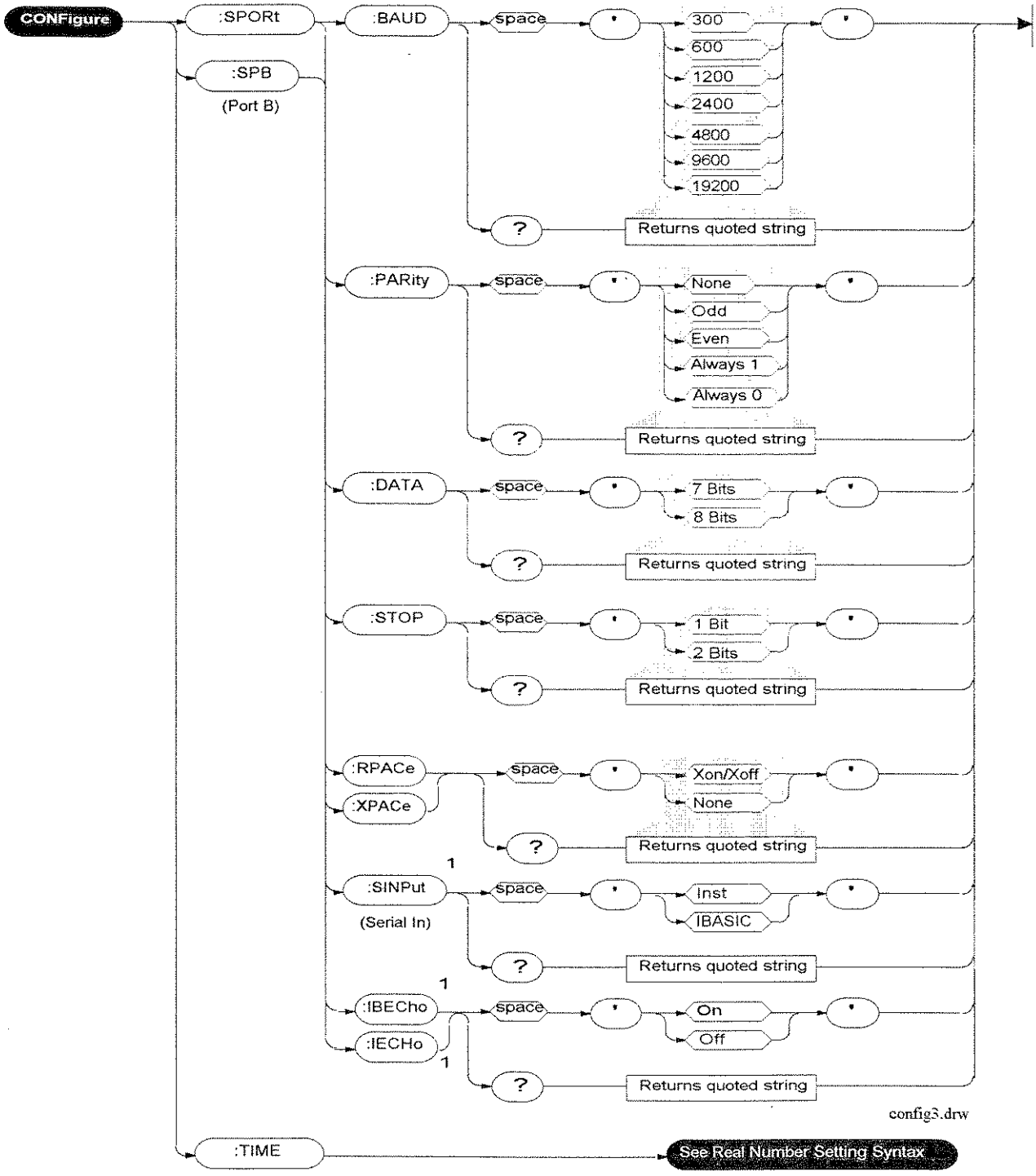


Note: RF Display, RF Chan Std, Base Freq, Chan Space, and (Gen)-(Anl) channel tuning fields are not accessible using HP-IB.



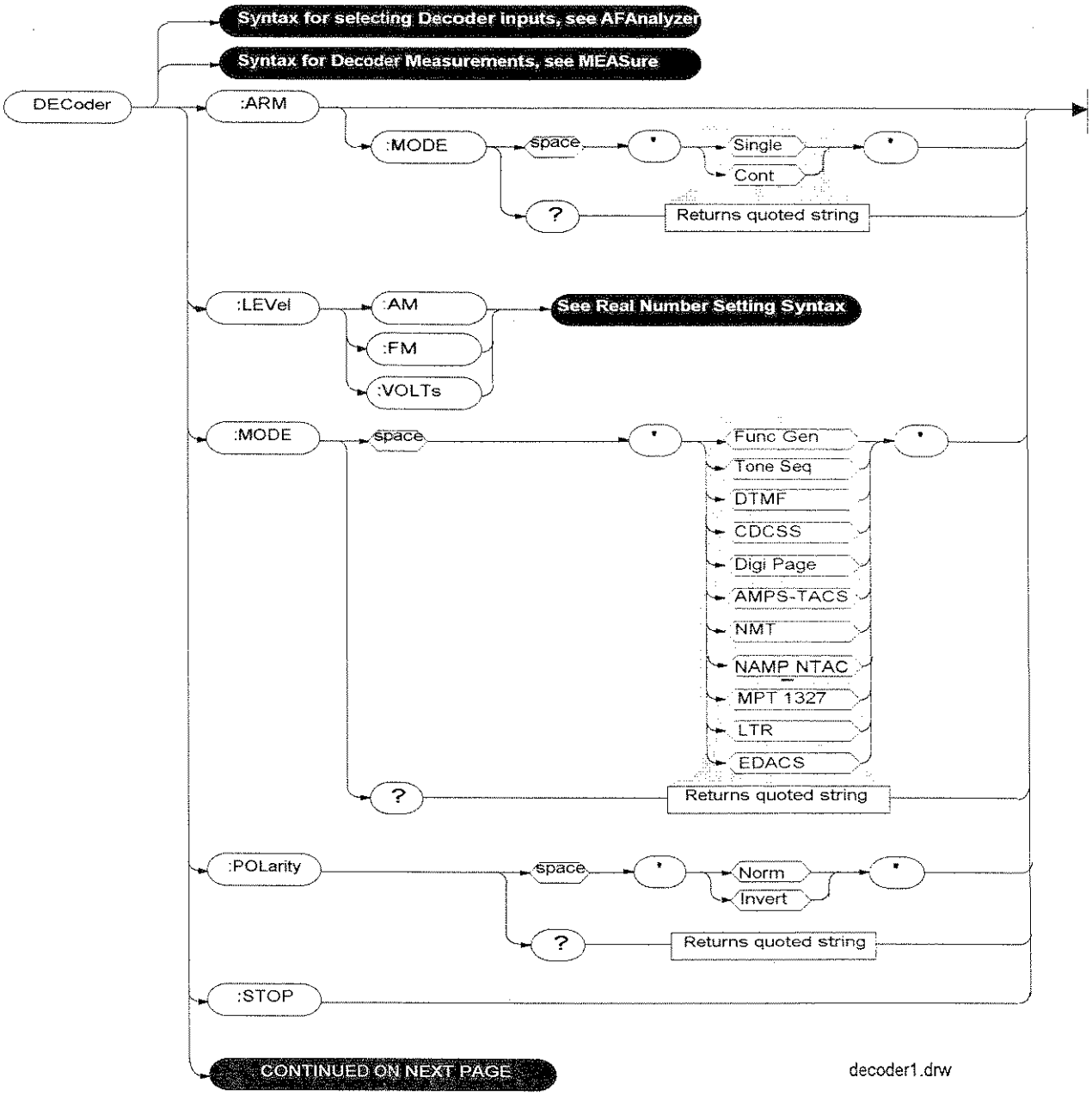
[1] RF Level Offset.

Configure, I/O Configure (Continued)

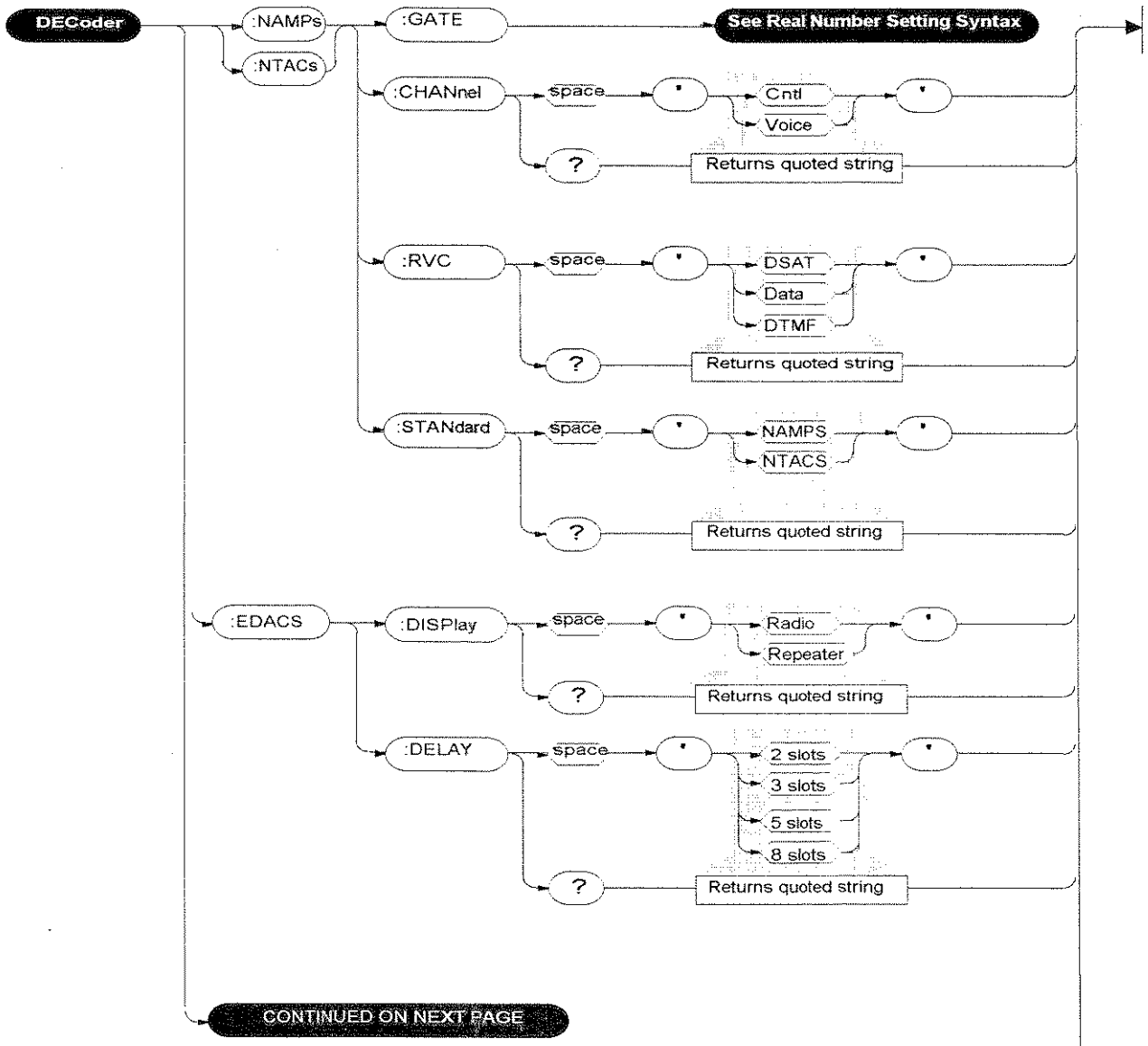


[1] The Serial In, IBASIC Echo, and Inst Echo functions are not used with Port B.

Decoder

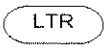

Syntax for selecting Decoder inputs, see AFAnalyzer
Syntax for Decoder Measurements, see MEASure
DECoder
:ARM
:MODE
space
'
Single
Cont
'
?
Returns quoted string
:LEVel
:AM
:FM
:VOLTs
See Real Number Setting Syntax
:MODE
space
'
Func Gen
Tone Seq
DTMF
CDCSS
Digi Page
AMPS-TACS
NMT
NAMP NTAC
MPT 1327
LTR
EDACS
'
?
Returns quoted string
:POLarity
space
'
Norm
Invert
'
?
Returns quoted string
:STOP
CONTINUED ON NEXT PAGE
decoder1.drw

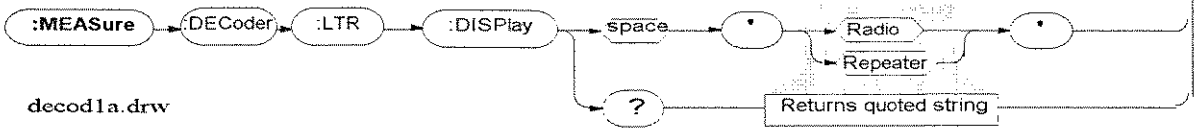
Decoder (continued)



RX TEST(Send)

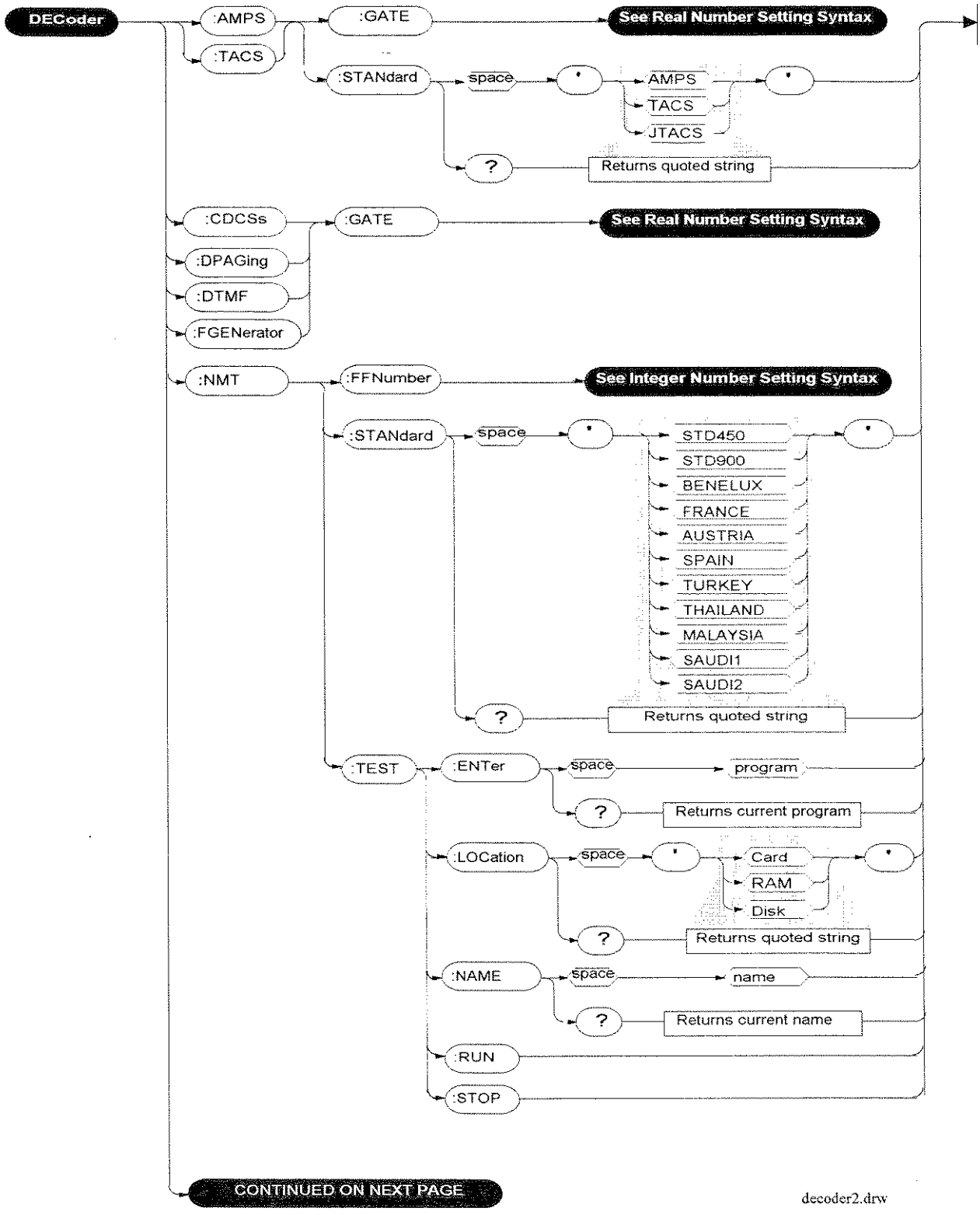
Automatically arms Decoder before sending.





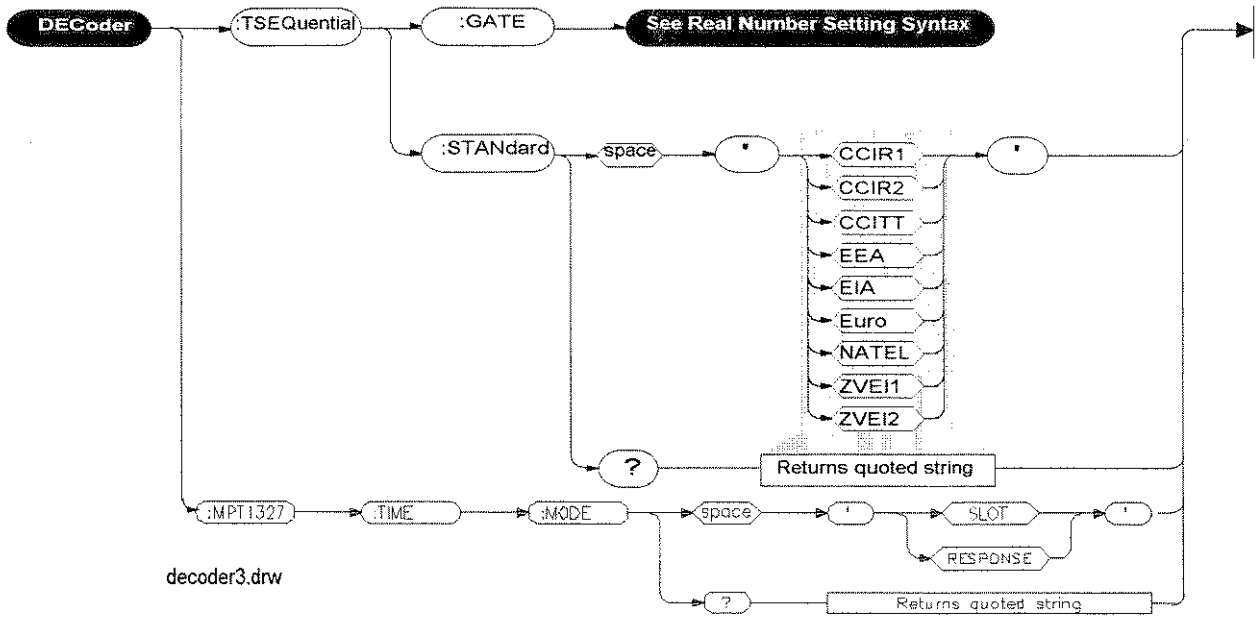
decod1a.drw

Decoder (continued)



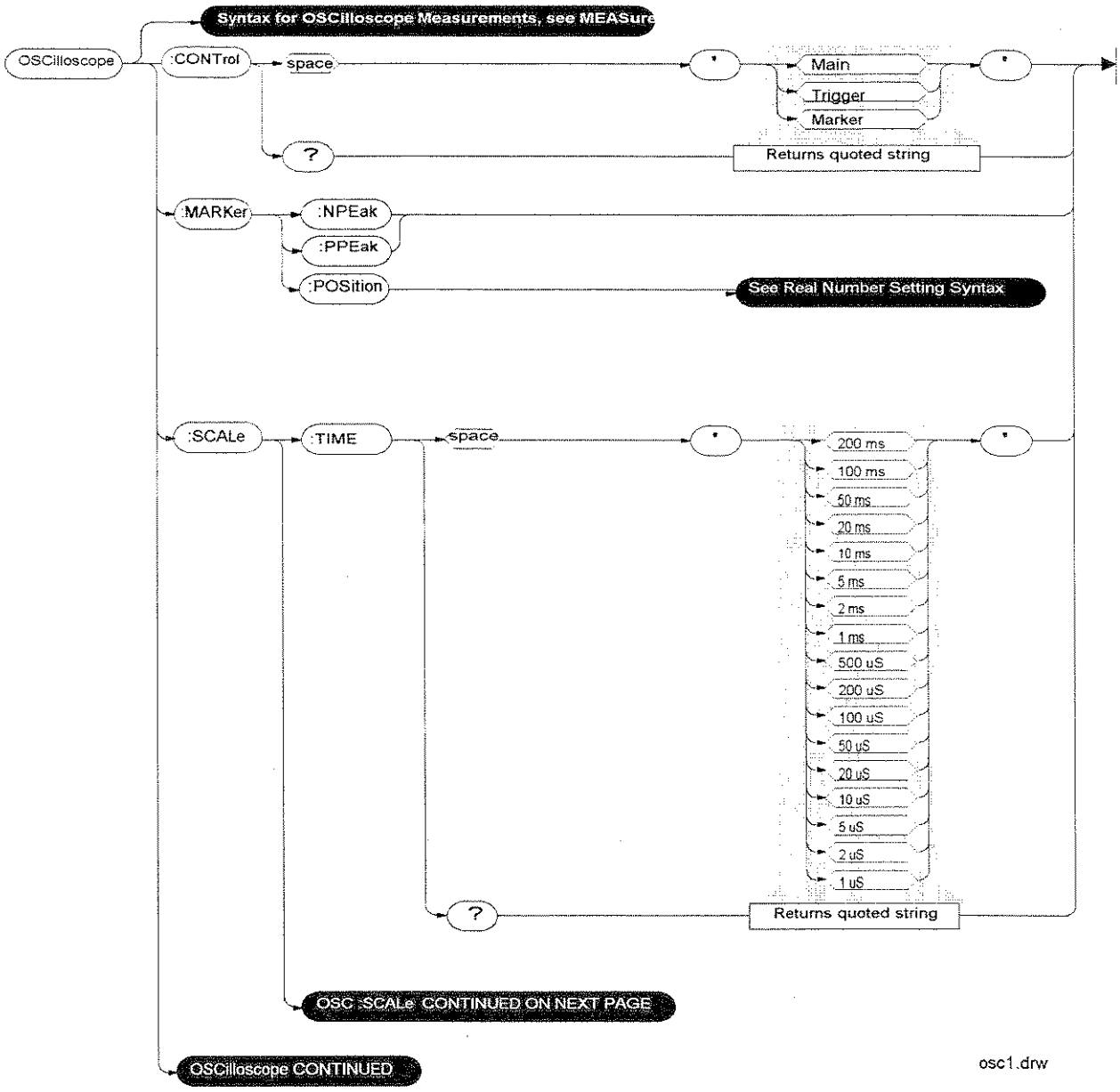
decoder2.drw

Decoder (continued)

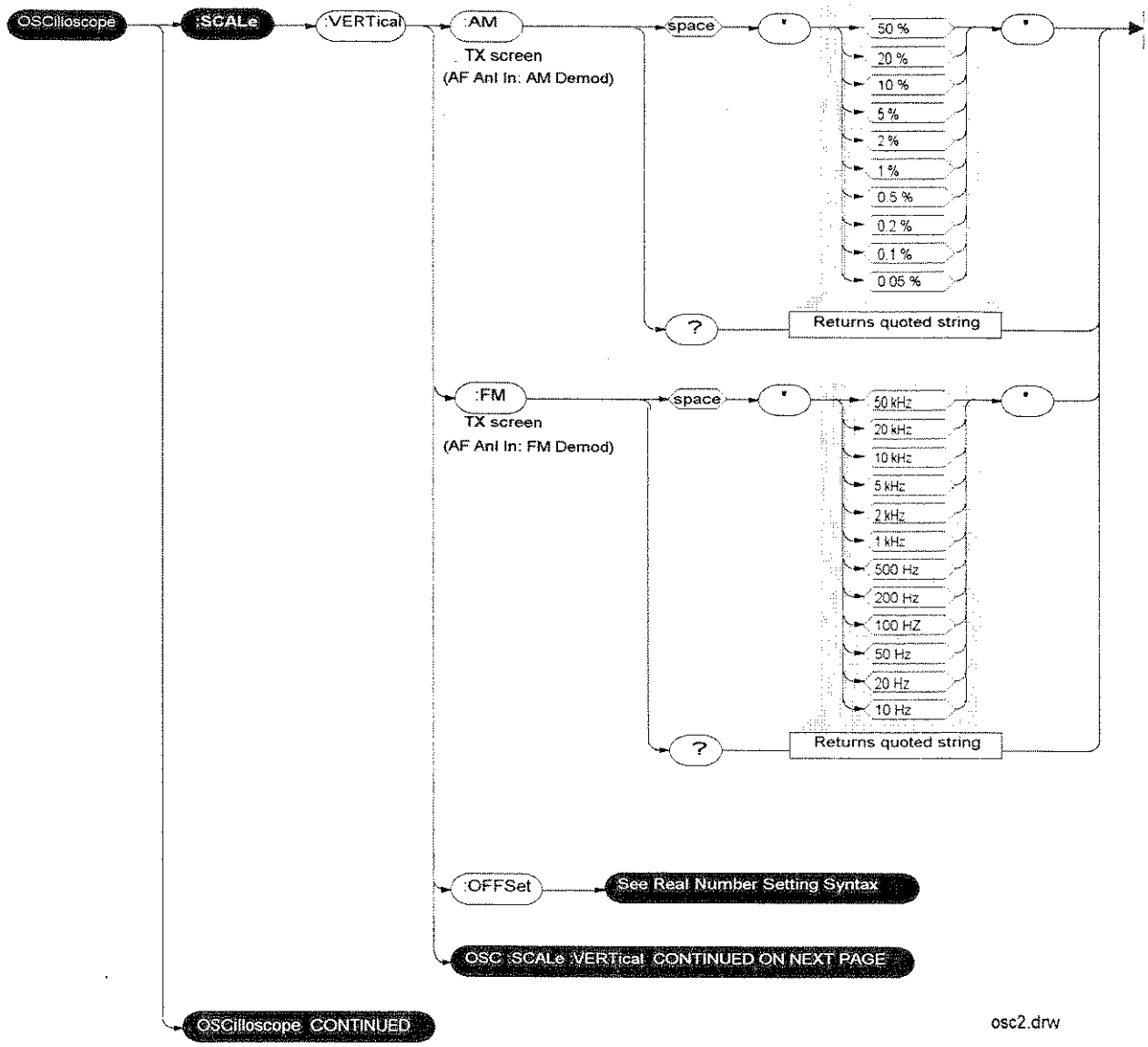


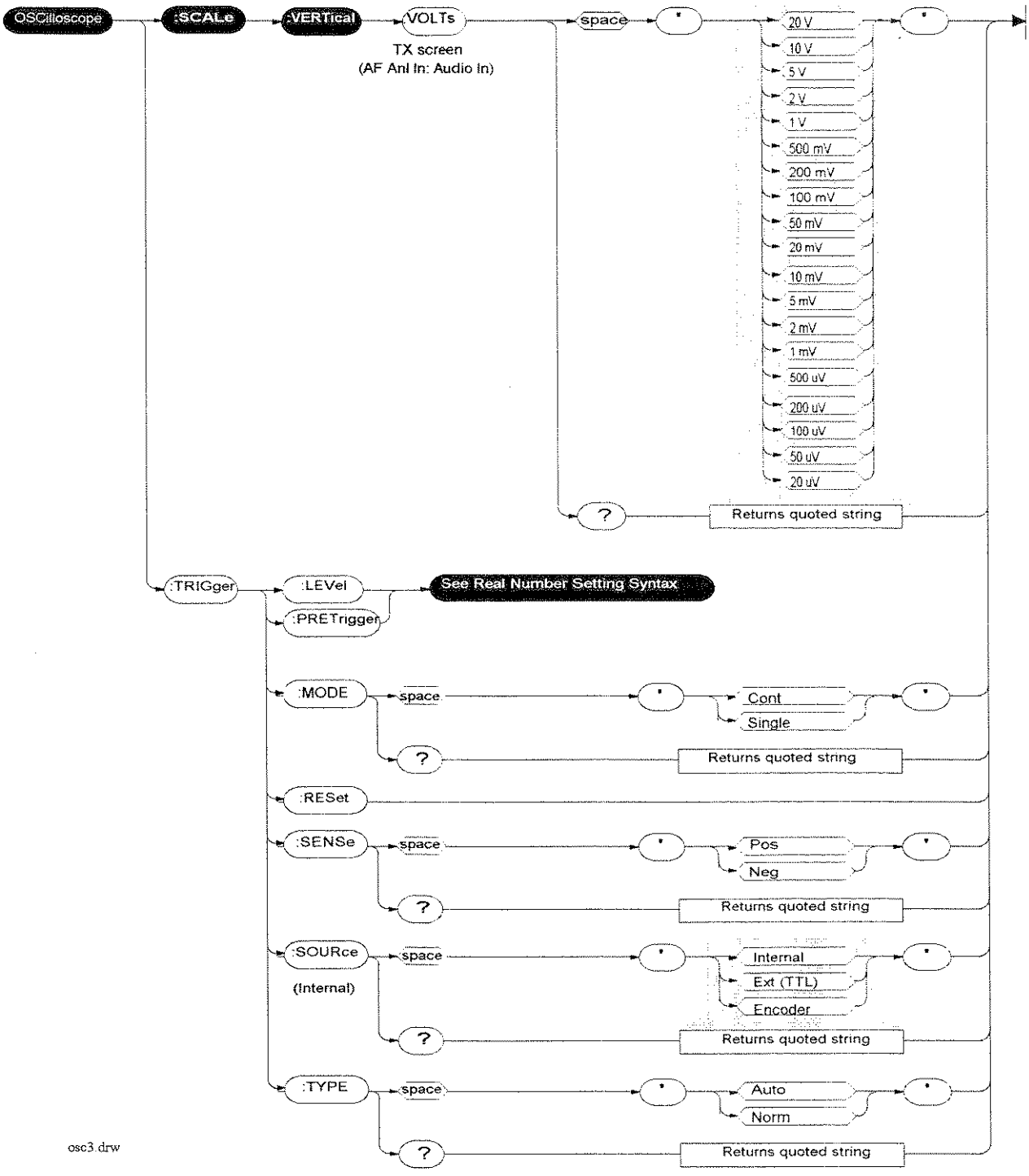
decoder3.drw

Oscilloscope

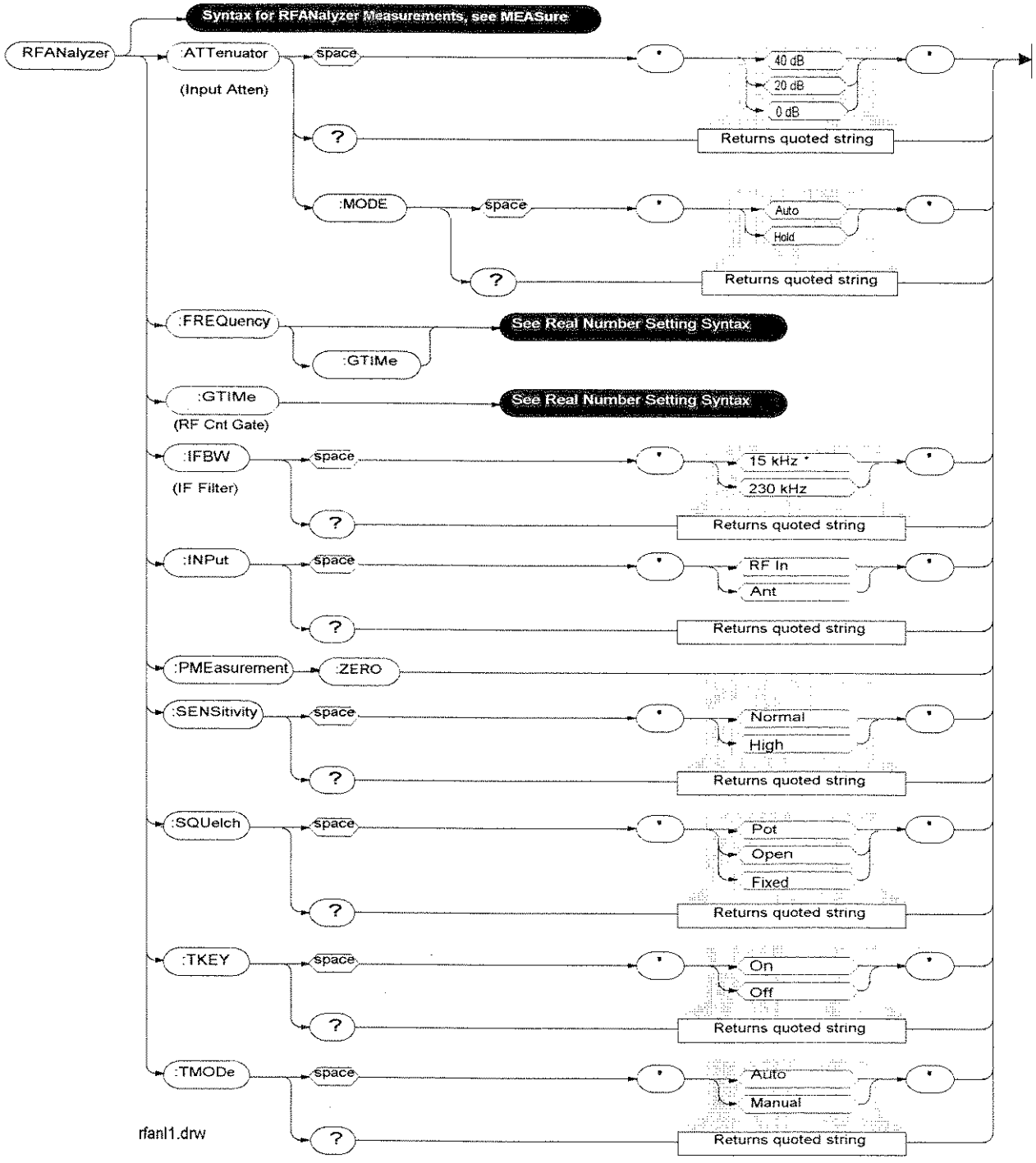


Oscilloscope (continued)



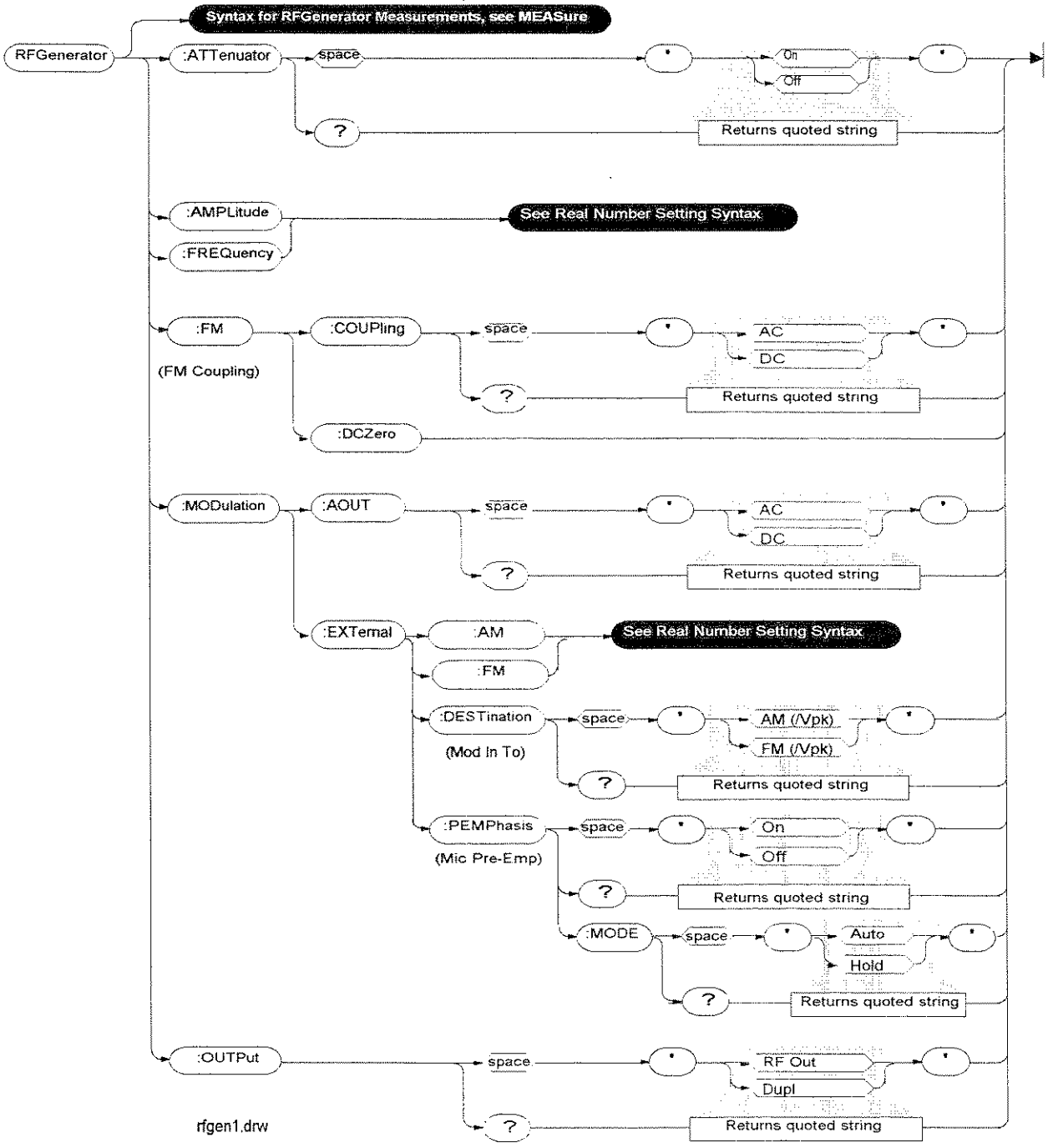

OSCilloscope
:SCALe
:VERTical
VOLTs
TX screen
(AF Ant In: Audio In)
space
'
20 V
10 V
5 V
2 V
1 V
500 mV
200 mV
100 mV
50 mV
20 mV
10 mV
5 mV
2 mV
1 mV
500 uV
200 uV
100 uV
50 uV
20 uV
?
Returns quoted string
:TRIGger
:LEVel
:PRETrigger
See Real Number Setting Syntax
:MODE
Cont
Single
:RESet
:SENSe
Pos
Neg
:SOURce
(Internal)
Internal
Ext (TTL)
Encoder
:TYPE
Auto
Norm
osc3.drw

RF Analyzer

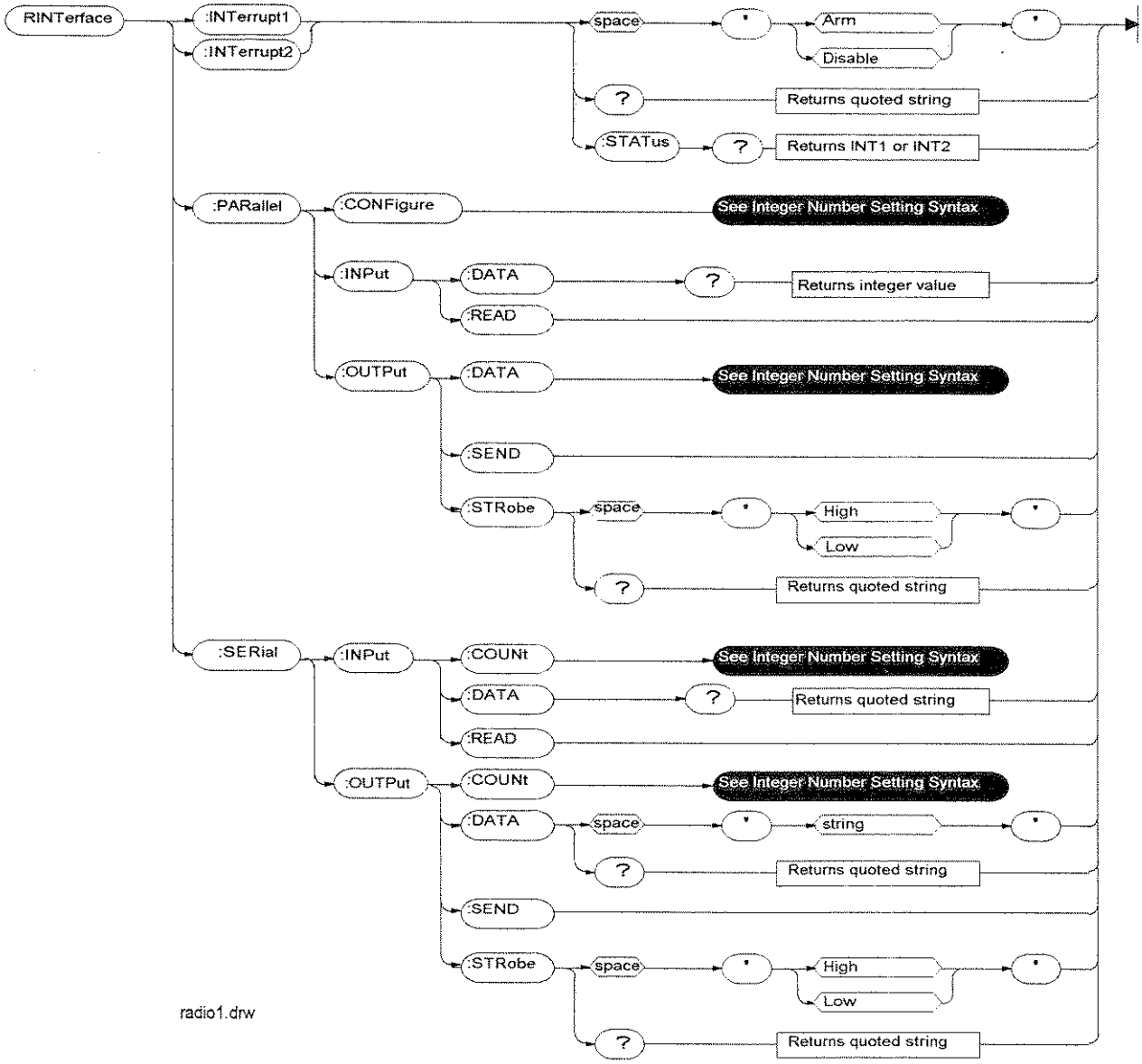


*The HP 8921A uses a 30 kHz filter in place of the HP 8920A's 15 kHz setting.

RF Generator

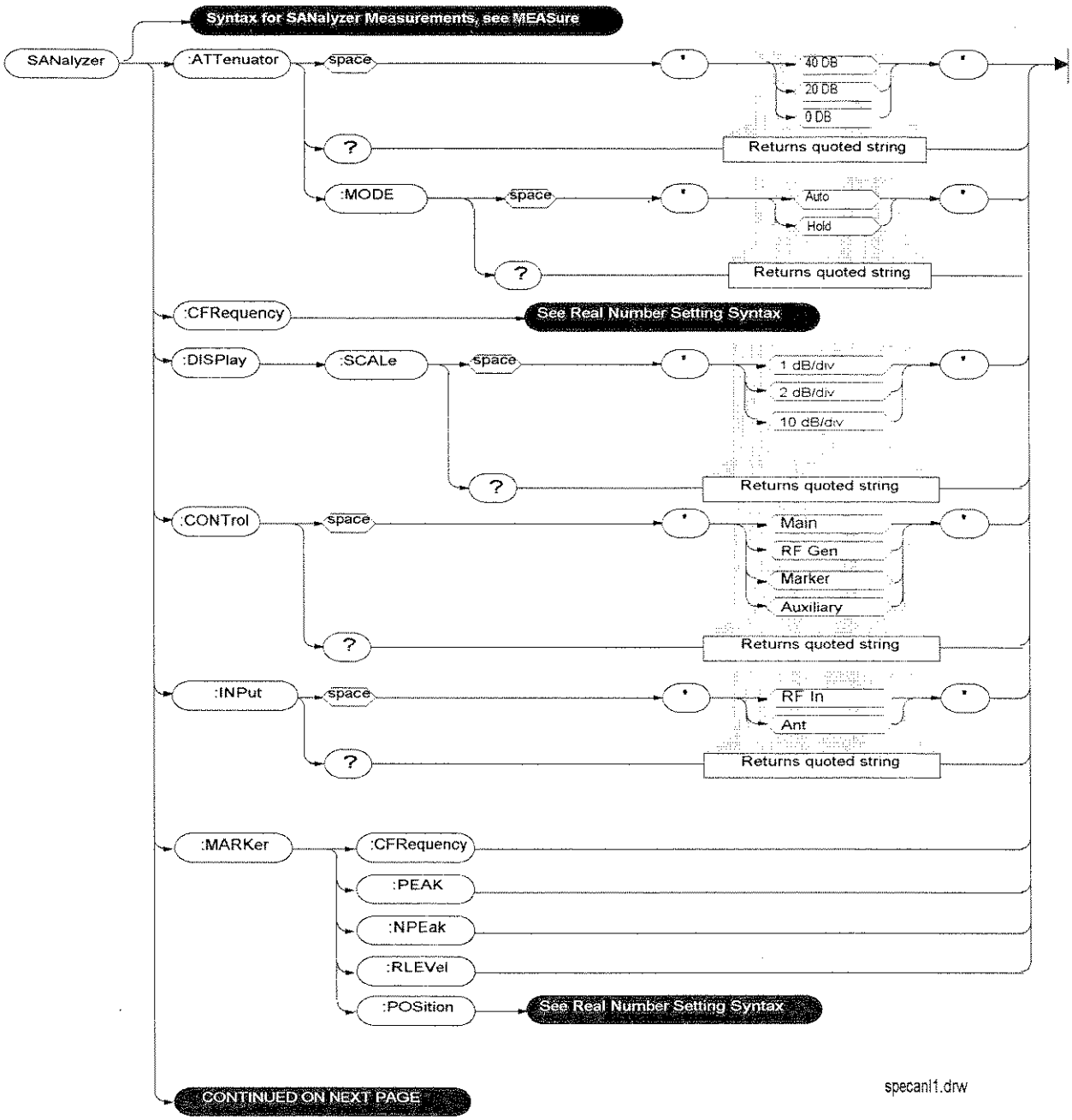


Radio Interface

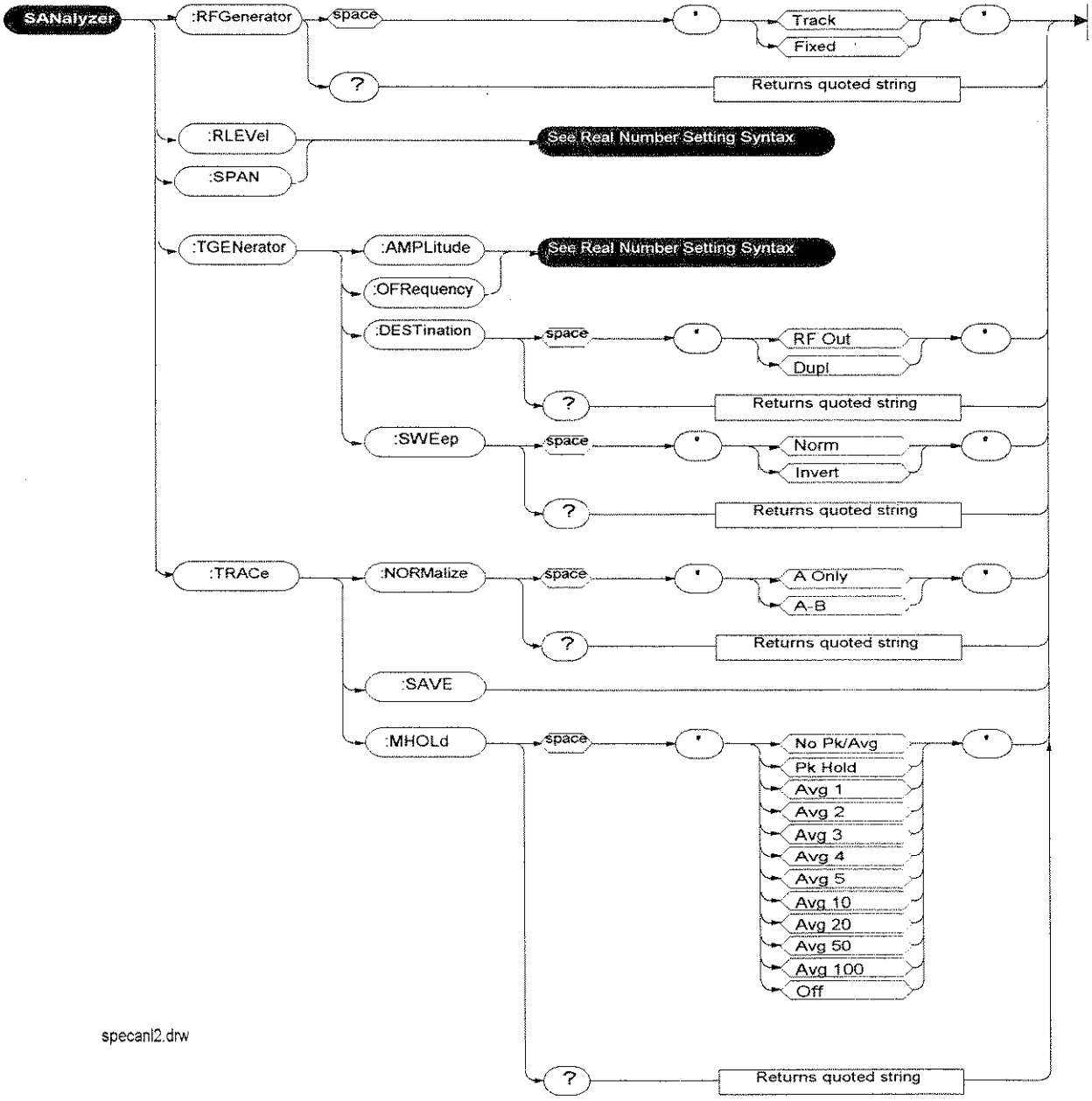


radio1.drw

Spectrum Analyzer

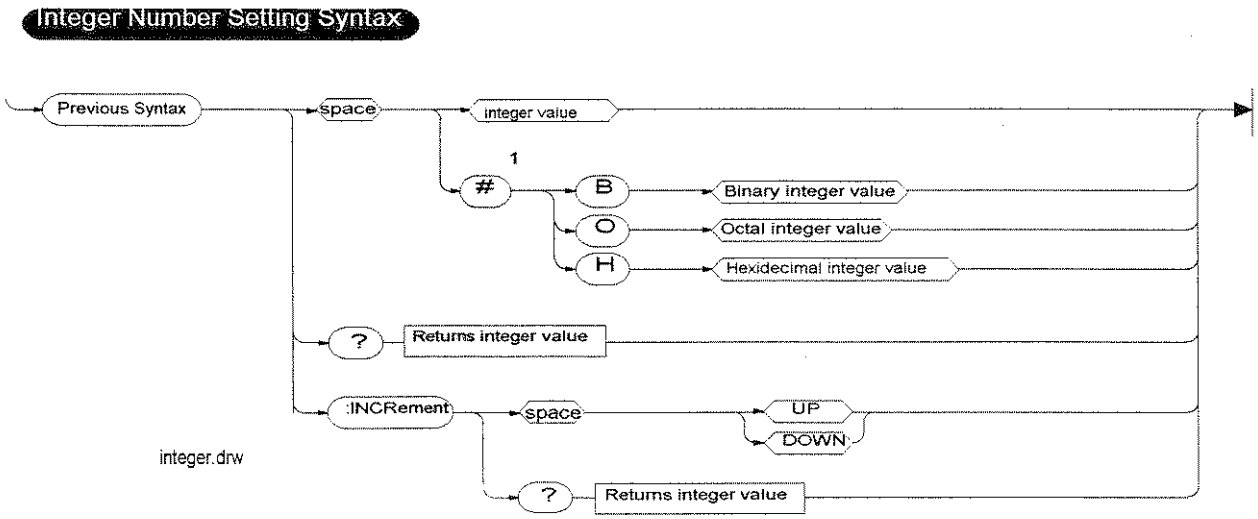


Spectrum Analyzer (continued)



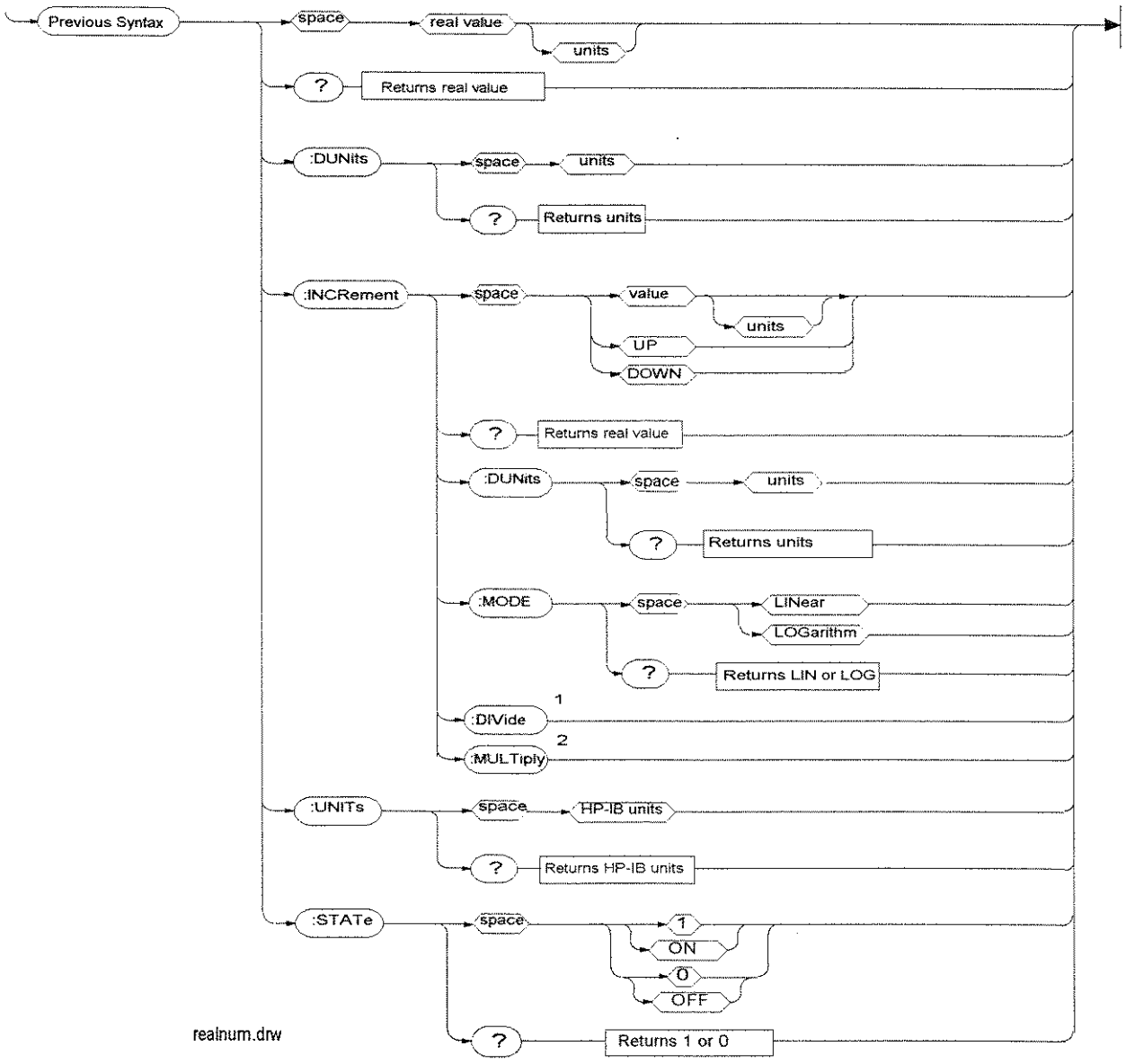
specanl2.drw

Integer Number Setting Syntax



Real Number Setting Syntax

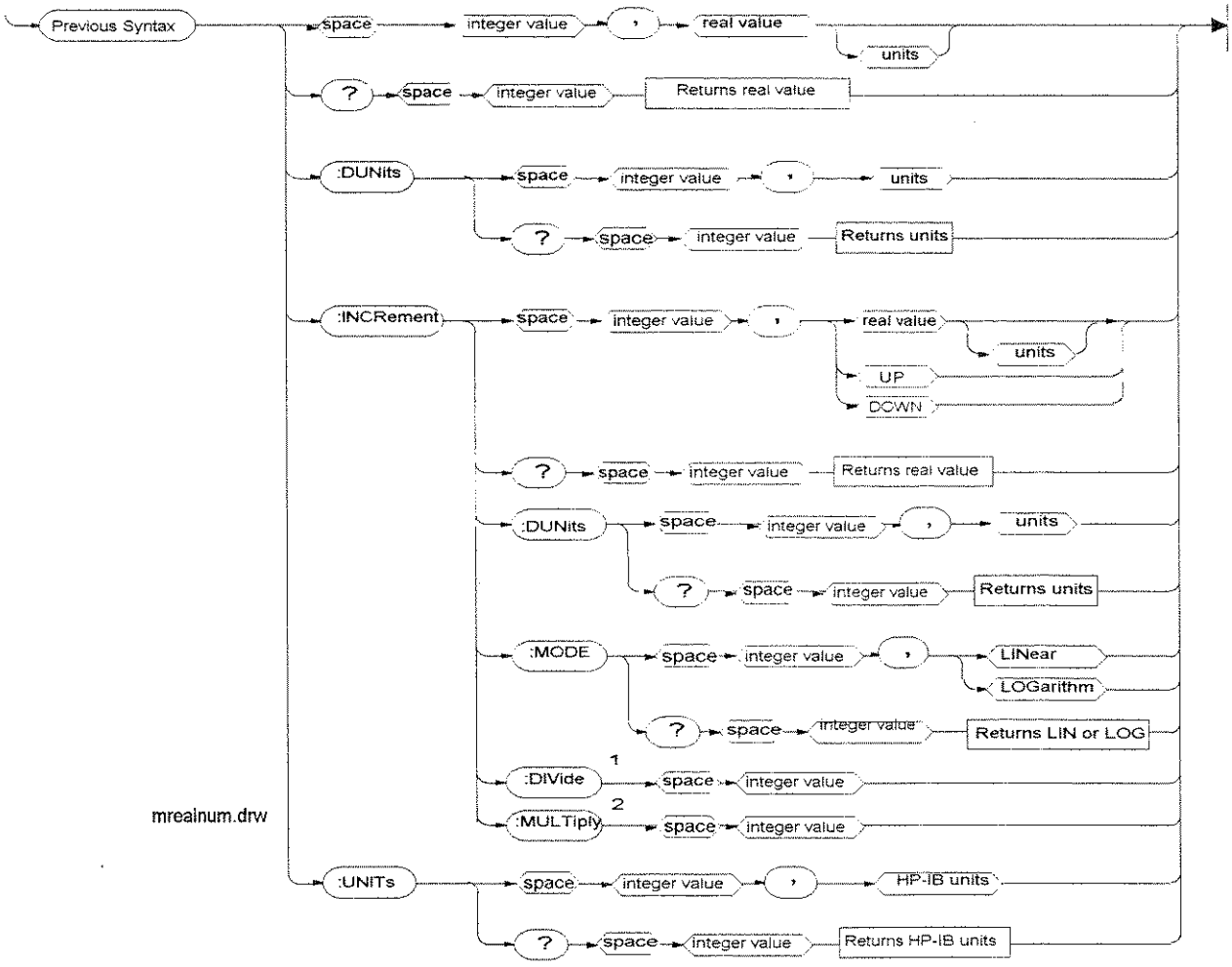
Real Number Setting Syntax



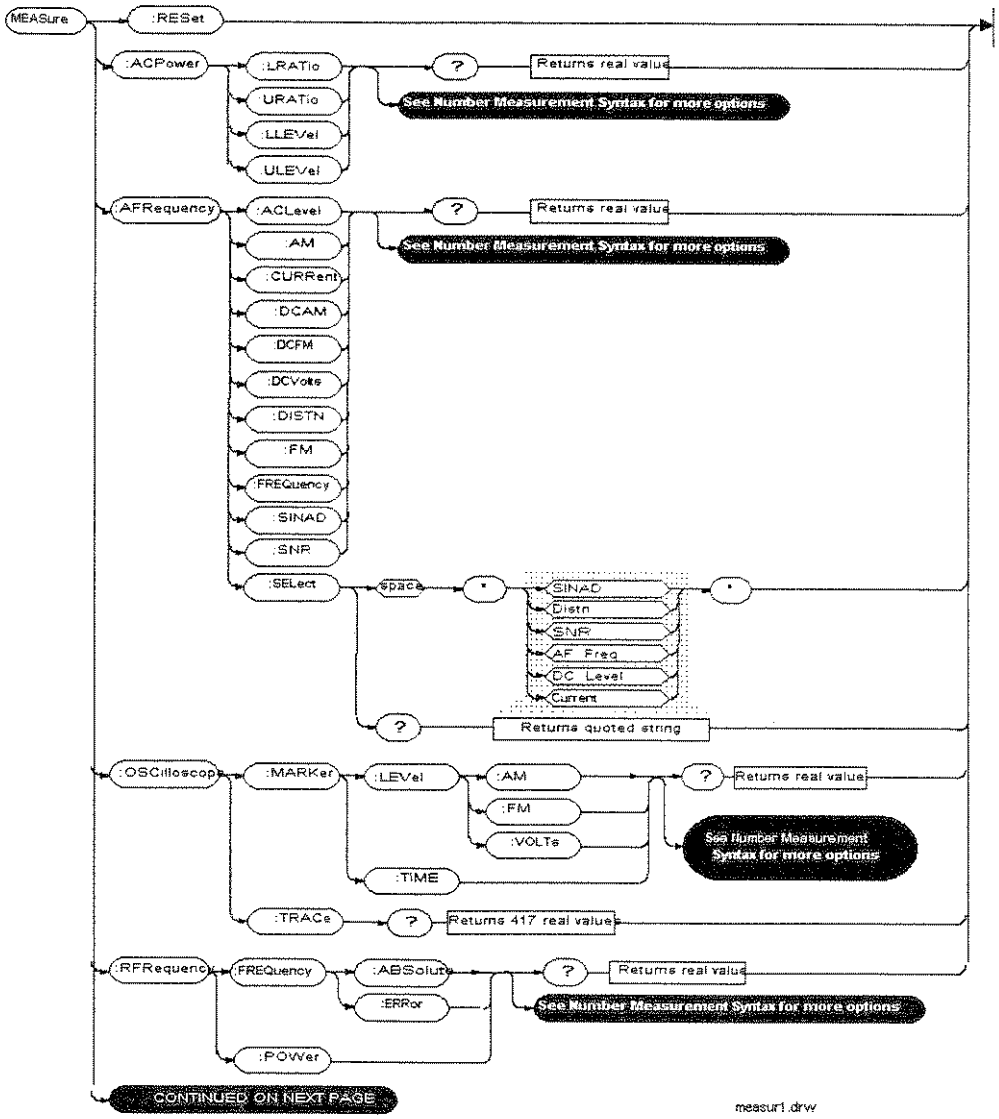
realnum.drw

Multiple Real Number Setting Syntax

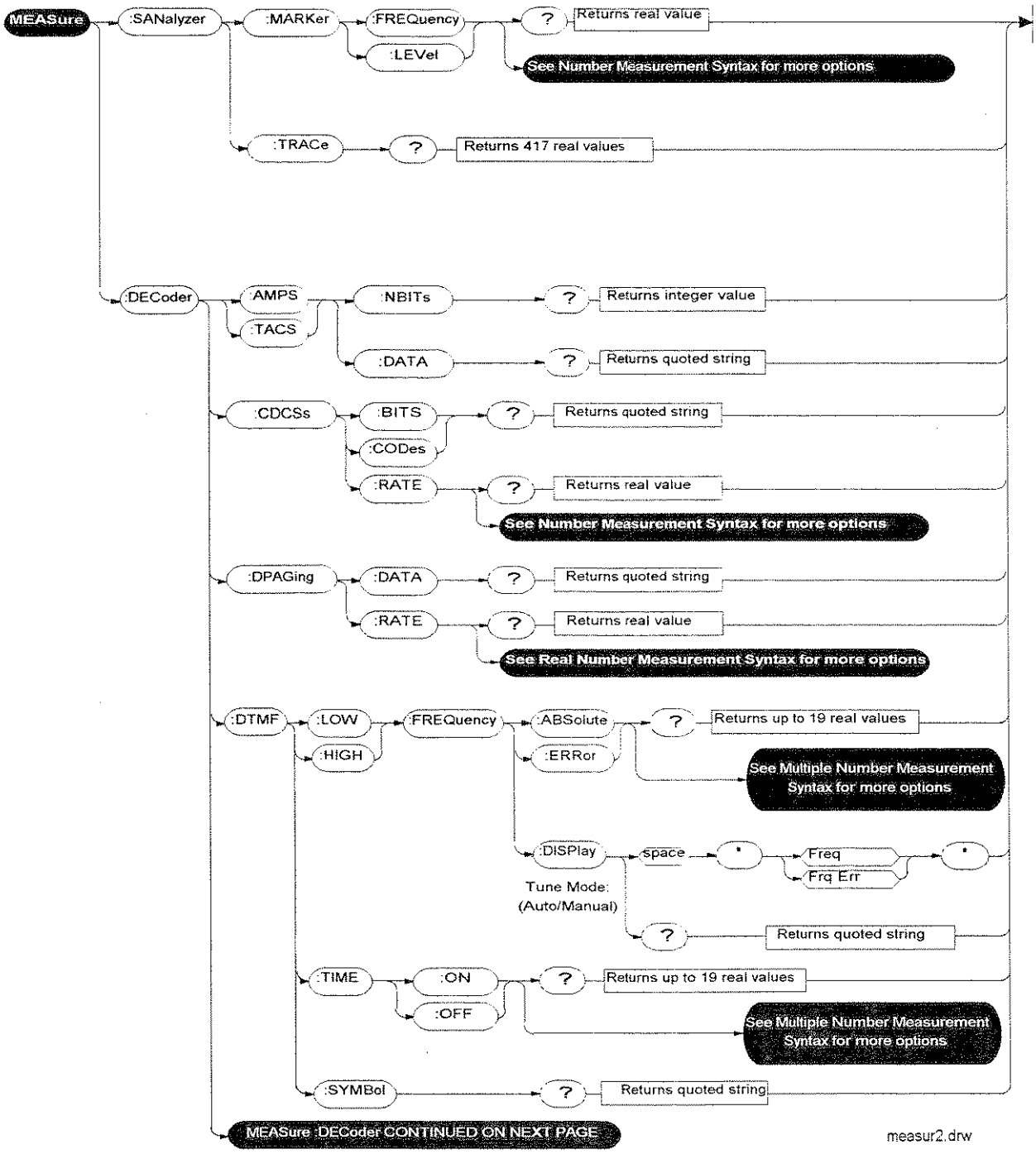
Multiple Real Number Setting Syntax

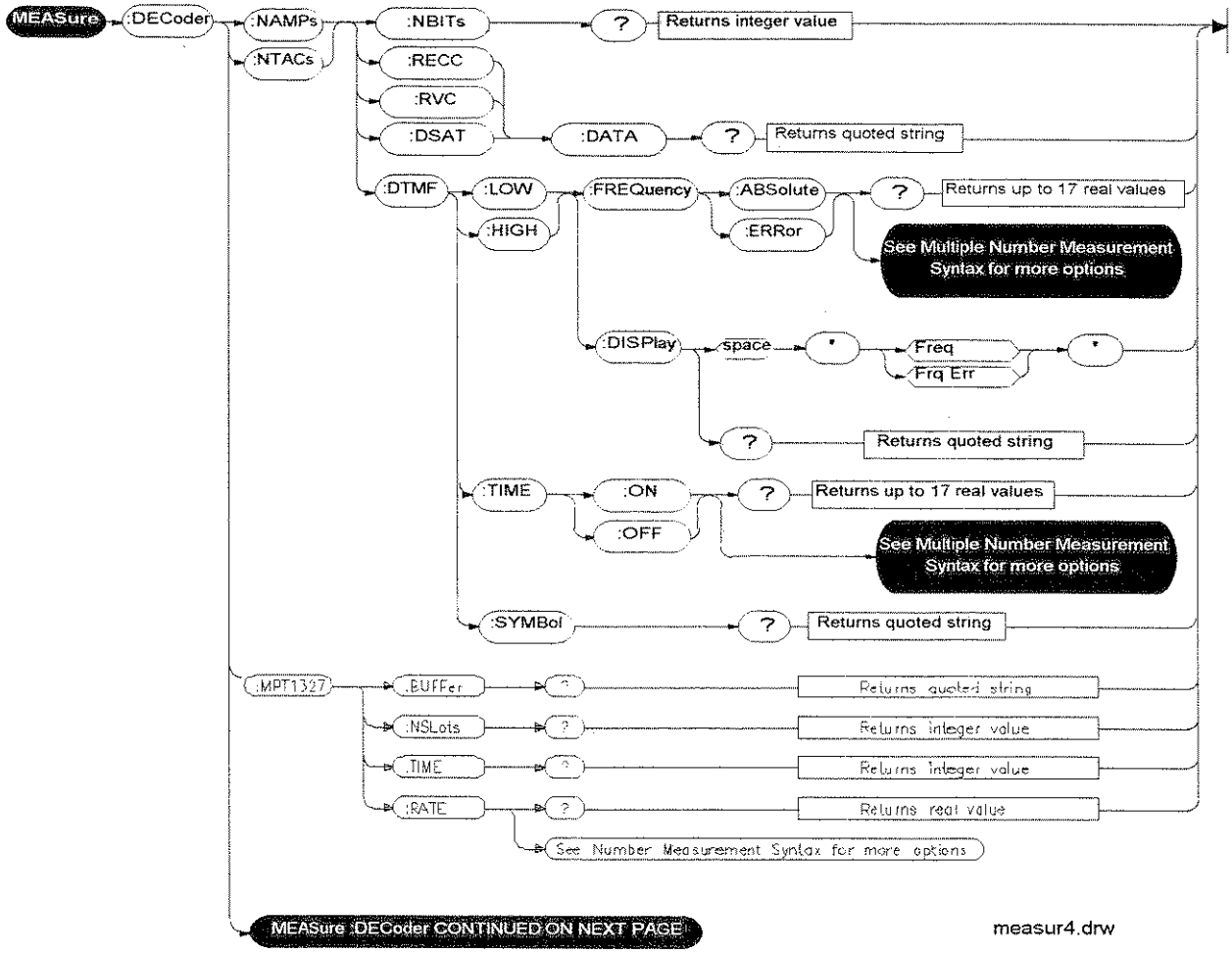


Measure

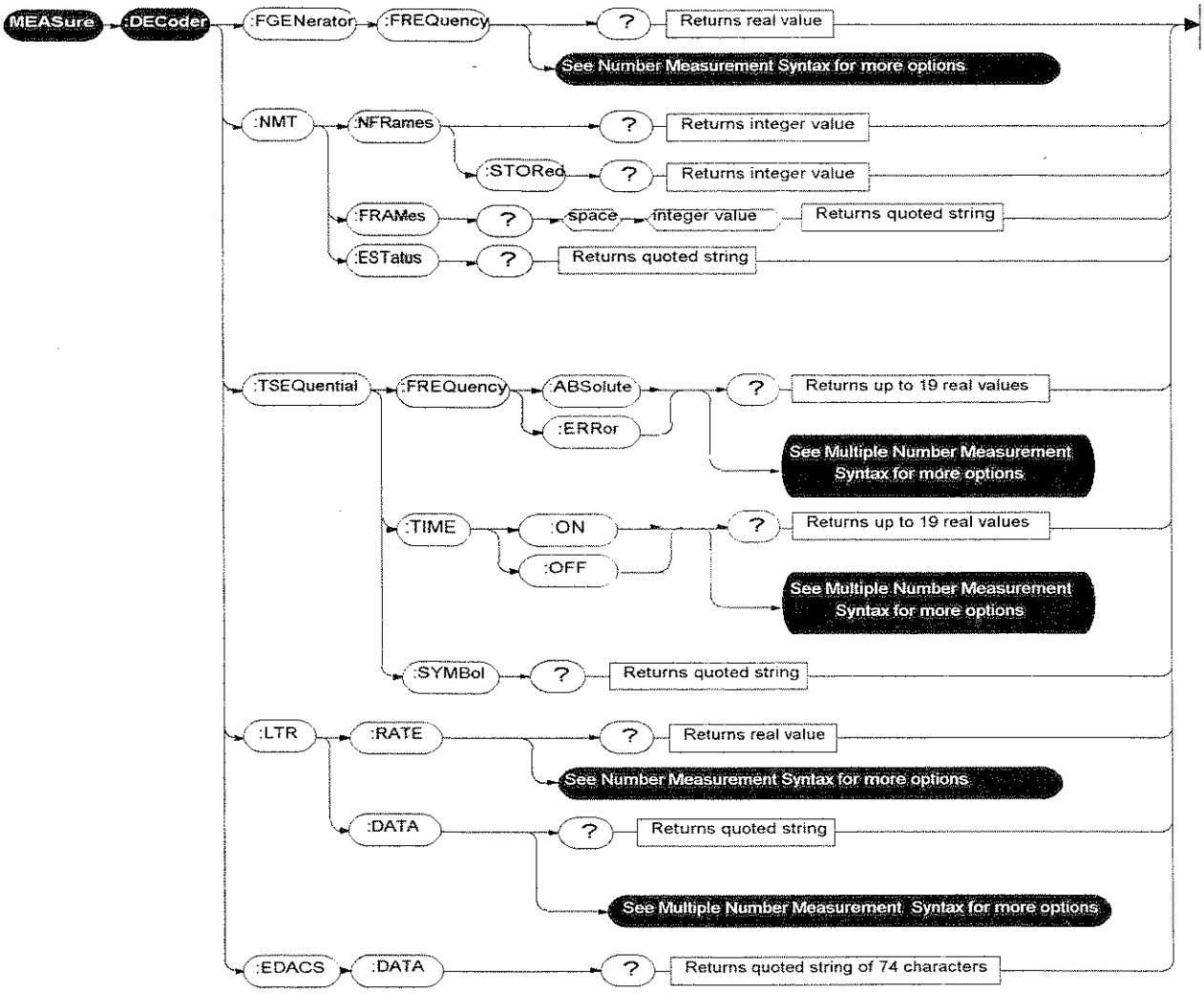

MEASure
:RESet
:ACPower
:LRATio
:URATio
:LLEVel
:ULEVel
?
Returns real value
See Number Measurement Syntax for more options
:AFRequency
:ACLevel
:AM
:CURRent
:DCAM
:DCFM
:DCVolts
:DISTN
:FM
:FREQuency
:SINAD
:SNR
:SELect
space
'
SINAD
Distn
SNR
AF Freq
DC Level
Current
Returns quoted string
:OSCilloscope
:MARKer
:LEVel
:TIME
:TRACe
Returns 417 real value
:RFRequency
:FREQuency
:ABSolute
:ERRor
:POWer
CONTINUED ON NEXT PAGE
measur1.drw

Measure (continued)



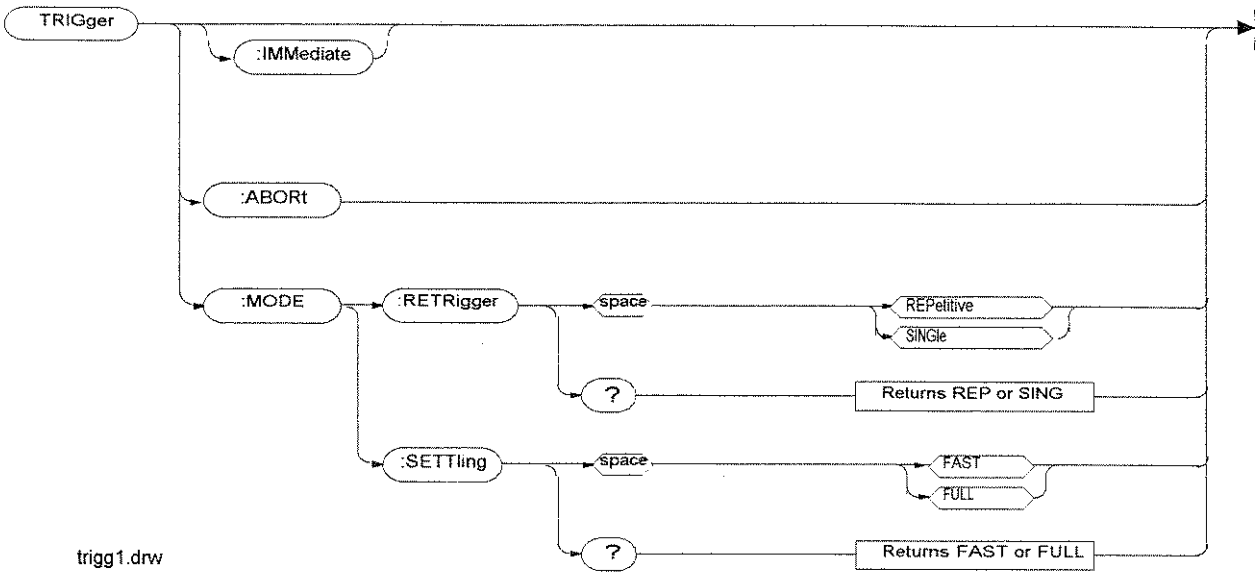

MEASure
:DECoder
:NAMPs
:NTACs
:NBITs
?
Returns integer value
:RECC
:RVC
:DSAT
:DATA
?
Returns quoted string
:DTMF
:LOW
:HIGH
:FREQuency
:ABSolute
:ERRor
?
Returns up to 17 real values
See Multiple Number Measurement Syntax for more options
:DISPlay
space
'
Freq
Frq Err
'
?
Returns quoted string
:TIME
:ON
:OFF
?
Returns up to 17 real values
See Multiple Number Measurement Syntax for more options
:SYMBol
?
Returns quoted string
:MPT1327
:BUFFer
Returns quoted string
:NSLots
Returns integer value
:TIME
Returns integer value
:RATE
Returns real value
See Number Measurement Syntax for more options
MEASure :DECoder CONTINUED ON NEXT PAGE
measur4.drw

Measure (continued)



measur3.drw

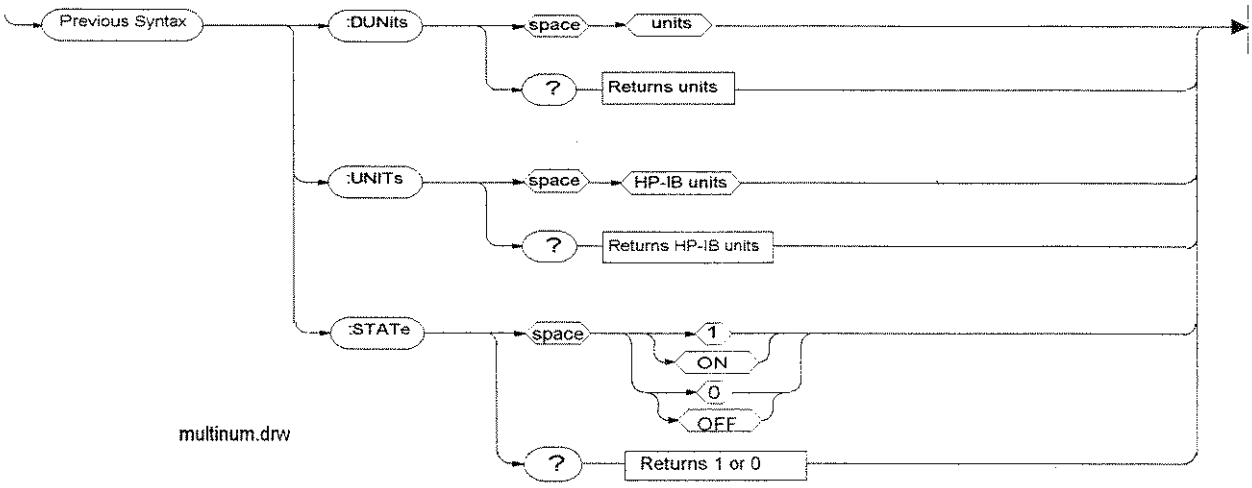
Trigger



trigg1.drw

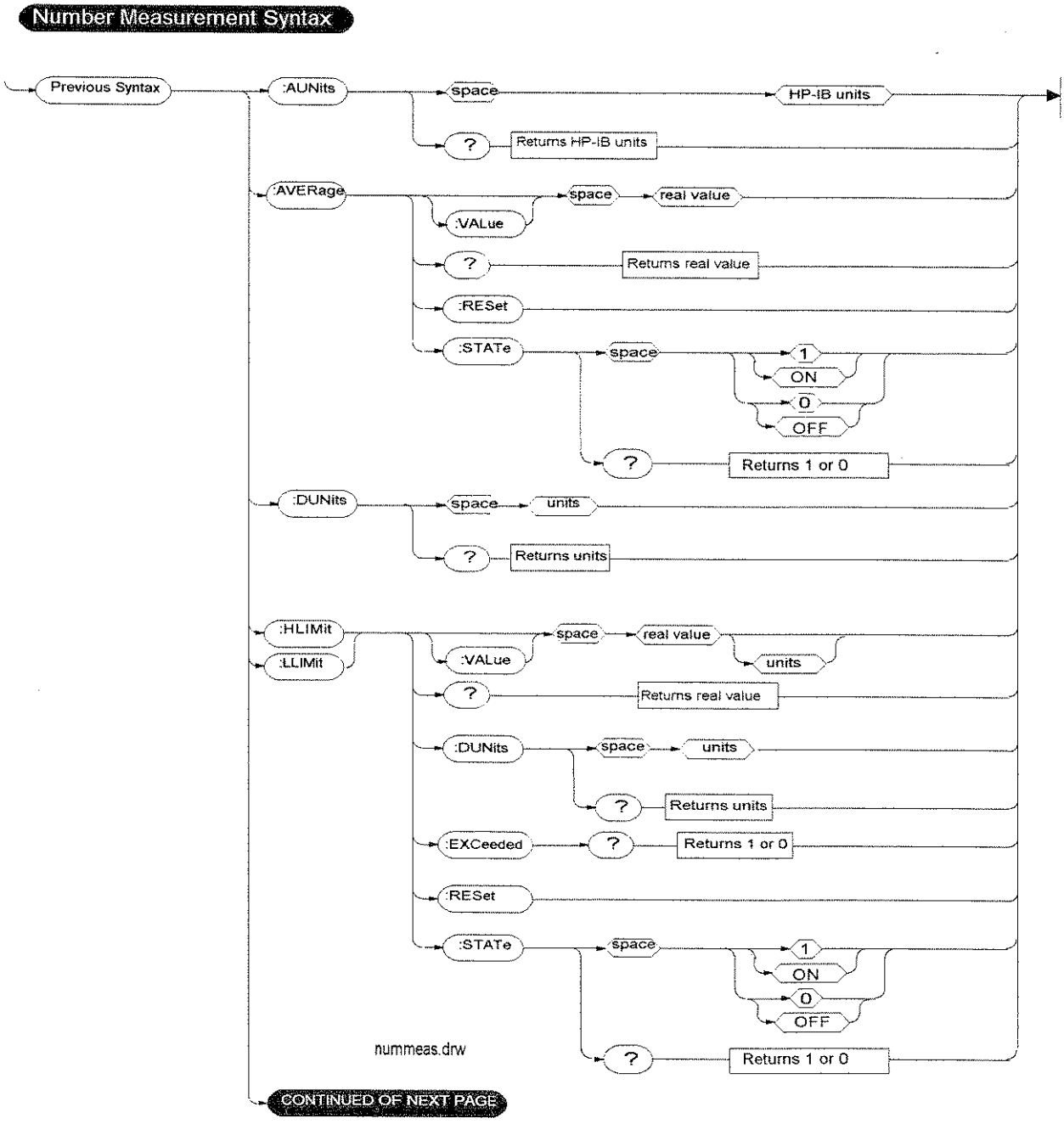
Multiple Number Measurement Syntax

Multiple Number Measurement Syntax

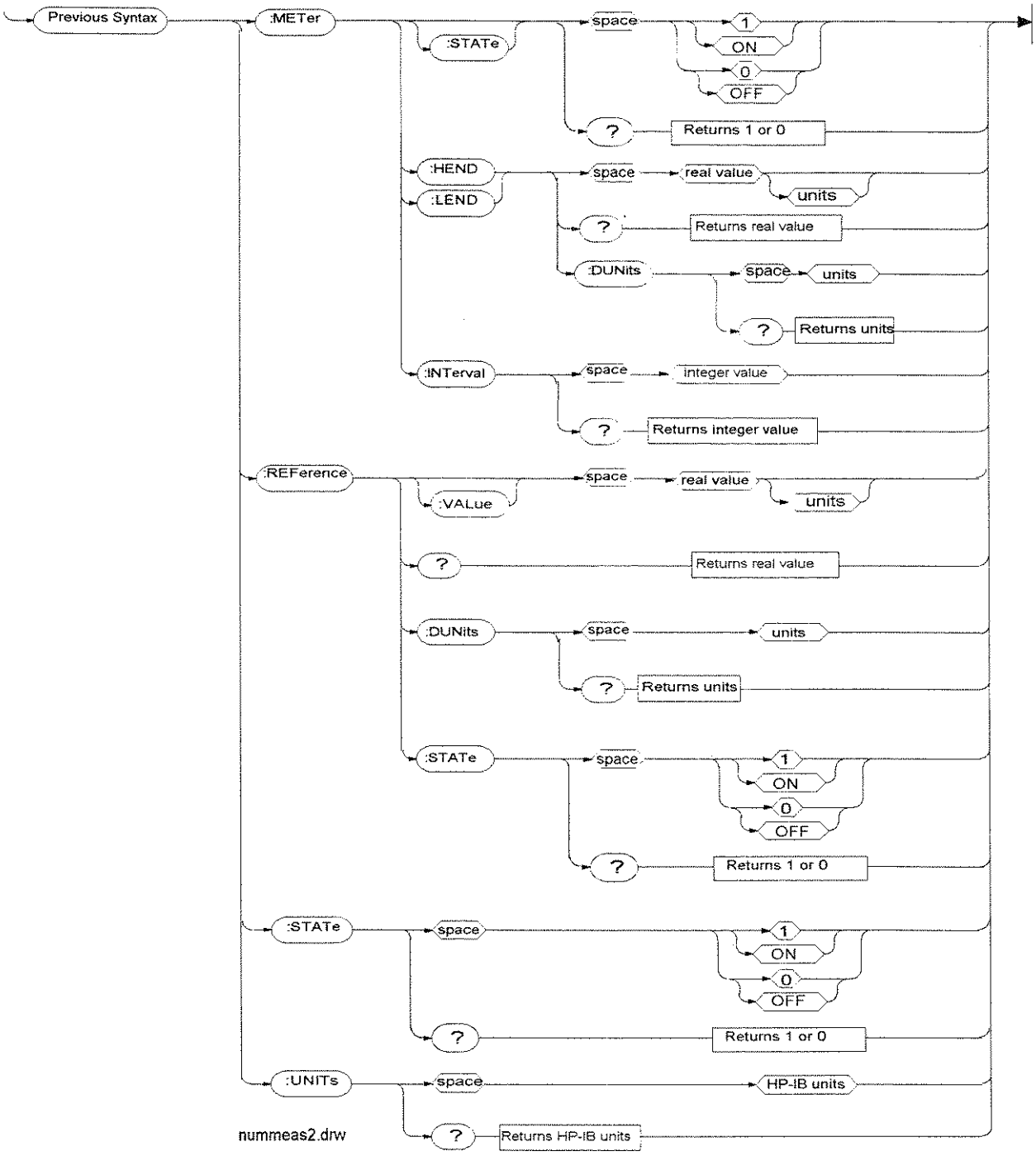


multinum.drw

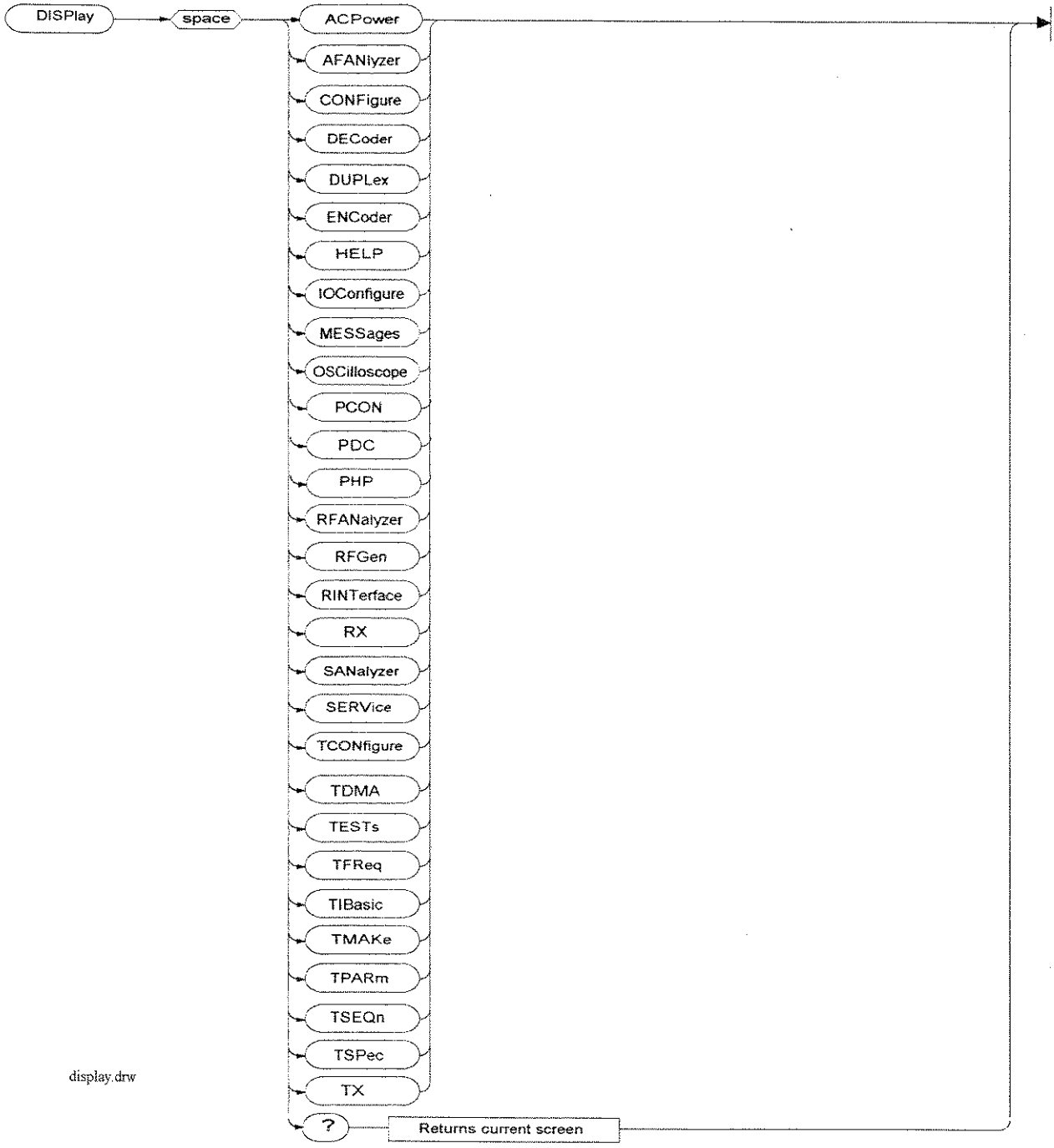
Number Measurement Syntax



Number Measurement Syntax

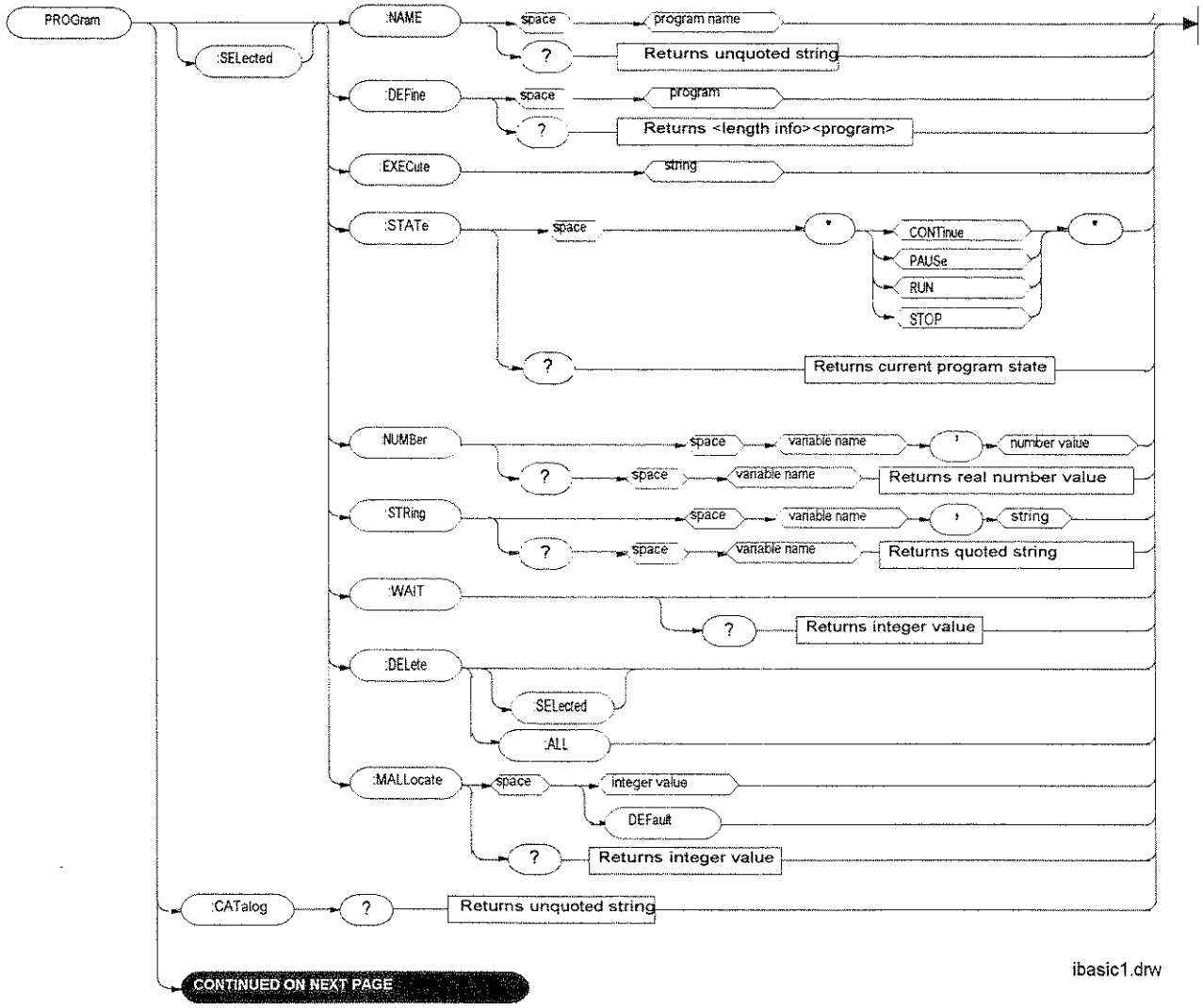


Display



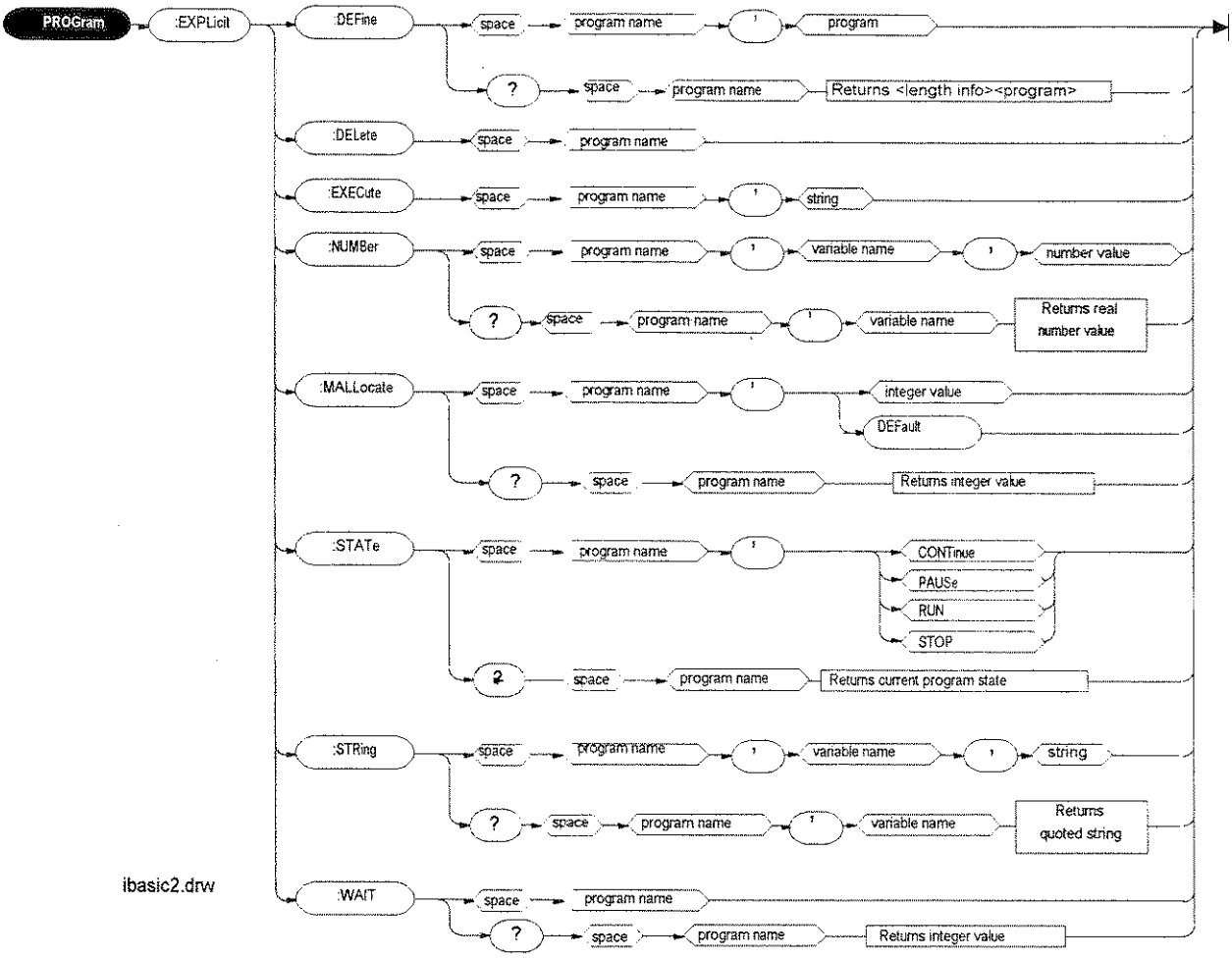
display.drw

Program



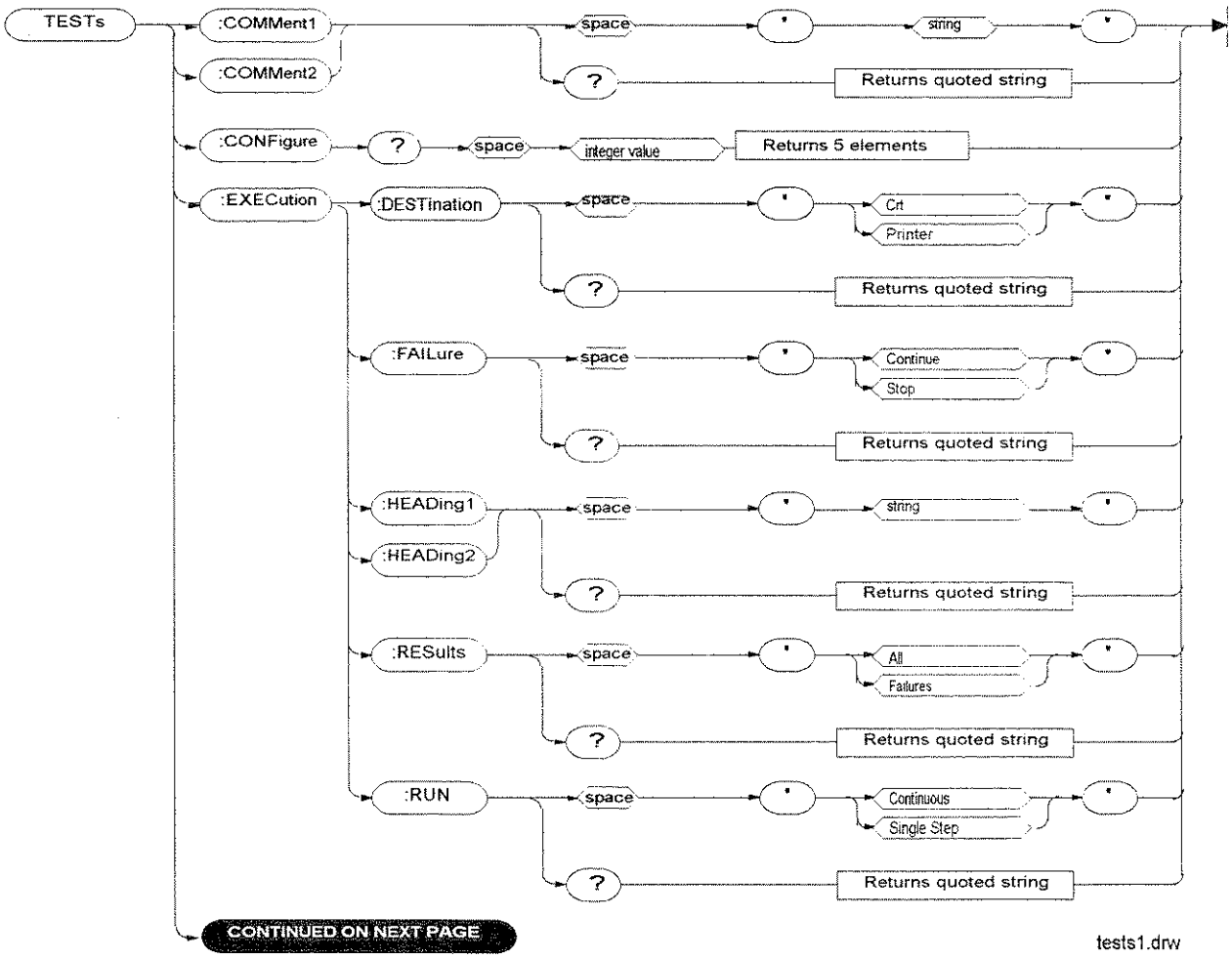
ibasic1.drw

Program (continued)

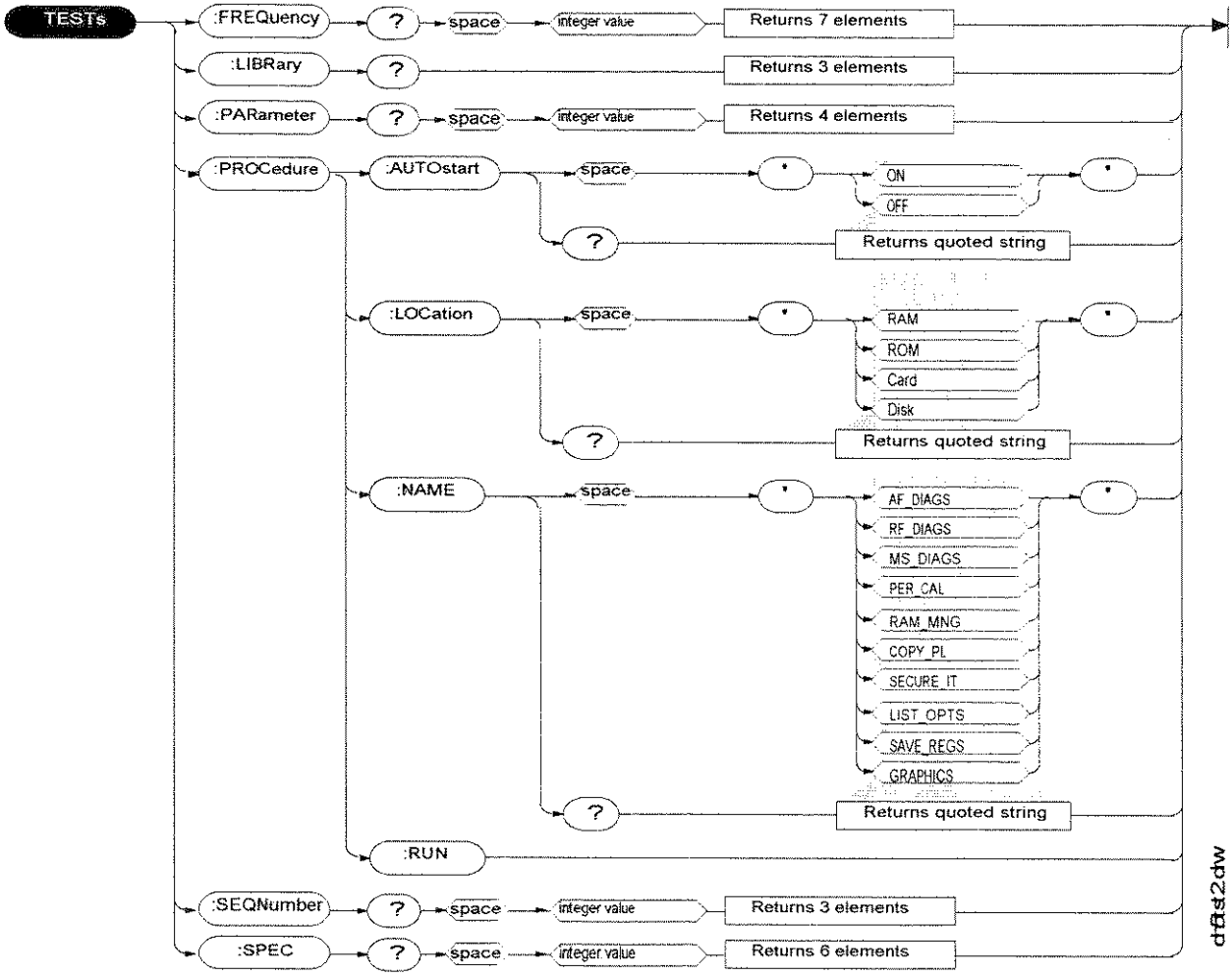


ibasic2.drw

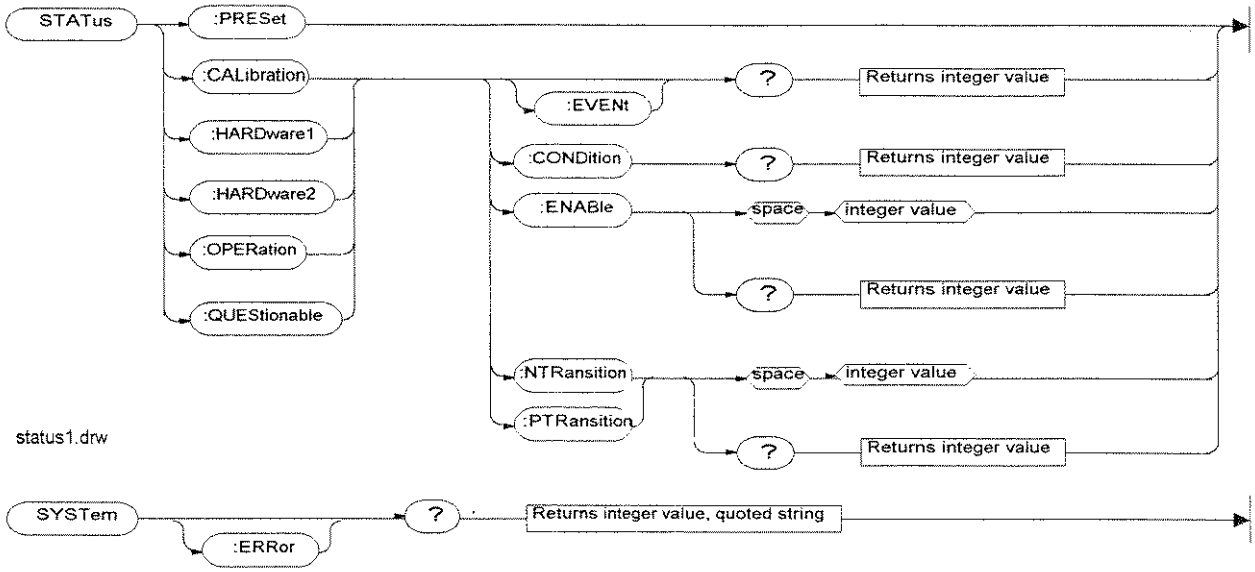
Tests


TESTs
:COMMent1
:COMMent2
space
'
string
?
Returns quoted string
:CONFigure
integer value
Returns 5 elements
:EXECution
:DESTination
Crt
Printer
:FAILure
Continue
Stop
:HEADing1
:HEADing2
:RESults
All
Failures
:RUN
Continuous
Single Step
CONTINUED ON NEXT PAGE
tests1.drw

Tests (continued)

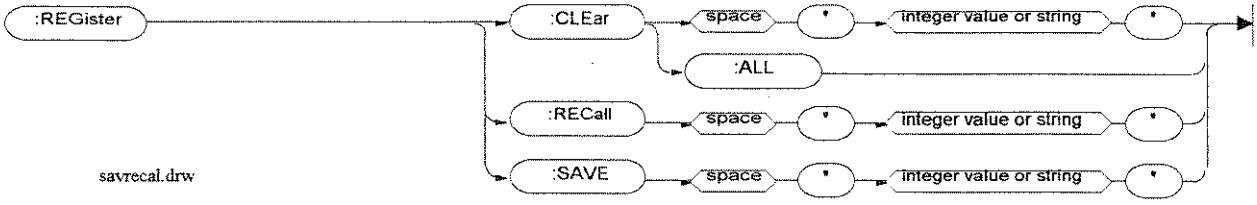


Status



status1.drw

Save/Recall Registers



savrecal.drw

Equivalent Front-Panel Key Commands

Most front-panel keys have an equivalent HP-IB command for remote use. The various key functions are explained in more detail in the User's Guide, chapter 3, *Operating Overview*, and in chapter 5, *Connector, Key, and Knob Descriptions*.

All command examples are in BASIC.

(SHIFT), (CANCEL), Cursor Knob

These functions are not required for HP-IB use, and have no equivalent HP-IB commands.

DATA Keys

Besides the number keys, the DATA keys contain the units keys, (ON/OFF), YES, NO, and (ENTER). The unit functions are described in *Specifying Units of Magnitude* in chapter 2. The (ON/OFF) function is described in *Using STATE Commands* in chapter 2. YES, NO, and (ENTER) are not required for HP-IB operation.

DATA FUNCTION Keys

The Data Functions enable you to change the way measurements are calculated and displayed, and provide measurement limit indicators.

The syntax shown illustrates only one possible form for each command. Refer to the *Number Measurement Syntax* diagram for any additional forms.

Each Data Function is described in chapter 3 of the User's Guide.

Guidelines for Using Data Functions

- Data Functions are turned on and off using STATe commands.
- Attribute units (AUNits) are used when some Data Functions (such as REF SET) are enabled. This allows you to discriminate between the units used for the measurement result and the units used for the Data Function being used.
- Data Function values, such as the number of Averages or Reference value, are retained for later use if the function is turned off. However, the values are forgotten under the following conditions:
  - The Test Set is turned off.
  - The Test Set is PRESET.
  - A saved register is recalled.

AVG

This command defines the measurement averaging function. Although this function can be accessed using HP-IB, it is primarily designed for manual instrument. If you need to average measurements, we recommend that you write a short averaging routine for your specific needs.

Syntax :AVERage:VALue <value>

Example Set an averaging value of 15 for the FM Deviation measurement.
OUTPUT 714;"MEAS:AFR:FM:AVER:VAL 15"

To Reset Averaging. Use the RES command to restart the averaging algorithm used to calculate an averaged measurement.

Syntax :AVERage:RESet

Example OUTPUT 714;"MEAS:AFR:FM:AVER:RES"

HI LIMIT/LO LIMIT

These commands set the measurement HI LIMIT and LO LIMIT values.

Syntax—:HLIMit:VALue <value>

Syntax—:LLimit:VALue <value>

Example Set a HI LIMIT of 7.5 kHz for the FM Deviation measurement.
OUTPUT 714;"MEAS:AFR:FM:HLIM 7.5 KHZ"

Example Set a LO LIMIT of 2.5 kHz for the FM Deviation measurement.
OUTPUT 714;"MEAS:AFR:FM:LLIM 2.5 KHZ"

Unless you specify a different unit, the HP-IB unit for the measurement is assumed when setting HI and LO limits.

Detecting an Exceeded Limit. Use the EXCeeded command to find out if a limit has been exceeded. The returned value is either 0 (no) or 1 (yes).

Syntax :HLIMit:EXCeeded?

Syntax :LLIMit:EXCeeded?

Example OUTPUT 714;"MEAS:AFR:FM:HLIM:EXC?"

Example OUTPUT 714;"MEAS:AFR:FM:LLIM:EXC?"

To Reset Limits. Use the RES command to reset a limit if it is has been exceeded.

Syntax :HLIMit:RESet

Syntax :LLIMit:RESet

Example OUTPUT 714;"MEAS:AFR:FM:HLIM:RES"

Example OUTPUT 714;"MEAS:AFR:FM:LLIM:RES"

INCR SET

This command specifies an increment value for appropriate numeric fields.

Syntax :INCR <value>

Example Set the increment value for the `RF Gen Freq` field to 2.5 MHZ.
OUTPUT 714;"RFG:FREQ:INCR 2.5 MHZ"

Note: HP-IB units are assumed for the increment value unless you specify a Display unit.

Integer-only fields (such as `Intensity` and `Print Adrs`) have a fixed increment of '1', and cannot be changed.

Specifying the INCRement Mode. The INCR MODE command specifies linear or logarithmic increments.

Syntax :INCR:MODE <LOGarithm or LINear>

Example OUTPUT 714;"RFG:FREQ:INCR:MODE LOG"

INCR÷10

This command reduces the current increment value for a setting by 10. Integer-only fields (such as `Intensity` and `Print Adrs`) have a fixed increment of '1', and cannot be changed.

Syntax :INCRement:DIVide

Example Reduce the increment value from 10 MHz to 1 MHz for the `RF Gen Freq` field.
OUTPUT 714;"RFG:FREQ:INCR:DIV"

INCR×10

This command increases the current increment value for a setting by 10. Integer-only fields (such as `Intensity` and `Print Adrs`) have a fixed increment of '1', and cannot be changed.

Syntax :INCRement:MULTiply

Example Increase the increment value from 1 MHz to 10 MHz for the `RF Gen Freq` field.
OUTPUT 714;"RFG:FREQ:INCR:MULT"

Increment Up/Down (Arrow Keys)

This command changes a setting's value by one increment value (up or down). The increment value is determined by the INCR SET (INCR) function.

Syntax :INCRement <UP or DOWN>

Example Increment the `RF Gen Freq` field's value.
OUTPUT 714;"RFG:FREQ:INCR UP"

METER

This command is used to turn the METER function on and off.

Syntax :METer:STATe <ON or 1, OFF or 0>

Example Change the `TX Power` measurement from a digital display to an analog display.
OUTPUT 714;"MEAS:RFR:POW:MET ON"

Meter Interval. This command defines the number of intervals displayed when the METER function is used.

Syntax :METer:INTerval <value>

Example OUTPUT 714;"MEAS:RFR:POW:MET:INT 5"

Meter Hi End/Lo End. These commands define the end values of the meter display.

Syntax :METer:HEND <value>

Syntax :METer:LEND <value>

Example OUTPUT 714;"MEAS:RFR:POW:MET:HEND 20W"

Example OUTPUT 714;"MEAS:RFR:POW:MET:LEND 20W"

Unless you specify a different unit, the HP-IB unit for the measurement is assumed when setting Meter end points.

Meter Lo End. This command defines the low end of the meter display.

Syntax :METer:LEND <value>

Example OUTPUT 714;"MEAS:RFR:POW:MET:LEND 0"

Unless you specify a different unit, the HP-IB unit for the measurement is assumed when setting Meter end points. In the above example, Watts are assumed for the units.

REF SET

This command defines the reference level when making a referenced measurement.

Syntax :REFerence <value>

Example OUTPUT 714;"MEAS:RFR:POW:REF 10"

Unless you specify a different unit, the HP-IB unit for the measurement is assumed when setting Meter end points.

INSTRUMENT STATE Keys

ADRS

This command allows you to change the HP-IB address of the Test Set. The current address is viewed by either pressing (SHIFT) then (LOCAL), or by looking at the `HP-IB Adrs` field on the **I/O CONFIGURE** screen.

All future HP-IB commands must use the new address after it is changed.

Syntax CONFigure:BADDress <number>

Example Change the HP-IB address to 15.
OUTPUT 714;"CONF:BADD 15"

LOCAL

This key returns control of the instrument to the front-panel controls after using HP-IB. This is a bus-level command.

Syntax LOCAL <address>

Example Restore manual control to the instrument at HP-IB address 714.
LOCAL 714

MEAS RESET

This command clears the measurement "history" for all of the instrument's measurement algorithms: AVG, LIMITs, Peak Hold, and autoranging. This re-starts all measurements that are in progress.

Syntax MEASure:RESet

Example OUTPUT 714;"MEAS:RES"

PRESET

This command resets the Test Set to its power-up state. This is an IEEE 488.2 Common Command.

Syntax *RST

Example OUTPUT 714;"*RST"

RECALL

This command recalls an instrument setup that has been saved. The REGister command in the syntax is optional.

Syntax REGister:RECall '<quoted string>'

Example Two ways of recalling register SETUP1-
OUTPUT 714;"REG:REC 'SETUP1'"
OUTPUT 714;"REC 'SETUP1'"

See Also

Refer to the *SAV and *RCL Common Commands described later in this section.

SAVE

This command saves an instrument setup for later use. The REGister command in the syntax is optional.

Syntax REGister:SAVe '<quoted string>'

Example Two ways of saving register SETUP1-
OUTPUT 714;"REG:SAVE 'SETUP1'"
OUTPUT 714;"SAVE 'SETUP1'"

Removing Saved Registers. One or all instrument setups you have saved can be removed from memory. The REGister command in the syntax is optional.

Syntax REGister:CLEar '<quoted string>'or :ALL

Examples
OUTPUT 714;"REG:CLE 'SETUP2'" - Removes register SETUP2.
OUTPUT 714;"CLE:ALL" - Removes all saved registers.

See Also

Refer to the *SAV and *RCL Common Commands described later in this section.

SCREEN CONTROL Keys

RX, TX, DUPLEX, TESTS, MSSG, HELP, CONFIG

The DISPlay command is used to access any screen, including those manually accessed using these keys.

Syntax DISPlay <screen>

Example OUTPUT 714;"DISP AFAN"

Table 3-1. Screen Mnemonics for the Display Command

ACPower	RX
AFANalyzer	SANalyzer (Spectrum Analyzer)
CONFigure	SERVice
DECoder	TCON (TESTS Edit Configuration Screen)
DUPLex	TDMA (Requires additional hardware)
ENCoder	TESTs (Main TESTS screen)
HELP	TFReq (TESTS Edit Frequencies Screen)
IOConfigure	TIBASIC (TESTS IBASIC Controller Screen)
MESSages	TMAKe (TESTS Procedure Manager Screen)
OSCilloscope	TPARm (TESTS Edit Parameters Screen)
PCONfigure	TSEQn (TESTS Edit Sequence Screen)
RFANalyzer	TSPec (test specs Setup Screen)
RFGenerator	TX
RINTerface	

HOLD

There is no equivalent single HP-IB command for the measurement HOLD key function. However, you can imitate this function using Single Triggering of measurements. Refer to the *Triggering Measurements* information later in this section.

PREV

There is no equivalent HP-IB command for the PREV key function.

PRINT

The PRINT function is used to print a 'pixel dump' of the displayed screen, and does not have an equivalent HP-IB command. To print measurement results using HP-IB, you must query the measurement and print the value in a format determined by your program.

USER Keys

There are no equivalent HP-IB commands for the USER keys.

Common Commands

>HP-IB:Common Commands

Common Commands refer to commands defined by the IEEE 488.2 bus standard to have a common response by any instrument connected to the bus. For example, the *RST command must be recognized by any instrument on the bus as an instrument Reset.

All Common Commands are immediately preceded by an asterisk (*).

Refer to the IEEE 488.2 standards for additional information on the functions and uses of these commands.

*CLS (Clear Status)

The Clear Status command clears the status registers and associated status data structures summarized in the Status Byte, such as the Event Status Register. This command also clears all status related queues (with the exception of the Output Queue) such as the error messages in the Error Message Queue.

*ESE (Event Status Enable)

The Standard Event Status Enable command sets the Standard Event Status Enable Register bits. The data is in decimal form. The device will round this number to an integer. Expressing this number in binary would then represent the values of the individual bits of the Standard Event Status Register. The number sent must be in the range of 0 to 255 or an Execution Error occurs.

For example, to enable bit 5 (Command Error) and bit 2 (Query Error) using BASIC, you would send the command-
OUTPUT 714;"*ESE 36"

The value is interpreted like this-

```
ESE command value (36 decimal) = 0   0   1   0   0   1   0   0  binary
Corresponding Register Bits        7   6   5   4   3   2   1   0
Binary weighting                 128  64  32  16   8   4   2   1
```

*ESE? (Event Status Enable Query)

The Event Status Enable query reads the contents of the Standard Event Status Enable Register. The Test Set responds by sending the contents of the Standard Event Status Enable Register back as an integer (0 to 255 decimal).

*ESR? (Event Status Register Query)

The Event Status Register query reads the contents of the Standard Event Status Register. The register is cleared when read. It returns an integer, which when converted to binary, represents the contents of the individual bits within the register. The number returned will be in the range 0 to 255 (decimal).

*IDN? (Identification Query)

The Identification query requests the Test Set to send its identification information over the bus as an ASCII Response Data element. The data is organized into four fields separated by commas. These fields are defined as follows:

```
Field 1: Manufacturer          returns "Hewlett-Packard"
Field 2: Model                 returns Test Set model number
Field 3: Serial Number         not used - returns "0"
Field 4: Firmware version      returns firmware rev. number
```

For example, the returned response from your Test Set may be:

```
Hewlett-Packard,8920A,0,A.12.01
```

In this example, all information but the serial number is returned. (The Test Set's serial number can not be read over HP-IB; it is written on the instrument's rear panel.)

*OPC (Operation Complete)

The Operation Complete command tells the Test Set to set bit #0 in the Standard Event Status Register when it completes all pending operations.

*OPC? (Operation Complete Query)

The Operation Complete query requests that an ASCII '1' (decimal 49) be placed in the Test Set's output queue when it completes all pending operations.

*OPT? (Option Identification Query)

The Option Identification query tells the Test Set to identify any reportable device options. The Test Set returns several fields on one line separated by commas. The entire length of the response will be $\leq$255 characters.

For example, the returned response from your Test Set may look like this:

```
OPTIONAL RAM,SIGNALING,I/O OPTION,SPECTRUM ANALYZER,0,HIGH
STABILITY REF,0,C MESSAGE
```

*PCB (Pass Control Back)

The Pass Control Back command tells potential controllers what bus address to pass control to. Refer to *Passing Control* in chapter 4.

*RCL (Recall)

The Recall command restores the instrument to the state saved in a memory register. The *RCL command is followed by a decimal number in the range of 0 to 99.

*RCL 38

This command cannot be used to retrieve registers saved using any letters or other non-numeric characters. To recall saved registers using a different name format, use the RECall command.

*RST (Reset)

The Reset command restores most of the Test Set's functions to the state they are in when the instrument is turned on. Most changes made after the instrument was turned on are forgotten.

Functions not affected by this command include some of the **CONFIGURE** screen settings, the **SERVICE** screen latch settings, and some **TESTS** screen fields. (Refer to the field descriptions for these screens in chapter 4 of the User's Guide to see which fields are not affected.)

The *RST command aborts all pending operations, and forces the Test Set to forget any previously-received *OPC or *OPC? commands.

The Service Request Enable (SRQ) Register and Standard Event Status Enable (ESE) Register are not affected by the *RST command.

Triggering is reset to full/repetitive.

*SAV (Save)

The Save command stores the present instrument state in a memory register. This command is followed by a decimal number in the range 0 to 99. The number is used to identify the instrument state for later retrieval using the *RCL command, or by using the RECall command.

To save a register using an alphanumeric name, or a number >99, use the SAVE command.

*SRE (Service Request Enable)

The Service Request Enable command sets the Service Request Enable Register. This register determines what bits in the Status Byte will cause a service request from the Test Set. The data sent with the command is in decimal form, and is converted to its binary form by the Test Set to set the individual bits of the Service Request Register.

*SRE? (Service Request Enable Query)

The Service Request Enable query reads the contents of the Service Request Enable Register. The Test Set returns an integer in the range of 0 to 63, or 128 to 191 (since bit 6, RQS, can not be set).

*STB? (Status Byte Query)

The Status Byte query reads the status byte with the MSS (Master Summary Status) bit. The Test Set returns an integer in the range of 0 to 255. Each bit in the binary equivalent of the returned integer represents the contents of the Status Byte. Bit 6 represents MSS rather than RQS (Request Service).

*TRG (Trigger)

The Trigger command causes all instruments on the bus that are addressed to listen to initiate a pre-programmed action simultaneously. The Test Set triggers all active measurements when this command is received.

*TST? (Self-Test Query)

The Self-Test query causes the Test Set to execute an internal self test and report any detected errors. This is a subset of the test the Test Set performs each time it is turned on.

The returned value over HP-IB is the decimal equivalent of the hexadecimal error code displayed on the Test Set.

For example, if a front-panel key is held in during the self test, the returned value over HP-IB is 128 (decimal). The message appearing at the top of the Test Set's display is 'Error code:0080' (hex). (128 decimal = 80 hex)

The following table gives the decimal values for any individual failures. Multiple failures produce a sum of the decimal values for the individual failures. (For example, a 12 indicates both ROM Checksum and Standard Non-Volatile System RAM failures.)

Table 3-2. Returned Decimal Values for Self-Test Failures

Detected Error	Returned Error Code (Decimal)
68000 Processor Failure	2
ROM Checksum Failure	4
Standard Non-Volatile System RAM Failure	8
Optional Non-Volatile System RAM Failure	16
6840 Timer Chip Failure	32
Real-time Clock Chip Failure	64
Keyboard Failure (stuck key)	128
RS-232 Chip (I/O option installed and not functioning correctly)	256
Serial Bus Communication Failure with a Standard Board	512
Signaling Board Self-Test Failure	1024
CRT Controller Self-Test Failure	2048

*WAI (Wait To Continue)

The Wait command causes the Test Set to wait until all previous commands or queries are completed. The Test Set then continues executing any commands that follow the *WAI command.

Triggering Measurements

>HP-IB:triggering measurements *(Refer to the* **TRIGger** *syntax diagram.)*

Measurements can be triggered several ways. Some modes are faster than others; some modes provide settling for signals that may contain initial transients when switched. The best triggering mode to use depends on your test requirements.

When a trigger command is sent to the Test Set, *all* active measurements are triggered at the same time. Active measurements are those that are on and can be accessed over HP-IB.

Local/Remote Triggering Changes

Triggering settings are always returned to their default states whenever the Test Set is in LOCAL mode, regardless of any changes in the triggering mode you made during HP-IB (REMOTE) control. If the Test Set returns to REMOTE mode, measurement triggering returns to the state it was last set to using HP-IB.

Note

>lock up:HP-IB bus **Bus Lock Up:** The HP-IB bus will go into an indefinite pause in its operation (lock up) if the Test Set does not successfully make a measurement.

You should include measurement timeout routines in your program to CLEAR the bus and ABORt the trigger if a measurement is not triggered within a specified amount of time. This provides a "clean" method of preventing the bus from locking up.

Trigger Syntax Descriptions

TRIGger

This command tells the Test Set to "start a measurement immediately". The type of triggering used depends on the triggering MODE settings. This command is the equivalent of the IEEE 488.2 common command *TRG.

The IMMediate statement is implied and is optional.

Syntax: TRIGger:IMMediate
Example: OUTPUT 714;"TRIG"

ABORt

This command tells the Test Set to stop trying to trigger measurements after a trigger command has been sent. If for any reason a valid measurement cannot be made, this command allows your program to continue without waiting any longer for the attempted measurement trigger.

Syntax: TRIGger:ABORt
Example: OUTPUT 714;"TRIG:ABOR"

Trigger MODE Commands

Two triggering settings are accessed using the MODE command: RETRiggering and SETTling.

RETRiggering determines how often measurements are triggered. >HP-IB:RETRiggering

REPetitive retriggering causes the Test Set to be continuously re-triggering all active measurements. This does not require a TRIGger command to be sent before a measurement can be made.

Any measurements that rely on external signals or hardware-generated events (such as the DTMF Decoder) are automatically re-armed after being triggered.

SINGle retriggering tells the Test Set to wait until a TRIGger command is received before triggering a measurement. When a measurement is made, the value for all active measurements are held until another trigger occurs. This lets you query a group of measurements that were triggered at the same time. This is the same as the front-panel HOLD function.

Any measurements that rely on external signals or hardware-generated events (such as the DTMF Decoder) must be re-armed with a new trigger command before another measurement can be made.

Syntax

```
TRIGger:MODE:RETRigger REPetitive
TRIGger:MODE:RETRigger SING
```

Examples

```
OUTPUT 714;"TRIG:MODE:RETR REP"
OUTPUT 714;"TRIG:MODE:RETR SING"
```

>HP-IB:trigger SETTling **SETTling** specifies how long a measurement is delayed to allow transient signals to settle.

FAST settling tells the Test Set to send the first measurement made after triggering. This provides faster measurements, but may allow switching transient signals to degrade the measurement.

FULL settling delays the measurement long enough to allow most transients to pass. This provides the most reliable measurements.

Syntax

```
TRIGger:MODE:SETTling FAST
TRIG:MODE:SETTling FULL
```

Examples

```
OUTPUT 714;"TRIG:MODE:SETT FAST"
OUTPUT 714;"TRIG:MODE:SETT FULL"
```

Default Triggering Settings

The Test Set uses FULL SETTling and REPetitive RETRiggering whenever is it turned on, reset, or put into LOCAL mode after being controlled using HP-IB. This is also the default HP-IB triggering after a *RST command is received.

Triggering Settings for Fastest Measurements

Use the following instrument and triggering setup for the fastest possible measurements. (See *Speeding-up Measurements* in chapter 4.)

1. Turn off all auto-ranging and auto-tuning functions.
2. Use REPetitive RETRiggering.
3. Use FAST SETTling. [1]
4. Turn off all measurements that are not required.

[1] Using FAST settling increases the possibility of transients occurring during the measurement, resulting in inaccurate measurements.

Triggering Settings for the Most Reliable Measurements

Use the following instrument and triggering setup for the most accurate and reliable measurements.

1. Turn on all auto-ranging and auto-tuning functions. (This is the Test Set's default turn-on and PRESET state.)
2. Use SINGle RETRiggering.
3. Use FULL SETTling.
4. Individually trigger each measurement.

Measurement Pacing

>HP-IB:measurement pacing

You can pace measurements using the IEEE 488.2 Common Commands *OPC, *OPC?, and *WAI. These commands use the assumption that an operation has not completed until all active measurements have made at least one valid measurement, and all generated signals are within specifications. Refer to the IEEE 488.2 Interfacing Standard for more information on using these commands.

Arming Hardware-Triggered Measurements

>HP-IB:arming measurements

Some measurements, such as the Tone Sequence Decoder, require an external signal to trigger the measurement. These measurements require you to arm the measurement using the HP-IB trigger command to allow it to be triggered.

Using SINGle triggering, the measurement must be re-armed by your program after a measurement has been made.

Using REPetitive triggering, the measurement is continually re-armed after a measurement has been made.

If the required triggering signal is not received, or if the signal level is incorrect, the bus will 'lock up' when trying to re-back a measurement. The bus stays locked up until a measurement is made (an external trigger is detected).

Note

Bus Lock Up: The HP-IB bus will go into an indefinite pause in its operation (lock up) if the Test Set does not successfully make a measurement.

You should include measurement timeout routines in your program to CLEAR the bus and ABORt the trigger if a measurement is not triggered within a specified amount of time. This provides a "clean" method of preventing the bus from locking up.

4

Advanced Operations

Speeding-Up HP-IB Measurements

Measurement times can be significantly decreased by optimizing the Test Set's settings for the desired measurement(s). This can be helpful when making a large number of measurements, but is usually unnecessary when making a few measurements.

This is the general process for speeding-up measurements:

1. Use STATe commands to turn off all unneeded measurements for the screen you are displaying.
2. Disable RF Auto-tuning by setting the `Tune Mode` field to `Manual` using the command:
   RFAN:TMOD 'MANUAL'
3. Disable RF Auto-ranging by setting the `Input Atten` field to `Hold` using the command:
   RFAN:ATT:MODE 'HOLD'
4. Disable AF Auto-ranging by setting the `Gain Cntl` field to `Hold` using the command:
   AFAN:RANG 'HOLD'
5. Use REPetitive RETRiggering (See *Triggering Measurements* in chapter 3.)
6. If you are making an AF Frequency measurement, set the AF Cnt Gate to 10 ms using the command:
   AFAN:GTIM 10 MS
7. If you are making a TX Frequency or TX Freq Error Measurement, set the RF Cnt Gate to 10 ms using the command:
   RFAN:GTIM 10 MS
8. In your test program, group measurements together that use the same, or nearly the same, instrument settings.

Disabling Automatic Functions

The auto-ranging and auto-tuning functions continually calculate and adjust several instrument settings to provide the optimum Test Set settings for each measurement. This results in greater measurement accuracy and increased measurement time. If you disable the automatic functions, you must manually set the settings they control to their required states in your program before making any measurements.

The best way to see what these settings should be is to perform your test manually using the automatic functions. Record the values for these settings and use them in your test program.

Status Reporting

The Status Byte is an 8-bit register that can be used to alert the operator to certain hardware and firmware conditions in the Test Set. Each binary-weighted bit corresponds to other status registers or message queues that monitor specific instrument conditions.

When a condition change occurs in any of these registers, the corresponding bit in the Status Byte is 'asserted' (turned on), causing a service request (SRQ) if enabled.

Querying the contents of the affected status register(s) allows you to read-back the condition that caused the service request.

Note

The following information is specific to the Test Set's status structures and commands, and is not meant as a complete tutorial on status reporting over HP-IB. For more detailed information on status reporting, refer to the IEEE 488.1 and IEEE 488.2 standards, or the *Tutorial Description of the Hewlett-Packard Interface Bus*, HP P/N 5952-0156.

Refer to the *Common Commands* in chapter 3 for more information on the commands used with the Status Byte and associated registers and queues.

Status Reporting Structure

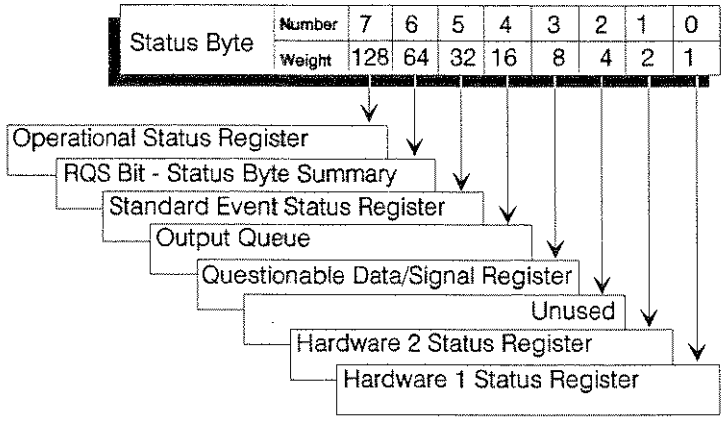


Figure 4-1. Status Byte Contents

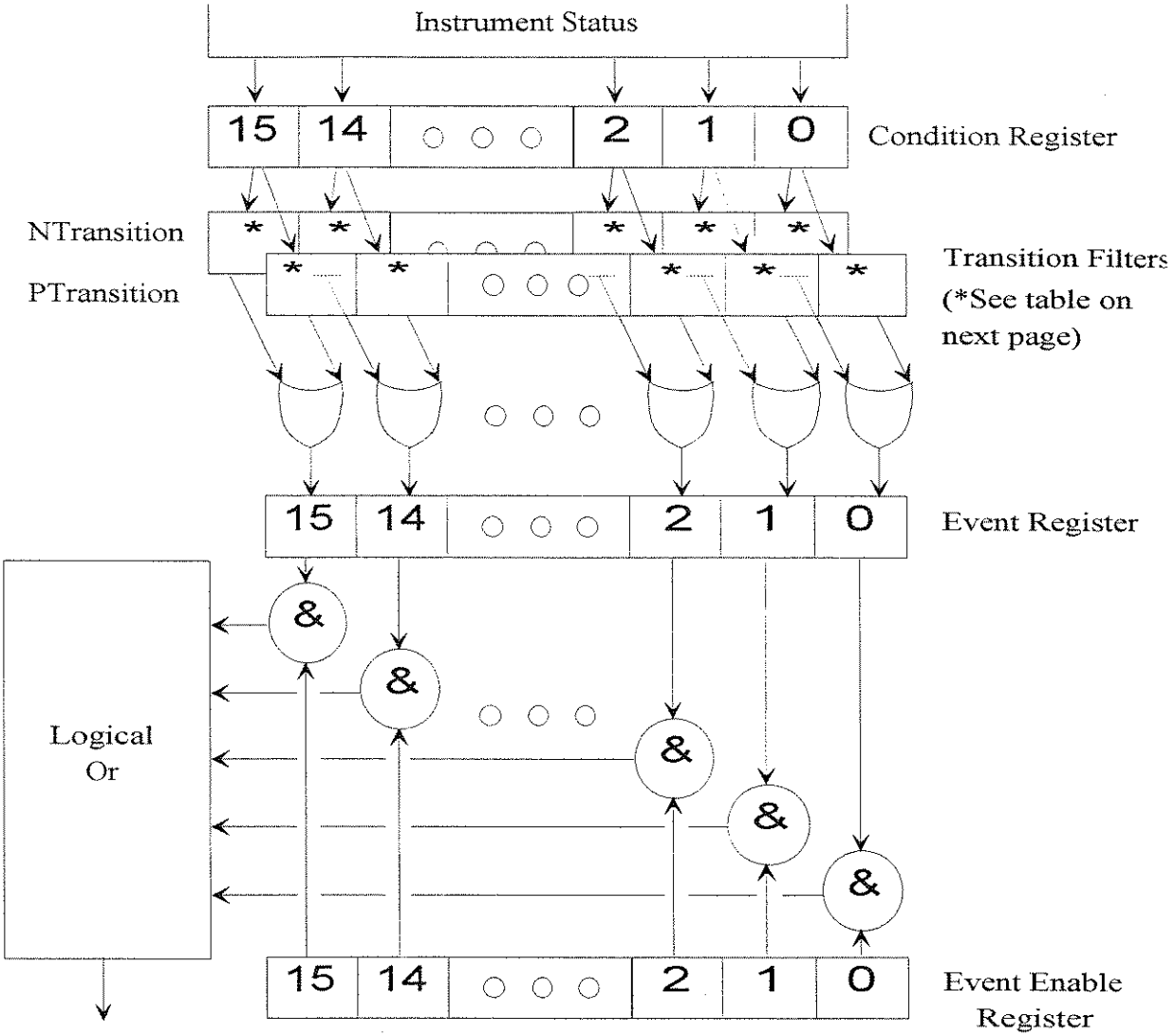


Figure 4-2. 16-bit Status Register Structure

Status Register Components

(Refer to the preceding figure.)

The CALibration, HARDWARE1, HARDWARE2, OPERation, and QUEStionable Data registers contain sub-registers that define the operation of each register. The sub-registers are used to record instrument conditions, filter the recorded information, and enable the register to report that information.

Condition Registers

The Condition registers continuously monitor each of the Test Set's operating conditions that have an assigned register bit. Whether an existing condition passes to the Event Register or not depends on the Transition Filter setting.

The CONDition? command lets you find out the instrument conditions associated with that register, independent of whether or not the register has been ENABled to signal the Status Byte. The value of the Condition registers is not affected when queried.

Syntax—STATus:<register>:CONDition?

Example—OUTPUT 714;"STAT:HARD1:COND?"

Transition Filters

The Positive and Negative Transition Filters are separately set to define what changes in the Condition register will get passed on to the Event register. The combination of these two filters results in four possible effects:

Table 4-1.

Filter Settings		Effect of Transition Filter On Condition Register Changes
Positive Transition	Negative Transition	
0	0	No changes passed to Event register
0	1	Negative-going changes passed to Event register
1	0	Positive-going changes passed to Event register
1	1	All changes passed to Event register

The default settings for all registers only allow positive-going transitions to be passed through all bit positions.

The Transition Filter bits are set by entering the decimal value that corresponds to the sum of the binary-weighted values for the desired bits.

Syntax

STATus:<register>:PTRansition <decimal value>

STATus:<register>:NTRansition <decimal value>

Examples

OUTPUT 714;"STAT:HARD1:PTR 5" !Set bits 0 and 2 positive-going.

OUTPUT 714;"STAT:HARD1:NTR 10" !Set bits 1 and 3 negative-going.

Event Register

This read-only register indicates what events have occurred since the register was last cleared. Events may be transient conditions that existed, such as HP-IB errors, or conditions that passed through the Transition Filters from the Condition register. Each bit in this register is logically ANDed with the Event Enable register bits, and the results are they ORed to determine if an assertion is passed on to the next register or Status Byte.

The returned value is equal to the sum of the binary-weighted bits of the Event register. Reading this register clears its contents. Refer to the *CLS Common Command description in chapter 3.

Syntax

STATus:<register>:EVENt?[1]

STATus:<register>?

[1]The :EVENT? syntax is optional.

Examples

OUTPUT 714;"STAT:HARD1:EVEN?"

OUTPUT 714;"STAT:HARD1?"

Event Enable Register

This register specifies what conditions in the Event register are allowed to cause an assertion in the next register or Status Byte. The Event register is logically ANDed with each bit in the Enable register. The results are then logically ORed to set the summary bit that is sent to the Status Byte or next higher register.

Enabling the Status Byte

The sum of the binary-weighted values for each desired register are used with the *SRE command to specify what register(s) can cause a service request (SRQ). The RQS bit (6) is **always** enabled, but can not be asserted unless one or more other register or queue summary bits are enabled.

The following example sets the Status Byte to allow the Standard Event Status Register and Output Queue summary bits to cause a service request:

```
OUTPUT 714;"*SRE 48"
```

Reading the Status Byte Enable Setting

Use the query form (*SRE?) to read-back the current *enable* setting of the Status Byte.

Example

```
OUTPUT 714;"*SRE?"
```

Since you cannot set bit 6 (RQS), the returned value will be in the range of 0 to 63, or 128 to 191.

Reading The Status Byte

The *STB? Common Command reads the value of the Status Byte. If the returned value is >0, a Service Request has been asserted by one or more registers you have enabled.

Syntax—*STB?
Example—OUTPUT 714;"*STB?"

You can also use a Serial Poll to query the Status Byte value: SPOLL(714)

Defining the Service Request Settings

Use the following sequence to define what conditions you want to allow a service request (assert bit 6 of the Status Byte).

1. From all of the status registers and queues, decide what condition(s) you want to cause a service request.
2. Enable the corresponding status register bits for those conditions.
3. Enable the Status Byte to allow the desired register(s) and queues to cause a service request.

Register and Queue Contents

All status registers and queues in the Test Set conform to the IEEE 488.2 status reporting requirements.

Operational Status Register

Table 4-2. Operational Status Register (Status Bit 7)

Bit Number	Binary Weighting	Condition
15	32768	Not Used (Always 0)
14	16384	PROGram Running (IBASIC)
13	8192	TMSL Reserved
12	4096	Unused
11	2048	Unused
10	1024	Unused
9	512	Unused
8	256	Unused
7	128	Unused
6	64	Unused
5	32	Unused
4	16	Unused
3	8	Unused
2	4	Unused
1	2	Unused
0	1	Unused

Table 4-3. Standard Event Status Register (Status Bit 5)

Bit Number	Binary Weighting	Function
15	32768	Not Used (Always 0)
14	16384	Reserved by IEEE 488.2 (Always 0)
13	8192	Reserved by IEEE 488.2 (Always 0)
12	4096	Reserved by IEEE 488.2 (Always 0)
11	2048	Reserved by IEEE 488.2 (Always 0)
10	1024	Reserved by IEEE 488.2 (Always 0)
9	512	Reserved by IEEE 488.2 (Always 0)
8	256	Reserved by IEEE 488.2 (Always 0)
7	128	Power on
6	64	User Request
5	32	Command Error
4	16	Execution Error
3	8	Device Dependent Error
2	4	Query Error
1	2	Request Control
0	1	Operation Complete

Output Queue (Status Bit 4)

After the Test Set receives a query, it sends the requested value to the Output Queue. Depending on what you query, the returned value may appear in the Output Queue almost immediately, or take several seconds (as is the case with some Decoder measurements).

By enabling the Status Byte to see the Output Queue, you can write your test program to trigger and query a measurement, and then perform other program functions until the returned measurement value appears in the queue, causing an SRQ.

Reading the returned value removes it from the queue.

Note

After you have queried the Test Set, you can not send any other commands **to the Test Set** until the queried information has been returned and entered into your program.

Questionable Data/Signal Status Register

Bit 8 of this register is the summary message for the Calibration register. You must enable this bit to allow a condition in the Calibration register to cause an SRQ with the Status Byte properly enabled.

Table 4-4. Questionable Status Register (Status Bit 3)

Bit Number	Binary Weighting	Condition
15	32768	Not Used (Always 0)
14	16384	Unexpected Parameter
13	8192	Unused
12	4096	Unused
11	2048	Unused
10	1024	Unused
9	512	Unused
8	256	Unused
7	128	Calibration (Summary Bit For Calibration Register)
6	64	Unused
5	32	Unused
4	16	Unused
3	8	Unused
2	4	Unused
1	2	Unused
0	1	Unused

Calibration Register

The summary message for this register appears as bit 8 in the Questionable Status/Signal register.

Table 4-5. Calibration Status Register

Bit Number	Binary Weighting	Condition
15	32768	Not Used (Always 0)
14	16384	Unused
13	8192	Unused
12	4096	Unused
11	2048	Unused
10	1024	Unused
9	512	Unused
8	256	Unused
7	128	Unused
6	64	Unused
5	32	Unused
4	16	TX Auto-zero Failed
3	8	Voltmeter Self-Calibration Failed
2	4	Counter Self-Calibration Failed
1	2	Sampler Self-Calibration Failed
0	1	Spectrum Analyzer Self-Calibration Failed

Table 4-6. Hardware Status Register #2: (Status Bit 1)

Bit Number	Binary Weighting	Condition
15	32768	Not Used (Always 0)
14	16384	Unused
13	8192	Unused
12	4096	Unused
11	2048	Unused
10	1024	Unused
9	512	Unused
8	256	Requested Audio Voltage Too Large
7	128	FM Too Large For RF Frequency
6	64	Requested Simultaneous AM & FM
5	32	Audio Auto Ranging Error
4	16	RF Auto Ranging Error
3	8	RF Auto Tuning Error
2	4	RF Gen/Anal/Offset Freq. Comp. Not Possible
1	2	RF Generator Level Too High For Output Port
0	1	Reference Level Too High/Low For Input Port

Hardware Status Register 1

Table 4-7. Hardware Status Register #1: (Status Bit 0)

Bit Number	Binary Weighting	Condition
15	32768	Not Used (Always 0)
14	16384	Radio Interface Interrupt #2 Tripped
13	8192	Radio Interface Interrupt #1 Tripped
12	4096	Decoder Result Available
11	2048	Decoder Input Level Too Low
10	1024	Decoder is Measuring
9	512	Decoder is Armed
8	256	Encoder Sending Aux Information
7	128	Encoder Sending
6	64	Unused
5	32	Measurement Limit(s) Exceeded
4	16	Power-up Self Test(s) Failed
3	8	Overpower Protection Tripped
2	4	Unused
1	2	External Mike Keyed
0	1	Low Battery Voltage

Using Service Requests in a Program

The following sample program enables the Test Set to cause an interrupt over HP-IB, and then sets the Test Set to make an AC Level measurement of AF Generator1. During the measurement setup, an invalid command is deliberately sent to cause an interrupt (see line 130). After the interrupt is detected, the operator is flagged with an error message, given the value of the Standard Event status register, and told to press CONT to continue testing.

Sample Status Reporting Program

```
10    Addr=714
20    OUTPUT Addr;"*RST" !Reset the Test Set.
30    OUTPUT Addr;"*CLS"  !Clear the Status reporting system.
40    OUTPUT Addr;"*SRE 32" !Enable the Status byte to see the
        Standard Event Register.
50    OUTPUT Addr;"*ESE 255" !Define the Standard Event register
        events that can cause an SRQ.
60    ON INTR 7 CALL Status !Jump to the status reporting
        subroutine when an SRQ is detected.
70    ENABLE INTR 7;2  !Enable an SRQ interrupt over the bus.
80    !START OF YOUR TEST PROGRAM
90    OUTPUT Addr;"DISP DUPLEX"
100   OUTPUT Addr;"AFG1:DEST 'AUDIO OUT'"
110   OUTPUT Addr;"AFG1:OUTP 1V"
120   OUTPUT Addr;"AFAN:INP 'AUDIO OUT'"
130   OUTPUT 714;"RFG:FREQ 2000 MHZ" !This bogus command
        causes an Execution Error in the Standard Event register.
140   OUTPUT Addr;"MEAS:AFR:ACL?"
150   ENTER Addr;Aclevel
160   PRINT "AC LEVEL IS ";Aclevel
170   END
180   ! End of your test program.
190   SUB Status
200   OFF INTR 7
210   CLEAR 714
220   PRINT "Test operation halted due to interrupt"
230   OUTPUT 714;"*ESR?"
240   ENTER 714;Status_val
250   PRINT "Standard Event status is ";Status_val
260   PRINT "PROGRAM PAUSED, HIT CONTINUE TO PROCEED"
270   PAUSE
280   SUBEND
```

Error Reporting

Error messages are stored in a first-in-first-out (FIFO) block of memory that holds up to 20 messages. A messages appears in this register any time bit 2, 3, 4, or 5 of the Standard Event Status register is asserted.

Each message contains a number and a quoted string description. If more than 20 errors are in the queue, and another error occurs, the last error is replaced with `error 350, "Too Many Errors"`.

Reading an error removes it from memory.

The types of errors reported are generally the same errors you can get using manual control. These include problems with the Test Set hardware, and executing or attempting to execute an improper command.

Messages are read-back by using the System Error query, and then entering the returned value into a string variable (be sure to DIMension the variable for $\leq$ 255 characters):
Syntax—SYSTem:ERRor?

Example of reading and printing an error message:

```
DIM Error$[255]
OUTPUT 714;"SYST:ERR?"
ENTER 714;Error_num,Error$
PRINT VAL$(Error_num)&Error$
```

Passing Control

In systems with more than one controller, only one can be active at a time. The currently-active controller can pass control to one of the other controllers, and a non-active controller can request control from the currently-active controller. Only the controller designated as the *system controller* can demand control of the bus.

The following general information provides the guidelines for pass control using the Test Set. Refer to the IEEE 488.2 standards for additional information.

System Control Using the Test Set

The Test Set can be configured as a stand-alone test system controller, eliminating the need for an externally-connected system controller.

To configure the Test Set as a system controller, set the **I/O CONFIGURE** screen's `Mode` field to `Control`. A **C** appears in the upper-right corner of the display to indicate the Test Set is in the Controller mode.

Pass Control Overview

Pass control allows your system controller to pass HP-IB control to the Test Set, making it the currently-active controller. After becoming the active controller, the Test Set can control other instruments or devices connected to the bus.

When to Pass Control

The Test Set must have active bus control under the following conditions:

1. Whenever the Test Set needs to control a connected HP-IB device, such as an external disk drive.
2. Printing a *screen image* to an HP-IB printer.
3. Running HP 11807 Radio Test Software that uses a connected HP-IB printer or other HP-IB device.
4. Running an IBASIC program that controls an HP-IB device using the external HP-IB bus.

Pass Control Back Conditions

After becoming an active controller, the Test Set passes control back to the system controller when any of the following conditions exist:

- An IBASIC program running in the Test Set is paused.
- An IBASIC program running in the Test Set is done executing.
- An IBASIC Reset occurs in the Test Set.
- The Test Set doesn't execute a controller command within 10 seconds of receiving control from the system controller (such as a command to print to an HP-IB printer or to perform a disk access operation from the **TESTS** screen).

As these conditions occur, you must pass control to the Test Set again to re-establish its role as the active controller if needed.

Pass Control Program Example

This example illustrates pass control by sending a program to the Test Set, passing control from the system controller to the Test Set, running the program that prints to a printer, and returning control to the system controller when the program stops.

Refer to *Status Reporting* earlier in this section regarding the use of the service requests that cause execution branching in this program.

```
10    Addr=714 !Define the Test Set's HP-IB address.
20    OUTPUT Addr;"prog:del:all"
30    OUTPUT Addr;"prog:def #0"
40    OUTPUT Addr;"10 EXECUTE(""REQUEST_CONTROL"")"
50    OUTPUT Addr;"20 WAIT 5"
60    OUTPUT Addr;"30 output 701;""Hello printer! This is from IBasic."""
70    OUTPUT Addr;"40 output 701;""This is the second line from IBasic"""
80    OUTPUT Addr;"50 end" END
90    OUTPUT 701;"System controller has downloaded the program."
100   PRINT "System controller has downloaded the program."
110   OUTPUT Addr;"*PCB 21" !*Tell Test Set where to pass control when done.*
120   OUTPUT Addr;"*CLS"          ! Setup SRQ for Request Control.
130   OUTPUT Addr;"*SRE 32"
140   OUTPUT Addr;"*ESE 2"
150   ON INTR 7 GOTO Pass_control !Look for Request Control command.
160   ENABLE INTR 7;2
170   OUTPUT 701;"System controller ready to receive Pass Control Request."
180   PRINT "System controller ready to receive Pass Control Request."
190   OUTPUT Addr;"DISP TIB"
200   WAIT 5 !Give Test Set time to change screens before running program.
210   OUTPUT Addr;"PROG:STATe RUN"  !Run program loaded into Test Set.
220 Delay_1:WAIT .01     ! Wait for Test Set to request control.
230   GOTO Delay_1
240    !
250 Pass_control:        ! Test Set has requested control
260   OFF INTR 7
270   PRINT "HP-IB interrupt seen"
280   Test_byte=SPOLL(Addr)
290   PRINT "Serial Poll response is "&VAL$(Test_byte)
300   OUTPUT Addr;"*CLS"      ! Look for IBasic program being done.
310   OUTPUT Addr;"STAT:OPER:PTR 0"
320   OUTPUT Addr;"STAT:OPER:NTR 16384"
330   OUTPUT Addr;"STAT:OPER:ENAB 16384"
340   OUTPUT Addr;"*SRE 128"
350   ON INTR 7 GOTO Ib_stopped
360   PASS CONTROL Addr     ! *** Passing control to the Test Set ***
370   ENABLE INTR 7;2
380 Delay_2:WAIT .01    ! Wait for the IBasic program to finish running.
390   GOTO Delay_2
400    !
410 Ib_stopped: !IBasic program has completed (control should be back)
420   OFF INTR 7
430   PRINT "HP-IB interrupt seen (IBasic stopped)"
440   Test_byte=SPOLL(Addr)
450   PRINT "Serial Poll response is "&VAL$(Test_byte)
460   OUTPUT 701;"RMB has received control again."
470   PRINT "System controller has received control again."
480   END
```

5

Memory Cards/Mass Storage

Introduction

This chapter contains information about the use of different types of mass storage for saving and retrieving files accessed by the Test Set's IBASIC Controller.

IBASIC language examples are provided to help you understand the use of mass storage, but are not intended as a comprehensive description of IBASIC mass storage commands and procedures. For additional IBASIC information, refer to the *HP Instrument BASIC Programmer's Guide.*

Types of Mass Storage

The Test Set can access information from four mass storage devices:

ROM - The Test Set's internal Read Only Memory containing factory supplied programs.

Cards - Two types of memory cards are used:

OTP (ROM) Memory Cards - Memory cards containing factory-supplied procedures and/or data that can not be over-written or erased by the user.

SRAM Memory Cards - Memory cards that contain Static Random Access Memory. Files can be written and over-written. Primarily used for program development and storage.

RAM - RAM disks are four battery-backed memory locations inside the Test Set's RAM that are used very much like RAM memory cards, but can not be removed.

Disk - External disk drives (flexible or hard).

Mass Storage Access

Programs and data saved on mass storage are accessed directly from these screens:

- The **IBASIC Controller** screen (accessed from the **TESTS** screen). You can access any type of file from this screen.
- The **TESTS** screen; using the `Procedure: Location` fields. These functions can only access "Procedure" files created using the **Procedure Manager** screen of the TESTS subsystem or the default procedure(s) shipped with HP 11807 software. When created, procedure file names are prefixed with a lower case **p** (pFM_TEST).

  A corresponding "Code" file (cFM_TEST) must reside on the same media for the procedure to work. (Refer to the **TESTS** screen description in chapter 4 of the User's Guide, and the *HP Instrument BASIC Programmer's Guide.*)
- The **Signaling Decoder** screen in NMT mode. Only user-written NMT tests can be accessed from this screen. NMT test files must be saved with a lower case **n** (nNMT_1).

Note

Hierarchical Files: The Test Set cannot save or access files using a hierarchical file system. Therefore, no directory path information can be used during mass storage operations.

DOS and LIF Format Considerations

Files can be saved and retrieved using DOS (Disk Operating System) and LIF (Logical Interchange Format) storage formats. LIF is the default format for all mass storage operations.

The **IBASIC Controller** screen can access either format using various commands and procedures listed in the *HP Instrument BASIC Programmer's Guide*.

The type of file created when saved is dependent on the storage format (DOS or LIF) and the IBASIC command used (SAVE or STORE).

Creating DOS and LIF Files

- Using the **SAVE** command to save a file to a LIF volume creates an **ASCII** file.
- Using the **STORE** command to save a file to a LIF volume creates an **HP-UX** file.
- Using either the **SAVE** or **STORE** command to save a file to a DOS volume creates a **DOS** file.

Files that have been SAVEd must be retrieved using the GET command:

```
SAVE "FM_TEST:,704,1"
GET "FM_TST:,704,1"
```

Files that have been STOREd must be retrieved using the LOAD command:

```
STORE "FM_TEST:,704,1"
LOAD "FM_TSTS:,704,1"
```

Use of Upper and Lowercase Filenames

LIF files preserve the use of uppercase and lowercase letters for storage and retrieval. If you use the name "**c**FM_TEST" to save the file to a LIF-formatted disk, the Test Set reads the file back as "**c**FM_TEST".

DOS always converts any lowercase letters to uppercase when files are saved. If you use the name "**c**FM_TEST" when saving a file, DOS saves it as "CFM_TEST".

Although the TESTS Subsystem looks for specific lowercase filename prefixes when retrieving files from the main **TESTS** screen or NMT Decoder, it will still get DOS files with all uppercase names. However, to be consistent in saving files, you should always use the correct lowercase filename prefix when saving any file for the TESTS Subsystem.

TESTS Subsystem DOS Restrictions

Since all DOS files have a **DOS** volume specifier, the IBASIC Controller cannot distinguish between DOS files that have been SAVEd, and those that were STOREd. This is a problem when trying to load a Procedure file from the **TESTS** screen. Procedure files relying on a DOS Code file that was STOREd will not run if you use the `Procedure: Location` fields to load them.

To use DOS program files with the TESTS Subsystem:

1. Access the **IBASIC Controller** screen and GET or LOAD your test program.
2. SAVE the program as a Code file, using a lower-case **c** as a prefix. Example: cFM_TEST
3. Use the **Procedure Manager** screen and make a procedure. If you catalogue mass storage, the procedure you made is prefixed by a lower-case **p** (pFM_TEST). (Refer to the **TESTS** screen Procedure Manager information in chapter 4 of the User's Guide.)

Using ROM

The Test Set comes with several procedures supplied on internal ROM that provide instrument diagnostics, periodic calibration, and a variety of helpful routines.

To see what each program does, use the `Procedure: Location` functions on the **TESTS** screen to select each program and read its description in the `Comment` field. Instructions for programs are displayed when the program is run.

ROM cannot be written to for user storage.

The ROM's memory designation is **:MEMORY,0,4**. For example: to catalogue the contents of internal ROM from the **IBASIC Controller** screen enter

```
CAT ":MEMORY,0,4"
```

.

Using Memory Cards

OTP (One Time Programmable) cards provide removable read-only storage. File editing and erasure are not possible. These cards cannot programmed by the Test Set; they require a special memory card programmer to save files.

SRAM cards provide removable read/write memory for your files, similar to a flexible disk. Data can be stored, re-stored, read, or erased as needed.

SRAM memory cards require a battery to maintain stored information.

Table 5-1. Memory Card Part Numbers

Memory	Type	Part Number
32 kilobytes	SRAM	HP 85700A
128 kilobytes	OTP	HP 85701A
128 kilobytes	SRAM	HP 85702A
256 kilobytes	OTP	HP 85703A
256 kilobytes	SRAM	HP 85704A
512 kilobytes	SRAM	HP 85705A
512 kilobytes	OTP	HP 85706A

Inserting and Removing Memory Cards

Figure 6-1 illustrates how to insert a memory card into the Test Set front panel. To remove a memory card, simply pull it out. The memory-card label is marked with an arrow that must be inserted on the same side as the arrow shown on the front-panel slot.

Memory cards may be inserted and removed with the Test Set powered on or off.

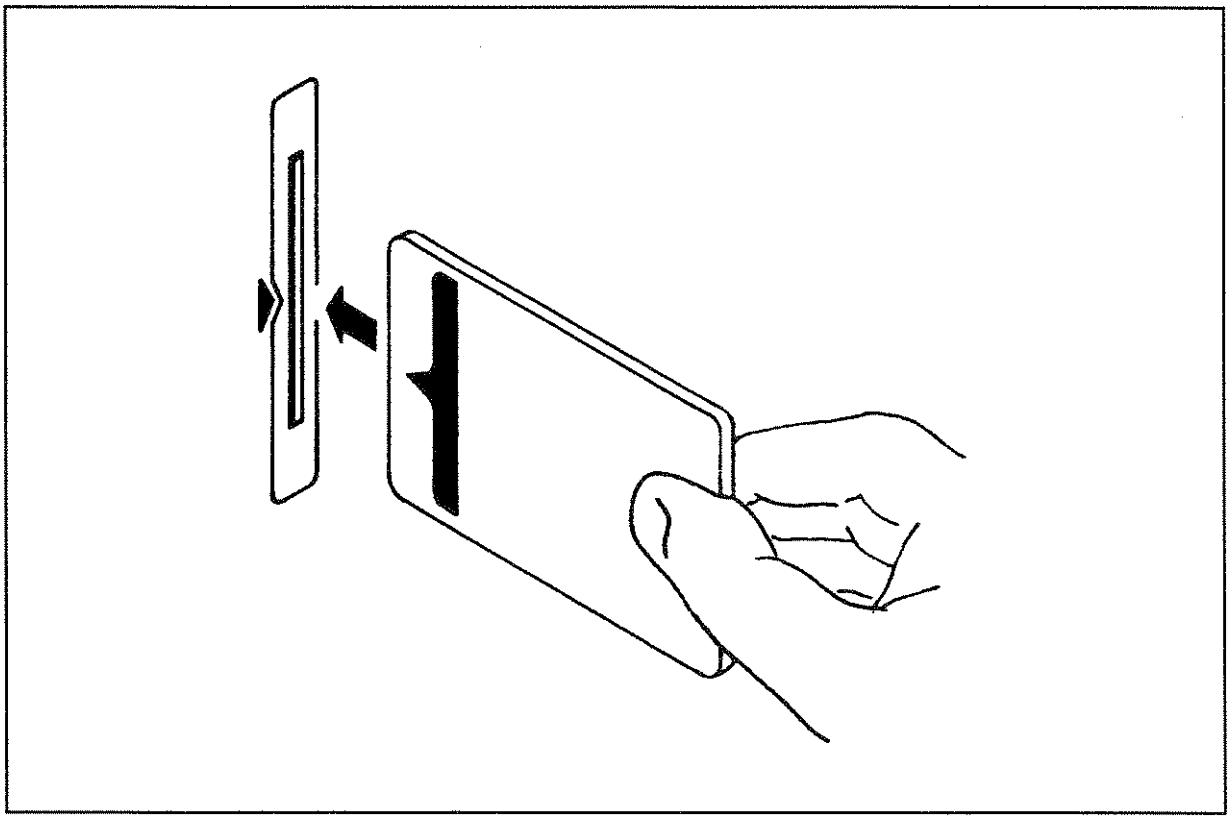


Figure 5-1. Inserting a Memory Card

Setting the Write-Protect Switch

The SRAM memory card's write-protect switch lets you secure its contents from being accidentally overwritten or erased. The switch has two positions:

- *Read-write* – The memory-card contents can be changed or erased, and new files may written on the card.
- *Read-only* – The memory-card contents can be read by the Test Set, but cannot be changed or erased.

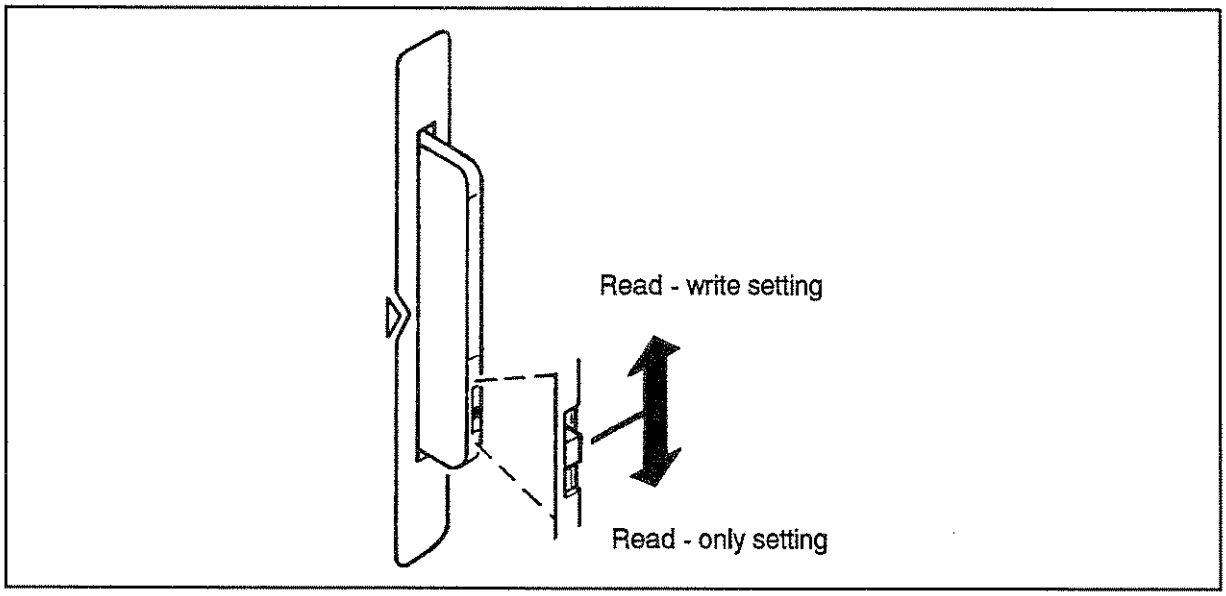


Figure 5-2. Setting the SRAM Write-Protect Switch

The Memory Card Battery

SRAM memory cards use a lithium battery: CR2016 or HP 1420-0383. SRAM cards typically retain data for over 1 year at 25° C. To retain data, the battery should be replaced annually.

Replacing the Battery

1. Turn the Test Set on and insert the memory card. An inserted memory card takes power from the Test Set, preventing the card's contents from being lost.
2. Hold the memory card in with one hand and pull the battery holder out with your other hand.
3. Install the battery with the side marked "+" on the same side marked "+" on the battery holder. Avoid touching the flat sides of the battery as finger oils may contaminate battery contacts in the memory-card.
4. Re-Insert the battery holder into the memory card.
5. Remove the memory card from the Test Set.

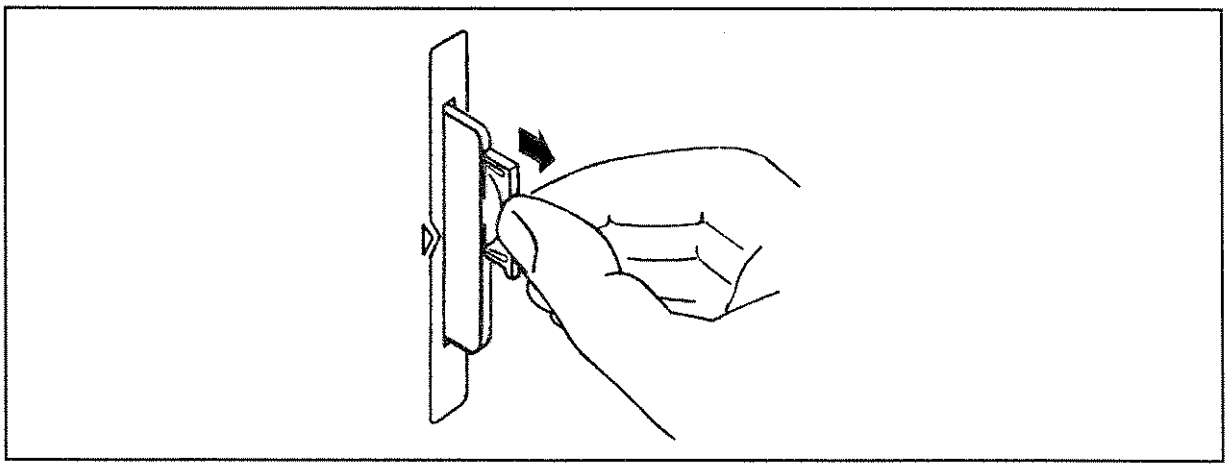


Figure 5-3. Replacing the Memory-Card Battery

Warning

Do not mutilate, puncture, or dispose of batteries in fire. The batteries can burst or explode, releasing hazardous chemicals. Discard unused batteries according to the manufacturer's instructions.

Addressing the Memory Card

The memory card slot is designated **:INTERNAL** memory, and is the default mass storage device for the Test Set. For example, to catalogue the contents of a memory card from the **IBASIC Controller** screen you could enter -

```
CAT ":INTERNAL"
```

or just

```
CAT
```

If you use the MSI (Mass Storage Is) command to change the mass storage default to a different device, the INTERNAL designation must be used to access the memory card slot. Any changes to the mass storage default are lost when the Test Set is turned off.

Initialization

All SRAM cards must be initialized before they can be used. The RAM_MNG procedure stored on internal ROM can be used to quickly initialize any SRAM memory card.

You can also initialize the card from the **IBASIC Controller** screen by inserting the memory card into the front panel slot and entering the IBASIC command:

```
INITIALIZE ":INTERNAL"
```

To verify that the memory card is properly initialized, enter the IBASIC commend:

```
CAT ":INTERNAL"
```

If the error message "`ERROR 85 Medium uninitialized`" appears on the screen, check the SRAM battery to ensure that it's charged and inserted correctly in the battery holder.

Backing Up Files

Making a copy of your memory card files helps guard against file loss due to memory card (or battery) failure.

Using the COPY_PL ROM Program

The program COPY_PL on Test Set ROM backs up **Procedure** *and* **Library** files onto a second SRAM memory card, and can also initialize an uninitialized SRAM memory card. This program does not copy executable program (Code) files, or copy files to OTP memory cards.

This program is intended for users with HP 11807 software or their own test software written to use the Tests Subsystem.

To Run COPY_PL

1. Access the **TESTS** screen.
2. Select the `Location` field and choose ROM.
3. Select the `Procedure` field and select `COPY_PL`.
4. Select the `Run Test` field to start the procedure.
5. Follow the displayed instructions.

Copying Files Using IBASIC Commands

Files can be copied from one mass storage device to another using IBASIC `COPY` commands. For example, to copy a file from an inserted memory card to the left drive of an external dual-disk drive with the address 704.0, you could enter the command:

```
COPY "FM_TEST:INTERNAL" TO "FM_TEST:,704,0"
```

You can back up files from one memory card to another by loading the program from the memory card into the Test Set, inserting a different initialized memory card, and then using the IBASIC `SAVE` command.

```
SAVE "FM_TEST:INTERNAL"
```

Using RAM

RAM Disk is a section of internal memory that acts much like a flexible disk. Programs can be stored, re-stored, erased, and retrieved.

The RAM disk is partitioned into four separate volumes; 0-3. Each volume is treated as a separate 'disk'. You can also specify the size of each disk in 256-byte increments.

The four RAM disk volumes are designated **:MEMORY,0,0** to **:MEMORY,0,3**. For example, to catalogue the contents of RAM disk volume '0' from the **IBASIC Controller** screen, enter

```
CAT ":MEMORY,0,0"
```

Volume 0's contents can be viewed and loaded from the three screens mentioned at the beginning of this section. Volumes 1, 2, and 3 can *only* be accessed from the **IBASIC Controller**.

Note

RAM Disk Erasure: Any existing programs or formatting on RAM is erased if you use the `RAM_MNG` or `COPY_PL` ROM programs, or the **SERVICE** screen's `RAM Initialize` function.

Therefore, you should only use RAM disks for short-term storage of files.

Initializing RAM Disks

Each RAM disk volume must be initialized before it can be used. Volume 0 can be initialized using the RAM_MNG procedure stored on internal ROM. Volumes 1, 2, and 3 must be initialized from the **IBASIC Controller** screen.

The optional 'volume size' in the following procedure lets you specify the memory area set aside for each disk in 256 byte blocks.

Follow these steps to initialize volumes 1, 2, or 3:

1. Access the **TESTS** screen.
2. Select `IBASIC` from the `Test Function` field to display the **IBASIC Controller** screen.
3. Using the knob, enter the command:

```
INITIALIZE ":MEMORY,0,<volume number 1-3>",<volume size>
   or
INITIALIZE ":MEMORY,0,1",50
```

Using Disks

The Test Set requires specific configuration and volume information to access external disk drives.

The **I/O CONFIGURE** screen's HP-IB `Mode` field must be set to `Control` any time an external disk is used.

To load files from the main **TESTS** screen or NMT **Signaling Decoder** screen, the disk address and volume must be entered in the `External Disk Specification` field on the **TESTS** (Edit Configuration) screen (example: **:,702,1**).

Initializing External Disks

Disks can be initialized for either LIF (Logical Interchange Format) or DOS (Disk Operating System) format using the Test Set. (See *DOS AND LIF Format Considerations* earlier in this chapter.)

Initializing a LIF Disk

LIF is the default format, and is assumed when initializing a disk unless DOS is specified:

```
INITIALIZE ":,<disk address>,<volume>"
      or
INITIALIZE ":,702,1"
```

Initializing a DOS Disk

You must specify the DOS format when initializing a DOS disk:

```
INITIALIZE "DOS:,<disk address>,<volume>"
      or
INITIALIZE "DOS:,702,1"
```

6

Test Set Instrument BASIC Overview

The Test Set contains an HP Instrument BASIC computer that can run programs to control the Test Set and any connected HP-IB equipped instruments. This provides a powerful test instrument and test system controller in one package.

The rest of this section of the manual refers to the HP Instrument BASIC Language simply as **IBASIC**.

The TESTS Subsystem and IBASIC

The Test Set's IBASIC computer is the "core" of an automated test environment referred to as the TESTS subsystem. The subsystem allows the operator to change the order and number of tests to be run, change test instrument operating parameters and frequencies, and change the specification limits for each test point. This allows programmers to write tests that can be easily run and altered by non-expert operators.

Programs can also be written that do not use the special TESTS subsystem capabilities, using only the IBASIC computer core.

Test Set Bus Architecture

The Test Set's IBASIC computer acts much like system controller connected by an HP-IB cable to the Test Set; but instead of a cable, the Test Set has its own internal control bus connected to the IBASIC controller. The internal bus address is **8xx**. (xx is any valid HP-IB address.) When you write programs to run on the Test Set's IBASIC computer to address Test Set functions, you must use the "8xx" address to output commands.

For example, if you want a program in the IBASIC computer to reset the Test Set at the start of a test procedure, the program code to do this would be written

```
OUTPUT 814;"*RST"
```

When the Test Set's HP-IB **Mode** field is set to **Control**, it takes on the role of system controller. This allows it to control other test instruments connected by HP-IB cables. Instruments controlled by the Test Set use the normal **7xx** HP-IB address prefix.

For example, if two Test Set's are used in a test system, and the second instrument's HP-IB address is 715, a program running in the *controlling* Test Set would output the command

```
OUTPUT 715;"*RST"
```

to reset the *controlled* Test Set.

Note

Multiple Controllers: Only one system controller can be connected to the bus at any time. If the Test Set is used in a test system that has its own controller, the Test Set cannot be used as a controller unless the system controller is turned off or disconnected from the bus.

If the Test Set is used as a controller in a system with another Test Set, the HP-IB **Mode** of the non-controller Test Set must be set to **Talk&Listen**.

Entering and Editing Programs

The **IBASIC Controller** screen is the "computer" for the TESTS subsystem. Programs can be entered into the IBASIC computer's RAM using a variety of methods:

- Using the IBASIC Controller screen and the Cursor Control knob.
- Downloading Programs from an external IBASIC controller connected to the Test Set by HP-IB.
- Using an external ASCII terminal or Personal Computer (PC) connected to the Test Set by RS-232.

Once a program is loaded into the **IBASIC Controller's** RAM, it can be run by using the knob to enter the `RUN` command, or by selecting the `Run` field in the top right corner of the screen.

Entering Programs Using the Cursor Control Knob

Using the "knob" to enter programs eliminates the need for Option 003 (HP-IB, RS-232, and current sensing capabilities). However, this method is very tedious to use especially if you have large programs to enter or edit.

1. Access the **TESTS** screen by pressing (TESTS).
2. Select the `Test Function` field at the bottom of the screen to display a list of choices.
3. Select `IBASIC` to display the **IBASIC Controller** screen.
4. After accessing the **IBASIC Controller** screen, position the cursor in front of the command line at the top of the screen and press the Cursor Control knob. A list of characters is displayed that you select from to enter your commands. A maximum of 100 characters may be entered into the command line. After the command is entered on the command line, select 'Done' at the top of the list of characters to execute it.

Commands and program lines are entered just as you would enter them using a keyboard. For example, to set the default mass storage device to the memory card slot, you would enter the command

```
MSI ":INTERNAL"
```

and select 'Done'

To list the contents of the default mass storage device, enter

```
CAT
```

and select 'Done'.

Downloading Programs Over HP-IB

The easiest way to enter and edit a program is to create it on your computer using your computer's editing features, and then download it into the Test Set. The usual development sequence is:

1. Write the program on your computer to control the Test Set using the normal **7xx** HP-IB address.
2. Run the program to verify that it controls the Test Set correctly.
3. Change the Test Set's HP-IB address in your program to **8xx**.
4. Download the program into the Test Set (See *Program Control*).
5. Run the program on the Test Set to verify correct operation.
6. Copy the program to a memory card for future use (See *Program Control*).

Entering Programs Using an ASCII terminal or PC

This procedure allows line-by-line entry of program code into the Test Set's IBASIC Computer.

Connecting the ASCII terminal or PC to the Test Set

1. Connect an RJ-11/RS-232 adapter (HP P/N 98642-66508) to the 25-pin RS-232 connector of your terminal or personal computer (PC). (If your PC has a 9-pin RS-232 port, use the appropriate adapter and use the table below to verify connections.)
2. Connect a 4-conductor RJ-11 cable (HP P/N 98642-66505) from the adapter to the Serial Port of the Test Set.

Note

RJ-11 Connectors: RJ-11 cables and adapters can be wired differently. If you buy a cable or adapter from a supplier other than HP, verify the connections for the pins indicated in the following table before connecting cables to the instruments.

Table 6-1. Cable and Adapter Pin Connections

Test Set RJ-11 Serial Port		Terminal/PC 25-Pin RS-232		Terminal/PC 9-Pin RS-232
Pin 2 (RX)	to	pin 2 (TX)	or	pin 3 (TX)
Pin 5 (TX)	to	pin 3 (RX)	or	pin 2 (RX)
Pin 4 (GND)	to	pin 7 (GND)	or	pin 5 (GND)

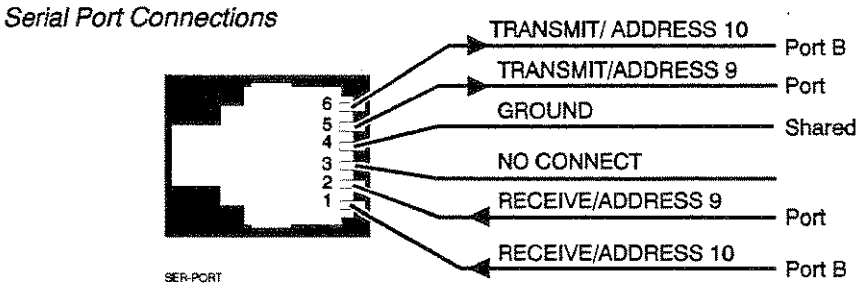


Figure 6-1. Serial Port Connections

Configuring the Test Set

1. Access the Test Set's **I/O CONFIGURE** screen.
2. Set `Serial In` field to `Inst` to allow the Test Set's IBASIC controller to accept characters from a PC or ASCII terminal.
3. Set `IBASIC Echo` to `On`.
4. Set `Inst Echo` to `On`.
5. Set the `Serial Baud` to 9600. (Baud can be altered as required by your terminal.)
6. Set `Parity` field to `None`.
7. Set `Data Length` to `8 bits`.
8. Set `Stop Length` to `1 bit`.
9. Set `Rcv Pace` to `None`.
10. Set `Xmt Pace` to `Xon/Xoff`.

Configuring Your Terminal or PC

If you are using a PC to enter programs, skip to *Configuring a PC With HP AdvanceLink*.

Configuring an ANSI Terminal.

1. Select **ANSI** operating mode.
2. Set **Baud Rate** to **9600** (if this rate is not available on your terminal, set it to a rate that can be selected on the Test Set's **I/O CONFIGURE** screen.
3. Set **Parity** to **none.**
4. Set **Data Bits** to 8.
5. Set **AnqAck** to **no** (or **none**).
6. Set **Receive/Transmit Pacing** to match the Test Set's settings.

Your terminal may have additional fields available for different configurations, but should be able to communicate with the Test Set if these settings are made.

Configuring a Personal Computer (PC) With HP AdvanceLink. HP AdvanceLink is a popular PC terminal emulator.. If you are using a different terminal emulator program on a PC, configure it using the previous ANSI Settings.

1. Load and run HP AdvanceLink on your PC.
2. Set the *Global Configuration* settings.

Keyboard	USASCII
Personality	HP
Language	ENGLISH
Terminal Mode	Alphanumeric
Remote To	(Enter your PC's serial port number.)
Printer I/F	None
Memory Size	32K
Plotter I/F	None
HP Mode	Yes
Video Type	(Select your display type.)
Forms Path	(Enter path if used.)
Screen Size	(Enter the size.)

3. Set the *Terminal Configuration* settings.

Terminal ID	2392A
Local Echo	OFF
CapsLock	OFF
Start Col	01
Bell	ON
XmitFnctn(A)	NO
SPOW(B)	NO
InhEolWrp(C)	NO
Line/Page(D)	LINE
InhHndShk(G)	No
Inh DC2(H)	NO
Esc Xfer(N)	YES
ASCII 8 Bits	YES
FldSeperator	US
BlkTerminator	RS
ReturnDef	CR
Copy	Fields
Type Ahead	No
ROW Size	80
Host Prompt Character	D1
Horiz. Scrolling Increment	08
Large [+] Key	+

4. Set the *Remote Configuration* settings

Baud Rate	4800
Parity/Data Bits	None/8
Eng Ack	No
Asterisk	OFF
Chk Parity	NO
SR(CH)	LO
Recv Pace	None
Xmit Pace	None
CS(CB)Xmit	No

Verifying Serial Port to Test Set Operation

1. Access the Test Set's **TESTS** screen.
2. Select `IBASIC` from the `Test Function` field to access the **IBASIC Controller** screen.
3. Position the cursor in the top left corner of the screen. (The top of the screen contains two command lines for entering commands and editing code.)
4. Type `SCRATCH`, [Enter] :**Note** - *this clears any existing programs in memory.*
5. Type `10 PRINT "HELLO WORLD"`, [Enter].
6. Type `20 END`, [Enter].
7. Press [k1] on the Test Set (or type RUN. [Enter] on your terminal) to run this two line program.
8. HELLO WORLD should be displayed on the Test Set and the terminal/PC's screen.

After the cable and adapter have been connected, and the Test Set and terminal (or PC) have been configured, you should be able to type on your terminal's keyboard and "talk" to the Test Set.

As you type each command, the letters appear on the Test Set's command lines and the terminal/PC screen. The letters appear on the terminal/PC screen because the `Inst Echo` field in the **I/O CONFIGURE** screen is set to `On`.

When the program is run, HELLO WORLD appears on the Test Set's display area and on the terminal/PC's screen because the `IBASIC Echo` field in the **I/O CONFIGURE** screen is `On`. Any non-graphic character that is printed to the Test Set's display area during a "print-to-screen" operation (CAT, LIST, PRINT,..) is also printed to the terminal/PC.

Program Control

This section describes procedures and commands for entering, editing, and moving IBASIC programs.

The Test Set uses Standard Commands for Programmable Instruments (SCPI) PROGram Subsystem commands to provide program control between an external controller and the Test Set's IBASIC controller. These commands are primarily used for the following operations:

- Downloading programs from an external controller into the Test Set.
- Uploading programs from the Test Set into an external controller's mass storage.
- Controlling IBASIC programs and executing IBASIC commands in the Test Set's IBASIC Controller from an external controller.

Executing Program Commands

The Test Set's IBASIC Controller has a PROGgram subsystem it uses to communicate with other computers over HP-IB. When sending a command to the IBASIC Controller from another computer, you must use a 'PROG' command.

You can use PROGram EXECute commands to display and edit programs in the Test Set using an external controller. This eliminates using the cursor control knob, providing a more efficient way of making small changes to your programs. The full range of IBASIC program commands can be executed from a connected controller.

The following operations are given as examples. 'Addr' is the address of the Test Set, and '<filename>' represents the name of a file you are saving or retrieving.

Entering or Editing a Program Line

Program lines in the Test Set's RAM can be entered and edited one line at a time from your computer using the PROG command -

```
OUTPUT Addr;"PROG:EXEC '<program line number/command> '"
```

with <program/command> representing any command or program line you want to enter.

For example, to enter or change line 20 of a program to '20 A=3.14', you would enter the following command on your computer

```
OUTPUT Addr;"PROG:EXEC '20 A=3.14 '"
```

Quoted strings, such as those used in PRINT commands, must use double quotes. Example -

```
OUTPUT Addr;"PROG:EXEC '30 PRINT ""TEST"" '"
```

Listing A Program

Enter the following command on your computer to list a program that has been loaded into the Test Set:

```
OUTPUT 714;"PROG:EXEC 'LIST'"
```

Downloading A Program Into the Test Set

The following procedure uses the PROGram Subsystem commands to transfer a BASIC program from your computer to the Test Set. This procedure assumes the Test Set's HP-IB address is set to 14.

1. Access the Test Set's **IBASIC Controller** screen.
2. Enter a program into your computer. (Use the sample program below if this is your first download. This program clears the Test Set's IBASIC Controller display area, and tells you the downloading procedure worked).

```
10   !THIS IS A SAMPLE PROGRAM
20   CLEAR SCREEN
30   PRINT "DOWNLOADING COMPLETED"
40   END
```

3. Execute the following commands on your computer:

```
OUTPUT 714;"PROG:DEL"
OUTPUT 714;"PROG:DEF #0"
LIST #714
OUTPUT 714;" "END
```

To verify that your code was down-loaded, type the following command:

```
OUTPUT 714;"PROG:EXEC 'LIST '"
```

Your program should be listed on the Test Set's **IBASIC Controller** screen.

You can now run your program directly by selecting the Run field on the **IBASIC Controller** screen.

Uploading a Program From the Test Set

The following IBASIC program copies a program from your Test Set's IBASIC Controller RAM to any mass storage device assigned by your controller.

When the upload program is loaded and run on your controller, you are prompted to provide a file name for the your program to be uploaded and saved. As the upload program is running, the total number of characters in your program, and the number of characters transferred, are displayed.

If you get an 'end of file' or similar error message while you are running the upload program, change the number of 256-byte records in line 50 of the upload program (the current setting is 10). The number of records required is approximately equal to the number of characters counted divided by 256.

Note

The following upload program is for 'stand-alone' programs, and is not intended for programs that are written to use the TESTS Subsystem (other than the **IBASIC Controller** screen). Programs written for the TESTS Subsystem require the creation of supporting Library and Procedure files, and must be written using a specific program structure. The HP 11807A Radio Test Software packages are examples of this type of program.

If you are creating or modifying tests that use the TESTS Subsystem, you use the Program Development Tools disk. The disk helps you create and edit programs.

```
10  !Upload the program in the Test Set to an external controller.
20  Addr=714                ! Test Set address
30  ASSIGN @I TO Addr
40  INPUT "Name of (ASCII) host file to create?",File_name$
50  CREATE ASCII File_name$,10
60  ASSIGN @File TO File_name$
70  OUTPUT @I;"PROG:DEF?"
80  ENTER @I USING "X,D,#";Count_len !Number of chars in char count
90  ENTER @I USING VAL$(Count_len)&"D,#";Chars_total
100 Char_cnt=Chars_total    !Characters of program data left to get
110 DIM Line$[200]
120 WHILE (Char_cnt>0)
130 ENTER @I;Line$
140 OUTPUT @File;Line$
150 Char_cnt=Char_cnt-(LEN(Line$)+2)  !CR/LF counted total, not LEN
160 DISP Chars_total-Char_cnt;"of";Chars_total;"characters
transferred."
170 END WHILE
180 ENTER @I;Line$   ! Clean up last <LF> to correctly terminate
190 END
```

Saving Programs To Memory Cards

This procedure saves a program from the IBASIC Controller's RAM memory to a memory card.

1. Press (LOCAL), (SHIFT), (CANCEL) on the Test Set to perform an IBASIC reset.
2. If your memory card has not been initialized, insert your memory card into the Test Set and enter the following command on your computer:

```
OUTPUT 714;"PROG:EXEC 'INITIALIZE"":INTERNAL""'"
```

3. Insert the initialized memory card into the Test Set.
4. Define the memory card as the Mass Storage device by entering the following command on your computer:

```
OUTPUT 714;"PROG:EXEC 'MSI "":INTERNAL""'"
```

5. Save your program to the memory card by entering the following command on your computer:

```
OUTPUT 714;"PROG:EXEC 'SAVE ""<filename>""'"
```

6. Press (LOCAL).

PROG Subsystem Commands

See the Program syntax diagram in chapter 3 for details of the PROG subsystem.

Command Notation

The following notation is used in the command descriptions:

[] = Optional keyword; this is the default state.

<> = Indicates specific SCPI parameter forms.

{} = Indicates one or more parameters may be included zero or more times.

| = Separates choices for a parameter. To be read the same as "or".

Commands

The following command descriptions and PROGram syntax diagrams are intended to provide general information on the SCPI commands that can be used with the Test Set. If you are not familiar with SCPI, it is recommended that you obtain a copy of the book —*A Beginner's Guide to SCPI* (ISBN 0-201-56350, Addison-Wesley Publishing Company).

:CATalog?. The CATalog query command lists all defined programs. The response is a list of comma-separated strings. Each string contains the name of a program. If no programs are currently defined, the response is a null string (" ").

SELected Commands. All the commands in this group access only the program that has been selected using the NAME command.

DEFine <program>. The DEFine command is used to create and download programs. The DEFine query is used to upload programs. The program name used for the definition is the currently selected program name.

With this command, the selected program name must be unique. If the same program name exists, you must delete it before trying to download and overwrite a new program with the same name. An error occurs if you try to define a second program.

<program> is an IBASIC program sent as IEEE 488.2 block data. Using the DEFine query, the selected program is uploaded as definite length arbitrary block response data.

DELete. This command deletes downloaded programs.

- :SELected causes the selected program to be deleted.
- :ALL specifies that all programs in IBASIC RAM are to be deleted. If any of the programs are in the RUN state, an "IBASIC running" error is generated and no program is deleted.

EXEcute <program_command>. This command executes program commands in the IBASIC Controller. <program_command> is string data representing any legal program command.

MALLocate <nbytes>|DEFault. The MEMory ALLocate command reserves memory space within the HP Test Set for run-time subroutine stack and DIM statements. COM statements are not affected.

If DEFault is specified, the instrument calculates and reserves the required memory space, although the amount reserved may greatly exceed what your program needs. <nbytes> is numeric data representing the required memory in bytes.

This command cannot be sent while a program is running.
*RST has no effect on the value of MALLocate.

NAME <progname>. This command defines the name of the program to be selected. If that program name already exists, then the existing program is selected. If the program name does not exist, then the new name is selected but no program is defined.

*RST causes the selected name to become "PROG".

NUMber <varname>{,<nvalues>}. This command is used to set and query the contents of numeric program variables and arrays in the currently selected program. The selected program must be DEFined, or an error occurs.

<varname> is the name of an existing variable in the selected program, and can be either character data or string data.

<nvalues> is a list of comma-separated numeric values which are used to set <varname>. An error occurs if the specified variable cannot hold all of the specified numeric values.

The query form NUMBer? <varname> returns the contents of the variable as a comma-separated list.

STATe RUN|PAUSe|STOP|CONTinue. This command sets or queries the state of the currently selected program. The following table indicates the effect of sending a STATe command when a program is in a different state.

Table 6-3. Effects of STATe Commands

Command Sent	Current State		
	RUNNING	PAUSED	STOPPED
RUN	error	RUNNING	RUNNING
CONT	error	RUNNING	error
PAUSE	PAUSED	PAUSED	STOPPED

STRing <varname>{,<svalues>}. This command is used to set and query contents of string program variables and arrays in the currently-selected program. An error occurs if the selected program has not been DEFined.

The variable <varname> must be the name of an existing variable in the selected program. <varname> can be either string or character data. An error occurs if the specified variable cannot hold all of the specified string values. If a string value is too long, it is truncated when stored in the program's variable.

The STRing? <varname> query command returns the contents of the variable as a comma-separated list.

WAIT <progname>. This command and its query form stops further commands or queries from being executed until the selected program exits the RUN STATe, or is either PAUSed or STOPped.

For the query form, a "1" is returned over HP-IB when the program is either STOPped or PAUSed.

EXPLicit Commands. All the commands under the EXPLicit node directly reference the desired program by name, allowing access to a program without having to change the selected program. The <progname> parameter is required for these commands.

DEFine <progname>,<program>. This command is used to create and download programs. The DEFine query is used to upload programs.

With this command, the selected program name must be unique. If the same program name exists, you must delete it before trying to download and overwrite a new program with the same name. An error occurs if you try to define a second program.

<program> is an IBASIC program sent as IEEE 488.2 block data. Using the DEFine query, the specified program name must be the name of an existing program. The <program> is uploaded as definite length arbitrary block response data.

DELete <progname>. This command deletes the specified downloaded program.

EXECute <progname>,<program_command>

This command executes program commands in the Test Set's IBASIC Controller. <program_command> is string data representing any legal program command.

MALLocate <progname>,(<nbytes>|DEFault). The MEMory ALLocate command reserves memory space within the HP Test Set for run-time subroutine stack and DIM statements. COM statements are not affected.

If DEFault is specified, the instrument calculates and reserves the required memory space, although the amount reserved may greatly exceed what your program needs. <nbytes> is numeric data representing the required memory in bytes.

This command cannot be sent while a program is running.
*RST has no effect on the value of MALLocate.

NUMBer <progname>,<varname>{,<nvalues>}. This command sets and queries contents of numeric program variables and arrays in the specified program. The specified program must be DEFined, or an error occurs.

The variable specified in <varname> is the name of an existing variable in the specified program. <varname> can be either character data or string data.

<nvalues> is a list of comma-separated numeric values used to set <varname>. An error occurs if the specified variable cannot hold all of the specified numeric values.

The query for NUMBer? <varname> returns the contents of the variable as a comma-separated list.

STATe <progname>,(RUN|PAUSe|STOP|CONTinue). The State command sets or queries the state of the currently-specified program. The following table indicates the effect of setting the STATe to the desired value from each of the possible current states.

Table 6-4. Effects of STATe Commands

Command Sent	Current State		
	RUNNING	PAUSED	STOPPED
RUN	error	RUNNING	RUNNING
CONT	error	RUNNING	error
PAUSE	PAUSED	PAUSED	STOPPED
STOP	STOPPED	STOPPED	STOPPED

STRing <progname>,<varname>{,<svalues>}. The String command is used to set and query contents of string program variables and arrays in the specified program. An error occurs if the specified program has not been DEFined.

The variable <varname> must be the name of an existing variable in the specified program. <varname> can be either string or character data. <svalues> is a list of comma-separated strings that set <varname>. An error occurs if the specified variable cannot hold all of the specified string values. If a string value is too long, it is truncated when stored in the programs variable.

The STRing? <varname> query command returns the contents of the variable as a comma-separated list.

WAIT. The Wait command stops further commands or queries from being executed until the selected program exits the RUN state, or is either PAUSed or STOPped.

For the query form, a "1" is returned over HP-IB when the program is either STOPped or PAUSed.

Writing Programs For the TESTS Subsystem

This section describes the concepts and tasks associated with the TESTS subsystem. It is intended to help the experienced programmer develop programs or modify existing programs such as the HP 11807A Radio Test Software.

When Should I Use the TESTS Subsystem?

Programs that do not use the TESTS subsystem have the following limitations:

- Tests are always run in the same order, unless you write your program to allow change (as opposed to choosing the test order before testing using the TESTS subsystem).
- Different test procedures cannot be created from the same program without modifying the test program and re-storing it; (as opposed to using the TESTS subsystem to create and store different procedures from the same program).
- If your program compares measured values to your specifications, you must either use the same specifications every time you run the test, or write your program to allow specification changes each time the test is run; (as opposed to using the TESTS subsystem to change specifications and store them with the associated procedure for future use).
- If you want to be able to change instrument settings used in a test (frequencies, amplitudes, filters,..etc), you must either change the test program's variables directly, or provide for operator interaction during the test; (as opposed to changing these parameters before testing using the TESTS subsystem).
- Programs must be loaded and run directly from the **IBASIC Controller** screen, instead of using the **TESTS** screen.

Unlike programs designed to run using the TESTS subsystem, programs that *don't* take advantage of the TESTS subsystem do not have to be structured in any pre-defined manner.

The TESTS subsystem's capabilities were designed to allow the operator to "pick and choose" the tests and parameters they need from a larger set, eliminating unnecessary tests and reducing test time. This is especially helpful when a very large program has been written containing several tests and a multitude of associated specifications, test parameters and frequencies. HP 11807 Radio Test Software packages operate this way.

Writing programs to run in this environment requires you to understand and adhere to the program structure and syntax required by the TESTS subsystem.

TESTS Subsystem File Descriptions

Three types of files are used in the TESTS subsystem to store different types of information.

Code Files

The first aspect of an automated definition is the code itself. This is just a standard IBASIC Code file that can reside either on the Memory card, on an external disk drive connected to the HP-IB port of the Test Set, or in an internal RAM disk. The name of this file is preceded by a lower case 'c'. This tells the TESTS subsystem that this particular file contains program code.

Library Files

A Library indicates all of the available test subroutines in the code, the set of all parameters that might be entered using the user-interface screens, and all specifications that might be used by the subroutines in the code to decide if a test point passes or fails.

Only one Library is defined for each Code file. The name of this file is preceded by a lower case 'l', telling the TESTS system that this is a Library file. Also, both the Library and Code file should have the same base name to indicate the relationship between them.

A Library is required if the you want to use the user-interface screen functions of the TESTS subsystem. If the program is simple enough that there is no need for user-input, or if all the user-input is simple enough to be accomplished through INPUT statements, then a [NO LIB] option is available.

Procedure Files

A Procedure allows the user to define which of the test subroutines, parameters, and specifications defined in the Library will be used to test a specific Radio. There may be many Procedures defined that use the same IBASIC Code and Library, each using a different subset of the choices available in the Library. These files are preceded with a lower case 'p', but are *not* required to have the same base name as either the Library or the Code. The name of the corresponding Library (if any) is stored in each Procedure file.

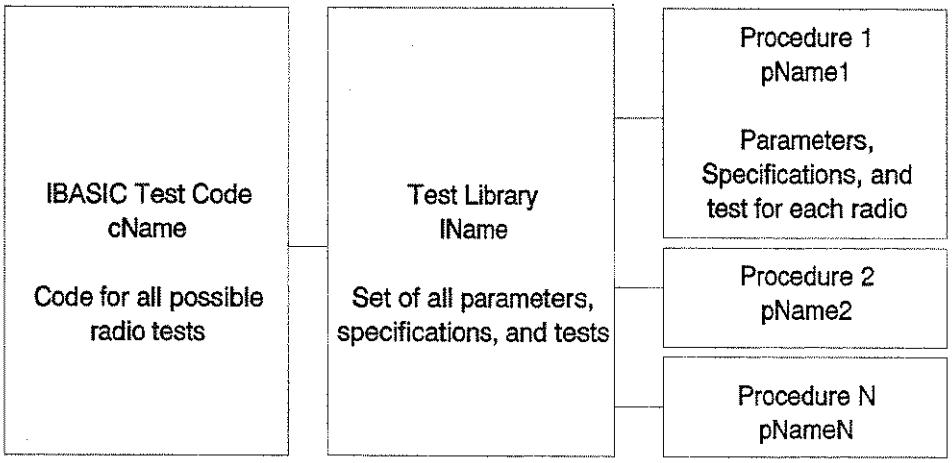


Figure 6-2. TESTS Subsystem File Relationship

TESTS Subsystem Screens

The TESTS subsystem uses several screens to create, select, and copy files, and to run tests.

The Main TESTS Subsystem Screen

Refer to Figure 6-3.

The Main TESTS screen is accessed by pressing the front panel (TESTS) key. Notice that the first line shows the currently selected Procedure. The associated Library is listed, as well as the location of the code.

The comment area is simply available to give the user a more complete explanation of this particular Procedure.

To view all the Procedures available on the selected media, simply select the `Procedure` field. A menu will appear in the lower right corner of the screen, displaying all the Procedures are available. This is not a listing of the full contents of the disk; it is only a list of the Procedures that have been stored.

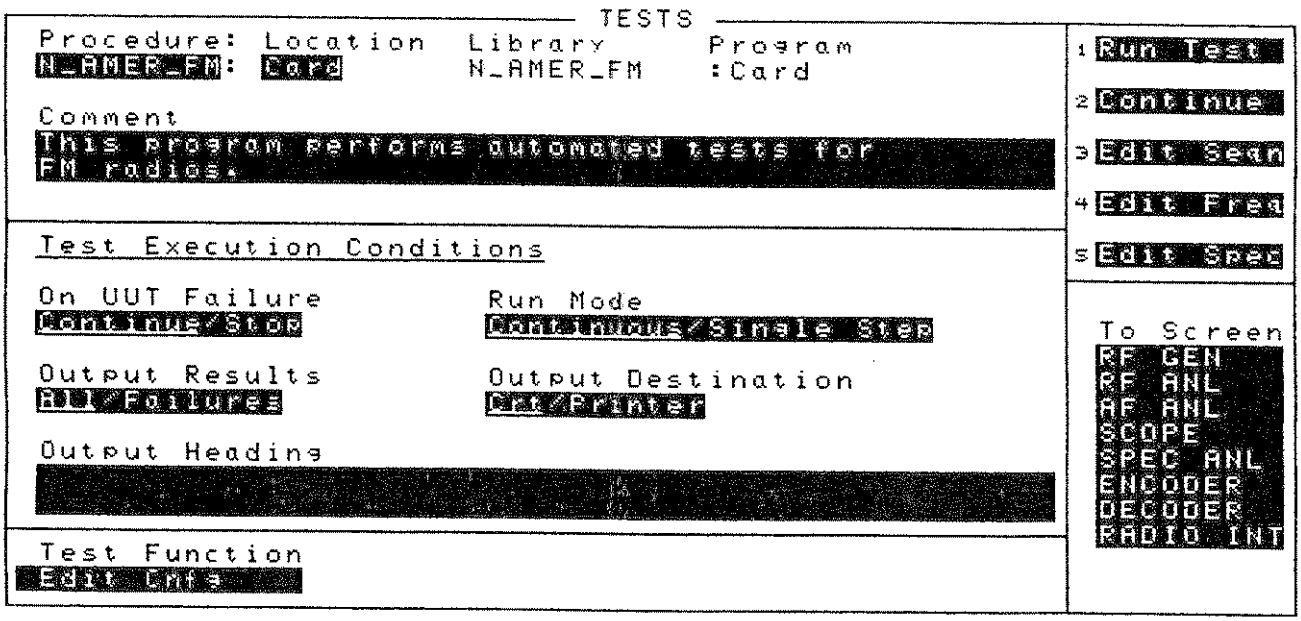


Figure 6-3. The Main TESTS Subsystem Screen

TESTS Subsystem User-Interface Screens

The TESTS subsystem allows the user to easily modify the test subroutines, parameters, specifications and configuration to correspond to the requirements of a specific Radio. There are several user-interface screens that allow the user to do this.

To access any of these screens, select the `Test Function` field at the bottom of the main TESTS screen to display the screen choices.

- The *Edit Sequence* screen lets you select the desired test(s) from the full set of available tests in the default Procedure file.
- The *Edit Frequencies* screen defines the transmit and receive frequencies used for the selected tests.
- The *Edit Specifications* screen defines the specifications used to generate pass/fail messages during testing.
- The *Edit Parameters* screen is used to define instrument settings and characteristics to match those of the radio being tested (audio load impedance, audio power, power supply voltage,..etc.).
- The *Edit Configuration* screen identifies all connected HP-IB equipped instruments and their HP-IB addresses.
- The *Procedure Manager* screen is used to make or delete Procedures.

Refer to the **TESTS** screen description in chapter 4 of the User's Guide, for information concerning how the different TESTS subsystem screens are used.

The use of the *IBASIC Controller* screen is also described in chapter 5 of this manual.

Program Structure for TESTS Subsystem Programs

Writing programs that take advantage of the TESTS subsystem capabilities requires the programmer to understand how to structure the program to access the TESTS subsystem user-interface screens.

General Organization

Here are the steps to a basic algorithm that can be used to execute a number of test subroutines at a number of different frequencies:

```
BEGIN
SET UP (Set up the COM area to hold the global variables.)
REPEAT (for all Test Frequencies)
  REPEAT (for all Defined Tests)
    DO SUBROUTINE (defined Test)
  UNTIL (All Defined Tests Done)
UNTIL (All Test Frequencies Tested)
END
SUBROUTINE1 (Defined Test 1)
SUBROUTINE2 (Defined Test 2)
SUBROUTINE3 (Defined Test 3)
```

Program Example

The following example IBASIC program uses the basic algorithm shown above and the TESTS subsystem to execute a number of test subroutines at a number of defined test frequencies. Also included are examples of how to interact with the user-interface to allow a user to access parameters, specifications, and configuration fields to define a specific set of test requirements.

An explanation of the program example is given at the end of the listing.

Program Listing

```
10    ! DEMO_1
20    !
30    ! THE FIRST LINE MUST CONTAIN THE NAME OF THE LIBRARY
40    !
50    !-------------------------------------------------------
60    !
70    ! THIS PROGRAM IS A DEMO PROGRAM TO DEMONSTRATE THE USE
80    ! OF THE TEST SUBSYSTEM ON THE Test Set
90    !
100   ! REVISION: 1 APRIL, 1991
110   !-------------------------------------------------------
120   !
130   COM /I_o/ I_o$[470]
140   ! INPUT OUTPUT STRING
150   COM /Freq/ Rx_f,Tx_f
160   ! PRESENT RX AND TX FREQUENCIES IN MHZ
170   !
180   INTEGER Test_return
190   !  TITLE SCREEN FOR OUR TESTS
200   CLEAR SCREEN
210   PRINT TABXY(2,2),"___DEMO PROGRAM FOR THE TESTS SUBSYSTEM___"
220   !
230   !  SET UP A SOFT KEY TO HALT THE PROGRAM
240   ON KEY 1 LABEL "Stop Test",5 GOTO Stp_test
250   !
260   !  CLEAR THE INTERNAL Test Set BUS
270   CLEAR 800
280   !
```

```
290   ! NOW READ THE TEST FREQUENCIES IN ONE AT A TIME AND DO THE
300   ! SEQUENCE OF TESTS ON THEM
310   Ch=1
320   REPEAT
330     OUTPUT 800;"TESTS:FREQ? "&VAL$(Ch)
340     I_o$=""
350     ENTER 800;I_o$
360     !  SET THE VALUE OF THE RX FREQUENCY
370     Rx_f=VAL(I_o$[4;12])
380     !  SET THE VALUE OF THE TX FREQUENCY
390     Tx_f=VAL(I_o$[30;12])
400     !  SET WHETHER TO TEST THIS FREQUENCY
410     T_it$=I_o$[56;1]
420     !  SET IF THIS IS A PRIME FREQUENCY
430     IF (LEN(I_o$)>57) THEN
440       Prime$=I_o$[58;1]
450     ELSE
460       Prime$="N"
470     END IF
480     !  IF THIS FREQUENCY IS TO BE TESTED
490     IF T_it$="Y" THEN
500       PRINT TABXY(2,6),"RX FREQUENCY = ",Rx_f
510       PRINT TABXY(2,7),"TX FREQUENCY = ",Tx_f
520       PRINT TABXY(2,8),"TEST THIS FREQUENCY ?",T_it$
530       Run_ts=1
540       !  RUN THROUGH THE SEQUENCE OF TESTS
550       REPEAT
560         Done_t=0
570         !  ENTER IN THE TEST SEQUENCE
580         OUTPUT 800;"TESTS:SEQN? "&VAL$(Run_ts)
590         I_o$=""
600         ENTER 800;I_o$
610         Tst=VAL(I_o$[4;2])
620         !  IF THIS TEST IS TO BE SKIPPED THEN SET THIS
630         IF I_o$[7;1]="N" THEN Tst=-Tst
640         !  IF THIS IS A PRIME FREQUENCY RUN THE TEST
650         IF Tst<0 AND Prime$="Y" THEN
660         !  CALLS THE SUBROUTINE NAME T(ABS(Tst))
670           T(ABS(Tst),Test_return)
680           IF (Test_return=1) THEN GOTO Test_error
690           Done_t=1
700         END IF
```

```
710         !  IF THIS TEST IS TO BE DONE AND IS NOT A PRIME FREQUENCY
720         IF Tst>0 AND NOT Done_t THEN
730         !  CALLS THE SUBROUTINE NAME T(ABS(Tst))
740           T(ABS(Tst),Test_return)
750           IF (Test_return=1) THEN GOTO Test_error
760         END IF
770         Run_ts=Run_ts+1
780       UNTIL Tst=0 OR Run_ts=51
790     END IF
800     Ch=Ch+1
810   UNTIL Ch=51 OR Tx_f=-1 OR Rx_f=-1
820 Stp_test:  !
830   CLEAR SCREEN
840   PRINT TABXY(2,10),"FINISHED TESTING"
850   GOTO End_program
860 Test_error:  !
870   CLEAR SCREEN
880   PRINT TABXY(2,10),"PROGRAM STOPPED, TEST ",ABS(Tst),"FAILED"
890 End_program:  !
900   END
910 T01:SUB T01(Test_return)
920     COM /I_o/ I_o$
930     COM /Freq/ Rx_f,Tx_f
940     DIM Calling_name$[22],Model$[22],Options$[22]
950   !  TEST ROUTINE NUMBER 1
960     PRINT TABXY(2,12),"DOING TEST NUMBER 1 FOR FREQ ",Rx_f
970     !  GET THE PARAMETER 1 FOR THIS TEST
980     OUTPUT 800;"TESTS:PARM? "&VAL$(1)
990     I_o$=""
1000    ENTER 800;I_o$
1010    !  IF THERE IS NO PARAMETER THEN PAUSE
1020    IF I_o$[1;5]="Error" THEN
1030      PRINT TABXY(2,14),"ERROR IN RECALLING THE PARAMETERS FOR TEST 1"
1040      Test_return=1
1050    END IF
```

```
1060    Parm_1=VAL(I_o$)
1070    !  GET CONFIGURATION 1 INFO FOR THIS TEST
1080    OUTPUT 800;"TESTS:CONF? "&VAL$(1)
1090    I_o$=""
1100    ENTER 800;I_o$
1110    Calling_name$=I_o$[4;21]
1120    Model$=I_o$[27;21]
1130    I1addr=VAL(TRIM$(I_o$[50]))
1140    Options$=I_o$[54]
1150    !  GET SPECIFICATION 1 FOR THIS TEST
1160    OUTPUT 800;"TESTS:SPEC? "&VAL$(1)
1170    I_o$=""
1180    ENTER 800;I_o$
1190    IF I_o$[1;5]="Error" THEN
1200      PRINT TABXY(2,14),"ERROR IN RECALLING THE SPECIFICATIONS FOR TEST 1"
1210      Test_return=1
1220    END IF
1230    Lower_limit=VAL(TRIM$(I_o$[4]))
1240    Upper_limit=VAL(TRIM$(I_o$[17]))
1250    Test$=TRIM$(I_o$[30])
1260  SUBEND
1270 T02:SUB T02(Test_return)
1280    COM /I_o/ I_o$
1290    COM /Freq/ Rx_f,Tx_f
1300  !  TEST ROUTINE NUMBER 2
1310    PRINT TABXY(2,13),"DOING TEST NUMBER 2 FOR FREQ ",Rx_f
1320  SUBEND
1330 T03:SUB T03(Test_return)
1340    COM /I_o/ I_o$
1350    COM /Freq/ Rx_f,Tx_f
1360  !  TEST ROUTINE NUMBER 3
1370    PRINT TABXY(2,14),"DOING TEST NUMBER 3 FOR FREQ ",Rx_f
1380  SUBEND
```

```
1390 T:SUB T(N,Test_return)
1400    ! CALL THE PASSED TEST NUMBER (N)
1410    SELECT N
1420    CASE 1
1430      T01(Test_return)
1440    CASE 2
1450      T02(Test_return)
1460    CASE 3
1470      T03(Test_return)
           ""
           ""
           ""
2380    CASE 49
2390      T49(Test_return)
2400    CASE 50
2410      T50(Test_return)
2420    END SELECT
2430  SUBEND
```

Program Listing Explanation

10: This first line must contain the name of the Library and the program. This is checked by the TESTS subsystem when loading the program.

130: Establish a common I_o$ string for the ENTER statements.

150: Establish common Rx_f and Tx_f that can be used by the subprograms (tests).

180: The Integer Test_return is used by the subprograms to indicate the test ended with some error condition. The meaning of Test_return could be expanded to include the status of the test (ie PASS/FAIL).

200: Clears the IBASIC Screen.

210: Prints and indication that the Demo program is running.

240: Allows the User to stop the program using a softkey.

270: Clear the Internal Bus of the Test Set

310: Ch keeps track of which channel we are currently testing.

320: Now Repeat for all Frequencies:

330: Request all the channel values from the Test Set.

340: I_o$ gets the string return.

370: The Rx frequency is pulled from the string.

390: The Tx frequency is pulled from the string.

410: T_it$ gets either a "Y" or an "N" depending on whether this frequency is to be tested.

430: If a Prime channel has been specified then Prime$ gets a value of "Y".

490: If this frequency is to be tested:

500-520: Print out some information on the test about to be performed.

530: Run_ts holds the value of the test currently being run.

550: Repeat for all Specified Tests:

560: Done_t is initialized to not completed.

580: Get the Test specifier for the current Test.

590: Initialize I_o$ to a null string.

600: I_o$ holds the value of the return string.

610: Tst now hold the value of the current test. This value is equal to the index of the Test Name in the Test selection list shown on the Test Seqn screen.

630: This tests whether this test is to be run for all channels. If not, the value is still kept around but is made negative. This will be used in later tests.

650: If the number of the test is indeed negative but the channel is prime, then the test is done.

670: This calls a subroutine that maps the number of the test with the subroutine that defines this test.

680: If there is an error, then the program stops and the error is reported.

690: Done_t is set to completed.

700: End this IF statement.

720: If Tst is suppose to be done, and has not yet been done, then now do it.

740: Again, This calls a subroutine that maps the number of the test with the subroutine that defines this test.

750: If there is an error, then the program stops and the error is reported.

760: End this IF statement.

770: Increment the step for the Test index.

780: If there are no more steps specified, or if the number of tests run is 51, then leave the test seqn loop.

790: End the Tst IF statement.

800: Increment the Channel number.

810: Stop stepping through the channels if the number of channels reaches 51, or if the Receive or Transmit frequencies are specified at −1.

820: The goto location for the stop test softkey.

830: Clear the screen

840: Indicate that the test is finished.

850: Goto the end statement.

860: The goto location if an error occurs in one of the subroutines.

870: Clear the screen.

880: Indicate that one of the tests have failed.

890: The goto for the end of the program.

900: End of the main program.

910: Subroutine T01-This corresponds with test #1. This subroutine illustrates how to enter values from the Parameters, Configuration, and Specification screens.

920-930: Includes the common variables.

940: Dimension some variables that will be used to store values from the configure screen.

960: Indicate that the first test is now active.

980: Enter the value of the first Parameter. This is the value of the first parameter on the Parameter Screen.

990: Initialize the I_o$ string.

1000: Enter the value.

1020-1050: If there is no defined parameter this string will catch the error and return it to the main program.

1080: Get the information for the first instrument stored on the configure screen.

1090: Initialize the I_o$ string.

1100: Enter the string.

1110: Calling_name$ now holds the string associated with the Calling Name field on the configure screen.

1120: Model$ now holds the string associated with the Model field on the configure screen.

1130: Iladdr equals the value in the Addr field on the configure screen.

1140: Options$ now holds the string associated with the Options field on the configure screen.

1160: Get the information for the first Specification listed on the Specification system.

1170: Initialize the I_o$ string to null.

1180: ENTER the I_o$ string.

1190-1220: If there is no specification defined for this specification number, then an Error will appear in the I_o$ string. If this occurs, stop the test and return the error to the main program.

1230: Set the lower limit from the value in the string.

1240: Set the upper limit from the value in the string.

1250: Set Test$ to whether "Upper", "Lower", "Both", or "None" of the specs are to be tested.

1260: End of this subroutine.

1270-1380: These are the second and third subroutines. They are labeled T02 and T03 to correspond with the second and third test routines defined on the Test Seqn screen.

1390-2430: SUB T maps the calls from the main program to the correct subroutine. The mapping is quite simple, with the main program specifying which test to run and this subroutine calling the correct subroutine based on the SELECT statement.

Creating A Library And Default Procedure File

Once the Code file has been created, an associated Library and default Procedure file for the Code file can also be created. Both of these files are BDAT files and are created using the **DEV_PL** program on the *Program Development Tool* disk shipped with this manual. The files on this disk are described later in this section.

Creating A Procedure File With No Library

If you do not want your program to use the different user-interface screens of the TESTS subsystem, you can create a Procedure from your Code file that does not have a Library associated with it. This is done using the **DEV_PL** program on the *Program Development Tool* disk shipped with this manual. When the test information is defined, [NO LIB] is selected for the Library Name.

When creating a procedure to run without a Library, the first line of your Code file must be an exclamation point followed by the Code file name. For example, if your procedure is called 'FM_TESTS' the first line of your Code file **must** be-

```
1    ! FM_TESTS
```

Using the Software Development Tools

The *Software Development Tool* disk contains a development program and example files to help you create tests to run in the TESTS subsystem environment.

Code, Library, and Procedure files can be created using the development program.

Several macro-level functions are available for file transfer and storage to eliminate line-by-line command entries.

Development Disk File Descriptions

DEV_PL is the main program for interfacing with the Test Set, external disk drives, printers; you use it when developing Code, Library, and Procedure files used with the TESTS Environment on the Test Set.

cDEMO is a demo test that makes use of the Library and Procedure parameters in the lDEMO and pDEMO_T files. This program contains sub-programs that are useful when developing other test code.

lDEMO is a BDAT file containing a demo set of library parameters.

pDEMO_T is a BDAT file containing a demo set of procedure parameters are used with the library file lDEMO.

INST_C_9 is a BDAT file containing the instrument names, model numbers, and addresses for the Test Set and the Printer (if used) that are used by the DEV_PL program.

MASS_S_9 is a BDAT file containing the locations of the external disk drives used by the DEV_PL program.

LINK is a program that the DEV_PL program sends to the Test Set when sending data to or receiving data from the Test Set. This data is used to create the Procedure and Library files on the Test Set. This program is not used when sending or receiving code to or from the Test Set.

STRT_DEV is the program that is loaded when the "Setup To Develop Code" function is selected in the DEV_PL program. The delivered code in this file is only an "END" statement. You can store the Code you are developing under this name for automatic loading when "Setup To Develop Code" is pressed in DEV_PL.

*_T_* is a temporary ASCII file that the code is placed in when it is received from the Test Set, and before it is loaded into the external controller. This file will be reused every time code is received from the Test Set.

Loading and Configuring the Development Software

1. Insert the *Software Development Tool* disk in your disk drive, and type `LOAD "DEV_PL:<location>"` and press ENTER.
2. After the program has loaded, type `RUN` and press ENTER to display the main menu.
3. Use your computer's ↑ ↓ keys and the (Select) softkey to select the **Configure System** function from the list at the right side of the menu.
4. Enter the required system information, using the examples listed on each screen. As each type of information is input, answer 'Yes' when prompted to store the new information on disk. This allows the system to use this information the next time the program is run, eliminating re-entry.
5. When you have entered all necessary system information, select **Done** to return to the main menu.

How to Use the Development Software

The main menu is displayed when the software is loaded. As each function from the right side of the screen is highlighted, a corresponding **Functional Description** is displayed. After you have configured the software, the order you use the functions depends on what you are doing.

Suggested Development Sequence

1. Insert an initialized memory card into your Test Set that contains enough memory to store your Code, Library, and Procedure files.
2. Use the **Setup To Develop Code** function to create your Code file and save it to a memory card. Remember to use the 'c' prefix when saving the file to identify it as a Code file.
3. Use the **Define Test Information** function to create a Library and Procedure file for your Code file. The functions associated with this screen should be performed in order, starting with **Define Library Name**, and ending with **Store Test Data to Disk**. Remember to use the same name for the Library file that you use for your Code file. (The Procedure file does not have to have the same name.)
4. Use the **Send Data To Test Set** function to transfer the Library and Procedure files you just created to the Test Set.

Your Code, Library, and Procedure files should now exist on your Test Set's memory card. To see if they are, use the **Test Set Command** function and enter the CAT command to list your memory card's contents on the Test Set's screen.

Other Functions

Load Test Data From Disk is used to retrieve existing Procedure and Library files from a disk drive. The information can then be edited and re-stored.

Rcv Data From Test Set is used to retrieve Library and Procedure files from a memory card. The information can then be edited and re-stored.

Rcv Code From Test Set is used to retrieve Code loaded in the Test Set's RAM. The information can then be edited and re-stored.

Print Test Data prints a listing of all Library and Procedure information for the currently loaded files.

Index