



SIGNAL FORGE 1010™
PORTABLE SIGNAL GENERATOR WITH REMOTE CONTROL

Programmer's Manual



Technical Support

Email: Support@signalforge.com

Phone: 512.275.3733

Contact Information

Web: www.signalforge.com

Customer Service Email: Sales@signalforge.com

Phone: 512.275.3733

Fax: 512.275.3735

Address: Signal Forge, LLC ▪ 2115 Saratoga Drive ▪ Austin TX 78733

Table of Contents

INTRODUCTION	6
Programming Interface.....	6
SCPI Conformance Information.....	6
INTERFACE SPECIFICATIONS	7
Firmware Modulation Accuracy.....	8
Parameter Significance and Accuracy.....	8
GETTING STARTED	9
Power Adapter.....	9
Connecting the Signal Generator to Your Computer.....	9
Programming Quick Facts.....	9
WAVEFORMS	11
Single Tone.....	11
ASK.....	11
BPSK.....	12
Chirp.....	13
FSK Unramped.....	13
FSK Ramped.....	14
FSK Triangle.....	14
OOK.....	15
Sweep.....	16
Arbitrary.....	16
SCPI SYNTAX	17
Command Message.....	17
Query Message.....	18
Data Parameters.....	18
Notational Conventions.....	18
General Notations.....	18
Parameter Notations.....	19
PROTOCOL	20
Responses.....	20
Suggested Programming.....	21
Query Version of Commands.....	21
Responses to Queries.....	21
ASCII <CANCEL> Command.....	22
POWER-UP	23
Saved and Sticky Data.....	23
BOOT BLOCK	24
Boot Block Commands.....	24
Downloading.....	25
SCPI COMMANDS	26
Common Commands.....	26
Mandated Common Commands.....	26
Optional Common Commands.....	28
CALibration Subsystem (CALibration).....	29
Calibration Procedure.....	30
OUTPut Subsystem.....	31
SFORge Subsystem.....	32
AM Subsystem.....	34
CM Subsystem.....	35
Non-Triggered Operation.....	36

Triggered Operation.....	36
Chirp Operation for High Frequencies.....	37
FM Subsystem.....	38
FREQuency Subsystem.....	39
LIST Subsystem.....	41
LIST Phases.....	42
Run Phase.....	43
Programming LIST.....	43
LIST Commands.....	44
Consideration for POWER and FREQuency components.....	47
OM Subsystem.....	48
PM Subsystem.....	49
POWER Subsystem.....	50
SWEep Subsystem.....	50
Sweep Considerations.....	53
VOLTage Subsystem.....	53
STATus Subsystem.....	53
Status System Programming.....	54
Status Group Reporting.....	55
Instrument Summary Status Group.....	55
Standard Event Status Group.....	56
Operational Status Group.....	56
Questionable Status Group.....	57
STATus Subsystem Commands.....	57
SYSTEM subsystem.....	59
TRIGger Subsystem.....	60
Trigger System Programming.....	60
TRIGger Command Descriptions.....	61
UNIT Subsystem.....	62
ERROR MESSAGES	63
Introduction.....	63
Error Queue.....	63
Error Query.....	63
Error Codes.....	64
Command Errors.....	64
Execution Errors.....	65
Device-Specific Errors.....	66
Query Errors.....	66
Operation Complete Event.....	67
Signal Forge-Specific Errors.....	67
SCPI Command Parser Errors.....	67
User Programming Errors.....	67
Internal Programming Errors.....	68
PROGRAMMING	69
Handling Errors.....	69
Suggested Error Handling.....	69
Initialization and Reset.....	69
Suggested Programming Sequence.....	70
Controlling Outputs.....	71
Common Operations.....	72
Simultaneous Operations.....	72
Programming Examples.....	73
Program a Fixed Frequency.....	73
Set a Differential Output Range 3 Frequency.....	73
Program a Sweep.....	73
Program a Pulsed Chirp.....	74
Program a Simple Non-Pulsed Chirp.....	74

Program FSK using Internal PWM-----	74	Signal Characteristics-----	78
Program a LIST using Multiple Components-----	75	SOFTWARE UPDATE PROCEDURE -----	80
Program a LIST Using Trigger-----	75	REVISION HISTORY -----	81
EXTERNAL CONTROL -----	77		
Recommended Cables/Connectors-----	77		
External Connector Pin Description-----	78		

Table of Tables

Table 1 UART Interface	7
Table 2 - Parameter Limits.....	7
Table 3 - LIST Dwell Times.....	7
Table 4 - Hardware Limits.....	7
Table 5 Tracking and Lock Time.....	8
Table 6 Modulation Accuracy.....	8
Table 7 Command / Query Timeout.....	20
Table 8 Parameter Responses Ex.....	22
Table 9 – Boot Clock Commands	24
Table 10 SCPI Mandated Common Commands	26
Table 11 IEEE 488.2 Optional Commands	29
Table 12 Calibration Commands	29
Table 13 Output Commands.....	31
Table 14 Selecting an Output	31
Table 15 SForge Command	32
Table 16 ASK Commands.....	34
Table 17 CM Commands	36
Table 18 FM Commands	38
Table 19 Frequency Commands	40
Table 20 Frequency Ranges	41
Table 21 Module Freq Range.....	41
Table 22 List Commands	44
Table 23 OM Commands.....	48
Table 24 PM commands	49
Table 25 Power Commands.....	50
Table 26 Sweep Commands	51
Table 27 Voltage Commands.....	53
Table 28 Summary Status Register	56
Table 29 Event Status Register	56
Table 30 Operational Status Codes.....	57
Table 31 Questionable Condition Register	57
Table 32 - Status Commands	58
Table 33 - Systems Commands.....	59
Table 34 - Trigger Commands.....	60
Table 35 - Command Error Codes	65

Table 36 - Execution Error Codes.....	66
Table 37 - Device-Specific Error Codes	66
Table 38 Operation Complete Code	67
Table 39 - SCPI Parser Error Codes.....	67
Table 40 - User Program Error Codes	67
Table 41 - Internal Error Code	68
Table 42 - Initialization Status	70
Table 43 - Output Control Commands...72	
Table 44 Common Operations	72
Table 45 External Control Pinout.....	77

Table of Figures

Figure 1. ASK Operation	12
Figure 2. BPSK Operation	12
Figure 3. Chirp Operation	13
Figure 4. FSK Unramped Modulation	14
Figure 5. FSK Ramped Modulation	14
Figure 6. FSK Triangle Modulation	15
Figure 7. Chirp Operation	36
Figure 8. SCPI Status Registers.....	55
Figure 9. External Control Pin Locations	77
Figure 10. External Control Pin Filter.....	79
Figure 11. FSK_BPSK Pin Operation	79

Introduction

This manual provides information for remote operation of Signal Forge signal generators using commands sent from an external controller using the SCPI protocol via an RS232 interface. Although based on the SCPI specification, this manual has all the information required for programming the device.

Programming Interface

IEEE488.2 and SCPI-1999.0 are the two standards used to define this instrument's programming interface.

IEEE488.2-2004 (Standard Digital Interface for Programmable Instrumentation) describes how to send commands to instruments and how to send responses to controllers. It defined some frequently used "housekeeping" commands explicitly, but each instrument manufacturer was left with the task of naming any other types of command and defining their effect

SCPI 1999.0 Standard Commands for Programmable Instruments (SCPI) is the new instrument command language for controlling instruments that goes beyond IEEE 488.2 to address a wide variety of instrument functions in a standard manner. SCPI promotes consistency, from the remote programming standpoint, between instruments of the same class and between instruments with the same functional capability. For a given measurement function such as frequency or voltage, SCPI defines the specific command set that is available for that function.

SCPI Conformance Information

Unless otherwise specified, commands and queries are performed as described by the SCPI standard.

The response for commands (not queries) is not SCPI compliant. A SCPI command response normally consists only of a terminator. This instrument first sends a one-character '0'-'9' to indicate the number of errors in the error queue. See the protocol section for specifics.

Interface Specifications

Parameter	Description
UART operation	115,200 baud (fixed), 8 data bits, 1 stop bit, no parity.
input buffer size	60 bytes not including the command terminator.
error queue size	3 errors
error string length	80 characters. This is the maximum response length for a SYST:ERR:NEXT? query.
maximum command/query response time	2 seconds

Table 1 UART Interface

Parameter	Limits
Sweep dwell (step) time	115us - 4 minutes
CM (chirp) pulse time	7us - 4 minutes
CM (chirp) idle time	0, or 5us - 4 minutes
Frequency step size (FSK, chirp, sweep)	1hz – <frequency range>
Internal modulating signal frequency (ASK, FSK, OOK, BPSK)	2500hz - 500khz
Internal modulating signal duty cycle	2-98% (smaller range near highest frequencies)

Table 2 - Parameter Limits

Operating State	Limits
LIST:FAST = OFF (all components)	30us – 1.677 seconds
LIST:FAST = ON	
component = FREquency	10.5us - 1.677 seconds
component = PHASe	7us - 1.677 seconds
component = POWer	5us - 1.677 seconds
component = FM, OM, or PM	5us - 1.677 seconds

Table 3 - LIST Dwell Times

See LIST subsystem for details.

Parameter	Limits
Trigger In	min high/low time - 1us
SYNC output signal	min high/low time – 200ns

Table 4 - Hardware Limits

Output and Range	Maximum Track / Lock Time
TTL (any frequency)	0ms / 0ms
single-ended range 1 (1khz – 102mhz)	0ms / 0ms

single-ended range 2+ (102mhz-1Ghz)	1ms / 5ms
Differential	1ms / 5ms

Table 5 Tracking and Lock Time

Track time is the rate at which the agile reference frequency can be followed by a downstream PLL. It defines the maximum ramp rate for operations like FSK, sweep, and chirp.

Lock time is similar to track time, but adds time for the frequency to settle (the frequency moves back and forth until it finally settles). The lock times above are for the completely settled frequency.

Firmware Modulation Accuracy

Firmware modulation accuracy depends on the operation performed as shown in the following table.

Operation	Accuracy
Chirp pulse, idle 1 and idle 2 times	1+ second accuracy is 1:100000 under 1 second portion accuracy is 1:256 under 100us accuracy 500ns
Sweep dwell time	same as chirp
LIST dwell time	accuracy is 1:256 under 100us accuracy 500ns

Table 6 Modulation Accuracy

Parameter Significance and Accuracy

Integer numbers (specified using <NR1> format) up to 32 bits (4,294,967,295) can be exactly represented (positive or negative). Floating-point values are stored with a significant value up to 32 bits (4,294,967,295). The accuracy of floating point parameters is the same regardless of which format is used (<NR1> or <NR2> or <NR3>).

Getting Started

Power Adapter

An AC power adapter is provided. The specifications are: Input 110-240V, 50/60 Hz; Output +15VDC, 1.3A. The power connector is on the rear panel. Use only the power adapter that came with your SF1010. Note: The operating requirements of the SF1010 are: 10V to 15V, 700mA

Connecting the Signal Generator to Your Computer

1. Make the physical PC to instrument connection

Serial Port: Attach a standard serial port cable with a male DB-9 connector and straight through pin out to the RS-232 port on the rear panel of the SF10101 and the other end to the serial port on your computer.

Standard serial cables may be purchased many sources including Signal Forge.

USB: To connect your computer to the SF10101 via USB connection, use an inline serial port to USB adapter. A USB to serial adapter is available from the Signal Forge web site, www.signalforge.com.

2. Install the Signal Forge EZ Wave software on your computer

Start EZ Wave (Vista, XP, W2K) or a custom program and configure it as follows:

- Select the appropriate COM port
- Set the COM port for 8 data bits, 1 stop bit, no-parity, no flow control, baud rate 115,200 (fixed). This is automatic when using EZ Wave.

Programming Quick Facts

This instrument closely follows the SCPI standard IEEE 488-1990.0 with a few minor differences. This section describes some short facts that will help someone already familiar with SCPI to quickly get started.

Read the Protocol and Power-Up chapters for starting communication.

The SF1010 supports the following features:

- The COM port is fixed at 115200 baud, 8 bits, 1 stop, no parity.
- Multiple command statements are not allowed – use one command or query per statement.
- The normal response to a command is <char><linefeed>, where <char> is one ASCII character '0' through '9' that indicates the number of errors in the error queue.

- The response to a query is <ASCII text><linefeed>. <ASCII text> does not include a terminating null character. <ASCII text> does not include quotes unless otherwise stated (e.g. SYS:ERR:NEXT? query has quotes but *IDN? does not).
- The response to an unknown command/query is an empty string (just a <linefeed>). This is the only case where an empty string is sent as a response so it always indicates an error.
- Paths only use the short version (e.g. FREQ - not FREQuency) even though the full name is shown in the manual for clarity.
- Literal parameters may be sent using the short or long version, and are always returned in the short version.
- You can use upper or lower case characters for the path and literal parameters. Upper case is always returned.
- SCPI <numeric_value> is different from the <NRf> format because it allows literals. This instruments allows all value parameters to use a query with parameter MINimum or MAXimum to query the min/max legal value, such as "FREQ:FIX? MIN".
- Unit suffixes are ignored - the UNITs subsystem is not supported. Frequencies are hertz, time is nanoseconds, power is dBm, voltage is volts, phase is degrees.

Waveforms

The SF1010 provides a wide range of waveform modulation operations including ASK, BPSK, FSK, OOK, chirp, sweep, and arbitrary modulation of frequency, phase, power, and/or OOK.

Single Tone

Programmed using the FREQUENCY subsystem.

The Single Tone waveform outputs a continuous tone at a user-selected output frequency.

Output types

- RF
- Differential
- Digital (TTL: 3.3V, LVTTTL: 2.5V, SVTTL: 1.8V)

ASK

Programmed using the AM subsystem.

Output types

- RF (single-ended)

Modulation Options

- Internal PWM signal (fixed frequency with user-specified duty cycle)
- External control

Amplitude-shift keying (ASK) is a form of modulation that represents digital data as variations in the amplitude of a carrier wave. The amplitude of an analog carrier signal varies in accordance with the bit stream (modulating signal), keeping frequency and phase constant. The amplitude level can be used to represent binary logic 0s and 1s.

ASK varies the output power (amplitude) between two user-selected settings that are 16dB apart. ASK may be controlled internally or externally. Internal control can be done using a simple PWM (pulse width modulation) signal. PWM is a fixed frequency with user-specified duty cycle.

Note that the LIST subsystem can be used to perform arbitrary amplitude modulation and is not limited to two power levels.

For externally controlled ASK, the supported frequency range is 1 Hz to 1 MHz (see drawing below). The ASK control pin on the External Control connector block on the front panel is used as an arbitrary modulation source to control the output carrier.

The drawing below illustrates ASK operation:

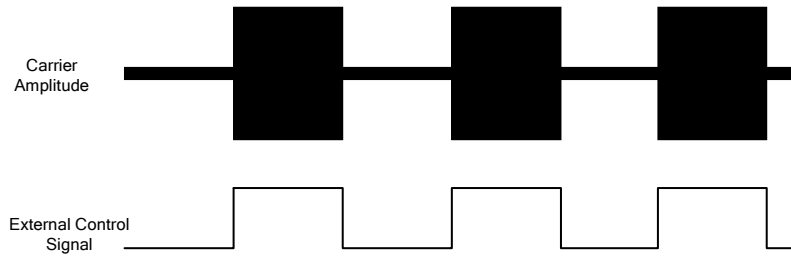


Figure 1. ASK Operation

BPSK

Programmed using the PM subsystem.

BPSK (binary or bipolar phase shift keying) outputs a fixed frequency that changes between two phase offsets. The offset can be programmed from 1 to 259 degrees but is normally 180°.

BPSK modulates at 1bit/symbol and is used for testing low data-rate applications such as IEEE 802.11g wireless LAN standard.

BPSK is also suitable for testing low-cost passive transmitters such as those used in the RFID standards which have been adopted for biometric passports, credit cards and other applications. ZigBee devices which operate in the 868–915 MHz frequency band also employ BPSK.

Due to synchronization the phase changes are not instantaneous, but the user can inspect the timing and adjust the phase values if compensation is desired.

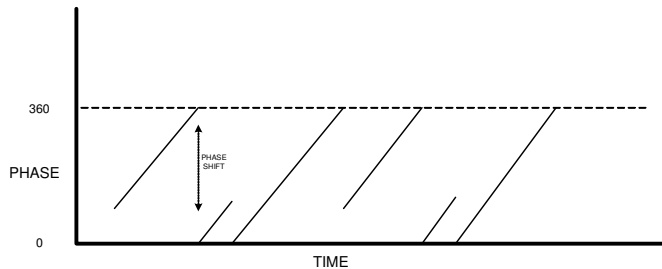


Figure 2. BPSK Operation

Output types

- RF range 1 (to 102 MHz)
- Digital (TTL)

Modulation Options

- Internal PWM signal (fixed frequency with user-specified duty cycle)
- Internal LIST (arbitrary)
- Externally controlled BPSK (FSK_BPSK pin) – low selects the phase 0

Note that the LIST subsystem can be used to do arbitrary nPSK under firmware control.

Chirp

Programmed using the CM subsystem.

A typical chirp ramps the output frequency for a specified time, waits for a specified time (may be 0), then starts over. This instrument adds a third state where the initial chirp frequency can be held for a specified period of time.

A rising or falling chirp is allowed.

The SYNC output signal can be used to trigger on the start of a chirp.

Output types

- RF
- Differential
- Digital (TTL)

Options

- triggered start of each chirp

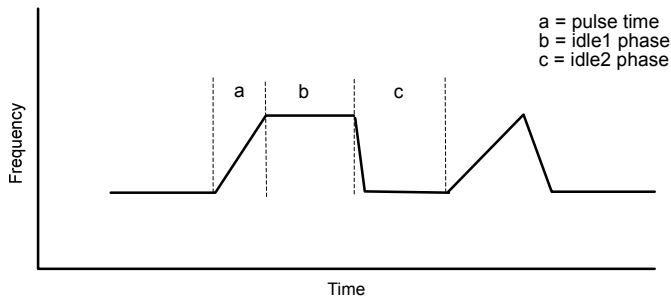


Figure 3. Chirp Operation

FSK Unramped

Programmed using the FREQUENCY and FM subsystems.

Frequency Shift Keying (FSK) unramped is modulating between two discrete output frequencies. Frequency control can be done internally or externally.

Output types

- RF
- Differential
- Digital (TTL: 3.3V, LVTTTL: 2.5V, SVTTL: 1.8V)

Modulation Options (direction control)

- Internal PWM signal (fixed frequency with user-specified duty cycle)
- Internal LIST (arbitrary)
- External using the front panel FSK_BPSK signal

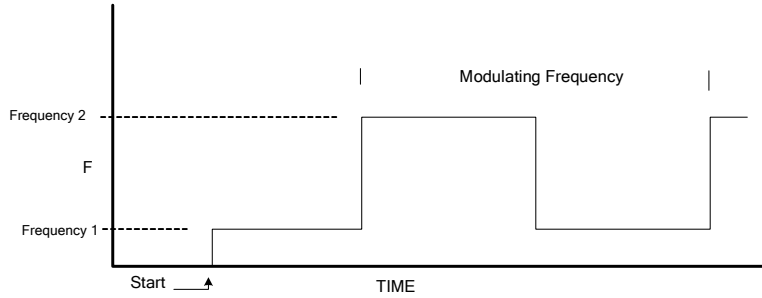


Figure 4. FSK Unramped Modulation

FSK Ramped

Programmed using the FREQUENCY and FM subsystems.

The FSK Ramped waveform varies the output frequency within a specified range. The delta frequency and dwell time specify the ramp characteristics (how fast the ramp occurs). Frequency control can be done internally or externally.

Output types

- RF
- Differential
- Digital (TTL: 3.3V, LVTTTL: 2.5V, SVTTL: 1.8V)

Modulation Options (direction control)

- Internal PWM signal (fixed frequency with user-specified duty cycle)
- Internal LIST (arbitrary)
- External using the front panel FSK_BPSK signal

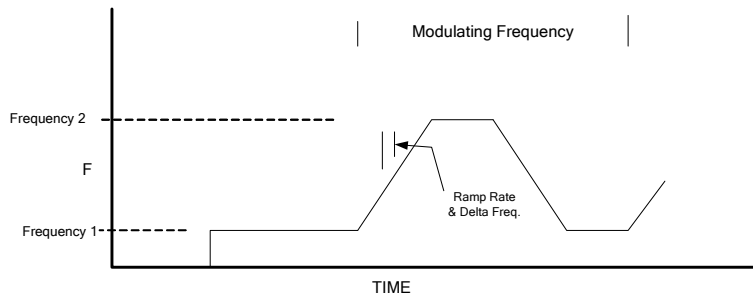


Figure 5. FSK Ramped Modulation

FSK Triangle

Programmed using the FM subsystem.

FSK Triangle is similar to FSK ramped, except that ramping occurs automatically. When an end frequency is reached the direction changes and ramping continues towards the other frequency. See drawing below:

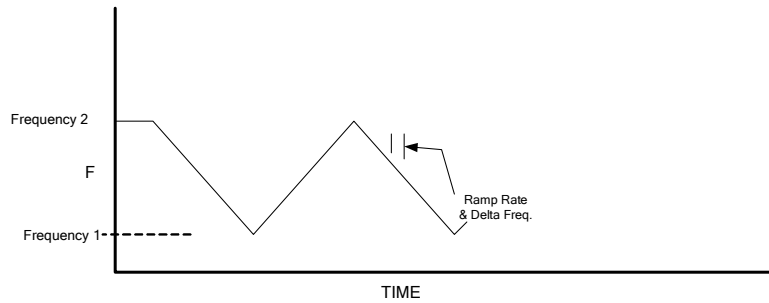


Figure 6. FSK Triangle Modulation

Output types

- RF
- Differential
- Digital (TTL: 3.3V, LVTTTL: 2.5V, SVTTL: 1.8V)

No Modulation Options

OOK

Programmed using the OM subsystem.

OOK (on/off keying) modulation is where the output is turned ON or OFF.

Output types

- RF range 1 (up to 102MHz)
- Differential (all frequencies)

Modulation Options

- Internal PWM signal (fixed frequency with user-specified duty cycle)
- Internal LIST (arbitrary)
- External using the front panel FSK_BPSK signal

Sweep

Programmed using the Frequency and Sweep subsystems.

Sweep is similar to FSK ramped operation with more features. Sweep operates under firmware control and supports:

- larger dwell times than FSK (up to 4 minutes)
- triggered operation
- bi-directional sweep
- dynamic pause, direction change, single-step, and jump to a frequency
- SYNC output signal (can use to trigger on start of sweep, or each step)

Output types

- RF
- Differential
- Digital (TTL: 3.3V, LVTTTL: 2.5V, SVTTL: 1.8V)

Arbitrary

Arbitrary waveforms are programmed using the LIST subsystem. An arbitrary number of points can be created with each point specifying any logical combination of frequency, phase, power, FSK control, BPSK control, or OOK control. Each point can specify a dwell time (how long the point executes).

Control operations can be specified for each point and include trigger input, SYNC output high/low, and alert message.

SCPI Syntax

The actual name of a command consists of one or more keywords since SCPI commands are based on a hierarchical structure, also known as a tree system. In such a system, associated commands are grouped together under a common node in the hierarchy.

A keyword or program mnemonic is limited to 12 characters, and a short form (max 4 characters) can be used to reduce the string length. Signal Forge products described in this manual only accept the short version for path members (short or full for literal parameters).

"White space" is defined as ASCII code 0 through 9, 11 through 32. When whitespace is seen in the input stream the whitespace character is replaced with a space.

Per IEEE 488.2 a <Terminated Program Message> consists of one < Program Message Element> followed by a <Program Message Terminator>. A <Program Message Element> may be one of the following:

- Command Message Unit
- Query Message Unit

The <Program Message Terminator> is a linefeed (ASCII 10).

Example <Terminated Program Message> commands:

```
*RST (set device to the default state)
OUTP:STAT ON (turn on the output)
FREQ:FIX 12345 (set output frequency to 12,345hz)
```

Example <Terminated Program Message> queries:

```
SYST:ERR:NEXT? (returns oldest error in error queue)
*FREQ:FIX? (returns current output frequency)
*FREQ:FIX? MIN (returns minimum output frequency)
```

Command Message

The <Common Command Program Header> is the format for the common commands defined by 488.2. Their distinguishing feature is that they are preceded by a "*" (asterisk). For example:

```
*IDN
*RST
*SRE
```

The <Compound Command Program Header> is a command path consisting of <Program Mnemonic> strings separated by ":" (colon) characters. A ":" may be added in front as well. For example:

```
FREQ:FIX
```

Query Message

A query is used to interrogate a device for information. A query has the same syntax as the command message except that a "?" is tacked on to the end (and no parameter is sent).

There are four elements to the talking (response) syntax:

- <Response Message Terminator>: The only legal SCPI terminator for the talking syntax is linefeed (10). This is used at the end of a stream of response data (for command or query).
- <Response Message Unit Separator>: This separator is defined as a ";" (semicolon), and is used to separate different responses in the response stream.
- <Response Data Data Separator>: This separator is defined as a "," (comma), and is used to separate data items in a response.
- <Response Header Separator>: This separator is defined as a " " (space), and is used to separate the response header (if applicable) from the response data.

Data Parameters

SCPI uses the parameter forms described in *IEEE 488.2-2004*, section 7.7, with some additional restrictions.

Data parameters, referred to simply as "parameters," are the quantitative values used as arguments for the command keywords. The parameter type associated with a particular SCPI command is determined by the type of information required to control the particular instrument function. For example, boolean (ON | OFF) parameters are used with commands that control switch functions.

The command descriptions specify the type of data parameter used with each command.

Notational Conventions

The SCPI interface standardizes command syntax and style that simplifies the task of programming across a wide range of instrumentation. As with any programming language, the exact command keywords and command syntax must be used. Unrecognized commands, or improper syntax, will generate an error.

Except for common commands, each keyword has a long and a short form. In this manual, the long form is presented with the short form in upper case and the remainder in lower case.

The short form keyword is usually the first four characters of the long form (example: `FREQ` for `FREQuency`). The exception to this is when the long form is longer than four characters and the fourth character is a vowel. In such cases, the vowel is dropped and the short form becomes the first three characters of the long form. Example: the short form of the keyword `TIMer` is `TIM`.

General Notations

The syntax conventions used for all SCPI command keywords and data parameter descriptions in this manual are described below:

- : A colon links command keywords together to form commands. The colon is not an actual part of the keyword but is a signal to the SCPI interface parser.
- [] Square brackets enclose one or more optional parameters.
- { } Braces enclose one or more parameters that may be included one or more times.
- | A vertical bar indicates "or" and is used to separate alternative parameter options.

Example: ON | OFF is the same as ON or OFF

<> Angle brackets enclose parameter descriptions.

sp space(s), referred to as whitespace, that must be used to separate keywords from their associated data parameters. It *must not* be used between keywords, or inside keywords.

For further information about SCPI command syntax and style, refer to the Standard Commands for Programmable Instruments (SCPI) 1999.0 document.

Parameter Notations

The following syntax conventions are used for all data parameter descriptions in this manual

<boolean> - ON | OFF. Can also be sent as 1 or 0, where 1 means ON and 0 means OFF. An <NRf> is rounded to an integer. A nonzero result is interpreted as 1. Boolean parameters are always returned as 1 or 0 in <NR1> format by query commands

<non_decimal_numeric> - is a number specified in hex, octal, or binary formats as follows:

#H<series of ASCII A-F>

#Q<series of ASCII 0-7>

#B<series of ASCII 0 or 1>

<numeric_list> - A numeric list is an expression format for compactly expressing numbers and ranges of numbers in a single parameter. Example: (3:7, 9:13).

<numeric_value> - decimal numeric element . This includes <NRf> and <non_decimal_numeric> parameters, as well as the special literal forms (character program data) listed here and as detailed in SCPI 1999.0. In this manual, when a parameter is specified as using <NRf> or <numeric_value> format, the data is stored and returned in <NR1> format unless otherwise specified.

The special form numeric parameters MINimum and MAXimum can be used in place of a value parameter to set the minimum or maximum value. The maximum and minimum are queryable by sending <header>? MAXimum|MINimum (e.g. FREQ:FIX? MIN).

DEFault, UP, DOWN, NAN, INFinity, and NINFinity may be implemented as described under each command.

NR1/2/3 and NRf also accept these literals (character program data).

<NAN> - not a number is represented as 9.91E37. This value is defined in IEEE754 but for this instrument it is used only as the query response when the parameter value is unknown or invalid for the current instrument state. This instrument only returns <NAN> as defined in the LIST subsystem.

<NR1> - a signed integer without a decimal point, such as 73.

<NR2> - real number without exponent, such as 53.7.

<NR3> - real numbers with exponents, such as 5.35E4. Whitespace is allowed before and after the “E” but not between the sign and digits, such as “+3.24 e -3”.

<NRf> - <NR1>|<NR2>|<NR3>. Known as flexible numeric representation. Data can be sent in any format. In this manual, when a parameter is specified as using <NRf> or <numeric_value> format, the data is stored and returned in <NR1> format unless otherwise specified. As of 8/24/08 the only parameters that use <NR3> are time values whose maximum value exceeds 4 seconds (exceeds a 32-bit value for nanoseconds). This includes sweep dwell time, and the chirp pulse and idle times.

<character data> - <CHARACTER PROGRAM DATA>, an unquoted string of ASCII characters.

Examples: FIXed, UP, and DOWN

<string data> - <STRING PROGRAM DATA> - ASCII characters surrounded by double quotes, example: “OFF”.

Protocol

The com port is fixed at 115200 baud, 8 bits, 1 stop bit, no parity.

The instrument supports a simple command/query and response format. The controller sends one command or query then waits for one response.

The instrument:

- Performs commands sequentially.
- Allows one command <Program Message Element> per <Terminated Program Message>.
- Allows one query per <Terminated Program Message>.
- Does not perform handshaking on the RS232 interface. Buffer control is assured by following the rule of waiting for a response after every command or query.

The controller should normally set its read response timeout as follows.

Model	Command / query timeout
SF1010	3 seconds

Table 7 Command / Query Timeout

This instrument does not have a (message) output queue because only single queries are allowed, there is no RS232 handshake (can't be stalled), and commands / queries are handled sequentially (no queue necessary). This means the MAV bit of the Instrument Summary Status Register will always be 0.

Responses

The normal response to a command is <char><linefeed>, where <char> is one ASCII character '0' through '9' that indicates the number of errors in the error queue (same as SYSTEM:ERROR:COUNT? query). This allows the control program to quickly determine if an error exists rather than querying the device after each command.

The normal response to a query is <ASCII text><linefeed>. <ASCII text> does not include a terminating null character. Most query responses do not include quotation marks (those cases are specified with the command description).

There are three cases where the response is only a <linefeed> (not a normal response):

1. Characters are received while a command/query is being processed. The following error is added to the queue: -362, "Busy". This will not happen if the simple protocol is followed.
2. The input buffer overflows. The following error is added to the queue: -363, "Input buffer overrun".
3. An error occurred parsing the command or query, and therefore the command/query did not execute. This includes unknown commands and syntax errors as defined in the Status subsystem (most falling in the range -100 to -199).

For case 1 (busy error), the device will complete the current command/query normally including sending the normal response. It waits to receive a <linefeed> then responds only with <linefeed>.

Note: If the device receives a terminating linefeed while still executing a command, the linefeed is discarded (as are all the other characters) so the controller will likely receive a UART timeout because there will be no response. The next command received by the device, even if complete and valid, will be ignored and the single linefeed response for the busy error will be returned.

A simple recovery for this condition is to issue "SYST:ERR:COUN?" queries until a '0' is returned.

Case 2 and 3 do not have the protocol problem like case 1. They are simple errors.

These cases do not use the normal response because a corrupted command is neither a command nor query. For example, a "1" response might be misinterpreted as a valid query response when in actuality the query was unrecognized.

Suggested Programming

A command error has occurred when the response is anything other than <char><linefeed> where <char> = '0'.

A query error has occurred when the response is a single <linefeed>. This is usually because the query was not recognized. A valid query always returns characters before the <linefeed>.

In any case, the simple thing to do is to query SYST:ERR:COUN? whenever one of the two cases above happens. If the count is not 0, an error has occurred. This method of programming means you don't have to continuously query the instrument (e.g. after every command) to determine if an error has occurred.

Query Version of Commands

All commands, unless otherwise noted, have an additional query form. As defined in IEEE488.2, a query is a command header with a question mark symbol appended (for example, `FREQ:CENT?`). There are normally no parameters in a query.

When a query form of a command is received, the current setting associated with the command is placed in the output buffer. Query responses do not include the command header. Execution of a query has no side effects – i.e. no internal parameters are changed and no errors are added to the error queue.

Numeric parameters have a special form of query that allow MINimum or MAXimum as a parameter for querying the min/max value for the parameter.

Example: `FREQ:FIX? MIN`

Both the command and query forms are implemented unless a comment or note indicating otherwise appears in the description

Responses to Queries

According to SCPI, the responses to queries are partly subject to stricter rules than in standard IEEE 488.2.

1. The requested parameter is transmitted without header.

Example: `FREQuency:FIX?`

Answer: 10555

2. Maximum values, minimum values and all further quantities, which are requested via a special text parameter are returned as numerical values

Example: `FREQuency:FIX? MAX`

Answer: 100000000

3. Boolean values are returned as 0 (for OFF) and 1 (for ON).

Example: OUTPut:STATe?

Answer (for ON): 1

4. Text (character data) is returned in the short form

Example: SElect:PORT?

Answer (for MODule): MOD

Unit suffixes are not returned with values. The default unit for a parameter is used.

Queries for numeric values return NR1 format when the value can be exactly represented (i.e. integer value +/- 0xFFFF_FFFF). All other values are returned in NR3 format as:

[-]<significant digits>E[-]<exponent>.

where “[-]” means the negative sign is sent (a “+” is never sent).

Command sends	Query returns
7900	7900
7.90	790E-2
7.900	7900E-3
-1.23	-123E-2
4294967296 (max integer +1)	429496729E1

Table 8 Parameter Responses Ex.

ASCII <CANCEL> Command

The ASCII <CANCEL> cancel character (decimal 24) can be used to determine which set of code has control (boot or application). It can be used to synchronize with the device at startup without creating messages for the error queue (or you can just wait for the front panel LED to stop blinking after about 4 seconds).

When a <CANCEL> character is received the device empties the input buffer and responds with <char><linefeed>. <char> is one of the following:

‘B’ if the boot block firmware is running,

‘A’ if the application firmware is running.

Do not send a linefeed with the <CANCEL> character as that would look like an empty command after the <CANCEL> is processed.

The <CANCEL> command is not required for normal programming, and is normally used by application code only when downloading firmware.

Power-up

See the boot block section for how firmware begins execution.

When application firmware gets control (LED stops blinking), it loads certain parameters from non-volatile memory:

- Calibration
- Sticky data – see 0

Application firmware does not respond to the UART until it has completed startup tests and run the optional startup macro. The <CANCEL> command can be used to determine when the instrument is ready without creating errors.

Saved and Sticky Data

“Sticky” data refers to any parameter or data that is saved in non-volatile memory and automatically restored after a power cycle. “Saved” data is only restored when a command is issued to restore it (e.g. a macro or LIST data).

When new firmware is downloaded, all “saved” and “sticky” data is reset to their default values unless otherwise specified. Saved and sticky data includes the following:

- Sticky calibration. This is not lost when new firmware is downloaded
- Sticky parameter SFORge:MACRO:ENABLE, which determines if a macro is run at power-up
- Saved Macro (list of commands)
- Saved LIST data

Boot Block

Firmware begins execution in the protected boot block of the instrument. The boot block is a small set of code that is always available in case the application firmware is ever corrupted (e.g. power fails during a firmware update).

When the instrument starts up, boot block code has control. If application firmware is valid, the instrument will jump to application firmware when a period of 3 seconds has occurred since the last command (or if no commands are sent). This can be overridden using the “!STAY” command.

The LED will blink with a regular rhythm while control remains in the boot block. When application code gains control, the LED is continuously ON.

The boot block input buffer size is sufficient for any legal boot block command.

Boot Block Commands

The boot block only recognizes the commands shown in this section. The boot block is not SCPI compatible but the responses to commands and queries are done in the same manner.

As in normal firmware, the response to commands is <char><LF>, where <char> is ASCII ‘0’ or ‘1’. ‘0’ means there is no error string. ‘1’ means there is one error string available. The error “queue” is only one deep and the only query for it is “SYST:ERR:NEXT?”.

The following commands are supported.

Command	Parameters	Comments
*IDN?	(none)	query only
SYST:ERR:NEXT?	(none)	query only
!DOWN	(none)	command only
!JUMP	(none)	command only
!STAY	(none)	command only

Table 9 – Boot Block Commands

*IDN?

Same as the SCPI query. The string portion for <Firmware revision level> starts with a “B”. The “B” is a flag that indicates you are speaking with boot block firmware.

Example:

Signal Forge LLC,SF1010,0,B3.2

SYST:ERR:NEXT?

Same format as SCPI query. Returns error code and error string in quotation marks. The error code is ‘0’ only if no error exists.

Examples:

0,“No error”

-211, "Settings conflict; invalid firmware"

!DOWN

Initiates downloading an application firmware file. When the command is recognized, operation begins as described in a section below.

Note that when downloaded firmware gets control, it will invalidate all saved data such as LIST data and macros because the saved data may not be compatible with the new firmware. Calibration is not affected.

!JUMP

This tells the device to jump to application firmware. If the application firmware is valid the instrument starts executing it (after returning the '0' or '1' response) and the LED should stop blinking immediately. If the application firmware is not valid then an error is posted (the LED will continue blinking to indicate the boot block still has control).

After sending this command, the controller should give time for the application firmware to initialize (it might miss UART characters during this time).

!STAY

Tells the device to continue execution in the boot block (even if application firmware is valid). The LED will continue to blink. This command can be user to recover from a (rare) condition where the application firmware is corrupted but marked as valid.

Downloading

After sending the download event command the controller will receive the usual command response of '0' or '1' plus the response terminator <LF>.

Downloading now commences. The controller sends each line terminated by <LF> then waits. The instrument processes the line then sends one of these characters followed by a response terminator <LF>:

- '0' – indicates no error, continue sending data.
- 'V' – indicates that the code terminator flag has been recognized and the firmware has been programmed and validated.
- '1' means an error was added to the queue and downloading has stopped.

If any code other than '0' is received, the controller should not send any more lines. The instrument has exited the downloader and returned to the normal command loop.

If an error occurs you will still be talking with the boot block firmware and can use the usual status commands to determine the error.

If the code indicated validated firmware ('V') the instrument will immediately reset itself, causing a reboot and loading of the new application firmware. The LED will stop blinking after a few seconds (means that application firmware has control). A "*IDN?" query can be used to verify that you are now talking with normal application firmware.

If the entire file is sent and no 'V' was received to indicate that the instrument saw the code terminator flag, the instrument must be physically reset (it will wait forever).

SCPI Commands

This instrument does not have an (message) output queue because only single queries are allowed, there is no RS232 handshake (can't be stalled), and commands / queries are handled sequentially (no queue necessary).

Common Commands

The common commands are taken from the standard SCPI-1999.0 volume 1 section 4.1.1. Identical commands have an identical effect in different instruments. The headers of these commands consist of an asterisk "*" followed by three letters. Many common commands affect the status reporting system.

Mandated Common Commands

Mnemonic	Parameters	Name
*CLS	(none)	No query. Clear Status Command
*ESE	<NR1> 0..255	Standard Event Status Register Enable. Powerup default is 0.
*ESR?	(none)	Query only. Event Status Register.
*IDN?	(none)	Query only. Identification Query
*OPC	(none)	Operation Complete Command
*RST	(none)	No query. Reset Command.
*SRE	<NR1> 0..255	Service Request Enable Command. Powerup default is 0.
*STB?	(none)	Query only. Reads Instrument Summary Status Register (a.k.a. Status Byte)
*TST?	(none)	Query only. Self-Test Query.
*WAI	(none)	No query. Wait-to-Continue Command.

Table 10 SCPI Mandated Common Commands

*CLS (Clear Status Command)

Clears all status data structures in the device (SCPI 4.1.3.2). All event registers are cleared, while enable registers are unaffected.

Clears these registers:

- Standard Event Status Register (IEEE-488.2 11.5.1.2.4)
- Instrument Summary Status (SCPI 4.1.3.2)
- Questionable Status & Event Register (SCPI 4.1.3.2)
- Operation Status & Event Register (SCPI 4.1.3.2)

Also clears the error queue, the OPC pending flag, and any other registers that are summarized in the Instrument Summary Status Register.

***ESE (Standard Event Status Enable Command)**

Sets the Standard Event Status Enable Register bits. The binary weighted <NR1> data parameter used with this command must have a value between 0 to 255. Refer to “Status System Programming”.

***ESE? (Standard Event Status Enable Query)**

Returns the value of the Standard Event Status Enable Register in <NR1> format. Refer to “Status System Programming”.

***ESR? (Standard Event Status Register Query)**

Returns the value of the Standard Event Status Register in <NR1> format. This command clears the Standard Event Status Register. Refer to “Status System Programming”.

***IDN? (Identification Query)**

This query returns an instrument identification string in IEEE- 488.2 specified <NR1> format (four fields separated by commas). The fields are: <Manufacturer>,<Model>,< Hardware revision level >,<Firmware revision level>; where the actual model number, serial number, and firmware version of the generator queried will be passed.

If the first character of the <Firmware revision level> is a “B” then you are talking with the boot block. The boot block has a limited command set. Refer to the boot block section of this document.

Information about attached modules can found in the SFORge subsystem commands.

Example of returned string for SF1010:

Signal Forge LLC,SF1010,2,3.2.0.3

The <firmware revision> portion consists of four parts:

<major version>.<minor version>.<build>.<revision>

- <major version> is changed when a significant feature is added.
- <minor version> is changed when a feature is modified, a command or query changes the way it works, or the <major version> is changed (<minor version> is set to 0).
- <build> is changed when the source code does not change, but a compile option or library changes.
- <revision> is changed when no features are changed, but the code changes. This is typically done for bug fixes, but may also be done for improvements (e.g. speed enhancement, more error checking).

***OPC (Operation Complete Command)**

Has no functional use for this device. Sets the Operation Complete bit in the Standard Event Status Register.

***OPC? (Operation Complete Query)**

This query is not SCPI compliant.

This query is used to synchronize with the instrument when it is running a time-consuming command. The time-consuming commands are:

running a macro (using the SFORge:MACRo:RUN command)

This query returns a ‘1’ if all commands are completed (instrument is not busy). It returns ‘0’ if the instrument is running a time-consuming command.

After starting a time-consuming command, this query should be used to determine when the instrument is ready for the next command. If not used, the controller will still execute the command/query but there could be side effects. For example, if you interrupt a macro then the new command could place the instrument in a state that will cause errors when the macro continues to run.

***RST (Reset Command)**

Resets the instrument to a pre-defined condition with all user programmable parameters set to their default values. These default parameter values are listed under each SCPI command in this manual. This command does not affect the Output Queue (this device does not have one), Instrument Summary Status Register, Standard Event Register, or calibration data.

Also see section 0 for reset and initialization.

Device-specific commands may be used to create a *RST default value for some parameters that are different from the factory default (IEE488 10.32.1). One of these is the calibration source (factory or user-specified).

Per IEE-488 section 10.32.1, the following items are not affected:

- The Output Queue - this instrument does not have an output (message) queue
- Any Event register (doesn't change the power-up default)
- Any Event Enable register (doesn't change the power-up default)
- The Service Request Enable register

***SRE (Service Request Enable Command)**

Accesses the Instrument Summary Enable Register (despite the misleading name). The integer data parameter used with this command must have a value between 0 to 255. A zero value resets the register. Refer to "Status System Programming" 0.

***SRE? (Service Request Enable Query)**

Returns the value of the Instrument Summary Enable Register in <NR1> format. Bit 6 is always zero.

***STB?**

Returns the content of the Instrument Summary Status Register (bits 0–5 and 7). Bit 6 is the Master Summary Status bit value. This command does not modify any register values.

***TST? (Self-Test Query)**

Causes a full internal self-test. Status messages that indicate self-test results are placed in the error queue in the order they occur. Bits in the status register are also affected. Returns the number of errors placed in the error queue. '0' means the unit passed self-test.

When self-test is complete, a *RST should be issued to put the instrument in a known state.

If any self-test fails a status bit indicating failure is set in the Questionable Status Register, and the error queue has one or more messages:

-330,"Self-test failed;<description>"

***WAI (Wait-to-Continue Command)**

This command suspends the execution of any further commands or queries until all operations for pending commands are completed. This instrument executes all commands in order so there are never any pending operations (this command has no effect).

Optional Common Commands

The following optional common commands are either entirely optional or they are required if certain instrument features are implemented.

Mnemonic	Parameters	Name
*OPT?	(none)	Query only.
*RCL	(none)	Not supported. See SFORge:MACRo.
*SAV	(none)	Not supported. See SFORge:MACRo.
*TRG	(none)	No query. Triggers device.

Table 11 IEEE 488.2 Optional Commands

***OPT? (Option Identification Query)**

This command returns a string identifying any device options. “None” is returned if there are no options. Note that this is not SCPI compatible because SCPI would return an empty string.

***TRG**

Causes a trigger event as described under the trigger subsystem. It is one of the highest priority commands.

CALibration Subsystem (CALibration)

The calibration subsystem contains the commands for frequency calibration.

All calibration parameters are “sticky” – they are automatically saved when changes are made and they are restored at power-up.

Command	Parameters	Comments
FACTory:FREQUENCY:SET	<numeric_value>	Query-only factory-set calibration frequency.
FACTory:FREQUENCY:ACTual	<numeric_value>	Query-only Factory-set frequency from counter.
USER:FREQUENCY:SET	<numeric_value>	User-set calibration frequency.
USER:FREQUENCY:ACTual	<numeric_value>	User-set frequency from counter.
:SElect	NONE FACTory USER	Selects no, factory, or user calibration. Ships selected as FACTory. Setting is saved and restored at power up.

Table 12 Calibration Commands

CALibration:FACTory:FREQUENCY:SET and :ACTUAL

These are query-only to protect against accidental writing.

When queried, the SET parameter is the frequency at which the factory calibration was performed (e.g. 10mhz). The ACTUAL parameter is the indicated frequency of the non-calibrated output at the SET frequency – i.e. the frequency seen by a frequency counter when no calibration was being done. The instrument uses the difference between SET and ACTUAL to calculate calibration.

CALibration:USER:FREQUENCY:SET and :ACTUAL

This is a user-specifiable calibration set. Queries return the current values.

The SET and ACTual commands are used to set the USER calibration. SET initiates calibration and ACTual completes calibration as described below.

Valid values for the SET frequency depend on the instrument model. For the SF1010, the single-ended or TTL output may be used and the valid frequencies are 1mhz, 10mhz, or 100mhz.

Each instrument determines what constitutes a valid ACTual value. It typically checks that the frequency is within a specific percentage difference from the SET frequency.

CALibration:SElect NONE | FACTory | USER

This setting selects which calibration numbers are used. Changing the setting does not affect the current output frequency but will be used for subsequent frequency updates.

- NONE – no frequency calibration is done. This may be used if the calibration is lost so that no errors are reported and the instrument can still be used.
- FACTory – factory calibration values are used to adjust the frequency.
- USER – user supplied calibration values are used to adjust the frequency.

Power Up Calibration

If calibration fails at power up, the resulting operation depends on how the calibration failed. Bit 8 of the Questionable Condition register (“calibration failed”) is set whenever the instrument is in a failed calibration state.

- NVRAM checksum error – SElect is set to FACTory, with “calibration failed” set. Note that if this was caused by a power outage while updating the NVRAM it may be possible to recover the factory settings. Do this by querying the factory settings, then using the values to program the USER calibration. This assumes the values look reasonable (they are checked when a new calibration is done).
- NVRAM checksum OK, SElect = NONE. No calibration is performed, and the “calibration failed” bit is not set.
- NVRAM checksum OK, but the selected FACTory or USER calibration data is invalid. “calibration failed” is set.

When “calibration failed” is set the device will report an error any time a frequency parameter is accessed. The error –313, “Calibration memory lost” is added to the error queue.

Calibration Procedure

Calibration should take place in an environment in which the temperature is the same at which the unit will be used.

1. Send *RST.
2. Select the single-ended or TTL output.
3. Program the output frequency anywhere between 15mhz to 100mhz and the output ON (when the output is OFF the power is much lower so the temperature is not normal).
4. Let the unit run for 1 hour prior to calibration to allow for temperature soak.
5. Send CALibration:SElect NONE
6. Send CALibration:USER:FREQUENCY:SET <frequency> to put the instrument into calibration mode (if the command completes successfully). The non-calibrated SET frequency will be output.
7. Verify you are in calibration mode by checking the calibrating bit of the Operation Condition register.
8. Use a frequency counter to read the frequency. Use the longest / most accurate counting setting of the counter.
9. Send CALibration:USER:FREQUENCY:ACTual to specify the frequency you read (EX: 10000007). When the command completes successfully, calibration is complete (saved to NVRAM).
10. Verify you are not in calibration mode by reading the calibrating bit of the Operation Condition register.
11. Send CALibration:SElect USER
12. Send FREQ:FIX <frequency> and verify that the frequency is now calibrated.

The instrument accepts and stores the user calibration when the following occurs:

- CALibrating bit is active and valid CALibration:FREQUENCY:ACTual <frequency> is received.

The instrument exits calibration mode when one of the following occur:

- A *RST command is received (restores calibration).
- CALibration:FREQuency:ACTual is written without error . This saves the new USER calibration to NVRAM. If SElect is set to USER, the new calibration is used, else the calibration is restored to what it was prior to calibrating (determined by SElect).

When in calibration mode, only status queries or other commands specified in this section should be sent. While this command is executing the CALibrating bit of the Operation Status Condition register is set.

If calibration mode is exited without saving new calibration parameters, all CALibration parameters are returned to the values prior to going into calibration mode.

OUTPut Subsystem

Signal Forge signal generators only provide one usable output at a time. The OUTPut subsystem selects the output to use, as well as turning it ON or OFF.

The POWer subsystem contains the commands for setting the output level, level control and level correction of the single-ended output. The VOLTage subsystem is the corollary for the TTL output.

Command	Parameters	Comments
:OSKRamp	<numeric_value>	*RST default is MAX.
:SElect:PORT	SE DIFF TTL MODule	Selects active output (single-ended, TTL, differential, or module). *RST default is SE.
:STATe	<boolean>	ON turns on the output. OFF turns it off. *RST default is OFF.

Table 13 Output Commands

OUTPut:OSKRamp

When using the single-ended output in range 1 (up to 102mhz), OOK/OSK (output shape keying) is possible as defined by the OM subsystem. When OOK is enabled (OM:STATe ON) this parameter controls how quickly the power (amplitude) is ramped up or down at the ON or OFF edge of the OOK signal. MIN / MAX for the SF1010 is 42us / 2.5ms.

OUTPut:SElect:PORT

This command selects the output to be used. The parameters are shown in the next table.

- SE selects the single-ended output.
- DIFF selects the differential output.
- TTL selects the TTL output.
- MODule selects the output of an attached module. If there is a module but the type is unknown, returns error – 221, "Settings conflict;unknown module".

This command causes hardware paths to change and turns the selected output OFF (if possible), lowest dBm or voltage as applicable. Selecting the output and range should be the first commands in a programming sequence (after *RST).

Once the output is changed from the *RST default it cannot be changed until after sending *RST.

Model	SElect value	Description
SF1010	SE	Single ended output
	DIFF	differential output
	TTL	TTL output
Any Module attached to SF1010	MODule	Module-specific (typically single ended).

Table 14 Selecting an Output

OUTPut:STATe

ON enables the output. OFF disables the output.

When OM:STATe is ON OUTPut:STATe commands are rejected with -221, "Settings conflict;OM:STATe ON".

NOTE: Modules usually cannot have their output turned OFF. Commands that try to turn the output OFF receive a -221, "Settings conflict" error and queries return "1" to indicate the output is always ON.

When FSKmux is set to external operation see the FSKmux parameter for output control.

SFORge Subsystem

The SFORge root keyword is used for miscellaneous commands not defined in SCPI.

Command	Parameters	Comments
:DOWNload	(none)	event only
:FSKMux:INTernal	<boolean>	*RST default is ON.
:MACRO		
:DEFine	<boolean>	Powerup default is OFF.
:ENABle	<boolean>	Factory default is OFF.
:RUN	(none)	no query
:SIZE?	<NR1>	query only
:MODule:NAME?	(none)	Returns attached module name.
:MODule:HWVersion?	(none)	*RST default '0'.
:RESet	(none)	no query

Table 15 SForge Command

SFORge:DOWNload

Initiates downloading firmware.

After the response terminator is sent the application firmware is invalidated and a CPU reset is issued to restart the instrument. When the instrument restarts, the application firmware is seen as invalid and the boot block code retains control (only boot block commands are available).

The download process is described in the boot block section.

SFORge:FSKMux:INTernal

Selects whether some hardware signals are controlled internally (ON) or externally (OFF). The signals are FSK_BPSK, DIFF_OOK, and SE_OOK. These signals are used by the following subsystems:

- CM (chirp)
- FM (FSK ramped and nonramped)
- OM (OOK)
- PM (BPSK)

Operations that require a specific FSKmux setting are rejected with an error if the FSKmux setting is invalid. The error is -221, "Settings conflict;incompatible FSKmux".

When FSKmux is set to external operation the FSK_BPSK, SE_OOK, and DIFF_OOK signals are user-controlled. DIFF_OOK is active regardless of the selected output and the value of OM:STATe. SE_OOK is only active if OM:STATe is ON and the output is single-ended. FSK_BPSK is only active when FM:STATe or PM:STATe is ON.

SFORge:MACRO:DEFine

A query returns ON if a macro has been saved (or if a macro is currently being recorded).

Enables storing one set of commands for later recall. This parameter is only cleared at power-up – it isn't cleared by *RST because *RST might be one of its commands.

When this parameter transitions to ON the current macro is cleared, the “Recording macro” bit is set in the operational status register, and the instrument begins recording commands (not queries). When MACRo:DEFine transitions to OFF, commands are no longer recorded and the macro is saved to NVRAM and is available for use.

If the macro space becomes full during the recording process, the “Recording macro” bit is cleared and a device-specific error "out of macro space" is added to the error queue. In that case you must set MACRo:DEFine = OFF, then ON again to start recording another macro.

There is no facility for reading back macro commands.

If you want to run a LIST with the macro, the best method is to define and save the LIST, then use one command in the macro to load the LIST (i.e. rather than sending multiple LIST commands as part of the macro).

If the macro contains a command that would normally start a firmware loop (LIST:STATe, SWEEp:STATe, or CM:STATe ON), that command should be the last in the macro because the entire macro will run before any firmware loops are started.

SFORge:MACRO:ENABLE

When ON the saved macro will be run at power-up. If set ON and there is not a valid macro, the command is rejected with -221, "Settings conflict".

This is a sticky parameter – it maintains its value between power cycles (value is lost if new firmware is downloaded).

SFORge:MACRO:RUN

Runs the saved macro. If there is no valid saved macro, the command is rejected with -221, "Settings conflict".

After starting the macro, you should use the “OPC?” query to determine when the macro has completed (“OPC?” returns ‘1’ when complete, else ‘0’ while still running). A command/query received while the macro is running will execute before the macro continues. This could have unintended side effects that are up to the programmer to control.

Note that this also applies if a macro runs at power-up (SFORge:MACRO:ENABLE = ON).

SFORge:MACRO:SIZE?

A normal query returns the number of commands in the current macro. Returns 0 if no macro saved. The MIN query returns 1, and the MAX query returns the maximum number of commands that can be saved as a macro.

SFORge:MODule:NAME?

Returns one of the following:

- “none” if no module is attached.
- “unknown” if a module is attached, but the model is not recognized.
- The module name, such as “SF2500M”.

Use the “SFORge:MODule:HWVersion?” query to determine if a known module is attached.

SFORge:MODule:HWVersion?

Returns the hardware version for a known module, such as “3”.

Returns “-1” if a module is not attached or it is unrecognized.

SFORge:RESet

This command causes the instrument to issue a CPU reset command which causes CPU registers to reset and operations to occur similar to a system power up (including starting operation in the boot block code). This

command is intended to be used to recover from software bugs that may prevent normal operation, such as an incomplete implementation of the *RST command or a fatal loop.

AM Subsystem

This subsystem contains the commands used to define ASK modulation, which is toggling between two output power levels using a single control signal (ASK). ASK is enabled when AM:STATe = ON. ASK operation is only possible when using the single-ended output.

Note that the LIST subsystem can be used to output a large number of user-specified power levels (not just two), but the maximum rate is slower than ASK.

When ASK is not enabled (the *RST default), the output power is internally compensated for operation across the defined power range (e.g. -13 to +7 dBm, a 20 dB range).

ASK enables direct access to the hardware attenuator for a range of 0-31 dB attenuation, with the ASK signal controlling the MSbit of the attenuator. The power output is now fully controlled by the programmer instead of automatically being compensated for by SF1010 firmware. The output power varies slightly depending on the frequency, so the AM:AMPLitude query is provided to see the uncompensated (no attenuation) power output for the frequency defined by POWER:FREQUENCY.

Command	Parameters	Comments
:DUTY	<numeric_value>	Sets the percentage of time the internal PWM modulating signal is low. *RST default is 50.
:FREQUENCY	<numeric_value>	Modulation frequency of the internal PWM source. *RST default is 2500hz.
:POWER:AMPLitude?	<numeric_value>	Query only. *RST default may vary (based on model and POWER:FREQUENCY).
:POWER:ATTenuator	<numeric_value>	*RST default is 31 (MAX). Range is 0-31.
:POWER:FREQUENCY	<numeric_value>	*RST default is 1khz.
:SOURce	EXTernal PWM	*RST default is EXTernal
:STATe?	<boolean>	Query only.

Table 16 ASK Commands

AM:DUTY

Used when AM:SOURce = PWM, this command sets the percentage of time the internal modulating signal is low (selects lower power).

AM:FREQUENCY

Used when AM:SOURce = PWM, this is the frequency of the modulating signal when internal modulation is used.

AM:POWER:AMPLitude?

This query returns the output power level (dBm) that would occur for a frequency of POWER:FREQUENCY if no attenuation was used. The ASK dBm range is therefore POWER:AMPLitude to (POWER:AMPLitude - 31dB).

AM:POWER:ATTenuator

Provides raw access to the 5-bit attenuator value. When AM:STATe is OFF, this value can be read/written but it is not actually used to control attenuation. Power compensation is automatically performed (the OUTPUT subsystem defines the output power level).

When AM:STATe is ON, this value is written directly to the attenuator hardware, giving an attenuation of 0 to 31 dB. No compensation is automatically performed (the OUTPUT subsystem does not define the output power level).

When AM:STATe is ON the attenuator is partially controlled by the ASK signal (internal or external). One of two powers are output depending on the state of the ASK signal. ASK overrides bit 4 of the attenuation value, so attenuation increases by 16dB when the ASK signal is high (0x10 = 16 decimal).

As an example, if this parameter was set to 7 then attenuation would be 7dB when ASK is low, 23dB when ASK is high. When ASK is enabled, bit 4 of this parameter is ignored because ASK overrides that bit (e.g. programming a 7 is same as 7+16).

AM:POWer:FREQuency

See previous description for AM:AMPLitude.

AM:SOURce

This parameter is only used when AM:STATe is ON, when the user has complete control over the entire 0-31dB attenuation range (no automatic compensation is performed).

AM:SOURce may be changed even when AM:STATe is ON. The ASK signal is automatically muxed to internal or external operation based on the SOURce setting.

- EXTERNAL - use external ASK signal to control the attenuator (ASK signal is automatically muxed to the user header).
- PWM - use an internal PWM signal to control the ASK signal using the AM:DUTY and AM:FREQuency parameters.

External operation is completely user controlled.

PWM can be done at extremely fast rates, but is a simple repeating signal that is high/low for a set period of time.

Note that the LIST subsystem can do arbitrary power control with multiple power levels and arbitrary timing. LIST is very flexible but the rate is limited because it is a software driven operation.

AM:STATe

Setting ON activates modulation.

CM Subsystem

This subsystem contains the commands used to define chirp modulation.

FREQuency and CM subsystem parameters should be written before setting CM:STATe = ON to start the operation.

Once started, no chirp-related parameters can be changed, but chirp may be stopped by setting CM:STATe = OFF.

The SFORge:FSKMux:INTernal parameter determines whether the modulation signal is generated internally or externally. It is shared by these subsystems: CM / FM / OM / PM. It must be set to INTERNAL for chirp operations (before setting STATE ON). The default is INTERNAL so it need not be programmed unless another subsystem changed the setting.

A chirp consists of three phases:

- pulse – the frequency is ramping up or down
- idle 1 – the final output frequency is output
- idle 2– the first output frequency is output

The pulse time is specified by CM:PULSe. Idle1 and 2 times are specified by CM:IDLE#:TIME.

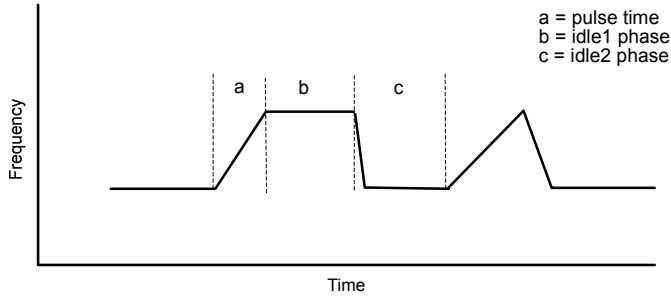


Figure 7. Chirp Operation

Non-Triggered Operation

The SYNC output goes high after the first frequency is output, stays high during the pulse phase, and goes low after the last frequency is output. Idle 1 phase is now entered.

Idle 1 phase is where the final output frequency is held for a time defined by CM:IDLE1:TIME. However, if CM:IDLE1:ENABLE is false then idle 1 is skipped.

Idle 2 phase is where the first output frequency is set and held for a time defined by CM:IDLE2:TIME. However, if CM:IDLE2:ENABLE is false then idle 2 is skipped.

The entire chirp sequence = CHIRp:PULSe + CHIRp:IDLE1:TIME + CHIRp:IDLE2:TIME.

Triggered Operation

In triggered operation (TRIGger:STATe = ON before CM:STATe = ON) the chirp is started by a trigger event. Idle 1 and 2 times are performed the same as non-triggered operation.

If the external trigger signal is enabled the SYNC signal is only high when waiting on a trigger. If the external trigger signal is not enabled SYNC works as described for non-triggered operation.

If alert is enabled (TRIGger:ALERT) the alert message is sent before looking for the trigger.

Command	Parameters	Comments
:ALERT	<boolean>	*RST default is OFF.
:IDLE#:ENABle	<boolean>	*RST default is OFF.
:IDLE#:TIME	<numeric_value>	*RST default is 10ms.
:PULSe	<numeric_value>	*RST default is 1ms.
:REVerse	<boolean>	*RST default is OFF.
:STATe	<boolean>	ON starts chirp modulation. *RST default is OFF.

Table 17 CM Commands

The control program specifies all chirp parameters - they are not automatically calculated. This allows the control program the maximum flexibility in controlling the frequency ramp characteristics.

In all chirp operations:

FREQ:DWELl sets the time of each frequency step.

FREQ:STEP sets the frequency change for each step.

FREQ:STARt sets the starting frequency (FREQ:STOP is not used)

CM:REVerse sets the direction (OFF means ascending frequency)

The ending frequency is determined by the dwell, step, and pulse time. The programmer should generate values that keep the frequency inside the min/max frequency values determined by the FREQUENCY:RANGE setting.

CM:ALERT

When ON an alert message is sent at the end of the idle 2 phase (before waiting on the trigger). The alert is a 2-byte message consisting of '!' and <linefeed>.

CM:IDLE#:ENABLE

Enables using the time for idle phase 1 or 2. The “#” is either 1 or 2, where 1 selects idle 1 and 2 selects idle 2. The “#” is optional – if not used then IDLE1 is assumed.

When OFF, the associated idle phase is skipped. When ON, the associated idle phase time is set by CM:IDLE#:TIME.

CM:IDLE#:TIME

This specifies the time for idle phase 1 or 2. The “#” is a header suffix, and should be either 1 or 2 where 1 selects idle 1 and 2 selects idle 2. The “#” is optional – if not used then IDLE1 is assumed.

CM:PULSE

Specifies how long each chirp lasts.

CM:REVERSE

When ON the chirp decreases in frequency instead of increases.

CM:STATE

This command activates (ON) or deactivates (OFF) chirp operation. Before setting ON, FREQUENCY:MODE must be CM. Setting ON locks FREQUENCY:MODE at it's current setting. When STATE goes OFF, FREQUENCY:MODE is released (unless another operation also locked it).

Chirp Operation for High Frequencies

When using range 1 with the TTL or single-ended output (roughly 0-100mhz), the frequency is agile – it can instantly jump from the last frequency to the first frequency. The following description does not apply to these cases.

When using any other output or range, all frequency changes are a ramp. This is because the agile (instant) change of a reference frequency must be tracked by a downstream PLL. The result is that the starting frequency may not be reached before the next pulse begins (the ending frequency would not be affected). This is a consideration for idle phase 2 where the frequency is tracking back towards the starting frequency.

The amount of lost frequency (if any) depends on:

- PLL tracking speed (related to lock time)
- difference between the end frequencies of the chirp
- idle 2 phase time

There are two ways to deal with this issue:

- Make sure the idle 2 phase time is long enough to allow tracking back to the starting frequency.
- Adjust the starting frequency (and/or step frequency) to account for the “lost” frequency.

It is possible to account for the “lost” frequency by entering a slightly lower starting frequency for a rising chirp or higher starting frequency for a falling chirp. You can use the PLL's stated track time to calculate the time it would take to ramp over a shorter range (i.e. the range of your chirp).

Example Using Simple Chirp

Program a 4ms pulse with idle 1 and 2 times disabled.

Chirp from 300 MHz to 400 MHz in 4ms.

This means you are using range 3 (200-400 MHz). Since you are doing 1/2 the possible frequency range, you would expect the track time in this example to be 500us. You need to lower the starting frequency by a number to represent 500us:

1. Ramp from 300-400 MHz in 4ms = 100 MHz/4ms = 25 MHz/ms = 25 KHz/us.
2. 25 KHz/us * 500us lock time = 12.5 MHz lost, so you need to subtract 12.5 MHz from the starting frequency.
3. You have now borrowed from 12.5 MHz from the ending frequency. You can compensate for this by increasing the step size or reducing the dwell time by 500us/4ms = or 0.125 – i.e. (step * 1.125) or (dwell * (1-0.125)).

Of course, the most accurate method is to use a spectrum analyzer to change F1 until the desired range is seen.

FM Subsystem

This subsystem contains the commands used to define FSK (frequency shift keying) modulation.

There are two basic modes of operation:

- When FM:MODE = NONRamped, the output is two discrete frequencies defined by FREQUENCY:START and :STOP selected by the FSK_BPSK signal).
- When FM:MODE = RAMPed, the output is a ramp whose direction is controlled by the FSK_BPSK signal. RAMPed operation is defined by the following parameters:
 - FREQUENCY:DWELL sets the time of each frequency step.
 - FREQUENCY:STEP:INCRement sets the frequency step size
 - FREQUENCY:START and :STOP set the starting and ending frequencies.
- When FM:MODE = TRIangle means the signal ramps from FREQ:START to FREQ:STOP under automatic hardware control. START may be larger than STOP. There is no modulating signal. DWELL and STEP are used to define the ramp.

Control of the FSK_BPSK signal can occur in three ways: EXTERNAL, LIST, or PWM.

If the internal PWM signal is used (FM:SOURCE = PWM), the following parameters define the operation of the PWM signal:

- FM:DUTY
- FM:FREQUENCY

Command	Parameters	Comments
:DUTY	<numeric_value>	Sets the percentage of time the internal PWM modulating signal is low. *RST default is 50.
:FREQUENCY	<numeric_value>	Modulation frequency of the internal PWM source. *RST default is 2500hz.
:MODE	RAMPed NONRamped TRIangle	*RST default is NONRamped.
:SOURCE	EXTERNAL LIST PWM	*RST default is EXTERNAL
:STATE	<boolean>	ON enables modulation. *RST default is OFF.

Table 18 FM Commands

FM:DUTY

Used when FM:SOURCE = PWM, this command sets the percentage of time the internal modulating signal is low (selects FREQUENCY:START). May be modified even when STATE is ON.

FM:FREQuency

Used when FM:SOURce = PWM, this is the frequency of the modulating signal when internal modulation is used. May be modified even when STATE is ON.

FM:MODE

Determines whether to do ramped or non-ramped FSK. May be modified even when STATE is ON.

- FSKNonramped means there are two discrete output frequencies. The modulating signal selects between frequencies.
- FSKRamped means there is a frequency ramp from FREQ:START to FREQ:STOP controlled by the modulating signal (it's polarity selects the direction of change). FREQuency:DWELL and FREQuency:STEP are used to define the ramp.
- TRlangle is like FSKRamped except the frequency ramps back-and-forth automatically (there is no modulating signal). When an endpoint frequency is reached (start or stop) the frequency ramp immediately reverses direction.

For FSKNonramped, the FSK_BPSK control signal selects the START frequency when low. The START frequency may be lower than the STOP frequency.

For FSKRamped, the frequency ascends when the FSK_BPSK signal is high, descends when low (until START or STOP is reached).

FM:SOURce

Selects modulation source. May be modified even when STATE is ON.

- EXTERNAL means use external signal (FSKmux must be set to external before selecting).
- LIST means use the LIST subsystem to modulate the OOK signal (FSKmux must be set to internal).
- PWM means use an internal PWM signal to control the OOK signal (FSKmux must be set to internal).

External operation is completely user controlled. PWM can be done at extremely fast rates, but is a simple repeating signal that is high/low for a set period of time. LIST is very flexible but the rate is limited because it is a software driven operation.

FM:STATE

Setting ON activates modulation. FREQuency:MODE must be set to FM before STATE is set ON, which locks FREQuency:MODE until FM:STATE goes OFF.

FREQuency Subsystem

This subsystem contains the commands used to define the frequency settings for the selected output. It also contains common frequency parameters used by other subsystems when MODE is SWEep, FM (FSK), or CM (chirp).

Command	Parameters	Comments
:DWELL	<numeric_value>	*RST default is MIN. MIN is 100ns. MAX is 3,495,200ns.
:FIXed	<numeric_value>	*RST default is 1khz.
:MODE	FIXed CM FM SWEep	*RST default is FIXED.
:RANGe	<numeric_value>	*RST default is 1.
:START	<numeric_value>	*RST value is 1khz.
:STEP:INCRement	<numeric_value>	*RST default is 100khz.
:STOP	<numeric_value>	*RST value is 102mhz.

Table 19 Frequency Commands

The MIN and MAX value for all FREQUENCY parameters depends on the selected output and range. The frequency range for each output is detailed in the OUTPUT subsystem.

A constant frequency is output using FREQUENCY:FIXed <numeric_value> when FREQUENCY:MODE = FIXed.

The following parameters are common for all multiple frequency operations such as sweep, FSK, and chirp:

- DWELI (not for sweep)
- STEP
- STARTt
- STOP

The frequency range for multiple frequency operations is defined by STARTt and STOP. Note that it is not always valid to have STARTt > STOP, and these cases are listed under each description that uses a frequency range.

FREQUENCY:FIXed

This parameter sets the frequency of the output signal when FREQUENCY:MODE = FIXed or CM (chirp).

FREQUENCY:DWELI

This sets the length of time a frequency is output during a frequency ramp. Used for operations where hardware automatically varies the frequency: FREQUENCY:MODE = FM (FSK) or CM (chirp).

Note that dwell time for sweep is set under the sweep subsystem.

FREQUENCY:MODE

Determines operation for the frequency subsystem.

The settings have the following meanings:

- FIXed: The frequency is determined by FREQUENCY:FIXed.
- CM: The operation is chirp modulation as defined by the CM subsystem.
- FM: The operation is defined by the FM subsystem (FSK).
- SWEEp: The frequency is swept as defined by the SWEEp subsystem.

FM subsystem operations (FSK) use these parameters:

FREQUENCY:DWELI sets the time of each frequency step.

FREQUENCY:STEP sets the frequency change for each step.

The frequencies for multiple frequency operations are determined by FREQUENCY:STARTt and STOP.

Example: FREQUENCY:MODE SWE

sets the SWEEp mode. The settings under FREQUENCY:STARTt and STOP become effective.

When MODE is FM, CM, or SWEEp the associated subsystem must be programmed and it's STATEt set ON to begin modulation.

FREQUENCY:RANGe

Selects the operating frequency range for all commands and forces the STARTt and STOP values to define the range. This command is rejected with -221, "Settings conflict" when FREQUENCY:MODE is not FIXED (query is OK).

The frequency ranges stated in the following table are as of October 2008, but they are subject to change. A good programming method is to query the device ranges (perhaps only once at startup).

Range	Output 1 Single-ended	Output 2 Differential	Output 3 TTL
1	1khz – 102mhz	50mhz – 102mhz	1khz – 110mhz
2	98mhz – 204mhz	98mhz – 204mhz	invalid
3	196mhz – 408mhz	196mhz – 408mhz	invalid
4	392mhz – 816mhz	392mhz – 816mhz	invalid
5	784mhz – 1ghz	784mhz – 1ghz	invalid

Table 20 Frequency Ranges

Range	2500M	1800M
1	1.5ghz – 2.6ghz	950mhz – 1.8ghz

Table 21 Module Freq Range

Determining Ranges

You can determine the ranges as follows:

- Use OUTPUT:SElect to select an output (SE, TTL, DIFF, or MODULE).
- Use the query FREQUENCY:RANGE? MAX to get the maximum range number.
- Set the range using FREQUENCY:RANGE <range number>.
 - Query FREQUENCY:FIXed? MIN, then MAX to get the frequency range.
 - Repeat for all ranges.

FREQUENCY:START

This command sets the start frequency for multiple frequency operations (sweep, FM/FSK, CM/chirp). START can be greater than STOP unless otherwise specified by the operation that uses the values (e.g. sweep requires START is smaller).

Normally the frequency allowed is based on FREQUENCY:RANGE but if FREQUENCY:MODE is SWEep then the frequency spans all the ranges (MIN and MAX query also expands).

FREQUENCY:STEP:INCREMENT

This sets the frequency change between frequencies in a ramp. Applies to FM (FSK operation), CM (chirp operation), and sweep.

FREQUENCY:STOP

This command sets the stop frequency for multiple frequency operations (sweep, FSK, chirp). STOP can be less than START unless otherwise specified by the operation that uses it (e.g. sweep requires START smaller than STOP).

Normally the frequency allowed is based on FREQUENCY:RANGE but if FREQUENCY:MODE is SWEep then the frequency spans all the ranges (MIN and MAX query also expands).

LIST Subsystem

The LIST subsystem commands are similar to SCPI but they are not compatible.

The list subsystem controls automatic sequencing through a list of user-defined points that implement one or more operations such as frequency or power. This instrument supports only one LIST.

A list is an array of points, each point having one or more components that describe what operations the list performs. Each point consists of:

- one or more OPERation components. These determine what operations are performed at each point (e.g. frequency, power, or phase)
- zero or more CONTRol components. These determine what control operations are performed at each point (e.g. SYNC output toggle, trigger input)
- zero or more DWELL components. If dwell is a component, each point has it's own dwell time. If dwell is not a component, one dwell time is specified that applies to all points. Determined by LIST:MDWell.

A LIST definition is not affected by *RST, but LIST:STATe is set to OFF and list operation halts. The list is only cleared with a "LIST:POINT:INDex 0" command.

LIST Phases

The list subsystem is always in one of three phases: definition, data, or run.

Definition Phase

The definition phase is used to specify which operations are to be performed at each point. This phase is entered by sending a LIST:POINT:INDex command with data = 0. This resets the list (deletes all components), unlocks the list definition, and saves the current output and frequency range (OUTPut:SElect and FREQuency:RANGe).

During the definition phase, one command is sent for each

LIST:POINT:OPERation:xxxx and LIST:POINT:CONTRol:xxxx component that you want to add to the list. Sending a command (not a query) adds that component to the list, and means that each point will program the specified feature. The value sent for the command sets the default for that component. For example,

"LIST:POINT:OPERation:FREQuency 1e6" adds the frequency component and makes the default value 1e6 (1mhz).

Components that are not specified for use in the list are defined by their usual location. For example, if frequency is not a LIST component then the output frequency is specified by FREQ:FIXed.

The LIST:MDWell and LIST:FAST parameters should be set during this phase.

During this phase, commands that try to add invalid components are rejected with the following error:

213, "LIST component invalid"

For example, you cannot add a phase component when the range is anything other than 1, and the power component is valid only for the single-ended output.

Data Phase

The data phase is where the list components for each point are filled with parameters (e.g. frequency, dwell, trigger, SYNC output). This phase is entered from definition phase by sending a LIST:POINT:INDex command with data = 1.

Saving the list to NVRAM can only be done during the data phase.

If frequency is a component, the saved output and frequency range is checked against the current instrument setting when a frequency component is written (not when queried). If it does not match, a -221, "Settings conflict; incompatible output or range" error is generated. They are also checked when LIST:STATe goes on, and a mismatch generates the same error.

During this phase, accesses to LIST:POINT:OPERation:xxxx or

LIST:POINT:CONTRol:xxxx that are not components are rejected as follows:

- command returns 212," Not a LIST component"
- query returns the <NAN> value: 9.91E37

Run Phase

The run phase is where component data has been converted to values used to control the hardware. This phase is entered from the data phase by setting LIST:STATE = ON.

During this phase, accesses to LIST:POINt:OPERation:xxxx or

LIST:POINt:CONTRol:xxxx that are not components are rejected as follows:

- command returns: 212,"Not a LIST component"
- query returns <NAN>: 9.91E37

Programming LIST

The following steps are required to operate the instrument in List mode:

- Set LIST:STATe = OFF.
- Send "LIST:POINt:INDex 0" to reset the LIST and go to definition phase.
- Send one command for each desired operation under LIST:POINt:OPERation.
Example: LIST:POIN:OPER:FREQ 10000000.
- Send one command for each desired control under LIST:POINt:CONTRol.
Example: LIST:POIN:CONT:SYNC 1.
- Condition LIST:MDWell and LIST:FAST as desired.
- Send "LIST:POINt:INDex 1". This locks the list definition and moves to the data phase. The instrument now calculates the minimum dwell time (which can't be calculated until the desired operations are specified).
- Send LIST:POINt:DWELl to specify the dwell time (the only dwell time if MDWell is OFF). If DWELl is a component but not specified for a particular point, the minimum valid dwell is used.
- Fill the list with values. For example:

```
// program index 1 (first point)
LIST:POIN:DWEL 1234500
LIST:POIN:OPER:FREQ 11010
// since trigger is not specified, this point uses the
// default value you specified in the initial
// LIST:POIN:CONT:TRIG command

// program index 2
LIST:POINt:INDex 2
LIST:POIN:DWEL 67000
LIST:POIN:OPER:FREQ 12000
LIST:POIN:CONT:TRIG 1
```

The List has now been specified, and the settings will not change even with *RST. To start LIST operation, set LIST:STATe = ON.

LIST operations are run under software control, so accessing the instrument once operation has begun will pause at one point while the command is being processed. If LIST completion status is required you can use the SYNC output signal or an alert message to signal completion.

SYNC Signal

The SYNC output signal normally is toggled high/low just after the hardware is updated with the point parameters. If SYNC is enabled as a LIST component, the SYNC output signal is toggled high/low only for the points specified.

The trigger controls the SYNC signal only in these cases:

- SYNC is not a component
- The trigger is enabled as a LIST component
- The trigger source is external (TRIGger:SOURce)

If these are all true the SYNC signal is high only when waiting for a trigger.

LIST Commands

Command	Parameters	Comments
:FAST	<boolean>	Power-up (and LIST reset) default is OFF.
:HOLD	<boolean>	Power-up (and LIST reset) default is OFF.
:INFO		
:OUTPut?	(none)	query only
:RANGe?	(none)	query only – no MIN MAX
:SIZE?	(none)	query only – no MIN MAX
:JUMP	<NR1>	Power-up (and LIST reset) default is 1.
:MDWell	<boolean>	Power-up (and LIST reset) default is OFF.
:PHASe	DEF DATA RUN	query only
:POINt:INDex	<NR1>	Power-up (and LIST reset) default is 0.
:POINt:OPERation		
:DWEll	<numeric_value>	Default is minimum based on operation selected. Max is 1.6 seconds.
:FREQuency	<numeric_value>	default specified by user
:PHASe	<numeric_value>	default specified by user
:POWEr	<numeric_value>	default specified by user
:FM	<boolean>	default specified by user
:OM	<boolean>	default specified by user
:PM	<boolean>	default specified by user
:POINt:CONTRol		
:ALERt	<boolean>	default specified by user
:SYNC	<boolean>	default specified by user
:TRIGger	<boolean>	default specified by user
:RCL	(none)	command only
:SAVE	(none)	command only
:STATe	<boolean>	*RST default is OFF.

Table 22 List Commands

LIST:FAST

When OFF, any logical combination of POINt:OPERation and POINt:CONTRol components can be used, as well as JUMP and HOLD.

When ON, this is a request for the fastest possible operation (FAST mode) - see Specifications section for minimum dwell times. FAST mode is limited as follows:

- select only one POINT:OPERation component
- select no POINT:CONTRol components
- LIST:MDWell must be OFF (use the same time for every point)
- no JUMP or HOLD commands are allowed (rejected with -221, "Settings conflict").

The state of this parameter is checked when the phase transitions from definition to data phase (:POINT:INDEX goes from 0 to 1).

LIST:HOLD

Used when LIST:STATe is ON to suspend sequencing. OFF means sequence through the LIST as programmed. ON means stay at the current point regardless of trigger.

When HOLD is ON no ALERT messages are sent. ALERT messages are temporarily disabled before sending the response for the HOLD = ON command. Alerts will resume after setting LIST:HOLD to OFF.

May be set before setting LIST:STATe ON so the first point is held when the LIST is started.

LIST:INFO:OUTPut?

This query returns the OUTPut:SElect parameter at the time the list moved from definition to data phase. If a LIST component requires a specific output then this value is checked when LIST:STATe goes ON.

This parameter is set to the current instrument setting when the list is reset.

LIST:INFO:RANGe?

This query returns the FREQuency:RANGe parameter value at the time the list moved from definition to data phase. If frequency is a LIST component then frequencies are limited to those defined by the output and frequency range. This value is checked when LIST:STATe goes ON.

This parameter is set to the current instrument setting when the list is reset.

LIST:INFO:SIZE?

This query returns the number of points in the LIST (which is the highest valid index number).

LIST:POINT:JUMP

Queries or sets the current LIST point.

When LIST:FAST is ON or LIST:STATe is OFF commands are rejected with -221, "Settings conflict".

The query returns the current point.

The command sets operation to the next point. You can use this with HOLD to move among points in any order.

LIST:PHASe?

This query returns the current list phase:

- DEF is the definition phase.
- DATA is the data phase.
- RUN is the run phase.

LIST:POINT:CONTRol

Each command under this path specifies a control operation that takes place at each point:

- :ALERT ON sends a 2-byte alert message consisting of '!' and <linefeed>. The alert message will never be embedded inside a command/query response and can be temporarily disabled by setting HOLD = ON.
- :SYNC The SYNC output is normally low. When SYNC is ON for a point the SYNC output is toggled high then low just after the new point parameters have been updated in hardware (e.g. frequency).
- :TRIGer ON means wait for a trigger event. The dwell time is performed before the trigger is looked for. This allows a level trigger to be used as a LIST throttle.

CONTRol parameters may be changed during the run phase.

Note that when alerts are used, the minimum LIST dwell time is 180us. This is because at 115200 baud a byte time is about 90us.

During the data or run phase access to a CONTRol parameter that is not a component is rejected as follows:

- command returns: 212,"Not a LIST component"
- query returns <NAN>: 9.91E37

LIST:POINT:MDWell

MDWell ("multiple dwells") ON means a dwell component is specified for every point. OFF means the same dwell is used for every point.

A query can occur at any time. A command is only allowed when the list is in the definition phase, else returns error -221,"Settings conflict;invalid list phase".

LIST:POINT:INDex

This is used to specify which point to access for reading or writing parameters.

When LIST:STATe is OFF, sending a POINT:INDex command with parameter value 0 resets the list and begins the definition phase. You then specify which components to use as previously described. A transition to the data phase occurs by sending a POINT:INDex command with parameter value 1.

During the data phase, use the query "LIST:POINT:INDex? MAX" to find the largest allowable point number (max LIST size).

During the data phase, use the POINT:INDex command to select which point you want to specify values for. The command with the highest index number specifies the LIST size (number of points) even if no OPERation or SIGNal parameters are specified (uses the defaults).

During the run phase, use the POINT:INDex command to select which parameter to query / modify.

Writing an invalid index value returns the following error:

211,"LIST index error"

LIST:POINT:OPERation

Each command under this path specifies an operation (and value) that takes place at each point:

- :DWELl specified dwell time.
- :FREQuency specifies output frequency.
- :PHASe specifies output phase.
- :POWer specifies dBm level.
- :FM drives the FSK signal hi or low (ON or OFF). This works in conjunction with the FM subsystem.
- :OM drives the OOK signal hi or low (ON or OFF). This works in conjunction with the OM subsystem. The specific OOK signal driven (SE_OOK or DIFF_OOK) depends on the selected output.
- :PM drives the BPSK signal hi or low (ON or OFF). This works in conjunction with the PM subsystem.

Dwell differs from the other operation components in that the dwell command cannot be sent during the definition phase or a -221, "Settings conflict" error is generated. Dwell can always be queried during the data or run phase (if dwell is not a LIST component a query returns the single dwell value used for all points).

NOTE: When dwell is queried during the run phase, the reported value is unlikely to be exactly what was originally programmed because the calculated value depends on the actual timer values. The reported value will be within 1:256 of the programmed value.

During the run phase OPERation parameters may be queried but not written. Commands are rejected with error 214, "Invalid LIST phase".

During the data or run phase access to a parameter that is not a component is rejected as follows:

- command returns: 212, " Not a LIST component"
- query returns <NAN>: 9.91E37

During the run phase the query value for a frequency component may be off by 0.5ppm. This is because the 48-bit math required to create the hardware values is not implemented in reverse. For example, if you set a frequency of 10mhz it may be reported as 10mhz +/- 5hz. The actual output will still be 10mhz.

Consideration for POver and FREQUENCY components

The POver (attenuation) hardware value is set based on a frequency. The attenuation required to keep the power output level may change by as much as 2dB over a given frequency range.

When power and frequency are both LIST components (or both not LIST components) then the power is compensated correctly.

If only one of power or frequency is a LIST component then the actual output power may be off by as much as 2dB more than the usual limits.

If FREQUENCY is a LIST component, but not POver, attenuation (power level) is set based on the FREQUENCY:FIXed parameter. The least amount of power error can be obtained by programming FREQUENCY:FIXed with the midpoint LIST frequency = (min + max)/2 + min. FREQUENCY:FIXed should be written before setting LIST:STATE = ON.

If POver is a LIST component, but not FREQUENCY, attenuation is set based on the FREQUENCY:FIXed parameter when the data phase is entered from definition phase. If the output frequency is not changed after this time, the power level will meet the usual limits.

LIST:RCL

Loads a saved list from EEPROM and puts the list into data phase (can read/write components). Can only be done when LIST:STATE is OFF.

If list cannot be read returns:

- 215, "LIST definition lost" if there was an error reading the definition structure (which defines what is contained in the data array)
- 216, "LIST data lost" if there was an error reading the component data array.

LIST:SAVE

Saves a list to EEPROM. Can only be done during the data phase.

LIST:STATE

Set ON to start LIST operation.

The following indicates requirements when LIST:STATE transitions to ON.

- If using OPERation:FREQUENCY component

- FREQ:MODE = FIXED
- The output and range are the same as when the list was created. This is done to assure the frequencies are valid (FREQ:MODE is locked while LIST:STATe is ON).
- If using OPERation:PHASe component
 - PM: STATe = OFF
 - The output is TTL, or single-ended range 1.
- If using OPERation:POWer component
 - AM:STATe = OFF
 - The output is single-ended.
- If using OPERation:FM component
 - FM:SOURce = LIST
 - FM:STATe = ON
- If using OPERation:OM component
 - OM:SOURce = LIST
 - OM:STATe = ON
 - The output is single-ended range 1, or differential.
- If using OPERation:PM component
 - PM:SOURce = LIST
 - PM:STATe = ON
- If using CONTRol:TRIGger component
 - TRIGger:STATe = ON

OM Subsystem

This subsystem contains the commands used to define OOK (On/Off Keying) modulation. OOK is possible using the single-ended output in range 1 (up to 102mhz), or the differential output (all ranges).

When doing OOK with the single-ended output the OUTPut:OSKRamp parameter defines how fast the output power ramps up/down

There are two OOK signals that can be controlled internally or externally: DIFF_OOK and SE_OOK. DIFF_OOK turns the differential output on/off. SE_OOK turns the single-ended output on/off. Signal low is the OFF condition.

The SFORge:FSKmux:INTernal parameter selects internal or external control of these signals.

If external control is selected DIFF_OOK is always active and should be kept low unless the differential output is selected. SE_OOK is only active when OM:STATe is ON.

Command	Parameters	Comments
:DUTY	<numeric_value>	Sets the percentage of time the internal PWM modulating signal is low. *RST default is 50.
:FREQuency	<numeric_value>	Modulation frequency of the internal PWM source. *RST default is 2500hz.
:SOURce	EXTERNAL LIST PWM	*RST default is EXTernal
:STATe?	<boolean>	Query only.

Table 23 OM Commands

OM:DUTY

Used when OM:SOURce = PWM, this command sets the percentage of time the internal modulating signal is low (output OFF). May be changed when STATE is ON.

OM:FREQuency

Used when OM:SOURce = PWM, this is the frequency of the internal modulating signal. May be changed when STATE is ON.

OM:SOURce

When FSKmux is set to internal (the *RST default) and OM:STATe is ON, this determines how the internal modulation signal is controlled. May be changed when STATE is ON.

LIST - use the LIST subsystem to modulate the OOK signals

PWM - use an internal PWM signal to control the OOK signals

External operation (FSKmux set to external) is completely user controlled. PWM can be done at extremely fast rates, but is a simple repeating signal that is high/low for a set period of time. LIST is very flexible but the rate is limited because it is a software driven operation.

OM:STATe

When ON this parameter enables OOK (on/off keying) modulation. OOK can be done for single-ended range 1 or the differential output (all ranges).

A unique case is when the differential output is selected and the FSKmux is set to external. In this case, the output is always under user control regardless of this parameter.

When OM:STATe is set ON, the output must be enabled (OUTPut:STATe = ON) or the command is rejected with -221, "Settings conflict;OUTP:STAT must be ON".

When OM:STATe is ON OUTPut:STATe commands are rejected with -221, "Settings conflict;OM:STATe ON".

PM Subsystem

This subsystem contains the commands used to define phase modulation for BPSK operation. BPSK operation is where a modulating signal selects between two phases.

The BPSK modulation signal is internal or external as defined by the FXKmux setting.

Command	Parameters	Comments
:DUTY	<numeric_value>	Sets the percentage of time the internal PWM modulating signal is low. *RST default is 50.
:FREQuency	<numeric_value>	Modulation frequency of the internal PWM source. *RST default is 2500hz.
:PHASe2		
:SOURce	EXTernal LIST PWM	*RST default is EXTernal
:STATe	<boolean>	ON enables modulation. *RST default is OFF.

Table 24 PM commands

PM:DUTY

Used when PM:SOURce = PWM, this command sets the percentage of time the internal modulating signal is low (selects default phase). May be changed when STATE is ON.

PM:FREQuency

Used when PM:SOURce = PWM, this is the frequency of the internal modulating signal. May be changed when STATE is ON.

PM:PHASe2

When BPSK is enabled (PM:STATe = ON), this parameter defines the phase offset for the second output phase. The FSK_BPSK signal selects between the default phase (0) and PHASe2. The default phase is output when the BPSK signal is low, PHASe2 when high.

PM:SOURce

Determines the type of modulation. Can be changed even when PM:STATe is ON.

- EXTernal means use external signal (FSKmux must be set to external).
- LIST means use the LIST subsystem to modulate the BPSK signal (FSKmux must be set to internal).
- PWM means use an internal PWM signal to control the BPSK signal (FSKmux must be set to internal).

External operation is completely user controlled. PWM can be done at extremely fast rates, but is a simple repeating signal that is high/low for a set period of time. LIST is very flexible but the rate is limited because it is a software driven operation.

PM:STATe

Setting ON activates modulation and locks FREQuency:MODE at it's current setting. When it goes OFF, FREQuency:MODE is released (unless another operation also locked it).

POWER Subsystem

This subsystem contains the commands for setting the output power level for the single-ended or module output. The AM subsystem controls amplitude modulation.

Command	Parameters	Comments
:LEVel:IMMediate:AMPLitude	<numeric_value>	Output power (dBm). *RST default is MIN for the device's default output.

Table 25 Power Commands

POWER:LEVel:IMMediate:AMPLitude

The command sets the RF output level (when AM:STATe = OFF). If LIST is programmed to control power this parameter is accepted and reported but the LIST subsystem controls the actual power level (when LIST:STATe is ON). The same applies for ASK operation (AM:STATe ON).

Example: POW:LEV:IMM:AMPL 3

sets the single-ended output to 3 dBm.

SWEep Subsystem

Similar to SCPI FREQuency:SWEep but since we only sweep frequency the FREQuency path member is removed.

This subsystem contains the commands for defining frequency sweeps. Frequency sweep is enabled by setting FREQuency:MODE to SWEep. Frequency sweep is similar to FM (FSK) but allows dynamic firmware and/or trigger control, the ability to run at slower rates, and control of the SYNC output signal.

Command	Parameters	Comments
:ALERt	<boolean>	*RST default is OFF
:BIDirectional	<boolean>	*RST default is OFF
:DOWN	<boolean>	*RST default is OFF.
:DWELI	<numeric_value>	

:FREQUENCY	<numeric_value>	default is minimum frequency for the selected range
:MODE	FREerun HOLD RESet SINGle STEP	*RST default is FREerun.
:STATE	(none)	*RST default is OFF.
:SYNC:STEP	<boolean>	*RST default is OFF.

Table 26 Sweep Commands

- FREQUENCY:START and STOP define the outer range of the sweep. No frequency outside the range will be output. START must be smaller than STOP (checked when STATE goes ON).
- FREQUENCY:STEP sets the frequency jump at each step. This must be set such that at least 2 frequencies would be output during the sweep, else a settings conflict is returned when sweep is started (SWEep:STATE is set ON).
- FREQUENCY:START, STOP, and STEP cannot be changed once sweeping has started, else a -221, "Settings conflict" is returned.
- Sweep subsystem parameters may be changed even while sweeping.

NOTE: You should set FREQUENCY:MODE to SWEep before querying (or programming) the min/max frequency for FREQ:START, FREQ:STOP, and SWEep:FREQUENCY. The mode is used as a flag to allow the frequency to span across all ranges. If FREQUENCY:MODE is not set to SWEep then the frequency allowed is based on the range (FREQUENCY:RANGe).

A frequency sweep is programmed as shown below:

1. *RST to put instrument into the default state (selects single-ended output).
2. Select output and frequency range, then set the sweep boundaries.

```
FREQUENCY:RANGe 3           // range 3
FREQUENCY:START 200000000   // 200 mhz
FREQUENCY:STOP 240000000   // 240 mhz
```

3. Set the step width and DWELL time.

```
FREQUENCY:STEP 1000000     // 1 khz
SWEep:DWELL 12000000      // 12 ms
```

4. Select the trigger for use by sweep (if desired).

```
TRIGger:STATE ON
```

5. Select the sweep mode and activate the sweep.

```
SWEep:MODE SING
FREQUENCY:MODE SWEep
SWEep:STATE ON
```

SWEep:ALERT

When ON, an alert message is sent after the last frequency in a sweep has been output (and the step time completed). The alert message is a 2-character sequence of '!' and <linefeed>. Can be modified while sweep is running.

SWEep:BiDirectional

The OFF state means a sweep goes from START to STOP, then starts over at START. ON means there are two sweeps defined, first START to STOP, then STOP to START (assuming DiRection is UP). Can be modified while sweep is running.

When **BIDirectional** is ON and the range defined by **START/STOP** is not an exact multiple of the **STEP**, the last frequency of the sweep is not output. This is because once the first frequency of the sweep is output (**START** or **STOP** depending on **DOWN**), we move by **STEP** hertz until the sweep completes. This means the final frequency can be a fraction of **STEP** from the end of the range. When the direction reverses, the same exact frequencies are output as the initial sweep.

SWEep:DOWN

Controls the direction of the sweep. When OFF the sweep is done from **START** to **STOP**. When ON the sweep is carried out from **STOP** to **START**. Can be modified while sweep is running.

If **BIDirectional** is ON, **DOWN** is automatically updated after a complete sweep.

SWEep:DWELI

Sets the time that each frequency is output. The dwell time is always elapsed before checking for a trigger. Can be modified while sweep is running.

SWEep:FREQuency

Can be used to set or query the current sweep frequency. If sweep is not running a min/max query will return the same values as **FREQuency:FIXed**. If sweep is running a min/max query will return the **FREQuency:START / STOP** values.

If used to set a new sweep frequency, the sweep will continue from the programmed value using the **STEP** size until a sweep is completed. The next sweep will begin with the **START** or **STOP** frequency.

Normally the frequency allowed is based on **FREQuency:RANGe** but if **FREQuency:MODE** is **SWEep** then the frequency spans all the ranges (**MIN** and **MAX** query also expands).

SWEep:MODE

Sets the sweep mode.

Operations can be done with or without the trigger subsystem, and may be dynamically changed to create the desired operation. Modes that define what to do when a trigger occurs require **TRIGger:STATe ON** (they are rejected as a settings conflict if OFF).

- **FREerun** – Ignore trigger (if enabled) and run until **MODE** changes.
- **HOLD** – Pause at the current frequency until **MODE** changes.
- **SINGle** – **TRIGger:STATe** must be ON. Each trigger causes one sweep. If **BIDirectional** is ON, a trigger is required for each direction.
- **RESet** – Output the first frequency. The **MODE** must then be changed to resume sweeping.
- **STEP** – **TRIGger:STATe** must be ON. Each trigger causes one frequency step.

A **SWEep:FREQuency** command can be sent while **MODE = HOLD** or **RESet** to output (and hold) any frequency in the sweep range.

SWEep:STATe

Set ON to begin sweep operation. Set OFF to stop sweep operation. **FREQuency:MODE** must be set to **SWEep** before turning **SWEep:STATe** ON.

Setting ON locks **FREQuency:MODE** at it's current setting. When **STATe** goes OFF **FREQuency:MODE** is released (unless another operation also locked it).

SWEep:SYNC:STEP

Defines how **SYNC** is output during sweep operations when not using the trigger (**TRIGger:STATe = OFF**).

SYNC is normally low (e.g. even before sweep sequence is enabled).

If TRIGger:STATe is OFF, SWEEp:SYNC:STEP controls SYNC as follows:

If OFF, SYNC goes high just after the first frequency is output (before delaying the dwell time). SYNC goes low just after the final frequency is output (before delaying the dwell time). This means SYNC will pulse low for one STEP time (low during the final output frequency).

If ON, SYNC pulses high/low just after each frequency is output (before delaying the dwell time).

When TRIGger:STATe is ON, SYNC is high only when waiting on the trigger.

Sweep Considerations

If sweep is programmed to cross frequency ranges there is an added delay (about 10ms) when the range boundary is crossed (this does not apply to the TTL or module outputs because they only have one range).

VOLTage Subsystem

Controls the TTL output level. For the AC output power see POWER.

Command	Parameters	Comments
:LEVel	1.8 2.5 3.3	Output voltage. *RST default is 1.8.

Table 27 Voltage Commands

VOLTage:LEVel

Sets the voltage level for the TTL output. This command is accepted regardless of the currently selected output (e.g. if single-ended output is currently selected).

STATus Subsystem

This system contains the commands for the status reporting system. Queries return the current value of the respective register, which permits a check of the device status.

This device supports more than the “Minimum Status Reporting Structure Required by SCPI”, and consists of the following registers:

- OPERation (all are 16 bits): Status /Condition, Event, Event Enable, Ptransition, Ntransition
- QUEStionable (all are 16 bits): Status /Condition, Event, Event Enable, Ptransition, Ntransition
- Standard Event Status (all are 8 bits): Status, Enable
- Instrument Summary (all are 8 bits): Status, Enable
- Error (message) Queue – described in section 0.

See illustration SCPI 1999.0 volume 1 figure 9-1.

The following apply to all status registers:

- The MS bit of a 16-bit register is always sent and received as 0.
- *RST has no effect on the status registers.

Some error messages require that the error must be eliminated before correct instrument operation can be ensured (e.g. calibration).

The parameters to the commands are <NRf>. The query form is always an <NR1> value.

Status System Programming

In general, a status group consists of a condition register, a transition filter, an event register, and an enable register. Each component is briefly described in the following paragraphs.

Condition Register

The condition register is continuously updated to reflect the current status of the instrument. There is no latching or buffering for this register, it is updated in real time. Reading the contents of a condition register does not change its contents.

Transition Filter

The transition filter is a special register that specifies which types of bit state changes in the condition register will set corresponding bits in the event register.

Negative transition filters (NTR) are used to detect condition changes from True (1) to False (0); positive transition filters (PTR) are used to detect condition changes from False (0) to True (1). Setting both positive and negative filters True allows an event to be reported anytime the condition changes. Transition filters are read-write. Transition filters are unaffected by queries or *CLS (clear status) and *RST commands.

The command STATus:PRESet sets all negative transition filters to all 0's and sets all positive transition filters to all 1's.

Event Register

The event register latches transition events from the condition register as specified by the transition filter. Bits in the event register are latched, and once set they remain set until cleared by a query or a *CLS command. Event registers are read only.

Enable Register

The enable register specifies the bits in the event register that can produce a summary bit. The instrument logically ANDs corresponding bits in the event and enable registers, and ORs all the resulting bits to obtain a summary bit. Summary bits are recorded in the Instrument Summary Status Register. Enable registers are read-write. Querying an enable register does not affect it. The command STATus:PRESet sets the Operational Status Enable register and the Questionable Status Enable register to all 0's.

Bits that are unused in the corresponding status register are also unused in the corresponding enable register. Unused bits are always written and read as 0 (even if written with bit set).

Enable registers are 0 on power up as required by IEEE488 11.4.2.3.4 which states that this is required if *PSC is not implemented or the power-on-status-clear flag is TRUE (10.26).

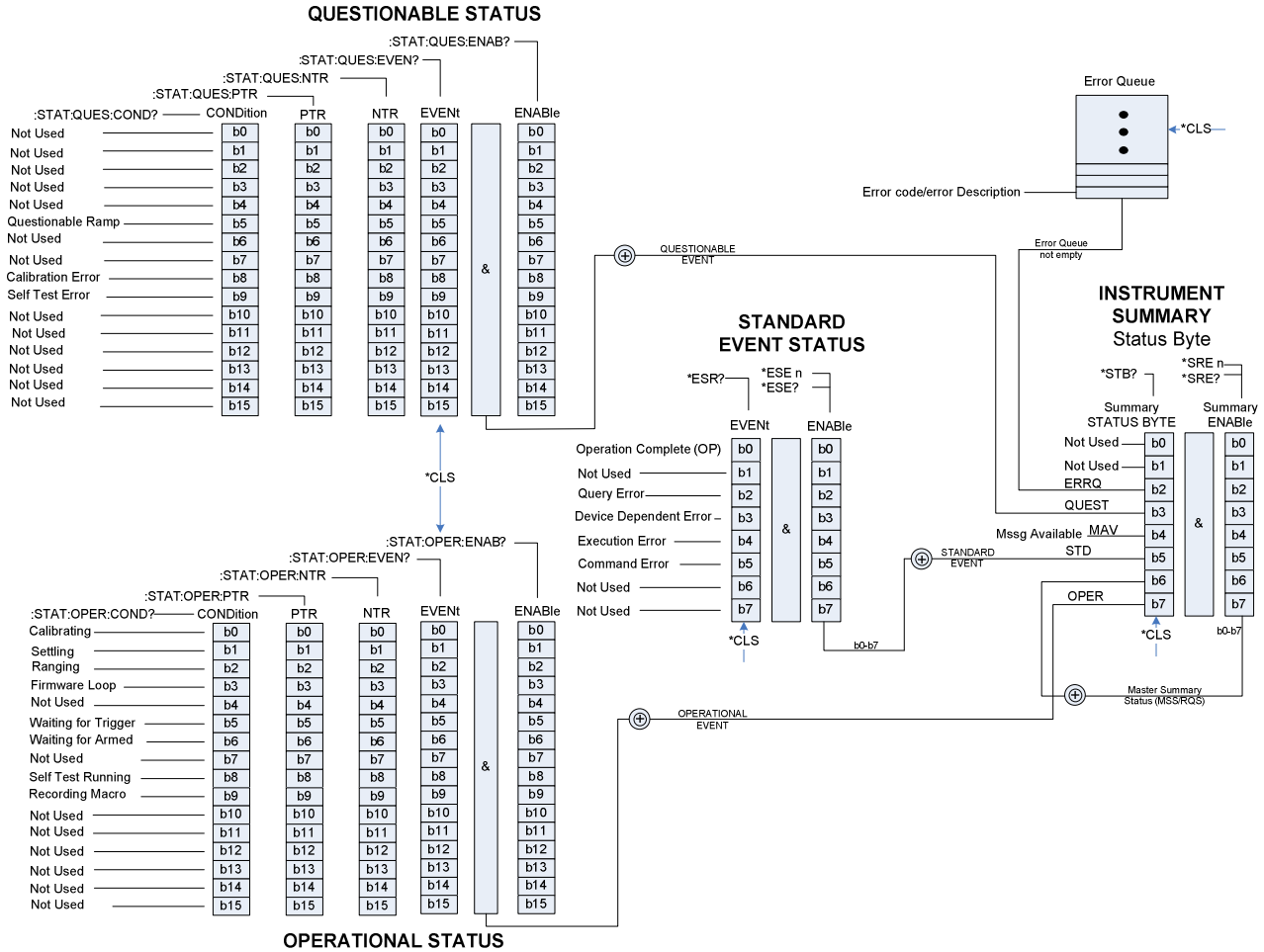


Figure 8. SCPI Status Registers

Status Group Reporting

The state of certain hardware and operational events and conditions can be determined by programming the status system. As shown in the SCPI Command Registers diagram above, the three lower status groups provide status information to the Instrument Summary Status Register group. The Instrument Summary Status Register group is used to determine the general nature of an event or condition and the other status groups are used to determine the specific nature of the event or condition.

The following paragraphs explain the information provided by each status group.

Instrument Summary Status Group

Reference SCPI section 9.

The Instrument Summary Status group, consisting of the Instrument Summary Status Enable Register and the Instrument Summary Status Register, is used to determine the general nature of an event or condition. The bits in the Instrument Summary Status Register (a.k.a. Status Byte) provide the following information (read using “*STB?”).

Bit	Description
0,1	Not Used. These bits are always set to 0.
2	Set to indicate the Error Queue contains data. The Error Query command can then be used to read the error message(s) from the queue.

3	Set to indicate the Questionable Status summary bit has been set. The Questionable Status Event register can then be read to determine the specific condition that caused the bit to be set.
4	MAV (message available) bit. Set to indicate that the instrument has data ready in its output queue. Since this instrument does not have an output queue this bit is always 0.
5	Set to indicate that the Standard Event Status summary bit has been set. The Standard Event Status register can then be read to determine the specific event that caused the bit to be set.
6	Set to indicate that the instrument has at least one reason to require service. This bit is also called the Master Summary Status Bit (MSS). The individual bits in the Instrument Summary Status Register are ANDed with their corresponding Instrument Summary Enable Register bits, then each bit value is ORed and input to this bit.
7	Set to indicate that the Operational Status summary bit has been set. The Operational Status Event register can then be read to determine the specific condition that caused the bit to be set.

Table 28 Summary Status Register

Standard Event Status Group

Reference SCPI section 9.

The Standard Event Status group, consisting of the Standard Event Status register (an Event register) and the Standard Event Status Enable register, is used to determine the specific event that set bit 5 of the Instrument Summary Status Register. The bits in the Standard Event Status register provide the following information (read using “*ESR?”).

NOTE: The Standard Event Status register is cleared by a query.

Bit	Description
0	Set to indicate that all pending operations were completed following execution of the “*OPC” command.
1	Not Used. The bit is always set to 0.
2	Set to indicate that a query error has occurred. Query errors have SCPI error codes from –499 to –400.
3	Set to indicate that a device-dependent error has occurred. Device-dependent errors have SCPI error codes from –399 to –300 and 1 to 32767.
4	Set to indicate that an execution error has occurred. Execution errors have SCPI error codes from –299 to –200.
5	Set to indicate that a command error has occurred. Command errors have SCPI error codes from –199 to –100.
6-7	Not Used. The bits are always set to 0.

Table 29 Event Status Register

Operational Status Group

Reference SCPI section 9.3.

The Operational Status group is used to determine the specific condition that set bit 7 in the Instrument Summary Status Register. The Operational Status group consists of the Operation Condition (Status) register, the Operation Positive Transition register, the Operation Negative Transition register, the Operation Event register, and the Operation Event Enable register.

Bits 8-12 of the Operation Condition register are for use by the designer. The Operation Condition register bits are defined in the following table (read using “STATus:OPERation:CONDition?”).

Bit	Description
0	Calibrating. See calibration subsystem.
1	Always 0. Settling.
2	Always 0. Ranging.
3	“Firmware Loop” is set to indicate that a firmware-driven operation is in progress. These are mutually exclusive firmware-driven operations. These operations are one of sweep, chirp, or LIST. Hardware modulations such as FSK and OOK do not set this bit.
4	Always 0. Measuring.
5	Instrument is in “waiting for trigger” state.
6	Instrument is in “waiting for armed” state.
7	Always 0. Indicates correction is in progress.

8	Self-test in progress.
9	"Recording macro". Set when a macro is being recorded (see SFORge:MACRO:DEFine).
10-12	Reserved for Signal Forge use.
13	Instrument summary. Never set by this device.
14	Always 0. User program running.
15	Always 0. Not Used.

Table 30 Operational Status Codes

The Operation Event register latches transition events from the Operation Status as specified by the transition filter.

Questionable Status Group

Refer to SCPI-99 Volume 1 section 9.4.

A bit set in the Questionable Condition register indicates that the data currently being generated is of questionable quality due to some condition affecting the parameter associated with that bit. For example, if the FREQ bit were set, this would mean that the frequency accuracy of the signal was of questionable quality.

The Questionable Status group, consisting of the Questionable Condition (Status) register, the Questionable Positive Transition register, the Questionable Negative Transition register, the Questionable Event register, and the Questionable Event Enable register, is used to determine the specific condition that set bit 3 in the Instrument Summary Status Register.

The bits in the Questionable Status register provide the following information (read using "STATus:QUESTIONable:CONDition?").

Bit	Description
0	Always 0. Indicates inaccurate output voltage.
1	Always 0. Indicates inaccurate output current.
2	Always 0. Indicates inaccurate time.
3	Always 0. Indicates inaccurate output power.
4	Always 0. Indicates inaccurate temperature.
5	Indicates that a frequency ramp (chirp, sweep, or LIST) is at the edge of PLL tracking capability.
6	Always 0. Indicates a phase-lock error or RF unlocked condition.
7	Always 0.
8	Indicates a calibration error. Cleared by successful calibration.
9	Indicates a self-test failure. Cleared by successful *TST? query.
10-12	Always 0. Reserved for device-specific use
13	Always 0. Instrument summary for multiple logical instruments.
14	Always 0. Command warning.
15	Always 0.

Table 31 Questionable Condition Register

The Questionable Event register latches transition events from the Questionable Status register as specified by the transition filter.

STATus Subsystem Commands

Command	Parameters	Comments
:OPERation:EVENT?	returns <NR1>	query only
:OPERation:CONDition?	returns <NR1>	query only
:OPERation:ENABLE	<NR1>	powerup default = 0.

		(unaffected by *RST)
:OPERation:NTRansition	<NR1>	powerup default = 0. (unaffected by *RST)
:OPERation:PTRansition	<NR1>	powerup default = 0. (unaffected by *RST)
:PRESet	(none)	event - no query
:QUEStionable:EVENT?	returns <NR1>	query only
:QUEStionable:CONDition?	returns <NR1>	query only
:QUEStionable:ENABle	<NR1>	powerup default = 0. (unaffected by *RST)
:QUEStionable:NTRansition	<NR1>	powerup default = 0. (unaffected by *RST)
:QUEStionable:PTRansition	<NR1>	powerup default = 0. (unaffected by *RST)
:QUEue:NEXT?	(none)	query only

Table 32 - Status Commands

STATus:OPERation:EVENT?

The query returns the value of the Operation Event Register. Reading the register clears all bits.

STATus:OPERation:CONDition?

The query returns the value of the Operation Condition register.

Example: STAT:OPER:COND?

Queries the Status:Operation:Condition register.

STATus:OPERation:ENABle

Accesses the Operation Event Enable register. Operation is as previously defined for all event enable registers.

STATus:OPERation:NTRansition

The command sets the bits of the Operation Ntransition register. If a bit is set, a transition from 1 to 0 in the condition part causes an entry to be made in the EVENT part of the register. The disappearance of an event in the hardware is thus registered, e.g. the end of an adjustment.

Example: STAT:OPER:NTR 0

A transition from 1 to 0 in the Operation Condition register does not cause an entry to be made in the Operation Event register.

STATus:OPERation:PTRansition

The command sets the bits of the Operation Ptransition register.

STATus:PRESet

The command has no query form. Per SCPI-99 section 20.2, affects the OPERation and QUEStionable register set as follows:

- All PTRansition registers are set to FFFFh - i.e. all transitions from 0 to 1 are detected.
- All NTRansition registers are set to 0 - i.e. a transition from 1 to 0 in a CONDition bit is not detected.
- All ENABle registers are set to 0 - i.e. all events in these registers are not passed on.

STATus:QUEStionable:EVENT?

The query returns the value of the Questionable Event Register. Reading the register clears all bits.

STATus:QUESTIONable:CONDition?

The query returns the value of the Questionable Condition register.

STATus:QUESTIONable:ENABle

Accesses the Questionable Event Enable register. Operation is as previously defined for all event enable registers.

STATus:QUESTIONable:NTRansition

The command sets the bits of the Questionable Ntransition register.

STATus:QUESTIONable:PTRansition

The command sets the bits of the Questionable Ptransition register.

STATus:QUEue:NEXT?

The command queries the oldest entry in the error queue and then deletes it. Positive error numbers denote device-specific errors, and negative error numbers denote error messages defined by SCPI. If the error queue is empty the following is returned:

0,"No error"

This query is identical to SYSTem:ERRor:NEXT?.

Example: STAT:QUE:NEXT?

Queries the oldest entry in the error queue.

SYSTem subsystem

The SYSTem subsystem contains a series of commands for general functions that do not directly affect signal generation.

Command	Parameters	Comments
:ERRor		
:CODE:NEXT?	(none)	query only
:COUNT?	(none)	query only
:NEXT?	(none)	query only
:VERsion?	(none)	query only

Table 33 - Systems Commands

SYSTem:ERRor:CODE:NEXT?

The command queries the oldest code entry in the error queue and then deletes it. Only the error number is returned.

SYSTem:ERRor:COUNT?

The command queries the number of entries in the error queue. If the error queue is empty, '0' is returned.

Example: SYST:ERR:CODE?

Response: 0

SYSTem:ERRor:NEXT?

The command queries the oldest entry in the error queue and then deletes it. Positive error numbers denote device-specific errors, and negative error numbers denote error messages defined by SCPI. If the error queue is empty the following is returned:

0,"No error"

This query is identical to STATus:QUEue:NEXT?.

Example Responses: 0, "No error"

-131,"Invalid suffix"

SYSTem:VERSion

This query returns an <NR2> formatted numeric value corresponding to the SCPI version number for which the instrument complies. The response has the form YYYY.V where the Ys represent the year-version and the V represents an approved revision number for that year. If no approved revisions are claimed, then this extension shall be 0.

Example response: 1999.0

TRIGger Subsystem

The TRIGger subsystem commands are used to control trigger operations. Only one trigger is supported. The trigger is enabled or disabled using TRIGger:STATe.

The ABORt command is not used. The SCPI INITiate subsystem is not used.

When a trigger command (*TRG) is received, but was ignored because of device timing considerations (e.g. not in "waiting for trigger" state), an error -211,"Trigger ignored" is generated. When using the trigger input signal this error will not occur.

All parameters may be changed even after the trigger system has been initiated.

Command	Parameters	Comments
:EXTernal:EDGE	<boolean>	Used when :SOURce = EXTernal. Select edge triggered or level. *RST default is OFF.
:EXTernal:SLOPe:POSitive	<boolean>	Trigger level. *RST default is ON.
:SOURce	NONE AUTO EXTernal	*RST default is NONE.
:STATe	<boolean>	*RST default is OFF.

Table 34 - Trigger Commands

Triggering can be performed for the following operations:

- sweep
- chirp
- LIST

Trigger System Programming

The Signal Forge trigger system has three states: idle, initiated ("wait for armed"), and "wait for trigger". There are two Operation Status (Operation Condition register) bits associated with the trigger subsystem: "waiting for armed" and "waiting for trigger".

"waiting for armed" is set whenever the state machine enters "initiated" state. It is cleared when exiting the state.

"waiting for trigger" is set whenever the state machine enters "waiting for trigger" state. It is cleared when exiting the state.

If desired, the Operation Event register could be used to determine that triggers are occurring by using it to detect edges of the “waiting for trigger” status bit.

Idle State

Turning power on or sending *RST forces the trigger system into the idle state. The only transition from idle is to the initiated (“waiting for arm”) state.

The traverse from idle state to initiated state is caused by setting TRIGger:STATe to ON. Once initiated, the traverse back to idle can be done by the *RST command or by setting TRIGger:STATe to OFF.

This device operates with implied INITiate:CONTinuous = ON.

Initiated State (“Waiting for armed”)

The initiated (“waiting for armed”) state means that the trigger system is enabled, but not currently looking for a trigger. It will transition to “waiting for trigger” state at times defined by those operations that use the trigger.

“Waiting for Trigger” State

When in this state, the instrument is looking for a trigger condition to be satisfied, which can occur by sending a *TRG command, or by satisfying an external signal condition (edge or level triggered).

When a trigger event occurs, a transition is made to the initiated state.

TRIGger Command Descriptions

TRIGger:EXtErnal:EDGe

When SOURce is EXtErnal, this is used to define the trigger as an edge detection event or a signal level. When OFF the trigger is level detected.

This device uses firmware edge-detection. This means the minimum valid/invalid pulse width in order for a transition to be recognized is larger than could be done in hardware. The required timing is specified in the specifications section.

TRIGger:EXtErnal:SLOPe:POSitive

Used when TRIGger:SOURce = EXtErnal to define a trigger as a positive edge or level.

TRIGger:SOURce

This determines what satisfies the trigger.

- NONE – only a software *TRG satisfies the trigger.
- AUTO – trigger is satisfied whenever it is being looked for.
- EXTERNAL – uses the TRIGGER user header input signal. A software trigger is also allowed.

Trigger control of the SYNC signal is determined when the state transitions to ON. When set ON and the trigger source is external then the SYNC signal is controlled by the trigger - SYNC is high only when waiting for a trigger.

When the state transitions to OFF the trigger no longer controls SYNC. If SYNC was controlled by the trigger and a firmware loop is running (sweep, chirp, LIST) then the SYNC signal is undefined.

TRIGger:STATe

Enables the trigger for use by sweep (SWEep:STATe), chirp (CM:STATe), or LIST (LIST:STATe).

If any of these operations has STATe = ON, TRIGger:STATe cannot be modified - returns -221, “Settings conflict”.

UNIT Subsystem

The UNIT subsystem is currently not supported by this device. No parameter suffixes are allowed.

The unit for time is NS (nanoseconds), and the unit for frequency is HZ (hertz).

Error Messages

Introduction

This chapter lists and describes each of the error messages related to signal generator operation. In addition, it provides information about the error message elements, the error query command, the error queue, and the classes of error messages.

Error Queue

As errors and events are detected, they are placed in a queue. This queue is first in, first out.

If the queue overflows, the last error/event in the queue is replaced with error -350, "Queue overflow". Any time the queue overflows, the oldest errors/events remain in the queue, and the most recent error/event is discarded.

Reading an error/event from the head of the queue removes that error/event from the queue, and opens a position at the tail of the queue for a new error/event, if one is subsequently detected. When all errors/events have been read from the queue, further error/event queries (SYSTem:ERRor:NEXT?) return:

0, "No error"

After a STATus:PRESet, the masking is changed so that only errors are reported.

If the error queue is not empty, bit 2 of the Instrument Summary Status Register is set. A query returns only the oldest error code and associated error description information from the error queue. To return all error codes and associated description information, use repetitive queries until an error value of zero is returned, or until bit 2 of the status register is 0.

The error queue is cleared when any of the following occur (IEEE 488.2, section 11.4.3.4):

- Upon power up
- Upon receipt of a *CLS command
- Upon reading the last error message from the queue

Error Query

The SYSTem:ERRor:NEXT? query is a request for the next entry in the instrument's error queue. Error messages in the queue contain an integer in the range [-32768, 32768] denoting an error code and associated descriptive text. Negative codes are reserved by the SCPI standard and defined first in this chapter. Positive error codes are instrument-dependent. An error code value of zero indicates that no error has occurred.

The STATus:QUEue:NEXT? query command is an alias to SYSTem:ERRor:NEXT?.

The instrument responds to the SYSTem:ERRor? query with an error message in the following format:

<error code>,"<error description>;<device-dependent info>"

The <error code> is a unique error descriptor. Certain standard error codes are described in this chapter. The <error description> is a short description of the error, optionally followed by further information about the error. Short descriptions of the standard error codes are given in this chapter. The <device-dependent information> part of the response may contain information that will allow the user to determine the exact error and context. For example:

-221,"Settings conflict;output is OFF"

The maximum string length of <error description> plus <device-dependent information> is 255 characters per SCPI, but this device limits the length as indicated in the Specifications section. The <error description> shall be sent exactly as indicated in this chapter including case.

Error Codes

The system-defined error codes are chosen on an enumerated ("1 of N") basis. The SCPI-defined error codes and the <error description> portions of the query response are listed here. The first error described in each class (for example -100, -200, -300, -400) is a "generic" error. In selecting the proper error code to report, more specific error codes are preferred, and the generic error is used if the others are inappropriate.

Command Errors

These error codes are defined by SCPI 1999.0 volume 2 section 21.8.9.

An <error code> in the range [-199, -100] indicates that an IEEE 488.2 syntax error has been detected by the instrument's parser. The occurrence of any error in this class should cause the command error bit (bit 5) in the standard event status register to be set. One of the following events has occurred:

An IEEE 488.2 syntax error has been detected by the parser. That is, a controller-to-device message is received which is in violation of the IEEE 488.2 standard. Possible violations include a data element that violates the device listening formats or whose type is unacceptable to the device.

An unrecognized header was received. Unrecognized headers include incorrect device-specific headers and incorrect or unimplemented IEEE 488.2 common commands.

A Group Execute Trigger (GET) was entered into the input buffer inside of an IEEE 488.2 <PROGRAM MESSAGE>.

Events that generate command errors shall not generate execution errors, device-specific errors, or query errors; see the other error definitions in this chapter.

Error Code	Description
-100	"Command error" - This is the generic syntax error for devices that cannot detect more specific errors.
-101	"Invalid character" - A syntactic element contains a character which is invalid for that type; for example, a header containing an ampersand, SETUP&. This error might be used in place of errors -114, -121, -141, and perhaps some others.
-102	"Syntax error" - An unrecognized command or data type was encountered. For example, a string was received when the device does not accept strings.
-103	"Invalid separator" - The parser was expecting a separator and encountered an illegal character; for example, the semicolon was omitted after a program message unit, *EMC 1:CH1:VOLTS 5.
-104	"Data type error" - The parser recognized a data element different than one allowed; for example, numeric or string data was expected but block data was encountered.
-105	"GET not allowed" - A Group Execute Trigger was received within a program message (see IEEE 488.2, 7.7).
-108	"Parameter not allowed" - More parameters were received than expected for the header; for example, the *SAV command only accepts one parameter, so receiving *SAV 0,1 is not allowed.
-109	"Missing parameter" Fewer parameters were received than required for the header; for example, the *SAV common command requires one parameter, so receiving *SAV is not allowed.

-110	"Command Header Error" - An error was detected in the header. This error message should be used when the device cannot detect the more specific errors described for errors –111 through –119.
-111	Header Separator Error" - A character which is not a legal header separator was encountered while parsing the header; for example, no white space followed the header, thus "GMC"MACRO" is an error.
-112	"Program mnemonic too long" - The header contains more than 12 characters (see IEEE 488.2, 7.6.1.4.1).
-113	"Undefined header" - The header is syntactically correct, but it is undefined for this specific device; for example, *XYZ is not defined for any device.
-114	"Header suffix out of range" - The value of the numeric suffix attached to a program mnemonic makes the header invalid.
-115	"Unexpected number of parameters" - The number of parameters received does not correspond to the number of parameters expected.
-120	"Numeric data error" - This error, as well as errors –121 through –129, are generated when parsing a data element which appears to be numeric, including the non-decimal types. This particular error message should be used if the device cannot detect a more specific error.
-121	"Invalid character in number" - An invalid character for the data type being parsed was encountered; for example, an alpha in a decimal numeric or a "9" in octal data.
-123	"Exponent too large" - The magnitude of the exponent was larger than 32000 (see IEEE 488.2, 7.7.2.4.1).
-124	"Too many digits" - The mantissa of a decimal numeric data element contained more than 255 digits excluding leading zeros (see IEEE 488.2, 7.7.2.4.1).
-128	"Numeric data not allowed" - A legal numeric data element was received, but the device does not accept one in this position for the header.
-130	"Suffix error" - This error, as well as errors –131 through –139, are generated when parsing a suffix. This particular error message should be used if the device cannot detect a more specific error.
-131	"Invalid suffix" - The suffix value does not follow the syntax described in IEEE 488.2, 7.7.3.2, or the suffix is inappropriate for this device.
-134	"Suffix too long" - The suffix contained more than 12 characters (see IEEE 488.2, 7.7.3.4).
-138	"Suffix not allowed" - A suffix was encountered after a numeric element which does not allow suffixes.
-140	"Character data error" - This error, as well as errors 141 through 149, are generated when parsing a character data element. This particular error message should be used if the device cannot detect a more specific error.
-141	"Invalid character data" - Either the character data element contains an invalid character or the particular element received is not valid for the header.
-144	"Character data too long" - The character data element contains more than 12 characters (see IEEE 488.2, 7.7.1.4).
-148	"Character data not allowed" - A legal character data element was encountered where prohibited by the device.

Table 35 - Command Error Codes

Execution Errors

These error codes are defined by SCPI 1999.0 volume 2 section 21.8.10.

An <error code> in the range [–299,–200] indicates that an error has been detected by the instrument's execution control block. The occurrence of any error in this class should cause the execution error bit (bit 4) of the standard event status register to be set. One of the following events has occurred:

- A <PROGRAM DATA> element following a header was evaluated by the device as outside its legal input range or is otherwise inconsistent with the device's capability.
- A valid program message could not be properly executed due to some device condition.

Execution errors are reported after rounding and expression evaluation operations have taken place. Rounding a numeric data element, for example, shall not be reported as an execution error.

Events that generate execution errors shall not generate command errors, device-specific errors, or query errors; see the other error definitions in this chapter.

Error Code	Description
-200	"Execution error" - This is a generic syntax error for devices that cannot detect more specific errors. This code indicates only that an execution error as defined in IEEE 488.2, 11.5.1.1.5 has occurred.
-211	"Trigger Ignored" - Indicates that a GET, *TRG, or triggering signal was received and recognized by the device but was ignored because of device timing considerations; for example, the device was not ready to respond.

-212	"Arm ignored" - Indicates that an arming signal was received and recognized by the device but was ignored.
-221	"Settings conflict" – Indicates that a legal program data element was parsed but could not be executed due to the current device state (see <i>IEEE 488.2</i> , 6.4.5.3 and 11.5.1.1.5.)
-222	"Data out of range" – Indicates that a legal program data element was parsed but could not be executed because the interpreted value was outside the legal range as defined by the device (see <i>IEEE 488.2</i> , 11.5.1.1.5.). This error is not returned for parameters that are rounded off.

Table 36 - Execution Error Codes

Device-Specific Errors

These error codes are defined by SCPI 1999.0 volume 2 section 21.8.11.

An <error code> in the range [-399, -300] or [1, 32767] indicates that the instrument has detected an error which is not a command error, a query error, or an execution error (e.g. an abnormal hardware or firmware condition).

The error codes in this section are specified by SCPI. They are used whenever the error can be described using SCPI-defined error codes. Positive error codes shown in a later section are device-dependent (defined by the manufacturer) and are used only when a predefined SCPI code does not accurately describe the error.

The occurrence of any error in this class causes the device-specific error bit (bit 3) in the standard event status register to be set.

Error Code	Description
-300	"Device-specific error" – This is the generic device-dependent error for devices that cannot detect more specific errors. This code indicates only that a Device-Dependent Error as defined in <i>IEEE 488.2</i> , 11.5.1.1.6 has occurred.
-311	"Memory error" – Indicates that an error was detected in the device's memory. The scope of this error is device-dependent.
-313	"Calibration memory lost" – Indicates that the nonvolatile calibration data selected by CAL:SElect has been lost.
-315	"Configuration memory lost" - Indicates that nonvolatile configuration data saved by the device has been lost. The meaning of this error is device-specific.
-320	"Storage fault" - Indicates that the firmware detected a fault when using data storage. This error is not an indication of physical damage or failure of any mass storage element.
-330	"Self-test failed".
-340	"Calibration failed". Not set by SF1010.
-350	"Queue overflow" - A specific code entered into the queue in lieu of the code that caused the error. This code indicates that there is no room in the queue and an error occurred but was not recorded.
-360	"Communication error" - This is the generic communication error for devices that cannot detect the more specific errors described for errors -361 through -363.
-361	"Parity error in program message"
-362	"Busy" – means we received a character before the current command completed. You cannot send characters while a command is executing.
-363	"Input buffer overrun". The command/query is too long.

Table 37 - Device-Specific Error Codes

Query Errors

These error codes are defined by SCPI 1999.0 volume 2 section 21.8.12.

An <error code> in the range [-499,-400] indicates that the output queue (message) control of the instrument has detected a problem with the message exchange protocol described in *IEEE 488.2*, Chapter 6.

This instrument does not have an output queue so these messages are never sent.

Operation Complete Event

These error codes are defined by SCPI 1999.0 volume 2 section 21.8.16.

Per SCPI volume 2 section 21.8.16, an <error/event number> in the range [-899:-800] is used when the instrument wishes to report a 488.2 operation complete event to the error/event queue. This event occurs when instrument's synchronization protocol, having been enabled by an *OPC command, completes all selected pending operations. This protocol is described in IEEE 488.2, section 12.5.2. This event also sets the operation complete bit (bit 0) of the Standard Event Status Register. See IEEE 488.2, section 11.5.1. Note: *OPC? does not set bit 0 nor does it enter any event in the error/event queue.

Code	Description
-800	"Operation complete" - The instrument has completed all selected pending operations in accordance with the IEEE 488.2, 12.5.2 synchronization protocol.

Table 38 Operation Complete Code

Signal Forge-Specific Errors

As required by SCPI, all instrument-specific error codes are positive values in the range 1 to 32767. These error codes are device-dependent (defined by the Signal Forge) and are used only when a predefined SCPI code does not accurately describe the error.

All Signal Forge-specific errors are described in this section.

SCPI Command Parser Errors

An <error code> in the range [1, 99] is generated by the instrument's parser in response to the error condition described.

Error Code	Description
10	"Parser error" – Used for parser errors not falling under any of the other codes.
11	"Query only" - Indicates the command is a query command only.
12	"Command only" - Indicates that a query form of the command is not available.

Table 39 - SCPI Parser Error Codes

User Programming Errors

An <error code> in the range [200, 299] is generated for errors related to invalid commands.

Error Code	Description
210	"LIST error" – generic LIST subsystem command error.
211	"LIST index error" – the index is outside the valid range.
212	"Not a LIST component" – tried to access a LIST component that does not exist.
213	"LIST component invalid" - tried to add a component during LIST definition phase but the component is not valid for the current configuration.
214	"LIST phase invalid" - the command is not valid for the current LIST phase.
215	"LIST definition lost" - there was an error reading the definition structure (which defines what is contained in the data array).
216	"LIST data lost" - there was an error reading the component data array.

Table 40 - User Program Error Codes

Internal Programming Errors

An <error code> in the range [300, 399] is generated for errors related to firmware bugs. The instrument contains self-checking code to prevent erroneous operation that might occur due to a firmware bug. The device-specific portion of the returned status may include filename and line number to help in locating the problem, such as: "FW error;loader.c line 321".

These types of errors are usually due to unanticipated combinations of programming or unusual parameter combinations. If you receive one of these errors, you can email technical support with a copy of the error string and some basic information about the instrument condition (e.g. what operating modes are programmed and what parameters were used).

Error Code	Description
300	"FW error" – Used for all firmware errors.

Table 41 - Internal Error Code

Programming

This section describes how to use the SCPI commands listed in this document to perform common tasks.

Handling Errors

SCPI-99 section 20.2 “requires” a particular sequence of commands when initializing the device. This sequence is shown in the next section and includes using these commands/queries: *CLS, *SRE, *ESE, STATus:PREset. In practice, these commands are not needed but they were implemented to support SCPI compatibility.

Suggested Error Handling

You can greatly simplify error handling by following the simple error detection as detailed in the protocol section, and by only supporting a few status registers. The transition and event registers are overkill.

The protocol means you know after each command whether an error is in the queue, so detecting errors is easy. You need only support the following commands and queries for complete status information:

- SYST:ERR:COUNT? returns how many errors are in the queue.
- SYST:ERR:NEXT? returns the next error from the queue.
- STAT:QUES:COND? returns the Questionable Condition register. It is likely that the only bit of interest is the calibration error bit. However, since frequency commands will return an error message that tells you calibration is failed (when this bit is set), you may not need to support this register.
- STAT:OPER:COND? returns the Operational Condition register. Note that these are status bits rather than error bits, and may never be required.

Initialization and Reset

If using the simple error handling as described above, you can clear the error queue at startup by doing SYST:ERR:NEXT? queries until the queue is empty (first char returned = '0').

Per SCPI-99 section 20.2, the following sequence of commands is required when initializing the device. This is not required for this instrument unless you want to use these command/queries in your error handling scheme.

- *CLS – Clears all event status registers and queues. This is NOT suggested because the error queue might have errors that occurred during startup if a macro is enabled for running at startup. The error queue will also have detailed failure text if power-up self-test fails (although there are status bits that indicate the failure, the detailed text description would be lost). You should first read the error queue until it is empty (using SYST:ERR:NEXT? queries).
- *SRE 0 – Clears the IEEE 488.2-mandated service request enable register
- *ESE 0 – Clears the IEEE 488.2-mandated standard event status enable register
- STATus:PRESet – Presets all other registers and queues.

The table below indicates the effects of various commands upon the status data structures in a device.

	SCPI transition filters	SCPI enable registers	SCPI event registers	SCPI Error/Event Queue Enable	SCPI Error/Event Queue	IEE-488 registers ESE SRE	IEE-488 registers SESR STB
*RST	none	none	none	none	none	none	none
*CLS	none	none	clear	none	clear	none	clear
STATus:PRESet	preset	preset	none	preset	none	none	none

Table 42 - Initialization Status

where “clear” means set all bits to zero, “preset” means set all bits to one.

There are several possible startup procedures, but here is the suggested method:

1. Power up the instrument.
2. Watch the front panel LED. It will blink while in the boot block then stay ON solid when application firmware gets control (takes about 5 seconds).
3. If you have enabled a startup macro, wait a sufficient time for the macro to execute. Most macros will execute faster than one second.
4. You can send commands and queries as desired.

If the LED is still blinking after 5 seconds, the application firmware has been corrupted (control stays with the boot block). This should only be caused by an interrupted firmware download. Downloading new firmware should clear the problem.

If you want to communicate with the device as soon as possible after power-up, and without regard to timing of a power-up macro, the simplest method is to wait for the LED to stop blinking then send the <CANCEL> character/command until an ‘A’ response is received. If you have enabled a power-up macro, send “*OPC?” queries until ‘1’ is returned.

Communication errors can occur based on the timing of commands versus where the instrument is in its power-up sequence. A few examples:

- If the first command is sent before the device is ready, some characters can be dropped and the result would be a path error added to the error queue. The same problem could occur when execution moves from the boot block to application firmware.
- If you send a boot block command when the application code has control (or vice versa), you may receive an unrecognized command error.

Suggested Programming Sequence

The main programming requirements in suggested order are:

- Issue *RST to get to a known state
- Select an output (OUTPut:SElect).
- Program the output power (single ended) or voltage level (TTL).
- Select a frequency range if required (FREQuency:RANGe).
- Turn the output ON (OUTPut:STATe ON).
- If using external signals to control operation, set FSKmux:INTernal to OFF.
- Enable the trigger if desired (TRIGger:STATe ON).
- If doing CM (chirp), FM (FSK) or sweep program the associated FREQuency subsystem parameters such as FREQuency:START, STOP, and DWELL.
- Select frequency operation mode (FREQuency:MODE). This is only necessary if not using the default FIXed mode (constant waveform).
- Program the parameters required by the selected mode.
- Program internal hardware modulation: AM (ASK), CM (chirp), FM (FSK), PM(BPSK), or OM(OOK). Write the parameters and set STATe to ON.
- Program and enable any number of external hardware modulations: AM (ASK), FM (FSK), PM(BPSK), or OM(OOK). Write the parameters and set STATe to ON.

Due to hardware or logical considerations, not all combinations of programming are allowed. For example, you cannot program internally modulated FM and BPSK at the same time because there is only one internal hardware modulation source. When a conflict of this type occurs you will get an error similar to the following:

-221, "Settings conflict;PWM resource in use".

Controlling Outputs

Only one output can be used at a time. The OUTPut subsystem selects the output, and is used to turn the output on and off (whether single-ended, differential, or TTL). Whenever an output is selected all other outputs are turned off.

Output	Selection and Control
single-ended	OUTP:SEL:PORT SE POW:LEV:IMM:AMPL -2 (sets -2dBm power level) OUTP:STAT ON (turns on the output)
differential	OUTP:SEL:PORT DIFF OUTP:STAT ON (turns on the output)
TTL	OUTP:SEL:PORT TTL VOLT:LEV 3.3 (sets 3.3V level, can use 1.8, 2.5, 3.3) OUTP:STAT ON (turns on the output)
module	OUTP:SEL:PORT MOD OUTP:STAT not required (module output is always enabled)
Operation	Selected By
constant frequency waveform (frequency = FREQuency:FIXed)	*RST or FREQuency:MODE = FIXed
FW modulated sweep	Sweep subsystem FREQuency:MODE = SWEep
HW modulated FSK: jump between two frequencies (unramped FSK)	FREQuency:MODE FM FM:MODE FSKNonramped
HW modulated FSK: ramp up and down between two frequencies (ramped FSK)	FREQuency:MODE FM FM:MODE FSKRamped

automatic ramp up and down between two frequencies (triangle FSK)	FM:MODE FSKTriangle
FW modulated chirp: ramp between two frequencies with 0+ delay between chirps	FREQUENCY:MODE CM CM subsystem
HW modulated jump between two power levels (ASK)	AM subsystem
HW modulated jump between two phases (BPSK)	PM subsystem
OOK (output on/off keying)	OOK subsystem
FW modulated control of user-specified frequency, power, phase, and/or OOK at arbitrary intervals	LIST subsystem

Table 43 - Output Control Commands

Common Operations

The following table lists common operations. “HW modulated” means the operation uses an internal or external modulation signal. “FW modulated” means the operation is controlled using firmware.

Operation	Selected By
constant frequency waveform (frequency = FREQUENCY:FIXed)	*RST or FREQUENCY:MODE = FIXed
FW modulated sweep	Sweep subsystem FREQUENCY:MODE = SWEep
HW modulated FSK: jump between two frequencies (unramped FSK)	FREQUENCY:MODE FM FM:MODE FSKNonramped
HW modulated FSK: ramp up and down between two frequencies (ramped FSK)	FREQUENCY:MODE FM FM:MODE FSKRamped
automatic ramp up and down between two frequencies (triangle FSK)	FM:MODE FSKTriangle
FW modulated chirp: ramp between two frequencies with 0+ delay between chirps	FREQUENCY:MODE CM CM subsystem
HW modulated jump between two power levels (ASK)	AM subsystem
HW modulated jump between two phases (BPSK)	PM subsystem
OOK (output on/off keying)	OOK subsystem
Firmware-modulated control of user-specified frequency, power, phase, and/or OOK at arbitrary intervals	LIST subsystem

Table 44 Common Operations

Simultaneous Operations

The output can be modulated in three major ways:

Using an external hardware modulation signal (FM, PM, OM, AM)

- Using an internal hardware modulation signal (FM, PM, OM, AM)
- Using firmware modulation to change the output (sweep or LIST operation). Firmware modulation can be frequency (sweep, LIST), phase (LIST), power (LIST), FM (LIST), OM (LIST), PM (LIST).

The FSKmux selects internal or external modulation for the FM, OM, and PM subsystems. AM is individually selectable for internal/external operation.

You can run the following operations concurrently (depending on the actual selections):

- up to three external modulations (FM or PM, OM, AM)
- one internal hardware modulation (FM or PM, OM, AM)
- one firmware modulation (sweep or LIST)

You can only do one modulation of each output characteristic: frequency, power, OOK (on/off keying), and phase. For example, you are not allowed to have FM (FSK) modulation and sweep running at the same time (they are both frequency operations).

Programming Examples

This section contains programming examples for various operations.

Program a Fixed Frequency

Reset, turn on the output, and set a frequency.

```
*RST
OUTP:STAT 1
FREQ:FIX 12345678
```

Set a new frequency.

```
FREQ:FIX 20111222
```

Set a Differential Output Range 3 Frequency

```
*RST
OUTP:SEL:PORT DIFF
OUTP:STAT 1
FREQ:RANG 3
FREQ:FIX 300111222
```

Program a Sweep

Program a sweep for 10MHz to 20Mhz.

```
*RST
OUTP:STAT 1
FREQ:FIX 50000000
FREQ:STAR 10000000
FREQ:STOP 20000000
FREQ:STEP:INCR 1000000
SWE:DWEL 1000000000
FREQ:MODE SWEEP
SWE:STAT 1
```

Program a Pulsed Chirp

A 2-second chirp that uses 2-second idle1 and idle2 times so the chirp is bounded by states that hold the starting and ending frequencies. The chirp starts at 50MHz and increments (the default).

```
*RST
OUTP:STAT 1
FREQ:FIX 50000000
FREQ:STEP:INCR 10
FREQ:DWEL 10000
FREQ:MODE CM
CM:IDLE1:ENAB 1
CM:IDLE1:TIME 2E9
CM:IDLE2:ENAB 1
CM:IDLE2:TIME 2E9
CM:PULS 2E9
CM:STAT ON
```

Program a Simple Non-Pulsed Chirp

A 2-second chirp that does not use idle1 and idle2 times so the chirp is a repeating ramp with no delays between chirps. The chirp starts at 50MHz and increments (the default).

```
*RST
OUTP:STAT 1
FREQ:FIX 50000000
FREQ:STEP:INCR 10
FREQ:DWEL 10000
FREQ:MODE CM
CM:IDLE1:ENAB 0
CM:IDLE2:ENAB 0
CM:PULS 2E9
CM:STAT ON
```

Program FSK using Internal PWM

This FSK example moves from 10Mhz to 20MHz with a modulating frequency of 2500hz, modulation signal duty cycle 70/30.

```
*RST
OUTP:STAT 1
FREQ:STAR 10000000
FREQ:STOP 20000000
```

```
FREQ:STEP:INCR 10
FREQ:DWEL 10000
FREQ:MODE FM
FM:SOUR PWM
FM:MODE RAMPED
FM:FREQ 2500
FM:DUTY 70
FM:STAT 1
```

Program a LIST using Multiple Components

This example programs a LIST with frequency, phase, and power component. It also programs the SYNC output level for each step. The LIST has 2 points and a different dwell is specified for each point.

```
*RST
OUTP:STAT 1
FREQ:FIX 50000000
LIST:POIN:IND 0
LIST:MDW 1
LIST:POIN:OPER:FREQ 12345678
LIST:POIN:OPER:PHAS 45
LIST:POIN:OPER:POW -6
LIST:POIN:CONT:SYNC 1
LIST:POIN:IND 1
LIST:POIN:OPER:DWEL 30000
LIST:POIN:IND 2
LIST:POIN:OPER:FREQ 23456789
LIST:POIN:OPER:PHAS 270
LIST:POIN:OPER:POW 5
LIST:POIN:CONT:SYNC 0
LIST:POIN:OPER:DWEL 50000
```

Program a LIST Using Trigger

A LIST with 4 points that wait for the trigger after the first and third points. Each point is one second long, and the frequency moves from 10MHZ to 40MHZ.

```
*RST
OUTP:STAT 1
FREQ:FIX 50000000
TRIG:STAT 1
```

TRIG:SOUR NONE
LIST:POIN:IND 0
LIST:MDW 1
LIST:POIN:OPER:FREQ 10000000
LIST:POIN:CONT:TRIG 1
LIST:POIN:IND 1
LIST:POIN:OPER:DWEL 1000000000
LIST:POIN:IND 2
LIST:POIN:OPER:FREQ 20000000
LIST:POIN:OPER:DWEL 1000000000
LIST:POIN:CONT:TRIG 0
LIST:POIN:IND 3
LIST:POIN:OPER:FREQ 30000000
LIST:POIN:OPER:DWEL 1000000000
LIST:POIN:CONT:TRIG 1
LIST:POIN:IND 4
LIST:POIN:OPER:FREQ 40000000
LIST:POIN:OPER:DWEL 1000000000
LIST:POIN:CONT:TRIG 0

External Control

The ten 2-pin connectors labeled External Control are provided on the front panel. These connectors enable you to control several modulation functions, such as frequency shift keying (FSK) or On/Off Keying (OOK). The function assignments are listed below.

POSITION	NAME (bottom row)		POSITION	NAME (top row)
1	SYNC		1	TRIG_IN
2	DIFF_OOK		2	GND
3	FSK_BPSK		3	GND
4	Reserved		4	GND
5	SE_OOK		5	GND
6	Reserved		6	GND
7	15V		7	ASK
8	15V		8	GND
9	Reserved		9	GND
10	Reserved		10	GND

Table 45 External Control Pinout

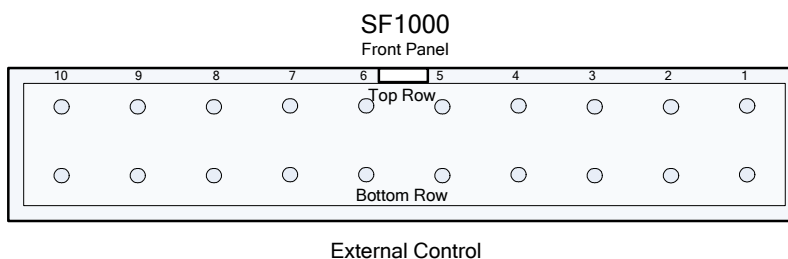


Figure 9. External Control Pin Locations

Recommended Cables/Connectors

You may attach either a 20-pin connector and cable assembly to an entire External Control block or individual 2-pin assemblies to control a single function. The recommended connectors are:

- 20-pin connector and cable assembly P/N: M3AAA-2006J-ND
- 2-pin connector housing and contacts may be used to connect to individual control positions

- 2-position housing P/N: A26921-ND
- Contacts (2 required) P/N: A26951-ND

Items may be purchased from www.digikey.com.

External Connector Pin Description

DIFF_OOK. The differential clock output supports OOK for the frequency range of 50 MHz to 1 GHz. This input pin controls the differential output: driving this pin low will stop the differential output.

FSK_BPSK. For FSK waveforms, this pin allows the user to shift the frequency output as defined previously. For BPSK waveforms, this pin selects between phase 1 and phase 2 and as defined previously.

SE_OOK. The SF1010 supports OOK (On/Off Keying) for the AC Coupled output in frequency range of 100 KHz to 100 MHz only. This input pin provides for external control of the output: placing this pin low stops the output of the SF1010.

ASK. This control pin allows you to modulate the AC Coupled output in order to implement an externally controlled Asynchronous Shift Keying. When this pin is driven low the AC coupled output will be attenuated by 16 dB over the full output power (when the signal is high).

Note:

The ASK control pin does not have a GND pin as its opposite, Any other GND pin on the External Control connector may be used instead.

TRIG_IN. This is the trigger input whose use is defined in the trigger subsystem.

SYNC. The SYNC output signal is driven during the firmware-modulated operations for chirp, sweep, and LIST. It can be used as a trigger source.

SYNC is normally low, but when a firmware loop is running SYNC goes high/low as defined by the operation. These operations are sweep, chirp, and LIST.

When an operation controls SYNC and is programmed to have SYNC follow the trigger, SYNC goes high just after the operation is looking for a trigger (i.e. just after it sets "waiting for trigger" state), and goes low just after the trigger is recognized.

Signal Characteristics

As described above, the SF1010 gives you the ability to control output characteristics using external control pins. Each of the control pins has a low pass filter placed at its input, except for the TRIG_IN pin. The low pass filter has a 3dB cutoff point of 150 KHz. The input signals are 5V compatible signals and must be referenced to the SF1010 GND (any of the GND pins of the 20-pin header). Input impedance is 2K ohms.

The External Control inputs are 5V tolerant and include a certain level of protection as shown in the following block diagram:

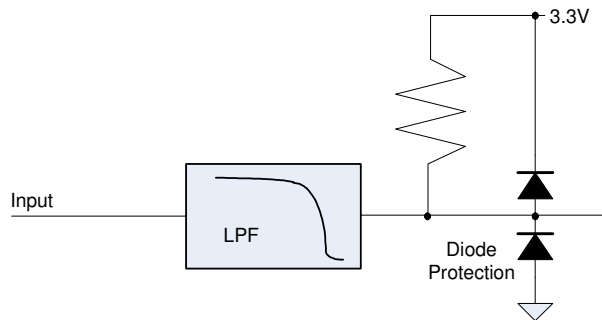


Figure 10. External Control Pin Filter

The Figure below gives an example of external control of a waveform. In the example chosen, an external signal controls the FSK_BPSK signal pin of the SF1010.

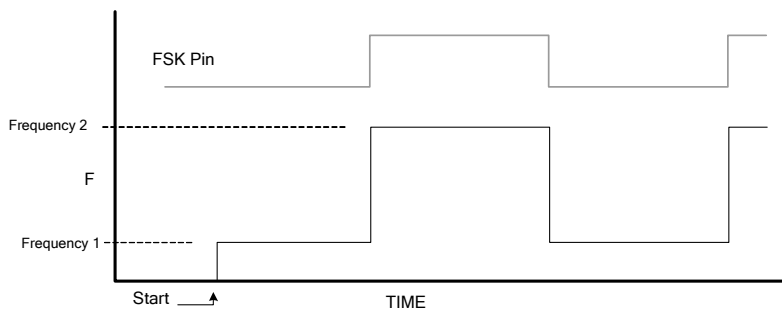


Figure 11. FSK_BPSK Pin Operation

Software Update Procedure

The EZWave user GUI program and the embedded operating software of the Signal Forge signal generators may be updated by the user by following the procedure below. Note that all output activity will be suspended while the software update is in process.

Software updates are posted to the [Support](http://www.signalforge.com/home/sf1/support_main.html) page of the Signal Forge web site, http://www.signalforge.com/home/sf1/support_main.html.

Firmware Download Procedure

1. Download the desired firmware from the Signal Forge website. The filename format for firmware is "sf1010_dwn_A_B_C_D" where A is the major version, B the minor version, C the build, and D the revision. The latest firmware has the highest numbered major version, minor version, then revision.
2. Start the EZWave program and select "Download Firmware" from the Options menu. Follow the on-screen instructions.
3. The screen will display download status and indicate success when the download is complete. Downloading should take one or two minutes. When firmware is successfully downloaded the new firmware will take effect - there is no need to turn the generator off and on.

Revision History

V 01-0.9

Initial draft and Alpha and Beta releases

SIGNAL FORGE, LLC

SCPI Programmer's Manual v1.0

2115 Saratoga Drive • Austin TX 78733
Phone 512.275.3733 • Fax 512.275.3735
www.signalforge.com