SERIES 690XXA SYNTHESIZED CW GENERATOR

SCPI PROGRAMMING MANUAL

490 JARVIS DRIVE
MORGAN HILL, CA 95037-2809

P/N: 10370-10304
REVISION: C
PRINTED: OCTOBER 1997

DECLARATION OF CONFORMITY

Manufacturer's Name: ANRITSU COMPANY

Manufacturer's Address: Microwave Measurements Division
490 Jarvis Drive
Morgan Hill, CA 95037-2809
USA

declares that the product specified below:

Product Name: Synthesized CW / Sweep / Signal Generator

Model Number: 690XXA, 691XXA, 692XXA, 693XXA

conforms to the requirement of:

EMC Directive 89/336/EEC as amended by Council Directive 92/31/EEC & 93/68/EEC
Low Voltage Directive 73/23/EEC as amended by Council directive 93/68/EEC

Electromagnetic Interference:

Emissions:	CISPR 11:1990/EN55011:1991 Group 1 Class A
Immunity:	IEC 1000-4-2:1995/prEN50082-1:1995 - 4kV CD, 8kV AD
	IEC 1000-4-3:1993/ENV50140:1994 - 3V/m
	IEC 1000-4-4:1995/prEN50082-1:1995 - 0.5kV SL, 1kV PL
	IEC 1000-4-5:1995/prEN50082-1:1995 - 0.5kV - 1kV LN
	0.5kV - 1kV NG
	0.5kV - 1kV GL

Electrical Safety Requirement:

Product Safety:	IEC 1010-1:1990 + A1/EN61010-1:1993

Jim Blasi

Manager of Corporate Quality

Morgan Hill, CA

5-SEPT-97

Date

European Contact: For Anritsu product EMC & LVD information, contact Anritsu LTD, Rutherford Close, Stevenage Herts, SG1 2EF UK, (FAX 44-1438-740202)

Table of Contents

Table of Contents (Continued)

Chapter 1
General GPIB Information

Table of Contents

Chapter 1
General GPIB Information

1-1 SCOPE OF MANUAL

This manual provides information for remote operation of the Series 690XXA Synthesized CW Generator using commands sent from an external controller via the IEEE-488 General Purpose Interface Bus (GPIB). It includes the following:

- ❑ A general description of the GPIB and the bus data transfer and control functions.
- ❑ A listing of the IEEE-488 Interface Function Messages recognized by the CW generator with a description of its response.
- ❑ A complete listing and description of all the Standard Commands for Programmable Instruments (SCPI) commands that can be used to control CW generator operation with examples of command usage.

This manual is intended to be used in conjunction with the Series 690XXA Synthesized CW Generator Operation Manual, P/N 10370-10300. Refer to that manual for general information about the 690XXA, including equipment set up and front panel (manual mode) operating instructions.

Electronic Manual

This manual is available on CD ROM as an Adobe Acrobat Portable Document Format (∗.pdf) file. The file can viewed using Acrobat Reader, a free program that is also included on the CD ROM. The file is “linked” such that the viewer can choose a topic to view from the displayed “bookmark” list and “jump” to the manual page on which the topic resides. The text can also be word-searched. Contact ANRITSU Customer Service for price and availability.

GPIB Programming Manual

In addition to the SCPI programming commands described in this manual, the 690XXA GPIB interface also accepts and implements a set of 690XXA GPIB Product-Specific (“NATIVE”) commands. These GPIB commands are listed and described in the Series 690XXA Synthesized CW Generator GPIB Programming Manual, P/N 10370-10302.

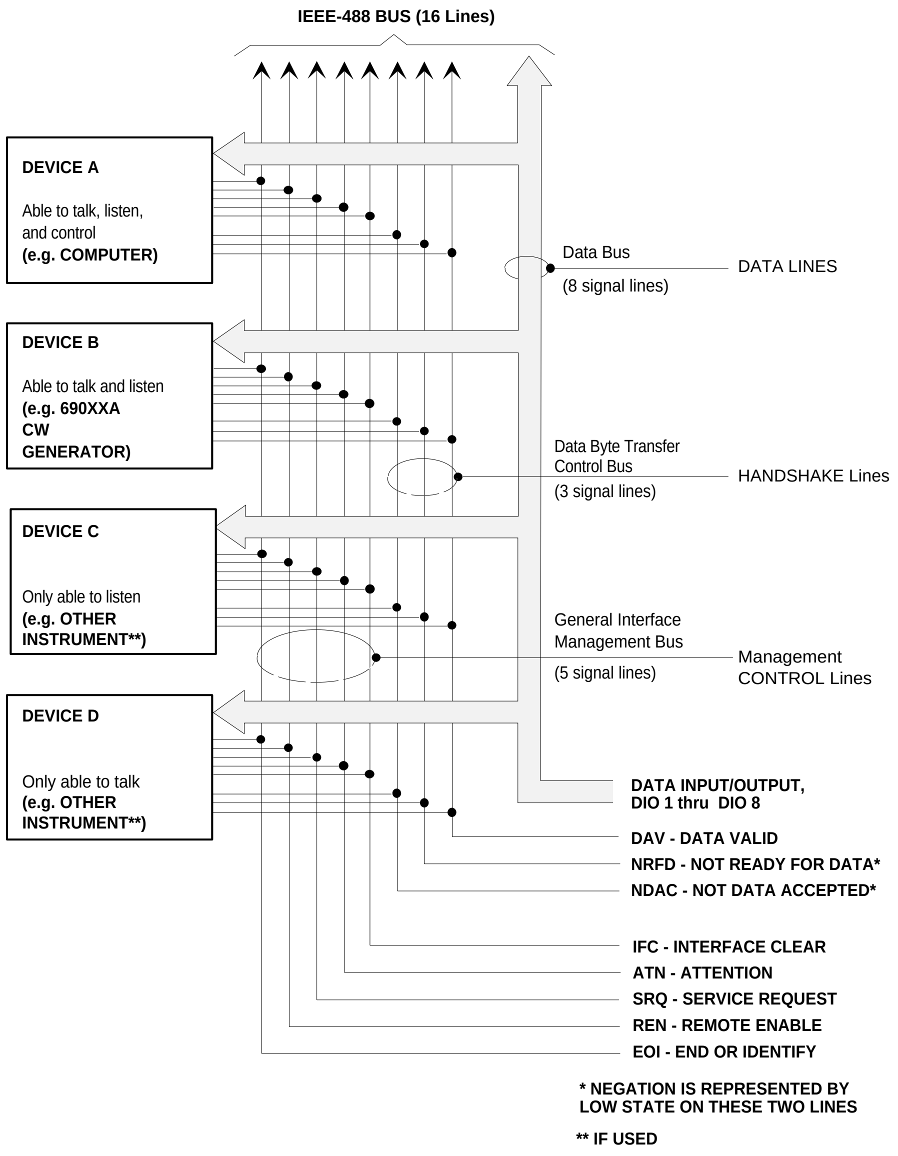


Figure 1-1. *Interface Connections and GPIB Bus Structure*

1-2 INTRODUCTION

This chapter provides a general description of the GPIB and the bus data transfer and control functions. It also contains a listing of the 690XXA's GPIB interface function subset capability and response to IEEE-488 interface function messages.

The GPIB information presented in this chapter is general in nature. For complete and specific information, refer to the following documents: ANSI/IEEE Std 488.1-1987 *IEEE Standard Digital Interface for Programmable Instrumentation* and ANSI/IEEE Std 488.2-1987 *IEEE Standard Codes, Formats, Protocols and Common Commands*. These documents precisely define the total specification of the mechanical and electrical interface, and of the data transfer and control protocols.

1-3 IEEE-488 INTERFACE BUS DESCRIPTION

The IEEE-488 General Purpose Interface Bus (GPIB) is an instrumentation interface for integrating instruments, computers, printers, plotters, and other measurement devices into systems. The GPIB uses 16 signal lines to effect transfer of information between all devices connected on the bus.

The following requirements and restrictions apply to the GPIB.

- ❑ No more than 15 devices can be interconnected by one contiguous bus; however, an instrumentation system may contain more than one interface bus.
- ❑ The maximum total cumulative cable length for one interface bus may not exceed twice the number of devices connected (in meters) , or 20 meters—whichever is less.
- ❑ A maximum data rate of 1 Mb/s across the interface on any signal line.
- ❑ Each device on the interface bus must have a unique address, ranging from 00 to 30.

The devices on the GPIB are connected in parallel, as shown in Figure 1-1. The interface consists of 16 signal lines and 8 ground lines in a shielded cable. Eight of the signal lines are the data lines, DIO 1 thru DIO 8. These data lines carry messages (data and commands), one byte at a time, among the GPIB devices. Three of the remaining lines are the handshake lines that control the transfer of message bytes between devices. The five remaining signal lines are referred to as interface management lines.

The following paragraphs provide an overview of the GPIB including a description of the functional elements, bus structure, bus data transfer process, interface management bus, device interface function requirements, and message types.

Functional Elements

Effective communications between devices on the GPIB requires three functional elements; a *talker*, a *listener*, and a *controller*. Each device on the GPIB is categorized as one of these elements depending on its current interface function and capabilities.

Talker
A talker is a device capable of sending device-dependent data to another device on the bus when addressed to talk. Only one GPIB device at a time can be an active talker.

Listener
A listener is a device capable of receiving device-dependent data from another device on the bus when addressed to listen. Any number of GPIB devices can be listeners simultaneously.

Controller
A controller is a device, usually a computer, capable of managing the operation of the GPIB. Only one GPIB device at a time can be an active controller. The active controller manages the transfer of device--dependent data between GPIB devices by designating who will talk and who will listen.

System Controller
The system controller is the device that always retains ultimate control of the GPIB. When the system is first powered-up, the system controller is the active controller and manages the GPIB. The system controller can pass control to a device, making it the new active controller. The new active controller, in turn, may pass control on to yet another device. Even if it is not the active controller, the system controller maintains control of the Interface Clear (IFC) and Remote Enable (REN) interface management lines and can thus take control of the GPIB at anytime.

Bus Structure

The GPIB uses 16 signal lines to carry data and commands between the devices connected to the bus. The interface signal lines are organized into three functional groups.

- ❑ Data Bus (8 lines)
- ❑ Data Byte Transfer Control Bus (3 lines)
- ❑ General Interface Management Bus (5 lines)

The signal lines in each of the three groups are designated according to function. Table 1-1 lists these designations.

Table 1-1. *Interface Bus Signal Line Designations*

Bus Type	Signal Line Name	Function
Data Bus	DIO1–DIO8	Data Input/Output, 1 thru 8
Data Byte Transfer Control Bus	DAV	Data Available
	NRFD	Not Ready For Data
	NDAC	Not Data Accepted
General Interface Management Bus	ATN	Attention
	IFC	Interface Clear
	SRQ	Service Request
	REN	Remote Enable
	EOI	End Or Identify

Data Bus Description

The data bus is the conduit for the transfer of data and commands between the devices on the GPIB. It contains eight bi-directional, active-low signal lines—DIO 1 thru DIO 8. Data and commands are transferred over the data bus in byte-serial, bit-parallel form. This means that one byte of data (eight bits) is transferred over the bus at a time. DIO 1 represents the least-significant bit (LSB) in this byte and DIO 8 represents the most-significant bit (MSB). Bytes of data are normally formatted in seven-bit ASCII (American Standard Code for Information Interchange) code. The eighth (parity) bit is not used.

Each byte placed on the data bus represents either a command or a data byte. If the Attention (ATN) interface management line is TRUE while the data is transferred, then the data bus is carrying a bus command which is to be received by every GPIB device. If ATN is FALSE, then a data byte is being transferred and only the active listeners will receive that byte.

Data Byte Transfer Control Bus Description

Control of the transfer of each byte of data on the data bus is accomplished by a technique called the “three-wire handshake”, which involves the three signal lines of the Data Byte Transfer Control Bus. This technique forces data transfers at the speed of the slowest listener, which ensures data integrity in multiple listener transfers. One line (DAV) is controlled by the talker, while the other two (NRFD and NDAC) are wired-OR lines shared by all active listeners. The handshake lines, like the other GPIB lines, are active low. The technique is described briefly in the following paragraphs and is depicted in Figure 1-2. For further information, refer to ANSI/IEEE Std 488.1.

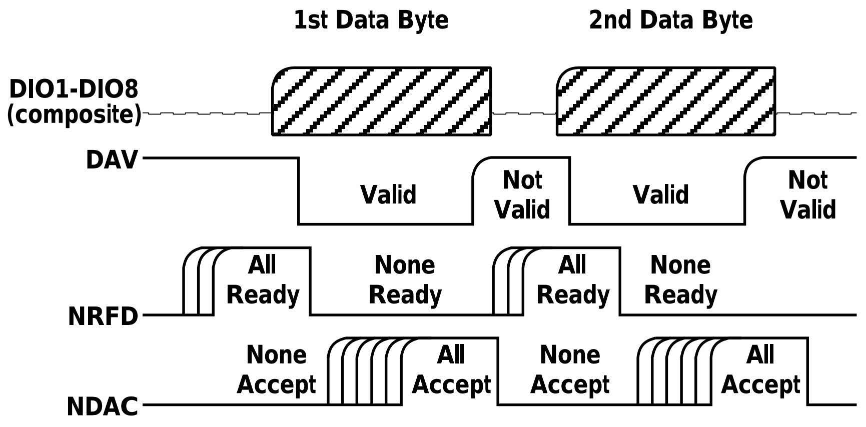


Figure 1-2. *Typical GPIB Handshake Operation*

DAV (Data Valid)
This line is controlled by the active talker. Before sending any data, the talker verifies that NDAC is TRUE (active low) which indicates that all listeners have accepted the previous data byte. The talker then places a byte on the data lines and waits until NRFD is FALSE (high) which indicates that all addressed listeners are ready to accept the information. When both NRFD and NDAC are in the proper state, the talker sets the DAV line TRUE (active low) to indicate that the data on the bus is valid (stable).

NRFD (Not Ready For Data)
This line is used by the listeners to inform the talker when they are ready to accept new data. The talker must wait for each listener to set the NRFD line FALSE (high) which they will do at their own

rate. This assures that all devices that are to accept the data are ready to receive it.

NDAC (Not Data Accepted)
This line is also controlled by the listeners and is used to inform the talker that each device addressed to listen has accepted the data. Each device releases NDAC at its own rate, but NDAC will not go FALSE (high) until the slowest listener has accepted the data byte.

General Interface Management Bus Description

The general interface management bus is a group of five signal lines used to manage the flow of information across the GPIB. A description of the function of each of the individual control lines is provided below.

ATN (Attention)
The active controller uses the ATN line to define whether the information on the data bus is a command or is data. When ATN is TRUE (low), the bus is in the command mode and the data lines carry bus commands. When ATN is FALSE (high), the bus is in the data mode and the data lines carry device-dependent instructions or data.

EOI (End or Identify)
The EOI line is used to indicate the last byte of a multibyte data transfer. The talker sets the EOI line TRUE during the last data byte.

The active controller also uses the EOI line in conjunction with the ATN line to initiate a parallel poll sequence.

IFC (Interface Clear)
Only the system controller uses this line. When IFC is TRUE (low), all devices on the bus are placed in a known, quiescent state (unaddressed to talk, unaddressed to listen, and service request idle).

REN (Remote Enable)
Only the system controller uses this line. When REN is set TRUE (low), the bus is in the remote mode and devices are addressed either to listen or to talk. When the bus is in remote and a device is addressed, it receives instructions from the GPIB rather than from its front panel. When REN is set FALSE (high), the bus and all devices return to local operation.

SRQ (Service Request)
The SRQ line is set TRUE (low) by any device requesting service by the active controller.

Device Interface Function Capability

An interface function is the GPIB system element which provides the basic operational facility through which a device can receive, process, and send messages. Each specific interface function may only send or receive a limited set of messages within particular classes of messages. As a result, a set of interface functions is necessary to achieve complete communications among devices on the GPIB. ANSI/IEEE Std 488.1 defines each of the interface functions along with its specific protocol.

ANSI/IEEE Std 488.2 specifies the minimum set of IEEE 488.1 interface capabilities that each GPIB device must have. This minimum set of interface functions assures that the device is able to send and receive data, request service, and repond to a device clear message. Table 1-2 lists the interface function capability of the series 690XXA CW generator.

Table 1-2. *690XXA Interface Function Capability*

Function Identifier	Function	681XXB Capability
AH1	Acceptor Handshake	Complete Capability
SH1	Source Handshake	Complete Capability
T6	Talker	No Talk Only (TON)
L4	Listener	No Listen Only (LON)
SR1	Service Request	Complete Capability
RL1	Remote/Local	Complete Capability
PP1	Parallel Poll	Complete Capability
DC1	Device Clear	Complete Capability
DT1	Device Trigger	Complete Capability

C0, 1, 2, 3, 28	Controller Capability Options	C0, No Capability;
		C1, System Controller;
		C2, Send IFC and Take Charge;
		C3, Send REN;
		C28, Send IF Messages

E2	Tri-State Drivers	Three-state bus drivers

Message Types

There are three types of information transmitted over the GPIB—interface function messages, device-specific commands, and data and instrument status messages.

Interface Function Messages
The controller manages the flow of information on the GPIB using interface function messages, usually called *commands* or *command messages*. Interface function messages perform such functions as initializing the bus, addressing and unaddressing devices, and setting device modes for remote or local operation.

There are two types of commands—multiline and uniline. Multiline commands are bytes sent by the active controller over the data bus (DIO1-DIO8) with ATN set TRUE. Uniline commands are signals carried by the individual interface management lines.

The user generally has control over these commands; however, the extent of user control depends on the implementation and varies with the specific GPIB interface hardware and software used with the external controller.

Device-Specific Commands
These commands are keywords or mnemonic codes sent by the external controller to control the setup and operation of the addressed device or instrument. The commands are normally unique to a particular instrument or class of instruments and are described in its documentation.

Device-specific commands are transmitted over the data bus of the GPIB to the device in the form of ASCII strings containing one or more keywords or codes.They are decoded by the device's *internal controller* and cause the various instrument functions to be performed.

Data and Instrument Status Messages
These messages are sent by the device to the external controller via the GPIB. They contain measurement results, instrument status, or data files that the device transmits over the data bus in response to specific requests from the external controller. The contents of these messages are instrument specific and may be in the form of ASCII strings or binary data.

In some cases data messages will be transmitted from the external controller to the device. For example, messages to load calibration data.

An SRQ (service request) is an interface function message sent *from the device* to the external controller to request service from the controller, usually due to some predetermined status condition or error. To send this message, the device sets the SRQ line of the General Interface Management Bus true, then sends a status byte on the data bus lines.

An SRQ interface function message is also sent by the device in response to a serial poll message from the controller, or upon receiving an Output Status Byte(s) command from the controller. The protocols associated with the SRQ functions are defined in the ANSI/IEEE Std 488.2 document.

The manner in which interface function messages and device-specific commands are invoked in programs is implementation specific for the GPIB interface used with the external controller. Even though both message types are represented by mnemonics, they are implemented and used in different ways.

Normally, the interface function messages are sent automatically by the GPIB driver software in response to invocation of a software function. For example, to send the IFC (Interface Clear) interface fuction message, one would call the ibsic function of the National Instruments software driver. On the other hand, the command *RST (Reset) is sent in a command string to the addressed device. In the case of the National Instruments example, this would be done by using the ibwrt function call.

1-4 690XXA GPIB OPERATION

All Series 690XXA Synthesized CW Generator functions, settings, and operating modes (except for power on/standby) are controllable using commands sent from an external controller via the GPIB. When in the remote (GPIB) mode, the CW generator functions as both a listener and a talker. The GPIB interface function capability of the 690XXA is listed in Table 1-2 (page 1-10).

Setting GPIB Operating Parameters

The 690XXA leaves the factory with the GPIB address value set to 5 and the data delimiting terminator set to carriage return and line feed (CR/LF). A different address value can be entered from the front panel using the Configure GPIB menu. Using this same menu, the data delimiting terminator can be changed to carriage return (CR) only. Refer to Chapter 2 of the Series 690XXA Synthesized CW Generator Operation Manual for the procedure.

Selecting the Interface Language

Series 690XXA Synthesized CW Generators with Option 19 can be remotely operated using one of two external interface languages—Native or SCPI. The Native interface language uses a set of 690XXA GPIB Product Specific commands to control the instrument; the SCPI interface language uses a set of the Standard Commands for Programmable Instruments commands to control the unit. Selecting which of these external interface languages is to be used can be made from the front panel using the Configure GPIB menu. Refer to Chapter 2 of the Series 690XXA Synthesized CW Generator Operation Manual for the procedure.

Response to GPIB Interface Function Messages

Table 1-3 (page 1-14) lists the GPIB Interface Function Messages that the 690XXA will recognize and respond to. With the exception of the Device Clear and Selected Device Clear messages, these messages affect only the operation of the 690XXA GPIB interface. The 690XXA response for each message is indicated.

Interface function messages are transmitted on the GPIB data lines and interface management lines as either unaddressed or addressed commands. The manner in which these messages are invoked in programs is implementation dependent. For programming information, refer to the documentation included with the GPIB Interface used for the external controller.

Table 1-3. *690XXA Response to GPIB Interface Function Messages*

Interface Function Message	Addressed Command	690XXA Response
Device Clear (DCL) Selected Device Clear (SDC)	No Yes	Resets the 690XXA to its default state. (Equivalent to sending the ∗RST command.)
Go To Local (GTL)	Yes	Returns the 690XXA to local (front panel) control.
Group Execute Trigger (GET)	Yes	Executes a string of commands, if programmed.
Interface Clear (IFC)	No	Stops the 690XXA GPIB interface from listening or talking. (The front panel controls are not cleared.)
Local Lockout (LLO)	No	Disables the front panel menu RETURN TO LOCAL soft-key.
Remote Enable (REN)	No	Places the 690XXA under remote (GPIB) control when it has been addressed to listen.
Serial-Poll Enable (SPE)	No	Outputs the serial-poll status byte.
Serial-Poll Disable (SPD)	No	Disables the serial-poll function.
Parallel-Poll Configure (PPC)	Yes	Responds to a parallel-poll message (PPOLL) by setting assigned data bus line to the logical state (1,0) that indicates its correct SRQ status.
Parallel-Poll Unconfigure (PPU)	No	Disables the parallel-poll function.

Chapter 2 Programming with SCPI Commands

Table of Contents

Chapter 2 Programming with SCPI Commands

2-1 INTRODUCTION

This chapter provides an introduction to SCPI programming that includes descriptions of the command types, hierarchial command structure, data parameters, and notational conventions. Information on 690XXA status system and trigger system programming is also provided.

2-2 INTRODUCTION TO SCPI PROGRAMMING

The *Standard Commands for Programmable Instruments (SCPI)* defines a set of standard programming commands for use by all SCPI compatible instruments. SCPI is intended to give the ATE user a consistent environment for program development. It does so by defining controller messages, instrument responses, and message formats for all SCPI compatible instruments. The IEEE-488 (GPIB) interface for the 690XXA was designed to conform to the requirements of SCPI 1993.0. The set of SCPI commands implemented by the 690XXA GPIB interface provides a comprehensive set of programming functions covering all the major functions of the 690XXA CW generator.

SCPI Command Types

SCPI commands, which are also referred to as SCPI instructions, are messages to the instrument to perform specific tasks. The 690XXA command set includes:

- "Common" commands (IEE488.2 mandated commands)
- SCPI required commands
- SCPI optional commands (per SCPI 1993.0)
- SCPI compliant commands that are unique to the 690XXA.

The SCPI conformance information for the 690XXA command set is contained in Appendix B — SCPI Conformance Information.

Common Commands

∗CLS	∗RST
∗ESE	∗SRE
∗ESE?	∗SRE?
∗ESR?	∗STB?
∗IDN?	∗TST?
∗OPC	∗WAI
∗OPC?	

SCPI Required Commands

```
:STATus
    :OPERation
        [:EVENt]?
        :CONDition?
        :ENABle
    :PRESet
    :QUEStionable
        [:EVENt]?
        :CONDition?
        :ENABle
:SYSTem
    :ERRor?
    :VERSion?
```

Common Commands

The required common commands are IEEE-488.2 mandated commands that are defined in IEEE-488.2 and must be implemented by all SCPI compatible instruments. These commands (see table at left) are identified by the asterisk (*) at the beginning of the command keyword. These commands are used to control instrument status registers, status reporting, synchronization, and other common functions. The common commands and their syntax are described in detail in Chapter 3, paragraph 3-2.

Required and Optional SCPI Commands

The required SCPI commands are listed in the table at left and are described in detail in Chapter 3, paragraphs 3-11 and 3-12. The optional SCPI commands and 690XXA unique commands comprise the remainder (major portion) of the 690XXA command set. They control the majority of the programmable functions of the 690XXA CW generator. They are described in detail in Chapter 3 starting at paragraph 3-3.

Query Commands

All commands, unless specifically noted in the syntax descriptions in Chapter 3, have a query form. As defined in IEEE-488.2, a query is a command with a question mark symbol appended (examples: ∗ESR?, and :FREQuency:CENTer?). When the query form of a command is received, the current setting associated with that command is placed in the output buffer.

Command Names

Typical SCPI commands consist of one or more keywords, parameters, and punctuation. SCPI command keywords can be a mixture of upper and lower case characters. 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. For example, the long form of the command keyword to control the instrument display is: DISPlay.

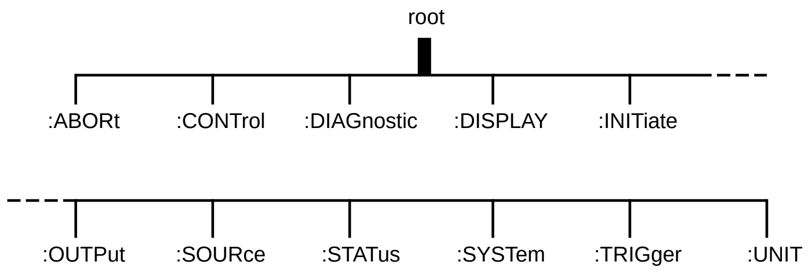
The short form keyword is usually the first four characters of the long form (example: DISP for DISPlay). The exception to this is when the long form is longer than four characters and the fourth character is a vowel. In such cases, the vowel is dropped and the short form becomes the first three characters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

Some command keywords may have a numeric suffix to differentiate between multiple instrument features such as a dual sweep capability. For example: keywords SWEep1 and SWEep2 (or SWE1 and SWE2) are used to differentiate between 690XXA frequency and power sweeps.

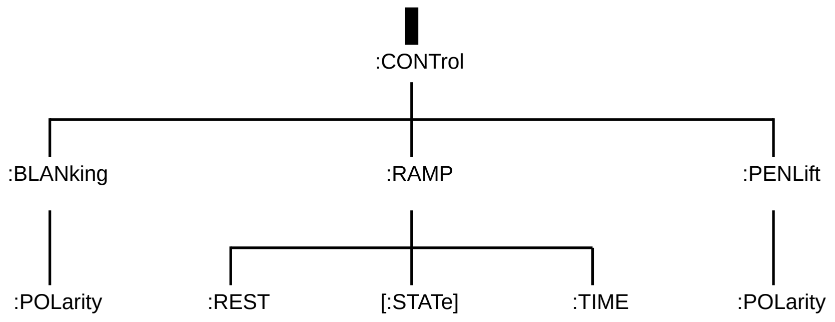
As with any programming language, the exact command keywords and command syntax must be used. The syntax of the individual commands is described in detail in Chapter 3. Unrecognized versions of long form or short form commands, or improper syntax, will generate an error. Error reporting is described in Chapter 4.

Hierarchical Command Structure

All SCPI commands, except the common commands, are organized in a hierarchical structure similar to the inverted tree file structure used in most computers. The SCPI standard refers to this structure as "the Command Tree." The command keywords that correspond to the major instrument control functions are located at the top of the command tree. The command keywords for the 690XXA SCPI command set are shown in the diagram below.



All 690XXA SCPI commands, except the ABORt command, have one or more subcommands (keywords) associated with them to further define the instrument function to be controlled. The subcommand keywords may in turn also have one or more associated subcommands (keywords). Each subcommand level adds another layer to the command tree. The command keyword and its associated subcommand keywords form a portion of the command tree called a command *subsystem*. The :CONTrol command subsystem is shown below.



An overall command tree for the 690XXA SCPI command set is shown in Figure A-1 of Appendix A.

Data Parameters

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) type parameters are used with commands that control switch functions.

The command descriptions in Chapter 3 specify the type of data parameter to be used with each command. The most commonly used parameter types are numeric, extended numeric, discrete, and Boolean.

Numeric
Numeric parameters comprise integer numbers, or any number in decimal or scientific notation and may include polarity signs. This includes <NR1>, <NR2>, and <NR3> numeric data as defined in Parameter Notations on page 2-9. This type of numeric element is abbreviated as <NRf> throughout this document.

Extended Numeric
Extended numeric parameters include values such as MAXimum and MINimum.

Discrete
Discrete parameters, such as INTernal and EXTernal, are used to control program settings to a predetermined finite value or condition.

Boolean
Boolean parameters represent binary conditions and may be expressed as ON, OFF, or 1, 0.

Unit Suffixes

Unit suffixes are not required for data parameters, provided the values are scaled for the global default units. The 690XXA SCPI default units are: Hz (Hertz) for frequency related parameters and S (seconds) for time related parameters. For example, the command below sets the 690XXA output frequency to 3 GHz.

```
    :SOURce:FREQuency:CW  3000000000
```

The global default units may be changed via use of the :UNIT Subsystem commands described in Chapter 3, paragraph 3-15.

2-3 NOTATIONAL CONVENTIONS

The SCPI interface standardizes command syntax and style which 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 (refer to Chapter 4 for error reporting).

General Notations

The syntax conventions that are 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 an instruction to the instrument parser to move down one level in the command tree hierarchy. A colon must precede a root keyword immediately following a semicolon. (See Notational Examples on page 2-10.)
;	A semicolon separates commands if multiple commands are placed on a single program line. (See Notational Examples on page 2-10.)
[]	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.
::=	means "is defined as." For example: <a>::=<b><c> indicates that <b><c> can replace <a>.
sp	space(s), referred to as whitespace, *must* be used to separate keywords from their associated data parameters. It *must not* be used between keywords, or inside keywords.
XXX	indicates a root command name.

For further information about SCPI command syntax and style, refer to the *Standard Commands for Programmable Instruments (SCPI) 1993.0* document.

Parameter Notations

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

<arg> ::=a generic command argument consisting of one or more of the other data types.

<bNR1> ::=boolean values in <NR1> format; numeric 1 or 0

<boolean> ::=ON|OFF. Can also be represented as 1 or 0, where 1 means ON and 0 means OFF. Boolean parameters are always returned as 1 or 0 in <NR1> format by query commands.

<integer> ::=an unsigned integer without a decimal point (implied radix point)

<NR1> ::=a signed integer without a decimal point (implied radix point).

<NR2> ::=a signed number with an explicit radix point.

<NR3> ::=a scaled explicit decimal point numeric value with and exponent (e.g., floating point number)

<NRf> ::=<NR1>|<NR2>|<NR3>

<nv>::=SCPI numeric value: <NRf>|MIN|MAX|UP |DOWN|DEF|NAN|INF|NINF or other types

<char> ::=<CHARACTER PROGRAM DATA>. Examples: CW, FIXed, UP, and DOWN

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

<block> ::=IEEE-488.2 block data format

<NA> ::=Not Applicable

Notational Examples

The following is an example showing command syntax (It is not an actual command):

[SOURce]:POWer[:LEVel][:IMMediate][:AMPLitude]:STEP[:INCRement] *sp* **dBm|DOWN|UP**

Command statements read from left to right and from top to bottom. In the command statement above, the :STEP keyword immediately follows the :AMPLitude keyword with no separating space. A space (*sp*) is used between the command string and its argument (a <nv> type data parameter).

Note that the first keyword in the command string does not require a leading colon; however, it is good practice to always use a leading colon for *all* keywords. Note also that the :SOURce keyword is optional. This is a SCPI convention for all signal source type instruments that allows shorter command statements to be used.

The following is an example of a multiple command statement that uses two seperate commands in a single statement. Note the semicolon used to join the commands. (Also note the leading colon used immediately after the semicolon.)

:FREQuency:STARt 10E6;:FREQuency:STOP 20E9

2-4 SCPI INTERFACE LANGUAGE SELECTION

The Series 690XXA Synthesized CW Generator can be remotely operated using one of two external interface languages—Native or SCPI. Before programming with SCPI commands it is necessary to select SCPI as the external interface language.

Front Panel Selection

SCPI can be selected as the 690XXA interface language from the front panel Configure GPIB menu.

To access the Configure GPIB Menu, first press the **SYSTEM** main menu key on the front panel to access the System Menu. At the menu display, press **Config** to access the System Configuration Menu. Then, press **GPIB**. The Configure GPIB Menu is displayed.

The Configure GPIB menu has an additional menu display. Language selection is made from this additional menu. To access the additional menu, press **More**. At the menu, press **Native SCPI** to select SCPI. The language selection will appear on the display.

Remote Selection

SCPI can be selected as the 690XXA interface language during remote operations.

To change the interface language from Native to SCPI use the command

```
SYST:LANG “SCPI”
```

Do ***not*** use the long form of the command and do ***not*** use a leading colon (:) with the command. The command :SYSTem:LANGuage "SCPI" results in a syntax error.

NOTE

When the series 690XXA CW generator is remotely operated using the SCPI interface language, cycling the power returns the instrument to reset condition.

2-5 STATUS SYSTEM PROGRAMMING

The 690XXA status system (shown in Figure 2-1) consists of the following SCPI-defined status-reporting structures:

- The Instrument Summary Status Byte Group
- The Standard Event Status Group
- The Operational Status Group
- The Questionable Status Group

The following paragraphs describe the registers that make up a status group and explain the status information that each status group provides.

Status Group Registers

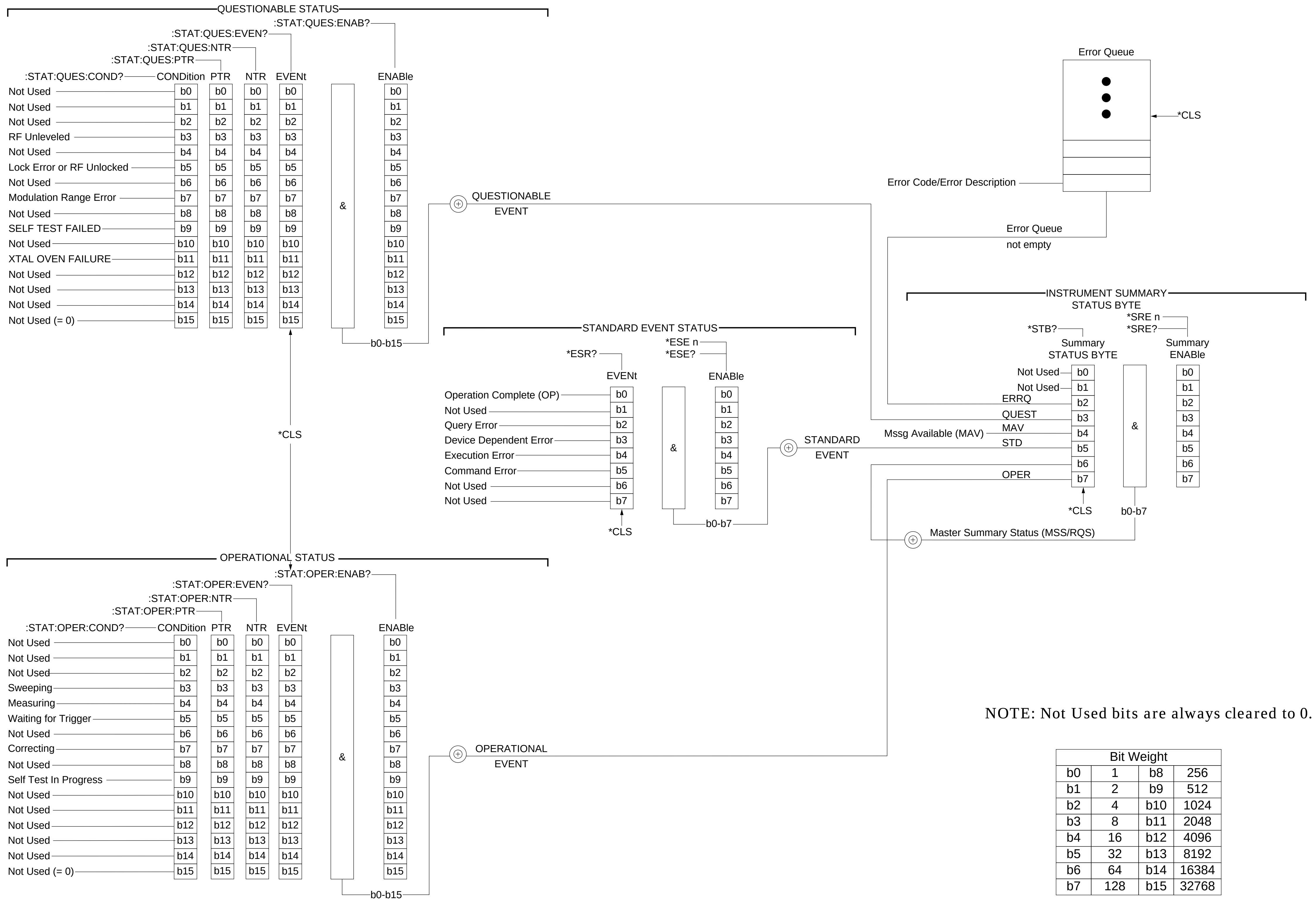
In general, a status group consists of a condition register, a transition filter, an event register, and an enable register. Each component is briefly described in the following paragraphs.

Condition Register
The condition register is continuously updated to reflect the current status of the 690XXA. 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); postive 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.



NOTE: Not Used bits are always cleared to 0.

Bit Weight			
b0	1	b8	256
b1	2	b9	512
b2	4	b10	1024
b3	8	b11	2048
b4	16	b12	4096
b5	32	b13	8192
b6	64	b14	16384
b7	128	b15	32768

Figure 2-1. 690XXA Status-Reporting Structure

Enable Register

The enable register specifies the bits in the event register that can produce a summary bit. The 690XXA 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 Summary Status Byte. 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.

Status Group Reporting

The state of certain 690XXA hardware and operational events and conditions can be determined by programming the status system. As shown in Figure 2-1, the three lower status groups provide status information to the Summary Status Byte group. The Summary Status Byte 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.

NOTE

Programming commands for the status system, including examples of command usage, can be found in Chapter 3, paragraph 3-11.

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

Summary Status Byte Group

The Summary Status Byte group, consisting of the Summary Status Byte Enable register and the Summary Status Byte, is used to determine the general nature of a 690XXA event or condition. The bits in the Summary Status Byte provide the following information:

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	Set to indicate that the 690XXA has data ready in its output queue.
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 690XXA has at least one reason to require service. This bit is also called the Master Summary Status Bit (MSS). The individual bits in the Status Byte are ANDed with their corresponding Service Request 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.

Standard Event Status Group

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 Summary Status Byte. The bits in the Standard Event Status register provide the following information:

Bit	Description
0	Set to indicate that all pending 690XXA 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 a 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.

Operational Status Group

The Operational Status group, consisting of the Operational Condition register, the Operational Positive Transition register, the Operational Negative Transition register, the Operational Event register, and the Operational Event Enable register, is used to determine the specific condition that set bit 7 in the Summary Status Byte. The bits in the Operational Event register provide the following information:

Bit	Description
0-2	Not Used. The bits are always set to 0.
3	Set to indicate that a sweep is in progress.
4	Set to indicate that the 690XXA is measuring.
5	Set to indicate that the 690XXA is in an armed “wait for trigger” state.
6	Not Used. The bit is always set to 0.
7	Set to indicate that the 690XXA is performing a correction.
8	Not Used. The bit is always set to 0.
9	Set to indicate that 690XXA self-test is in progress.
10-14	Not Used. The bits are always set to 0.
*15	Always 0. The use of Bit 15 is not allowed by SCPI.

Questionable Status Group

The Questionable Status group, consisting of the Questionable Condition 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 Summary Status Byte. The bits in the Questionable Event register provide the following information:

Bit	Description
0-2	Not Used. The bits are always set to 0.
3	Set to indicate an RF unleveled condition.
4	Not Used. The bit is always set to 0.
5	Set to indicate a phase-lock error or RF unlocked condition.
6	Not Used. The bit is always set to 0.
7	Set to indicate a modulation range error.
8	Not Used. The bit is always set to 0.
9	Set to indicate that self-test failed.
10	Not Used. The bit is always set to 0.
11	Set to indicate a failure of the crystal oven.
12-14	Not Used. The bits are always set to 0.
*15	Always 0. The use of Bit 15 is not allowed by SCPI.

2-6 TRIGGER SYSTEM PROGRAMMING

The 690XXA trigger system is used to synchronize sweep generator actions with software trigger commands. The 690XXA follows the layered trigger model used in SCPI instruments. The following paragraphs describe operation and programming of the sweep generator trigger system. The structure and components of the 690XXA trigger model are shown in Figure 2-2.

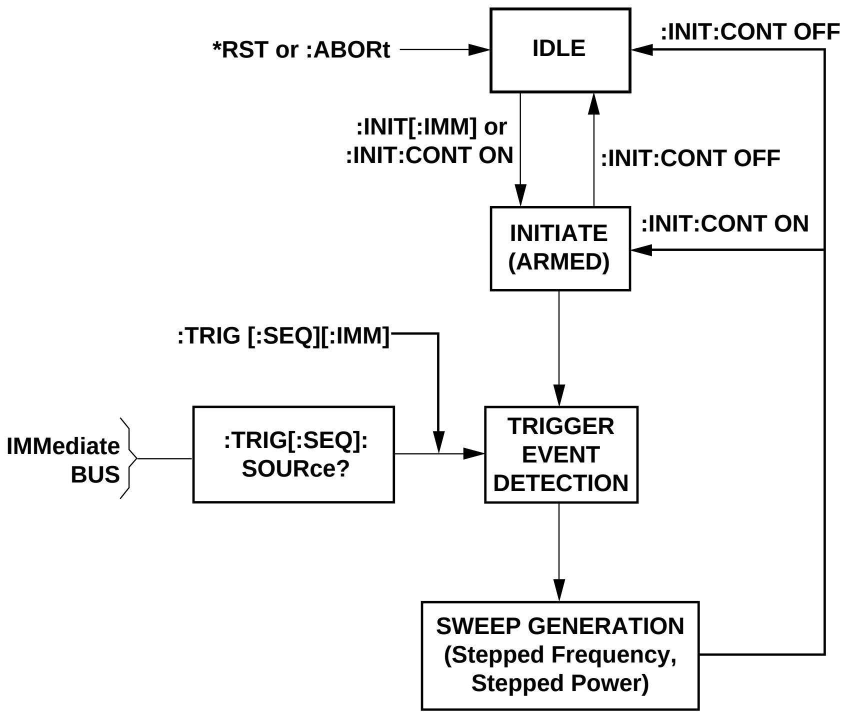


Figure 2-2. *690XXA Trigger Model*

Trigger System Operation

Turning power on, or sending ∗RST or :ABORt forces the trigger system into the *idle* state. The trigger system remains in the *idle* state until it is initiated. Trigger system initiation can happen on a continuous basis (:INITiate:CONTinuous ON) or on a demand basis (:INITiate:CONTinuous OFF). When the command :INITiate:CONTinuous is set to OFF, the trigger system is initiated by the :INITiate[:IMMediate] command. Note that ∗RST sets :INITiate:CONTinuous to OFF.

Once initiated, the trigger system enters an *armed* (wait for trigger) state. The trigger signal selected by the command :TRIGger[:SEQuence]:SOURce is examined until a TRUE condition is detected. The trigger signal selections are:

IMMediate	the trigger signal is always TRUE.
BUS	the trigger signal is either the GPIB <GET> (Group Execute Trigger) message or the ∗TRG command.
HOLD	the trigger signal is never TRUE.

When a TRUE condition is detected, sweep generation of the selected stepped sweep starts.

The command :TRIGger[:SEQuence][:IMMediate] provides a one-time override of the normal downward path in the trigger-event-detection state by forcing a TRUE trigger signal regardless of the setting for :TRIGger[:SEQuence]:SOURce.

Upon sweep completion, if :INITiate:CONTinuous is set OFF, the trigger system returns to the *idle* state. If :INITiate:CONTinuous is set to ON, the trigger system returns to the *armed* (wait for trigger) state.

Auto Trigger Mode
Setting the command :INITiate:CONTinuous to ON and the command :TRIGger[:SEQuence]:SOURce to IMMediate, places the trigger system in an auto trigger mode. This causes continuous generation of the selected stepped sweep.

ABORt
The :ABORt command resets any sweep in progress and immediately returns the trigger system to the *idle* state. Unlike ∗RST, :ABORt does not change the settings programmed by other commands.

Chapter 3 Programming Commands

Table of Contents

Table of Contents (Continued)

Table of Contents (Continued)

Table of Contents (Continued)

Chapter 3 Programming Commands

3-1 INTRODUCTION

This chapter contains information on all SCPI programming commands accepted and implemented by the Series 690XXA Synthesized CW Generator.

3-2 COMMON COMMANDS

Common commands are used to control instrument status registers, status reporting, synchronization, data storage, and other common functions. All common commands are identified by the leading asterisk in the command word. The common commands are fully defined in IEEE 488.2.

IEEE 488.2 Mandated Commands

The 690XXA implements the following IEEE-488.2 mandated common commands.

∗CLS (Clear Status Command)
Clear the Status Byte, the Questionable Status Event Register, the Standard Event Status Register, the Operational Status Event Register, the error queue, the OPC pending flag, and any other registers that are summarized in the Status Byte.

∗ESE *sp* <nv> (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" in Chapter 2.

∗ESE? (Standard Event Status Enable Query) ?
Returns the value of the Standard Event Status Enable Register in <NR1> format. Refer to "Status System Programming" in Chapter 2.

∗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" in Chapter 2.

∗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>, <Serial #>, <Firmware revision level>; where the actual model number, serial number, and firmware version of the 690XXA queried will be passed.

∗OPC (Operation Complete Command)
Enables the Operation Complete bit in the Standard Event Status Register after all pending operations are complete.

∗OPC? (Operation Complete Query)
Places an ASCII "1" in the Output Queue and sets the MAV bit true in the Status Byte when all pending operations are completed (per IEEE-488.2 section 12.5.3). Message is returned in <NR1> format.

∗RST (Reset Command)
Resets the 690XXA 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, Status Byte Register, Standard Event Register, or calibration data.

NOTE
This command clears the current front panel setup. If this setup is needed for future testing, save it as a stored setup using the ∗SAV command before issuing the ∗RST command.

∗SRE *sp* **<nv> (Service Request Enable Command)**
Sets the Service Request Enable Resister (Summary Status Byte Enable) bits. 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" in Chapter 2.

∗SRE? (Service Request Enable Query)
Returns the value of the Service Request Enable Register (Summary Status Byte Enable) in <NR1> format. Bit 6 is always zero.

∗STB? (Read Status Byte Query)
Returns the content of the Summary Status Byte Register (bits 0–5 and 7). Bit 6 is the Master Summary Status bit value. This command does not reset the status byte values.

∗TST? (Self-Test Query)
Causes the 690XXA to perform a full internal self-test. Status messages which 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.

CAUTION

690XXA self-test requires RF output power to be on. ***Always*** disconnect sensitive equipment from the unit before performing a self-test.

∗WAI (Wait-to-Continue Command)
This command suspends the execution of any further commands or queries until all operations for pending commands are completed. For example, the command ∗TRG;∗WAI permits synchronous sweep operation. It causes the 690XXA to start a sweep and wait until the sweep is complete before executing the next command.

Optional Common Commands

The 690XXA implements the following IEEE 488.2 optional common commands:

∗OPT? (Option Identification Query)
This command returns a string identifying any device options.

∗RCL *sp* <n> (Recall Stored State)
This command restores the 690XXA to a front panel setup state that was previously saved to local (instrument) memory using the ∗SAV command (below). The ∗RCL *sp* <n> command restores setup <n>, where n shall be in the range of 0 to 9.

∗SAV *sp* <n> (Save Current State)
Saves the current front panel setup parameters in local (instrument) memory. The new stored setup state will be assigned the Setup Number specified by <n>, where n shall be in the range of 0 to 9.

∗TRG (Trigger Command)
Triggers instrument if :TRIGger:SOURce command data parameter is BUS. Refer to INITiate and TRIGger subsystem commands.)

Performs the same function as the Group Execute Trigger <GET> command defined in IEEE 488.1.

3-3 SUBSYSTEM COMMANDS

Subsystem commands control all CW generator functions and some general purpose functions. All subsystem commands are identified by the colon used between keywords, as in :INITiate:CONTinuous.

The following information is provided for each subsystem command:

- The path from the subsystem root command.
- The data parameters used as arguments for the command. This includes the parameter type, the available parameter choices, the range for numeric parameters, and the default parameter that is set by the ∗RST command.
- A description of the purpose of the command.
- The query form of the command (if applicable).
- An example of the use of the command.
- Where necessary, notes are included to provide additional information about the command and its usage.

An overall command tree for the 690XXA SCPI command set is shown in Figure A-1 of Appendix A.

3-4 ABORT COMMAND (SUBSYSTEM)

The :ABORt command is a single command subsystem. There are no subcommands or associated data parameters, as shown below. The :ABORt command, along with the :TRIGger and :INITiate commands, comprise the "Trigger Group" of commands.

:ABORt

Parameters:	None
Description:	Forces the trigger system to the *idle* state. Any sweep in progress is aborted as soon as possible.
Query Form:	None
Example:	:ABORt *Sets 690XXA trigger system to idle state.*
Associated commands:	:TRIGger and :INITiate

3-5 CONTROL SUBSYSTEM

The :CONTrol subsystem sets the state of the following rear panel control outputs; RETRACE BLANK OUT, PENLIFT OUT, and HORIZ OUT. The subsystem commands and parameters are described below.

KEYWORD	PARAMETER FORM	NOTES
:CONTrol		
:BLANking		
:POLarity	**NORMal|INVerted**	Default: NORmal
:PENLift		
:POLarity	**NORMal|INVerted**	Default: NORmal
:RAMP		
:REST	**STARt|STOP**	Default: STOP
[:STATe]	**<boolean>**	Default: OFF
:TIME	**<numeric_value>**	Default: 30 ms

:CONTrol
:BLANking
:POLarity

Parameters: NORMal | INVerted
Type: <char>
Default: NORMal

Description: Sets the level of the retrace blanking signal output (rear panel AUX I/O connector, pin 6) during sweep retrace as follows:
NORMal causes the blanking signal to be a +5V level.
INVerted causes the blanking signal to be a –5V level.

Query Form :CONTrol:BLANking:POLarity?

Examples: :CONTrol:BLANking:POLarity *sp* INVerted
Set a –5V level for the rear panel blanking signal output during sweep retrace.

:CONTrol:BLANking:POLarity?
Requests the currently programmed level for the rear panel blanking signal output during sweep retrace.

:CONTrol

:PENLift

:POLarity

Parameters:	NORMal | INVerted
Type:	<char>
Default:	NORMal
Description:	Sets the internal penlift relay contacts to control the state of the penlift relay output (optionally available at the rear panel) as follows: NORMal sets the relay contacts to be normally open. INVerted sets the relay contacts to be normally closed.
Query Form:	:CONTrol:PENLift:POLarity?
Examples:	:CONTrol:PENLift:POLarity *sp* INVerted *Set the penlift relay contacts to be normally closed.* :CONTrol:PENLift:POLarity? *Requests the currently programmed state of the penlift relay contacts.*

:CONTrol

:RAMP

:REST

Parameters:	STARt | STOP
Type:	<char>
Default:	STOP
Description:	Sets the sweep rest point for the rear panel HORIZ OUT sweep ramp as follows: STARt sets the sweep to rest at the bottom of the sweep ramp. STOP sets the sweep to rest at the top of the sweep ramp.
Query Form:	:CONTrol:RAMP:REST?
Examples:	:CONTrol:RAMP:REST *sp* STOP *Set the sweep to rest at the top of the sweep ramp.* :CONTrol:RAMP:REST? *Requests the currently programmed rest point for the sweep ramp.*

:CONTrol

:RAMP

[:STATe]

Parameters:	ON | OFF | 1 | 0
Type:	<boolean>
Default:	OFF
Description:	Turns the rear panel HORIZ OUT sweep ramp signal on/off.
Query Form:	:CONTrol:RAMP[:STATe]?
Examples:	:CONTrol:RAMP:STATe *sp* ON *Turns the rear panel* HORIZ OUT *sweep ramp signal on.* :CONTrol:RAMP:STATe? *Requests the currently programmed state of the* HORIZ OUT *sweep ramp signal.*

:CONTrol

:RAMP

:TIME

Parameters:	sweep time (in seconds) | MIN | MAX
Type:	<NRf>
Range:	30 ms to 99 sec
Default:	30 ms

Description: Sets the rear panel HORIZ OUT sweep ramp signal time. [:SOURce]:SWEep:TIME will also be changed. May not be changed while the unit is sweeping.

Query Form: :CONTrol:RAMP:TIME?

Examples: :CONTrol:RAMP:TIME *sp* 100 ms

Sets the rear panel HORIZ OUT *sweep ramp signal time to 100 ms.*

:CONTrol:RAMP:TIME?

Requests the currently programmed time for the HORIZ OUT *sweep ramp signal.*

3-6 DIAGNOSTIC SUBSYSTEM

The :DIAGnostic subsystem consists of the query command described below.

KEYWORD
:DIAGnostic
:SNUM?

:DIAGnostic

:SNUM?

Description:	Allows the serial number of the instrument to be read.
Query Form	:DIAGnostic:SNUM?

3-7 DISPLAY SUBSYSTEM

The :DISPlay subsystem controls the display of all frequency and power level parameters on the front panel data display.

KEYWORD	PARAMETER FORM	NOTES
:DISPlay		
[:WINDow]		
:TEXT		
:STATe	**<boolean>**	Default ON

:DISPlay

[:WINDow]

:TEXT

:STATe

Parameters:	ON | OFF | 1 | 0
Type:	<boolean>
Default:	ON

Description: Turns the display of the frequency and power level parameters on the front panel data display on/off.

Query Form :DISPlay:TEXT:STATe?

Example: :DISPlay:TEXT:STATe *sp* OFF
Turns off the display of the frequency and power level parameters on the 690XXA front panel data display (Secure mode of operation).

3-8 INITIATE SUBSYSTEM

The :INITiate subsystem controls the state of the 690XXA trigger system. The subsystem commands and parameters are described below. The :INITiate commands, along with the :ABORt and :TRIGger commands, comprise the Trigger Group of commands.

KEYWORD	PARAMETER FORM	NOTES
:INITiate		
[:IMMediate]	(none)	
:CONTinuous	<boolean>	Default: OFF

:INITiate

[:IMMediate]

Parameters:	none
Description:	Places the 690XXA trigger system into the armed state from the idle state. If trigger system is not in idle state, or if :INITiate:CONTinuous is ON, will produce error –213.
Query Form:	None
Example:	:INITiate:IMMediate *Sets 690XXA trigger to the armed state.*
Associated commands:	:ABORt and :TRIGger

NOTES:

When :INITiate or :TSWeep is received by the 690XXA, all sweep-related parameters are checked for compatibility and bounds. The system will not arm is any errors exist. These errors are reported in the error queue.

:INITiate

:CONTinuous

Parameters:	ON | OFF | 1 | 0
Type:	<boolean>
Default:	OFF
Description:	Continuously rearms the 690XXA trigger system after completion of a triggered sweep.
Query Form:	:INITiate:CONTinuous?
Examples:	:INITiate:CONTinuous *sp* ON *Sets 690XXA trigger to continuously armed state.*
Associated commands:	:ABORt and TRIGger

NOTE:

:INITiate:CONTinuous ON has the same action as :INITiate:IMMediate plus it sets an internal flag that causes the trigger system to rearm after completing a triggered action.

If :TRIGger:SOURce IMMediate, :INITiate will start a sweep if one is not already in progress. In this case, to abort and restart a sweep either send :ABORt;:INITiate or :TSWeep .

If the trigger system is not idle, :INITiate will cause the error:
–213, "Init ignored, trigger not idle"

3-9 OUTPUT SUBSYSTEM

The :OUTPut subsystem controls the 690XXA RF output power. The commands are used to turn the RF output power on/off and to set the state of the RF output power during frequency changes in CW and step sweep modes and during sweep retrace. The subsystem commands and parameters are described below.

KEYWORD	PARAMETER FORM	NOTES
:OUTPut		
[:STATe]	<boolean>	Default: OFF
:PROTection	<boolean>	Default: ON
:RETRace	<boolean>	Default: OFF
:IMPedance?		

:OUTPut

[:STATe]

Parameters:	ON | OFF | 1 | 0
Type:	<boolean>
Default:	OFF (see note below)
Description:	Turns 690XXA RF output power on/off.
Query Form	:OUTPut[:STATe]?
Example:	:OUTPut:STATe *sp* ON *Turns 690XXA RF output power on.*

NOTE:

The SCPI programming mode reset default for RF output power state is OFF.
The 690XXA Native GPIB programming mode reset default for the RF output power state is ON.

:OUTPut

:PROTection

Parameters:	ON | OFF | 1 | 0
Type:	<boolean>
Default:	ON
Description:	ON causes the 690XXA RF output to be turned off (blanked) during frequency changes in CW or step sweep mode. OFF leaves RF output turned on (unblanked).
Query Form	:OUTPut:PROTection?
Example:	:OUTPut:PROTection *sp* OFF *Causes the 690XXA RF output signal to be left on during frequency changes in CW or step sweep mode.* :OUTPut:PROTection? *Requests the currently programmed state of the 690XXA RF output during frequency changes in CW or step sweep mode.*

:OUTPut

:PROTection

:RETRace

Parameters:	ON | OFF | 1 | 0
Type:	<boolean>
Default:	OFF
Description:	ON causes the 690XXA RF output to be turned off during sweep retrace. OFF leaves RF output turned on.
Query Form	:OUTPut:PROTection:RETRace?
Example:	:OUTPut:PROTection:RETRace *sp* ON *Turns the 690XXA RF output off during sweep retrace.* :OUTPut:PROTection:RETRace? *Requests the currently programmed state of the 690XXA RF output during sweep retrace.*

:OUTPut

:IMPedance?

Description:	Queries the 690XXA RF output impedance. The impedance is nominally 50 ohms and is not settable.
Query Form	:OUTPut:IMPedance?

3-10 SOURCE SUBSYSTEM

The [:SOURce] subsystem provides control of a majority of the 690XXA functions. The subsystem commands are used to control the frequency and power level of the RF output signal. The [:SOURce] subsystem commands and parameters are listed in the table contained on this and the following two pages. The subsytem commands are described in detail on following pages.

Note that the [:SOURce] keyword is optional for all command statements in the :SOURce subsystem. This is a SCPI convention for signal source type instruments that allows shorter command statements to be used.

:SOURce Subsystem Commands (1 of 3)

KEYWORD	PARAMETER FORM	NOTES
[:SOURce]		
:CORRection		
[:STATe]	**<boolean>**	Default: OFF
:CSET		
:SELect	**NONE | USER1 | USER2 | USER3 | USER4 | USER5**	Default: NONE
:FREQuency		
[:CW | :FIXed]	**<numeric_value>**	Default: (MIN+MAX)/2
STEP		
[:INCRement]	**<numeric_value>**	Default: 0.1 GHz
CENTer	**<numeric_value>**	Default: (MIN+MAX)/2
:MODE	**CW |FIXed| SWEep[1] | SWCW | ALSW**	Default: CW
:SPAN	**<numeric_value>**	Default: MAX–MIN
:FULL		
:START	**<numeric_value>**	Default: MIN
:STOP	**<numeric_value>**	Default: MAX
:MULTiplier	**<numeric_value>**	Default: 1
:MARKer<n>		Where: 1 ≤ n ≥ 10
:AOFF		
:FREQuency	**<numeric_value>**	
:STATe	**<boolean>**	Default: OFF
:VIDeo	**<boolean>**	Default: OFF
:POLarity	**POSitive | NEGative**	Default: POSitive

:SOURce Subsystem Commands (2 of 3)

KEYWORD	PARAMETER FORM	NOTES
[:SOURce]		
:POWer		
[:LEVel]		
[:IMMediate]		
[:AMPLitude]	**<numeric_value>**	Default: 0 dBm
:STEP		
[:INCRement]	**<numeric_value>**	Default: 0.1 dB
ALTernate	**<numeric_value>**	Default: 0 dBm
:ALC		
:GAIN	**<numeric_value>**	Default: 128
:STEP		
[:INCRement]	**<numeric_value>**	Default: 1
:SOURce	**INTernal | DIODe[1] | DIODe[2] |**	
	PMETer[1]|| PMETer[2] | FIXed	Default: INTernal
:ATTenuation	**<numeric_value>**	Default: 0 dB
:STEP		
[:INCRement]	**<Numeric_value>**	Default: 10 dB
:AUTO	**<boolean>**	Default: ON
:DISPlay		
:OFFSet	**<numeric_value>**	Default: 0 dB
:STATe	**<boolean>**	Default: OFF
:SLOPe	**<numeric_value>**	Default: 128
:STEP		
[:INCRement]	**<numeric_value>**	Default: 1
:STATe	**<boolean>**	Default: OFF
:PIVot	**<numeric_value>**	Default: 2 GHz
:MODE	**CW | FIXed | SWEep[1] | SWEep2 | ALSW**	Default: FIXed
	<numeric_value>	
:CENTer	**<numeric_value>**	Default: (MIN+MAX)/2
:SPAN		Default: (See Command)
:FULL	**<numeric_value>**	
:START	**<numeric_value>**	Default: MIN
:STOP		Default: MAX

:SOURce Subsystem Commands (3 of 3)

KEYWORD	PARAMETER FORM	NOTES
[:SOURce]		
:SWEep<n>		SWEep1 = freq sweep; SWEep2 = power sweep (see text). Default: 1
:DIRection	**UP | DOWN**	Default: UP
:DWELl	**<numeric_value>**	Default: 1 ms
:AUTO	**<boolean>**	Default: ON
:GENeration	**STEPped**	Default: STEPped
:POINts	**<numeric_value>**	Default: (See Command)
[:FREQuency]		
:STEP	**<numeric_value>**	Default: 1,999,900 Hz
:POWer		
:STEP	**<numeric_value>**	Default: (See Command)
:TIME	**<numeric_value>**	Default: (See Command)
:LLIMint	**<numeric_value>**	Default: 2 ms
:AUTO	**<boolean>**	Default: ON

The [:SOURce]:CORRection command and its subcommands comprise the Correction Subsystem within the :SOURce subsystem. These commands are used to select and apply level flatness correction to the 690XXA RF output. (Refer to “Leveling Operations” in Chapter 3 of the 690XXA Synthesized CW Generator Operation Manual.)

[:SOURce]

:CORRection

[:STATe]

Parameters:	ON | OFF | 1 | 0
Type:	<boolean>
Default:	OFF
Description:	Turns the selected user level flatness correction power-offset table on/off.
Query Form	[:SOURce]:CORRection[:STATe]?
Example:	[:SOURce]:CORRection:STATe *sp* ON *Turns on the selected user level correction power-offset table.*

NOTE:

If :CORRection:CSET:SELect is NONE, sending the command :CORRection:STATe ON returns an error.

[:SOURce]

:CORRection

:CSET

:SELect

Parameters:	NONE | USER1 | USER2 | USER3 | USER4 | USER5
Type:	<char>
Default:	NONE
Description:	Selects the user level flatness correction power-offset table to be applied to the 690XXA output by the command [:SOURce]:CORRection:STATe ON.
Query Form	[:SOURce]:CORRection:CSET:SELect?
Example:	[:SOURce]:CORRection:CSET:SELect *sp* USER3 *Selects user level flatness correction power-offset table #3.*

The [:SOURce]:FREQuency command and its subcommands make up the Frequency Subsystem within the :SOURce subsystem. These commands control the frequency characteristics of the 690XXA.

[:SOURce]

:FREQuency

[:CW | :FIXed]

Parameters:	frequency (in Hz) | UP | DOWN | MIN | MAX
Type:	<nv>
Range:	MIN to MAX (see notes below)
Default:	(MIN + MAX) / 2
Description:	Sets the RF output frequency of the 690XXA to the value entered. Parameters UP | DOWN increment/decrement the frequency by the value set by [:SOURce]:FREQuency:STEP:INCRement command.
Query Form:	[:SOURce]:FREQuency[:CW]?
Examples:	[:SOURce]:FREQuency:CW *sp* 3 GHz *or:* :FREQ *sp* 3 GHz *Sets the RF output frequency of the 690XXA to 3 GHz.* [:SOURce]:FREQuency:CW? *Requests the current value of the frequency parameter.*

NOTES:

Keywords :CW and :FIXed are equivalent and may be used interchangeably; they also are optional and may be omitted.

MIN ≤ frequency ≥ MAX; values for MINimum and MAXimum frequencies for each 690XXA model are listed in the table on the following page.

The query [:SOURce]:FREQuency:CW? *sp* MAX will return the upper frequency to which the particular model 690XXA may be programmed. Similarly, the query [:SOURce]:FREQuency:CW? *sp* MIN will return the lower frequency limit.

Model 690XXA MINimum and MAXimum Frequencies

Model	MINimum	MAXimum
69037A	2 GHz	20 GHz
69045A	500 MHz	20 GHz
69047A	10 MHz	20 GHz
69053A	2 GHz	26.5 GHz
69055A	500 MHz	26.5 GHz
69059A	10 MHz	26.5 GHz
69063A	2 GHz	40 GHz
69065A	500 MHz	40 GHz
69069A	10 MHz	40 GHz
69075A	500 MHz	50 GHz
69077A	10 MHz	50 GHz
69085A	500 MHz	60 GHz
69087A	10 MHz	60 GHz
69095A	500 MHz	65 GHz
69097A	10 MHz	65 GHz

[:SOURce]

:FREQuency

[:CW | :FIXed]

:STEP

[:INCRement]

Parameters:	frequency (in Hz)
Type:	<NRf>
Range:	1 KHz to (MAX – MIN) (see note below)
Default:	0.1 GHz

Description: Sets the step increment size used with the :FREQuency:CW command.

Query Form: [:SOURce]:FREQuency[:CW]:STEP [:INCRement]?

Examples:

```
[:SOURce]:FREQuency:CW:STEP
   :INCRement sp 1 MHz
   or:        :FREQ:STEP sp 1 MHz
```

Set the step increment value for the frequency parameter to 1 MHz.

```
[:SOURce]:FREQuency:CW:STEP:INCRement?
   or:        :FREQ:STEP?
```

Requests the current step increment value of the frequency parameter.

NOTE:

For 690XXAs equipped with Option 11, the minimum value for frequency step increment is 0.1 Hz. (The frequency resolution for standard models is 1.0 kHz; for models with Option 11 it is 0.1 Hz.)

[:SOURce]

:FREQuency

:CENTer

Parameters:	frequency (in Hz)
Type:	<NRf>
Range:	MIN to MAX (See Notes)
Default:	(MIN + MAX) / 2

Description: Sets the 690XXA RF output center frequency to the value entered. :CENTER and :SPAN frequencies are coupled values. Entering the value for one will cause the other to be recalculated. (See notes under :FREQuency :SPAN)

Query Form: `[:SOURce]:FREQuency:CENTer?`

Examples: `[:SOURce]:FREQuency:CENTer sp 4GHz`

Set the 690XXA RF output center frequency to 4GHz.

`[:SOURce]:FREQuency:CENTer?`

Requests the current value of the RF output center frequency.

NOTES:

Stepped Sweep Center Range = MIN to MAX, where:
MIN = MIN + minimum step size
MAX = MAX – minumum step size

[:SOURce]

:FREQuency

:MODE

Parameters:	CW | FIXed | SWEep[1] | SWCW | ALSW
Type:	<char>
Default:	CW

Description: Specifies which command subsystem controls the 690XXA frequency, as follows:

CW | FIXed	=	[:SOURce]:FREQuency:CW | FIXed
SWEep[1]	=	[:SOURce]:SWEep[1] (see notes)
SWCW	=	(see notes)
ALSW	=	(see notes)

In :SWEep[1] mode, output frequency is controlled by :STARt, :STOP, CENTer, and :SPAN commands. :SWEep and :SWEep1may be used interchangeably.

Query Form: [:SOURce]:FREQuency:MODE?

Examples: [:SOURce]:FREQuency:MODE *sp* CW

Specifies that the 690XXA RF frequency output is to be controlled by [:SOURce]:FREQuency:CW | FIXed *commands.*

[:SOURce]:FREQuency:MODE?

Requests the currently selected programming mode for frequency control.

NOTES:

In SWEep[1] mode, frequency will be determined by programmed values for the following :FREQuency subsystem commands: :CENTer and :SPAN, or, :STARt and :STOP.

Setting ALSW will cause the 690XXA to do alternate sweeping when properly triggered.

Setting FIXed will return CW upon query.

Setting SWCW will set CW and turn on CW ramp, the same as the command statement :FREQuency:MODE CW;:CONTrol:RAMP ON
A query returns CW.

[:SOURce]

:FREQuency

:SPAN

Parameters:	frequency (in Hz)
Type:	<NRf>
Range:	1 KHz to (MAX – MIN)
Default:	MAX – MIN

Description: Sets sweep span for SWEep[1] to value entered. :SPAN and :CENTer are coupled values (see notes below).

Query Form: [:SOURce]:FREQuency:SPAN?

Examples:

```
[:SOURce]:FREQuency:SPAN sp 2 GHz
   or:      :FREQ:SPAN sp 2 GHz
```

Set the SWEep[1] *sweep span to 2 GHz.*

```
[:SOURce]:FREQuencySPAN:?
```

Requests the current value for SWEep[1] *sweep span.*

NOTES:

:SPAN, :CENTer, :STARt, and :STOP are coupled values. Entering the value for :SPAN causes the values for :STARt and :STOP to be recalculated.

At ∗RST, :SPAN = Fmax – Fmin

[:SOURce]
:FREQuency
:SPAN
:FULL

Parameters:	None
Description:	Sets frequency span for SWEep[1] to (MAX ä MIN) (see notes under [:SOURce]:FREQuency:CW | FIXed).
Query Form:	None
Example:	[:SOURce]:FREQuency:SPAN:FULL *Set the* SWEep[1] *frequency span to its maximum value.*

[:SOURce]

:FREQuency

:SPAN2

Parameters:	frequency (in Hz)
Type:	<NRf>
Range:	1 KHz to (MAX – MIN)
Default:	MAX – MIN
Description:	Sets sweep span for the alternate sweep to value entered. :SPAN and :CENTer are coupled values (see notes below).
Query Form:	[:SOURce]:FREQuency:SPAN2?

Examples:

```
[:SOURce]:FREQuency:SPAN2 sp 2 GHz
   or:         :FREQ:SPAN2 sp 2 GHz
```

Set the sweep span for the alternate sweep to 2 GHz.

```
[:SOURce]:FREQuencySPAN2:?
```

Requests the current value of the sweep span for the alternate sweep.

NOTES:

:SPAN, :CENTer, :STARt, and :STOP are coupled values. Entering the value for :SPAN causes the values for :STARt and :STOP to be recalculated.

At ∗RST, :SPAN = Fmax – Fmin

[:SOURce]

:FREQuency

:SPAN2

:FULL

Parameters:	None
Description:	Sets frequency span for the alternate sweep to (MAX – MIN) (see notes under [:SOURce] :FREQuency:CW | FIXed).
Query Form:	None
Example:	[:SOURce]:FREQuency:SPAN:FULL *Set the frequency span for the alternate sweep to its maximum value.*

[:SOURce]

:FREQuency

:STARt

Parameters:	frequency (in Hz) | MIN
Type:	<nv>
Range:	MIN to MAX (See Notes)
Default:	MIN
Description:	Sets start frequency for SWEep[1] to the value entered. (MINimum is defined in the notes under [:SOURce]:FREQuency:CW | FIXed).
Query Form:	[:SOURce]:FREQuency:STARt?
Examples:	[:SOURce]:FREQuency:STARt *sp* 2.5 GHz *Set the start frequency (for* SWEep[1] *to 2.5 GHz .* [:SOURce]:FREQuency:STARt? *Requests the current value for* SWEep[1] *start frequency.*

NOTES:

Stepped Sweep Start Range = MIN to MAX, where:
MAX = MAX – 2 × minimum frequency step size

[:SOURce]

:FREQuency

:STARt2

Parameters:	frequency (in Hz) | MIN
Type:	<nv>
Range:	MIN to MAX (See Notes)
Default:	MIN

Description: Sets start frequency for the alternate sweep to the value entered. (MINimum is defined in the notes under [:SOURce]:FREQuency:CW | FIXed).

Query Form: [:SOURce]:FREQuency:STARt2?

Examples: [:SOURce]:FREQuency:STARt2 *sp* 3.5 GHz

Set the start frequency for the alternate sweep to 3.5 GHz .

[:SOURce]:FREQuency:STARt2?

Requests the current value for the alternate sweep start frequency.

NOTES:

Stepped Sweep Start Range = MIN to MAX, where:
MAX = MAX – 2 × minimum frequency step size

[:SOURce]

:FREQuency

:STOP

Parameters:	frequency (in Hz) | MAX
Type:	<nv>
Range:	MIN to MAX (See Notes)
Default:	MAX

Description: Sets stop frequency for SWEep[1] to the value entered. (MAXimum is defined in the notes under [:SOURce] :FREQuency:CW | FIXed).

Query Form: [:SOURce]:FREQuency:STOP?

Examples: [:SOURce]:FREQuency:STOP *sp* 15 GHz

Set the stop frequency (for SWEep[1] *to 15 GHz.*

[:SOURce]:FREQuency:STOP?

Requests the current value for SWEep[1] *stop frequency.*

NOTES:

Stepped Sweep Stop Range = MIN to MAX, where:
MIN = MIN + 2 × minimum frequency step size

[:SOURce]

:FREQuency

:STOP2

Parameters:	frequency (in Hz) | MAX
Type:	<nv>
Range:	MIN to MAX (See Notes)
Default:	MAX

Description: Sets stop frequency for the alternate sweep to the value entered. (MAXimum is defined in the notes under [:SOURce]:FREQuency:CW | FIXed).

Query Form: [:SOURce]:FREQuency:STOP2?

Examples: [:SOURce]:FREQuency:STOP2 *sp* 13 GHz

Set the stop frequency for the alternate sweep to 13 GHz.

[:SOURce]:FREQuency:STOP2?

Requests the current value for the alternate sweep stop frequency.

NOTES:

Stepped Sweep Stop Range = MIN to MAX, where:
MIN = MIN + 2 × minimum frequency step size

[:SOURce]
:FREQuency
:MULTiplier

Parameters:	reference multiplier value | MIN | MAX
Type:	<nv>
Range:	0.1 to 14; MIN = 0.1; MAX = 14
Default:	1

Description: Sets the value of the reference multiplier for the frequeny scaling function. This command affects all entered and displayed frequencies, but does not affect the output of the CW generator.

Query Form: [:SOURce]:FREQuency:MULTiplier?

Examples: [:SOURce]:FREQuency:MULTiplier *sp* 4

Set the frequency scaling reference multiplier value to 4.

[:SOURce]:FREQuency:MULTiplier?

Requests the current value for the frequency scaling reference multiplier.

NOTES:

The coupling equation is:
Entered/Displayed Frequency = (Hardware Frequency x Multiplier)

At ∗RST, the value is set to 1.

The [:SOURce]:MARKer command and its subcommands comprise the Marker Subsystem within the :SOURce subsystem. These commands control the Frequency Marker function of the 690XXA.

[:SOURce]

:MARKer<n> (see note)

:AOFF

Parameters:	None
Description:	Turns *all* markers off. This command is an event, therefore there is no data parameter and no query form.
Query Form:	None
Examples:	[:SOURce]:MARKer:AOFF *Turn all markers off.*

NOTES:

A numeric suffix <n> may be appended to any of the MARKer command headers. This specifies which of the 10 markers (1 to 10) is being altered by the command. In some cases, the command function is global to all 10 markers, such as the command :MARKers<n>:AOFF (all markers off). In these cases, the <n> in the command header is ignored.

For further information about frequency markers, refer to Frequency Markers in the Series 690XXA Synthesized CW Generator Operation Manual.

[:SOURce]

:MARKer<n> (1 ≤ **n** ≤ 10 = selected marker; see notes)

:FREQuency

Parameters:	frequency (in Hz) | MIN | MAX
Type:	<nv>
Range:	MIN to MAX
Default:	See default values in notes below
Description:	Sets frequency of selected marker to value entered. (MINimum and MAXimum are defined in the notes under [:SOURce]:FREQuency:CW | FIXed).
Query Form:	[:SOURce]:MARKer<n>:FREQuency?
Examples:	[:SOURce]:MARKer3:FREQuency *sp* 2 GHz *Set the frequency of marker #3 to 2 GHz.* [:SOURce]:MARKer5:FREQuency? *Requests the current frequency value of marker #5.*

NOTES:

:MARKer (or :MARK) is equivalent to :MARKer1 (or :MARK1).

When 690XXA is powered up, or when ∗RST command is issued: markers 1 – 10 are set to their default frequency values as follows:

MARKER:	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
FREQ (GHz):	2.0	20.0	2.0	5.0	8.0	11.0	14.0	17.0	20.0	3.5

Marker 10 accesses what is displayed as Marker 0 on the 690XXA front panel data display.

[:SOURce]

:MARKer<n> (1 ≤ **n** ≤ 10 = selected marker)

:STATe

Parameters:	ON | OFF | 1 | 0
Type:	<boolean>
Default:	OFF
Description:	Turns selected marker on/off (tags/untags the selected marker).
Query Form:	[:SOURce]:MARKer<n>:STATe?
Examples:	[:SOURce]:MARKer4:STATe *sp* ON *Turn marker #4 on.* [:SOURce]:MARKer7:STATe? *Requests current state of marker #7.*

[:SOURce]

:MARKer<n>

:VIDeo

Parameters:	ON | OFF | 1 | 0
Type:	<boolean>
Default:	OFF
Description:	Turns video markers on/off.
Query Form:	[:SOURce]:MARKer:VIDeo?
Examples:	[:SOURce]:MARKer:VIDeo *sp* ON *Turns on video markers.*

NOTE:

Setting [:SOURce]:MARKer:VIDeo to OFF turns off all markers.

[:SOURce]

:MARKer<n>

:POLarity

Parameters:	POSitive | NEGative
Type:	<char>
Default:	POSitive
Description:	Selects +5V or –5V pulse output for each video marker as follows: POSitive selects a +5V pulse output for each marker. NEGative selects a –5V pulse output for each marker.
Query Form:	[:SOURce]:MARKer:POLarity?
Examples:	[:SOURce]:MARKer:POLarity *sp* NEGative *Specifies –5V pulse output for each video marker.* [:SOURce]:MARKer:POLarity? *Requests current polarity of the pulse output for each video marker.*

The [:SOURce]:POWer command and its subcommands comprise the Power Subsystem within the :SOURce subsystem. These commands control the RF power output level of the 690XXA.

[:SOURce]

:POWer

[:LEVel]

[:IMMediate]

[:AMPLitude]

Parameters:	power level (in dBm) | UP | DOWN | MIN | MAX
Type:	<nv>
Range:	MIN to MAX (see notes below)
Default:	0 dBm

Description: Sets the power level of the unswept RF output signal (see notes below).

Query Form:
```
[:SOURce]:POWer[:LEVel][:IMMediate]
   [:AMPLitude]?
```

Examples:
```
[:SOURce]:POWer:LEVel:IMMediate
   :AMPLitude sp 10 dBm
```
Sets the RF output power level to +10 dBm.

```
[:SOURce]:POWer:LEVel:IMMediate
   :AMPLitude?
```
Requests the value currently programmed for the 690XXA RF output power level.

```
[:SOURce]:POWer? MAX
```
Requests the maximum RF output power value that can be programmed for the particular 690XXA model.

NOTES:

The MINimum and MAXimum RF output power levels that can be entered for each 690XXA model are listed in the table on the following page. Note that these power levels are the limits of what can be entered, but *do not guarantee leveled operation*.

For units equipped with an attenuator (Option 02), changes in RF power output level >10 dB may change attenuator setting (see command [:SOURce]:POWer:ATTenuation:AUTO).

Use related command :OUTPut[:STATe] *sp* ON | OFF to turn the 690XXA RF power output on/off.

Model 690XXA MINimum and MAXimum Power Levels

Model	W/O Step Attenuator ①		W/ Step Attenuator (Option 2) ②	
	MINimum	MAXimum	MINimum	MAXimum
69037A	–20 dBm	+20 dBm	–130 dBm	+20 dBm
69045A	–20 dBm	+20 dBm	–130 dBm	+20 dBm
69047A	–20 dBm	+20 dBm	–130 dBm	+20 dBm
69053A	–20 dBm	+20 dBm	–130 dBm	+20 dBm
69055A	–20 dBm	+20 dBm	–130 dBm	+20 dBm
69059A	–20 dBm	+20 dBm	–130 dBm	+20 dBm
69063A	–20 dBm	+20 dBm	–130 dBm	+20 dBm
69065A	–20 dBm	+20 dBm	–130 dBm	+20 dBm
69069A	–20 dBm	+20 dBm	–130 dBm	+20 dBm
With Option 15A (High Power) Installed				
69037A	–10 dBm	+30 dBm	–120 dBm	+30 dBm
69045A	–10 dBm	+30 dBm	–120 dBm	+30 dBm
69047A	–10 dBm	+30 dBm	–120 dBm	+30 dBm
69053A	–10 dBm	+30 dBm	–120 dBm	+30 dBm
69055A	–10 dBm	+30 dBm	–120 dBm	+30 dBm
69059A	–10 dBm	+30 dBm	–120 dBm	+30 dBm
69063A	–10 dBm	+30 dBm	–120 dBm	+30 dBm
69065A	–10 dBm	+30 dBm	–120 dBm	+30 dBm
69069A	–10 dBm	+30 dBm	–120 dBm	+30 dBm
69075A	–10 dBm	+30 dBm	–120 dBm	+30 dBm
69077A	–10 dBm	+30 dBm	–120 dBm	+30 dBm
69085A	–10 dBm	+30 dBm	–120 dBm	+30 dBm
69087A	–10 dBm	+30 dBm	–120 dBm	+30 dBm
69095A	–10 dBm	+30 dBm	Not Available	
69097A	–10 dBm	+30 dBm	Not Available	

① or units with a step attenuator (Option 2) and :POWer:ATTenuation:AUTO OFF

② :POWer:ATTenuation:AUTO ON

[:SOURce]
:POWer
[:LEVel]
[:IMMediate]
[:AMPLitude]
:STEP
[:INCRement]

Parameters:	power level (in dB)
Type:	<NRf>
Range	Model dependent (see notes below)
Default:	0.1 dB

Description: Sets the step increment size used with the :POWer:LEVel:IMMediate:AMPLitude command.

Query Form:
```
[:SOURce]:POWer[:LEVel][:IMMediate]
   [:AMPLitude]:STEP[:INCRement]?
```

Examples:
```
[:SOURce]:POWer:LEVel:IMMediate
   :AMPLitude:STEP:INCRement sp 5 dBm
```
Set the step increment value for RF output power level parameter to 5 dBm.

```
[:SOURce]:POWer:LEVel:IMMediate
   :AMPLitude:STEP:INCRement?
```
Requests the current step increment value for the RF output power level parameter.

NOTES:

For standard 690XXA models, a maximum step size up to 28 dB may be used (up to 131 dB for models with option 2 step attenuator). For 690XXA models with option 15A, a maximum step size up to 22 dB can be used (up to 125 dB with option 2 step attenuator). However, the step size, in conjunction with the initial RF power output setting, must not produce a programmed power level below the minimum leveled output power for the particular 690XXA model. Refer to Appendix B — Performance Specifications — in the Series 690XXA Synthesized CW Generator Operation Manual.

[:SOURce]

:POWer

[:LEVel]

:ALTernate

Parameters:	power level (in dBm) | MIN | MAX
Type:	<nv>
Range:	MIN to MAX (see notes below)
Default:	0 dBm
Description:	Sets the RF output power level for the alternate sweep if the command :POWer:MODE ALSW is set.
Query Form:	[:SOURce]:POWer[:LEVel]:ALTernate?
Examples:	[:SOURce]:POWer:LEVel:ALTernate *sp* 5 dBm *Sets the RF power level for the alternate sweep to +5 dBm.* [:SOURce]:POWer:LEVel:ALTernate? *Requests the currently programmed value for the alternate sweep.*

NOTES:

The MINimum and MAXimum RF output power levels that can be entered for each 690XXA model are listed in the table on page 3-48. Note that these power levels are the limits of what can be entered, but *do not guarantee leveled operation*.

For units equipped with an attenuator (Option 02), changes in RF power output level >10 dB may change attenuator setting (see command [:SOURce]:POWer:ATTenuation:AUTO).

Use related command :OUTPut[:STATe] *sp* ON | OFF to turn the 690XXA RF power output on/off.

[:SOURce]

:POWer

:ALC

:GAIN

Parameters:	<integer> | UP | DOWN | MIN | MAX
Type:	<nv>
Range:	0 – 255; MIN = 0; MAX = 255
Default:	128

Description: Sets the ALC gain when using the feedback signal from an external detector or power meter. This is accomplished by setting the Level Reference DAC to the binary weighted integer value entered. (Refer to “Leveling Operations” in Chapter 3 of the Series 690XXA Synthesized CW Generator Operation Manual.)

Parameters UP | DOWN increment/decrement the DAC setting by the value set by the [:SOURce]:POWer :ALC:GAIN:STEP[:INCRement] command.

Query Form: [:SOURce]:POWer:ALC:GAIN?

Examples: [:SOURce]:POWer:ALC:GAIN *sp* 100

Sets the ALC gain (Level Reference DAC setting) to a value of 100.

[:SOURce]:POWer:ALC:GAIN?

Requests the currently programmed value for the Level Reference DAC setting (ALC gain).

[:SOURce]
:POWer
:ALC
:GAIN
:STEP
[:INCRement]

Parameters:	ALC gain step size
Type:	<nv>
Range:	0 – 255
Default:	1

Description: Sets the ALC gain (Level Reference DAC setting) step increment size used with the :POWer:ALC:GAIN command.

Query Form: [:SOURce]:POWer:ALC:GAIN:STEP [:INCRement]?

Examples [:SOURce]:POWer:ALC:GAIN:STEP :INCRement *sp* 5

Set the step increment value for the Level Reference DAC setting (ALC gain) to 5.

[:SOURce]:POWer:ALC:GAIN:STEP :INCRement?

Requests the current step increment value for the Reference Level DAC setting (ALC gain).

[:SOURce]

:POWer

:ALC

:SOURce

Parameters:	INTernal | DIODe[1] | DIODe2 | PMETer[1] | PMETer2 | FIXed
Type:	<char>
Default:	INTernal

Description: Selects (1) whether the ALC loop controls the output power level and (2) the source of the feedback signal for the ALC.

FIXed places the instrument in a fixed gain power level mode (ALC off). RF output power is unleveled; use the [:SOURce]:POWer[:LEVel][:IMMediate][:AMPLitude] command to set the output power to the desired level.

The other parameters turn on the ALC function and select the source of the ALC feedback signal. INTernal specifies the ALC feedback signal from the instrument's internal level detector. The remaining parameter choices select an external detector or power meter signal as the feedback signal source for the ALC, as follows:

DIODe[1]	=	Detector output connected to the front panel EXTERNAL ALC input
PMETer[1]	=	Power Meter output connected to the front panel EXTERNAL ALC input
DIODe2	=	Detector output connected to the rear panel EXT ALC IN input
PMETer2	=	Power Meter output connected to the rear panel EXT ALC IN input

Query Form: `[:SOURce]:POWer:ALC:SOURce?`

Examples: `[:SOURce]:POWer:ALC:SOURce` *sp* `PMETer2`

Select external power meter output connected to the rear panel EXT ALC IN *input as the feedback signal for the ALC.*

`[:SOURce]?:POWer:ALC:SOURce?`

Requests the currently programmed source of the feedback signal for the ALC.

[:SOURce]

:POWer

:ATTenuation

Parameters:	attenuation (in dB) | UP | DOWN | MIN | MAX
Type:	<nv>
Range:	0 – 110 dB(0 – 90 dB for 69075A, 69077A, 69085A, and 69087A models)
Default:	0 dB

Description: *This command applies only to 690XXAs equipped with an internal step attenuator (Option 02).* This command sets the step attenuator (in 10 dB steps) throughout its 110 dB range (90 dB range for 69075A, 69077A, 69085A, and 69087A models); see notes below.

This command decouples the step attenuator from the automatic leveling control (ALC) system (see the command :POWer:ATTenuation:AUTO OFF).

Query Form: [:SOURce]:POWer:ATTenuation?

Examples: [:SOURce]:POWer:ATTenuation *sp* 100

Sets the 690XXA step attenuator setting to 100 dB.

[:SOURce]:POWer:ATTenuation?

Requests the currently programmed value for the step attenuator setting.

WARNING

If POWer:ATTenuation:AUTO was set ON, the command POWer:ATTenuation <arg> will momentarily set output power and attenuation to 0 before going to the programmed attenuation. If this could possibly damage equipment, set :OUTput[:STATe] OFF before issuing this command.

NOTES:

Values entered for parameter <dB> must be exact multiples of 10 db. Parameters UP and DOWN can be used to increase/decrease the step attenuator setting in 10 dB steps.
MIN = 0 dB attenuation, MAX = 110 dB attenuation.

If the 690XXA does not have a step attenuator installed, sending this command will result in –241,"Hardware missing;Attenuator" .

[:SOURce]

:POWer

:ATTenuation

:STEP

[:INCRement]

Parameters:	attenuator step increment sized (in dB)
Type:	<NR1>
Range:	0 dB | 10 dB
Default:	10 dB

Description: Sets the attenuator step increment size used with the :POWer:ATTenuation command. The step size can only be set to 0 dB or 10 dB.

Query Form:
```
[:SOURce]:POWer:ATTenuation:STEP
   [:INCRement]?
```

Examples:
```
[:SOURce]:POWer:ATTenuation:STEP
   :INCRement sp 10
```
Sets the step increment value to 10 dB.

```
[:SOURce]:POWer:ATTenuation:STEP
   :INCRement?
```
Requests the current step increment size for the step attenuator.

[:SOURce]

:POWer

:ATTenuation

:AUTO

Parameters:	ON | OFF | 1 | 0
Type:	< boolean>
Default:	ON

Description: *This command applies only to 690XXAs equipped with an internal step attenuator (Option 02).* Setting to ON couples the step attenuator to the ALC system; setting to OFF decouples the step attenuator from the ALC system.

Query Form: [:SOURce]:POWer:ATTenuation:AUTO?

Examples: [:SOURce]:POWer:ATTenuation:AUTO *sp* ON

Couple the step attenuator to the ALC system.

[:SOURce]:POWer:ATTenuation:AUTO?

Requests the currently programmed coupling state for the step attenuator.

WARNING

POWer:ATTenuation:AUTO OFF/ON sets the output power and attenuation to 0. If this could possibly damage equipment, set :OUTput[:STATe] OFF before issuing this command.

NOTES:

In 690XXAs without a step attenuator, this command's value is always OFF. Attempting to set it ON results in the error message –241,"Hardware missing;Attenuator".

[:SOURce]

:POWer

:DISPlay

:OFFSet

Parameters:	power level display offset (in dB)
Type:	<NRf>
Range:	–100.00 to +100.00 dB
Default:	0 dB

Description: Sets the offset value for the power display offset function (see :POWer:DISPlay:OFFSet:STATe).

Query Form: [:SOURce]:POWer:DISPlay:OFFSet?

Examples: [:SOURce]:POWer:DISPlay:OFFSet *sp 10 dBm*

Sets the power level display offset value to +10 dBm.

[:SOURce]:POWer:DISPlay:OFFSet?

Requests the current power level display offset value.

[:SOURce]
:POWer
:DISPlay
:OFFSet
:STATe

Parameters:	ON | OFF | 1 | 0
Type:	<boolean>
Default:	OFF

Description: Turns the power display offset function on/off. When the function is turned on, the offset value, set by the command :POWer:DISPlay:OFFSet <arg>, is added to the displayed RF output power level. A negative offset value decreases the displayed power level.

Query Form: [:SOURce]:POWer:DISPlay:OFFSet:STATe?

Examples:

```
[:SOURce]:POWer:DISPlay:OFFSet
   :STATe sp ON
```

Turns on the power display offset function.

```
[:SOURce]:POWer:DISPlay:OFFSet:STATe?
```

Requests the current state of the power display offset function.

[:SOURce]

:POWer

:SLOPe

Parameters:	slope characteristic value | UP | DOWN | MIN | MAX | DEF
Type:	<nv>
Range:	0 to 255; MIN = 0; MAX = 255
Default:	128

Description: Sets the value of the slope characteristic parameter for the ALC power slope function (refer to “Leveling Operations” in Chapter 3 of the 690XXA Synthesized CW Generator Operation Manual). This parameter is a relative number that defines the slope characteristic; a value of 128 produces a 0 dB/V slope.

Parameters UP | DOWN increment/decrement the characteristic parameter by the value set by the [:SOURce]:POWer:SLOPe:STEP[:INCRement] command.

Query Form: [:SOURce]:POWer:SLOPe?

Examples: [:SOURce]:POWer:SLOPe *sp* 140

Sets the value of the slope characteristic for the power slope function to 140.

[:SOURce]:POWer:SLOPe?

Requests the current value of the slope characteristic for the power slope function.

[:SOURce]
:POWer
:SLOPe
:STEP
[:INCRement]

Parameters:	slope step increment size
Type:	<NR1>
Range:	0 to 255
Default:	1

Description: Sets the step increment size used with the :POWer :SLOPe command (ALC power slope function).

Query Form: [:SOURce]:POWer:SLOPe:STEP[:INCRement]?

Examples: [:SOURce]:POWer:SLOPe:STEP :INCRement *sp* 5

Sets the step increment size to 5 for the ALC power slope function parameter.

[:SOURce]:POWer:SLOPe:STEP:INCRement?

Requests the current step increment size for the ALC power slope parameter.

[:SOURce]

:POWer

:SLOPe

:STATe

Parameters:	ON | OFF | 1 | 0
Type:	<boolean>
Default:	OFF
Description:	Turns ALC power slope function on/off (refer to “Leveling Operations” in Chapter 3 of the 690XXA Synthesized CW Generator Operation Manual).
Query Form:	[:SOURce]:POWer:SLOPe:STATe?
Examples:	[:SOURce]:POWer:SLOPe:STATe *sp* ON *Turns the ALC power slope function on.*

[:SOURce]

:POWer

:SLOPe

:PIVot

Parameters:	frequency (in Hz)
Type:	<NR1>
Range:	Frequency range of the 690XXA model
Default:	2 GHz

Description: Sets the frequency where the ALC power slope function correction is zero (pivot point). The frequency range for this function is model dependent (see notes under [:SOURce]:FREQuency[:CW | :FIXed]).

Query Form: [:SOURce]:POWer:SLOPe:PIVot?

Examples: [:SOURce]:POWer:SLOPe:PIVot *sp* 5 GHz

Sets the ALC power slope function pivot point to 5 GHz.

[:SOURce]

:POWer

:MODE

Parameters:	CW | FIXed | SWEep[1] | SWEep2 | ALSW
Type:	<char>
Default:	FIXed

Description: Specifies which set of commands controls the 690XXA RF power output level determining function, as follows:

```
CW | FIXed   =   [:SOURce]:POWer[:LEVel]
                    [:IMMediate][:AMPLitude]
SWEep[1]     =   [:SOURce]:SWEep[1]
SWEep2       =   [:SOURce]:SWEep2
ALSW         =   [:SOURce]:POWer[:LEVel]
                   :ALTernate
```

Query Form: [:SOURce]:POWer:MODE?

Examples: [:SOURce]:POWer:MODE *sp* CW

Specifies that the 690XXA RF power output level is to be controlled by [:SOURce]:POWer:CW | FIXed *commands.*

[:SOURce]:POWer:MODE?

Requests the currently selected programming mode for RF power output level control.

NOTES:

In SWEep[1] and SWEep2 modes, RF output power level is determined by programmed values for the following :POWer subsystem commands: :CENTer and :SPAN, or, :STARt and :STOP.

When [:SOURce]:POWer:Mode is set to CW, the query [:SOURce]:POWer:MODE? will return FIXed.

[:SOURce]

:POWer

:CENTer

Parameters:	power level (in dBm)
Type:	<NRf>
Range:	MIN to MAX (see notes below)
Default:	(MIN + MAX) / 2

Description: Sets the RF output power level at the center of the power sweep to the value entered. See notes below.

Query Form: [:SOURce]:POWer:CENTer?

Examples: [:SOURce]:POWer:CENTer *sp* –20 dBm

Set the RF output power level at the center of the power sweep span to –20 dBm.

[:SOURce]:POWer:CENTer?

Requests the currently programmed value for the RF output power level at the center of the power sweep span.

NOTES:

The MINimum and MAXimum RF output power levels that can be entered for each 690XXA model are listed in the table on page 3-48. Note that these power levels are the limits of what can be entered, but *do not guarantee leveled operation*.

:CENTer and :SPAN are coupled values. Setting the value for one will cause the other to be recalculated. See notes for the command :POWer:SPAN.

[:SOURce]

:POWer

:SPAN

Parameters:	power level (in dBm)
Type:	<NRf>
Range:	MIN to MAX (see notes).
Default:	Leveled power span of the instrument
Description:	Sets sweep span for power sweep to value entered. See notes below.
Query Form:	[:SOURce]:POWer:SPAN?
Examples:	[:SOURce]:POWer:SPAN *sp* 10 dBm *Set the power sweep span to 10 dBm.* [:SOURce]:POWer:SPAN:? *Requests the current value for power sweep span.*

NOTES:

:SPAN, :CENTer, :STARt, and :STOP are coupled values. Entering the value for :SPAN cause the values for :STARt and :STOP to be recalculated.

When the 690XXA is powered up, or when ∗RST command is issued, :SPAN is set to the leveled power span of the instrument, and :CENTer is set to (MIN + MAX) / 2.

The MINimum and MAXimum RF output power levels that can be entered for each 690XXA model are listed in the table on page 3-48. Note that these power levels are the limits of what can be entered, but *do not guarantee leveled operation*.

[:SOURce]

:POWer

:SPAN

:FULL

Parameters:	None
Description:	Sets the power sweep span to (MAX – MIN). See notes under :POWer:SPAN command.
Query Form:	None
Example:	[:SOURce]:POWer:SPAN:FULL *Set the power sweep span to its maximum value.*

[:SOURce]

:POWer

:STARt

Parameters:	power level (in dBm) | MIN
Type:	<nv>
Range:	MIN to MAX
Default:	MIN

Description: Sets start RF output power level for the power sweep to the value entered. See notes under :POWer:SPAN command.

Query Form: [:SOURce]:POWer:STARt?

Examples: [:SOURce]:POWer:STARt *sp* –10 dBm

Set the start RF output power level for the power sweep to –10 dBm.

[:SOURce]:POWer:STARt?

Requests the currently programmed start RF output power level for the power sweep.

[:SOURce]

:POWer

:STOP

Parameters:	power level (in dBm) | MAX
Type:	<nv>
Range:	MIN to MAX
Default:	MAX

Description: Sets stop RF output power level for the power sweep to the value entered. See notes under :POWer:SPAN command.

Query Form: [:SOURce]:POWer:STOP?

Examples: [:SOURce]:POWer:STOP *sp* 10 dBm

Set the stop RF output power level for the power sweep to +10 dBm.

[:SOURce]:POWer:STOP?

Requests the currently programmed stop RF output power level for the power sweep.

The [:SOURce]:SWEep command and its subcommands comprise the Sweep Subsystem within the :SOURce subsystem. These commands control the standard stepped frequency sweep functions and the step power level sweep function of the 690XXA.

[:SOURce]

:SWEep<n> (1 ≤ **n** ≤ 2; see note)

:DIRection

Parameters:	UP | DOWN
Type:	<char>
Default:	UP

Description: Selects the direction of sweep, in the :SWEep2 (power level sweep) mode only. Sweep direction is always in the UP direction for :SWEep1 (frequency sweeps).

Query Form: [:SOURce]:SWEep:DIRection?

Examples: [:SOURce]:SWEep:DIRection *sp* DOWN
Set :SWEep2 *(power level sweep) from a high level to a low level.*

[:SOURce]:SWEep:DIRection?
Requests the currently programmed sweep direction.

NOTE:

:SWEep1 (or :SWEep) signifies frequency sweep; :SWEep2 signifies power level sweep.

[:SOURce]

:SWEep<n> (1 ≤ **n** ≤ 2)

:DWELl

Parameters:	dwell time (in seconds)
Type:	<nv>
Range:	1 ms to 99 sec
Default:	1 ms

Description: Sets the dwell time for each step in a stepped frequency sweep or power level sweepto the value entered. See notes below.

Query Form: [:SOURce]:SWEep:DWELl?

Examples: [:SOURce]:SWEep:DWELl *sp* 100 ms

Set dwell time for each step in the sweep to 100 milliseconds.

[:SOURce]:SWEep:DWELl?

Requests the currently programmed value for sweep step dwell time.

NOTES:

The value entered for dwell time cannot be less than :TIME/:POINts.

When encountered, the command :SWEep:DWELl <arg> command sets :SWEep:DWELl:AUTO to off.

[:SOURce]

:SWEep<n> (1 ≤ **n** ≤ 2)

:DWELl

AUTO

Parameters:	ON | OFF | 1 | 0
Type:	<boolean>
Default:	ON
Description:	ON signifies that :DWELl ≈ :TIME/:POINts. See note below.
Query Form:	[:SOURce]:SWEep:DWELl:AUTO?
Examples:	[:SOURce]:SWEep:DWELl:AUTO *sp* ON *Set* :SWEep:DWELl *to its default value.* [:SOURce]:SWEep:DWELl:AUTO? *Requests the currently programmed* :SWEep:DWELl *default setting (on/off).*

NOTES:

:DWELl = (:TIME/:POINts – internal settling time)

:DWELl, :SPAN, :TIME, :STEP, and :POINts are all interrelated. Entering too large a value for :DWELl may invalidate the setting for :TIME.

The table on the following page shows the interaction between Dwell, Sweep Time, and Points.

Interaction between Dwell, Sweep Time, and Points

:SWEep:XXXX:AUTO switches		Interaction
:DWELl	:TIME	
OFF	OFF	No coupling between :SWEep:DWELl, :SWEep:TIME, and :SWEep:POINts.
OFF	ON	:SWEep:TIME is always set to the minimum value that is compatible with other settings, as follows: If :SWEep:GENeration STEPped, then :SWEep:TIME = (:DWELl · :POINts)
ON	OFF	When :SWEep:TIME or :SWEep:POINts are changed, :SWEep:DWELl is set to the larger of (:DWELl = :TIME/:POINts) or :DWELl Minimum.
ON	ON	:SWEep:TIME is always set to minimum. In stepped sweep mode, :SWEep:DWELl is adjusted to its minimum value, then :SWEep:TIME is computed. If :SWEep:GENerator STEPped, then :SWEep:DWELl = :DWELl Min (the larger of .001 or :TIME:LLIMit/:POINts) :SWEep:TIME = :DWELl Min · :POINts

[:SOURce]

:SWEep<n> (1 ≤ **n** ≤ 2)

:GENeration

Parameters:	STEPped
Type:	<char>
Default:	SWEep1 is STEPped; SWEep2 is STEPped
Description:	Selects stepped frequency and power level sweeps.
Query Form:	[:SOURce]:SWEep:GENeration?
Examples:	[:SOURce]:SWEep:GENeration *sp* STEPped *Set* SWEep[1] *for stepped frequency sweep.* [:SOURce]:SWEep:GENeration? *Requests the currently programmed frequency sweep mode.*

Sweep Mode Compatibility and Action Chart

Mode		Sweep		Frequency Action	Power Action
:FREQ	:POW	:SWE:GEN	:SWE2:GEN		
CW	FIX	x	x	CW, F1	Fixed, L0
SWCW	FIX	x	x	CW, F1, Ramp on	Fixed, L0
SWE	FIX	STEP	x	Stepped Sweep, F1-F2	Fixed, L0
SWE	SWE	STEP	x	Stepped Sweep F1-F2 :SWE:POINts :SWE:STEP	Step Power after each Freq Sweep L1-L2 :SWE:POINts :SWE:POW:STEP
SWE	SWE2	STEP	STEP	Stepped Sweep F1-F2 :SWE:POINts :SWE:STEP	Step Power after each Freq Sweep L1-L2 :SWE2:POINts :SWE2:POW:STEP
CW	SWE	STEP	x	CW, F0	Step Power Sweep L1-L2
SWCW	SWE	STEP	x	CW, F0, Ramp on	Step Power Sweep L1-L2
CW	SWE2	x	STEP	CW, F0	Step Power Sweep L1-L2
SWCW	SWE2	x	STEP	CW, F0, Ramp on	Step Power Sweep L1-L2
ALSW	FIX	x	x	Alternate Stepped Sweep F1-F2 F3-F4	Fixed L0 Trigger n L0 Trigger n+1
ALSW	ALSW	STEP	x	Alternate Stepped Sweep F1-F2 F3-F4	Alternate values L0 Trigger n L1 Trigger n+1

[:SOURce]

:SWEep<n> (1 ≤ **n** ≤ 2)

:POINts

Parameters:	number pf points | MIN | MAX
Type:	<nv>
Range:	2 to 10,001 (MAXimum) for Stepped Frequency sweeps
Default:	10,001 for SWEep1 The default value for SWEep2 depends on the power range of the particular 690XXA model (see notes).
Description:	Sets the number of points in each sweep in the stepped frequency sweep or power level sweep to the value entered. See notes below.
Query Form:	[:SOURce]:SWEep:POINts?
Examples:	[:SOURce]:SWEep:POINts *sp* 500 *Set the number of points for each sweep to 500.* [:SOURce]:SWEep:POINts? *Requests the currently programmed value for points/sweep.*

NOTES:r

The default values for SWEep2 points and time are set up to give the maximum number of points at the minimum step size for the particular 690XXA model. The default values are dependent on the instrument's available power range.

:POINts and :STEP are coupled values. Entering the value for one will cause the other to be recalculated, per formula below. Entering a new value for either parameter will not change :SPAN.

:POINts = (:SPAN/:STEP) + 1

An error will be generated if the :POINts value entered results in a step size (1) less than 1 kHz (or 0.1 Hz for models with Option 11) for a stepped frequency sweep or (2) less than 0.01 dB for a power level sweep.

[:SOURce]

:SWEep<n> (1 ≤ **n** ≤ 1)

[:FREQuency]

:STEP

Parameters:	frequency (in Hz) | MIN | MAX
Type:	<nv>
Range:	(see notes below)
Default:	1,999,000 Hz

Description: Sets the step size for each step in SWEep[1] (frequency) stepped sweep mode to the value entered. See notes below.

Query Form: [:SOURce]:SWEep:FREQuency:STEP?

Examples: [:SOURce]:SWEep:FREQuency:STEP *sp* 5 GHz

Set the step size for each step in the stepped frequency sweep to 5 GHz.

[:SOURce]:SWEep:FREQuency:STEP?

Requests the currently programmed step size for each step in the stepped frequency sweep.

NOTES:

:SWEep2 is not valid for this command; only :SWEep[1] will be recognized.

:POINts and :STEP are coupled values. Entering the value for one will cause the other to be recalculated; see notes under :SWEep:POINts command.

The maximum frequency sweep step size is equal to maximum frequency span for the particular 690XXA model/(minimum points – 1). For the model 69047A with option 11 (0.1 Hz Frequency Resolution), the minimum step size is 0.1 Hz and the maximum step size is 19,990,000,000 Hz.

Refer to notes under :FREQuency:CW | :FIXed command for the frequency span values for each 690XXA model.

An error will be generated if the :STEP size value entered exceeds the value for :SPAN.

[:SOURce]

:SWEep<n> (1 ≤ **n** ≤ 2)

:POWer

:STEP

Parameters:	power level (in dB) | MIN | MAX
Type:	<nv>
Range:	MINimum = 0.01 dB MAXimum is model dependent (see notes below)
Default:	0.01 dB for SWEep1 0.02 dB for SWEep2
Description:	Sets the step size for each step in a power level sweep to the value entered. See notes below.
Query Form:	[:SOURce]:SWEep<n>:POWer:STEP?
Examples:	[:SOURce]:SWEep2:POWer:STEP *sp* 10 dB *Set the step size for each step in the* SWEep2 *power level sweep to 10 dB.* [:SOURce]:SWEep2:POWer:STEP? *Requests the currently programmed step size for each step in the* SWEep2 *power level sweep.*

NOTES:

:STEP and :POINts are coupled values. Entering the value for one will cause the other to be recalculated; see notes under :SWEep:POINts command.

For standard 690XXA models, a maximum step size up to 28 dB may be used (up to 131 dB for models with option 2 step attenuator). For 690XXA models with option 15A, a maximum step size up to 22 dB can be used (up to 125 dB with option 2 step attenuator). However, the step size, in conjunction with the initial RF power output setting, must not produce a programmed power level below the minimum leveled output power for the particular 690XXA model. Refer to Appendix B — Performance Specifications — in the Series 690XXA Synthesized CW Generator Operation Manual.

[:SOURce]

:SWEep<n> (1 ≤ **n** ≤ 2)

:TIME

Parameters:	sweep time (in seconds) | MIN | MAX
Type:	<nv>
Range:	30 ms to 99 sec
Default:	30 ms for SWEep1 The default value for SWEep2 depends on the power range of the particular 690XXA model (see notes).
Description:	Sets the sweep time for the associated SWEep[1] or SWEep2 sweep to the value entered. See notes below.
Query Form:	[:SOURce]:SWEep<n>:TIME?
Examples:	[:SOURce]:SWEep1:TIME *sp* 100 ms *Set the sweep time for* SWEep1 *sweep to 100 ms.* [:SOURce]:SWEep2:TIME? *Requests the currently programmed time for* SWEep2 *sweep.*

NOTES:

The default values for SWEep2 points and time are set up to give the maximum number of points at the minimum step size for the particular 690XXA model. The default values are dependent on the instrument's available power range.

When :SWEep<n>:TIME <arg> is implemented, :SWEep<n>:TIME :AUTO is set to OFF.

:TIME, :DWELl, :SPAN, :STEP, and :POINts are all interrelated. Entering too large a value for any of the four other parameters may invalidate the entered value for :TIME.

[:SOURce]

:SWEep<n> (1 ≤ **n** ≤ 2)

:TIME

:LLIMit

Parameters:	sweep time (in seconds) | MIN | MAX
Type:	<nv>
Range:	2 ms to 98.998 sec
Default:	2 ms

Description: Sets the lower limit for :SWEep<n>:TIME to the value entered. No sweep time will be shorter than this value. See notes below.

Query Form: [:SOURce]:SWEep<n>:TIME:LLIMit?

Examples: [:SOURce]:SWEep1:TIME:LLIMit *sp* 80 ms
Set the lower limit for SWEep1 *sweep time to 80 ms.*

[:SOURce]:SWEep2:TIME:LLIMit?
Requests the currently programmed lower limit for SWEep2 *sweep time.*

NOTES:

When :SWEep<n>:TIME:AUTO is set ON, the internally computed sweep time will not be smaller than the sweep duration set with the :SWEep<n>:Time:LLIMit command

:SWEep2:TIME:LLIMit will be set to the same value as :SWEep:TIME :LLImit and vice versa.

[:SOURce]

:SWEep<n> (1 ≤ **n** ≤ 2)

:TIME

:AUTO

Parameters:	ON | OFF | 1 | 0
Type:	<boolean>
Default:	ON

Description: ON specifies that the sweep time for the associated sweep (SWEep[1] or SWEep2) is to be calculated internally and is dependent on the sweep SPAN value. See note below.

Query Form: [:SOURce]:SWEep:TIME:AUTO?

Examples: [:SOURce]:SWEep:TIME:AUTO *sp* ON

Specifies that the SWEep[1] *sweep time is to be calculated internally.*

[:SOURce]:SWEep:TIME:AUTO?

Requests the currently programmed mode for sweep time determination.

NOTES:

When the 690XXA is powered up, or when ∗RST command is issued, :SWEep:TIME:AUTO is set to ON.

3-11 STATUS SUBSYSTEM

The :STATus subsystem controls the SCPI-defined status-reporting stuctures of the 690XXA. The subsystem commands and parameters are described below.

KEYWORD	PARAMETER FORM	NOTES
:STATus		
:OPERation		
[:EVENt]?		
:CONDition?		
:ENABle	**<numeric_value>**	Default: 32767
:PTRansition	**<numeric_value>**	Default: 32767
:NTRansition	**<numeric_value>**	Default: 0
:PRESet		
:QUEStionable		
[:EVENt]?		
:CONDition?		
:ENABle	**<numeric_value>**	Default: 32767
:PTRansition	**<numeric_value>**	Default: 32767
:NTRransition	**<numeric_value>**	Default: 0
:QUEue		
[:NEXT]?		

:STATus
:OPERation
[:EVENt]?

Parameters Returned: event register contents

Type: <NR1>

Description: Returns the the contents of the 690XXA Operational Event register. *When executed, this command clears the Operational Event register.*

Example: :STATus:OPERation:EVENt?

Requests that the contents of the the Operational Event register be returned.

:STATus

:OPERation

:CONDition?

Parameters Returned:	condition register contents
Type:	<NR1>
Description:	Returns the the contents of the 690XXA Operational Condition register. When executed, this command does *not* clear the Operational Condition register.
Example:	:STATus:OPERation:CONDition? *Requests that the contents of the the Operational Condition register be returned.*

:STATus
:OPERation
:ENABle

Parameters:	mask
Type:	<NRf>
Range:	0 – 32767
Default:	32767 (All 1's)

Description: Sets the bits of the Operational Enable register associated with the Operational Event register to the binary weighted integer value specified by the mask parameter.

Query Form :STATus:OPERation:ENABle?

Example: :STATus:OPERation:ENABle *sp* 8

Set the 690XXA Operational Enable register to a value of 8. (This will unmask the Sweeping *status bit).*

Example: :STATus:OPERation:ENABle?

Requests the current value of the 690XXA Operational Enable register.

:STATus

:OPERation

:PTRansition

Parameters:	mask
Type:	<NRf>
Range:	0 – 32767
Default:	32767 (All 1's)

Description: Sets the bits of the positive transition filter for the Operational Condition register to the binary weighted integer value specified by the mask parameter.

Query Form :STATus:OPERation:PTRansition?

Example: :STATus:OPERation:PTRansition *sp* 512

Sets the 690XXA Positive Transition Filter for the Operational Condition register to a value of 512. When the filter detects a False to True transition in bit 9 of the Operational Condition register, indicating that 690XXA self-test is in progress, bit 9 of the Operational Event register will be set to "1".

Example: :STATus:OPERation:PTRansition?

Requests the current value of the 690XXA Positive Transition Filter for the Operational Condition register.

:STATus

:OPERation

:NTRansition

Parameters:	mask
Type:	<NRf>
Range:	0 – 32767
Default:	0 (All 0's)

Description: Sets the bits of the negative transition filter for the Operational Condition register to the binary weighted integer value specified by the mask parameter.

Query Form :STATus:OPERation:NTRansition?

Example: :STATus:OPERation:NTRansition *sp* 8

Sets the 690XXA Negative Transition Filter for the Operational Condition register to a value of 8. When the filter detects a True to False transition in bit 3 of the Operational Condition register, indicating that the 690XXA is finished sweeping, bit 3 of the Operational Event register will be set to "1".

Example: :STATus:OPERation:PTRansition?

Requests the current value of the 690XXA Negative Transition Filter for the Operational Condition register.

:STATus

:PRESet

Parameters:	None
Description:	This command is an event that configures the SCPI and device-dependent status reporting structures so that device-dependent events are summarized and reported. This command performs the following functions: Sets the Operational Enable register to all 0's. Sets the Operational Positive Transition Filter to all 1's. Sets the Operational Negative Transition Filter to all 0's. Sets the Questionable Enable register to all 0's. Sets the Questionable Positive Transition Filter to all 1's. Sets the Questionable Negative Transition Filter to all 0's.
Query Form	None
Example:	:STATus:PRESet *Configure the status reporting structures for device-dependent event reporting.*

:STATus

:QUEStionable

[:EVENt]?

Parameters Returned:	event register contents
Type:	<NR1>
Description:	Returns the the contents of the 690XXA Questionable Event register. *When executed, this command clears the Questionable Event register.*
Example:	:STATus:QUEStionable:EVENt? *Requests that the contents of the the Questionable Event register be returned.*

:STATus

:QUEStionable

:CONDition?

Parameters Returned:	condition register contents
Type:	<NR1>
Description:	Returns the the contents of the 690XXA Questionable Condition register. When executed, this command does *not* clear the Questionable Condition register.
Example:	:STATus:QUEStionable:CONDition? *Requests that the contents of the the Questionable Condition register be returned.*

:STATus

:QUEStionable

:ENABle

Parameters:	mask
Type:	<NRf>
Range:	0 – 32767
Default:	32767 (All 1's)

Description: Sets the bits of the Questionable Enable register associated with the Questionable Event register to the binary weighted integer value specified by the mask parameter.

Query Form :STATus:QUEStionable:ENABle?

Example: :STATus:QUEStionable:ENABle *sp* 32

Set the 690XXA Questionable Enable register to a value of 32. (This will unmask the Lock Error or RF Unlocked *status bit).*

Example: :STATus:QUEStionable:ENABle?

Requests the current value of the 690XXA Questionable Enable register.

:STATus

:QUEStionable

:PTRansition

Parameters:	mask
Type:	<NRf>
Range:	0 – 32767
Default:	32767 (All 1's)

Description: Sets the bits of the positive transition filter for the Questionable Condition register to the binary weighted integer value specified by the mask parameter.

Query Form `:STATus:QUEStionable:PTRansition?`

Example: `:STATus:QUEStionable:PTRansition sp 512`

Sets the 690XXA Positive Transition Filter for the Questionable Condition register to a value of 512. When the filter detects a False to True transition in bit 9 of the Questionable Condition register, indicating that 690XXA self-test failed, bit 9 of the Questionable Event register will be set to "1".

Example: `:STATus:QUEStionable:PTRansition?`

Requests the current value of the 690XXA Positive Transition Filter for the Questionable Condition register.

:STATus

:QUEStionable

:NTRansition

Parameters:	mask
Type:	<NRf>
Range:	0 – 32767
Default:	0 (All 0's)

Description: Sets the bits of the negative transition filter for the Questionable Condition register to the binary weighted integer value specified by the mask parameter.

Query Form :STATus:QUEStionable:NTRansition?

Example: :STATus:QUEStionable:NTRansition *sp* 8

Sets the 690XXA Negative Transition Filter for the Questionable Condition register to a value of 8. When the filter detects a True to False transition in bit 3 of the Questionable Condition register, indicating that the RF is no longer unleveled, bit 3 of the Questionable Event register will be set to "1".

Example: :STATus:QUEStionable:PTRansition?

Requests the current value of the 690XXA Negative Transition Filter for the Questionable Condition register.

:STATus

:QUEue

[:NEXT]?

Parameters Returned:	error code, error message string
Type:	<NR1><string>
Description:	Returns and deletes the oldest uncleared error code and error description from the error queue. Optional device dependent information about the error event may also be included. See notes below. The error codes and error description information for the 690XXA are listed in Chapter 4.
Example:	:STATus:QUEue:NEXT? *Requests that the oldest error code/error description be returned from the error queue.*

NOTES:

As errors are detected, they are placed in the error queue. The queue is first in, first out and can hold a maximum of 10 messages.

If the error queue is not empty, bit 2 of the Summary Status Byte 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 byte is 0.

If the error queue overflows, the last error message in the queue is replaced by the error –350, “Queue overflow”

3-12 SYSTEM SUBSYSTEM

The :SYSTem subsystem commands are used to implement functions that are not related to 690XXA performance. These include error query, interface language selection, system preset, and version query. The subsystem commands and parameters are described below.

KEYWORD	PARAMETER FORM	NOTES
:SYSTem		
:ERRor?		
:LANGuage	<string>	
:PRESet		
:VERSion?		

:SYSTem

:ERRor?

Parameters Returned:	error code, error message string
Type:	<NR1><string>
Description:	Returns and deletes the oldest uncleared error code and error description from the error queue. Optional device dependent information about the error event may also be included. See notes below. The error codes and error description information for the 690XXA are listed in Chapter 4.
Example:	:SYSTem:ERRor? *Requests that the oldest error code/error description be returned from the error queue.*

NOTES:

As errors are detected, they are placed in the error queue. The queue is first in, first out and can hold a maximum of 10 messages. If the error queue is not empty, bit 2 of the Summary Status Byte 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 byte is 0.

If the error queue overflows, the last error message in the queue is replaced by the error –350, “Queue overflow”

:SYSTem

:LANGuage

Parameters:	"SCPI" | "NATIVE" | "TMSL"
Type:	<string>
Default:	Dependent upon the selection made at the 690XXA front panel Configure GPIB menu.
Description:	Selects the instrument's external interface language. "TMSL" is an alias for "SCPI". Entering either will return "SCPI" when queried. The double quotes are required and will be returned with the query reply. "NATIVE" changes the instrument's external interface language to the 690XXA GPIB mode. Any commands issued within 1 second of the change may be garbled or lost.
Query Form:	:SYSTem:LANGuage?
Examples:	:SYSTem:LANGuage *sp* "NATIVE" *Selects "NATIVE" (690XXA GPIB mode) as the external interface language.* :SYSTem:LANGuage? *Requests the current selection for 690XXA external interface language.*

NOTES:

When changing from NATIVE to SCPI interface language, use the command SYST:LANG "SCPI" . Do ***not*** use the long form of the command and do ***not*** use a leading colon (:) with the command. The command :SYSTem:LANGuage "SCPI" results in a syntax error.

:SYSTem

:PRESet

Parameters:	None
Description:	This command is synonomous with *RST and is included for programming compatibility with other instruments.
Query Form:	None
Example:	:SYSTem:PRESet *Sets all user programmable 690XXA parameters to their default values.*

:SYSTem

:VERSion?

Parameters Returned:	version number (see note)
Type:	<NR2>
Description:	Returns the SCPI version number that the instrument software complies with.
Example:	:SYSTem:VERSion? *Requests the SCPI version number that the instrument software complies with.*

NOTE:

The query response shall have the form *YYYY.V* where the *Ys* represent the year-version (i.e. 1993) and the *V* represents the approved revision number for that year.

3-13 TRIGGER SUBSYSTEM

The :TRIGger subsystem commands are used to control the sweep triggering functions of the 690XXA. The subsystem commands and parameters are described below. The :TRIGger command, along with the :ABORt and :INITiate commands, comprise the Trigger Group of commands. (Refer to Chapter 2, paragraph 2-6 for a description of trigger system programming.)

KEYWORD	PARAMETER FORM	NOTES
:TRIGGer		
[:SEQuence | :STARt]		
[:IMMediate]		
:SOURce	**BUS | IMMediate | HOLD**	Default: BUS

:TRIGger [:SEQuence | :STARt] [:IMMediate]

Parameters	None
Description:	Causes the previously selected sweep to be triggered immediately if the trigger system is in armed state due to a previous :INITiate:IMMediate command.
Query Form	None
Example:	:TRIGger:SEQuence:IMMediate *Trigger selected sweep immediately if 690XXA is in trigger armed state.*
Associated commands:	:ABORt and :INITiate

NOTES:

Keyword :SEQuence is equivalent to keywords :SEQuence1 or :START. These keywords may be used interchangeably.

:TRIGger

[:SEQuence | :STARt]

:SOURce

Parameters	BUS | IMMediate | HOLD
Type:	<char>
Default:	BUS
Description:	Selects the trigger source for the previously selected sweep. The source selections are: BUS – The source is the group execute trigger command from the GPIB. The trigger will occur when either a <GET> or ∗TRG command is received. IMMediate – The trigger signal is always true. HOLD – Do not trigger on any sweep.
Query Form	:TRIGger:SEQuence:SOURce?
Example:	:TRIGger:SEQuence:SOURce *sp* IMMediate *Select the trigger signal to be true.*
Example:	:TRIGger:SEQuence:SOURce? *Requests the currently programmed sweep trigger source.*

NOTES:

Only one trigger source can be specified at a time, and all others will be ignored.

Sending :TRIGger:SOURce IMMediate;:INITiate:CONTinuous ON places the trigger system in auto trigger mode. This causes continuous generation of the selected sweep.

3-14 :TSWeep COMMAND

The :TSWeep command is a convenience command. It is equivalent to sending :ABORt;:INITiate[:IMMediate].

3-15 UNIT SUBSYSTEM

The :UNIT subsystem commands set the default units for the frequency and time parameters that are used with all 690XXA SCPI commands described in this manual. The units selected apply to the designated command parameters for both command and response. The subsystem commands and parameters are described below.

KEYWORD	PARAMETER FORM	NOTES
:UNIT		
:FREQuency	**HZ | KHZ | MHZ | GHZ**	Default: HZ
:TIME	**S | MS | US | NS**	Default: S

:UNIT

:FREQuency

Parameters	HZ | KHZ | MHZ | GHZ
Type:	<char>
Default:	HZ
Description:	Selects the global default frequency unit for all frequency related parameters used with all 690XXA SCPI commands.
Query Form	:UNIT:FREQuency?
Example:	:UNIT:FREQuency *sp* GHZ *Select* GHz *as default frequency unit for all 690XXA frequency related command parameters.*
Example:	:UNIT:FREQuency? *Requests the currently selected frequency unit.*

:UNIT

:TIME

Parameters	S | MS | US | NS
Type:	<char>
Default:	S

Description: Selects the global default for all time related parameters used with all 690XXA SCPI commands.

S	= second
MS	= millisecond
US	= microsecond
NS	= nanosecond

Query Form :UNIT:TIME?

Example: :UNIT:TIME *sp* MS

Select millisecond as default time unit for all 690XXA time related command parameters.

Example: :UNIT:TIME?

Requests the currently selected time unit.

Chapter 4
Error Messages

Table of Contents

Chapter 4
Error Messages

4-1 INTRODUCTION

This chapter lists and describes each of the error messages related to 690XXA CW 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.

4-2 ERROR QUERY

The :SYSTem:ERRor? query command 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 (see paragraph 4.3).

The :SYSTem:ERRor? query command is required of all SCPI implementations. The :STATus:QUEue[:NEXT]? query command when implemented is an alias to :SYSTem:ERRor?.

The instrument responds to the :SYSTem:ERRor? query command 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 which will allow the user to determine the exact error and context. For example:

```
–241,"Hardware missing;Attenuator"
```

The maximum string length of <error description> plus <device-dependent information> is 255 characters. The <error description> shall be sent exactly as indicated in this chapter including case.

4-3 ERROR QUEUE

As errors are detected, error messages are placed in a queue. This queue is first in, first out and can hold a maximum of 10 messages. If the queue overflows, the last error message in the queue is replaced with the error message

–350, “Queue overflow”

Any time the queue overflows, the least recent error messages remain in the queue, and the most recent error message is discarded. Reading an error message from the head of the queue removes that error message from the queue, and opens a position at the tail of the queue for a new error message, if one is subsequently detected.

When all error messages have been read from the queue, further error queries shall return

0, “No error”

The error queue shall be 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.

4-4 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.

4-5 NO ERROR

This message indicates that the device has no errors.

Error Code	**Error Description [description/explanation/examples]**
0	"No error" The queue is completely empty. Every error in the queue has been read or the queue was purposely cleared by power-on, *CLS, etc.

4-6 COMMAND ERRORS

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 which 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	**Error Description [description/explanation/examples]**
–102	"Syntax error" An unrecognized command or data type was encountered; for example, a string was received when the device does not accept strings.
–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.
–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 nondecimal 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 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.
–150	"String data error" This error, as well as errors –151 through –159, are generated when parsing a string data element. This particular error message should be used if the device cannot detect a more specific error.

–151	"Invalid string data" A string data element was expected, but was invalid for some reasons (see IEEE 488.2, 7.7.5.2); for example, an END message was received before the terminal quote character.
–158	"String data not allowed" A legal string data element was encountered but was not allowed by the device at this point in parsing.
–160	"Block data error" This error, as well as errors –161 through –169, are generated when parsing a block data element. This particular error message should be used if the device cannot detect a more specific error.
–161	"Invalid block data" A block data element was expected, but invalid for some reason (see IEEE 488.2, 7.7.6.2); for example, an END message was received before the length was satisfied.
–168	"Block data not allowed" A legal block data element was encountered but was not allowed by the device at this point in parsing.
–170	"Expression error" This error, as well as errors –171 through –179, are generated when parsing an expression data element. This particular error message should be used if the device cannot detect a more specific error.
–171	"Invalid expression" The expression data element was invalid (see IEEE 488.2, 7.7.7.2); for example, unmatched parentheses or an illegal character.
–178	"Expression data not allowed" A legal expression data element was encountered but was not allowed by the device at this point in parsing.
–180	"Macro error" This error, as well as errors –181 through –189, are generated when defining a macro or executing a macro. This particular error message should be used if the device cannot detect a more specific error.

–181	"Invalid outside macro definition" Indicates that a macro parameter placeholder ($<number) was encountered outside of a macro definition.
–183	"Invalid inside macro definition" Indicates that the program message unit sequence, sent with a ∗DDT or ∗DMC command, is syntactically invalid (see IEEE 488.2, 10.7.6.3).
–184	"Macro parameter error" Indicates that a command inside the macro definition had the wrong number or type of parameters.

4-7 EXECUTION ERRORS

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 shall be reported by the device 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	**Error Description [description/explanation/examples]**
–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.
–201	"Invalid while in local" Indicates that a command is not executable while the device is in local due to a hard local control (see IEEE 488.2, 5.6.1.5); for example, a device with a rotary switch receives a message which would change the switches state, but the device is in local so the message cannot be executed.
–202	"Settings lost due to rtl" Indicates that a setting associated with a hard local control (see IEEE 488.2, 5.6.1.5) was lost when the device was changed to LOCS from REMS or to LWLS from RWLS.
–210	"Trigger error" A trigger error occurred in the sweep generator.
–211	"Trigger Ignored" Indicates that a GET, ∗TRG, or triggering signal was received and recognized by the device but was ig-

	nored because of device timing considerations; for example, the device was not ready to respond. Note: a DTO device always ignores GET and treats *TRG as a command error.
–212	"Arm ignored" Indicates that an arming signal was received and recognized by the device but was ignored.
–213	"Init ignored" Indicates that a request for measurement initiation was ignored as another measurement was already in progress.
–214	"Trigger deadlock" Indicates that the trigger source for the initiation of a measurement is set to GET and subsequent measurement query is received. The measurement cannot be started until a GET is received, but the GET would cause an INTERRUPTED error.
–215	"Arm deadlock" Indicates that the arm source for the initiation of a measurement is set to GET and subsequent measurement query is received. The measurement cannot be started until a GET is received, but the GET would cause an INTERRUPTED error.
–220	"Parameter error" Indicates that a program data element related error occurred. This error message should be used when the device cannot detect the more specific errors described for errors –221 to –229.
–221	"Settings conflict" Indicates that a legal program data element was parsed but could not be executed due to the current device state (see IEEE 488.2, 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 IEEE 488.2, 11.5.1.1.5).
–223	"Too much data" Indicates that a legal program data element of block, expression, or string type was received that contained more data than the device could handle due to memory or related device-specific require-

	ments.
–224	"Illegal parameter value" Used where exact value, from a list of possibles, was expected.
–230	"Data corrupt or stale" Possibly invalid data; new reading started but not completed since last access.
–231	"Data questionable" Indicates that measurement accuracy is suspect.
–240	"Hardware error" Indicates that a legal program command or query could not be executed because of a hardware problem in the device. Definition of what constitutes a hardware problem is completely device-specific. This error message should be used when the device cannot detect the more specific errors described for errors –241 through –249.
–241	"Hardware missing" Indicates that a legal program command or query could not be executed because of missing device hardware; for example, an option was not installed. Definition of what constitutes missing hardware is completely device specific.
–250	"Mass storage error" Indicates that a mass storage error occurred. This error message should be used when the device cannot detect the more specific errors described for errors –251 through –259.
–251	"Missing mass storage" Indicates that a legal program command or query could not be executed because of missing mass storage; for example, an option that was not installed. Definition of what constitutes missing mass storage is device-specific.
–252	"Missing media" Indicates that a legal program command or query could not be executed because of missing media; for example, no disk. Definition of what constitutes missing media is device-specific.
–253	"Corrupt media" Indicates that a legal program command or query

	could not be executed because of corrupt media; for example, bad disk or wrong format. Definition of what constitutes corrupt media is device-specific.
–254	"Media full" Indicates that a legal program command or query could not be executed because the media was full; for example, there is no room on the disk. The definition of what constitutes a full media is device-specific.
–255	"Directory full" Indicates that a legal program command or query could not be executed because the media directory was full. The definition of what constitutes a full media directory is device-specific.
–256	"File name not found" Indicates that a legal program command or query could not be executed because the file name on the device media was not found; for example, an attempt was made to read or copy a nonexistent file. The definition of what constitutes a file not being found is device-specific.
–257	"File name error" Indicates that a legal program command or query could not be executed because the file name on the device media was in error; for example, an attempt was made to copy to a duplicate file name. The definition of what constitutes a file name error is device-specific.
–258	"Media protected" Indicates that a legal program command or query could not be executed because the media was protected; for example, the write-protect tab on a disk was present. The definition of what constitutes protected media is device-specific.
–260	"Expression error" Indicates that an expression program data element related error occurred. This error message should be used when the device cannot detect the more specific errors described for errors –261 through –269.
–261	"Math error in expression" Indicates that a syntactically legal expression program data element could not be executed due to a math error; for example, a divide-by-zero was at-

	tempted. The definition of math error is device -specific.
–270	"Macro error" Indicates that a macro-related execution error occurred. This error message should be used when the device cannot detect the more specific errors described for errors –271 through –279.
–271	"Macro syntax error" Indicates that a syntactically legal macro program data sequence, according to IEEE 488.2, 10.7.2, could not be executed due to a syntax error within the macro definition (see IEEE 488.2, 10.7.6.3).
–272	"Macro execution error" Indicates that a syntactically legal macro program data sequence could not be executed due to a some error in the macro definition (see IEEE 488.2, 10.7.6.3).
–273	"Illegal macro label" Indicates that the macro label defined in the ∗DMC command was a legal string syntax, but could not be accepted by the device (see IEEE 488.2, 10.7.3 and 10.7.6.2); for example, the label was too long, the same as a common command header, or contained invalid header syntax.
–274	"Macro parameter error" Indicates that the macro definition improperly used a macro parameter placeholder (see IEEE 488.2, 10.7.3).
–275	"Macro definition too long" Indicates that a syntactically legal macro program data sequence could not be executed because the string or block contents were too long for the device to handle (see IEEE 488.2, 10.7.6.1).
–276	"Macro recursion error" Indicates that a syntactically legal macro program data sequence could not be executed because the device found it to be recursive (see IEEE 488.2, 10.7.6.6).
–277	"Macro redefinition not allowed" Indicates that a syntactically legal macro label in the ∗DMC command could not be executed because the macro label was already defined (see IEEE

	488.2, 10.7.6.4).
–278	"Macro header not found" Indicates that a syntactically legal macro label in the ∗GMC? query could not be executed because the header was not previously defined.
–280	"Program error" Indicates that a downloaded program-related execution error occurred. This error message should be used when the device cannot detect the more specific errors described for errors –281 through –289.
–281	"Cannot create program" Indicates that an attempt to create a program was unsuccessful. A reason for the failure might include not enough memory.
–282	"Illegal program name" The name used to reference a program was invalid; for example, redefining an existing program, deleting a nonexistent program, or in general, referencing a nonexistent program.
–283	"Illegal variable name" An attempt was made to reference a nonexistent variable in a program.
–284	"Program currently running" Certain operations dealing with programs may be illegal while the program is running; for example, deleting a running program might not be possible.
–285	"Program syntax error" Indicates that a syntax error appears in a downloaded program. The syntax used when parsing the downloaded program is device-specific.
–286	"Program runtime error"

4-8 DEVICE-SPECIFIC ERRORS

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; some device operations did not properly complete, possibly due to an abnormal hardware or firmware condition. These codes are also used for self-test response errors. The occurrence of any error in this class should cause the device-specific error bit (bit 3) in the standard event status register to be set. The meaning of positive error codes is device-dependent and may be enumerated or bit mapped; the <error message> string for positive error codes is not defined by SCPI and available to the device designer. Note that the string is not optional; if the designer does not wish to implement a string for a particular error, the null string should be sent (for example, 42,""). The occurrence of any error in this class should cause the device-specific error bit (bit 3) in the standard event status register to be set. Events that generate device-specific errors shall not generate command errors, execution errors, or query errors; see the other error definitions in this chapter.

Error Code	**Error Description [description/explanation/examples]**
–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 IEEE 488.2, 11.5.1.1.6 has occurred.
–310	"System error" Indicates that some error, termed “system error” by the device, has occurred. This code is device-dependent.
–311	"Memory error" Indicates that an error was detected in the device's memory. The scope of this error is device-dependent.
–312	"PUD memory lost" Indicates that the protected user data saved by the ∗PUD command has been lost.
–313	"Calibration memory lost" Indicates that nonvolatile calibration data used by the ∗CAL? command has been lost.
–314	"Save/recall memory lost" Indicates that the nonvolatile data saved by the ∗SAV? command 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.
–330	"Self-test failed"
–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.

4-9 QUERY ERRORS

An <error code> in the range [–499,–400] indicates that the output queue control of the instrument has detected a problem with the message exchange protocol described in IEEE 488.2, chapter 6. The occurrence of any error in this class should cause the query error bit (bit 2) in the standard event status register to be set. These errors correspond to message exchange protocol errors described in IEEE 488.2, section 6.5. One of the following is true:

- ❑ An attempt is being made to read data from the output queue when no output is either present or pending.
- ❑ Data in the output queue has been lost.

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

Error Code	**Error Description [description/explanation/examples]**
–400	"Query error" This is the generic query error for devices that cannot detect more specific errors. This code indicates only that a Query Error as defined in IEEE 488.2, 11.5.1.1.7 and 6.3 has occurred.
–410	"Query INTERRUPTED" Indicates that a condition causing an INTERRUPTED Query error occurred (see IEEE 488.2, 6.3.2.3); for example, a query followed by a DAB or GET before a response was completely sent.
–420	"Query UNTERMINATED" Indicates that a condition causing as UNTERMINATED Query error occurred (see IEEE 488.2, 6.3.2.2); for example, the device was addressed to talk and an incomplete program message was received.
–430	"Query DEADLOCKED" Indicates that a condition causing a DEADLOCKED Query error occurred (see IEEE 488.2, 6.3.1.7); for example, both input and output buffer are full and the device cannot continue.
–440	"Query UNTERMINATED after indefinite response" Indicates that a query was received in the same program message after a query requesting an indefinite response was executed (see IEEE 488.2, 6.5.7.5.7).

4-10 PARSER ERRORS

An <error code> in the range [201, 212] is generated by the instrument's parser in response to the error condition described.

Error Code	**Error Description [description/explanation/examples]**
201	"Query only" Indicates the command is a query command only.
202	"No query allowed" Indicates that a query form of the command is not available.
203	"Parameter(s) not expected" Indicates that the command does not take any parameter(s).
204	"Constant not allowed in STATUS subsystem"
207	"Enumeric value not in union" Indicates an illegal parameter was sent with the command.
208	"Illegal number of parameters" Indicates an illegal number of parameters was sent with the command.
210	"Run out of memory handle" Indicates an internal parser problem.
211	"Unit not matched" Indicates the parameter unit does not match any defined unit for this parameter.
212	"Unit not required" Indicates no unit is required with this parameter.

4-11 SELF-TEST ERRORS

An <error code> in the range [100, 199] indicates that a failure has occurred during instrument self-test. The error messages are placed in the error queue in the order they occur.

Error Code	Error Description [description/explanation/examples]
100	"Failed: DVM ground offset" Indicates self-test failed because of a calibration-related problem.
101	"Failed: DVM +10V ref." Indicates self-test failed because of either a calibration-related problem or a defective +10 Volt reference.
102	"Failed: DVM –10V ref." Indicates self-test failed because of either a calibration-related problem or a defective –10 Volt reference.
105	"Failed: Power Supply Voltages(s) out of regulation" Indicates one or more of the voltages from the power supply is out of regulation.
106	"Not locked: Power supply" Indicates the power supply is not phase-locked to the 400 kHz reference frequency.
107	"Failed; Sweep time circuitry" Indicates the sweep timing is out of tolerance or has failed.
108	"Not ready: Crystal oven cold" Indicates the 100 MHz crystal oven or the Option 16 high-stability 10 MHz crystal oscillator has not reached operating temperature.
109	"Not locked: Ext 10MHZ" Indicates the reference loop is not phase-locked to the external 10 MHz reference.
110	"Not locked: High Stability crystal" Indicates the reference loop is not phase-locked to the high stability 10 MHz crystal oscillator.
111	"Not locked: Fine Loop" Indicates one or more of the oscillators within the fine loop is not phase-locked.

112	"Not locked: Coarse Loop" Indicates the coarse loop oscillator is not phase-locked.
113	"Not locked: YIG Loop" Indicates the YIG loop is not phase-locked.
114	"Not locked: Down Converter" Indicates the local oscillator in the down converter assembly is not phase-locked.
115	"Failed: Not Locked indicator" Indicates failure of the not phase-locked indicator circuit.
116	"Failed: FM loop gain circuit" Indicates the FM loop has failed or the loop gain is out of tolerance.
117	"Failed: Linearizer circuit" Indicates failure of the Linearizer circuit on the A12 PCB.
118	"Failed: Marker Switch Point circuit" Indicates failure of the Marker Switch Point circuit on the A12 PCB.
119	"Failed: Center Frequency circuit" Indicates failure of the Center Frequency circuit on the A12 PCB.
120	"Failed: Delta-F Ramp circuit" Indicates failure of the ΔF Ramp circuit on the A12 PCB.
121	"Failed: Unleveled indicator" Indicates failure of the not leveled detector circuit on the A10 PCB.
122	"Failed: Level reference" Indicates failure of the level reference circuit on the A10 PCB.
123	"Failed: Detector log amp" Indicates failure of the level detector log amplifier circuit on the A10 PCB.
124	"Unleveled and not locked: Full band" Indicates failure of both YIG-tuned oscillators.

125	"Unleveled and not locked: 8.4-20 GHz range" Indicates failure of the 8.4 to 20 GHz YIG-tuned oscillator.
126	"Unleveled and not locked: 2-8.4 GHz range" Indicates failure of the 2 to 8.4 GHz YIG-tuned oscillator.
127	"Failed: A10 Detector input circuit" Indicates failure of the level detector input circuit on the A10 PCB.
128	"Failed: 0.01-2 GHz range unleveled" Indicates failure of the Down Converter leveling circuitry.
129	"Failed: Switched filter or level detector" Indicates failure of either the switched filter or level detector circuitry.
130	"Failed: 2-3.3 GHz Switched filter section or level detector" Indicates failure of either the 2 to 3.3 GHz switched filter path or the level detector circuitry.
131	"Failed: 3.3-5.5 GHz Switched filter section or level detector" Indicates failure of either the 3.3 to 5.5 GHz switched filter path or the level detector circuitry.
132	"Failed: 5.5-8.4 GHz Switched filter section or level detector" Indicates failure of either the 5.5 to 8.4 GHz switched filter path or the level detector circuitry.
133	"Failed: 8.4-13.25 GHz Switched filter section or level detector" Indicates failure of either the 8.4 to 13.25 GHz switched filter path or the level detector circuitry.
134	"Failed: 13.25-20 GHz Switched filter section or level detector" Indicates failure of either the 13.25 to 20 GHz switched filter path or the level detector circuitry.
135	"Failed: A9 modulator or driver" Indicates failure of either the modulator in the switched filter assembly or the modulator driver circuitry on the A9 PCB.

138	"Failed: Freq extension unit or driver" Indicates failure of the frequency extension unit (FEU) or FEU driver circuitry on the A9 PCB.
139	"Failed: 33-40 GHz section of freq extension unit" Indicates failure of the 33 to 40 GHz section of the FEU.
140	"Failed: 26.5-33 GHz section of freq extension unit" Indicates failure of the 26.5 to 33 GHz section of the FEU.
141	"Failed: 20-26.5 GHz section of freq extension unit" Indicates failure of the 20 to 26.5 GHz section of the FEU.
142	"Failed: Sample and hold circuit" Indicates failure of the sample and hold circuitry on the A10 PCB.
143	"Failed: Slope DAC or associated circuit" Indicates failure of the level slope DAC circuitry on the A10 PCB.
144	"Failed: RF was off when self test started" Indicates that some tests were not performed because the RF Output was selected OFF when self-test was started.
145	"Failed: A12 +10V ref." Indicates failure of the +10 Volt reference circuit on the A12 PCB.
146	"Failed: A12 –10V ref." Indicates failure of the –10 Volt reference circuit on the A12 PCB.
199	"Self Test Complete"

Appendix A Overall Command Tree

A-1 INTRODUCTION

This appendix provides an overall command tree for the Series 690XXA Synthesized CW Generator SCPI command set. The command tree is shown in Figure A-1. Refer to Chapter 3 for information on the individual SCPI commands.

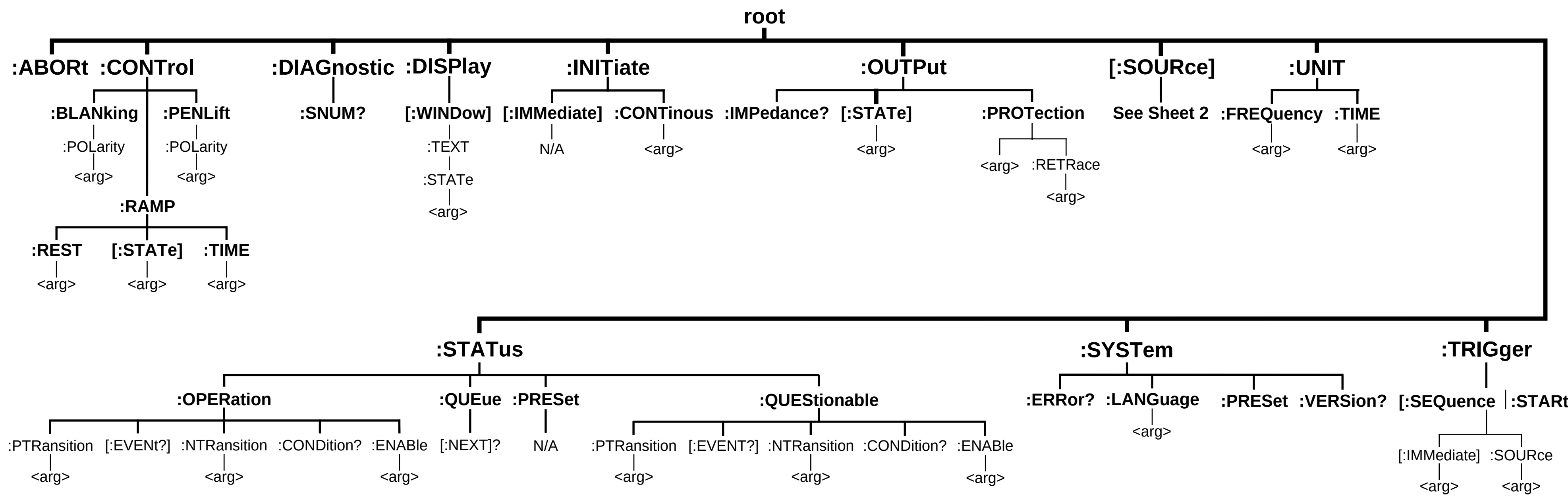


Figure A-1. *Overall 690XXA SCPI Command Tree (Sheet 1 of 2)*

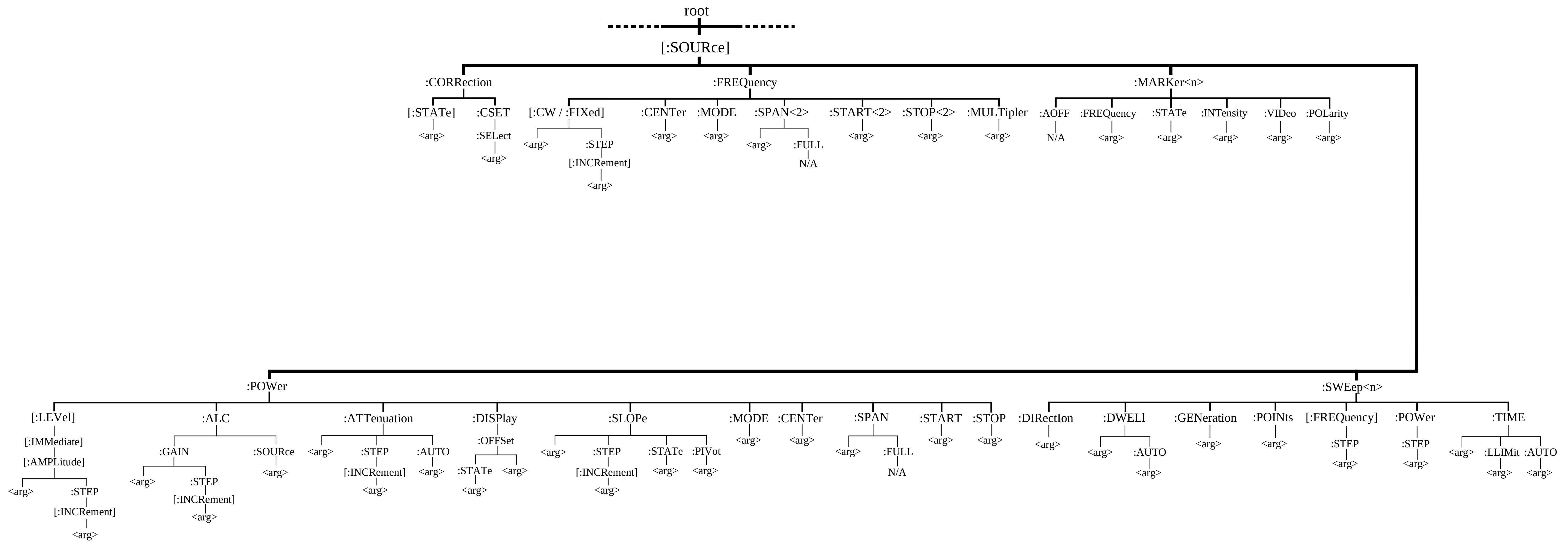


Figure A-1. *Overall 690XXA SCPI Command Tree (Sheet 2 of 2)*

Appendix B SCPI Conformance Information

B-1 INTRODUCTION

This appendix provides SCPI conformance information for the 690XXA SCPI command set in the form of a command summary. (The 690XXA SCPI command set commands and queries are described individually in Chapter 3 – Programming Commands.)

The SCPI version that the 690XXA software supports is *Standard Commands for Programmable Instruments (SCPI)* 1993.0. The first part of the table lists the SCPI Common Commands along with the conformance information for each command. The categories used for these commands are: IEEE 488.2 Required, and IEEE 488.2 Optional. The remainder of the table lists all SCPI commands and queries in the command set. The conformance categories used are: SCPI Confirmed and Non-SCPI.

NOTE
In the table, a question mark enclosed in parentheses [**(?)**] at the end of an entry indicates that the the particular command exists in both command and query forms.

690XXA SCPI Command Conformance (1 of 4)

SCPI Command	Status
∗CLS	IEEE 488.2 Required
∗ESE <nv>	IEEE 488.2 Required
∗ESE?	IEEE 488.2 Required
∗ESR?	IEEE 488.2 Required
∗IDN?	IEEE 488.2 Required
∗OPC	IEEE 488.2 Required
∗OPC?	IEEE 488.2 Required
∗RST	IEEE 488.2 Required
∗SRE <nv>	IEEE 488.2 Required
∗SRE?	IEEE 488.2 Required
∗STB?	IEEE 488.2 Required

690XXA SCPI Command Conformance (2 of 4)

SCPI Command	Status
∗TST?	IEEE 488.2 Required
∗WAI	IEEE 488.2 Required
∗OPT?	IEEE 488.2 Optional
∗RCL <n>	IEEE 488.2 Optional
∗SAV <n>	IEEE 488.2 Optional
∗TRG	IEEE 488.2 Optional
:ABORt	SCPI Confirmed
:CONTrol:BLANking:POLarity(?)	Non-SCPI
:CONTrol:PENLift:POLarity(?)	Non-SCPI
:CONTrol:RAMP:REST(?)	Non-SCPI
:CONTrol:RAMP[:STATe](?)	Non-SCPI
:CONTrol:RAMP:TIME(?)	Non-SCPI
:DIAGnostic:SNUM?	Non-SCPI
:DISPlay[:WINDow]:TEXT:STATe(?)	SCPI Confirmed
:INITiate[:IMMediate]	SCPI Confirmed
:INITiate:CONTinuous(?)	SCPI Confirmed
:OUTPut[:STATe](?)	SCPI Confirmed
:OUTPut:PROTection(?)	Non-SCPI
:OUTPut:PROTection:RETRace(?)	Non-SCPI
:OUTPut:IMPedance?	Non-SCPI
[:SOURce]:CORRection[:STATe](?)	SCPI Confirmed
[:SOURce]:CORRection:CSET:SELect(?)	SCPI Confirmed
[:SOURce]:FREQuency[:CW|FIXed](?)	SCPI Confirmed
[:SOURce]:FREQuency[:CW|FIXed] :STEP[:INCRement](?)	Non-SCPI
[:SOURce]:FREQuency:CENTer(?)	SCPI Confirmed
[:SOURce]:FREQuency:MODE(?)	SCPI Confirmed
[:SOURce]:FREQuency:SPAN(?)	SCPI Confirmed
[:SOURce]:FREQuency:SPAN:FULL	SCPI Confirmed
[:SOURce]:FREQuency:STARt(?)	SCPI Confirmed

690XXA SCPI Command Conformance (3 of 4)

SCPI Command	Status
[:SOURce]:FREQuency:STOP(?)	SCPI Confirmed
[:SOURce]:FREQuency:MULTiplier	SCPI Confirmed
[:SOURce]MARKer<n>:AOFF	SCPI Confirmed
[:SOURce]MARKer<n>:FREQuency(?)	SCPI Confirmed
[:SOURce]MARKer<n>:STATe(?)	SCPI Confirmed
[:SOURce]MARKer<n>: VIDeo(?)	Non-SCPI
[:SOURce]MARKer<n>:POLarity(?)	Non-SCPI
[:SOURce]:POWer[:LEVel][:IMMediate][:AMPLitude](?)	SCPI Confirmed
[:SOURce]:POWer[:LEVel][:IMMediate] [:AMPLitude]:STEP[:INCRement](?)	Non-SCPI
[:SOURce]:POWer[:LEVel]:ALTernate(?)	Non-SCPI
[:SOURce]:POWer:ALC:GAIN(?)	Non-SCPI
[:SOURce]:POWer:ALC:GAIN:STEP[:INCRement](?)	Non-SCPI
[:SOURce]:POWer:ALC:SOURce(?)	SCPI Confirmed
[:SOURce]:POWer:ATTenuation(?)	SCPI Confirmed
[:SOURce]:POWer:ATTenuation:STEP[:INCRement](?)	Non-SCPI
[:SOURce]:POWer:ATTenuation:AUTO (?)	SCPI Confirmed
[:SOURce]:POWer:DISPlay:OFFSet (?)	Non-SCPI
[:SOURce]:POWer:DISPlay:OFFSet:STATe(?)	Non-SCPI
[:SOURce]:POWer:SLOPe(?)	Non-SCPI
[:SOURce]:POWer:SLOPe:STEP[:INCRement](?)	Non-SCPI
[:SOURce]:POWer:SLOPe:STATe(?)	Non-SCPI
[:SOURce]:POWer:SLOPe:PIVot(?)	Non-SCPI
[:SOURce]:POWer:MODE(?)	SCPI Confirmed
[:SOURce]:POWer:CENTer(?)	SCPI Confirmed
[:SOURce]:POWer:SPAN(?)	SCPI Confirmed
[:SOURce]:POWer:SPAN:FULL	SCPI Confirmed
[:SOURce]:POWer:START(?)	SCPI Confirmed
[:SOURce]:POWer:STOP(?)	SCPI Confirmed
[:SOURce]:SWEep<n>:DIRection(?)	SCPI Confirmed
[:SOURce]:SWEep<n>:DWELl(?)	SCPI Confirmed

690XXA SCPI Command Conformance (4 of 4)

SCPI Command	Status
[:SOURce]:SWEep<n>:DWELl:AUTO(?)	SCPI Confirmed
[:SOURce]:SWEep<n>:GENeration(?)	SCPI Confirmed
[:SOURce]:SWEep<n>:POINts(?)	SCPI Confirmed
[:SOURce]:SWEep<n>[:FREQuency]:STEP(?)	Non-SCPI
[:SOURce]:SWEep<n>:POWer:STEP(?)	Non-SCPI
[:SOURce]:SWEep<n>:TIME(?)	SCPI Confirmed
[:SOURce]:SWEep<n>:TIME:LLIMit(?)	SCPI Confirmed
[:SOURce]:SWEep<n>:TIME:AUTO(?)	SCPI Confirmed
:STATus:OPERation[:EVENt]?	SCPI Confirmed
:STATus:OPERation:CONDition?	SCPI Confirmed
:STATus:OPERation:ENABle(?)	SCPI Confirmed
:STATus:OPERation:PTRansition(?)	SCPI Confirmed
:STATus:OPERation:NTRansition(?)	SCPI Confirmed
:STATus:PRESet	SCPI Confirmed
:STATus:QUEStionable[:EVENt]?	SCPI Confirmed
:STATus:QUEStionable:CONDition?	SCPI Confirmed
:STATus:QUEStionable:ENABle(?)	SCPI Confirmed
:STATus:QUEStionable:PTRansition(?)	SCPI Confirmed
:STATus:QUEStionable:NTRansition(?)	SCPI Confirmed
:STATus:QUEue[:NEXT]?	SCPI Confirmed
:SYSTem:ERRor?	SCPI Confirmed
:SYSTem:LANGuage(?)	SCPI Confirmed
:SYSTem:PRESet	SCPI Confirmed
:SYSTem:VERSion?	SCPI Confirmed
:TRIGGer[:SEQuence|:STARt][:IMMediate]	SCPI Confirmed
:TRIGGer[:SEQuence|:STARt]:SOURce(?)	SCPI Confirmed
:TSWeep	Non-SCPI
:UNIT:FREQuency(?)	Non-SCPI
:UNIT:TIME(?)	SCPI Confirmed

Subject Index