



*SmartWave™*  
*Switching*  
*Amplifier*

**SCPI Programming Manual**

*ELGAR ELECTRONICS CORPORATION*

**9250 Brown Deer Road  
San Diego, CA 92121-2294  
1-800-733-5427  
Tel: (858) 450-0085  
Fax: (858) 458-0267  
Email: [sales@elgar.com](mailto:sales@elgar.com)  
[www.elgar.com](http://www.elgar.com)**

©2002 by Elgar Electronics Corporation

This document contains information proprietary to Elgar Electronics Corporation. The information contained herein is not to be duplicated or transferred in any manner without prior written permission from Elgar Electronics Corporation.

July 24, 2002

Document No. M162000-03 Rev D



# CONTENTS

## SECTION 1 SCPI SPECIFICATION

1.1	SCPI Conformance Information.....	1-1
1.1.1	Parameter Definitions .....	1-1
1.1.2	Units.....	1-1
1.1.3	Conventions .....	1-2
1.1.4	Queries .....	1-2
1.2	Common Commands.....	1-3
1.2.1	Status Register Definitions.....	1-4
1.3	Power-On Conditions.....	1-4
1.3.1	Factory Defaults.....	1-4
1.3.2	Reset Conditions.....	1-4
1.4	Calibration Subsystem.....	1-5
1.4.1	Description .....	1-5
1.4.2	Terminology .....	1-6
1.4.3	Procedure .....	1-7
1.4.4	Calibration Command Summary .....	1-8
1.4.5	Calibration Commands.....	1-8
1.5	Display Subsystem .....	1-10
1.6	Edit Subsystem.....	1-10
1.6.1	Edit Command Summary.....	1-10
1.6.2	Edit Commands .....	1-11

---

1.7	Measure Subsystem .....	1-12
1.7.1	Measure Command Summary .....	1-12
1.7.2	Measure Commands .....	1-13
1.8	Mass Memory Subsystem .....	1-14
1.8.1	Mass Memory Command Summary .....	1-14
1.8.2	Mass Memory Commands .....	1-15
1.9	Output Subsystem .....	1-17
1.9.1	Output Command Summary .....	1-17
1.9.2	Output Commands .....	1-17
1.10	Source Subsystem .....	1-18
1.10.1	Source Command Summary .....	1-18
1.10.2	Source Commands .....	1-19
1.11	Status Subsystem .....	1-24
1.11.1	Status Command Summary .....	1-24
1.11.2	Status Commands .....	1-25
1.12	System Subsystem .....	1-26
1.12.1	System Command Summary .....	1-26
1.12.2	System Commands .....	1-27
1.13	Running a Sequence .....	1-31
1.13.1	Using the Event Status Register .....	1-34
1.14	Waveform Record Definitions .....	1-35
1.15	External Mode Compatibility .....	1-36

## SECTION 2 STATUS REGISTER DEFINITIONS

2.1	Status Byte .....	2-1
2.2	Standard Event Status Register .....	2-2
2.3	Operation Status and Questionable Status Registers .....	2-3
2.4	Error/Event Queue .....	2-3
2.5	Serial Poll Operation .....	2-3

---

## SECTION 3 STANDARD WAVEFORMS AND SEQUENCES

### SECTION 4 ERROR CODES

4.1	Error Codes Returned by SYSTEM:ERRor? Query .....	4-1
4.2	SCPI Error Codes .....	4-1
4.3	Device Specific Error Codes .....	4-2
4.3.1	Fault Codes .....	4-2
4.3.2	Memory Errors .....	4-4
4.3.3	Calibration Errors .....	4-5
4.3.4	Sequence Errors .....	4-5
4.3.5	Calibration Errors .....	4-5

### SECTION 5 SAMPLE PROGRAMS

5.1	Introduction .....	5-1
5.2	Output Programming .....	5-2
5.2.1	Phase A Programming .....	5-2
5.2.2	Simple Three Phase AC Programming .....	5-2
5.2.3	Three Phase Programming with AC and DC Output .....	5-4
5.3	Measurements .....	5-5
5.4	Editing Waveforms .....	5-6
5.4.1	Modify a Sine Wave .....	5-6
5.4.2	Modify a Sine Wave for a Specified Time .....	5-7
5.5	Sequence Examples .....	5-8
5.5.1	Voltage Ramp for a Specified Time Period .....	5-8
5.5.2	Voltage Ramp to a Value .....	5-10
5.5.3	Insert a Waveform for One Cycle .....	5-12
5.6	GPIB Programs .....	5-13
5.6.1	Uploading ASCII Waveform Files to the SmartWave .....	5-13
5.6.2	Downloading ASCII Waveform Files from the SmartWave .....	5-15
5.6.3	Uploading Binary Waveform Files to the SmartWave .....	5-17
5.6.4	Downloading Binary Waveform Files from the SmartWave .....	5-19

This page intentionally left blank.

# SECTION 1

## SCPI SPECIFICATION

### 1.1 SCPI Conformance Information

The Elgar SmartWave™ system (SW Series) conforms to all specifications for devices as defined in IEEE 488.2, and complies with SCPI command syntax version 1995.0.

Confirmed Commands are those commands which are approved commands in the SCPI 1995 Specification, Volume 2: Command Reference. They are denoted by a (C) in the following command reference. Any commands that are not Confirmed Commands have been submitted to the SCPI Consortium and are labeled as Not Approved (N).

#### 1.1.1 Parameter Definitions

Type	Valid Arguments
<boolean>	"ON", "OFF", 0 or 1
<value>	Integer or Floating point number
<name>	String enclosed by single or double quotes

#### 1.1.2 Units

The SW Series will accept the following units as suffixes to numeric values:

Type of Unit	Valid Suffix
Voltage	"Volts" or "V"
Current	"Amps" or "A"
Frequency	"Hz"
Time	"ms" (milliseconds), "s" (seconds) or "min" (minutes).

### 1.1.3 Conventions

Commands enclosed by "[ ]" are optional.

For example:

**SOURce:VOLTage:LEVel:IMMediate:AMPLitude 120.0**

can be written as:

**SOURce:VOLTage 120.0.**

### 1.1.4 Queries

The query syntax is identical to the command syntax, with a "?" appended.

For example, to query the programmed voltage on phase A, send the string:

**SOURce:VOLTage?.**

A subsequent device read will return a value such as "120.00". All queries are terminated with a carriage return and line feed (0x0D 0x0A) for those GPIB controllers that require termination characters.

When the SW Series has nothing to report, its output buffer will contain three ASCII characters; a space, carriage return, and linefeed (in decimal the values are: <32><13><10>).

## 1.2 Common Commands

The following commands are common to all SCPI instruments and declared mandatory by IEEE 488.2. In this table, the SW Series is defined as the "device" on the GPIB bus.

Command	IEEE 488.2 Definition	
<b>*CLS</b>	Clear Status Command. Clears all status reporting data structures, including the Status Byte, Standard Event Status Register and Error Queue. Enable masks are not cleared.	C
<b>*ESE</b>	Standard Event Status Enable Command. Sets the Standard Event Status Enable Register, which determines which bits can be set in the Standard Event Status Register.	C
<b>*ESE?</b>	Standard Event Status Enable Query. Returns value of Standard Event Status Enable register.	C
<b>*ESR?</b>	Standard Event Status Register Query. Returns value of Standard Event Status Register. The ESR and the Status Byte ESR bit are cleared.	C
<b>*IDN?</b>	Identification Query. Returns the device identity as an ASCII string: <manufacturer>, <model>, <serial number>, <firmware level>. Example: Elgar, SW Series 5250, 1234, 1.0	C
<b>*OPC</b>	Operation Complete Command. Causes the Operation Complete bit to be set in the Standard Events Status Register when all pending operations are complete.	C
<b>*OPC?</b>	Operation Complete Query. Causes an ASCII "1" to be placed in the output queue when all pending operations are complete.	C
<b>*RST</b>	Reset Command. Resets the device to the state defined in section 3.0. Clears all status reporting data structures, including the Status Byte, Standard Event Status Register and Error Queue. Enable masks are not cleared.	C
<b>*SRE</b>	Service Request Enable Command. Sets the Service Request Enable Register, which determines which bits in the Status Byte will cause a service request from the device.	C
<b>*SRE?</b>	Service Request Enable Query. Returns contents of Service Request Enable register. Values range from 0 to 63 or 128 to 191.	C
<b>*STB?</b>	Read Status Byte Query. Returns the Status Byte with bit 6 representing the Master Summary Status (MSS) instead of RQS. The MSS bit acts as a summary bit for the Status Byte, and indicates whether or not the device has at least one reason to request service, based on the MAV and SESR bits. The return value is in the range of 0 - 255. The Status Byte is cleared after the read.	C
<b>*TST?</b>	Self-Test Query. Causes the device to execute an internal self-test and report whether or not it detected any errors. A value of "0" indicates the test completed without detecting any errors.	C
<b>*WAI</b>	Wait-to-Continue Command. Makes the device wait until all previous commands and queries are complete before executing commands following the *WAI command.	C

### 1.2.1 Status Register Definitions

See Section 2.

## 1.3 Power-On Conditions

The following sections define the factory and reset power-on conditions of the unit. Note that if the power-on self-test fails, no GPIB commands will be serviced until the fault window on the front panel LCD is cleared by the keypad <enter> key.

### 1.3.1 Factory Defaults

When the SW is first powered up, the following factory defaults will be in place:

GPIB Address	25
RS-232 Baud Rate	57600
Number of phases (amplifiers) in chassis	3

### 1.3.2 Reset Conditions

At power-up or when the \*RST command is sent, the SW Series will be reset to the following states:

#### Amplifier Outputs

- Voltage 0.0
- Current Limit 5.0
- Frequency 60.0
- Phase Angle 0, 120, 240
- Function Sine

#### Measurement Subsystem

- All measurements are cleared

#### Waveform Subsystem

- The waveform scratchpad is cleared.

#### Sequence Subsystem

- The sequence scratchpad is cleared.
- The Sequence Execute RUN/STOP field is set to STOP. Any sequences running when a \*RST command is received are terminated.
- User Sync is set to "Every Segment".

## Instrument Subsystem

- Range is set to 156 volts.
- Coupling is set to AC.
- Sync is set to Cycle Start.
- Current Mode is set to SHUTDOWN.
- Current Time-out value is set to 100ms.
- Over Voltage limit is set to 195.0Vpeak (120Vrms + 10%)

## System Subsystem

- Self-test status will be reset.
- All External modes off.

## Mass Memory Subsystem

- The default upload/download format is ASCII.

## 1.4 Calibration Subsystem

### 1.4.1 Description

There are two categories of calibration on the SW: Output and Measurement. The CALibration:OUTPut commands calibrate the programmable output voltage and current limit. The CALibration:MEASure commands calibrate the internal test and measurement board. Calibration of the SW requires an external voltmeter, a current shunt, and a programmable load connected to its output.

There are a total of 44 calibration factors in a three phase system with a Test and Measurement board installed (16 for output calibration and 30 for test and measurement). Each number is based on the system configuration (three phase or single phase output) and the range (156V/312V).

Parameter	Output Configuration	Output/Input Channel	Range
Output Voltage	Three phase	Phase A	Low
		Phase B	Low
		Phase C	Low
		Phase A	High
		Phase B	High
		Phase C	High
Output Current	Three phase	Phase A	Low
		Phase B	Low
		Phase C	Low
		Phase A	High
		Phase B	High
		Phase C	High

Parameter	Output Configuration	Output/Input Channel	Range
	Single phase	Phase A	Low
		Phase A	High
Measured Voltage	Three phase	Phase A	N/A
		Phase B	N/A
		Phase C	N/A
Measured RMS Current	Three phase	Phase A	Low
		Phase B	Low
		Phase C	Low
		Phase A	High
		Phase B	High
		Phase C	High
	Single phase	Phase A	Low
		Phase A	High
Measured Peak Current	Three phase	Phase A	Low
		Phase B	Low
		Phase C	Low
		Phase A	High
		Phase B	High
		Phase C	High
	Single phase	Phase A	Low
		Phase A	High
Measured Power	Three phase	Phase A	Low
		Phase B	Low
		Phase C	Low
		Phase A	High
		Phase B	High
		Phase C	High
	Single phase	Phase A	Low
		Phase A	High
Measured Phase Angle	Three phase	Phase A	N/A
		Phase B	N/A
		Phase C	N/A

#### 1.4.2 Terminology

There are separate calibration factors for the different operating modes of the SW. For example, the output voltage has calibration factors for both three phase and single phase operation, and for low range and high range in each mode; a total of four calibration factors for a single amplifier. Every calibration function requires a sample number and the sample value. The calibration factor affected will depend on the programmed range (low or high) and mode (single/three phase) settings.

The calibration subsystem commands use the following notation:

Variable	Description
<#>	Sample number. <ul style="list-style-type: none"> <li>• Each OUTPut calibration factor is based on 1 or the average of 2 measurements. The sample number range is from 1 to 2.</li> <li>• Each MEASure calibration factor is based on the average of 2 measurements.</li> <li>• A sample number of 0 will instruct the SW to average the entered measurement data.</li> </ul>
<value>	Represents the measurement value from an external meter. Each value must be preceded by a sample number.

### 1.4.3 Procedure

For output voltage calibration, best results are achieved using a single data point at the voltage level specified in the procedure; 135V for low range and 270V for high range. For output current and all measurement calibration two data points should be used.

To calibrate the **output voltage** in three phase mode and low range at a single point:

- Configure the SW to three phase mode, low range
- Connect a voltmeter to the output of phase A
- Send "CALibration1:OUTPut:CLEAr to reset the current and voltage cal factors for the active configuration: three phase and low range.
- Program the output to 135 volts (270 volts for high range)
- Read the voltage from the voltmeter (this is the <value>)
- Send "CALibration1:OUTPut:VOLTagE 1 <value>". 1 is the measurement number.
- Send "CALibration1:OUTPut:VOLTagE 0 0" to calculate the calibration factor based on the one sample.

To calibrate the **measured voltage** in three phase mode and low range at two points:

- Configure the SW to three phase mode, low range
- Connect a voltmeter to the output of phase A
- Send "CALibration1:MEASure:CLEAr to reset the measured voltage, current, peak current, and power cal factors for the active configuration: three phase and low range.
- Program the output voltage to 20% of full scale
- Read the voltage from the voltmeter (this is the <value>)
- Send "CALibration1:MEASure:VOLTagE 1 <value>". 1 is the measurement number.
- Program the output voltage to 80% of full scale
- Read the voltage from the voltmeter (this is the <value>)
- Send "CALibration1:MEASure:VOLTagE 2 <value>". 2 is the measurement number.
- Send "CALibration1:MEASure:VOLTagE 0 0" to calculate the calibration factor based on the two samples.

**NOTES**

- The CLEAr command must be sent before programming the output voltage in any calibration procedure (see above example). The CLEAr command should only be used once for each phase (both OUTPut and MEASure) before calibrating the unit.
- When calibrating the Measurement System, time must be allowed for the Test and Measurement board to measure the parameter being calibrated before programming the unit to different output values. Therefore, a delay must occur after all CALibration commands. The delay time is dependent on the particular measurement, and can be as long as 20 seconds for POWer. It is recommended that a delay of at least 20 seconds be allowed after every CALibration command.

1.4.4 Calibration Command Summary

```

CALibration
  :MEASure
    :CLEAr
    :CURRent <meas number> <value>
    :PEAK <meas number> <value>
    :POWer <meas number> <value>
    :VOLTage <meas number> <value>
  :OUTPut
    :CLEAr
    :CURRent <meas number> <value>
    :INTerharmonic
      :CLEAr
    :VOLTage <meas number> <value>
      :INTerharmonic <meas number> <value>
    
```

1.4.5 Calibration Commands

CALibration[n]	System Calibration subsystem. n = 1, 2, or 3 indicating output phase to be calibrated.	C
:MEASure	Calibrate internal test and measurement board. A query command will return a gain and offset.	N
:CLEAr	Clears all output calibration parameters and restores factory defaults for the specified phase[n]. The CLEAr command must be sent to reset the calibration factors before calibration begins.	C
:CURRent <#> <value>	Calibrate RMS current measurement. The "#" parameter is the measurement number, and the value is the measurement value from an external meter.	N

:PEAK <#> <value>	Calibrate peak current measurements.	N
:POWer <#> <value>	Calibrate power measurements.	N
:VOLTage <#> <value>	Calibrate output voltage measurement. There are three voltage input channels.	N
:OUTPut	Calibrate programmable outputs. A query command will return the gain.	
:CLEar	Clears all calibration parameters and restores factory defaults for the specified phase[n]. The CLEar command must be sent to reset the calibration factors before calibration begins.	C
:CURRent <#> <value>	Set programmable output current scale factor. The "#" parameter is the measurement number, and the value is the measurement value. The calibration factor affected is dependent on the programmed range (low/high) and mode (single/three phase).	N
:INTerharmonic		N
:CLEar	Resets the IWG output calibration constant to a default value of 1.0	N
:VOLTage <#> <value>	Set programmable output voltage scale factor. The "#" parameter is the measurement number, and the value is the measurement value. The calibration factor affected is dependent on the programmed range (low/high) and mode (single/three phase).	N
:INTerharmonic <#> <value>	Calculates the IWG output calibration constant. The "#" parameter is the measurement number; the value is the measurement value from an external meter. The calibration factor affected is dependent on the programmed range (low/high) and mode (single/three phase). Follow normal calibration procedure, using 1-point calibration, with SW programmed voltage set to 0V. Applies to the SWAE only.	N

## 1.5 Display Subsystem

DISPlay		C
:CONTRast <value>	Value ranges from 0 to 1, 1 being full intensity.	C

## 1.6 Edit Subsystem

This subsystem is used to edit waveforms or sequences in scratchpad memory. To load, save, or delete waveforms or sequences, use the mass memory subsystem.

### 1.6.1 Edit Command Summary

```

EDIT
  :WAVEform
    :AMPLitude      <value>
    :FREQUENCY      <value>
    :PHASe          <value>
    :START          <value>
    :STOP           <value>
    :DURATION       <value>
    :VRMS           <value>
  :SEQUence
    :CLEAR
    :COPY
      [:SEGment]    <[src seg#] [dest seg #]>
    :CYCLES         <value>
    :DELETE
      [:SEGment]    < seg#>
    :DURATION       <value>
    :FREQUENCY      <value>
    :RAMP           <boolean>
    :INSERT         <seg #>
    :SOURCE[n]
      :AMPLitude    <value>
      :RAMP          <boolean>
      :FUNCTION      <name>
      :PHASe        <value>
    :SEGment        <seg #>
    :SYNC           <boolean>

```

## 1.6.2 Edit Commands

EDIT		N
:WAVEform	Waveform record creation/modification subsystem.	N
:AMPLitude <value>	Transient voltage amplitude. This is the modified amplitude between the start and stop phase angles defined below.	N
:FREQuency <value>	Set waveform frequency 40 - 5kHz. This value is only used in creating the waveform; the waveform may be played back at any frequency.	N
:PHASe	Sets limits of transient amplitude.	N
:STARt <value>	Starting transient phase angle	N
:STOP <value>	Stop transient phase angle	N
:DURation <value>	Time duration of transient starting from the PHASe : STARt offset. The default units are seconds.	N
:VRMS <value>	RMS voltage value of waveform	N
:SEQuence	Sequence creation/modification subsystem	N
:CLEar	Deletes all segments in the scratchpad.	
:COPY		N
[:SEGment] <src # dst #>	Copies the segment number specified by "src" to the segment number specified by "dst". Both "src" and "dst" must already exist.	N
:CYCLes <value>	Number of segment cycles. Writing this value will modify the DURation value. A comma is required to separate numeric fields.	N
:DELeTe		N
[:SEGment] <seg_no>	Deletes a single segment in the scratchpad.	N
:DURation <value>	Segment duration. Writing this value will modify the CYCLes value. Default units are seconds.	N
:FREQuency <value>	Segment frequency. Writing this value will modify the CYCLes value.	N
:RAMP <boolean>	Frequency ramp select.	N

:INSert <seg_no>	Inserts a segment at the segment number specified and sets the segment number 1 greater than the preceding segment. For Example, if only segment 0 exists, sending EDIT:SEQUENCE:INSERT 1 will create segment 1 and move the current segment pointer to segment 1.	N
:SOURce[n]	Select phase output to modify. Valid values for n are 1, 2 or 3. If n is not specified, n = 1 is assumed.	N
:AMPLitude <value>	Output amplitude.	N
:RAMP <boolean>	Amplitude ramp select.	N
:FUNction <name>	Output function.	N
:PHASe <value>	Output phase angle.	N
:SEGment <value>	Select segment to modify.	N
:SYNC <boolean>	Enable/disable sync output for the current segment. The Selected Segments option must be selected for this field to take effect when running a sequence.	N

## 1.7 Measure Subsystem

The Measure Subsystem utilizes the SW Series Test and Measurement Board. The values are returned in floating point format with a unit suffix of V, A, Hz or DEG. These functions will not be available if the Test and Measurement option is not installed.

### 1.7.1 Measure Command Summary

```

MEASure[n]
  :CLEAr
  :CURRent?
    :PEAK?
  :FREQuency?
  :PHASe?
  :POWER?
    :TOTal?
  :POWERFACtor?
    :TOTal?
  :VA?
    :TOTal?
  :VOLTage?
    :VAB?
    :VBC?
    :VCA?

```

## 1.7.2 Measure Commands

MEASure[n]	n = 1, 2, or 3 for Phase A, B or C. The default (e.g., MEAS:VOLT?) is phase A. n = 0 is invalid for this subsystem.	C
:CLEAr	Deletes all measurements.	C
:CURRent?	Returns RMS output current in amps.	C
:PEAK?	Returns peak output current in amps.	
:FREQuency?	Returns output frequency in hertz.	C
:PHASe?	Returns output phase angle relative to phase A in degrees. Note that all phase offsets are in terms of phase lead with respect to the reference signal and phase A.	C
:POWer?	Returns output power in watts.	C
:TOTal?	Returns total power. The phase [n] has no effect.	C
:POWERFACtor?	Returns power factor.	C
:TOTal?	Returns total power factor. The phase [n] has no effect.	C
:VA?	Returns VA.	C
:TOTal?	Returns total VA. The phase [n] has no effect.	C
:VOLTage?	Returns output voltage. AC RMS for sinusoidal waveforms, DC for DC waveforms.	C
:VAB?	Returns line to line voltage for phases A and B. The phase [n] has no effect.	N
:VBC?	Returns line to line voltage for phases B and C. The phase [n] has no effect.	N
:VCA?	Returns line to line voltage for phases C and A. The phase [n] has no effect.	N

## 1.8 Mass Memory Subsystem

The MMemory Subsystem is used to create/modify waveforms stored in the SW Series. To conform to SCPI syntax, there are many "types" of memory storage. Each entry of a particular type of memory is called a TABLE. Memory types are described below:

Type	Description
Waveform	Waveform storage including standard factory waveforms stored in ROM as well as user defined waveforms stored in non-volatile RAM.
Sequence	Sequence storage including standard factory sequences stored in ROM as well as user defined sequences stored in non-volatile RAM.
Setup	User defines front panel setups stored in non-volatile RAM.

Mass Memory subsystem SCPI commands relating to total amount of memory, memory available, storing or deleting tables and entries refer to non-volatile RAM. Only the LOAD command can access table entries stored in ROM.

### 1.8.1 Mass Memory Command Summary

```

MMEMory
  :CATalog
    :SEQuence?
    :WAVEform?
  :DELete
    :SEQuence <seq name>
    :WAVEform <wfrm name>
  :DOWNload
    :WAVEform
  :FORMat <ascii or binary>
  :LOAD
    :SEQuence <seq name>
    :WAVEform <wfrm name>
  :SCALef
    :WAVEform <wfrm name> <value>
  :STORE
    :SEQuence <seq name>
    :WAVEform <wfrm name>
    :RMS <wfrm name>
  :UPload
    :WAVEform

```

## 1.8.2 Mass Memory Commands

MMEMory		C
:CATalog	<p>Queries information on the current contents and state of the mass memory. The returned data will have the following form:</p> <p style="text-align: center;">&lt;value 1&gt;, &lt;value 2&gt; {,&lt;string&gt;}</p> <p>value 1 = memory used, in bytes. value 2 = memory available, in bytes.</p> <p>&lt;string&gt; has the following format:</p> <p style="text-align: center;">&lt;name&gt;, &lt;type&gt;, &lt;size&gt;</p> <p>name = name of item type = waveform, sequence or setup size = size in bytes</p> <p>All information is returned in ASCII.</p>	C
:SEQuence?	Catalog sequences.	N
:WAVEform?	Catalog waveforms.	N
:DELete	Deletes a file from the specified mass storage area.	C
:SEQuence <name>	Delete sequence.	N
:WAVEform <name>	Delete waveform.	N
:DOWNload	Commands SW Series to download selected information to the host computer.	C
:WAVEform	Download waveform. All waveform downloads originate from the Waveform Scratchpad. Library waveforms should be placed in the scratchpad with the MMEM:LOAD:WAVEform command. A waveform consists of 4096 12-bit data points. In ASCII format 16386 bytes are transferred, in binary format, 8192 bytes are transferred. Sample programs are in Section 5.	N

:FORMat <value>	Sets upload/download format. A value of 0 = ASCII, 1 = Binary. The default is ASCII. Note that in Binary mode: <ul style="list-style-type: none"> <li>• The MSB (most significant byte) needs to be sent first (before the LSB) for uploads.</li> <li>• The MSB is the first byte returned from the SW on downloads.</li> <li>• The System GPIB Controller must be configured for an EOI (end or identify) terminator and no EOS (end of string) terminator.</li> </ul>	C
:LOAD	Loads a file from non-volatile memory.	C
:SEQuence <seq_name>	Loads a sequence from non-volatile memory to sequence scratchpad.	N
:WAVEform <wfrm_name>	Loads a waveform from non-volatile memory to the waveform scratchpad.	N
:SCALef	Set/query scale factor information	N
:WAVEform <name> <value>	Sets the user-defined waveform scale factor. This command will only work on user-defined waveforms stored in non-volatile memory. To query a scale factor, use the format: MMEM:SCALEF:WAVE? <name>.	N
:STORe	Stores a file to non-volatile memory.	C
:SEQuence <seq_name>	Saves a sequence from the sequence scratchpad to non-volatile memory	N
:WAVEform <wfrm_name>	Stores a waveform from the waveform scratchpad to non-volatile memory. The waveform scale factor is calculated to maintain the peak voltage of the original waveform fundamental (the waveform loaded into the scratchpad before editing). This will cause the output RMS voltage to be higher than programmed for spikes, and lower than programmed for drop-outs.	N
:RMS <wfrm_name>	Stores a waveform from the waveform scratchpad to non-volatile memory. The waveform scale factor is calculated to output the programmed output RMS voltage.	N

:UPload	Commands SW Series to receive selected information from the host computer. An upload will put the data into a scratchpad area, which can then be saved to non-volatile memory. A waveform consists of 4096 12-bit data points. In ASCII format 16386 bytes are required to be transferred, in binary format, 8192 bytes. Section 5 contains sample programs.	C
:WAVEform	Upload to waveform scratchpad.	N

## 1.9 Output Subsystem

Controls the output port.

### 1.9.1 Output Command Summary

#### OUTPut

```

:AUX          <boolean>
:COUPling    <AC | DC>
:INTerharmonic <boolean>
[:STATe]     <boolean>

```

### 1.9.2 Output Commands

OUTPut	Output subsystem. This command may be followed by a boolean indicating the output terminal state (see STATE command). All phases are affected by this command.	C
:AUX <boolean>	Controls state of auxiliary relay. This is an option on some models. *RST value is OFF.	N
:COUPling <AC   DC>	Controls whether the output is in AC or AC+DC mode. *RST value is AC.	C
:INTerharmonic <boolean>	Sets the output switch to no output (0) or to the programmed output (1). This command applies to the SWAE only.	N
[:STATe] <boolean>	Controls whether the output terminals are open or closed. Valid arguments are ON (1) or OFF (0). *RST value is OFF. Note: a delay of 1 second is required after changing the relay state before any program command is sent.	C

## 1.10 Source Subsystem

### 1.10.1 Source Command Summary

SOURce[n]	
:CURRent	
[:LEVel]	
[:IMMediate]	
[:AMPLitude]	<value>
:PROTection	
:CLEAr	
:CURTimeout	
:STATe	<boolean>
[:TIME]	<value>
[:LEVel]	<value>
:STATe	<boolean>
:TRIPped?	
:LOCK	<boolean>
:FREQuency	<value>
:INTerharmonic	<value>
:FUNCTion	
:INTerharmonic	<"wfrm name">
:LOCK	<boolean>
:MODE	<mode>
[:SHAPE]	<wfrm name>
:PHASe	
[:ADJust]	<value>
:LOCK	<boolean>
:SEQuence	
:LOAD	<seq name>
:LOOP	<value>
:MODE	
:RUN	<REPeat   SINGle   LOOP>
:STOP	<ZERO   PROGram   SEGment>
:SEGment	
:COMPLete?	
:REPeat	
:STEP	
[:STATe]	<RUN   STEP   STOP>
:SYNC	<ALL   SElect>
:VOLTage	
[:LEVel]	
[:IMMediate]	
[:AMPLitude]	<value>
:PROTection	
[:LEVel]	<value>
:STATe	<boolean>
:TRIPped?	
:INTerharmonic	<value>
:LOCK	<boolean>
:RANGe	<value>

## 1.10.2 Source Commands

SOURce[n]	n = 0, 1, 2, or 3 The default phase is 1 (phase A). To program all phases at the same time, use the LOCK command and program SOURce1 (phase A). A value of n = 0 will program all phases.	C
:CURRent	Programs the current level in current foldback mode. This command may be followed by an amplitude. At *RST, the current amplitude is set to maximum. Foldback mode cannot be used when SYSTem:EXTernal:LOWFREQuency is on.	C
[:LEVel]	Sets the output current amplitude. This command may be followed by an amplitude.	C
[:IMMediate]	Process new amplitude without waiting for more commands. Must be followed by the :AMPLitude command.	C
[:AMPLitude] <value>	Sets the output current amplitude.	C
:PROTection	Programs the current level in current shutdown mode. This command may be followed by a value.	C
:CLEar	Clears protection circuit. This is the same as using the STATe OFF command.	C
:CURTimeout	Programs the current level in current foldback mode for the value specific in TIME, then reverts to current shutdown mode for the current level specified. Current time-out mode cannot be used when SYSTem:EXTernal:LOWFREQuency is on.	N
:STATe <boolean>	Enable/disable current mode time-out	N
[:TIME] <value>	Set current mode time-out in ms	N
[:LEVel] <value>	Sets output level at which the output protection circuit will trip. At *RST, the value is set to maximum.	C
:STATe <boolean>	Enable/disable protection circuit. At *RST, the value is on.	C
:TRIPped?	Returns 1 if protection circuit has tripped, 0 if it is untripped. This query clears the tripped flag. A current limit shutdown will open the output relay. To recover from a trip, close the output relay.	C

:LOCK <boolean>	Locks the current register of all three phases. Only phase A may be programmed, and any value programmed will affect all three phases. The Lock function was created for front panel control. Use the SOURce0 command for GPIB control. The "0" will cause all three phases to be programmed.	N
:FREQuency <value>	The Frequency Subsystem controls the output frequency of the instrument. Note that a frequency change to any phase of a multi-phase instrument will cause a frequency change in all phases of that unit. This command may be followed by a frequency value. At *RST, the value is set to 60 Hz.	C
:INTerharmonic <value>	Sets the IWG board to the programmed frequency specified by "value." Frequency range is 40-500Hz. This command applies to the SWAE only.	N
:FUNCTion	The Function Subsystem controls the shape and attributes of the output signal.	C
:INTerharmonic <wfrm name>	Sets the IWG board to the programmed function specified by "wfrm name." Text for programmed function is case-sensitive and must be enclosed in double quotes. Currently, the only waveform available for this command is "Sine." This command applies to the SWAE only.	N
:LOCK <boolean>	Locks the function register of all three phases. Only phase A may be programmed, and any value programmed will affect all three phases. The Lock function was created for front panel control. Use the SOURce0 command for GPIB control. The "0" will cause all three phases to be programmed.	N
:MODE <mode>	Determines which signal characteristic is being controlled. Possible values are VOLTage, CURRent or POWer. Only the VOLTage mode is currently implemented. At *RST, the mode is set to VOLTage.	C

[:SHAPE] <wfrm_name>	Selects the shape of the output signal. A complete list of standard waveform names is included in Section 3. At *RST, the output waveform is set to SINE.	C
:PHASe	The Phase Subsystem controls the phase offset (lead) of each phase relative to an internal reference. This command may be followed by a phase value. Phase offsets are programmed/queried in degrees. At *RST, phase A = 0, phase B = 120 and phase C = 240 degrees.	C
[:ADJust] <value>	Controls the phase offset value relative to the reference.	C
:LOCK <boolean>	Locks the phase register of all three phases. Only phase A may be programmed, and any value programmed will affect all three phases. The Lock function was created for front panel control. Use the SOURce0 command for GPIB control. The "0" will cause all three phases to be programmed.	N
:SEQuence		N
:LOAD <seq_name>	Selects sequence to execute. A sequence must be loaded before it can be executed. This command is not the same as the function to load a sequence into the sequence scratchpad.	N
:LOOP <loop count>	Selects the number of times to repeat the loaded sequence. The Run Mode must be set to LOOP for this field to have an effect. The range is from 1 to 9999.	N
:MODE		N
:RUN <REPeat   SINGLE   LOOP>	Controls the run mode of the loaded sequence. A sequence is started with the SOURCE:SEQ [:STATE] command. <b>REPeat</b> will cause the sequence to loop until SOURCE:SEQ [:STATe] STOP is selected. <b>SINGLE</b> will cause the sequence to execute once and terminate. <b>LOOP</b> will cause the sequence to repeat the number of times specified by SOURCE:SEQ:LOOP <#>, and terminate. A query will return REPEAT, SINGLE, or LOOP.	N

<p>:STOP &lt;ZERO   PROGRAM   SEGMENT&gt;</p>	<p>Controls the termination mode of the loaded sequence. A sequence is started with the SOURCE:SEQ [:STATE] command.</p> <p><b>ZERO</b> will cause the outputs to be programmed to 0 volts.</p> <p><b>PROGRAM</b> will cause the outputs to be set to the values in the values programmed before the sequence began.</p> <p><b>SEGMENT</b> will cause the outputs to remain at the values of the last cycle of the last segment executed. Note that in this mode the values in the program registers will not match the outputs. A query will return REPEAT, SINGLE, or LOOP.</p>	<p>N</p>
<p>:SEGMENT</p>		<p>N</p>
<p>:COMPLETE?</p>	<p>Query returns 1 when current segment is complete, or 0 if current segment is still executing. This command is only valid when running a sequence in “step” mode.</p>	<p>N</p>
<p>:REPEAT</p>	<p>Repeats the last segment executed when running a sequence in STEP mode.</p>	<p>N</p>
<p>:STEP</p>	<p>Executes the next segment when running a sequence in STEP mode.</p>	<p>N</p>
<p>[:STATE] &lt;RUN   STEP   STOP&gt;</p>	<p>Starts or Stops the execution of the loaded sequence. The operating mode of the sequence is determined by SOURCE:SEQ:MODE:RUN (Repeat, Single, and Loop), and the SOURCE:SEQ:MODE:STOP (Zero, Program, or Segment).</p> <p><b>RUN</b> will start the sequence.</p> <p><b>STEP</b> will start the sequence and pause at the end of each segment until a :SEG:STEP or :SEG:REPEAT command is received.</p> <p><b>STOP</b> will terminate the sequence at the current segment. Note that when a sequence is stopped in this manner, the values in the program registers will not reflect the actual output values.</p> <p>A query will return the CONTROL mode of operation: RUN, STEP, or STOP.</p>	<p>N</p>

:SYNC <ALL   SElect>	Sets the SYNC Output configuration to Selected Segments or All Segments. The default setting is Selected Segments, in which case only those segments that have been enabled with the EDIT:SEquence:SYNC 1 command will generate a sync pulse.	N
:VOLTage	The Voltage Subsystem controls the output signal amplitude. This command may be followed by a voltage value. At *RST, all the outputs are set to 0 volts.	C
[:LEVel]	Programs output voltage amplitude. This command may be followed by a voltage value.	C
[:IMMEDIATE]	Process command without waiting for further commands.	C
[:AMPLitude] <value>	Set the output voltage amplitude.	C
:PROTection	Programs the maximum <b>peak voltage</b> level at which the output protection circuit will trip. The overvoltage protection circuit can not be disabled.	C
:[LEVel] <value>	Programs the maximum <b>peak voltage</b> level at which the output protection circuit will trip.	C
:STATe <boolean>	Enable/disable protection circuit. This command is present for compatibility. The overvoltage circuit cannot be disabled. A query will always return 1 (enabled).	C
:TRIPped?	Returns 1 if protection circuit has tripped, 0 if untripped. This query clears the tripped flag. A voltage limit shutdown will program the output voltage to 0 volts and open the output relay. To recover from a trip, program the output voltage and close the output relay.	C
:INTerharmonic <value>	Sets the IWG board to the programmed voltage specified by "value." Voltage range is 0-46V. This command applies to the SWAE only.	N

:LOCK <boolean>	Locks the voltage register of all three phases. Only phase A may be programmed, and any value programmed will affect all three phases. The Lock function was created for front panel control. Use the SOURce0 command for GPIB control. The “0” will cause all three phases to be programmed.	N
:RANGe <value>	Sets output voltage range. The value can be 0 or 1, corresponding to 156 or 312 volts respectively. This command will cause all outputs to be programmed to 0 V before the range change takes effect. At *RST, the range is set to 156 volts.	C

## 1.11 Status Subsystem

This subsystem controls the SCPI-defined status-reporting structures.

### 1.11.1 Status Command Summary

```

STATus
  :OPERation
    :CONDition?
    :ENABle      <value>
    [:EVENT]?
  :QUESTionable
    :CONDition?
    :ENABle      <value>
    [:EVENT]?
  :PRESet
    
```

## 1.11.2 Status Commands

STATus	Status Subsystem	C
:OPERation		C
:CONDition?	Returns contents of the Operation Condition register. The query is supported, but will always return "0" indicating operational condition.	C
:ENABle <value>	Sets the enable mask of the Operation Event Register, which allows true conditions to be reported in the summary bit of the Operation Condition Register. Values may be written and queried, but have no effect on the Operation Condition Register.	C
[:EVENT]?	Returns the contents of the Operation Event Register. This query is supported, but always returns a value of "0", indicating operational condition.	C
:QUESTionable		C
:CONDition?	Returns contents of the Questionable Condition register. The query is supported, but will always return "0" indicating operational condition.	C
:ENABle <value>	Sets the enable mask of the Questionable Event Register, which allows true conditions to be reported in the summary bit of the Questionable Condition Register. Values may be written and queried, but have no effect on the Questionable Condition Register.	C
[:EVENT]?	Returns the contents of the Questionable Event Register. This query is supported, but always returns a value of "0", indicating operational condition.	C
:PRESet	Preset the enable registers to all 1's.	C

## 1.12 System Subsystem

### 1.12.1 System Command Summary

```
SYSTem
:AMPLIFIER?
:BEEPer
  :STATe      <boolean>
  :TEST       <boolean>
:COMMunicate
  :GPIB
    :ADDRess  <value [, <value]>
:CONFig
  :BOOST      <boolean>
:ERRor?
:EXTernal
  :CLOCK      <OFF | INPut | OUTPut>
  :LOCK?
  :DIRect     <boolean>
  :GAIN       <boolean>
  :IECLoad    <boolean>
  :LOWFREQuency <boolean>
  :MODulation <boolean>
  :XLOAD      <boolean>
:PARALlel    <boolean>
:SYNC        <0 | 1 | 2> // off, on, or event
:VERSion?
```

## 1.12.2 System Commands

SYSTem	System Subsystem.	C
:AMPLIFier?	Returns the number of installed amplifiers in the chassis.	C
:BEEPer	This subsystem controls the audible beeper of the instrument.	C
:STATe <boolean>	Enables/disables the beeper. Valid commands are ON (1) or OFF (0).	C
:TEST <boolean>	Turns beeper on (1) and off (0).	C
:COMMunicate	This subsystem configures the communication interfaces.	C
:GPIB	This subsystem configures the GPIB interface.	C
:ADDRess <value> [,<value>]	Sets the GPIB address of the instrument. The optional secondary addressing is not currently used. Power must be cycled before the new address will take effect.	C
:CONFig	System Configuration Subsystem	N
:BOOSt <boolean>	<p>Available on SWA only. Default for BOOST is on. This improves load regulation for the following conditions:</p> <ul style="list-style-type: none"> <li>• output voltage &lt; 30</li> <li>• load current &gt; 20% of full scale</li> <li>• frequency &gt; 200Hz</li> </ul> <p>Turning BOOST off improves the low frequency transient response of the RMS servo system, and is the mode used in sequence operation. This enables the SWA to faithfully reproduce programmed voltage sags and surges with quick transitions.</p> <p>When running a sequence with the stop mode set to "PROGram," BOOST should be set to off to prevent any transient voltage levels from occurring during the transition out of sequence mode.</p>	N

:ERRor?	<p>Query the error queue for the next error/event entry (queue is first in, first out). The entry contains an error number and descriptive text. A return value of 0 indicates no error occurred; negative numbers are reserved by the SCPI standard. The maximum return string length is 255 characters. The queue holds up to 10 error/entries. All entries are cleared by the *CLS command.</p> <p>The error status line on the bottom of the front panel LCD is cleared in response to this query.</p>	C
:EXTernal		N
:CLOCK <OFF   INPut   OUTPut>	<p>Configures the external clock/lock input on the rear panel.</p> <ul style="list-style-type: none"> <li>• OFF will disable all clock/lock functions.</li> <li>• INPut will attempt to sync the outputs to the input signal (0 - 5V TTL). Note that the FREQ field of the Program Menu must be set the clock/lock input signal frequency before INPut is selected. The SW internal PLL must have a lock on the clock/lock input before the output relay is allowed to close.</li> <li>• OUTPut will configure the clock/lock line as an output representing the frequency of the output waveforms.</li> </ul>	N
:LOCK?	<p>Query returns 0 or 1, indicating internal PLL is not locked (0), or locked (1). The output relay will not close in EXTernal:CLOCK INPut mode unless the PLL is locked. If lock is lost after the output relay is closed, a fault will occur and the output relay will automatically open.</p>	N

:DIRect <boolean>	<p>Configures amplifier reference signals to run from internal source (off) or external input on rear panel (on). If external input is turned on, the following system defaults will be programmed on all output phases:</p> <ul style="list-style-type: none"> <li>• The output amplitude will be programmed to maximum (156 or 312Vrms)</li> <li>• If the output current limit is greater than 10A, it will be programmed to 10A to remain under the unit power limit. If less than 10A, it will not be modified.</li> <li>• The output function will be programmed to a SINE wave.</li> </ul> <p>These values will be reflected in the Program Menu and GPIB queries. All values may be modified after the direct mode has been selected.</p>	N
:GAIN <boolean>	Enables/disables external gain control from the rear panel input.	N
:IECLoad <boolean>	Sets output impedance to equal $0.400+j0.25 \Omega$ at 50 Hz. This command applies to the SWAE and SWA-106 only.	N
:LOWFREQuency <boolean>	This option should be used for external AC input signals between DC and 40Hz. It should not be used in normal operation, with an external DC input signal, or with frequencies above 40Hz. The only valid current mode is current shutdown (not foldback or time-out).	N
:MODulation <boolean>	Enables/disables external modulation from the rear panel input.	N
:XLOAD <boolean>	This option can be used to reduce the undershoot, overshoot and ringing with unusual reactive loads. This option should not be used with normal loads. This command applies to the SWA only.	N
:PARALlel <boolean>	For systems with more than one amplifier, parallel ON will parallel the output phases to act as a single phase A. Parallel OFF will allow each phase to act independently. WARNING: rear panel wiring must agree with software configuration.	N

:SYNC <0   1   2>	<p>Configures the Sync Output when running in normal (non-sequence) mode.</p> <p>0 = OFF. Allows triggering exclusively on selected segments - those segments with a trigger enabled with the EDIT:SEQ:SYNC 1 command.</p> <p>1 = ON. Sync Output is a square wave at the current output frequency.</p> <p>2 = EVENT. A sync pulse will be generated on any programming change of voltage, frequency, phase angle, and function (waveform). For example, SOUR:VOLT 100, SOUR:FREQ 50, SOUR:PHASE 90, SOUR:FUNC "Triangle" would all cause a sync pulse to be generated.</p>	N
:VERSion?	Returns a numeric value corresponding to the SCPI version number for which the instrument complies. The response is in the format YYYY.V where the Y's represent the year and V represents the approved version number for that year.	C

## 1.13 Running a Sequence

Once a sequence has been selected, an execution mode must be configured. The configuration is accomplished through the Sequence Execute Menu from the front panel, or the SOURCE:SEQ: remote commands. The remote commands correlate directly with the front panel, so it is usually helpful to experiment with the Sequence Execute Menu to become familiar with the various operating modes.

To begin execution of a sequence:

1. Set Run Mode to Repeat, Single, or Loop.
2. Set Stop Mode to Zero, Program, or End Seg (Segment for GPIB).
3. Set Control to Run or Step to start execution.

An explanation of the different modes is described below in the format of the front panel Sequence Execute Menu. Refer to the SOURCE:SEQ commands for exact GPIB syntax.

A Sequence can be executed in any combination of the following modes:

- A sequence can be run in its entirety, or stepped through one segment at a time. When in this “step” mode, the most recent segment can be repeated.
- A sequence can be executed only once, repeated a specified number of times, or repeated until the STOP command is selected.
- A sequence can be terminated with the outputs automatically programmed to 0 volts, restored to the waveforms and values before the sequence began, or remain at the waveforms and values of the last segment in the sequence. The last two options will occur with no interruption in output power.

The Sequence Execution Menu is shown below:

<b>LOAD SEQ</b>	Loads a sequence from the sequence library or the sequence scratchpad. This field was used to select the sequence name in previous versions. When running under GPIB control, the Event Status Register and serial polling can be used to indicate when the sequence loading is complete. The front panel display will indicate "Processing sequence..." while loading, and "Sequence loaded" when complete.
<b>SYNC</b>	Configures the SYNC Trigger output to generate a pulse for every segment or only selected segments.
<b>CONTROL</b>	<p>Used to start a sequence, stop a sequence, or start a sequence in STEP mode. A sequence must be loaded before RUN or STEP can be selected. All operations have an immediate effect.</p> <p><b>RUN</b> - Run the previously loaded sequence. A sequence does not need to be reloaded when switching between Run, Step, or Stop. The sequence will begin executing immediately after a Run or Step command.</p> <p><b>STEP</b> - Step through the previously loaded sequence. Each segment will execute and remain at the value of the last cycle until instructed to execute the next segment, or repeat the current segment. When stepping through a sequence under front panel control, the F1 function key is used to execute the next segment, and the F2 key is used to repeat the current segment.</p> <p><b>STOP</b> - Stops the active sequence. Note that if Stop Mode is not set to PROGRAM, the Program Menu (and GPIB program readback values) will not be updated to the current output values.</p>
<b>RUN MODE</b>	<p>Specifies the running mode of the sequence. This parameter takes effect once a sequence has begun.</p> <p><b>REPEAT</b> - Repeats sequence until a STOP command is received.</p> <p><b>SINGLE</b> - Executes the sequence only once, then returns to the operation specified in the Stop Mode field.</p> <p><b>LOOP</b> - Repeat sequence the number of times specified by the LOOP count field.</p>

<b>STOP MODE</b>	<p>Specified the mode of operation when a sequence is terminated. This can occur when running in the SINGLE execution mode, or when STOP is selected.</p> <p><b>ZERO</b> - Programs the outputs to 0 volts when the sequence is terminated.</p> <p><b>PROGRAM</b> - The outputs are restored to the waveforms and values in the Program Menu. This mode can be used for a continuous output between sequences. The Program Menu cannot be modified while a sequence is running.</p> <p><b>END SEG</b> - The outputs will remain at the waveforms and values of the last segment in the sequence</p>
<b>LOOP</b>	<p>Specified the number of times to repeat the sequence when Run Mode is set to LOOP.</p>

### 1.13.1 Using the Event Status Register

The length of time required to load a sequence before execution is dependent upon the number of segments and the number of different waveforms in the sequence; the more segments and different waveforms, the longer the load time. Once a sequence is loaded, the Run Mode, Stop Mode, and Control fields may be changed without reloading the sequence.

When loading a sequence with the SOURCE:SEQ:LOAD “name” command, serial polling and the ESR register can be used to determine when loading is complete. An example function is shown below:

```

/*-----
Load a sequence. The name of the sequence to be loaded is passed as name. The
function send_cmd( ) sends a string to the SW over the GPIB.
-----*/
int load_sequence(char *name)
{
    unsigned char sp_byte;      // serial poll byte
    long ctr;                   // time-out counter

    send_cmd("*ESE 1");          // configure ESR mask to enable OPC, bit 0
    printf("Loading sequence: %s...\n", name); // send command to load sequence
    send_cmd("SOUR:SEQ:LOAD \"%s\"", name);

    printf("Serial Poll for OPC...\n");          // wait for load complete
    ctr = 0;
    sp_byte = serial_poll( );
    while (!(sp_byte & 0x20) && (ctr < 10000)) // wait for bit 5 to be set, or time-out
    {
        printf("Polls: %d\r", ctr);
        sp_byte = serial_poll( );
        ctr++;
        delay(1000);                // delay 1000ms
    }
    printf("\n");

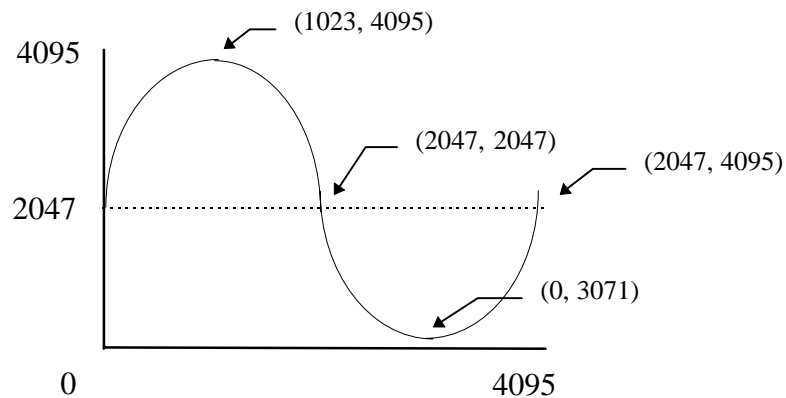
    if (ctr == 10000)
    {
        printf("SEQUENCE LOAD TIMEOUT !!!!!\n");
        return (ERROR);
    }
    return (OK);
}

```

## 1.14 Waveform Record Definitions

Waveforms are stored in the Smart Wave as 4096 x 4096 records. On the vertical axis (amplitude), a value of 2047 corresponds to 0V output, a value of 0 corresponds to maximum negative voltage, and a value of 4095 corresponds to maximum positive voltage. On the horizontal axis (time), a value of 0 represents the start of the waveform record, and 4095 the end. A waveform does not have a specified output voltage or frequency; any waveform may be programmed to any output value.

An example of a sine wave record is shown below:



Waveform records may be created as 8192 byte binary files, or as 16384 byte ASCII files. The GPIB command `MMEMory:FORMat <value>` selects ASCII (value = 0) or binary (value = 1). All uploading and downloading of waveforms is done through the waveform scratchpad area in the SmartWave. To upload and save a waveform in the SW non-volatile RAM, the waveform is uploaded to the scratchpad using the `MMEMory:UPload` command, and then saved using the `MMEMory:STORE:WAVEform <wfrm name>` or `MMEMory:STORE:WAVEform:RMS <wfrm name>` command. To download a waveform, first load the waveform from the NV RAM of the SW with the `MMEMory:LOAD:WAVEform <wfrm name>` command, and then download from the scratchpad using the `MMEMory:DOWNload:WAVEform` command.

## 1.15 External Mode Compatibility

The following chart shows which external modes are compatible with each other. “Yes” indicates both modes can be active at the same time; “no” indicates they cannot.

	<b>Amplitude Modulation</b>	<b>External Gain</b>	<b>External Direct</b>	<b>Xload</b>	<b>Low Frequency</b>	<b>Clock Lock</b>
Amplitude Modulation	—	no	no	yes	yes	yes
External Gain	no	—	no	yes	yes	yes
External Direct	no	no	—	yes	yes	no
Xload	yes	yes	yes	—	yes	yes
Low Frequency	yes	yes	yes	yes	—	yes
Clock Lock	yes	yes	no	yes	yes	—

## SECTION 2

# STATUS REGISTER DEFINITIONS

The SW Series supports the IEEE 488.2 and SCPI 1993.0 status reporting data structures. These structures are comprised of status registers and status register enable mask pairs. These pairs are described below.

### 2.1 Status Byte

The Status Byte status register can be read by the \*STB? command or by issuing a GPIB serial poll. Either operation will clear the contents of the Status byte. The \*CLS command will clear the Status Byte.

The SW Series unit can be configured to request service from the GPIB controller by setting the appropriate bits in the Service Request Enable register. The SRE register has the same bit pattern as the Status Byte. The SRE register is modified using the \*SRE command, and can be read with the \*SRE? command. For example, if the SRE register is set to 0x10 (MAV), when the SW Series unit has a message available, the Status Byte register will contain 0x50 (RQS and MAV) and the SRQ line of the GPIB bus will be pulled low indicating a request for service.

Bit	Hex Value	Description
0	01	Not used
1	02	Not used
2	04	Error/event queue message available
3	08	Questionable Status flag. Indicates quality of current data being acquired. This bit is not used.
4	10	Message available (MAV)
5	20	Standard Event Status Register (ESR)
6	40	Request Service flag (RQS) for serial polling, or Master Summary Status (MSS) in response to *STB?
7	80	Operation Status flag. Indicates the current operational state of the unit. This bit is not used.

**Bit 2, Error/event queue information available**

This bit is set when any error/event is entered in the System Error queue. It is read using the `SYSTEM:ERRor?` query.

**Bit 4, Message Available**

Indicates a message is available in the GPIB output queue. This bit is cleared after the GPIB output buffer is read.

**Bit 5, Standard Event Status Register**

This is a summary bit for the ESR. It is set when any of the ESR bits are set, and cleared when the ESR is read.

**Bit 6, Request Service/Master Summary Status**

If service requests are enabled (with the `*sre` command) this bit represents the RQS and will be sent in response to a serial poll, then cleared. If RQS is not enabled, the bit represents the MSS bit and indicates the device has at least one reason to request service. Even though the device sends the MSS bit in response to a status query (`*STB?`), it is not sent in response to a serial poll. It is not considered part of the IEEE 488.1 Status Byte.

## 2.2 Standard Event Status Register

The ESR can be read by the `*ESR?` command. Reading this register, or the `*CLS` command will clear the ESR.

Bits in the ESR will be set only when the corresponding bit in the Standard Events Status Enable register is set. Use the `*ESE` to set bits, and the `*ESE?` to read this register.

To configure the SW Series to generate SRQ's based on the ESR, both the Standard Event Status Enable register and the Service Request Enable registers must be programmed.

Bit	Hex Value	Description
0	01	Operation Complete
1	02	Request Control – Not used
2	04	Query Error
3	08	Device Dependent Error
4	10	Execution Error (e.g., range error)
5	20	Command Error (e.g., syntax error)
6	40	User Request – Not used
7	80	Power On

**Operation Complete**

Set whenever the last command is completed and the SW is ready to accept another command, or when query results are available.

**Query Error**

Set when a query is made for which no response is available.

**Device Dependent Error**

Set for device specific errors. These errors are entered in the System Error Queue and have error codes greater than 0. See Section 5 for error descriptions.

**Execution Error**

Set when a parameter exceeds its allowed range.

**Command Error**

Set for a syntax error

**Power On**

Set once at power-up. The Status Byte ESR bit is not set.

## 2.3 Operation Status and Questionable Status Registers

The Operation Status and Questionable Status registers will always return 0 when queried. The Operation Status Enable and Questionable Status Enable registers can be programmed and queried to allow SCPI compatibility, but have no effect on the Operation Status and Questionable Status registers.

## 2.4 Error/Event Queue

The SW Series maintains an Error/Event Queue as defined by SCPI. The queue holds up to 10 errors and events. It is queried using the `SYSTEM:ERROR?` command, which reads in a first in, first out manner. The read operation removed the entry from the queue. The `*CLS` command will clear all entries from the queue.

## 2.5 Serial Poll Operation

Performing a serial poll will not modify the Status Byte other than to clear the RQS (bit 6) for a SW requesting service. Queries affecting the status registers and subsequent serial polls are listed below:

- `*STB?` clears the Status Byte
- `*ESR?` clears the ESR and bit 5 of the Status Register
- `SYSTEM:ERROR?` will clear bit 2 of the Status Register if the queue is empty

This page intentionally left blank.

## SECTION 3

# STANDARD WAVEFORMS AND SEQUENCES

The following standard waveforms are provided in the SW Series:

Name	Description	Scale Factor
Clip0	A sine wave with the positive halfcycle clipped at 0V from 50 to 130 degrees.	1.0
ClipPos	A sine wave with the positive halfcycle clipped from 50 to 130 degrees	1.0
ClipNeg	A sine wave with the positive halfcycle set to a 50% negative value from 50 to 130 degrees	1.0
DC+	Used to operate the SW as a positive DC source (true rms)	0.7071
DC-	Used to operate the SW as a negative DC source (true rms)	0.7071
DcRip03+	+DC with 3% ripple. The frequency of the ripple is determined by the programmed output frequency. (true rms)	0.7071
DcRip03-	-DC with 3% ripple. The frequency of the ripple is determined by the programmed output frequency. (true rms)	0.7071
DcRip10+	+DC with 10% ripple. The frequency of the ripple is determined by the programmed output frequency. (true rms)	0.7071
DcRip10-	-DC with 10% ripple. The frequency of the ripple is determined by the programmed output frequency. (true rms)	0.7071
Drop45	Sine with a drop-out above 0V from 45 - 60 degrees.	1.0
DropNeg	Sine with a drop-out to -Vpeak from 45 to 50 degrees	1.0
DropPeak	Sine wave with drop-out to 0V from 80 to 110 degrees.	1.0
FlatTp05	Flat-top sine wave with 5% distortion (true rms)	0.9344
FlatTp10	Flat-top sine wave with 10% distortion (true rms)	0.8894
FlatTp15	Flat-top sine wave with 15% distortion (true rms)	0.8545
FlatTp20	Flat-top sine wave with 20% distortion (true rms)	0.8251
Four3	Fourier square wave with 1 and 3rd harmonics (true rms)	0.8946
Four5	Fourier square wave with 1, 3, and 5th harmonics (true rms)	0.8703

Name	Description	Scale Factor
Four7	Fourier square wave with 1, 3, 5, and 7th harmonics (true rms)	0.8595
Four9	Fourier square wave with 1, 3, 5, 7, and 9th harmonics (true rms)	0.8537
FulRect+	Positive DC full rectified sine wave	1.0
FulRect-	Negative DC full rectified sine wave	1.0
HalfCyc	Sine wave with 90 - 180 degrees at 0Volts	1.0
HlfRect+	Positive DC half rectified sine wave	1.0
HlfRect-	Negative DC Half rectified sine wave	1.0
Inrush	Sine wave with first quarter (0-90 degrees) at 0Volts	1.0
Noisz010	Sine with 10% noise at zero crossings (true rms)	1.0
Noisz100	Sine with 100% noise at zero crossings (true rms)	1.0
Pcntl110	Phase control at 110 degrees	1.0
Pcntl170	Phase control at 170 degrees	1.0
Scratch	Waveform scratchpad. This is the waveform edit work area. Waveforms being edited can be output for testing purposed before they are saved to non-volatile memory.	1.0
Sine	Sine wave (true rms)	1.0
SinDc01+	Sine wave with +1% DC offset	1.0
SinDc01-	Sine wave with - 1% DC offset	1.0
SinDc10+	Sine wave with +10% DC offset	1.0
SinDc10-	Sine wave with - 10% DC offset	1.0
SinDc20+	Sine wave with +20% DC offset	1.0
SinDc20-	Sine wave with - 20% DC offset	1.0
SinDc50+	Sine wave with +50% DC offset	1.0
SinDc50-	Sine wave with - 50% DC offset	1.0
Spike200	Sine with spikes at peaks. Spikes are from 85 - 95 degrees and calibrated to be 200Vpeak when programmed to 120Vrms	1.1785
Spike250	Sine with spikes at peaks. Spikes are from 85 - 95 degrees and calibrated to be 250V peak when programmed to 120Vrms.	1.4731
Spike300	Sine with spikes at peaks. Spikes are from 85 - 95 degrees and calibrated to be 300Vpeak when programmed to 120Vrms	1.7678
Spike400	Sine with spikes at peaks. Spikes are from 85 - 95 degrees and calibrated to be 400Vpeak when programmed to 120Vrms	2.3570
Square	Square wave with 50% duty cycle (true rms)	0.7071
Step_06	Six voltage step sine wave	1.0
Step_12	Twelve voltage step sine wave	1.0
Step_24	Twenty-four voltage step sine wave	1.0
Taylor	5th harmonic Taylor series wave (true rms)	0.9136
Triangle	Triangle wave (true rms)	1.2246

The following standard sequences are provided in the SW Series:

<b>Name</b>	<b>Description</b>
M704Fsag	Mil-704 frequency sag
M704Fsg	Mil-704 frequency surge
M704Vsag	Mil-704 voltage sag
M704Vsg	Mil-704 voltage surge
TEST1	Elgar voltage test sequence

This page intentionally left blank.

## SECTION 4

# ERROR CODES

### 4.1 Error Codes Returned by SYSTEM:ERRor? Query

The following error codes are defined in the SCPI 1993.0 specification, and are supported by the SW Series. Error codes are in the range of [-32768, 32767]. SCPI reserves the negative error codes and 0, while error codes greater than 0 are device specific errors.

### 4.2 SCPI Error Codes

#### **0, No error**

The error queue is empty.

#### **-102, Syntax error**

An unrecognized command or data type was encountered.

#### **-200, Execution error**

An error/event number in the range [-299, -200] indicates that an error has been detected by the instruments execution control block. The occurrence of any error in this class shall cause the execution error bit (bit 4) in the event status register to be set.

An execution error may be the result of:

- A <program data> element out of range, such as programming 200 volts in low (156 volt) range.
- A command could not be executed due to the current condition of the device, such as attempting to change ranges while the output relay is closed. The output relay must be opened first.

#### **-225, Out of memory**

There is not enough memory to perform the requested operation.

**-241, Hardware missing**

A legal command or query could not be executed because the option is not installed.

**-284, Program currently running**

A legal command or query could not be executed because a sequence is currently running.

**-292, Referenced name does not exist****-292, Referenced name already exists****-330, Self-test failed****-350, Queue overflow**

The error queue can contain up to 10 entries. If more than 10 error/event conditions are logged before the SYSTEM:ERRor? query, an overflow will occur; the last queue entry will be overwritten with error -350. When the queue overflows, the least recent error/events remain in the queue and the most recent errors/events are discarded.

## 4.3 Device Specific Error Codes

### 4.3.1 Fault Codes

**1, SHUTDOWN-F3:OV**

Overvoltage shutdown.

**2, SHUTDOWN-DC ERROR**

Shutdown due to DC detected at the output.

**3, ROV, RELAY OPEN**

Output relay opened by redundant over voltage circuit.

**4, SHUTDOWN-ROV**

Shutdown by redundant over voltage circuit.

**5, GND FAULT-RLY OPEN**

Output relay opened by ground fault detection.

**6, SHUTDOWN-GND FAULT**

Shutdown by ground fault detection.

**7, 48V LOW-RLY OPEN**

Output relay opened due to low housekeeping supply.

**8, SHUTDOWN-48V LOW**

Shutdown due to low housekeeping supply.

## 9, UNDER VOLTAGE

### **10, PLL\_FREQ\_RANGE**

This error can only occur in external clock/lock input mode, after lock has been established and the output relay closed. It indicates the clock/lock input frequency is more than +/- 10% of the target frequency. This fault causes the output relay to open.

### **11, F2:OV-OCCURRED**

Over voltage detected; DC buss voltage greater than 320V for 45 seconds. No action taken.

### **12, F2:OV-RLY OPEN**

Output relay opened due to over voltage condition.

### **13, SHUTDOWN-F2:OV**

Shutdown due to over voltage.

### **14, F4:OT-RLY OPEN**

Output relay opened due to over temperature.

### **15, SHUTDOWN-F4:OT**

Shutdown due to over temperature.

### **16, CURR LIMIT**

Current limit detected.

### **17, CURRENT LIMIT**

Output relay opened due to current limit.

### **18, RMS\_OV- OCCURRED**

Output RMS over-voltage detected.

### **19, RMS\_OV-RLY OPEN**

Output relay opened due to output RMS over-voltage.

### **20, ROC**

Redundant over current detected. Unit requires service.

### **21, SLAVE FAULT**

Slave fault occurred.

### **22, SLAVE FAULT-RLY OPEN**

Output relay opened due to slave fault.

**23, NOT USED****24, OUTPUT RELAY FAULT**

Output relay failed to open/close.

**25, OUT RLY FAILED-5SEC**

Output relay failed to open/close within 5 seconds.

**26, ROV: SENSE LEAD ERR**

Redundant over voltage circuit detected sense lead error.

**50, MAXIMUM FREQUENCY EXCEEDED**

When configured in the SYSTem:EXTernal:DIRect mode, the output frequency is monitored and the outputs will be programmed to 0V if the output frequency exceeds 5kHz.

**4.3.2 Memory Errors****100, ECDI EPROM Checksum error**

A power-up or self-test checksum failure has occurred.

**101, ECDI Non-Volatile RAM Bank 0 checksum error**

A power-up or self-test checksum failure has occurred.

**102, ECDI Non-Volatile RAM Bank 1 checksum error**

A power-up or self-test checksum failure has occurred.

**103, ECDI System RAM failure**

The volatile system RAM failed it's memory test.

**110, DWSB EPROM Checksum error**

A power-up or self-test checksum failure has occurred.

**111, DWSB Non-Volatile RAM checksum error**

A power-up or self-test checksum failure has occurred.

**112, DWSB System RAM failure**

The volatile system RAM failed it's memory test.

**120, TEST BOARD EPROM Checksum error**

A power-up or self-test checksum failure has occurred.

**121, TEST BOARD System RAM failure**

The volatile system RAM failed it's memory test.

#### 4.3.3 Calibration Errors

**200, OUTPUT VOLTAGE CALIBRATION**

The output voltage calibration factor is out of range.

**201, OUTPUT CURRENT CALIBRATION**

The output current calibration factor is out of range.

#### 4.3.4 Sequence Errors

**300, SEQUENCE RAMP TIME TOO SHORT**

An attempt to create a ramp in a sequence failed because the ramp time was less than one cycle of the longest period.

#### 4.3.5 Calibration Errors

**400, WAVEFORM IN USE**

An attempt to delete a waveform or change a scale factor failed because the waveform was in use by an output phase.

This page intentionally left blank.

## SECTION 5

# SAMPLE PROGRAMS

### 5.1 Introduction

This section provides examples to demonstrate how to program the output voltage, current limit, frequency, phase angle, and function (waveform).

The number of output phases is dependent on the particular model: 5250 (3 phase), 3500 (2 phase), or 1750 (1 phase). The examples list only the SCPI commands; the code required to send the commands is dependent on the type of language you are using (e.g., C or Basic) and GPIB hardware (e.g., National Instruments or CEC).

SCPI commands have both long and short forms. The short form consists of the first four characters of the long form unless the fourth character is a vowel, in which case the short form is the first three character of the long form.

A valid command is either the long or short form - nothing in between. For example, SOURce can be used as SOUR or SOURCE. SOURC is a syntax error. All SCPI commands are case insensitive. Source, source, and SOURCE are equivalent.

## 5.2 Output Programming

### 5.2.1 Phase A Programming

Objective: Program phase A to 120V, 60Hz Sine wave (Factory default settings are 60 Hz Sine wave, but this example will program them anyway). Note that the SOURce[n] command (where “n” represents a particular output phase) default is n = 1, or phase A.

```
Source:voltage 120      // program output to 120 volts
Source:current 2.5     // program current limit to 2.5 in foldback mode
Source:frequency 60    // program output frequency
Source:phase 0        // program output phase angle to 0 degrees
Source:function "Sine" // program output waveform. Waveform names are case
                      // sensitive.
Output on             // close output relay. Valid parameters are on, off, 0, 1
<delay 1s>           // delay 1 second
```

The particular output phase could be included, in which case Source would be replaced with Source1. The Output command closes the output relays for all three phases, so a phase number is not allowed; “Output1” would generate a syntax error.

```
Source1:voltage 120
Source1:current 2.5
Source1:frequency 60
Source1:phase 0
Source1:function "Sine"
Output on
<delay 1s>           // delay 1 second
```

### 5.2.2 Simple Three Phase AC Programming

Objective: Program phase A, B, and C to the following values:

Phase	Amplitude	Current	Frequency	Phase Angle	Function
A	115.0	5.0	60.0	0	Sine
B	115.0	5.0	60.0	120	Sine
C	115.0	5.0	60.0	240	Sine

The following code will program all parameters individually on all three phases. The output frequency is only programmed for Phase A. All phases must run at the same output frequency. Changing the frequency of any Phase will change the frequency of all Phases.

The SW can accept only positive numbers when programming phase angle offset. Because phase angle offset represents a phase lead with respect to the internal reference, the formula below should be used for a selected phase to represent phase angle in terms of phase lag:

$$\text{Desired Phase Angle Lag} = 360^\circ - (\text{Ph.B } \theta \text{ Angle} - \text{Ph.A } \theta \text{ Angle})$$

The examples below are for a two-phase quadrature system:

- For phase angles Ph.A = 0°, Ph.B = 270°, phase lag would be 90°.  
90° = 360° - (270° - 0°)
- For phase angles Ph.A = 45°, Ph.B = 120°, phase lag would be 285°.  
285° = 360° - (120° - 45°)

```
Source1:voltage 115.0      // Phase A
Source1:current 5.0
Source1:frequency 60      // only need to program for one phase
Source1:phase 0
Source1:function "Sine"
```

```
Source2:voltage 115.0      // Phase B
Source2:current 5.0
Source2:phase 120
Source2:function "Sine"
```

```
Source3:voltage 115.0      // Phase C
Source3:current 5.0
Source3:phase 240
Source3:function "Sine"
```

Alternatively, the SOURce[n] command could be used with n = 0, which would program all three phases to the same value.

```
Source0:voltage 115.0      // program Phase A, B, and C to 115 V
Source0:current 5.0        // program Phase A, B, and C to 5.0 A
Source0:frequency 60      // program Phase A, B, and C to 60 Hz
Source0:function "Sine"    // program Phase A, B, and C to Sine waveform
```

```
Source1:phase 0           // Since the phase angles for A, B, and C are
Source2:phase 120         // different, they are programmed individually.
Source3:phase 240
```

## 5.2.3 Three Phase Programming with AC and DC Output

Objective: Program phase A, B, and C to the following values:

Phase	Amplitude	Current	Frequency	Phase Angle	Function
A	115.0 VAC	2.5	400.0	0	Sine
B	28.0 VDC	1.0	400.0	0	DC+
C	80.0 VAC	5.0	400.0	0	Square

Note that Phase A is AC, Phase B is DC, and Phase C is a Square Wave.

```

Output OFF // Open output relay so we can change coupling
<delay 1s> // delay 1 second
Output:Coupling DC // Allow both AC and DC waveforms.

Source1:voltage 115.0 // Phase A
Source1:current 2.5
Source1:frequency 400 // Program frequency of all three phases to 400
Source0:phase 0 // Program phase angle of all phases to 0
Source1:function "Sine"

Source2:function "DC+" // Phase B. The DC+ waveform is set first,
Source2:voltage 28.0 // because a change from AC to DC or DC to
Source2:current 1.0 // AC will zero the voltage value as a safety
// mechanism.

Source3:voltage 80.0 // Phase C
Source3:current 5.0
Source3:function "Square"

Output ON
<delay 1s> // delay 1 second

```

### 5.3 Measurements

Objective: Program phase A, B, and C to the following values:

Phase	Amplitude	Current	Frequency	Phase Angle	Function
A	230.0 Vac	5.0	50	0	Sine
B	230.0 Vac	5.0	50	120	Sine
C	230.0 Vac	5.0	50	240	Sine

```
Output OFF           // open output relay so we can change coupling
<delay 1s>          // delay 1 second
```

```
Source:volt:range 1 // configure for 312V high range
```

```
Source0:voltage 230.0 // program Phase A, B, and C to 230V
Source0:current 5.0   // program Phase A, B, and C to 5.0A
Source0:frequency 50  // program Phase A, B, and C to 50 Hz
Source0:function "Sine" // program Phase A, B, and C to Sine waveform
```

```
Output ON           // close output relay
<delay 1s>         // allow time to stabilize
```

```
Measure1:volt?     // initiate measurement of phase A voltage
<read device>     // read result
Measure2:volt?     // initiate measurement of phase B voltage
<read device>     // read result
Measure3:volt?     // initiate measurement of phase C voltage
<read device>     // read result
```

```
Measure1:curr?    // initiate measurement of phase A current
<read device>    // read result
Measure2:curr?    // initiate measurement of phase A current
<read device>    // read result
Measure3:curr?    // initiate measurement of phase A current
<read device>    // read result
```

```
Measure1:freq?    // initiate measurement of phase A frequency
<read device>    // read result
Measure1:freq?    // initiate measurement of phase B frequency
<read device>    // read result
Measure3:freq?    // initiate measurement of phase C frequency
<read device>    // read result
```

## 5.4 Editing Waveforms

A waveform is a record of 4095 x 4095 points. It has no output parameters associated with it, so it is independent of voltage, current, frequency, or phase angle; these parameters appear in the Edit Subsystem for the sole purpose of making the editing easier to visualize.

### 5.4.1 Modify a Sine Wave

Objective: Edit the factory supplied Sine waveform to create a voltage drop at 90 degrees to 180 degrees, and save the new waveform.

```
Output OFF // disconnect any load for safety
<delay 1s>

MMemory:load:waveform "Sine" // load Sine wave into waveform scratchpad

Edit:waveform:vrms 156.0 // amplitude is not important for this example, so
// set RMS value of Sine wave to maximum
Edit:waveform:freq 60.0 // frequency is not important because we know
// the start and stop phase angle
Edit:waveform:phase:start 90.0 // starting phase angle
Edit:waveform:phase:stop 180.0 // stop phase angle
Edit:waveform:ampl 15.0 // set absolute voltage level between 90 and 180 to
// 15 volts.

MMemory:store:waveform "Examp1" // store new waveform

Output:coupling DC // the new waveform now contains a net DC voltage

Source:func "Examp1" // set phase A waveform to Examp1
Source:volt 100 // at this point, the waveform can be viewed at the
// front panel bnc connector before the output
// relay is closed and the waveform appears at the
// output.

Output ON // close output relay
<delay 1s>
```

## 5.4.2 Modify a Sine Wave for a Specified Time

Objective: Edit the factory supplied Sine waveform to create a voltage spike at 45 degrees for 2ms, and save the new waveform. In this example, the frequency field is important because it affects the stop angle.

```
Output OFF // disconnect any load for safety
<delay 1s>

MMemory:load:waveform "Sine" // load Sine wave into waveform scratchpad

Edit:waveform:vrms 115.0 // amplitude is not important for this example, so
// set RMS value of Sine wave to maximum
Edit:waveform:freq 60.0 // frequency is not important because we know
// the start and stop phase angle
Edit:waveform:phase:start 45.0 // starting phase angle
Edit:waveform:duration 2ms // stop phase angle calculated based on start angle
// duration
Edit:waveform:ampl 200.0 // set absolute voltage level between 45 and 88.2
// degrees to 200 volts.

MMemory:store:waveform "Examp1" // Store new waveform. If an "Examp1"
// waveform already exists, delete it with
// Mmemory:delete:waveform "Examp1".

Output:coupling DC // the new waveform now contains a net DC voltage

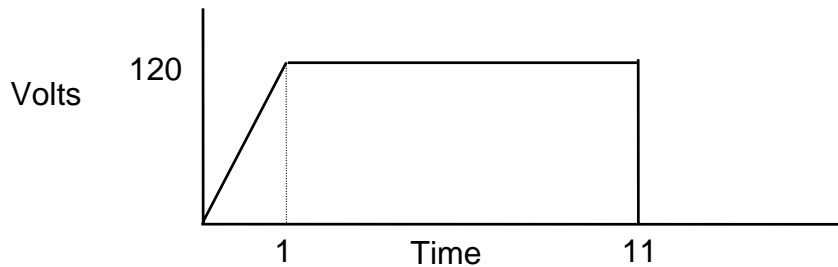
Source:func "Examp1" // set phase A waveform to Examp1
Source:volt 100 // at this point, the waveform can be viewed at the
// front panel bnc connector before the output
// relay is closed and the waveform appears at the
// output.

Output ON // close output relay
<delay 1s>
```

## 5.5 Sequence Examples

### 5.5.1 Voltage Ramp for a Specified Time Period

Objective: Create a sequence to ramp phase A voltage from 0 to 120V in 1 second, stay at 120V for 10 seconds, then go to zero.



Starting point:

Phase	Amplitude	Current	Frequency	Phase Angle	Function
A	0	5.0	60	0	Sine

Sequence:

Segment	Phase	Freq	Cycles	Time	Ampl	Ramp	Function
0	A	60	—	1s	0	Yes	Sine
1	A	60	—	10s	120	No	Sine

An important point to remember is that a ramp must always have a beginning and end segment value. This means that the smallest ramp sequence possible consists of two segments, and the last segment in a sequence can never be a ramp segment.

When creating a new sequence, use the `Edit:sequence:clear` command to erase any old sequence data in the sequence scratchpad. Only segment 0 will be available with all parameters set to default:

Parameter	Default Value
Freq	0
Freq Ramp	0
Cycles	0
Time	0
Ampl	0 for all phases
Ampl Ramp	Off for all phases
Func	Sine for all phases
$\theta$ Ang	0, 120, 240 for phase A, B, C

```
// ----- Create Segment 0 -----
Edit:seq:clear           // Clear sequence scratchpad. Take advantage
                        // of defaults and only set changed params
Edit:seq:dur 1          // set segment time to 1 sec
Edit:seq:sour1:ampl:ramp on // set phase A amplitude ramp on

// ----- Create Segment 1 -----
Edit:seq:insert 1       // Create segment 1. Segment pointer is
                        // automatically incremented to seg 1.
Edit:seq:dur 10         // set segment time to 10 sec
Edit:seq:sour1:ampl 120 // set phase A amplitude to 120V

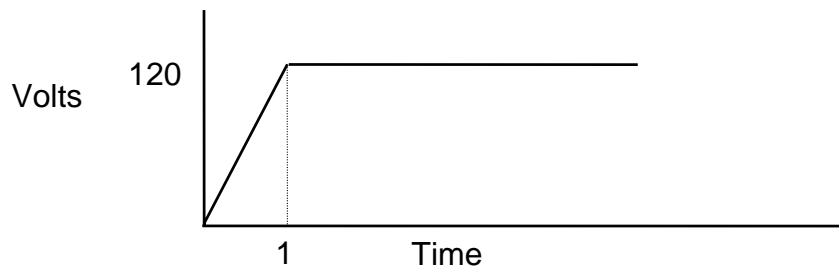
// At this point, the sequence exists only in the sequence scratchpad area. It
// can be saved with a name and executed, or executed directly from the
// scratchpad. In this example, it will be run from the scratchpad.

//----- setup sequence execution parameters -----
Source:seq:mode:run single // execute sequence once and stop
Source:seq:mode:stop zero  // set output voltage to 0 when seq stops
Source:seq:load "SCRATCH"  // load seq scratchpad

Output on                // close output relay
Source:seq run           // execute sequence
```

## 5.5.2 Voltage Ramp to a Value

Objective: Create a sequence to ramp phase A voltage from 0 to 120V in 1 second. The sequence will terminate, but the voltage will remain at 120V. To accomplish this, the sequence run mode will be set to SINGLE, and the sequence stop mode will be set to SEGMENT to cause the output voltage to remain at the value programmed by the last segment in the sequence.



Starting point:

Phase	Amplitude	Current	Frequency	Phase Angle	Function
A	0	5.0	60	0	Sine

Sequence:

Segment	Phase	Freq	Cycles	Time	Ampl	Ramp	Function
0	A	60	—	1s	0	Yes	Sine
1	A	60	1	—	120	No	Sine

Notice that in segment 1 the Cycle field is set to one, and the Time field is not used. Since a ramp always requires two segments, the ending segment has been set to the smallest possible time period: 1 cycle of the output frequency. At 60Hz, this would be 16.67ms.

```
// ----- Create Segment 0 -----
Edit:seq:clear // Clear sequence scratchpad. Take advantage
                // of defaults and only set changed params
Edit:seq:dur 1 // set segment time to 1 sec
Edit:seq:sour1:ampl:ramp on // set phase A amplitude ramp on

// ----- Create Segment 1 -----
Edit:seq:insert 1 // Create segment 1. Segment pointer is
                  // automatically incremented to seg 1.
Edit:seq:cycles 1 // set segment time to 1 cycle
Edit:seq:sour1:ampl 120 // set phase A amplitude to 120V

// At this point, the sequence exists only in the sequence scratchpad area. It
// can be saved with a name and executed, or executed directly from the
// scratchpad. In this example, it will be run from the scratchpad.

//----- setup sequence execution parameters -----
Source:seq:mode:run single // execute sequence once and stop
Source:seq:mode:stop segment // set output voltage to value in last segment
Source:seq:load "SCRATCH" // load seq scratchpad

Output on // close output relay
Source:seq run // execute sequence
```

## 5.5.3 Insert a Waveform for One Cycle

Objective: Cause a sine waveform with the positive halfcycle clipped at 0V from 50 to 130 degrees to appear at the output for one cycle without output power interruption. The Trigger output signal will be configured to send out a pulse when the waveform appears at the output.

A sequence can be created that can be executed while the output is at a specified voltage. The sequence run mode will be set to SINGLE, and the sequence stop mode will be set to PROGRAM to cause the output voltage to return to the value programmed before the sequence was run.

Starting point:

Phase	Amplitude	Current	Frequency	Phase Angle	Function
A	120	5.0	60	0	Sine

Sequence:

Segment	Phase	Freq	Cycles	Time	Ampl	Ramp	Function
0	A	60	1	—	120	No	Sine

```
// ----- Create Segment 0 -----
Edit:seq:clear           // Clear sequence scratchpad. Take advantage
                        // of defaults and only set changed params
Edit:seq:cycles 1       // set segment time to 1 cycle
Edit:seq:sour1:ampl 120 // set phase A amplitude
Edit:seq:sour1:func "Clip0" // set phase A waveform to Clip0 factory waveform
Edit:seq:sync on       // generate sync pulse at the beginning of this segment

System:sync 0          // disable sync signal when a sequence is not running

//----- setup sequence execution parameters -----
Source:seq:mode:run single // execute sequence once and stop
Source:seq:mode:stop program // set output voltage to value before seq began
Source:seq:load "SCRATCH" // load seq scratchpad

Output on              // close output relay
<delay 1s>

Source:volt 120        // program phase A voltage
<delay 1s>            // wait for output to stabilize

Source:seq run         // execute sequence
```

## 5.6 GPIB Programs

### 5.6.1 Uploading ASCII Waveform Files to the SmartWave

```
/*-----
UPASCII.CPP - GPIB PROGRAM TO UPLOAD ASCII FILES TO ELGAR SW SERIES
```

This program will read an ASCII file of 16384 bytes (4096 waveform data points \* 4 ASCII characters per data point) and upload it to an Elgar SW 5250. The file name is specified on the command line. Note that all uploads are sent to the Waveform Scratchpad area. If multiple uploads are desired, each waveform must be saved from the waveform scratchpad to a user-defined memory location in the waveform library before the next waveform is uploaded.

Initial release: 8/22/94 by Tom Kenney

```
-----*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include "decl.h"          // National Instruments GPIB header

/*-----
main
-----*/
void main(int argc, char *argv[])
{
    char  FileName[20], DevName[20];
    char  *tempbuf;
    FILE  *fp;
    int   DevId;

    //----- initialize global variables -----
    strcpy(DevName, "SW5250");                // device name

    //----- process command line arguments -----
    if (argc > 1)
        strcpy(FileName, argv[1]);
    else
    {
        printf("Usage: upascii <filename>\n");
        exit(0);
    }

    //----- open file to download -----
    if ((fp = fopen(FileName, "rt")) == NULL)
    {
        printf("Error opening %s \n", FileName);
        return;
    }

    //----- allocate temporary storage -----
    if ((tempbuf = (char *)malloc(17000)) == NULL)
    {
        printf("Error allocating memory\n");
```

```
return;
}

//----- find gpib device -----
printf("STARTING PROGRAM \n");
if ((DevId = ibfind(DevName)) < 0) // get device handle
{
printf("Could not find device %s \n", DevName);
exit(1);
}

//----- identify device to confirm communications -----
ibwrt(DevId, "*idn?", 5); // query id
ibrd(DevId, tempbuf, 100); // read string
tempbuf[ibcnt] = '\0'; // terminate string
printf("Device id: %s \n", tempbuf); // display

//----- send upload commands -----
ibwrt(DevId, "MMEM:FORMAT 0", 13); // ascii format
ibwrt(DevId, "MMEM:UPLOAD:WAVEFORM", 20); // setup command

//----- read file data and send to SW5250 -----
fread(tempbuf, 1, 16384, fp); // read all wfrm points
ibwrt(DevId, tempbuf, 16384); // send wfrm data

//----- close input file and free memory -----
fclose(fp);
free(tempbuf); // free storage buffer
printf("Sent ASCII waveform \n");
}
```

## 5.6.2 Downloading ASCII Waveform Files from the SmartWave

```
/*-----
DNASCII.CPP - GPIB PROGRAM TO DOWNLOAD ASCII FILE FROM AN ELGAR SW5250
```

A waveform may be downloaded from the Waveform Scratchpad in the SW5250. This program will read an ASCII file of 16384 bytes (4096 waveform data points \* 4 ASCII characters per data point) from the SW5250 and store it in a file specified on the command line. Note that all downloads originate from the SW5250 waveform scratchpad area. Therefore, a waveform must be loaded into the waveform scratchpad before this program execution. This can be accomplished either from the front panel or the MMEMory:LOAD:WAVEform GPIB command. In this example, the desired waveform is loaded into the waveform scratchpad with a GPIB command before the download takes place.

Initial release: 8/22/94 by Tom Kenney

```
-----*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include "decl.h"          // National Instruments GPIB header

/*-----
main
-----*/
void main(int argc, char *argv[])
{
    char  DevName[20], WaveName[20], FileName[20];
    char  *tempbuf;
    FILE  *fp;
    int   DevId;

    //----- initialize global variables -----
    strcpy(DevName, "SW5250");          // device name

    //----- process command line arguments -----
    if (argc > 2)
    {
        strcpy(WaveName, argv[1]);
        strcpy(FileName, argv[2]);
    }
    else
    {
        printf("Usage: dnascii <Waveform name> <File name>\n");
        exit(0);
    }

    //----- open file to store waveform -----
    if ((fp = fopen(FileName, "wt")) == NULL)
    {
        printf("Error opening %s \n", FileName);
        return;
    }

    //----- allocate temporary storage -----
```

```
if ((tempbuf = (char *)malloc(17000)) == NULL)
{
printf("Error allocating memory\n");
return;
}

//----- find gpib device -----
printf("STARTING PROGRAM \n");
if ((DevId = ibfind(DevName)) < 0)           // get device handle
{
printf("Could not find device %s \n", DevName);
exit(1);
}

//----- identify device to confirm communications -----
ibwrt(DevId, "*idn?", 5);                    // query id
ibrd(DevId, tempbuf, 100);                  // read string
tempbuf[ibcnt] = '\0';                      // terminate string
printf("Device id: %s \n", tempbuf);        // display

//----- send download commands -----
ibwrt(DevId, "MMEM:FORMAT 0", 13);          // ascii format
sprintf(tempbuf, "MMEM:LOAD:WAVEFORM \"%s\"", WaveName);
ibwrt(DevId, tempbuf, strlen(tempbuf));     // load scratchpad
ibwrt(DevId, "MMEM:DOWNLOAD:WAVEFORM", 22); // setup command

//----- read waveform data and write to file -----
ibrd(DevId, tempbuf, 16400);
fwrite(tempbuf, 1, ibcnt, fp);

//----- close file and free memeory -----
fclose(fp);
free(tempbuf);                             // free storage buffer
printf("Received %d characters. Created file: %s\n", ibcnt, FileName);
}
```

## 5.6.3 Uploading Binary Waveform Files to the SmartWave

```
/*-----
UPBIN.CPP - GPIB PROGRAM TO UPLOAD BINARY FILES TO ELGAR SW SERIES
```

This program will read a binary file of 8192 bytes (4096 waveform data points \* 2 bytes per data point) and upload it to an Elgar SW 5250 in the binary file format mode. The file name is specified on the command line. Note that all uploads are sent to the Waveform Scratchpad area. If multiple uploads are desired, each waveform must be saved from the waveform scratchpad to a user-defined memory location in the waveform library before the next waveform is uploaded.

Initial release: 8/22/94 by Tom Kenney

```
-----*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include "decl.h"          // National Instruments GPIB header

short   Wfrm[4096];      // waveform storage

/*-----
main
-----*/
void main(int argc, char *argv[])
{
    char   FileName[20], DevName[7], tempbuf[80];
    FILE   *fp;
    int    DevId, i;

    //----- initialize global variables -----
    strcpy(DevName, "SW5250");                // device name

    //----- process command line arguments -----
    if (argc > 1)
        strcpy(FileName, argv[1]);
    else
    {
        printf("Usage: upbin <filename>\n");
        exit(0);
    }

    //----- find gpib device -----
    printf("STARTING PROGRAM \n");
    if ((DevId = ibfind(DevName)) < 0)        // get device handle
    {
        printf("Could not find device %s \n", DevName);
        exit(1);
    }

    //----- identify device to confirm communications -----
    ibwrt(DevId, "*idn?", 5);                 // query id
    ibrd(DevId, tempbuf, 100);                // read string
    tempbuf[ibcnt] = '\0';                    // terminate string
```

```
    printf("Device id: %s \n", tempbuf);    // display

//----- open file to download -----
if ((fp = fopen(fileName, "rb")) == NULL)
{
    printf("Error opening %s \n", fileName);
    return;
}

//----- send upload commands -----
ibwrt(DevId, "MMEM:FORMAT 1", 13);        // binary format
ibwrt(DevId, "MMEM:UPLOAD:WAVEFORM", 20); // setup command

//----- read file data and send to SW5250 -----
fread(Wfrm, sizeof(short), 4096, fp);    // read all wfrm points
    ibwrt(DevId, (char *)Wfrm, 8192);    // send wfrm data

//----- close input file and free memory -----
fclose(fp);
    printf("Sent binary waveform \n");
}
```

## 5.6.4 Downloading Binary Waveform Files from the SmartWave

```
/*-----
DNBIN.CPP - GPIB PROGRAM TO DOWNLOAD BINARY WAVEFORM FILES FROM AN
ELGAR SW5250
```

This program will read 8192 bytes (4096 waveform data points \* 2 bytes per data point) from an Elgar SW 5250 in the binary file format mode. The file is written to the file name specified on the command line. Note that all downloads originate from the SW5250 waveform scratchpad area. Therefore, a waveform must be loaded into the waveform scratchpad before this program execution. This can be accomplished either from the front panel or the MMEMory:LOAD:WAVEform GPIB command.

Initial release: 8/22/94 by Tom Kenney

```
-----*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include <ctype.h>
#include "decl.h"          // National Instruments GPIB header

//----- function prototypes -----
short swap_short(short  n);

//----- global variables -----
short  Wfrm[4096];        // waveform storage

/*-----
main
-----*/
void main(int argc, char *argv[])
{
    char  FileName[20], DevName[20], tempbuf[80];
    FILE  *fp;
    int   DevId, i;

    //----- initialize variables -----
    strcpy(DevName, "SW5250");                // device name

    //----- process command line arguments -----
    if (argc > 1)
        strcpy(FileName, argv[1]);
    else
    {
        printf("Usage: dnbin <filename>\n");
        exit(0);
    }

    //----- open file to download -----
    if ((fp = fopen(FileName, "wb")) == NULL)
    {
        printf("Error opening %s \n", FileName);
        return;
    }
}
```

```

//----- find gpib device -----
printf("STARTING PROGRAM \n");
if ((DevId = ibfind(DevName)) < 0)           // get device handle
{
    printf("Could not find device %s \n", DevName);
    exit(1);
}

//----- identify device to confirm communications -----
ibwrt(DevId, "*idn?", 5);                    // query id
ibrd(DevId, tempbuf, 80);                   // read string
tempbuf[ibcnt] = '\0';                      // terminate string
printf("Device id: %s \n", tempbuf);        // display

//----- send download commands -----
ibwrt(DevId, "MMEM:FORMAT 1", 13);          // binary format
ibwrt(DevId, "MMEM:DOWNLOAD:WAVEFORM", 22); // download scratchpad data

//----- read waveform binary data and write file -----
ibrd(DevId, (unsigned char *)Wfrm, 8200);   // get wfrm data
for (i = 0; i < 4096; i++)
    Wfrm[i] = swap_short(Wfrm[i]);
fwrite(Wfrm, sizeof(short), 4096, fp);     // write all wfrm points

//----- close input file -----
fclose(fp);
printf("Created file: %s \n", FileName);
}

/*-----
Swaps the upper and lower bytes of a short integer
-----*/
short swap_short(short n)
{
    union {
        unsigned char byte[2];
        short word;
    } number;

    unsigned char tempchar;

    number.word = n;
    tempchar = number.byte[0];
    number.byte[0] = number.byte[1];
    number.byte[1] = tempchar;

    return (number.word);
}

```