User Manual

Models SE5081 / SE5082

5 GS/s Single / Dual Channel Arbitrary Waveform Generators

Publication No. 011117

Rev. 1.1

Tabor Electronics Ltd.

Tabor Electronics Ltd.
P.O. Box 404, Tel Hanan Israel 20302
Tel: +972-4-821-3393, FAX: +972-4-821-3388

PUBLICATION DATE: April 29, 2018

WARRANTY STATEMENT

Products sold by Tabor Electronics Ltd. are warranted to be free from defects in workmanship or materials. Tabor Electronics Ltd. will, at its option, either repair or replace any hardware products which prove to be defective during the warranty period. You are a valued customer. Our mission is to make any necessary repairs in a reliable and timely manner.

Duration of Warranty

The warranty period for this Tabor Electronics Ltd. hardware is five years, except software and firmware products designed for use with Tabor Electronics Ltd. Hardware is warranted not to fail to execute its programming instructions due to defect in materials or workmanship for a period of ninety (90) days from the date of delivery to the initial end user.

Return of Product

Authorization is required from Tabor Electronics before you send us your product for service or calibration. Call your nearest Tabor Electronics support facility. A list is located on the last page of this manual. If you are unsure where to call, contact Tabor Electronics Ltd. Tel Hanan, Israel at 972-4-821-3393 or via fax at 972-4-821-3388. We can be reached at: support@tabor.co.il

Limitation of Warranty

Tabor Electronics Ltd. shall be released from all obligations under this warranty in the event repairs or modifications are made by persons other than authorized Tabor Electronics service personnel or without the written consent of Tabor Electronics.

Tabor Electronics Ltd. expressly disclaims any liability to its customers, dealers and representatives and to users of its product, and to any other person or persons, for special or consequential damages of any kind and from any cause whatsoever arising out of or in any way connected with the manufacture, sale, handling, repair, maintenance, replacement or use of said products.

Representations and warranties made by any person including dealers and representatives of Tabor Electronics Ltd., which are inconsistent or in conflict with the terms of this warranty (including but not limited to the limitations of the liability of Tabor Electronics Ltd. as set forth above), shall not be binding upon Tabor Electronics Ltd. unless reduced to writing and approved by an officer of Tabor Electronics Ltd.

This document may contain flaws, omissions or typesetting errors. No warranty is granted nor liability assumed in relation thereto. The information contained herein is periodically updated and changes will be incorporated into subsequent editions. If you have encountered an error, please notify us at support@taborelec.com. All specifications are subject to change without prior notice.

Except as stated above, Tabor Electronics Ltd. makes no warranty, express or implied (either in fact or by operation of law), statutory or otherwise; and except to the extent stated above, Tabor Electronics Ltd. shall have no liability under any warranty, express or implied (either in fact or by operation of law), statutory or otherwise.

PROPRIETARY NOTICE

FOR YOUR SAFETY

Before undertaking any troubleshooting, maintenance or exploratory procedure, read carefully the **WARNINGS** and **CAUTION** notices.

This equipment contains voltage hazardous to human life and safety, and is capable of inflicting personal injury.

If this instrument is to be powered from the AC line (mains) through an autotransformer, ensure the common connector is connected to the neutral (earth pole) of the power supply.

Before operating the unit, ensure the conductor (green wire) is connected to the ground (earth) conductor of the power outlet. Do not use a two-conductor extension cord or a three-prong/two-prong adapter. This will defeat the protective feature of the third conductor in the power cord.

Maintenance and calibration procedures sometimes call for operation of the unit with power applied and protective covers removed. Read the procedures and heed warnings to avoid “live” circuits points.

Before operation this instrument:

1. Ensure the instrument is configured to operate on the voltage at the power source. See Installation Section.
2. Ensure the proper fuse is in place for the power source to operate.
3. Ensure all other devices connected to or in proximity to this instrument are properly grounded or connected to the protective third-wire earth ground.

If the instrument:

- fails to operate satisfactorily
- shows visible damage
- has been stored under unfavorable conditions
- has sustained stress

Do not operate until performance is checked by qualified personnel.

DECLARATION OF CONFORMITY

We: Tabor Electronics Ltd.
9 Hatasia Street, Tel Hanan
ISRAEL 3688809

declare, that the 5GS/s Single and Dual Channel Arbitrary Waveform Generators

Models SE5081 and SE5082

complies with the requirements of the Electro Magnetic Compatibility 2014/108/EU, class A and the Low Voltage Directive 2014/35/EU. Compliance was demonstrated to the following specifications as listed in the official Journal of the European Communities:

Safety:
IEC/EN 61010-1:2010

EMC:

IEC 61326-1:2013	Class A Radiated and Conducted Emission
EN 61000-4-2	ESD
EN 61000-4-3	Radiated Immunity
EN 61000-4-4	Burst/fast transients
EN 61000-4-5	Surge
EN 61000-4-6	Conducted Immunity
EN 61000-4-8	Power frequency magnetic field
EN 61000-4-11	Voltage dips and fluctuations
EN 55011	Radiated and conducted Emissions Class A

Models 5081 and 5082 are built on the same platform and share specifications and features except the number of channels. The tests were performed on a typical configuration.

Table of Contents

List of Figures

List of Tables

Chapter 1
Getting Started

Title **Page**

What's in This Chapter

This chapter contains a general description of the SE5082 **Waveform Generator**, including an overall functional description of the instrument. Additionally, the front and rear panel connectors and indicators are explained in detail.

NOTE

This manual is common to Models SE5081 and SE5082. Features and functions are described for the Model SE5082. For model SE5081, please ignore all references to the second channel in this manual.

Conventions Used in this Manual

The following icons may appear in this manual:

NOTE

A Note contains information relating to the use of this product

TIP

A Tip containing recommendation relating to the use of a feature

CAUTION

A Caution contains information that should be followed to avoid personal damage to the instrument or the equipment connected to it.

WARNING

A Warning alerts you to a potential hazard. Failure to adhere to the statement in a WARNING message could result in personal injury.

Safety Symbols

The following safety symbols may appear in this manual and on this product:

Alternating current

Standby supply. Unit is not completely disconnected from ac mains when switch is off.

SE5081/2 Feature Highlights

- 5Gs/s (10GS/s equivalent in RF mode), 12 bits single or dual channel waveform generators
- Directly generate RF signals up to X-band
- Extremely fast rise and fall time of under 85ps
- Multi-Nyquist zone operation capability, up to the 4th Nyquist zone
- Selectable sampling modes for optimizing performance depending on the required Nyquist zone
- Independent or synchronized channels with 10ps inter channel skew resolution
- Up to 64M of waveform memory
- Various output amplifier modules utilized to solve numerous applications in different domains
- Smart trigger enables trigger hold-off, detect, wait, abort and restart
- Advanced sequencer for step, loop, nest and jumps scenarios
- Built-in fast dynamic segments and sequences hop control
- Two programmable markers (positions, width and levels) per channel
- Multi instrument synchronization

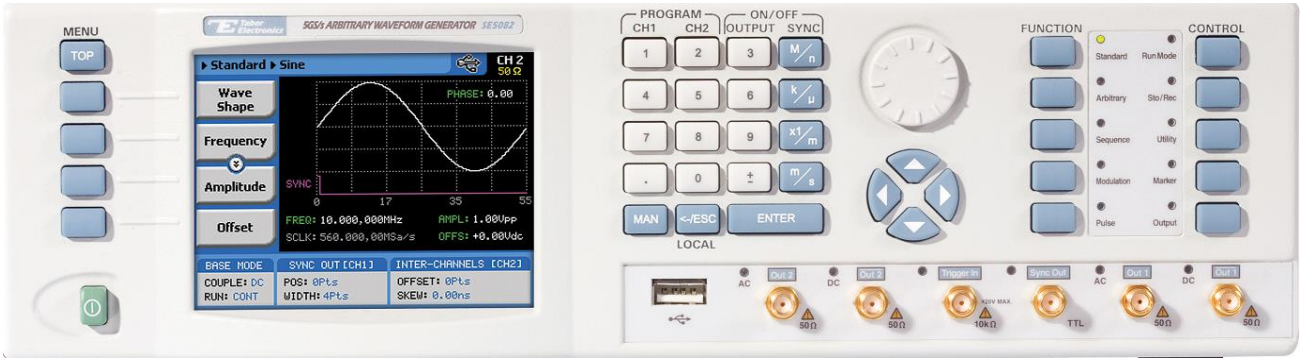

Figure 1-1, Model SE5082

Introduction

Detailed functional description is provided, following the general description of the features and functions available with the SE5082**.**

General

The SE5082 is a dual-channel universal waveform source. In addition to its standard ability to generate arbitrary shapes, functions or modulation, it can also be used as a full-featured pulse generator, as well as a wireless signal source. The SE5082 offers unmatched performance, even when compared to instruments designed to generate fewer types of signals. Its affordable footprint saves space and cost without compromising bandwidth and signal integrity. Cutting-edge technology is utilized to assure that this will be the only source needed for many years to come.

Multi Nyquist Zones Operation

All AWGs use Digital to Analog Convertors (DAC) in order to generate the required signals. However, very few AWGs offer the ability to generate signals beyond the first Nyquist zone and are therefore limited in bandwidth to half the DAC sampling rate. The SE5082 utilizes a new ground breaking DAC that provides an analog bandwidth that extends well beyond the first Nyquist zone of 2.5GHz and well into the third and even fourth Nyquist zones at frequencies up to 7GHz, thus enabling direct generation of RF signals in the S, C and X-bands. The DAC incorporates four sampling modes which enable optimizing the output power across multiple Nyquist zones. Together with the proper band pass filter and amplifier, the user can use the images from higher Nyquist zones to generate signals well beyond the DAC sampling frequency.

Two Synchronized Channels

The SE5082 has two output channels, both of which can operate either independently, or synchronized to share the same sample clock source. As two separate channels, one has the advantage of having two separate instruments in one box, with each having the ability to be programmed to output different function shapes, amplitude levels and/or to operate in different run modes.

Alternatively, the advantage of having two synchronized channels is very significant in applications that require an accurate and controlled phase between the two channels. Many applications require X-Y modes and I&Q outputs and the SE5082 is ideal for this usage.

Stable and Accurate Output Signals

As standard, the instrument is equipped with a frequency reference that has 1ppm accuracy and stability over a period of 1 year. An external frequency reference input is provided on the rear panel for applications requiring greater accuracy and stability.

Signal Integrity

As technology evolves and new devices are continuously developed, faster and more complex signals are needed to simulate and stimulate these new devices. The SE5082 incorporates a unique DAC technology that offers up to 5GS/s sampling rate with excellent spectral purity. The 12 bits, 7GHz wide bandwidth DAC, offers the ability to work in higher Nyquist zones and offer solutions to applications in the microwave frequencies.

Configurable Output Modules

The SE5082 offers several output amplifier modules. The default configuration is that with a direct DAC output module which offers an AC coupled output that reaches up to 540mVpp (single ended) and is available for applications requiring the highest bandwidth and best spectral purity. An HV output module is available for applications that require a higher voltage level, it reaches up to 2Vp-p but is limited to 600MHz bandwidth. The DC output module is available for applications that require DC coupled path and offers a very fast rise time of <120ps and reaches up to 1.2Vpp.

Versatile Run Modes

Having a waveform generator that has such an enormous waveform creation capability without the ability to control how waveforms start and stop is unlikely to perform as desired in a real-life environment. For this purpose, the SE5082 is equipped with a highly sophisticated mechanism that allows precise control over the start and stop timing, which is exactly what is required for interactive systems. When the instrument is operated on the bench, most applications require that the output is generated continuously or triggered from an external source, but as a system component and as part of a large array of other equipment, the waveform can be halted for a specified time, then operated for another period of time and stopped when it is no longer required. The enable/disable sequence is available when the SE5082 is placed in armed remote operation only and the start and stop commands can be programmed to come through different inputs.

Besides armed operation, the waveform generator responds to various trigger sources such as: external trigger signal from the trigger input, button press on the front panel, external signal which is applied through the event input and finally, if none of the external stimuli are available, an internal trigger generator with a programmable trigger period can be utilized to generate repeated

triggers.

As well as the above-described trigger facilities, the SE5082 has two additional trigger modes that enhance its performance. The first is an override mode, where each trigger stops the running waveform and overrides the output to start over, and the second is a re-trigger mode, where the stop of the waveform triggers the next start after a predefined and programmable interval. While the first mode assures that every trigger starts a new cycle, the second mode assures that no new cycle is initiated before the previous cycle has been completed and a pre-programmed interval has lapsed before the next cycle is started.

Besides its trigger facilities, the SE5082 can be gated to generate waveforms only when a specific level of the trigger input is reached. It also can be programmed to generate a counted burst of waveform cycles upon request, whether it comes from a trigger signal or from a remote command.

The SE5082 also employs a delay circuit, where an external event triggers a delay which only after its programmed interval, the output is allowed to start generating waveforms.

Finally, to make sure no trigger slips away; the SE5082 has a smart trigger detection that inhibits triggers if they are below or above a certain trigger pulse width and also holds off all triggers that are considered false, before the true signal appears and does the work.

Arbitrary Waveforms and Memory Segmentation

Waveform memory is the internal scratchpad where the waveforms reside. Larger memory banks provide for longer waveforms. One can use the entire memory for a single waveform or split the length to smaller segments. In this case, many waveforms can be stored in the same memory and replayed, one at a time, when recalled to the output. The memory segmentation feature may be combined with a sequence generator that can take different memory segments and link and loop them in any order as required for the specific application or test. The ability to loop waveform segments in a sequence can save a lot of memory and thus extend the capability of the generator to produce longer, more complex waveforms. Each of the SE5082 channels can be programmed to have a unique setting of sequence steps and loops.

Dynamic Segment / Sequence Control

A control input is available on the SE5082 rear panel that provides control over the replay of a specific memory segment. The control connector has 9 pins and hence allows replay of one of 256 waveform segments. A valid line must be asserted to validate segment change. Having the dynamic control feature, in effect, can serve as replacement of the sequence table where the real-time application can decide when and for how long a waveform will be

generated. For much more complex applications, this same input may serve as a dynamic switch for complete sequences and thus create real-life scenarios for real-time applications.

Extended Memory Option 1

The SE5082 comes with a standard 32M arbitrary waveform memory. However, the instrument can be ordered with an extended memory configuration, option 1, having 64M of waveform memory per channel. This specific model configuration is not field upgradable and must be ordered as such from the factory. Besides increasing the amount of available memory, this option also offers better trigger, sync output, inter channel skew and arbitrary waveform resolutions. For more details please refer to the specifications in Appendix A.

Built-in Waveform Gallery

Care to use the instrument as a function generator? No need to calculate complex waveforms, as the SE5082 does the work for you. Select the standard waveforms button and start generating any of ten waveforms that are pre-computed and available for immediate use.

Included are: sine, triangle, square, pulse, ramp, sinc and others. Remember that waveforms are created from sampled waveform points, and therefore some of the waveforms cannot be generated above certain frequencies, where the number of points is insufficient to draw a perfect shape. Nevertheless, by using advanced techniques, the SE5082 generates standard waveforms frequencies up to 2.5 GHz.

Wireless Waveforms

Wireless communication, with its continuous evolution, emerging standards, and increasing bandwidth and complexity is an area where traditional generators often require replacement or upgrading in order to keep up with modern applications. The SE5082**,** offers a flexible and powerful wireless signal generation tool. Define and analyze virtually any test stimulus, whether it be at baseband, IF or RF frequencies, with a choice of analog or digital. Using the SE5082 you can generate baseband I & Q, IF that are fed directly to an up-converter, or even RF signals.

Modulated Waveforms

The SE5082 is capable of producing an array of modulation, which places it in-line with stand-alone, high performance modulation generators. AM, FM, Sweep, Chirps, FSK, PSK, ASK, and Frequency and amplitude Hops are just few examples of the available modulation schemes available on the SE5082.

Pulse Waveforms

Need to generate fast, accurate and jitter-free pulses? Just set the main operating mode to pulse generator and the instrument will be

transformed into a full-featured pulse generator. Need to control pulse transitions and placement? Just program each channel to output pulses with linear or fast transitions and control-edge placement with ns resolution. Further, if your application requires more than just a fixed duty cycle or programmable pulse width, then modulate and control your leading edge with any standard or arbitrary waveform shape. Combine all of these and you have an extremely versatile pulse generation tool. While using the pulse generator, do not forget that the SE5082 is a digital instrument and as such, creates the pulses from a digital memory. Therefore, be sure to observe the limitations that have evolved in the creation of such an advanced instrument: while most of analog pulse generators can be emulated with the SE5082, one should observe the number of points that are required to generate the pulse shape, and take into consideration the time that is required to compute and output the waveform.

Pulse Patterns

Major consideration has been given to build in the capability to design special and complex pulse patterns, such as user-defined pulse patterns and Pseudo Random Bit Sequence (PRBS).

The SE5082 pattern functionality is implemented as an extension of the Pulse Mode. In addition to RZ pulses, it allows the generation of pattern sequences that are using NRZ formatting with adjustable transition times

Local Operation

Operating the SE5082 from the front panel is intuitive and extremely easy. A large and user-friendly 4” back-lit color LCD display facilitates browsing through menus, updating parameters and displaying detailed waveform information. Combined with a numeric keypad, cursor position control and a knob, the front panel controls simplify the operation of this universal waveform source.

Remote Operation

Access speed is an increasingly important requirement for test systems. Ethernet, USB and GPIB interfaces are available so that the most suitable interface for the application may be selected. Remote control of instrument functions, parameters and waveform downloads is easily tailored to specific system environments, regardless of whether control is via a laptop computer or full-featured ATE system. IVI drivers and factory support will speed up system integration and minimize test development time and costs.

Remote Calibration

Normal calibration cycles in the industry range from one to three years, where instruments are sent to a service center, opened to allow access to trimmers, calibrated and certified for repeated usage. Leading-edge technology was employed on the SE5082 to

allow calibration from any SE5082 remote interface such as USB, GPIB or LAN. Calibration factors are stored in a flash memory, thus eliminating the need to open instrument covers.

Manual Changes

Technical corrections to this manual (if any) are listed in the back of this manual on an enclosed MANUAL CHANGES sheet.

Safety Considerations

The SE5082 has been manufactured according to international safety standards. The instrument meets IEC 61010-1, EN61010-1, CSA C22.2 No. 61010-1 and UL 61010-1 (Safety requirements for electrical equipment for measurement, control, and laboratory use – Part 1: General requirements). It has a metal chassis that is directly connected to earth via the chassis power supply cable (Safety Class I).

Only qualified, service-trained personal should remove instrument covers. Always disconnect the power cable and any external circuits before removing the instrument cover. Any adjustment, maintenance and repair of an opened, powered-on instrument must be performed by authorized service personal.

Do not use this product in any manner not specified by the manufacturer. The protective features of this product may be impaired if it is used in a manner not specified in the operation instructions.

WARNING

When the SE5082 is turned off with the front panel power switch, the instrument is placed in standby mode but is still connected to the mains. To disconnect the mains from the instrument, remove the power cord.

Do not remove instrument covers when operating the instrument or when the power cord is connected to the mains.

Only qualified, service-trained personal should remove instrument covers. Always disconnect the power cable and any external circuits before removing the instrument cover. Any adjustment, maintenance and repair of an opened, powered-on instrument must be performed by authorized service personal.

Do not use this product in any manner not specified by the manufacturer. The protective features of this product may be impaired if it is used in a manner not specified in the operation instructions.

Supplied Accessories

The instrument is supplied with a power cord and CD, which contains an electronic manual, IVI driver and supporting files. USB, LAN and synchronization cables are available upon request.

Specifications

Instrument specifications are listed in Appendix A. These specifications are the performance standards or limits against which the instrument is tested. Specifications apply under the following conditions: output terminated into 50 Ω after 30 minutes of warm up time, and within a temperature range of 20°C to 30°C. Specifications outside this range are degraded by 0.1% per °C.

Functional Description

A detailed functional description is given in the following paragraphs. The description is divided into logical groups: front panel input and output connectors, rear panel input and output connectors, operating modes, run modes, sampling modes, output type, output state, synchronization, and front panel indicators.

Front Panel Connectors and Indicators

The SE5082 has six SMA connectors on its front panel: two for each main output, one used for accepting triggers and one for the SYNC output. Each connector on the front panel has a LED associated with it, indicating when the output or input is active (LED on), or when inactive (LED off). The function of each of the front panel connectors is described in the following paragraphs.

Main Output - Channels 1 and 2

There are two connectors for each output channel, marked Normal and Inverted. When using the DC coupled output modules, the output is generated differentially through these two connectors. The AC coupled output module is single-ended and consequently, only the Normal connector outputs AC signals. The main output connectors generate standard waveforms to 2.5 GHz, arbitrary waveforms with 5 GS/s sampling rate, and sequenced waveforms that are made of arbitrary waveform segments. These outputs generate modulated and pulse waveforms, as well. Output source impedance is 50Ω (or 100Ω when connected differentially) therefore the cable connected to this output should be terminated with 50 Ω load resistance. For different load resistance, determine the actual amplitude from the following equation:

$$V_{out} = 2V_{prog}\left(\frac{R_L}{50+R_L} \right)$$

The output amplitude is doubled when the output impedance is above roughly 10 kΩ. Also, the output can be turned on and off. However, turning the output off stops the signal, but leaves low impedance on the output terminals.

SYNC Output

The SYNC output generates single or multiple TTL pulses for synchronizing other instruments (i.e., an oscilloscope) to the output waveform. The SYNC signal always appears at a fixed point relative to the waveform. The location of the sync pulse along the waveform is programmable. Since there is only one SYNC output, the output is associated with the channel 1 output, but can be changed to be sourced and synchronized to channel 2.

Note that the SYNC output is also used as a frequency marker when the SE5082 is set to generate one of the modulation functions.

TRIG IN

In general, the trigger input is used for stimulating output waveforms at the main output connector(s). The trigger input is inactive when the generator is in continuous operating mode. When placed in trigger, gated or burst mode, the trigger input is made active and waits for the right condition to trigger the instrument. The trigger input is edge sensitive, i.e., it senses transitions from high to low or from low to high.

Trigger level and edge sensitivity are programmable for the trigger input. For example, if your trigger signal rides on a dc level, you can offset the trigger level to the same level as your trigger signal, thus assuring correct the threshold for the trigger signal. The trigger level is adjustable from -5V to +5V.

The trigger input is common to both channels. Therefore, if one of the SE5082 channels is placed in trigger mode, the trigger input will affect this same channel. If both channels are placed in trigger mode, the trigger input will affect both channels simultaneously.

USB

This connector accepts standard USB-1 disk-on-key memory sticks. The main purpose of this interface is to provide for external storage and download of setups and waveforms. This I/O cannot be used for controlling the instrument from remote.

Front Panel Controls

The front panel controls and keys are grouped in logical order to provide efficient and quick access to instrument functions and parameters. Refer to Figure 1-2 throughout the following description to understand the purpose and effect of each front panel control.

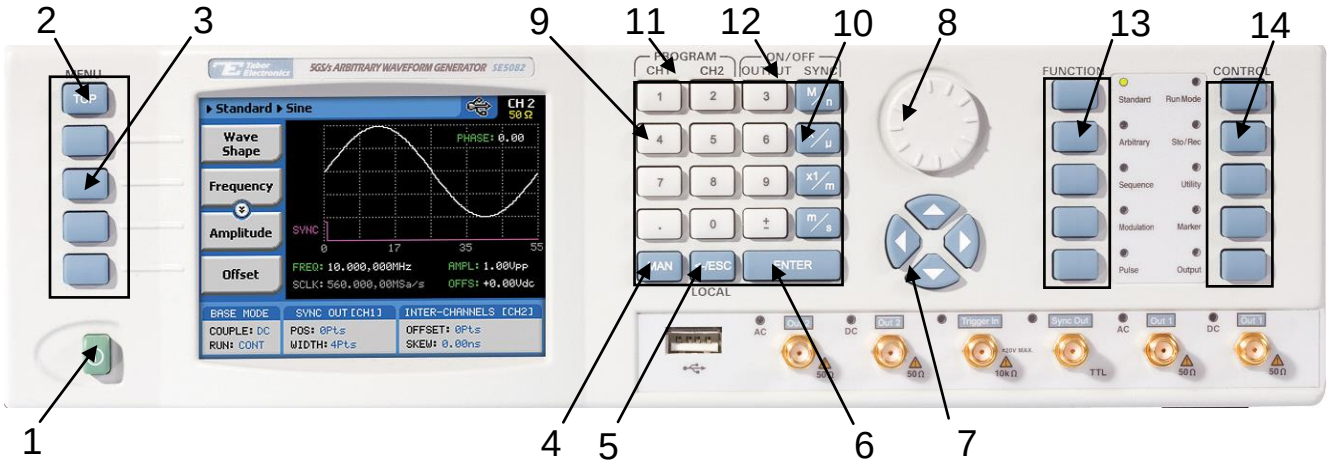

Figure 1-2, SE5082 Front Panel Controls

Note

The index in the following paragraphs point to the numbered arrows in Figure 1-2.

1. *Power Switch* – Toggles SE5082 power ON and OFF
2. *Menu Top* – For certain functions, selects the root menu. This button is disabled during parameter editing
3. *Menu Soft Keys* – Select parameters to be audited.

   These buttons are disabled during parameter editing.
4. *MAN* – Manual trigger button, used in lieu of an external trigger signal.
5. *<-Esc (Local)* – Has two functions:
   1) When in edit mode, cancels edit operation, restores last value and returns to the main function screen
   2) When operating the SE5082 from a remote interface, none of the front panel buttons are active except the Local button. When depressed, it restores control to front panel buttons

6. *Enter* – Has two functions:
    1) When multiple parameters are displayed on the screen, the cursor and the dial scroll through the parameters. Pressing Enter selects the parameter for edit. After the parameter has been modified, the Enter button locks in the new variable and releases the buttons for other operations
    2) When a parameter is modified, Enter can be used to replace the x1 suffix key
7. *Cursor UP, Down, Left and Right* – Has two functions:
    1) When multiple parameters are displayed on the screen, the cursor and the dial scroll through the parameters
    2) When a parameter is selected for editing, cursor buttons right or left move the cursor accordingly. Cursor buttons up or down modifies parameter value accordingly.
8. *Dial* – Has similar functionality as the cursor UP and Down keys.
9. *Numeric keypad* – These keys are used for modifying an edited parameter value.
10. *Parameter Suffixes (M/n, k/μ, x1/m and m/s)* – These keys are used to place a suffix at the end of the parameter. They are also used for terminating an edit operation.
11. *Program* CH1, CH2 – Use Program CH1 to modify the screen to display channel 1 parameters. Use Program CH2 to modify the screen to display channel 2 parameters. These keys can be used only when the SE5082 is not in edit mode
12. *ON/OFF Output, Sync* – These keys can be used only when the SE5082 is not in edit mode. The Output ON/OFF toggles output waveform, at the output connector, ON and OFF. The Sync ON/OFF toggles the sync waveform, at the SYNC output connector, ON and OFF
13. *Function* – These keys select one of 5 function menus that the SE5082 can generate. The output functions are: Standard, Arbitrary, Sequenced, Modulated and Pulse. A LED lights next to the selected function.
14. *Control* – These keys select control menus that program the SE5082 operating modes. The control menus are: Run Mode, Store/Recall, Utility, Markers and Output.

Rear Panel Input & Output Connectors

The SE5082 has a number of connectors on its rear panel. These connectors are described below. Figure 1-3 shows rear panel plugs, indicators, connectors and other parts.

Figure 1-3, SE5082 Rear Panel

Sample Clock In

This SMA connector accepts an external signal that will replace the internal sample clock generator. This input accepts signals covering the instrument's entire sample clock range, with an input power of 0 dBm to 10 dBm. The sample clock input is available for those applications requiring improved phase noise characteristics for the output signal. With a suitable source, the overall phase noise can be improved by a factor of up to 20 dBc/Hz. The sample clock input is active only after selecting the external SCLK source option.

NOTE

The internal sample clock generator is sensitive to noise that might be injected through the Sample Clock input. Any signal that is connected to this input may impair the operation of the internal clock. Therefore, connect a cable to the Sample Clock In connector only if you intend to use it as an external clock source. For system applications, where cables are always connected to the Sample Clock In, make sure that the external source signal is turned off at all times when the internal sample clock source is in use.

Segment / Sequence Control In

This 9-pin connector accepts TTL signals. Pin 1 to 8 control bits, 0 to 7 and pin 9 accepts a valid signal. This connector is used for controlling segments or sequences dynamically, depending if the output generates arbitrary waveforms or sequences. A specific segment is selected by applying a binary code to the segment control input. The output is changed to the selected segment after a

valid signal is asserted. The same works for sequences where a specific sequence is associated with a binary code and selected by asserting the valid signal. Channels 1 and 2 have similar inputs and each channel can be controlled to have a unique set of waveform and sequences controlled from this connector.

Marker 1 / 2 Outputs

There are two markers available for each output channel. These are marked Marker 1 and Marker 2. The markers are generated differentially through SMB connectors. From the front panel one may control marker position, width delay and amplitude. When used remotely, programmers may set multiple markers and program different marker properties for each transition instance creating complex digital patterns.

Instrument Synchronization Port

The SE5082 is equipped with a 9W4 combo D-sub connector, which accepts and arbitrates signals that are necessary to synchronize two SE5082's. The built-in synchronization feature enables fully synchronized four-channel system, without the slightest degradation of performance of individual instruments and with complete control over the start phase of each of the synchronized channels.

A special cable is required to connect between two instruments. More information on how to synchronize SE5082 instruments is given in Chapter 2.

Event In

This BNC connector duplicates the operation of the front panel trigger input, except it is used for synchronizing SE5082 operation with external events. The effect of the event input is shown in various tables of this chapter. The most common is to cause a sequence step to advance to another step.

The event input is level and edge sensitive and can be programmed just as the trigger input for trigger level and trigger slope. The event input affects both channels simultaneously if they are programmed to sense transitions from this input.

Ref In

This BNC connector accepts signals with the frequency of 10, 20, 50 or 100 MHz that reference the sample clock generator. This input is normally used for synchronizing system components to a single clock reference. The SE5082 has to be programmed to reference frequency value and placed in external reference mode before it will use this input as reference.

Ref Out (option)

This BNC connector outputs the 100MHz internal reference or if using an external reference the Ref Out outputs the external reference frequency. This output can be used to synchronize other system components to the SE5082 clock reference.

LAN

This RG45 connector accepts standard Ethernet cable. Correct setting of the IP address is required to avoid conflicts with other instruments or equipment on the network. Information on how to change IP address and load instrument drivers to the computer is provided in the Installation chapter of this manual.

USB

This connector accepts standard USB-2 cable. The connection to the host computer is automatic and does not require any address setting from within the SE5082.

GPIB

This 24-pin connector accepts standard GPIB cable. The GPIB address is configured using the front panel Utility menu. The SE5082 conforms to the IEEE-488.2 standard. Programming protocol is SCPI version 1993.0. GPIB cables are available separately from your Tabor dealer.

AC Line In

This 3-prong AC LINE connector accepts AC line voltage. The SE5082 senses the line voltage and sets the appropriate range automatically. Therefore, the traditional line voltage selector is not available on the rear panel. To avoid potentially hazardous situations, always connect the center pin to mains ground using the line cord that is supplied with the instrument.

AC Fuse

The AC fuse protects the SE5082 from excessive current. Always replace the fuse with the exact type and rating as printed on the rear panel. The fuse value is T1.25AH/250V. If the fuse blows again after replacement, we strongly recommend that you refer your instrument immediately to the nearest Tabor service center.

Run Modes

The SE5082 can be programmed to operate in one of three basic run modes: Continuous, Triggered and Gated. There are features associated with each basic run mode, such as: arming the output for an event to enable generation, delayed trigger, smart trigger and more. The SE5082 can also be prepared to accept triggers from various sources, such as: front panel trigger input, rear panel event input, an internal trigger generator and more. In addition, the generator can be programmed to respond differently to a trigger in a way that each trigger can override preceding triggered cycles or launch a continuous trigger burst where the waveform stop triggers a new waveform start after a pre-programmed interval.

Summary of all run modes, trigger sources and trigger features is given below.

Continuous Run Mode

Bench operation usually requires that a continuous waveform is available at the output terminals when the instrument is powered up and the output is turned on. On the other hand, for system applications that tolerate signals only at a specific time frame, the SE5082 can be armed from remote to generate waveforms only after it receives an enable command, and then aborted when the signal is no longer required. Table 1-1 summarizes the conditions in which the SE5082 can operate when set to continuous run mode. Armed operation is described below.

Arming the Generator in Continuous Run Mode

Arming the SE5082 is an operation that is normally carried out when the generator is operated from remote. For this reason, only a remote command can modify the arm option to Armed. When armed, the output can be enabled using valid events that are sensed by one of the front panel trigger input the rear panel event input or a remote enable command. Once enabled, the waveform can be stopped by a remote abort signal only.

Table 1-1 summarizes the various controls for each output function of which the SE5082 can be operated when set to continuous run mode. The various controls are explained below.

Waveform – basic output function that the SE5082 can generate. There are five output functions available: Standard, Arbitrary, Sequenced (including advance sequencing), Modulated and Pulse.

Advance mode – affects the generator in sequenced and advanced sequencing only.

Arm Options – bench operation normally defaults to self-armed where the output generates waveforms immediately after turning it on. Arming the SE5082 places the outputs in a wait-for-valid-enable signal before waveforms are generated through the output connectors.

Idle Waveform – defines the shape of the waveform when the generator is self-armed or when armed but has not been enabled yet.

Enable Signal – defines the source of the enable signal. Trigger defines the trigger input as the source for the enable signal. Event defines the event input as the source for the enable signal. The BUS option disables the event and trigger inputs and only a remote command will enable the output. Only one source is active at a time.

Abort Signal – defines the source of the abort signal. In this case, either the Front Panel or BUS will cause the signal to stop.

Wave Loops – attributed to the sequence modes only, it defines how many times a segment will loop in a specific sequence setting.

Seq Loops – attributed to the sequenced waveforms only, it defines how many times a sequence will loop in a specific advanced sequence setting.

Jump Flag – attributed to sequence functions only, it defines if a segment or sequence will advance to the next step or wait for an external event before the jump.

Jump Signal – defines the source of the event that will cause a segment to jump to the next step in a specific sequence setting.

Table 1-1, Arming the SE5082 in Continuous Run Mode

Waveform	Advance Mode	Arm Options	Idle Waveform	Enable Signal	Abort Signal	Wave Loops	Seq Loops	Jump Flag	Jump Signal
Standard	-	Self Armed	Wave	- (*)	-	-	-	-	-
	-	Armed	DC	Trigger | BUS | Event	BUS | F.P	-	-	-	-
Arbitrary	-	Self Armed	Wave	-	-	-	-	-	-
	-	Armed	DC	Trigger | BUS | Event	BUS | F.P	-	-	-	-
Sequenced	Auto	Self Armed	Sequence	-	-	1-1M	-	Bit (0 | 1)	Event
	-	Armed	Wave	Trigger | BUS | Event	BUS | F.P	1-1M	-	Bit (0 | 1)	Event
	Once	Armed	Wave	Trigger | BUS | Event	BUS | F.P	1-1M	1-1M	Bit (0 | 1)	-
	Stepped	Armed	Wave	Trigger | BUS | Event	BUS | F.P	-	-	-	Event
Advance Sequence	Auto	Self Armed	-		-	1-1M	-	Bit (0 | 1)	Event
	-	Armed	Sequence	Trigger | BUS | Event	BUS | F.P	1-1M	-	Bit (0 | 1)	Event
	Once	Armed	Sequence	Trigger | BUS | Event	BUS | F.P	1-1M	1-1M	Bit (0 | 1)	-
	Stepped	Armed	Sequence	Trigger | BUS | Event	BUS | F.P	-	-	-	Event
Modulated	-	Self Armed	-	-	-	-	-	-	-
	-	Armed	Wave	Trigger | BUS | Event	BUS | F.P	-	-	-	-
Pulse	-	Self Armed	-	-	-	-	-	-	-
	-	Armed	Wave	Trigger | BUS | Event	BUS | F.P	-	-	-	-

(*) defines not relevant for this mode

Triggered Run Mode

In triggered mode, the SE5082 circuits are always armed to generate one output waveform. The trigger circuit is sensitive to transitions at the trigger input. Select between positive, negative or either transitions to trigger the instrument. You may also program the trigger level to the desired threshold level. When triggered, the generator outputs one waveform cycle and remains idle at the last point of the waveform.

The SE5082 can be triggered from a number of sources:

1) Front panel connector, designated as TRIG IN
2) Rear panel connector, designated as Event IN,
3) Front panel button marked as MAN TRIG, and
4) Bus commands that are applied to the instrument from any interface, LAN, USB or GPIB.

Descriptions of the various trigger source options are detailed in the following paragraphs.

The trigger signal, whether it comes from an external source or from an interface command, is routed through electrical circuits. These circuits cause a small delay known as system delay. System delay cannot be eliminated completely. System delay is a factor that must be considered when applying a trigger signal. It defines the time that will lapse from a valid trigger edge or software command to the instant that the output reacts.

Table 1-2 summarizes the various controls for each output function of which the SE5082 can be operated, when set to triggered run mode. The various controls are explained below.

Waveform – basic output function that the SE5082 can generate. There are five output functions available: Standard, Arbitrary, Sequenced (including advance sequencing), Modulated and Pulse.

Advance mode – affects the generator in sequenced and advanced sequencing only.

Arm Options – bench operation normally defaults to self-armed where the output generate waveforms immediately after turning it on. Arming the SE5082 places the outputs in a wait-for-valid-enable signal before waveforms are generated through the output connectors.

Idle Waveform – defines the shape of the waveform when the generator is self-armed or when armed but hasn't been enabled yet.

Initiate Signal – defines the source from where a trigger is expected to initiate a waveform cycle. F.P defines the front panel MAN button as a trigger source. Event defines the rear panel Event In as a trigger source. The BUS option disables the trigger and event inputs and only a remote command will enable the output. Trigger defines the trigger input as the source for the trigger signal. Only one source is active at a time.

Abort Signal – defines the source of the abort signal. In this case, only BUS will cause the signal to stop.

Wave Loops – for standard and arbitrary waveforms, this field defines a loop counter (burst). For sequence modes only, it defines how many times a segment will loop in a specific sequence setting.

Seq Loops – attributed to the sequenced waveforms only, it defines how many times a sequence will loop in a specific advanced sequence setting.

Jump Flag – attributed to sequence functions only, it defines if a segment or sequence will advance to the next step or wait for an external event before the jump.

Jump Signal – defines the source of the event that will cause a segment to jump to the next step in a specific sequence setting.

Smart Trig – defines if smart trigger controls can be applied to the active trigger function

Table 1-2, SE5082 Triggered Run Mode Controls

Waveform	Advance Mode	Arm Options	Idle Waveform	Initiate Signal	Abort Signal	Wave Loops	Seq Loops	Jump Flag	Jump Signal	Smart Trig
Standard	-	Self Armed	-(*)	-	-	-	-	-	-	-
		Armed	DC	F.P.|BUS|Trigger	BUS	1-1M	-	-	-	Yes
Arbitrary	-	Self Armed	-	-	-	-	-	-	-	-
		Armed	DC	F.P.|BUS|Trigger|	BUS	1-1M	-	-	-	Yes
Sequenced	Auto	Self Armed	-	-	-	-	-	-	-	-
		Armed	DC	F.P.|BUS|Trigger	BUS	1-1M	-	Bit (0 | 1)	Event	Yes
	Once	Armed	DC	F.P.|BUS|Trigger	BUS	1-1M	1-1M	Bit (0 | 1)	Event	-
	Stepped	Armed	DC	F.P.|BUS|Trigger	BUS	-	-	-	Event	Yes
Advanced Sequence	Auto	Self Armed	-	-	-	-	-	-	-	-
		Armed	DC	F.P.|BUS|Trigger|	BUS	1-1M	-	Bit (0 | 1)	Event	Yes
	Once	Armed	DC	F.P.|BUS|Trigger	BUS	1-1M	1-1M	Bit (0 | 1)	Event	-
	Stepped	Armed	DC	F.P.|BUS|Trigger	BUS	-	-	-	Event	Yes
Modulated	-	Self Armed	-	-	-	-	-	-	-	-
		Armed	DC	F.P.|BUS|Trigger	BUS	1-1M	-	-	-	Yes
Pulse	-	Self Armed	-	-	-	-	-	-	-	-
		Armed	DC	F.P.|BUS|Trigger	BUS	1-1M	-	-	-	Yes

(*) defines not relevant for this mod

Trigger Run Mode Extensions

Trigger operation is available to assure that waveforms start only when a valid trigger is sensed at the trigger input connector. So it is with most waveform generators. However, the SE5082 provides further refinement of this function in a way that the trigger circuit can receive triggers from multiple sources, arbitrate and delegate the signal to the main trigger circuit and also delay the start of the waveform by a pre-defined period. This is to make sure events are not wasted and waveforms re-played only when they are absolutely required in the system. The various trigger-run mode extensions are described below.

Delayed Trigger

The delayed trigger function is exactly the same as the trigger mode, except a programmable delay circuit inhibits the generation of the output waveform for a pre-determined interval following a valid trigger event. The programmed delay time defines the time that will lapse from a valid trigger (hardware or software) to output. The delay is programmable in steps of sample clock period and can be set from 0 to 8e6 sample clock periods. Delayed trigger can be used in conjunction with external trigger, front panel manual trigger, or remote trigger command. It has no effect on the internal trigger timer, nor does it affect the event input.

Trigger Override

Normal trigger sequence requires that a waveform completes its cycle before the generator can be triggered to generate another waveform. Some applications, however, cannot wait for the end of the waveform, but must initiate another trigger cycle. The SE5082 features an override mode that ignores the output and upon trigger, initiates a fresh output waveform, whether the last one has been completed or not. Note that the trigger override mode operates on single segments only; it does not affect sequences and/or advanced sequence waveforms.

Internal Timer

The internal timer generator operates as a free-running asynchronous trigger generator. It may be used for applications that require periodical and constant generation of output cycles, or to replace external trigger devices. The internal trigger generator is programmed in units of time through the range of 200 ns to 20 seconds. Note: in order to prevent errors, the period of the internal generator must be larger than the period of the output waveform.

Delay Timer

Contrary to the trigger override mode that was described before, certain applications cannot tolerate new cycles before the last one comes to a halt. The SE5082 features a special re-trigger mode that is very similar to the internal trigger generator where the output is self-triggered by an internal trigger generator, but the new waveform starts only after a programmed delay interval, and therefore assures that triggers are spaced equally and never override active sequences. The delay value is programmed in units ranging from 152 to 8e6 sample clock periods.

Counted Burst

Counted burst is an extension of the trigger run mode. It allows generation of a counted number of output cycles, which are triggered by a single trigger event. The burst counter can be used in conjunction with all waveform functions that the SE5082 generates. The burst counter is programmed from 1 to 1,000,000.

Smart Trigger

Smart trigger is an additional extension of the trigger capability, where it narrows down to a specific pulse event. The trigger circuit can detect: a pulse having a pulse width larger than a programmed time value (<time), a pulse having a pulse width smaller than a programmed time value (>time), or a pulse having a pulse width between two limits (<>time).

An additional feature of the smart trigger is the holdoff function where the output is held idle after the first trigger and starts a waveform cycle only with the first valid trigger after a holdoff interval has lapsed.

Trigger Source

The SE5082 can be stimulated to produce waveform functions from a number of trigger sources. These are described in the following paragraphs.

Summary of trigger source options and trigger features are listed in Tables 1-3 and 1-4, identifying legal run modes, trigger sources and trigger features and listing possible setting conflicts.

Trigger Input

When selecting the External trigger source, the front panel TRIG IN connector becomes active and every valid signal that is applied to this input is stimulating the SE5082 to generate a new waveform cycle. When the external trigger source is active, all other sources are disabled, except the MAN front panel button. The characteristics of the trigger input are listed in Appendix A. The trigger input is sensitive to transitions and may be programmed to react on positive, negative or both positive and negative transitions. It can also be programmed to a specific trigger levels setting from -5 V to 5 V.

Manual Button

Alternately, if an external signal is not available, the front panel MAN button may also be used to trigger the instrument. This button is active only when EXT trigger option is selected.

Remote Command

When selecting Bus as a trigger source, the front panel TRIG IN connector and MAN button are disabled and only trigger commands from a remote interface are accepted by the instrument. In this case, the event input is also disabled. Make sure that the appropriate trigger source is selected if you mix remote and local operation.

Event Input

The event input is a rear-panel connector that duplicates functionality of the front panel trigger input, but can be used for detecting events or advancing sequence steps. For most applications, this input should not be used as a second trigger input. One of the main applications for this input is to serve as an enable port for armed operation in continuous run mode.

Similar to the front panel trigger input, the event input is sensitive to transitions and may be programmed to react on positive or negative transitions. It can also be programmed to a specific trigger levels setting from -5 V to 5 V.

Internal Timer

The internal timer is normally used when an external source trigger is not available and an application requires triggers at constant intervals. The internal trigger generator can be programmed for start-to-start trigger intervals, using time units and for stop-to-start trigger intervals, using sample clock period units.

When one of the internal timers is activated, all external and remote triggers are blocked to prevent interference to the internal timer from external devices.

Table 1-3, Run Modes and Trigger Source Options Summary

Run Mode	**Trigger Option**	**Status**
Continuous (Armed)	External – TRIG IN External – EVENT IN Internal Front panel MAN button Remote command	Active Active Disabled Disabled Active
Triggered	External – TRIG IN External – EVENT IN Internal Front panel MAN button Remote command	Active Active Active Active Active
Gated	External – TRIG IN External – EVENT IN Internal Front panel MAN button Remote command	Active Active Disabled Disabled Disabled

Table 1-4, Trigger Sources and Trigger Features Options Summary

Trigger Source	Trigger Feature	Status
TRIG IN	Self armed / Armed	Disabled
	Delayed trigger	Active
	Burst counter	Active
	Internal timer	Disabled
	Internal delay (re-trigger)	Disabled
	Normal / Override	Active
	<>time pulse width detector	Active
	Holdoff	Active
EVENT IN	Self armed / Armed	Active
	Delayed trigger	Active
	Burst counter	Active
	Internal timer	Active
	Internal delay (re-trigger)	Active
	Normal / Override	Disabled
	<>time pulse width detector	Disabled
	Holdoff	Disabled
Internal	Self armed / Armed	Disabled
	Delayed trigger	Disabled
	Burst counter	Active
	Internal timer	Active
	Internal delay (re-trigger)	Active
	Normal / Override	Disabled
	<>time pulse width detector	Disabled
	Holdoff	Disabled
Bus	Self armed / Armed	Active
	Delayed trigger	Active
	Burst counter	Active
	Internal timer	Disabled
	Internal delay (re-trigger)	Disabled
	Normal / Override	Active
	<>time pulse width detector	Active
	Holdoff	Active
MAN	Self armed / Armed	Disabled
	Delayed trigger	Active
	Burst counter	Active
	Internal timer	Disabled
	Internal delay (re-trigger)	Disabled
	Normal / Override	Active
	<>time pulse width detector	Disabled
	Holdoff	Disabled

Gated Run Mode

In gated mode, the SE5082 generates output waveforms on a stable gate level between two gate transitions. The gate opens on the first trigger transition and closes on the second transition. Gate level and transition slopes are programmable. Trigger delay and re-trigger do not apply to the gated run mode.

Table 1-5 summarizes the various controls for each output function of which the SE5082 can be operated when set to gated run mode. The various controls are explained below.

Table 1-5, SE5082 Gated Run Mode Controls

Waveform	Advance Mode	Arm Options	Idle Waveform	Initiate Signal	Abort Signal	Wave Loops	Seq Loops	Jump Flag	Jump Signal	Smart Trig
Standard	-	Self Armed	-(*)	-	-	-	-	-	-	-
		Armed	DC	Trigger|Event	BUS	-	-	-	-	Yes
Arbitrary	-	Self Armed	-	-	-	-	-	-	-	-
		Armed	DC	Trigger|Event	BUS	-	-	-	-	Yes
Sequenced	Auto	Self Armed	-	-	-	-	-	-	-	-
		Armed	DC	Trigger|Event	BUS	1-1M	-	Bit (0 | 1)	Event	Yes
	Once	Armed	-	-	-	-	-	-	-	-
	Stepped	Armed	-	-	-	-	-	-	-	Yes
Advanced Sequence	Auto	Self Armed	-	-	-	-	-	-	-	-
		Armed	DC	Trigger|Event	BUS	1-1M	1-1M	Bit (0 | 1)	Event	Yes
	Once	Armed	-	-	-	-		-	-	-
	Stepped	Armed	-	-	-	-	-	-	Event	Yes
Modulated	-	Self Armed	-	-	-	-	-	-	-	-
		Armed	DC	Trigger|Event	BUS	1-1M	-	-	-	Yes
Pulse	-	Self Armed	-	-	-	-	-	-	-	-
		Armed	DC	Trigger|Event	BUS	1-1M	-	-	-	Yes

(*) defines not relevant for this mode

Waveform – basic output function that the SE5082 can generate. There are five output functions available: Standard, Arbitrary, Sequenced (including advance sequencing), Modulated and Pulse.

Advance mode – affects the generator in sequenced and advanced sequencing only.

Arm Options – bench operation normally defaults to self-armed when the output generates waveforms immediately after turning it on. Arming the SE5082 places the outputs in a wait-for-valid-enable signal before waveforms are generated through the output connectors. Self-armed is not optional for gate operation.

Idle Waveform – defines the shape of the waveform when the generator is armed, but has not been enabled yet.

Initiate Signal – defines the source from where a gating signal is expected to initiate an output. F.P defines the front panel MAN button as a manual gating source.

Abort Signal – defines the source of the abort signal. In this case, only BUS will cause the signal to stop.

Wave Loops – for standard and arbitrary waveforms, this field defines a loop counter (burst). For sequence modes only, it defines how many times a segment will loop in a specific sequence setting.

Seq Loops – attributed to the sequenced waveforms only, it defines how many times a sequence will loop in a specific advanced sequence setting.

Jump Flag – attributed to sequence functions only, it defines if a segment or sequence will advance to the next step or wait for an external event before the jump.

Jump Signal – defines the source of the event that will cause a segment to jump to the next step in a specific sequence setting.

Smart Trig – defines if smart trigger controls can be applied to the active trigger function

Sampling Modes

Conventional DACs use a standard non-return to zero (NRZ) sampling. The frequency response of NRZ sampling is the classic sin(x)/x or Sinc(x) function roll off with nulls at integer multiples of the sampling frequency, as can be seen in figure 1-4. This response delivers best power performance over the first Nyquist Zone. However, beyond the first Nyquist zone, the signal power is severely attenuated due to the notch centered at the sampling frequency (Fs). So effectively, standard DACs can deliver good signal power only up to about the mid-point of the second Nyquist Zone.

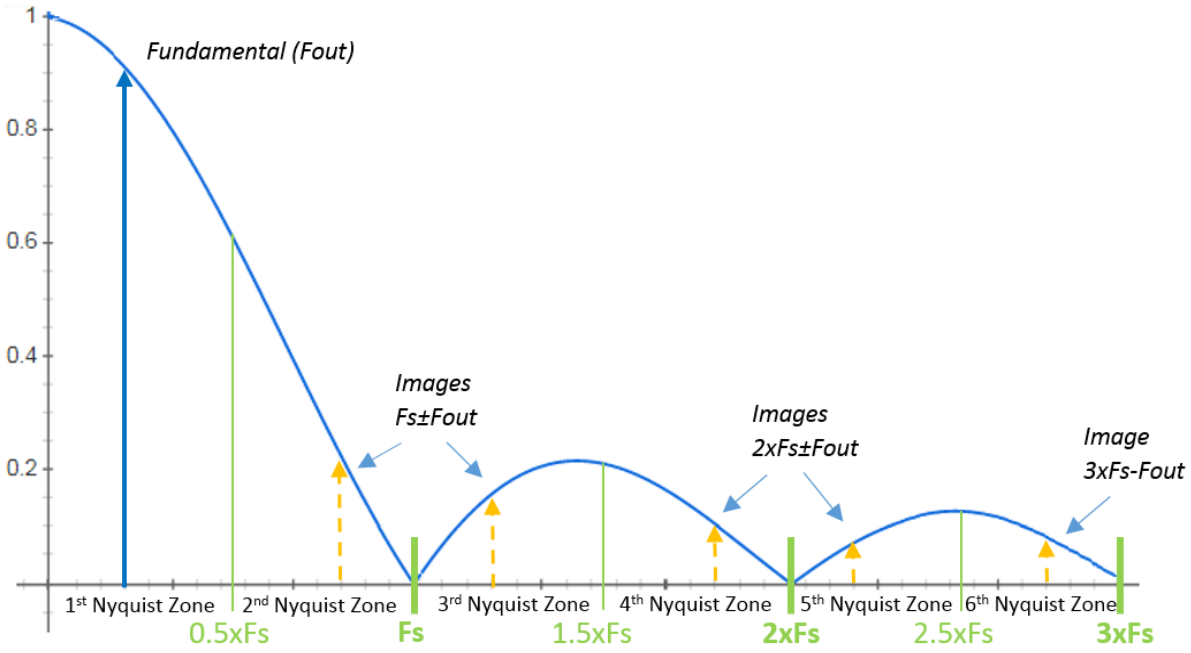

Figure 1-4, Frequency response of ideal DAC using NRZ coding

The new SE5082 incorporates new DAC technology that enables selecting different sampling modes which enable tailored power delivery across multiple Nyquist zones up to the 4th zone and even beyond. This ability to shift the frequency response so that more power is delivered in higher Nyquist zones enables generating direct RF signals well beyond the sampling frequency of the DAC. This is equivalent to having a DAC with a higher sampling rate.

There are four sampling modes in the SE5082, Non-return to zero (NRZ), return to zero (RTZ), narrow return to zero (NRTZ) and broadband operation (RF). Each sampling mode has an optimal Nyquist zone for generating signals as detailed by table 1-6. The difference in the frequency response of the DAC for each sampling mode can be seen in figure 1-5.

Table 1-6, Comparison between the SE5082 four sampling modes

Mode	ONZ*	Advantages	Trade-offs
NRZ	1st only	Best 1st NZ noise performance.	Steep dynamic tailoff > 1st NZ. Legacy mode.
RTZ	2nd & 3rd	Best SFDR mode. Extended bandwidth. Possible operation in 4th & 5th NZ.	6 dB carrier power loss in 1st NZ. Reduced SFDR. Strong spur at Fclk.
NRTZ	1st & 2nd	Peak carrier power in 1st & 2nd NZ. Extended 2nd NZ dynamics (better than NRZ).	3rd NZ notch, spur at Fclk.
RF	2nd & 3rd	Best for 2nd & 3rd NZ power. Validated operation in 4th NZ. Peak power at Fs! Uses 2x rate clock	Clock spurs at Fclk and 2xFclk
*ONZ = Optimum Nyquist zone			

NRZ mode

This is the default sampling mode in the SE5082. It is the optimal mode for signals generated in the first Nyquist zone and its frequency response follows the standard Sinc(x) roll off.

RTZ mode

The RTZ sampling mode, using a 50% re-sampling clock, the DAC's Sinc(x) function is spread over double the frequency band (notch frequency shifts out to 2xFs). The clear benefit is that operation in NZ2 & NZ3 is now possible. There is also usable output power in NZ6 & NZ7. The downside is a 6dB loss of NZ1 power.

NRTZ mode

The NRTZ sampling mode, offers a trade-off between noise performance and bandwidth – taking the best of NRZ and RTZ modes. Clearly for maximum power transfer, intuition suggests that NRZ is the optimum mode for DAC operation. However, by enabling a narrow excursion to zero (NRTZ) every clock cycle, several benefits arise.

- Improvement of the spectral purity at the highest signal frequencies.
- Transient noise is eliminated and noise to power ratio is improved.

RF mode

The RF sampling mode, is the widest band capability offered by the SE5082. Here the DAC produces positive and negative going output pulses for each clock cycle which can more than double the output attenuation bandwidth, pushing the second notch to well beyond 2xFs. Note that peak output power is achieved when the carrier signal frequency coincides with the sample frequency Fs. Consequently the maximum output power occurs exactly at the boundary between NZ2 and NZ3.

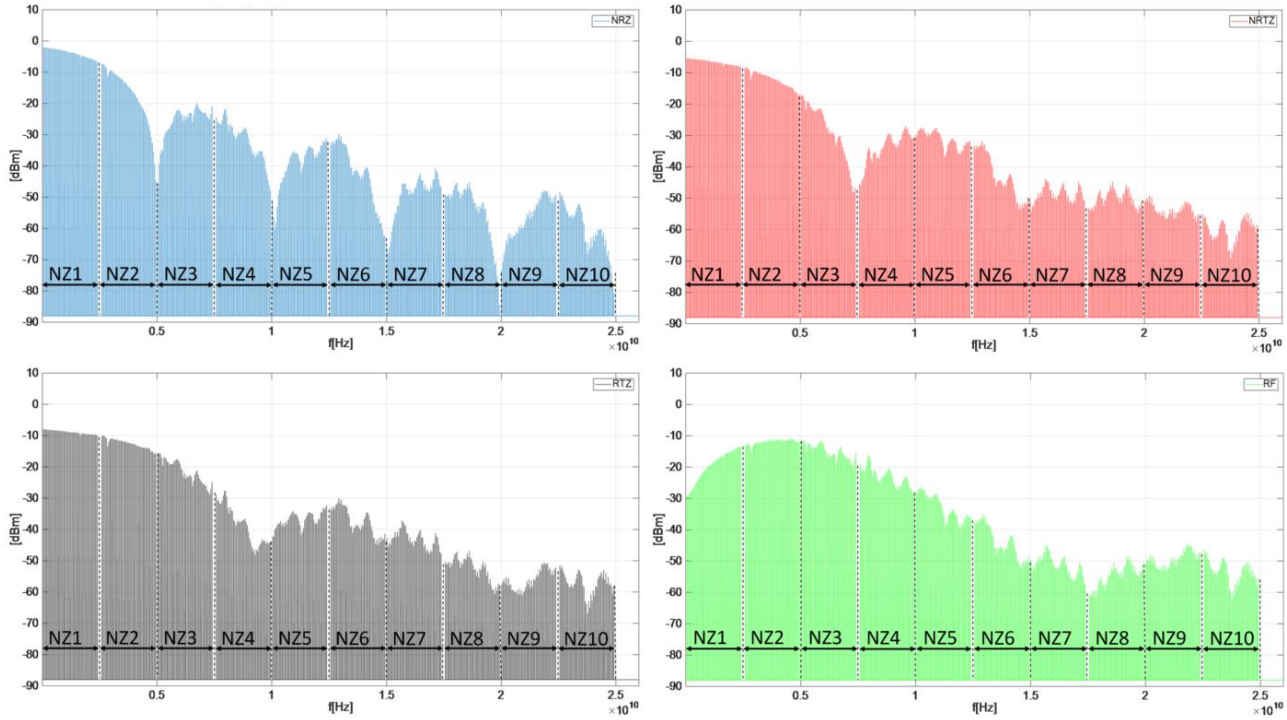

Figure 1-5, SE5082, DAC output frequency response for each sampling mode

Output Type

The SE5082 can output various types of waveforms, including: Standard, Arbitrary, Sequenced, Modulated and Pulse. The various output types are described in the following paragraphs.

Standard Waveforms

An array of built-in standard waveforms is available when the SE5082 is placed in its standard waveforms mode. The waveforms are calculated mathematically from known equations and converted to waveform coordinates that are then downloaded to the working memory. Unlike analog function generators that use electrical circuits to produce waveform shapes, the SE5082 must compute its waveform coordinates every time a new function is selected or every time that parameters of the function change.

The SE5082 can produce 10 standard waveforms: sine, triangle, square, ramp and pulse, sinc, gaussian and exponential pulses, DC and Pseudo-random noise. Some of the waveform parameters can be modified, such as: start phase for sine and triangle, duty cycle for square, rise and fall times for pulses, etc. The standard waveforms are the most commonly used wave shapes and therefore, were collected to a library of standard waveforms that can be used without the need to do off-product computation and use remote devices to download waveform coordinates.

The repetition rate of the standard waveforms is given in units of Hz. Since each channel has its own clock source, different waveforms, frequencies and run modes can be generated from its two outputs. On the other hand, when a common sample clock feed is selected, the two outputs must share the same run mode, but can still generate different waveforms and a unique set of parameters for each channel without interference between the channels. When synchronization between channels is turned on, minimal skew of the waveform starting edges is maintained between the two channels.

Figure 1-6 shows typical display for the standard waveforms function.

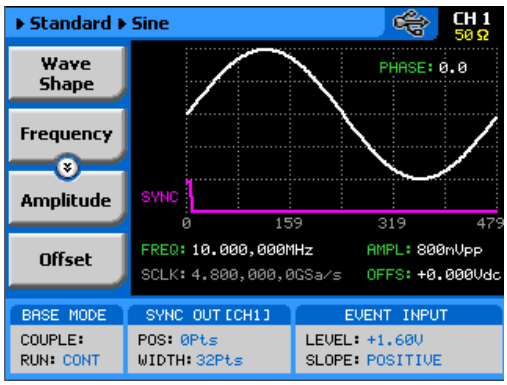

Figure 1-6, Typical SE5082 Standard Waveforms Display

Arbitrary Waveforms

One of the main functions of the SE5082 is generating real-life waveforms. These are normally not sine waves and squares, but user specific waveforms. Generating such waveforms requires external utilities such as MATLAB or even spreadsheets, but having the program alone is not enough for the SE5082. Once the waveform is computed and defined, it must be converted to coordinates and then to a format, which the instrument can accept.

Arbitrary waveforms are stored as digital XY coordinates in a special memory, normally referred to as working memory. Each coordinate is referred to as waveform point, or waveform sample. The waveform is better defined if it has many waveform points. For example: with only 8 points, a sine waveform will hardly resemble the shape of a sine wave and will look more like an up and down staircase, but with 100 points, the same sine waveform will look almost perfect.

The final shape of the waveform is produced by a DAC (Digital to Analog Converter). The waveform samples are clocked to the DAC at a rate defined by the sample clock frequency. The output of the DAC converts the digital data to analog levels and passes on the signal to the output amplifier. The shape of the function is more or less the same as it comes out of the DAC, except it could be amplified or attenuated, depending on the required amplitude level.

The size of the working memory is limited to the way the hardware was designed. The SE5082 has 32M points available as standard (64M point optional) to build one or more waveforms. There is no need to use the entire memory for only one waveform, as the memory can be divided into smaller segments and loaded with different waveforms. The generator can be then programmed to output one segment at a time.

The SE5082 has separate arbitrary waveform memories for each channel and each channel can be loaded with different waveforms. Channels are not limited by the number of segments and by the shape of the waveforms.

Figure 1-7 shows a typical display for the arbitrary waveform function.

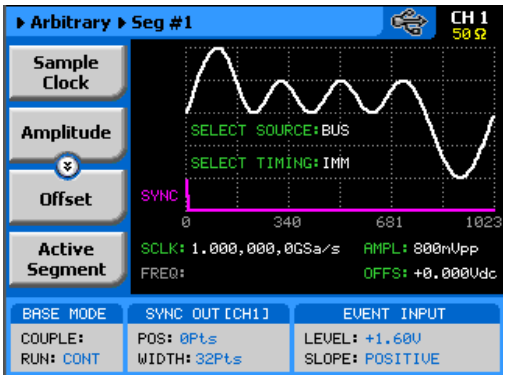

Figure 1-7, Typical SE5082 Arbitrary Waveforms Display

Sequenced Waveforms

The sequence generator is a very powerful tool that lets you link and loop segments in any way you desire. The SE5082 has two separate sequence generators – one for each channel. Each sequence generator is dedicated to its own channel. Figure 1-8 shows a typical display for the sequenced waveforms function.

The sequence circuit is useful for generating long waveforms with repeated sections. The repeated waveform has to be programmed once and the repeater will loop on this segment as many times as selected. When in sequenced mode, there is no time gap between linked or looped segments. Sequence tables must be loaded to the generator before sequenced waveforms can be generated. The data for the sequence table is first prepared on an external platform and then downloaded to the generator using external utilities such as MATLAB.

As a simple example of a sequenced waveform, look at Figures 1-9 through 1-11. The waveforms shown in these figures were placed in memory segments 1, 2 and 3, respectively. The sequence generator takes these three waveforms links and loops them in a predefined order to generate the waveform shown in Figure 1-12.

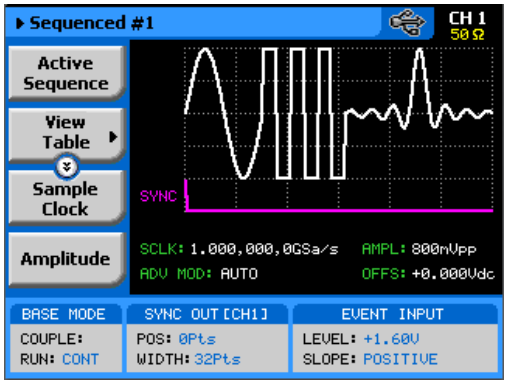

Figure 1-8, Typical SE5082 Sequenced Waveforms Display

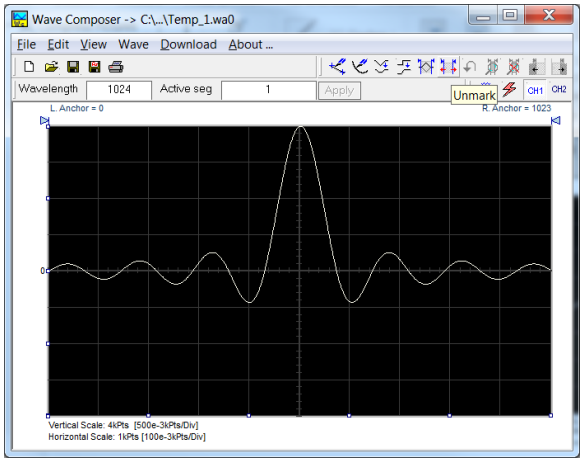

Figure 1-9, Segment 1 Waveform – Sinc

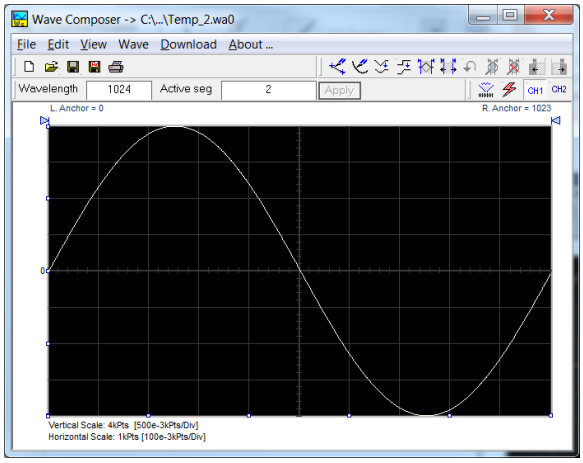

Figure 1-10, Segment 2 Waveform - Sine

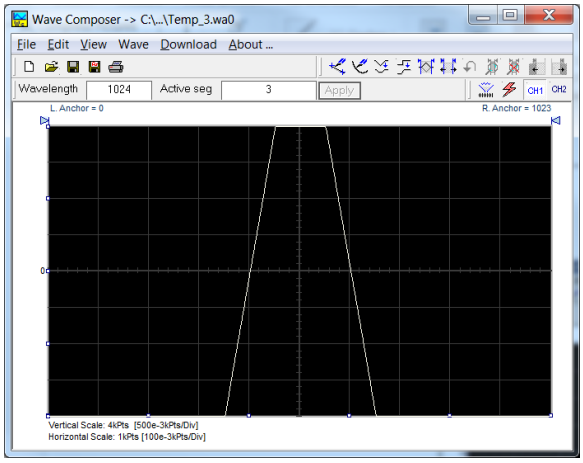

Figure 1-11, Segment 3 Waveform – Pulse

The following sequence was made of segment 2 repeated twice, segment 1 repeated four times, and segment 3 repeated two times.

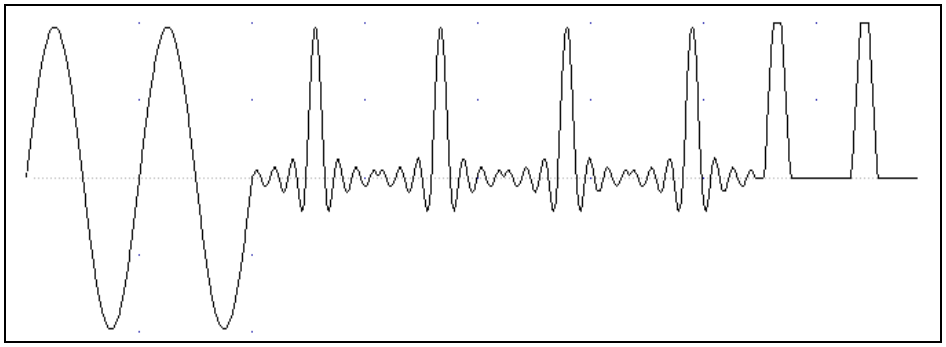

Figure 1-12, Sequenced Waveform

Figure 1-13 shows a typical front panel entry for a simple sequence table.

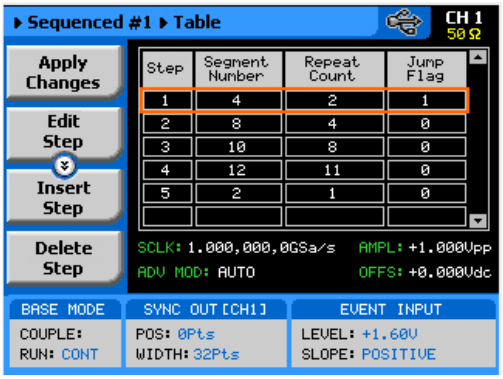

Figure 1-13, Typical Display of a Sequence Table

Advanced Sequencing

Advanced sequencing is an extension of the sequencer functionality. It describes a function that is capable of sequencing complete sequences and therefore creating very complex scenarios that otherwise would have required extremely large memory banks. For example, a sequence of initiating communication between ground station and airborne vehicle requires repetition of a certain sequence of transmissions, and then the message itself is embedded in a separate transmission sequence. Such a complex communication scheme is easily achieved with the advanced sequencing technique that has been employed by the SE5082. Information how to build sequences and generate advanced sequences is given in Chapter 3.

Sequence Advance Mode

As shown above, sequences are built as a simple table, defining link, segment, loops and advance bits. When placed in sequenced mode, the output is changed from step-to-step in ascending order. The term Sequence Advance Modes defines what is causing the instrument to step through the table rows. There are three basic advance modes that can be selected for the sequence mode: Auto, Once and Stepped. The various advance mode options are summarized separately for each run mode option in Tables 1-2, 1-4 and 1-5. These modes are explained in the following paragraphs.

Auto Advance Mode

Auto advance sequence is the mode used when the sequence is expected to run continuously from the first step in the sequence table to the last, and then resume the same sequence automatically from its first step. There are no interruptions between steps and no interruptions between the last and the first step of the sequence. When auto mode is selected, the SE5082 can also be armed to start only when an enable signal is established.

Once Advance Mode

The Once sequence advance is the mode used when the sequence is expected to advance on trigger events only. The trigger source is selectable from an external, front panel MAN button, or a remote command. The sequence will run upon valid initiation signal and will stop and wait for another trigger to commence with another cycle. Jump bits can be placed in various sequence steps to allow dwelling on a specific waveform segment and the next step will start only after a valid jump signal is asserted.

The Once advance mode has a repeat counter for applications requiring a counted number of sequences. In this case, the counter can be programmed to generate 1 to 1,000,000 sequences automatically and then wait for a fresh trigger event to repeat this scenario.

Stepped Advance mode

Stepped advance sequence is the mode used when the sequence is expected to advance on specific events only. This advance mode operates in conjunction with all Run Modes, as listed in Tables 1-1, 1-2 and 1-5. When stepped advance mode is selected, the generator steps through sequence steps on valid events only. After the last step in the sequence table, the generator advances to the first row and repeats the sequence automatically.

Sequence Advance Source

As explained above, the Once and Stepped sequence advance modes require a stimulus signal to advance to the next step in the sequence table. There are a number of inputs where the SE5082 can be programmed to wait for an event that implies go-to-next-step instruction, including: Front panel MAN button, Front panel TRIG IN connector, Rear panel EVENT IN connector, or a software command that is asserted to the generator through one of its interface options. Table 1-1, 1-2 and 1-5 list all advance source options for each of the basic run modes: Continuous, Triggered and Gated.

Modulated Waveforms

Modulated waveforms are computed mathematically and downloaded to the arbitrary waveform memory and generated at the output connector as an arbitrary waveform. The SE5082 is capable of producing an array of modulation, which places this generator in-line with stand-alone, high performance modulation generators. The SE5082 can produce: AM, FM, Sweep, Chirps, FSK, PSK, ASK, and Frequency and amplitude Hops. The SE5082 can also generate many types of (n)PSK and (n)QAM schemes. The various modulation schemes are described below.

Figure 1-14 shows a typical display for the modulated waveform function. General description of all modulation functions is given in the following.

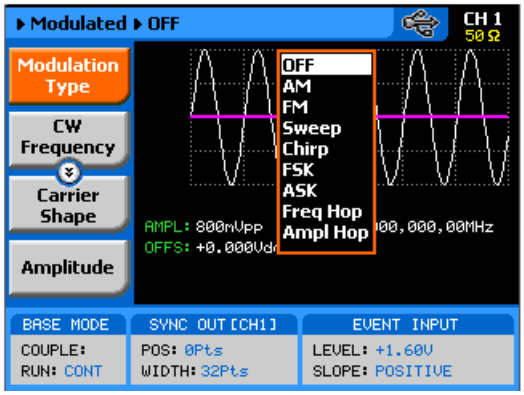

Figure 1-14, Typical SE5082 Modulated Waveform Display

Modulation Off

In modulation OFF, the output generates continuous Carrier Waveform frequency. The carrier waveform is sine wave and its frequency can be programmed using the CW Frequency menu. The value programmed for the CW Frequency parameter, is used for all other modulation functions.

AM

The AM function enables amplitude modulation of a carrier waveform (CW). The carrier waveform is sine wave and it is being modulated by an internal waveform, normally referred to as envelope waveform. The envelope waveform can be selected from sine, triangle square or ramp shapes.

FM

The FM function allows frequency modulation of a carrier waveform (CW). The carrier waveform is sine wave and it is being modulated by an internal waveform, normally referred to as modulating waveform. The modulating waveform can be selected from sine, triangle or square waveforms.

Sweep

Sweep modulation allows carrier waveform (CW) to sweep from one frequency, defined by the sweep start parameter to another

frequency, defined by the sweep stop parameter. Note that CW is sine wave only. The start and stop frequencies can be programmed throughout the entire frequency range of the instrument.

Chirp

Chirp modulation allows carrier waveform (CW) to sweep from one frequency, defined by the chirp start parameter to another frequency, defined by the chirp stop parameter. And at the same time swing from a certain amplitude level defined by the start amplitude to another amplitude level, defined by the stop amplitude. One may select to sweep the frequency and amplitude using linear steps but can also define logarithmic steps or mix of both.

The start and stop frequencies, as well as, the start and stop amplitude levels can be programmed throughout the entire frequency and amplitude ranges of the instrument.

FSK

FSK (Frequency Shift keying) modulation allows frequency hops between two pre-programmed frequencies: Carrier Waveform Frequency and Shifted Frequency. Note that CW is sine wave only and that the switch between two frequencies is always coherent.

ASK

ASK (Amplitude Shift keying) modulation allows amplitude hops between two pre-programmed amplitude levels. Note that CW is sine wave only. The signal level can hop between two amplitude levels throughout the entire amplitude range without crossing range or relay ranges.

Frequency Hop

In frequency hop mode, the output waveform (sine wave) hops from frequency to frequency in a sequence defined by the hop table. There are two hop types:

1. Frequency hops with fixed dwell time and
2. Frequency hops with variable dwell time

Amplitude Hop

In amplitude hop mode, the output waveform (sine wave) hops from amplitude to amplitude in a sequence defined by the hop table. There are two hop types:

1. Amplitude hops with fixed dwell time and
2. Amplitude hops with variable dwell time

Pulse Waveforms

The pulse generator function transforms the instrument into a pulse generator with the capability to generate pulses, exactly as they would be generated by a stand-alone pulse generator instrument. However, one should not forget that the SE5082 is a digital instrument and therefore the pulses are computed and placed in the waveform memory and then replayed at the output as arbitrary waveforms. When using this pulse function, one could program all

pulse parameters in units of time (seconds) just as it would have been done on an analog instrument, but as a digital instrument, some of the parameters may take time to be computed and placed at the output connector.

All pulse parameters are programmable, including: period, pulse width, rise and fall times, delay, polarity and more. As a dual channel instrument, you may program different pulse settings for each channel, operate each channel separately or synchronize the two channels so that both outputs transition at the same time. Operating instructions for the pulse generator are given in Chapter 3.

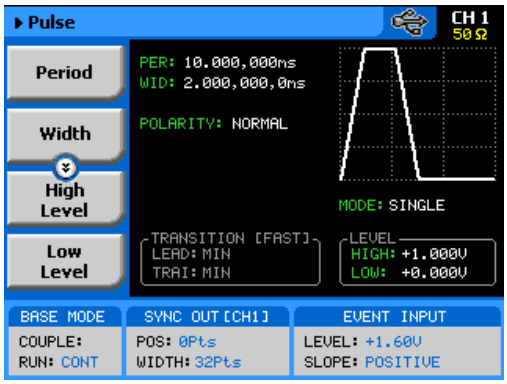

Figure 1-15, Typical SE5082 Pulse Generator Display

Pattern Waveforms

The pattern generator function transforms the instrument into a pulse generator with the capability to generate pulse patterns, exactly as they would be generated by a stand-alone pattern generator instrument. This includes user defined or PRBS random patterns, multi-level pattern for generating PAM signals and initialization or preamble pattern definition. However, one should not forget that the SE5082 is a digital instrument and therefore the patterns must be downloaded to the pattern memory and then replayed at the output as arbitrary waveforms.

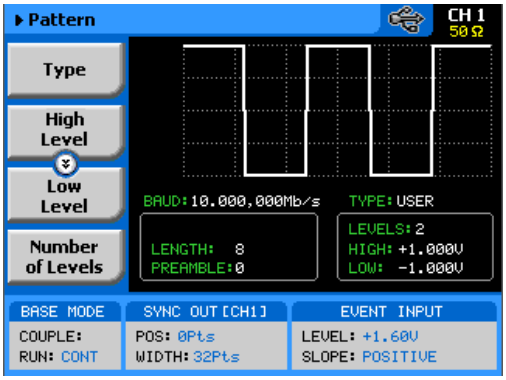

Figure 1-16, Typical SE5082 Pulse Pattern Generator Display

Output State

The main outputs can be turned on or off. However, the internal circuit is left connected to the output connector and so the impedance on this output remains low. If you do not intend to use a specific output, always remove the cables from this connector to prevent accidental application of externals signals to the output circuit. For application safety reason, when the instrument is powered up, the outputs are always off. For the same reason, when front panel settings are recalled from a stored setup, the outputs are always off.

Programming the SE5082

Some of the SE5082 functionality can be operated from the front panel and is mostly done with built-in libraries of waveforms. However, most applications require that waveforms are downloaded from a host computer and therefore, remote operation is essential for this class of instrument. Programming the SE5082 requires that the appropriate software utilities be installed in the computer and the rest is a matter of practice and knowledge of the language in use. There are other system considerations, such as address selection, that have to be settled before programming the instrument. These topics are discussed in later chapters.

Low level programming of the SE5082 is done using SCPI commands. Programming aspects are covered in Chapters 4. High-level drivers like IVI drivers are beyond the scope of this manual. Contact your Tabor representative for more information about high level drivers for the SE5082.

This page was intentionally left blank

Chapter 2
Installation

Title **Page**

Installation Overview

This chapter contains information and instructions necessary to prepare the SE5082 for operation. Details are provided for initial inspection, grounding safety requirements, repackaging instructions for storage or shipment, installation information and Ethernet address configuration.

Unpacking and Initial Inspection

Unpacking and handling of the generator requires standard precautions and procedures applicable to handling of all sensitive electronic equipment. The contents of all shipping containers should be gone through to be sure that all accessories are included and checked against the packing slip to determine that the shipment is complete.

Safety Precautions

CAUTION

This product is intended for use by qualified persons who are familiar with the safety precautions required to avoid possible injury. Read the operating information carefully before using the product.

WARNING

For maximum safety, do not touch the product, test cables, or any other instrument parts while power is applied to the circuit under test. ALWAYS remove power from the entire test system before connecting cables or jumpers, installing or removing cards from the computer, or making internal changes such as changing the module address.

Do not touch any object that could provide a current path to the common side of the circuit under test or power line (earth) ground. Always handle the instrument with dry hands.

WARNING

Always keep the lid closed when power is applied to the device under test conditions. Carefully read the Safety Precautions instructions that are supplied with your test fixtures. Any adjustment, maintenance and repair of an opened, powered-on instrument must be performed by authorized service personnel.

Performance Checks

The instrument has been inspected for mechanical and electrical performance before shipment from the factory. It is free of physical defects and in perfect electrical order. Check the instrument for possible damage in transit and perform the electrical procedures outlined in the section entitled Unpacking and Initial Inspection.

Power Requirements

The waveform generator may be operated from a wide range of mains voltage from 100 to 240 Vac. Voltage selection is automatic and does not require switch setting. The instrument operates over the power mains frequency range of 50 to 60Hz. Always verify that the operating power mains voltage is the same as that specified on the rear panel. The instruments power consumption is 150VA max.

The SE5082 should be operated from a power source with neutral or near ground (earth potential). The instrument is not intended for operation from two phases of a multi-phase ac system or across the legs of a single-phase, three-wire AC power system. Crest factor (ratio of peak voltage to rms.) should be typically within the range of 1.3 to 1.6 at 10% of the nominal rms. mains voltage.

Grounding Requirements

To ensure the safety of operating personnel, the U.S. O.S.H.A. (Occupational Safety and Health) requirement and good engineering practice mandate that the instrument panel and enclosure be “earth” grounded. Although BNC housings are isolated from the front panel, the metal part is connected to earth ground.

WARNING

Do not attempt to float the output from ground, as it may damage the Model SE5082 and your equipment.

Long Term Storage or Repackaging for Shipment

If the instrument is to be stored for a long period of time or shipped to a service center, proceed as directed below. If repacking procedures are not clear to you or, if you have questions, contact your nearest **Tabor** Representative, or the **Tabor** Customer Service Department.

1. Repack the instrument using the wrappings, packing material and accessories originally shipped with the unit. If the original container is not available, purchase similar replacement materials.
2. Be sure the carton is well-sealed with strong tape or metal straps.
3. Mark the carton with the model and serial number. If it is to be shipped, show sending and return addresses on two sides of the box.

NOTE

If the instrument is to be shipped to Tabor for calibration or repair, attach a tag to the instrument, identifying the owner. Note the problem, symptoms, and service or repair desired. Record the model and serial number of the instrument. Show the RMA (Returned Materials Authorization) order as well as the date and method of shipment. ALWAYS OBTAIN AN RMA NUMBER FROM THE FACTORY BEFORE SHIPPING THE SE5082 TO TABOR.

Preparation for Use

Preparation for use includes removing the instrument from the container box, installing the software and connecting the cables to its input and output connectors.

Installation

If the intention is to mount the instrument in a rack, it must be installed in a way that clears air passage to its cooling fans. For inspection and normal bench operation, place the instrument on the bench so it is clear of any obstructions to the rear fan, to ensure proper airflow.

CAUTION

Using the SE5082 without proper airflow will result in damage to the instrument.

Installing Software Utilities

The SE5082 is supplied with a CD that contains an IVI Driver, and some other utilities to aid you with the operation of the instrument. For bench operation, all that you need from the CD is this manual. However, it is recommended that you stow away the CD in a safe place in case you want to use the SE5082 from a host computer or a system in the future.

The *IVI driver* is a useful utility that provides standard communication and commands structure to control the SE5082 from remote. Programming examples are also available to expedite your software development and can be found on Tabor's tutorial webpage. The IVI driver comes free with the SE5082**.** However, you'll need the IVI engine and visa32.dll run time utilities to be able to use the IVI driver. The additional utilities can be downloaded for free from **Tabor**'s web site – www.taborelec.com.

Controlling the Instrument from Remote

In general, the SE5082 can be controlled from remote, using one of the following interfaces: USB, Ethernet or GPIB. Only USB remote interface cable is supplied with the instrument, so if you plan on using one of the other remote programming options, make sure you have a suitable cable to connect to your host computer. The following paragraphs describe how to connect and configure the SE5082 to operate from remote. The description is given for computers fitted with Windows 7, but installing software on other Windows versions is quite similar.

Connecting to a Remote interface

You can connect your **Tabor** SE5082 to GPIB, USB, or LAN adapters, depending on the application and requirements from your system. Installing interface adapters on your computer are not described in this manual, since the installation procedures for these adapters change frequently. Follow the instructions supplied with your particular adapter. Before proceeding with the remote interface installation, install an adapter card and follow the instructions as follows:

GPIB Connection

Direct connection between a host computer and a single device with GPIB is not recommended, since GPIB adapters are usually expensive and not really required for direct connection. Use a GPIB connection in cases where download speed is critical to the system or when you already have a GPIB system in place and you are adding the SE5082 as a GPIB device. The GPIB port is connected with a special 24-wire cable. Refer interconnection issues to your GPIB supplier. After you connect the SE5082 to the GPIB port, proceed to the GPIB Configuration section in this chapter for instructions how to select a GPIB address.

USB Connection

Direct connection between a single-host computer and a single

device with USB is recommended, as this does not require any specific considerations or device configurations. Just connect your SE5082 to your PC using a standard USB cable and the interface will self-configure. After you connect the SE5082 to the USB port, proceed to the USB Configuration section in this chapter for instructions how to install the USB driver.

LAN Connection

Direct connection between a single-host computer and a single device with 10/100/1000 Base-T is possible. If your site is already wired, connect the SE5082 via twisted-pair Ethernet cable. Be sure to use twisted-pair wires designed for 10/100/1000 Base-T network use (phone cables will not work). Refer interconnection issues to your network administrator. After you connect the SE5082 to the LAN port, proceed to the LAN Configuration section in this chapter for instructions how to set up LAN parameters.

Selecting a Remote interface

The SE5082 is supplied by the factory with the active remote interface set to USB. If you intend to use USB connection, all you need to do is connect your USB cable and proceed with the USB Configuration instructions as given in this chapter to install the USB driver and configure the USB port (first connection only). If you already used your instrument in various platforms and want to re-select your interface, you need to access the Select Interface screen as shown in Figure 2-1. To access this screen, press the Utility key in the Control group, then select the Remote Interface soft key and then the Select Interface soft key button.

Use the curser keys left and right to point to the required interface option, then press Enter. The new interface will initialize and the icon at the top will be updated and will flag the active interface option.

The interface icon is always displayed at the top of the screen, so if you are not sure which of the interfaces is selected, compare the following icons to what you have on the screen:

Designates GPIB interface is selected and active. GPIB configuration is required to communicate with your PC.

Designates USB interface is selected and active. First connection requires USB configuration and software driver installation to communicate with your PC.

Designates LAN interface is selected and active. LAN configuration is required to communicate with your PC.

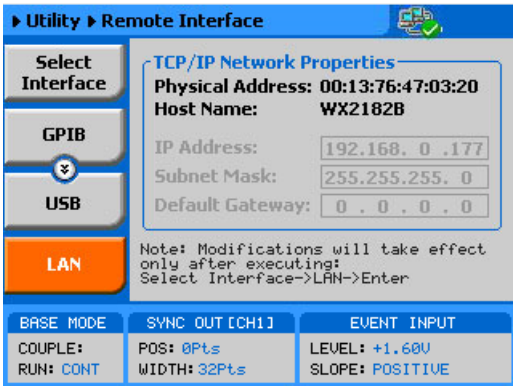

Figure 2-1, Selecting a Remote Interface

GPIB Configuration

GPIB configuration requires an address setting only. If you intend to use more than one instrument on the bus, you have to make sure each device has a unique address setting. GPIB address is programmed from the front panel Utility menu as shown in Figure 2-2. To access this screen, press the Utility control button, then press the Remote Interface soft key button and then select the GPIB soft key button. The display will be updated with the current GPIB address. The default address is 4. To modify the address, press the Enter key and use the dial or keypad to select the new address. Press Enter for the SE5082 to accept the new address setting.

Note

Configuring your GPIB address setting does not automatically select the GPIB as your active remote interface. Setting a remote interface is done from the Select interface menu. Information how to select an Interface is given earlier in this chapter.

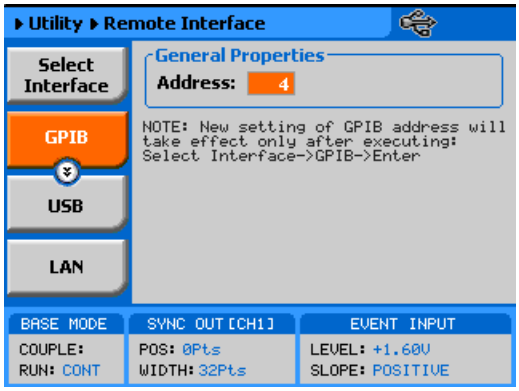

Figure 2-2, GPIB Configuration Screen

USB Configuration

Make sure you select USB as the Remote Interface from the front panel of the SE5082. Otherwise, the USB requires no front panel configuration parameters. Simply connect your Tabor SE5082 to your PC using a standard USB cable and the interface will self-configure. The first time you connect the generator to your PC, the new hardware will be detected and the icon, as shown in Figure 2-3, will appear. Double clicking on the Icon will open the dialog box as shown in Figure 2-4.

Figure 2-3, USB Device Detected, 1st Message

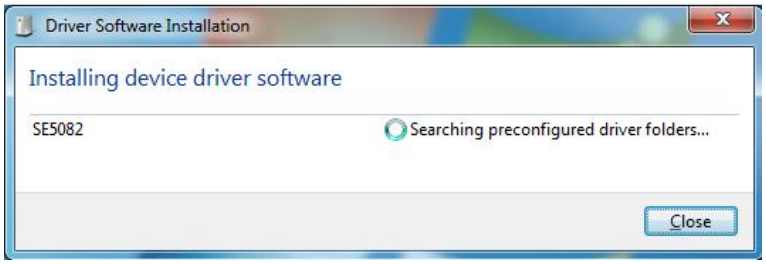

Figure 2-4, USB Device Detected, 2nd Message

After a few moments, a new message will appear as shown in Figure 2-5 confirming the installation of the device driver. Note that NI-VISA must be installed on your computer for the driver to install automatically.

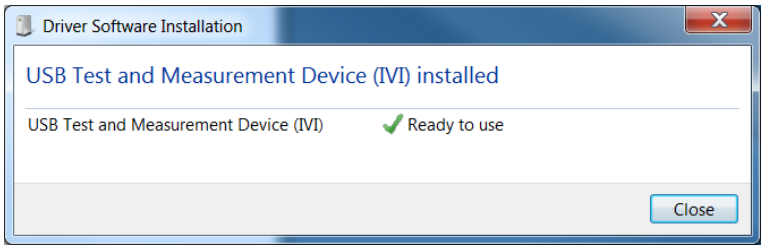

Figure 2-5, Found New Hardware Wizard

Figure 2-6 shows an example of the device manager where the Tabor SE5082 USB Waveform Generator has been found and the software driver installed.

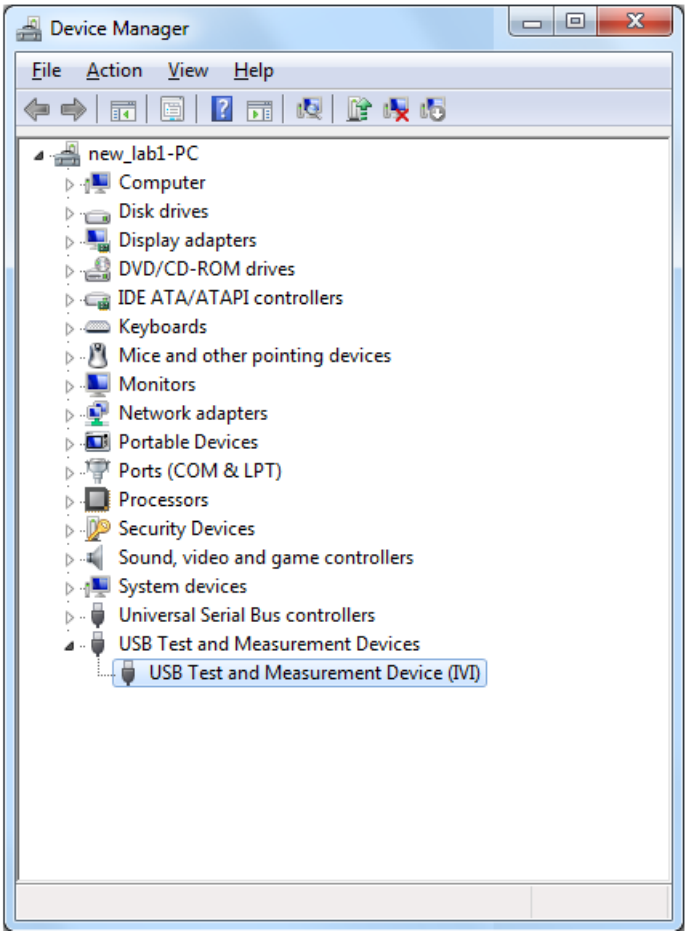

Figure 2-6 , Model SE5082 Configured for USB Operation

Note

Configuring your USB setting does not automatically select the USB as your active remote interface. Setting a remote interface is done from the Select Interface menu. Information how to select an Interface is given earlier in this chapter.

LAN Configuration

There are several parameters that you may have to set, in order to establish network communications using the LAN interface. Primarily, you'll need to establish an IP address. You may need to contact your network administrator for help in establishing communications with the LAN interface. To change LAN configuration, you need to access the LAN screen, as shown in Figure 2-7.

To access this screen, press the Utility control button, then press the Remote Interface soft key button and select the LAN soft key button. The display will be updated with the current LAN settings.

Note: there are some parameters that are shown on the display that cannot be accessed or modified: Physical Address and Host Name. These parameters are set in the factory and are unique for this specific product. The only parameters that can be modified are the IP Address, the Subnet mask and the Default gateway. Correct setting of these parameters is essential for correct interfacing with the LAN network. Description of the LAN settings and information on how to change them is given in the following.

Note

Configuring your LAN setting does not automatically update LAN parameters or select the LAN as your active remote interface. Setting a remote interface is done from the Select interface menu. LAN parameters will be updated only after re-selecting the LAN interface from the Select Interface menu. Information how to select an Interface is given earlier in the chapter.

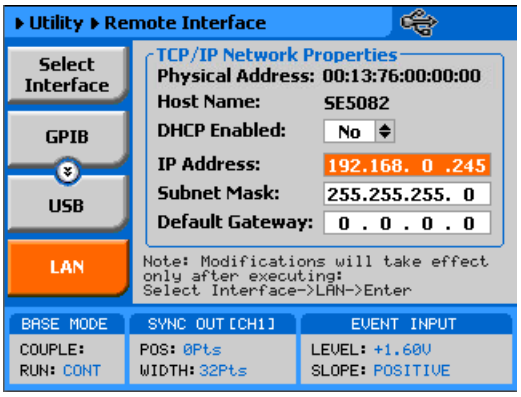

Figure 2-7, LAN Configuration Screen

There are three LAN parameters in this screen that can be modified and adjusted specifically to match your network setting, as described below. Consult your network administrator for the setting that will best suit your application.

- **IP address** - The unique, computer-readable address of a device on your network. An IP address typically is represented as four decimal numbers separated by periods (for example, 192.160.0.233). Refer to the next section: Choosing a Static IP Address.
- **Subnet mask** - A code that helps the network device determine whether another device is on the same network or a different network.
- **Gateway IP** - The IP address of a device that acts as a gateway, which is a connection between two networks. If your network does not have a gateway, set this parameter to 0.0.0.0.

Choosing a Static IP Address

For a Network Administered by a Network Administrator
If you are adding the Ethernet device to an existing Ethernet network, you must choose IP addresses carefully. Contact your network administrator to obtain an appropriate static IP address for your Ethernet device. Also have the network administrator assign the proper subnet mask and gateway IP.

For a Network without a Network Administrator
If you are assembling your own small Ethernet network, you can choose your own IP addresses. The format of the IP addresses is determined by the subnet mask. You should use the same subnet mask as the computer you are using with your Ethernet device. If your subnet mask is 255.255.255.0, the first three numbers in every IP address on the network must be the same. If your subnet mask is 255.255.0.0, only the first two numbers in the IP addresses on the network must match.

For either subnet mask, numbers between 1 and 254 are valid choices for the last number of the IP address. Numbers between 0 and 255 are valid for the third number of the IP address, but this number must be the same as other devices on your network if your subnet mask is 255.255.255.0.

Table 2-1 shows examples of valid and invalid IP addresses for a network using subnet mask 255.255.255.0. All valid IP addresses contain the same first three numbers. The IP addresses in this table are for example purposes only. If you are setting up your own network, you probably do not have a gateway, so you should set these values to 0.0.0.0.

Table 2-1, Valid and Invalid IP Addresses for Subnet Mask 255.255.255.0

IP Address	Comment
123.234.45.211	Valid.
123.234.45.213	Valid. The first three numbers match the previous IP address. The fourth number must be a unique number in the range of 1 to 254.
123.202.45.214	Invalid. Second number does not match the previous IP addresses. The first three numbers must match on all IP addresses with subnet mask 255.255.255.0.
123.234.45.0	Invalid. The first three numbers are valid but the fourth number cannot be 0.
123.234.45.255	Invalid. The first three numbers are valid but the fourth number cannot be 255.

TIP

To find out the network settings for your computer, perform the following steps:

1. **Open a DOS prompt.**
2. **Type IPCONFIG.**
3. **Press <Enter>.**

If you need more information, you can run ipconfig with the /all option by typing IPCONFIG /all at the DOS prompt. This shows you all of the settings for the computer. Make sure you use the settings for the LAN adapter you are using to communicate with the LAN device.

LAN Configuration Initialize (LCI)

The LCI Mechanism resets the instrument LAN settings to its default state. To access the LCI button, as shown in figure 2-9, press the utility button, select remote interface, then press the Top button and then scroll down.

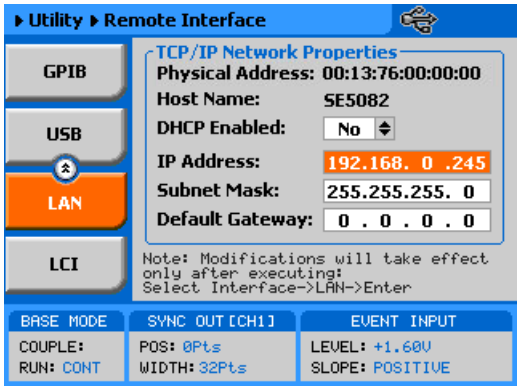

Figure 2-8 LAN Configuration Initilalize

The LAN default settings are shown below in Table 2-2

Table 2-2, LAN default Settings

Item	Value
IP Address Configuration: • DHCP • Static IP	 Enabled 192.168.0.223
Subnet Mask	255.255.255.0
Default Gateway	0.0.0.0
Hostname	TEDevice
Username	admin
Password	password

LAN eXtension for Instruments (LXI)

Aside from the front panel and the ability to control the unit via a remote interface using SCPI commands, the SE5082 offers a built-in web interface. The web interface provides a convenient way of controlling and configuring the instrument simply by entering the instruments IP address in your web browser. Once this is done a welcome page will appear as shown in Figure 2-8. .

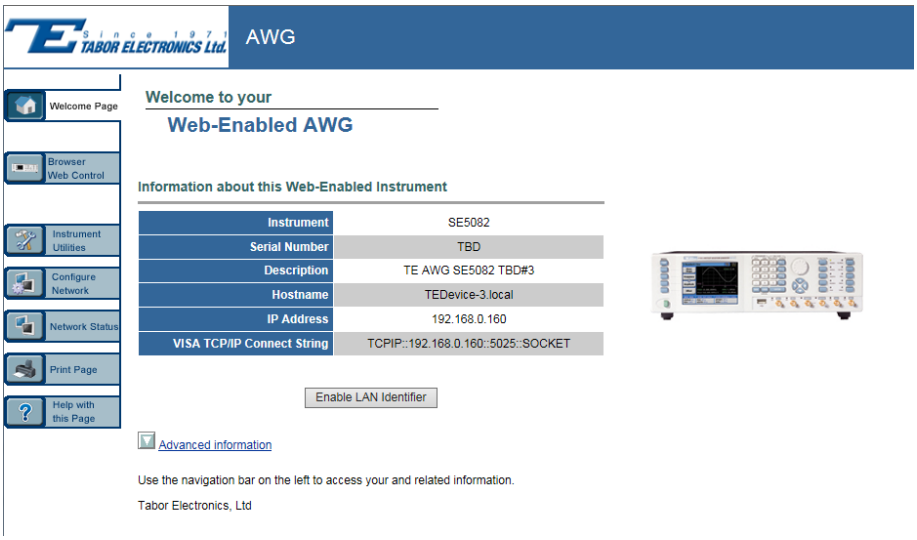

Figure 2-8, Welcome page for a Web-controlled SE5082

On the upper left side of the welcome page there are 6 tabs, Browser Web Control, Instrument Utilities, Configure Network, Network Status, Print Page, and Help with this Page. Each tab is described and explained in the following paragraph.

Note

When trying to access some of the functionality of the web interface the user will be prompted to enter a user name and password. The factory default user name is "admin" and password is "password". You are strongly advised to change the user name and password to restrict the access to the instrument only to authorized users. The login settings can be changed under the Configure Network tab.

Browser Web Control - opens a screen where you can choose how to control and access the instrument, whether remote front panel or remote programming.

The remote front panel link opens a new browser window where a simulation of the instruments front panel will be presented. Note, you will be prompted to install ActiveX plug-in and a Tabor Electronics application. The user can then control the instrument through this remote front panel as if the instrument was in front of him. Detailed information on how to use the instrument is given in Chapter 3 of this manual.

The remote programming link opens a new browser window with a command editor field for entering the SCPI commands and a Response field for reading the instrument's response to queries. Detailed information on programming the instrument and its SCPI commands is given in chapter 4 of this manual.

Instrument Utilities - The instrument utilities tab allows the user to view different properties of the instrument. These include, the installed options, Firmware version, and calibration status.

Configure Network - To configure the various network properties click on the Configure Network tab. The main screen will show the current network configuration. To change the configuration click on the Modify Configuration button. Here you will be able to modify the different properties of the network environment or restore the factory default settings. Furthermore, this is where the factory default login settings can be changed.

Network Status – The network status tab provides an updated information on the network speed and packets exchange.

Print Page – enables the user to print out a copy of the current screen being displayed.

Help with this page – open a help screen with information on the current page.

Master Slave Operation

For applications that require more than 2 channels the built in synchronization feature allows for synchronizing two **SE5082** to create a fully synchronized four channel system. The procedure is described in the following paragraphs.

Connecting the Instruments

Two instruments can be synchronized only when cross-connected with a designated synchronization cable. To connect the cable, turn the power off on both instruments. Locate the rear-panel Master/Slave connector and attach each side of the cable to these connectors. Note that the connectors do not lock into position without screwing in the bolts on each side. Use flat tip screwdriver to latch the cables into their receptacle housing and make sure the connectors are straight and firmly locked into place. After you connect the cable you can power the instruments on again.

Resetting the Two Instruments

Once the two instruments have been connected using the synchronization cable the next step is to perform a factory reset on both units. Perform the following steps in both units:

1. Press the Utility Button
2. Press the Factory Reset soft key button
3. Use the left key to choose yes
4. Press the enter button

Selecting a Master

After the two instruments have been set to their factory default setting, the next step is to select one instrument as Master and the other as slave. Perform the following steps:

1. On the designated slave unit, press the output button
2. Press the X-Instr. soft key button
3. Use the dial or the up/down keys to scroll to the Role option
4. Press the Enter button
5. Change the field to “Slave”
6. Press the Enter button
7. Turn on the channel outputs
8. On the designated master unit, press the output button
9. Press the X-Instr. soft key button
10. Use the dial or the up/down keys to scroll to the State option
11. Press Enter, change the field to “ON” and press Enter again
12. Turn on the channel outputs

Operating Synchronized Instruments

When operating two synchronized instruments, there are some important limitations to be familiar with in order to achieve optimal performance.

1. Connecting the synchronization cable is pre-requisite however, additional steps must be performed to set one unit in master mode and select the other as slave.

2. Before activation of the synchronization, both Master and Slave AWGs must be set to the same sample clock, run mode and function mode.

3. The synchronization must be activated from the Master unit.

4. Each instrument can have a unique set of waveforms, active segment, amplitude, offset parameters and markers. When using sequence mode both master and slave must have a sequence table with the same number of steps.

5. Trigger signal is applied to the master input and is common to both master and slave units.

6. The waveform can be delayed from the master output by a predefined number of sample clocks or by a set amount of time. Press the output button and then press the X.-Instr. Soft key button. There are two fields, Offset and Skew. The offset field defines the number of sample clock cycles, or waveform points that the instrument will hold off before it will start generating the output waveform. The standard resolution is 16 waveform cycles (8 with option 1) and the time is defined as 16 clock cycles (8 with option 1), which is set on the master unit. The offset is used for coarse tuning. The Skew defines fine offset between the instruments and is set in unit of time. The skew can be varied from -5ns to +5ns at a 10ps resolution and is used for fine tuning.

This page was intentionally left blank

Chapter 3
Using the Instrument

Title **Page**

Overview

This chapter contains information about how to operate the Tabor SE5082. Operation is divided into two general categories: basic bench operation, and remote operation (GPIB, USB and ENET). Basic bench operation, which is covered in this section, describes how to operate the arbitrary waveform generator using front panel sequences.

The following paragraphs describe the various modes of operation and give examples of how to program the SE5082. The manual is organized by instrument function, and instructions are given in each paragraph on how to use this function from the front panel.

Inter-Channel Dependency

The SE5082 has two completely independent output channels. Each channel can be programmed to output a unique set of functions and parameters, which can be generated using separate run modes. As each channel is completely separate, there is no influence from one channel on its adjacent channel. There are a few controls that are shared by the two channels including: trigger and event inputs, sync output and remote interface connections.

On the other hand, there are many applications that require two (or more) channels. These are normally expected to share output frequency, sample clock and retain a controllable phase dependency. In this case, the SE5082 can be placed in a common sample clock feed mode and transform its operation to a dual channel generator where run mode and output function are shared, but each channel remains independent to generate different waveforms and amplitudes.

Output Termination

During use, output connectors must be properly terminated to minimize signal reflection or power loss due to impedance mismatch. Proper termination is also required for accurate amplitude levels at the output connectors. Use 50Ω cables and terminate the main and SYNC cables with terminating resistors. Always place the 50 Ω termination at the far end of the cables.

The SE5082 has two output connectors for each channel, marked “Normal” and “Inverted”. If single-ended output is required, the other output must be terminated with a 50 Ω plug termination, right at the output connector. The DC is generated differentially, but the AC output path is available through the Normal output only as a single-ended signal and therefore, termination on the other output is not a requirement.

Note that the display reading of the amplitude level is calibrated to show the actual level on the load, when the load impedance is roughly 50 Ω. In cases where the load has different impedance, the display will indicate a reading that does not match the actual amplitude level on the load.

Input / Output Protection

The SE5082 provides protection for the internal circuitry that is connected to input and output connectors. Refer to the specifications in Appendix A to determine the level of protection associated with each input or output connector.

WARNING

The outputs can only be connected to resistive loads. Connecting the SE5082 to inductive or capacitive loads may damage the output and void the warranty on the instrument.

Power On/Reset Defaults

The SE5082 default power on state is the factory default setting. Every time the unit powers on the factory default settings are restored. User can change the power on state from factory default to last setting in the Power on State field under the Utility>System menu shown in figure 3-1. When set to Last Setting the SE5082 utilizes non-volatile memory backup that automatically stores the last setup before the generator has been turned off. Every time you turn on the instrument, the non-volatile memory updates the front panel setting with modes, parameters and waveforms from its last setting with only one exception. For safety reasons, the outputs remain off even if they were turned on before powering down the SE5082.

After power on, the instrument displays information messages and updates the display with the last setup information. The SE5082 can always be reset to its default values. Information on how to restore default parameters is given below.

If you are not yet fully familiar with front panel operation of the SE5082, you may find yourself locked into a "dead-end" situation where nothing operates the way it should. The fastest way to restore the generator to a known state is by resetting the instrument to factory defaults.

As seen below in Figure 3-1 you can reset parameters to factory defaults as follows:

1. Press the Utilities control key
2. Press the Factory Rests soft key
3. Select Yes or No in the pop-up dialog box

Table 3-1 summarizes factory defaults for the most common parameters. A complete list of all parameters and their defaults, as well as their maximum and minimum values is given in the Programming Reference chapter.

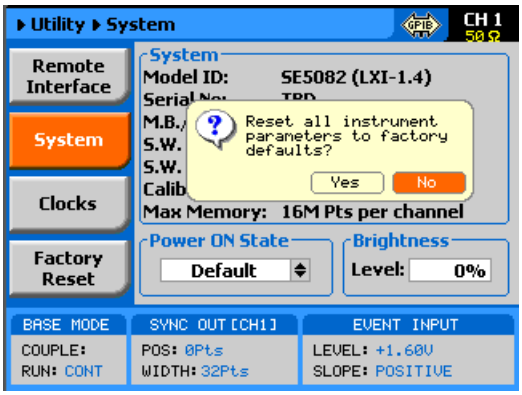

Figure 3-1, Reset SE5082 to Factory Defaults

Table 3-1, Default Conditions after Reset

Function / Parameter	Default
Outputs State	Off
SYNC State	Off
Operating Mode	Continuous
Active Channel	1
Marker Output State	Off
Output Function	Standard
Output Function Shape	Sine
Standard Wave Frequency	10 MHz
User Wave Sample Clock	1 GSa/s
Sample Clock Feed	Separate
Sample Clock Source & Reference	Internal
Amplitude	V
Trigger Slope:	Positive
Trigger Level:	1.6 V
Trigger Source:	External

Controlling the SE5082

Controlling SE5082 function, modes and parameters is simply a matter of pressing once or twice on the appropriate button as described in the following paragraphs. Refer to Figure 3-2 throughout this description.

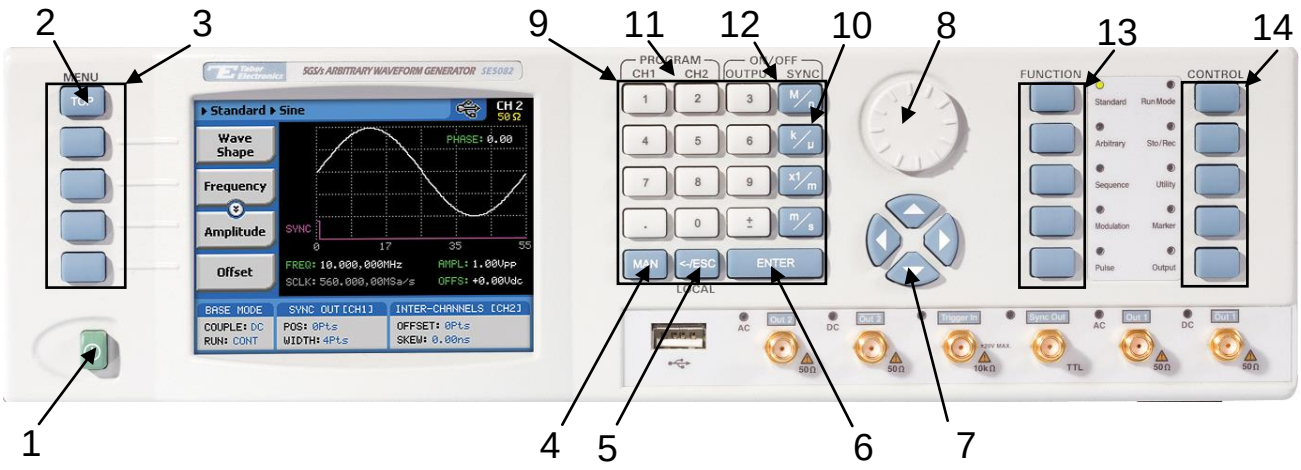

Figure 3-2, SE5082 Front Panel Operation

Note

The index in the following paragraphs point to the numbered arrows in Figure 3-2.

1. ***Power Switch*** – Toggles SE5082 power ON and OFF
2. ***Menu Top*** – For certain functions, selects the root menu. This button is disabled during parameter editing
3. ***Menu Soft Keys*** – These select parameter to be audited

   These buttons are disabled during parameter editing
4. ***MAN*** – Manual trigger button, used in lieu of an external trigger signal
5. ***<-Esc / (Local)*** – Has two functions:
   1) When in edit mode, cancels edit operation, restore last value and returns to the main function screen
   2) When operating the SE5082 from a remote interface, none of the front panel buttons are active except the Local button. When pressed, it restores control to front panel buttons

6. ***Enter*** – Has two functions:
    1) When multiple parameters are displayed on the screen, the cursor and the dial scroll through the parameters. Pressing Enter selects the parameter for edit. After the parameter has been modified, the Enter button locks in the new variable and releases the buttons for other operations
    2) When a parameter is modified, Enter can be used to replace the x1 suffix key
7. ***Cursor UP, Down, Left and Right*** – Has two functions:
    1) When multiple parameters are displayed on the screen, the cursor and the dial scroll through the parameters.
    2) When parameter is selected for editing, cursor buttons right or left move the cursor accordingly. Cursor buttons up or down modifies parameter value accordingly.
8. ***Dial*** – Has similar functionality as the cursor UP and Down keys.
9. ***Numeral keypad*** – These keys are used for modifying an edited parameter value.
10. ***Parameter Suffixes*** *(M/n, k/μ, x1/m and m/s)* – These keys are used to place suffix at the end of the parameter. They are also used for terminating an edit operation.
11. ***Program*** **CH1, CH2** – Use Program CH1 to modify the screen to display channel 1 parameters. Use Program CH2 to modify the screen to display channel 2 parameters. These keys can be used only when the SE5082 is not in edit mode.
12. ***ON/OFF Output, Sync*** – These keys can be used only when the SE5082 is not in edit mode. The Output ON/OFF toggles output waveform, at the output connector, ON and OFF. The Sync ON/OFF toggles the sync waveform, at the SYNC output connector, ON and OFF.
13. ***Function*** – These keys select one of 5 function menus that the SE5082 can generate. The output functions are: Standard, Arbitrary, Sequenced, Modulated and Pulse. An LED lights next to the selected function.
14. ***Control*** – These keys select control menus that program the SE5082 operating modes. The control menus are: Run Mode, Store/Recall, Utility, Markers and Output.

SE5082 Front Panel Menus

The SE5082 has over 150 parameters that control functions, modes, waveforms and auxiliary functions and an enormous combination of these parameters to define an appropriate function for a specific application. Due to the complexity of the product, the functions were divided into logical groups and sub-groups and access to these groups is provided using the soft key menus.

On the right side of the instrument there are two groups of buttons: Function and Control. The function buttons select one of five waveforms to be generated at the output connectors and cause the display to show parameters that are associated with the output function.

The other group is marked Control. These buttons control operating features and auxiliary functions that are complementary to the Function buttons. On the left, there is a group of five soft-key buttons. These are used for accessing a specific parameter associated with the output function.

Tables 3-2 and 3-3 list all menus that are accessible from the Function group and Table 3-4 lists all menus that are accessible from the Control group. Short explanation of the various menus is also given. Note that the descriptions in these tables are given to provide a general understanding of what is available in terms of operating the instrument. For detailed instructions, check the appropriate section of the manual.

Table 3-2, Front Panel Function Menus

Soft Key	Function Menu	2nd Level Menu	3rd Level Menu	Notes
Top	Standard			
A		Wave Shape	Waveform list	Select from a wave shapes list
B		Frequency	Value	Programs standard waveform frequency
C		Amplitude	Value	Programs output amplitude
(C) *		Power	Value	Available for AC coupled output path only
D		Offset	Value	Programs output amplitude offset
(D)		Phase	Value	Parameters depend on selected shape
Top	Arbitrary			
A		Sample Clock	Value	Programs sample clock frequency
B		Amplitude	Value	Programs output amplitude
C		Offset	Value	Programs output amplitude offset
D		Active Segment	Value	Selects the active arbitrary waveform segment
↓D		Jump Event	BUS / External	Selects which source will cause segment transitions
↓D		Jump Timing	Coherent / Immediate	Defines how segments transition
↓D		Wave Composer	Waveform studio	Provides access to the waveform composer
↓D		Delete Segments	Value	

* () Denotes conditional menu, not always shown. Available in conjunction with a specific mode only

**↓ Denotes you have to scroll down to access the menu. Scroll using the arrows up or down or the dial.

Table 3-2, Front Panel Waveform Menus (continued)

Soft Key	Function Menu	2nd Level Menu	3rd Level Menu	Notes
Top	Sequenced			
A		Active Sequence	String	View and select the active sequence name
B		View Table	Table view and edit	Displays current sequence step, and properties
C		Sample Clock	Value	Programs the sequence sample clock
D		Amplitude	Value	Programs the waveform amplitude
↓D		Offset	Value	Programs the waveform offset
↓D		Config	Seq. Configuration	Opens a new set of configuration menus.
	Config			
A		Sequence Type	Normal / Advanced	Selects between normal or advanced sequencing
B		Advance Mode	Auto / Once / Step	Selects the advance mode
(C)		Once Count	Value	Sequence loop counter for the once advance mode
C		Select Source	BUS / External	Defines the source that will select a sequence
D		Select Timing	Coherent / Immediate	Defines how the sequence is switched
↓D		Jump Event	BUS / Event	Selects the source to detect a jump event of a step
↓D		Sync Lock	Value	Defines with which segment the Sync signal is generated
Top	Modulation			
A	OFF	Modulation Type	OFF/AM/FM/Sweep/Chirp/ FSK/ASK/ Freq Hop/Ampl Hop	Select from a modulation type list
B		CW Frequency	Value	Programs the modulation carrier frequency
C		Carrier Shape	Sine/Triangle/Square	Selects the shape of the carrier waveform
C		Amplitude	Value	Programs output amplitude
↓D		Offset	Value	Programs output amplitude offset
A	AM	Modulation Type	OFF/AM/FM/Sweep/Chirp/ FSK/ASK/ Freq Hop/Ampl Hop	Select from a modulation type list
B		Modulation Shape	Sine/tri/squ/ramp	Programs the modulating waveform shape
C		Modulation Depth	Value	Programs the modulation depth
D		Modulation Freq	Value	Programs the modulating frequency
↓D (**)		CW Frequency	Value	Programs the modulation carrier frequency
↓D		Carrier Shape	Sine/Triangle/Square	Selects the shape of the carrier waveform
↓D		Amplitude	Value	Programs output amplitude
↓D		Offset	Value	Programs output amplitude offset
A	FM	Modulation Type	OFF/AM/FM/Sweep/Chirp/ FSK/ASK/ Freq Hop/Ampl Hop	Select from a modulation type list
B		Modulation Shape	Sine/tri/squ	Programs the modulating waveform shape
C		CW Frequency	Value	Programs the modulation carrier frequency
D		Freq. Deviation	Value	Programs FM frequency deviation
↓D		Modulation Freq	Value	Programs the modulating frequency

* () Denotes conditional menu, not always shown. Available in conjunction with a specific mode only

**↓ Denotes you have to scroll down to access the menu. Scroll using the arrows up or down or the dial.

Table 3-2, Front Panel Waveform Menus (continued)

Soft Key	Function Menu	2nd Level Menu	3rd Level Menu	Notes
↓D		Marker	Value	Programs marker frequency position
↓D		Carrier Shape	Sine/Triangle/Square	Selects the shape of the carrier waveform
↓D		Amplitude	Value	Programs output amplitude
↓D		Offset	Value	Programs output amplitude offset
A	Sweep	Modulation Type	OFF/AM/FM/Sweep/ Chirp/FSK/ASK/ Freq Hop/Ampl Hop	Select from a modulation type list
B		Sweep Type	Linear/Logarithmic	Selects the sweep step
C		Direction	Up/Down	Selects the sweep direction
D		Start Frequency	Value	Programs the start frequency
↓D		Stop Frequency	Value	Programs the stop frequency
↓D		Sweep Time	Value	Programs the sweep time
↓D		Marker	Value	Programs marker frequency position
↓D		Carrier Shape	Sine/Triangle/Square	Selects the shape of the carrier waveform
↓D		Amplitude	Value	Programs output amplitude
↓D		Offset	Value	Programs output amplitude offset
A	Chirp	Modulation Type	OFF/AM/FM/Sweep/ Chirp/FSK/ASK/ Freq Hop/Ampl Hop	Select from a modulation type list
B		Sweep Type	Linear/Logarithmic	Selects the sweep step
C		Direction	Up/Down	Selects the sweep direction
D		AM Shape	Linear/Logarithmic	Selects the amplitude steps
↓D		AM Direction	Up/Down	Selects the amplitude direction
↓D		Start Frequency	Value	Programs the start frequency
↓D		Stop Frequency	Value	Programs the stop frequency
↓D		Pulse Width	Value	Programs width of entire chirp pulse
↓D		Pulse Repetition	Value	Programs chirp pulse repetition rate
↓D		AM Depth	Value	Programs the modulation depth
↓D		Marker	Value	Programs marker frequency position
↓D		Carrier Shape	Sine/Triangle/Square	Selects the shape of the carrier waveform
↓D		Amplitude	Value	Programs output amplitude
↓D		Offset	Value	Programs output amplitude offset
A	FSK	Modulation Type	OFF/AM/FM/Sweep/ Chirp/FSK/ASK/ Freq Hop/Ampl Hop	Select from a modulation type list
B		FSK Data	Data	Program the sequence of frequencies
C		CW Frequency	Value	Programs the modulation carrier frequency
D		Shifted Frequency	Value	Programs the shifted frequency of logic level ‘1’
↓D		Baud	Value	Programs the bit rate
↓D		Marker	Value	Programs marker position

* () Denotes conditional menu, not always shown. Available in conjunction with a specific mode only

** ↓ Denotes you have to scroll down to access the menu. Scroll using the arrows up or down or the dial.

Table 3-2, Front Panel Waveform Menus (continued)

Soft Key	Function Menu	2nd Level Menu	3rd Level Menu	Notes
↓D		Carrier Shape	Sine/Triangle/Square	Selects the shape of the carrier waveform
↓D		Amplitude	Value	Programs output amplitude
↓D		Offset	Value	Programs output amplitude offset
A	ASK	Modulation Type	OFF/AM/FM/Sweep/ Chirp/FSK/ASK/ Freq Hop/Ampl Hop	Select from a modulation type list
B		ASK Data	Data	Program the sequence of frequencies
C		CW Frequency	Value	Programs the modulation carrier frequency
D		Start Amplitude	Value	Programs the initial amplitude level
↓D		Shifted Amplitude	Value	Programs the shifted amplitude level of logic level '1'
↓D		Baud	Value	Programs the bit rate
↓D		Carrier Shape	Sine/Triangle/Square	Selects the shape of the carrier waveform
↓D		Marker	Value	Programs marker position
↓D		Offset	Value	Programs output amplitude offset
A	Freq Hop	Modulation Type	OFF/AM/FM/Sweep/ Chirp/FSK/ASK/ Freq Hop/Ampl Hop	Select from a modulation type list
B		Hop Data	Data	Program the frequency hop table
C		Dwell Type	Fixed/Variable	Selects fixed or variable dwell time
D		Dwell time	Value	Programs the lapse of time for a hop step
↓D		Marker	Value	Programs marker position
↓D		Carrier Shape	Sine/Triangle/Square	Selects the shape of the carrier waveform
↓D		Amplitude	Value	Programs output amplitude
↓D		Offset	Value	Programs output amplitude offset
A	Ampl Hop	Modulation Type	OFF/AM/FM/Sweep/ Chirp/FSK/ASK/ Freq Hop/Ampl Hop	Select from a modulation type list
B		Hop Data	Data	Program the frequency hop table
C		CW Frequency	Value	Programs the modulation carrier frequency
D		Dwell Type	Fixed/Variable	Selects fixed or variable dwell time
↓D		Dwell time	Value	Programs the lapse of time for a hop step
↓D		Carrier Shape	Sine/Triangle/Square	Selects the shape of the carrier waveform
↓D		Marker	Value	Programs marker position
↓D		Offset	Value	Programs output amplitude offset

* () Denotes conditional menu, not always shown. Available in conjunction with a specific mode only

** ↓ Denotes you have to scroll down to access the menu. Scroll using the arrows up or down or the dial.

Pulse Menus

The menus that are described in Table 3-2 are somewhat different than the pulse menus as they are relatively straightforward as each menu controls a separate, single parameter. There is a slight difference when it comes to operating a pulse generator; simple pulse generation requires just a few parameters such as: period, pulse width, and hi and low levels however, it gets complicated when the application requires transition times, pulse delays and different level control. For this purpose, the start-up menu for the pulse function is the simplest and provides access to the commonly used parameters, but as you select more complex modes, other menus are automatically turned on, allowing access to mode parameters.

The following table is presented not to show you how to use the pulse generator, but to give you a general overview of all menus available for this function. A detailed description on the pulse function is given later in this chapter.

Table 3-3, Front Panel Pulse Function Menus

Soft Key	Function Menu	2nd Level Menu	3rd Level Menu	Notes
Top	Pulse			
A		Period	Value	Programs pulse repetition rate
B		Width	Value	Programs the pulse width
C		High Level	Value	Programs the pulse high level
(C)*		Delay	Value	Programs delay for the delayed/double pulse mode
(C)		Amplitude	Value	Programs amplitude for the level control mode
D		Low Level	Value	Programs the pulse low level
(D)		Offset	Value	Programs offset for the level control mode
↓(D)**		Leading Edge	Value	Programs rise time for the linear/symmetrical transition type
↓(D)		Trailing Edge	Value	Programs fall time for the linear/symmetrical transition type
↓D		Config	Pulse Configuration	Opens a new set of configuration menus.
		Pulse Composer	Pulse Composer table	Opens a table for creating pulses and pulse trains
		Pattern	Pattern Composer	Opens a menu for crating Patterns
	Config			
A		Pulse Parameters	Time / Percent	Pulse parameter settings in time or percent of period
B		Pulse Mode	Single / Delayed / Double	Programs pulse mode
C		Polarity	Normal / Complemented / Inverted	Selects pulse polarity
D		Transition Type	Fast / Linear	Programs pulse transitions
↓D		Level Control	High / Low / Amplitude / Offset / Positive / Negative	Defines how the pulse levels are programmed

* () Denotes conditional menu, not always shown. Available in conjunction with a specific mode only

** ↓ Denotes you have to scroll down to access the menu. Scroll using the arrows up or down or the dial.

Table 3-3, Front Panel Pulse Function Menus (continued)

Soft Key	Function Menu	2nd Level Menu	3rd Level Menu	Notes
	Pattern			
A		Type	PRBS7/9/11/15/23/31/ USER	Select pattern type
B		High Level	Value	Programs the pattern high level
C		Low Level	Value	Programs the pattern low level
D		Number of levels	Value	Programs the number of levels
↓D		Baud	Value	Programs the bit rate
↓D		Length	Value	Programs the length of the pattern
↓D		Preamble	Value	Programs the preamble bit

* () Denotes conditional menu, not always shown. Available in conjunction with a specific mode only

** ↓ Denotes you have to scroll down to access the menu. Scroll using the arrows up or down or the dial.

Control Menus

The Control menus buttons, unlike the Function menus buttons, display menus and parameters that modify the output to various run modes, turns the outputs on and off, programs markers and provides access to utility functions. You may also use these buttons to store and recall instrument settings and waveforms.

Table 3-4 lists all menus that are accessible from the Control group. A short explanation of the various menus is also given. Note that the descriptions in these tables are provided to give you a general overview of what is available in terms of operating the instrument. For detailed instructions, check the appropriate section of the manual.

Table 3-4, Front Panel Control Menus

Soft Key	Control Menu	2nd Level Menu	3rd Level Menu	Notes
	Run Mode			
A		Continuous	Enable Mode	Selects between Self Armed and Armed
			Enable Source	Selects between BUS, Event or Trigger inputs
			Event Input Level	Programs the event threshold level
			Event Input Slope	Selects Positive, Negative or, Either transition
B		Trigger	Trigger Mode	Selects between Normal and Override
			Trigger Source	Selects between BUS, Event or Trigger inputs
			Trigger Level	Programs the trigger threshold level
			Trigger Slope	Programs the trigger slope
			Trigger Delay	Programs the delayed trigger and re-trigger timer
(B)			Trigger Timer	Programs the internal timer period (INT source only)
			Trigger Input	Programs trigger input impedance (Ext trigger only)
			Trigger Count	Programs the burst counter value
			Smart Trigger >time	Programs the more than pulse period value
			Smart Trigger <time	Programs the less than pulse period value
			Trigger Hold off	Programs trigger hold off time

* () Denotes conditional menu, not always shown. Available in conjunction with a specific mode only

** ↓ Denotes you have to scroll down to access the menu. Scroll using the arrows up or down or the dial

Table 3-5, Front Panel Control Menus

Soft Key	Control Menu	2nd Level Menu	3rd Level Menu	Notes
C		Gated	Gated Level	Programs the gate threshold level
			Gated Slope	Selects Positive or Negative levels
	Sto/Rec			
A		Store	Target	Toggles between internal and USB
			Memory Cell	Selects the active memory cell
			Store	Defines what will be stored
B		Recall	Source	Selects the source to recall setups
			Memory Cell	Selects the active memory cell
C		List	Source	Toggles between internal and USB
(D)		Clear Cell	Value	In list view Select cell number to be deleted
D		Update		Executes the store or recall
	Utility			
A		Remote Interface	Select Interface	Provides access to interface setup: GPIB / USB / LAN
			GPIB	GPIB address setting
			USB	No parameter setting
			LAN	Programs LAN IP address, subnet mask and gateway
			LCI	Initializes LAN communication settings
B		System	Power on State	Selects between Default and Setup
			Brightness	Programs display brightness
C		Clocks		
			Sample Clock Feed	Programs separate or common feed
			Ext. SCLK Freq.	Prepares the external sample frequency range
			CH1 Source	Selects between Internal and External source
			CH1 Divider	Programs a dividing ratio for the external clock
			CH2 Source	Grayed out. Selection is automatic, depending on the Sample Clock Feed setting
			Ref Clock Source	Selects between Internal and External source
			Ref Clock Ext. Freq.	Prepares the external reference frequency range
D		Factory Reset		Resets all instrument parameters to factory defaults
A	Marker		State	Toggles marker outputs on and off
		CH1 – Marker 1/2	Delay	Programs the marker delay value
			Pos.	Programs the marker position value
			Width	Programs the marker width value
			High	Programs the marker high level value
			Low	Programs the marker low level value
B		CH2 – Marker 1/2	Delay	Programs the marker delay value
			Pos.	Programs the marker position value
			Width	Programs the marker width value
			High	Programs the marker high level value
			Low	Programs the marker low level value
C		Apply		Updates the marker memory with the above settings

* () Denotes conditional menu, not always shown. Available in conjunction with a specific mode only

** ↓ Denotes you have to scroll down to access the menu. Scroll using the arrows up or down or the dial

Table 3-5, Front Panel Control Menus (continued)

Soft Key	Control Menu	2nd Level Menu	3rd Level Menu	Notes
A	Output	Main Output	Output CH1	Toggles the main output ON and OFF
			Output CH2	Toggles the main output ON and OFF
B		SYNC Output	State	Toggles the SYNC output ON and OFF
			Source	Selects between CH1 and CH2
			Type	Selects between Pulse and WCOM
			Position	Programs the SYNC position value
			Width	Programs the SYNC width value
C		Sampling Mode	CH1	Selects DAC sampling method NRZ, NRTZ, RTZ,RF
			CH2	Selects DAC sampling method NRZ, NRTZ, RTZ,RF
D		X-Channel	Offset CH2->CH1	Programs CH2 2 offset value referenced to CH1
			Skew CH2->CH1	Programs CH2 2 skew value referenced to CH1
		X-Instr.	Role	Selects between Master and Slave
			State	Turns couple state on and off
			Offset	Programs an offset between two coupled instruments

Enabling the Outputs

For safety reasons, the main outputs default setting is OFF. The outputs can be turned on and off using either the hot keys, or the Output Menu. Please see Figure 3-3 and disable or enable the main outputs using the procedure described below. The same procedure can be used for enabling and disabling the SYNC output.

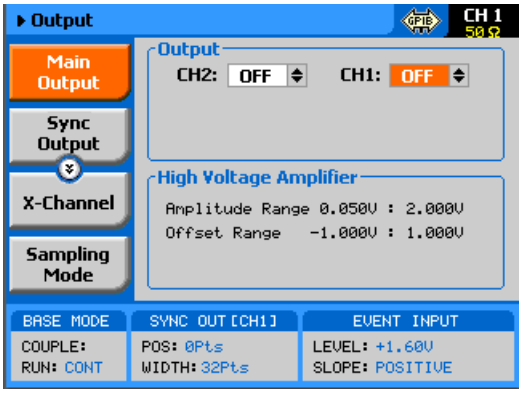

Figure 3-3, Enabling and Disabling the Outputs

1. Press the Output button in the Control group buttons. Observe the Output display, as it appears in Figure 3-3.
2. Use the dial or arrow keys to access the required field, which will appear orange.
3. To edit the field press Enter. The edited field will turn white with orange borders
4. Use the dial or arrow keys to change the field
5. Press Enter again to lock in the setting

Alternately, the outputs can be turned on and off using the hot-key buttons as marked above the keypad.

1. While not editing any parameter, select the channel you want to turn on using the PROGRAM CH1 or CH2 keys
2. Press ON/OFF OUTPUT or SYNC to toggle main and sync output on and off

Selecting a Function Mode

There are five function modes that the SE5082 can produce: Standard, Arbitrary, Sequenced, Modulated and Pulse. The standard, modulated and pulse waveforms are computed from equations and tables that are built into the program, but for the arbitrary and sequenced functions, waveform coordinates must first be loaded into its working memory. To generate one of the function types, simply press one of the Function buttons on the right side of the product. The display will change to display parameters that are associated with the selected function. For standard, modulated and pulse functions, the output will immediately generate the selected function. Arbitrary and sequenced waveforms will be generated only after waveform coordinates have been downloaded to the working memory.

Note

The picture in the SE5082 LCD display is an icon only. The actual output waveform may look entirely different.

Changing the Output Frequency

You should be careful not to confuse waveform frequency with sample clock frequency. Frequency applies to standard waveforms only and controls waveform frequency at the output connector. The sample clock frequency parameter is valid for arbitrary and sequenced waveforms only, and defines the frequency of the generator clocks data points.

Pulse waveform period is measured in units of seconds. Standard waveform frequency is measured in units of Hz and Arbitrary waveform sample clock frequency is measured in units of Sa/s (samples per second). The frequency of a given arbitrary waveform at the output connector is dependent on sample clock frequency, the number of data points, and other specific waveform definitions.

The frequency of the output waveform will change only if a standard waveform is generated. First, select a standard waveform as described earlier and then proceed with frequency modification.

Please see Figure 3-4 and modify frequency using the following procedure:

1. Press the Frequency soft key to select the frequency parameter.
2. Use the numeric keypad to program the new frequency value.
3. Press M/n, k/μ, x1/m or m/s to terminate the modification process.

Alternatively, you can modify the frequency value with the dial and arrow keys, but then the termination of the process is by pressing Enter only.

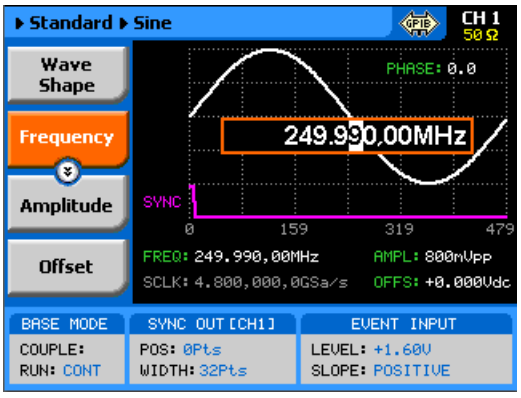

Figure 3-4, Modifying the Output Frequency

Note

If you use the dial or arrow keys to modify the frequency parameter, the output is updated immediately, as soon as you modify the parameter. The final value will be locked in as soon as you press Enter. If you choose to leave the old value, press Cancel to terminate the process and discard of any change made to this parameter.

Changing the Sample Clock Frequency

The frequency of the sample clock will affect the output waveform only if arbitrary or sequenced waveforms are generated. First, select an arbitrary waveform as described earlier and then proceed with sample clock frequency modification.

Please see Figure 3-5 to modify the sample clock using the following procedure:

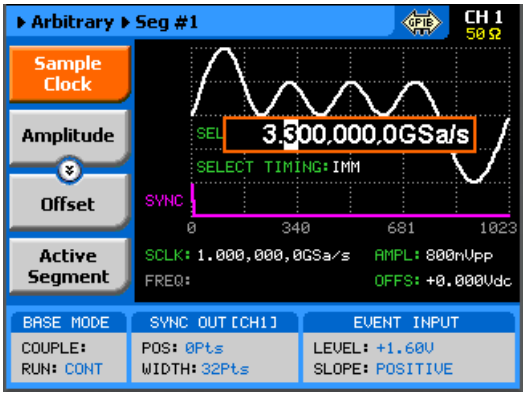

Figure 3-5, Modifying the Sample Clock Frequency

1. Press the Sample Clock soft key to select the Sample Clock parameter.
2. Use the numeric keypad to dial the new sample clock frequency value.
3. Press “M/n”, “k/μ”, “x1/m” or “m/s” to terminate the modification process. To select GSa/s enter the value in MSa/s (i.e. 1000 MSa/s will program 1 GSa/s).

Alternatively, you can modify the sample clock frequency value with the dial and arrow keys, but then the termination of the process is by pressing Enter only.

Note

If you use the dial or arrow keys to modify the sample clock frequency parameter, the output is updated immediately, as soon as you modify the parameter. The final value will be locked in as soon as you press Enter.

If you choose to leave the old value, press Cancel to terminate the process and to discard any change made to this parameter.

Programming the Amplitude and Offset

Output amplitude and offset can be programmed independently and separately for each channel. The active channel is displayed on the upper right corner of the LCD display.

When the display shows CH 1 on the upper right corner, you are currently programming channel 1 parameters. Keypads “1” and “2” are used as hot keys for channel selection. When not editing any parameter, press key “2” to program channel 2 parameters.

When the display shows CH 2 on the upper right corner, you can proceed with channel 2 programming.

The amplitude and offset parameters are duplicated in multiple screens. When changed for a specific function shape, the new value is updated on all screens, regardless of the selected function shape. Figure 3-6 shows an example of how to program the amplitude for channel 1. The same procedure can be used for channel 2 and for the offset parameter. Note that the example is given for a standard waveform function, but the same procedure will be used for all other functions.

1. Press the Standard button in the Function group.
2. Press Enter to edit the Amplitude value.
3. Use the numeric keypad to program the new value.
4. Press “m” for mV, or “x1” for volts to select the suffix letter.
5. Press Enter to lock in the value.

Alternatively, you can modify the amplitude value with the dial and arrow keys, but then the process must be terminated by pressing Enter.

Offset is programmed the same way as amplitude, except select Offset from the soft key menus to access the offset parameter.

Note

If you use the dial or arrow keys to modify the amplitude or offset parameters, the output is updated immediately, as soon as you modify the parameter. The final value will be locked in as soon as you press Enter.

If you choose to leave the old value, press Cancel to terminate the process and to discard any change made to this parameter.

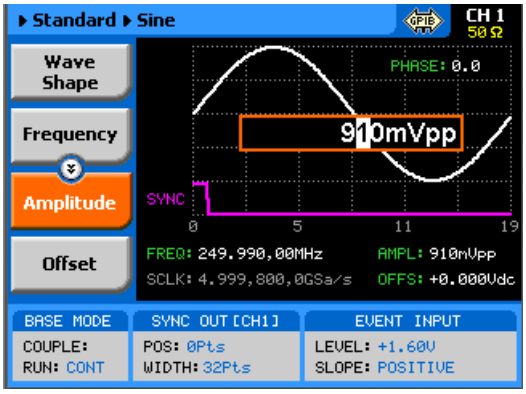

Figure 3-6, Programming Channel 1 Amplitude Example

Note

As explained above, the amplitude and offset are parameters that are associated with the DC coupled output modules. When using the DR (DAC) output module, the offset parameters has no effect..

Selecting a Run Mode

The SE5082 offers three run modes: Continuous, Triggered and Gated.

The selected waveform is repeated continuously, when the instrument is set to operate in Continuous mode. The continuous output can be armed to turn on and off using an Event input and a remote interface, consequently controlling the timing of the waveform from external sources. The operating mode defaults to continuous and self-armed mode after reset. The continuous signal can be aborted at any time, using a remote command.

Triggered and Gated modes require an external signal to initiate output cycles. In some cases, an internal trigger generator is available to generate the required trigger stimuli, without the need to connect to external devices. Figure 3-7 shows the run mode options screen. Press the Run Mode key in the Control group and then select the required run mode option using one of the soft keys on the left of the screen.

Descriptions of the various run modes and the parameters associated with each run mode are given in the following paragraphs.

Note

A selected run mode option applies to all of the available function shapes. For Separate SCLK feed mode, each channel can be programmed to a different run mode option. For Common SCLK feed mode, where both channels are synchronized, selecting a specific run mode option for one channel automatically updates the mode to match the other channel.

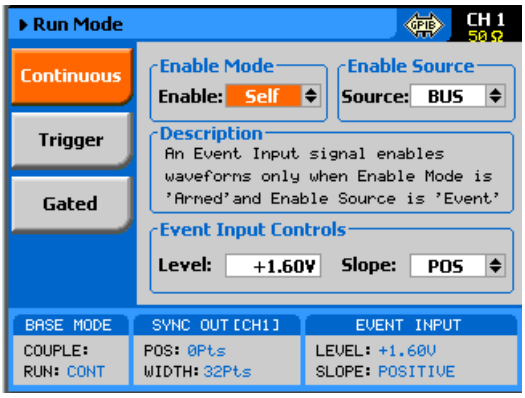

Figure 3-7, The Typical Run Mode Display (Continuous Mode Shown)

Continuous Run Mode

The continuous run mode is useful for applications that require continuous generation of waveforms. The continuous run mode is the factory default setting, so every time the instrument defaults to this mode.

Using the continuous run mode, the SE5082 generates waveforms at its output terminals without control when the waveform starts or stops. This is a normal requirement for most applications, however, some test routines require precise control over waveform timing and in this case the instrument has two enable modes: Self-Armed and Armed.

The self-armed is the standard continuous mode that is widely used and is customary on most waveform generators. This mode does not require special handling, as the output will commence with waveform generation as soon as the output is turned on. The Armed mode needs an event signal to begin waveform generation.

Description of the various controls in the Continuous Run Mode display is given below. Refer to Figure 3-7 throughout this description. To access the required parameters, use the dial or the up/down buttons. When the selected parameter is highlighted, press the enter key and use the up/down buttons to select the required option. Press Enter to complete the action and lock the selected mode.

Enable mode – defines if the generator will be self-armed or armed to commence with waveform generation. Control signals are disabled during self-armed operation, so waveforms are generated continuously at the output terminals as soon as the output function is selected and the output is turned on.

When selecting the Armed mode, the generator goes into the idle state, where the output resides on either a DC level, waveform segment, or certain sequence, depending on the selected output function. The generator will transition from idle to waveform generation immediately upon receipt of an enable signal, and will cease waveform generation immediately after receipt of an abort signal. Table 1-1 in Chapter 1 lists the idle waveform for the various output functions as well as the source options of the enable and abort signals.

Enable Source – this parameter affects the generator only when placed in Armed mode. It defines from where the SE5082 is expected to receive the enable signal. It has three options: BUS – a remote command from the host computer, EVENT – a valid signal at the rear-panel event input, and TRIG – a valid signal at the front-panel trigger input. Note that the parameters for the event input are programmed from the continuous menu and therefore, it is highly recommended that the continuous armed mode be used in conjunction with the event input. The trigger input parameters are programmed from the trigger run mode display and so it is less convenient to use this input with the continuous armed run mode.

Event Input Control – this group is used for programming the threshold level and the edge sensitivity for the rear-panel event input. The threshold level is programmable from -5 V to 5 V and the slope can be selected from positive, negative or either transitions.

Triggered Run Mode

In Triggered mode, the output remains at a DC level and waits for a valid trigger event to initiate an output cycle. Each time a trigger occurs, the SE5082 generates one complete output waveform. At the end of the output cycle, the output resumes position at a DC level that is equal to the amplitude of the first point of the waveform.

The instrument may be triggered from one of the following sources: front panel TRIG IN, front panel Manual button, rear panel event input and a remote command such as *TRG. When placed in EXT (external) trigger source, remote commands are ignored and the instrument monitors the TRIG IN connector or the MAN TRIG control. When in BUS, the hardware inputs are ignored and only

remote commands can trigger the instrument. Once initiated, the waveform can be left alone to complete its output cycle or aborted using a remote command. Table 1-2 in Chapter 1 lists the idle waveform for the various output functions as well as the source options of the initiate and abort signals.

Description of the various controls in the Triggered Run Mode display is given below. Refer to Figure 3-8 throughout this description. To access the required parameters, use the dial or the up/down buttons. When the selected parameter is highlighted, press the enter key and use the up/down buttons to select the required option. Press Enter to complete the action and to lock the selected mode.

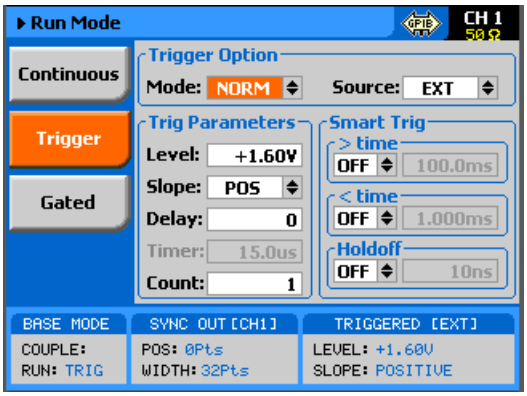

Figure 3-8, Trigger Run Mode Parameters

Trigger Options – has two fields: Mode and Source. These two fields are explained below.

Mode – the mode field selects between NORM and OVER. NORM defines the normal trigger sequence where every trigger generates a waveform cycle however, triggers are ignored for the duration of the output waveform. OVER defines a trigger override mode where triggers are accepted, regardless if the waveform has been completed or not, and a new cycle initiated.

Source – defines for the instrument from where it should expect to receive the trigger signal. Only one source for a trigger signal is legal and therefore, all other inputs are disabled automatically. The source options are: EXT, BUS, INT and EVENT. EXT enables the front panel trigger input, BUS enables remote commands, INT turns on one of the internal timed auto-triggers and EVENT enables the rear panel event input. Additional description of the internal timed auto-trigger generators is given below.

Trigger Parameters – has five fields: Level, Slope, Delay, Timer and Count. These fields are explained below.

Level – use this field to program the threshold level for the trigger input. This adjustment is necessary if your trigger signal is riding on a DC offset, positive or negative. The level is programmable from -5 V to 5 V.

Slope – defines edge sensitivity for the trigger input. The trigger input can be programmed to trigger on the rising, trailing or even both edges.

Delay – "0" setting defines no delay, any other number turns on the delayed trigger function. The delay field is used for the delayed trigger function, when the instrument is using external triggers but it is also used for the delayed auto-trigger function when the instrument is programmed to its internal trigger timer. The delayed trigger function is explained in detail in the following section.

Timer – programs the period of the internal timed auto-trigger generator. Additional information is given below.

Input – selects between 50Ω and 10kΩ input impedance for the Trigger In connector. This parameter is only available when selecting EXT in the source field.

Count – defines how many cycles will execute from a single trigger event. The counter can be programmed to generate a burst of 1 to 1,000,000 output cycles

Smart Trig – has three fields: >time, <time and Holdoff. The smart trigger filter can zoom in on a certain pulse widths featuring only values below, above or in-between to be regarded as valid trigger signals. The smart trigger fields are explained below.

>time – use this field to program the "more than" pulse width time that will pass through the filter as a valid trigger event. OFF disables this filter.

<time – use this field to program the "less than" pulse width time that will pass through the filter as a valid trigger event. OFF disables this filter.

Note that if both filters are ON, the smart trigger will consider pulse width values "less than" and "more than" as a gate entry for legal trigger events.

Holdoff – use this field to program a holdoff period where first trigger disables the input for the programmed holdoff value and only then allows entry to a legal trigger event.

Using the Delayed Trigger

The delayed trigger function, when enabled, delays the start of the output waveform for a pre-determined period following a valid trigger. The delay time value is programmed in units of sample clock periods from 0 to 8,000,000 samples. When the delay value is set to “0”, the delayed trigger function is turned off.

Note that even with the delay set to OFF, the output is always delayed from the trigger signal by a value that is specified as “system delay”. The system delay is normally small, but should always be considered as part of the overall delay period. Refer to Appendix A for the system delay specification.

Using the Built-in Auto-Trigger Generators

The correct way of using the SE5082 in applications that require synchronization between system components is in conjunction with external trigger events. In this case, another component in the system generates a trigger signal, either hardware or software that stimulates the output of the generator exactly at the time that the signal is required in the loop. In addition, the SE5082 can be used as a system administrator that generates stimulating signals to the rest of the system. For this purpose the instrument has a built-in timer that generates internal triggers at pre-determined intervals. The built-in trigger generator has no reference from the main sample clock and therefore, the clock is asynchronous with the output waveform. In fact, there are two built-in auto-trigger generators within the SE5082 that behave differently, but both are derived from the same trigger generator circuit. These are: *Timed* internal trigger generator and *Delayed* internal trigger generator.

Timed Internal Trigger Generator – generates equally spaced periodical and asynchronous triggers. The Timer field in the Trigger Run Mode displays programs on the timer interval in units of seconds ranging from 200 ns to 2 seconds. When programmed to this mode, all other inputs to the trigger circuit are disabled to prevent interference from external devices. The timer interval is measured from start of trigger, to start of the next trigger and therefore, the period of the internal timer value must be larger than the period of the output signal.

Delayed Internal Trigger Generator – this function is similar to the timed internal trigger generator in the way that it generates periodical and asynchronous triggers that are equally spaced, however, the delay is programmed in units of sample clock periods and measured between waveforms. Using the delayed internal trigger generator, there is no need to worry about waveform period vs. delay period because the delay is always measured from the end of one waveform to the start of the next one. There is also another advantage when using this function, because although the trigger generator is asynchronous with the output waveform, the delay is synchronous and therefore there is no jitter of ±1 sample clock period due to the digital error common to all other systems. The delayed internal trigger generator can be programmed from

152 sample clock periods to 8,000,000 sample clock periods, provided that the numbers are integers and divisible by 4.

Gated Run Mode

In gated mode, the output remains at a DC level and waits for a valid gate event to initiate an output cycle. Each time a gate opens the SE5082 generates output waveforms. When the gate closes, the output completes the waveform and resumes position at a DC level that is equal to the amplitude of the first point of the waveform.

The instrument may be gated from one of the following sources: front panel TRIG IN or rear panel event input. Once initiated, the waveform can be left alone to complete its output cycle or aborted using a remote command. Table 1-3 in Chapter 1 lists the idle waveforms for the various output functions, as well as the source options of the initiate and abort signals.

Description of the various controls in the Gated Run Mode display is given below. Refer to Figure 3-9 throughout this description. To access the required parameters, use the dial or the up/down buttons. When the selected parameter is highlighted, press the enter key and use the up/down buttons to select the required option. Press Enter to complete the action and lock the selected mode.

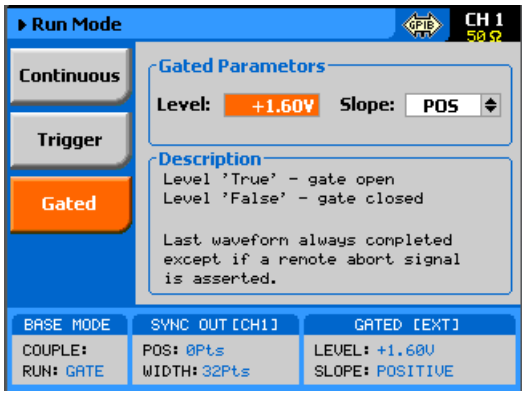

Figure 3-9, Gated Run Mode Parameters

Gated Parameters – has two fields: Level and Slope. These fields are explained as follows:

Level – use this field to program the threshold level for the trigger input. This adjustment is necessary if your gating signal is riding on a DC offset, positive or negative. The level is programmable from -5 V to 5 V.

Slope – defines polarity of the gate. The trigger input can be programmed to gate on positive levels or on negative levels. Note that the positive and negative slope definition does not apply to absolute levels, but to relative levels. For example, if the trigger level is set at 1 V and the slope is set to positive,

any level at the trigger input that is above 1 V is considered true (gate open) and anything below 1 V is considered false (gate closed).

Abort

The abort button appears only in continuous run mode and when the Enable Mode is set to Armed and the Enable Source is either BUS or EVENT or TRIG. The function of the abort soft key is to stop the waveform at a specific time. The waveform can be aborted using a remote command as well, but then the enable source must be set to BUS.

The Abort button and the Continuous Run Mode display that makes this button operational is shown in Figure 3-10. Place the instrument in Continuous run mode, select Armed as the Enable Source and select EVENT as the source for the enable signal. The generator will commence with waveform generation immediately after a valid event signal is applied to the Event input. The waveform will stop at the output immediately after you press the Abort soft key.

Figure 3-10, Typical Continuous Run Mode Display with the Abort Soft Key

Using the Manual Trigger

The manual trigger allows you to trigger or gate the SE5082 directly from the front panel. This button is active only when the generator is programmed to accept trigger stimuli from an external trigger source. In triggered run mode, a single push on the button initiates an output waveform. In gate run mode, the output generates waveforms as long as you press and hold the MAN button. The waveforms will cease to generate immediately after you release the button.

Using the SYNC Output

For safety reasons, every time you toggle the power to the SE5082 OFF and ON, the SYNC output defaults to OFF. If you want to use the SYNC output, you must turn it on immediately after you power up the generator. You can turn the SYNC on using the ON/OFF SYNC hot key as was explained earlier in this chapter or you can do it from the Outputs menu shown in Figure 3-11. To access the SYNC output menus press the Output button in the Control group on the right of the instrument and then select the SYNC OUTPUT soft key.

The SYNC output has parameters that you can modify: State, Source, Type, Position and Width. The functions of these parameters are explained below.

State – controls the state of the SYNC output ON and OFF. When the SYNC output is turned on, the LED next to the output terminal is lit, indicating that a SYNC signal is present at this output. OFF turns off the light, but leaves a low impedance path from the output terminal.

Source – selects the channel from where the SYNC output is derived, CH1 or CH2. Single channel versions do not have this control.

Type – defines the shape of the SYNC waveform. There are two options to select from this field: Pulse or WCOM. The pulse has a minimum width of 32 waveform points and its position along the waveform can be adjusted, as well as its width. The WCOM type SYNC output (waveform Complete) has a fixed width that cannot be modified, nor moved from its original position.

Position – applied to the pulse type only. This parameter defines the position of the SYNC pulse along the output waveform. The position is programmed in units of waveform points (or sample clock periods). Placement resolution is 32 points. As default, the sync signal is positioned at the beginning of the waveform.

Width – applies to the pulse type only. This parameter can modify the width of the SYNC pulse from a minimum of 32 waveform points to the maximum length of the waveform in increments of 32 points.

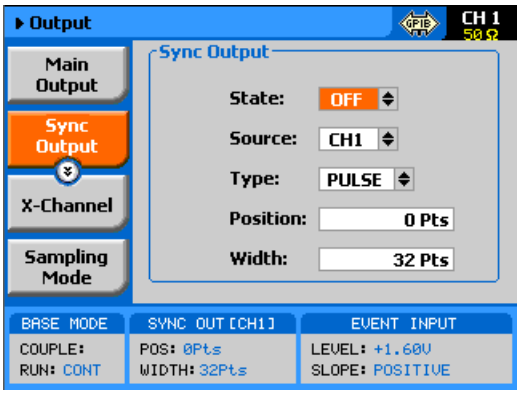

Figure 3-11, SYNC Parameters

Synchronizing Channel 1 and Channel 2

The model SE5082 comes with two channels, each of which can be operated as a stand-alone waveform generator. There are very few circuits and controls that are shared by the two channels, namely: display, remote interfaces, external sample clock and trigger input connectors and the SYNC output. In addition, each channel can be programmed to operate uniquely in terms of functions, modes and parameters. For example, channel 1 can generate continuous sequences of arbitrary waveforms and channel 2 can generate triggered pulse patterns. There are no limitations how one channel is programmed, compared to the other channel, as long as the connectors that are shared by the channels do not interfere in the way they are used.

The two channels can be synchronized to operate from a single sample clock source and in this case, the output frequency is identical and the generator provides a tight relationship between phase offset and waveform start. Still, while synchronized, the two channels can generate different waveform functions, amplitudes and DC offsets, but the same run mode option should be selected.

Synchronizing the two channels requires a single operation – modifying the SCLK Feed from separate to common. This is done from the Utility ->Clock ->Sample Clock Feed field as shown in Figure 3-12. When the selected parameter is highlighted, press the enter key and use the up/down buttons to select the required option. Press Enter to complete the action and to lock the mode.

The cross-channel parameters that control phase offset and skew are discussed in the following paragraphs. There are other parameters in this display that relate to the EXT SCLK Source and Divider and the external reference clock; these are discussed later in the relevant sections of this chapter. Notice, however, that when the Common SCLK Feed is selected, access to CH2 parameters are denied because there is only one sample clock source left to control in this mode.

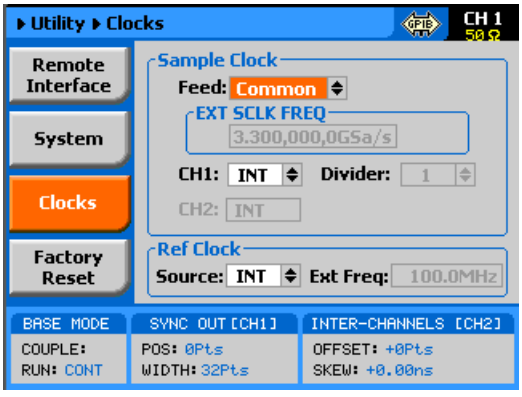

Figure 3-12, Modifying the SCLK Feed to Common

Controlling Cross-Channel Propagation

Cross-channel propagation is relevant only after the two channels have been synchronized and are fed from the same sample clock source. To access the Cross-Channel Propagation menu, press the Output key in the Control group and then press the X-CHANNEL soft key button. The display, as shown in Figure 3-13, now provides access to the synchronization parameters. If you did not synchronize the two channels, this display will be grayed out and you will have no access to the fields. Use the up/down buttons to highlight the required parameter and then press Enter. Modify the parameter and press Enter again to complete the action and to lock the selected value. There are two parameters that you can modify from this display: Offset CH2 ->CH1and Skew CH2 ->CH1. These parameters are explained below.

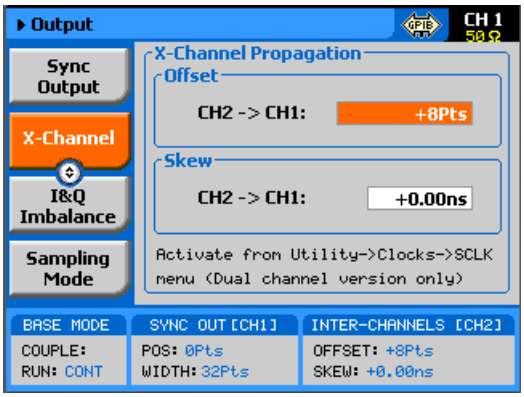

Figure 3-13, Controlling Cross-Channel Propagation

Offset CH2 -> CH1 – with channel 1 as reference, this allows programming of an offset value in units of waveform points (or sample clock periods). The CH2 -> offset can be programmed from 0 to the ±full length of the output waveform.

Skew CH2 -> CH1 – with channel 1 as reference, it defines the skew between the channels in units of ns. The skew is programmable from 0 ns to±3 ns with increments of 10 ps.

Using External Clock References

In cases where synchronization to other instruments in a system is required, you have two options: use an external clock source for the reference clock or replace the internal sample clock generator entirely with an external clock generator. Either way, this is a major twist in the SE5082 basic operation because, if for any reason, you leave one or both parameters selected from an external source and do not apply the necessary signal to the input, the operation of the generator will be impeded without visual references that something is wrong.

The SCLK and the reference clock source menus were placed in the Utility menu, as shown in Figure 3-14. Change these settings only if you are absolutely sure that another reference source is available at the appropriate inputs. Note that the sample display shows that all clocks are internal and therefore, some of the fields that are associated with the external references are grayed out.

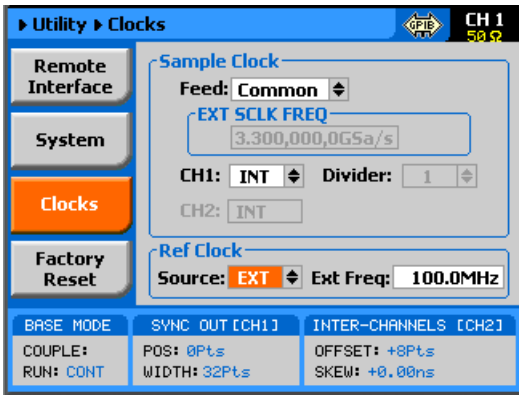

Figure 3-14, Modifying the SCLK and 10MHz Clock Source

Using an External SCLK Source

The SCLK input is useful for applications requiring improved phase noise or clock stability. Normally, by using a near-perfect source, the phase noise of the SE5082 can be improved. The external sample clock input is an SMA connector that is located on the rear panel. It accepts signals within the entire instrument's frequency range. There is only one input that can be used for one channel or another and by both channels simultaneously. Note that the channels do not have to be synchronized to use a single external clock feed. In addition, the EXT SCLK option is available only in Arbitrary or Sequence function modes.

As was described before and shown in Figure 3-14, access the External Sample Clock parameters from the Utility menu. When the selected parameter is highlighted, press the enter key and use the up/down buttons to select the required option. Press Enter to complete the action and to lock the selected mode.

As shown in Figure 3-15, there are three parameters that can be accessed in the Sample Clock Source group: EXT SCLK FREQ and CH1 Source and Divider. These are explained below.

EXT SCLK FREQ – this field becomes active only after you select an external source feed to one of the channels. This field is necessary for the instrument to establish a frequency range for the internal phase lock loops and therefore must programmed near or exactly as the external source frequency. Figure 3-15 shows an example of a setup that provides external feed to channel 1.

CH1 Source – use this field to select between INT (internal) or EXT (external) sample clock feeds. Updating the EXT SCLK FREQ field with the external frequency value is necessary for the external sample clock feed function to operate correctly.

CH1 Divider – since there is only one input for the external sample clock, feeding the same frequency to both channels is somewhat limiting if different frequencies need to be generated from each channel. The divider field accepts values in the form of 2n, from 1 to 256, that will divide the external sample clock frequency before it is applied to the internal circuits.

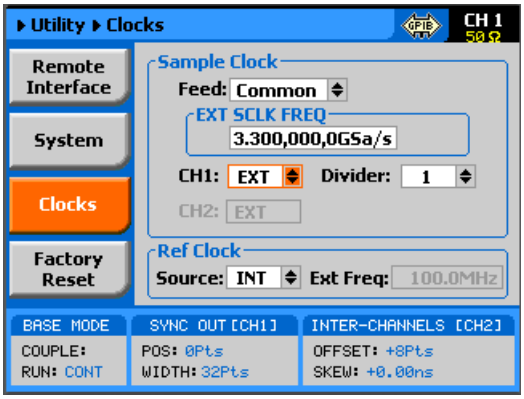

Figure 3-15, Using an External Sample Clock Example

Using an External Clock Reference Source

The External Reference input is useful for applications requiring synchronization to other components in a system, and can be used to improve clock stability and accuracy, but it does not necessarily improve phase noise. The external reference input is a BNC connector that is located on the rear panel. It accepts signals of frequency 10, 20, 50 and 100 MHz. There is only one input that feeds both sample clock generators simultaneously. Note that the channels do not have to be synchronized to use the external reference feed.

As was described before and shown in Figure 3-14, access the External Reference Clock parameters from the Utility menu. When the selected parameter is highlighted, press the enter key and use the up/down buttons to select the required option. Press Enter to complete the action and to lock the selected mode.

As shown in Figure 3-16, there are two parameters that can be accessed in the Reference Clock group: Source and EXT FREQ. These are explained below.

Source – use this field to select between INT (internal) or EXT (external) reference clock feeds. Note that updating the EXT FREQ field with the external frequency value is necessary for the external reference clock feed function to operate correctly.

EXT FREQ – this field becomes active only after you select an external reference feed. This field is necessary for the instrument to establish a frequency range for the internal phase lock loops and therefore must programmed near or exactly as the external reference source frequency. Figure 3-16 shows an example of a setup that provides external feed frequency of 20 MHz.

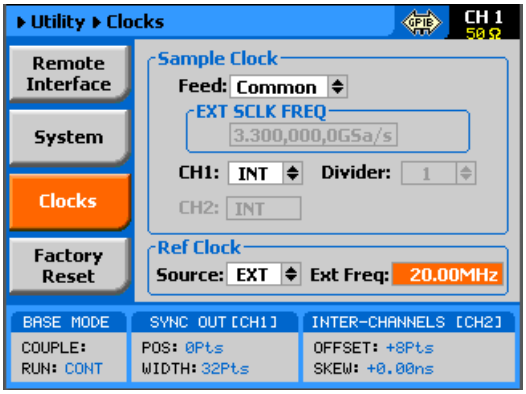

Figure 3-16, Using an External Clock Reference Example

Selecting Sampling Modes

The SE5082 offers four selectable sampling modes, NRZ, RTZ, NRTZ and RF. The difference between the sampling modes is explained in chapter 1 of this manual. Depending on the selected sampling mode the DAC frequency response can be optimized to enable increased power delivery across multiple Nyquist Zones. Combined with the proper band pass filter and amplifier the SE5082 can directly generate high frequency RF signals up to 7GHz.

To access the sampling mode menu press the Output key in the Control group and then press the Sampling Mode soft key button. The display as shown in figure 3-17, provides access to the DAC sampling mode. Each channel can be set to a different DAC sampling mode. To change sampling mode simply press the enter button and use the arrow keys or dial to select the required sampling mode.

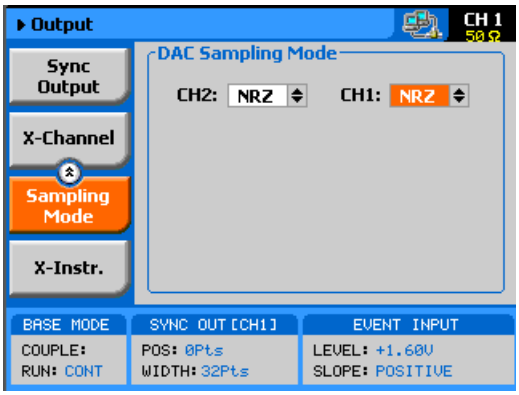

Figure 3-17, Selecting sampling mode

Generating Standard Waveforms

The majority of applications require the use of common waveforms such as: sinusoidal, triangular and square. In fact, these are the only waveforms that function generators can produce and therefore, one should expect that these waveforms be available even in a complex generator such as this. The SE5082, being a completely digital instrument, has a library of built-in waveforms that allow generation of these basic waveforms, plus many more.

By default, the SE5082 is programmed to generate standard waveforms so when you reset the generator, it will commence with generating a standard sine waveform as soon as the output is turned on. Besides the standard sine waveform, the instrument has a built-in library of waveforms referred to as Standard Waveforms. The meaning of this term is that these waveforms have standard characteristics, commonly associated with these waveforms. For example, sine waveform has known spectral and power distribution that could be compared to published mathematical equations. The quality of the generator determines the proximity of the waveform generation to its pure mathematical expression.

Figure 3-17 shows a list of waveforms that the instrument can generate, however, don’t forget that the waveforms are generated digitally on “real-time” from standard equations, utilizing maximum possible SCLK to achieve maximum frequency resolution and therefore, each time a new waveform is selected, there’s a slight delay between the time the waveform was selected to when it is being generated at the output connector.

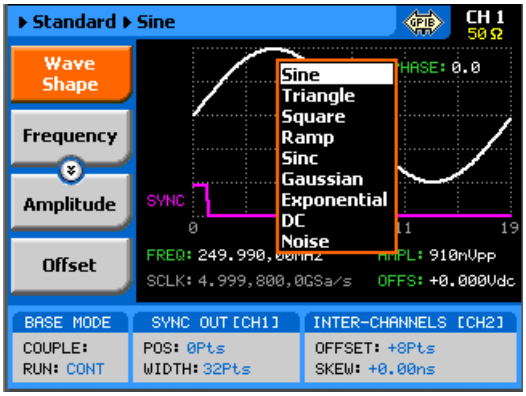

Figure 3-18, Built-in Standard Waveforms Menu

The SE5082 has a library of 9 standard waveforms: Sine, Triangle, Square, Ramp, Sinc, Gaussian, Exponential, DC and Noise. Some of the parameters for these waveforms can be modified to fine-tune the waveforms for specific applications. For example, changing the sine start phase of the 2nd channel can create a 2-phase sine system.

Sine Wave

The sine waveform is the most commonly used waveform. There are certain menus that provide access to sine waveform parameters. These are:

Frequency – programs the frequency of the sine waveform. Note that the waveform has to be re-computed every time and therefore, when you modify the frequency, the output wanders until the waveform is re-computed and restored to full accuracy.

Amplitude – programs the amplitude of the output waveform. Note that amplitude and offsets can be programmed freely within the specified amplitude window, as explained in the Programming Amplitude and Offset section of this chapter. Note that setting the amplitude parameter in this menu overrides amplitude settings in all other menus.

Offset – programs the offset of the output waveform. Note that offset and amplitude can be programmed freely within the specified amplitude window, as explained in the Programming Amplitude and Offset section in this chapter. Note that setting the offset parameter in this menu overrides offset settings in all other menus.

Phase – sets the start phase of the output waveform. You will not be able to see any change in the waveform if you generate a continuous sine waveform. However, if you place the generator in triggered run mode, the output will start the sine wave generation from a point defined by the Phase parameter. The start phase is programmed in units of degree.

Triangle Wave

The triangle waveform is a commonly used waveform. There are specific menus providing access to triangle waveform parameters. These are:

Frequency – programs the frequency of the triangle waveform. Note that the waveform has to be recomputed every time and therefore, when you modify the frequency, the output wanders until the waveform is recomputed and then restored to full accuracy.

Amplitude – programs the amplitude of the output waveform. Note that amplitude and offsets can be programmed freely within the specified amplitude window, as explained in the Programming Amplitude and Offset section in this chapter. Note that setting the amplitude parameter in this menu overrides amplitude setting in all other menus.

Offset – programs the offset of the output waveform. Note that offset and amplitude can be programmed freely within the specified amplitude window, as explained in the Programming Amplitude and Offset section in this chapter. Note that setting the offset parameter in this menu overrides offset setting in all other menus.

Phase – sets the start phase of the output waveform. You will not be able to see any change in the waveform if you generate a continuous triangular waveform. However, if you place the generator in triggered run mode, the output will start the triangle wave generation from a point defined by the Phase parameter. The start phase is programmed in units of degree.

Square Wave

The square waveform is a commonly used waveform. There are specific menus that provide access to square waveform parameters. These are:

Frequency – programs the frequency of the square waveform. Note that the waveform has to be recomputed every time and therefore, when you modify the frequency, the output wanders until the waveform is recomputed and then restored to full accuracy.

Amplitude – programs the amplitude of the output waveform. Note that amplitude and offsets can be programmed freely within the specified amplitude window, as explained in the Programming Amplitude and Offset section in this chapter. Note that setting the amplitude parameter in this menu overrides amplitude settings in all other menus.

Offset – programs the offset of the output waveform. Note that offset and amplitude can be programmed freely within the specified amplitude window, as explained in the Programming Amplitude and Offset section in this chapter. Note that setting the offset parameter in this menu overrides offset settings in all other menus.

Duty Cycle – programs the square wave duty cycle (pulse width to period ratio). The duty cycle is programmed as a percent of the period. The default value is 50%.

Ramp Wave

The ramp waveform is a special instance of the triangular waveform with a slight difference: the ramp can be adjusted for its rise and fall times. The ramp waveform is a very common waveform and is required for numerous applications. There are specific menus that provide access to ramp waveform parameters. These are:

Frequency – programs the frequency of the ramp waveform. Note that the waveform has to be recomputed every time and therefore, when you modify the frequency, the output wanders until the waveform is recomputed and then restored to full accuracy.

Amplitude – programs the amplitude of the output waveform. Note that amplitude and offsets can be programmed freely within the specified amplitude window, as explained in the Programming Amplitude and Offset section in this chapter. Note that setting the amplitude parameter in this menu overrides amplitude settings in all other menus.

Offset – programs the offset of the output waveform. Note that offset and amplitude can be programmed freely within the specified amplitude window, as explained in the Programming Amplitude and Offset section in this chapter. Note that setting the offset parameter in this menu overrides offset settings in all other menus.

Delay – sets the delay time for the ramp start. The delay is programmed as a percent of the ramp period.

Rise – programs the ramp rise time. The rise time is programmed as percent of the ramp period.

Fall – programs the ramp fall time. The fall time is programmed as a percent of the ramp period.

Note that the sum of the delay, rise and fall times cannot exceed 100%. If the sum is less than 100%, the end of the ramp will remain at a DC level until the completion of the period.

Sinc Wave

The sinc pulse (sine x/x) waveform is a very common waveform and is required in many applications. There are specific menus that provide access to sinc pulse waveform parameters. These are:

Frequency – programs the frequency of the sinc waveform. Note that the waveform has to be re-computed every time and therefore, when you modify the frequency, the output wanders until the waveform is recomputed and then restored to full accuracy.

Amplitude – programs the amplitude of the output waveform. Note that amplitude and offsets can be programmed freely within the specified amplitude window, as explained in the Programming

Amplitude and Offset section in this chapter. Note that setting the amplitude parameter in this menu overrides amplitude settings in all other menus.

Offset – programs the offset of the output waveform. Note that offset and amplitude can be programmed freely within the specified amplitude window, as explained in the Programming Amplitude and Offset section in this chapter. Note that setting the offset parameter in this menu overrides offset settings in all other menus.

#Cycles – sets the number of "0" crossing cycles for the sinc function. Note that the default value is 4. Changing the value to a different number requires recalculation of the waveform and may take a few seconds until the waveform is computed and generated at the output connector.

Gaussian Wave

The gaussian pulse waveform is useful in many applications. There are specific menus that provide access to gaussian pulse waveform parameters. These are:

Frequency – programs the frequency of the gaussian waveform. Note that the waveform has to be recomputed every time and therefore, when you modify the frequency, the output wanders until the waveform is recomputed and then restored to full accuracy.

Amplitude – programs the amplitude of the output waveform. Note that amplitude and offsets can be programmed freely within the specified amplitude window, as explained in the Programming Amplitude and Offset section in this chapter. Note that setting the amplitude parameter in this menu overrides amplitude settings in all other menus.

Offset – programs the offset of the output waveform. Note that offset and amplitude can be programmed freely within the specified amplitude window, as explained in the Programming Amplitude and Offset section in this chapter. Note that setting the offset parameter in this menu overrides offset settings in all other menus.

Exponent – sets the exponent factor for the gaussian function. Changing the default exponent value to a different number requires recalculation of the waveform and may take a few seconds until the waveform is computed and generated at the output connector.

Exponential Wave

The exponential pulse waveform is useful in applications simulating capacitor charge or discharge. There are specific menus that provide access to exponential pulse waveform parameters. These are:

Frequency – programs the frequency of the exponential waveform. Note that the waveform has to be recomputed every time and

therefore, when you modify the frequency, the output wanders until the waveform is recomputed and then restored to full accuracy.

Amplitude – programs the amplitude of the output waveform. Note that amplitude and offsets can be programmed freely within the specified amplitude window, as explained in the Programming Amplitude and Offset section in this chapter. Note that setting the amplitude parameter in this menu overrides amplitude settings in all other menus.

Offset – programs the offset of the output waveform. Note that offset and amplitude can be programmed freely within the specified amplitude window, as explained in the Programming Amplitude and Offset section in this chapter. Note that setting the offset parameter in this menu overrides offset settings in all other menus.

Exponent – sets the exponent factor for the exponential function. Setting the exponent to a negative value inverts the exponential function. Changing the default exponent value to a different number requires recalculation of the waveform and it may take a few seconds until the waveform is computed and generated at the output connector.

DC Wave

The DC waveform is a useful application simply requiring an accurate DC level (Not available with AC output module).

There are specific menus that provide access to the DC waveform parameters. These are:

DC Level – programs the level of the DC output function. The amplitude is programmed in units of volts and generated continuously at the output connector in way similar to the way a power supply generates its output. Note however, that the amplitude is calibrated when the output is terminated into 50Ω load impedance.

Noise Wave

The noise waveform is useful in applications requiring generation of simple noise. The spectral spread of the noise is pseudo-random and therefore is bandwidth limited. There are specific menus that provide access to noise waveform parameters. These are:

Amplitude – programs the amplitude of the output waveform. Note that amplitude and offsets can be programmed freely within the specified amplitude window, as explained in the Programming Amplitude and Offset section in this chapter. Note that setting the amplitude parameter in this menu overrides amplitude settings in all other menus.

Offset – programs the offset of the output waveform. Note that offset and amplitude can be programmed freely within the specified amplitude window, as explained in the Programming Amplitude and

Offset section in this chapter. Note that setting the offset parameter in this menu overrides offset settings in all other menus.

Note that while generating noise, keep in mind that the noise is generated in a certain memory size and is repeated over and over until the function is disabled. Therefore, the noise is not really random, in the purest meaning of the word.

Standard Waveforms and Run Mode Options

Run Mode options define if the standard waveforms will be generated continuously, triggered, or gated. Refer to the Selecting Run Modes section of this manual to learn more about these functions. Table 3-5 provides information on the standard waveform function in conjunction with the run mode options. This table is useful, as it summarizes all the controls that start and stop the waveforms. The Glossary defines the terms used in this table.

Table 3-6, Standard Waveforms in Various Run Mode Options

Run Mode	Waveform	Arm Options	Idle Waveform	Enable Signal	Abort Signal	Initiate Signal	Wave Loops	Smart Trig
Continuous	Standard	Self Armed	Wave	- (*)	-	-	-	-
		Armed	DC	Trigger | BUS | Event	BUS | F.P.	-	-	-
Triggered	Standard	Self Armed	-	-	-	-	-	-
		Armed	DC	-	BUS	F.P.| BUS | Trigger | Event	1-1M	Yes
Gated	Standard	Self Armed	-	-	-	-	-	-
		Armed	DC	-	BUS	Trigger | Event	-	-

- (*) – Not Relevant

Glossary of Terms

Run Mode – defines basic instrument run mode options:

Continuous – waveform is generated continuously.

Triggered – output waits for trigger, waveform is generated once.

Gated – output waits for gating signal, waveform is generated as long as the gating signal is true

Waveform – standard, built-in library of standard waveforms.

Arm Options – defines if a waveform is self-enabled or requires a control signal to initiate.

Self Armed – the output does not require a control signal to generate waveforms.

Armed – a control signal is required to enable waveform generation.

Idle Waveform – defines the state of the output waveform before an event (enable, trigger or gate) initiates a waveform.

DC – first waveform point.

Wave – first waveform in a sequence.

Enable Signal – defines the source, from where an enable signal is expected. Affects the output only when Arm Option is set to Armed.

BUS – a remote command enables the output.

Event – a transition at the event input enables the output.

Trigger – a valid signal at the trigger input enables the output.

Abort Signal – defines the source from where an abort signal is expected.

BUS – a remote command aborts all output activities. This signal does not affect the generator when set to operate in self- armed mode.
F.P. – a front panel push button is used to abort all outputs activities

Initiate Signal – defines the wait for the trigger entry port to initiate a waveform. Relevant for trigger and gated run modes only.
F.P. – a front panel push button is used to initiate a waveform or a sequence.
BUS – a remote command initiates a waveform or a sequence. Selecting this option automatically disables front panel operation.
Trigger – a valid signal at the trigger input initiates a waveform or a sequence.
Event – a transition at the event input initiates the waveform

Wave Loops – defines how many times the waveform repeats after a valid Initiate Signal.

Smart Trigger – defines if smart trigger features are available for a specific function.

Generating Arbitrary Waveforms

In general, the SE5082 cannot create arbitrary waveforms by itself. If you want to use arbitrary waveforms, you must first load them into the instrument. The SE5082 is supplied with waveform creation and editing software, called **ArbConnection**. Figure 3-18 shows an example of a waveform that was created with an external utility. Once the waveform is created on the screen, downloading it to the SE5082 is just a click of a mouse away.

Detailed information on the structure of the arbitrary waveform and the commands that are needed to download arbitrary waveforms to the SE5082, are given in the programming chapter of this manual. Information in this Chapter will give you a general idea of what arbitrary waveforms are all about.

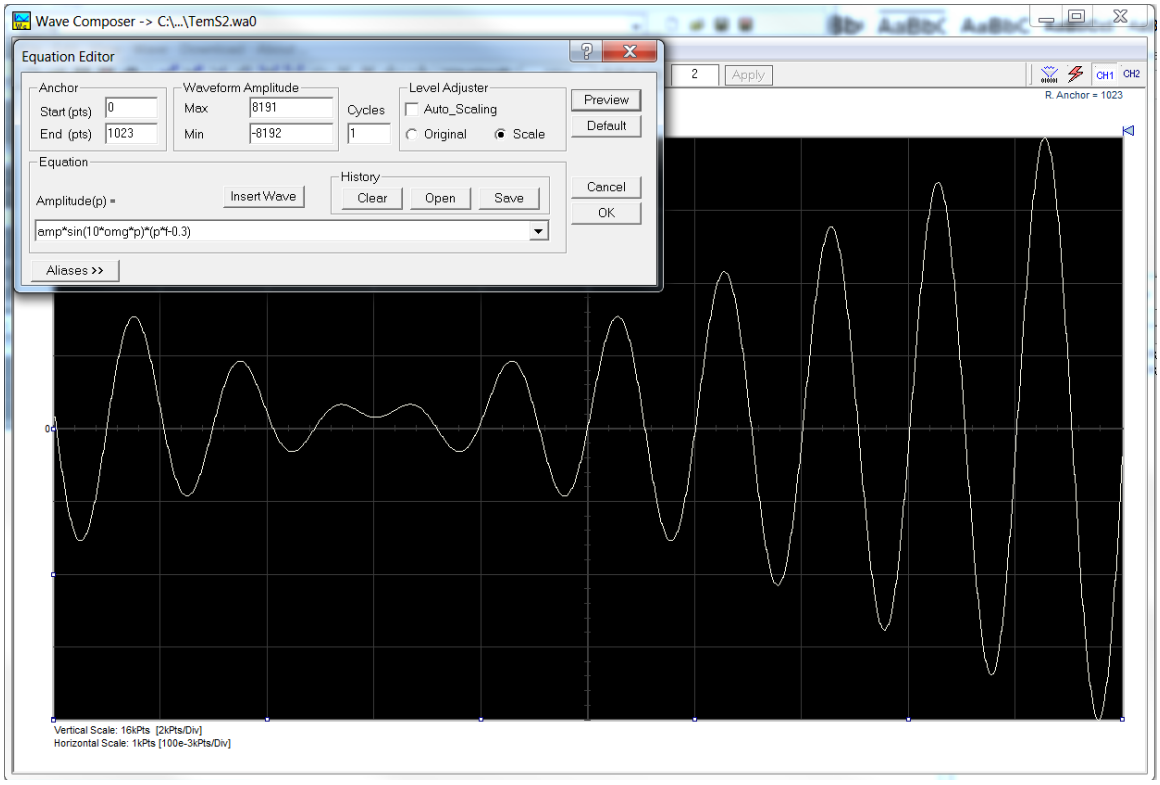

Figure 3-19, Wave Composer Tool Example for Generating Arbitrary Waveforms

What Are Arbitrary Waveforms?

Arbitrary waveforms are generated from digital data points, which are stored in a working memory. The working memory is connected to a digital-to-analog converter (DAC) and a sample clock generator clocks the data points, one at a time, to the output circuit. In slow motion, the output generates a waveform that resembles the look of a staircase. In reality, the DAC is generating amplitude hops that depend on bit arrangement and sample clock speed.

The working memory has two major properties: vertical resolution and memory depth.

Vertical Resolution – This term defines the precision along the vertical axis of which data points can be placed and generated by the DAC. The SE5082 uses 12-bit DAC's to generate arbitrary waveforms. Converting 12 bits to precision shows that each data point can be placed along the vertical axis with a precision of 1/4096.

Memory Depth – Defines how many data points can be stored for a single waveform cycle. The SE5082 has a 32,000,000 waveform points as basic waveform memory capacity and 64,000,000 waveform points, as an option. Note that minimal segment size must be at least 384 points and is of multiples of 32 points.

Having such large memory capacity is an advantage. Modern applications in the telecommunications industry require simulation of long waveforms without repeatable segments. The only way to create such waveforms is having sufficient memory depth. On the other hand, if you do not need to use very long waveforms but must have many other waveforms stored in your working memory, the SE5082 lets you divide the memory bank to smaller segments and load different waveforms into each segment.

Generating Arbitrary Waveforms

Downloading waveforms to the SE5082 and managing arbitrary memory are explained in the programming section of this manual. This section assumes that you have already downloaded waveforms and want the instrument to output these waveforms.

Refer to Figure 3-19 and use the following description to learn how to output arbitrary waveforms, as well as how to program arbitrary waveform parameters. To access the Arbitrary display, press on the Arbitrary button in the Function group. The output will immediately resume with arbitrary waveform generation and the display will be updated with arbitrary waveform parameters, as shown in Figure 3-19.

Note the channel you are currently programming and make sure the icon on the upper right corner agrees with your required programming sequence. Use the following procedure to modify the parameters that are associated with the arbitrary waveform function:

1. Press the soft key next to the required parameter to display the edit field.
2. Punch in the value using the numeric keypad. Be careful not to exceed parameter limits while you key the numbers.
3. Select and press a suffix.
4. Press Enter to lock in the new value.

Alternatively, after you display the edit field, you may use the dial and/or the arrow keys to modify the field, then press Enter to lock in

the new value. If you didn't make programming errors and didn't make any mistakes while downloading your waveform segment(s), then the output should generate your desired waveform.

There are several parameters that are available for programming in this window: Sample Clock, Amplitude, Offset, Active Segment, Jump Event, Jump Timing, Wave Composer and Delete Segments. These parameters are explained as follows:

Sample Clock – Defines the sample clock frequency for the arbitrary waveform. Information how to modify the sample clock is given later in this chapter.

Amplitude – Defines the amplitude of the arbitrary waveform. Note that regardless of the amplitude setting, the vertical resolution of the generated waveform is always 12 bits.

Offset – defines the offset value of the arbitrary waveform.

Active Segment – Defines which of the segments in the working memory is currently active at the output connector. As was discussed earlier, the working memory can be divided to 32k segments and different waveforms loaded in each segment. Any segment is available at the output connector, only if it has been selected to be the active segment. The segment selection field lets you select any segment, regardless if it contains waveform data or not, so be careful when you select a segment number, as it may be empty and no output will be generated.

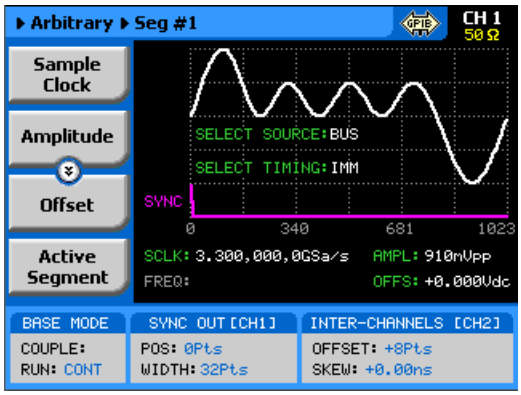

Figure 3-20, Arbitrary Parameters Display Example

Jump Event – selects between remote command, which is the default option and the rear-panel segment control connector. In case the external control is selected and signals are not hooked-up to the connector, the output automatically assumes that the first segment in the table is selected. Information how to use the external control input is given following this section.

Jump Timing – When a new waveform segment is selected, the generator is presented with a choice: switch at the end of the segment – coherent switch, or switch immediately without waiting for the segment to complete. The jump-timing menu provides access to this selection. Coherent is the default choice.

Wave Composer – opens a new display that allows creation of simple arbitrary waveforms from the front panel. The Wave Composer display is shown in Figure 3-20.

Delete Segments – Allows distractive removal of all segments from the memory. In fact, this command does not erase the memory but only removes the table that defines start and stop addresses for each segment location. If you have recorded your segment sizes you can always redefine the segment table, which will restore the original waveforms in each segment. There is, however, no way back to the original waveforms if you perform a new download action after deleting the segment table.

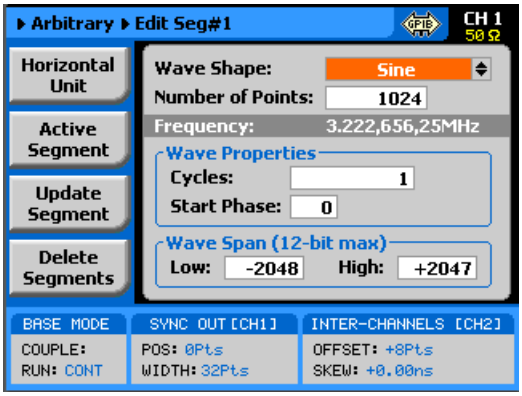

Figure 3-21, The Wave Composer Display

Using the Dynamic Sequence / Segment Control

A Sequence / Segment control input is available on the SE5082 rear panel that provides control over the replay of a specific memory segment. The control connector has 9 pins and therefore, allows replay of one of 256 waveform segments. A valid line must be asserted to validate segment change. Having the dynamic control feature, in effect, can serve as replacement of the sequence table where the real-time application can decide when and for how long a waveform will be generated.

The Sequence / Segment control input accepts TTL level signals. Bit 1 is connected to pin 1; it weighs 2^0 binary (decimal 1) and bit 8 is connected to pin 9 and weighs 2^7 binary (decimal 128) so if all lines are low, the output generates segment number 1 (the firmware counts the segments from 0 to 255) and if all lines are high, the output generates segment 256. Any combination from 0 to 255 can be produced at the inputs of this connector but the output will change only when a valid signal is asserted at pin 9 of the same connector. Figure 3-21 shows a typical example of external sequence / segment control; the first valid transition selects segment 37, the second selects segment 57 and the third transition selects segment 194.

Two sequence / segment control inputs are available on the rear

panel, one for channel 1 and one for channel 2, and each channel can be modified to have a specific output segment. Note however, that if you intend to synchronize the outputs, the waveforms that are selected with this control must have exactly the same length. Figure 3-22 shows the Sequence / Segment control connector and Table 3-8 describes its pin assignments.

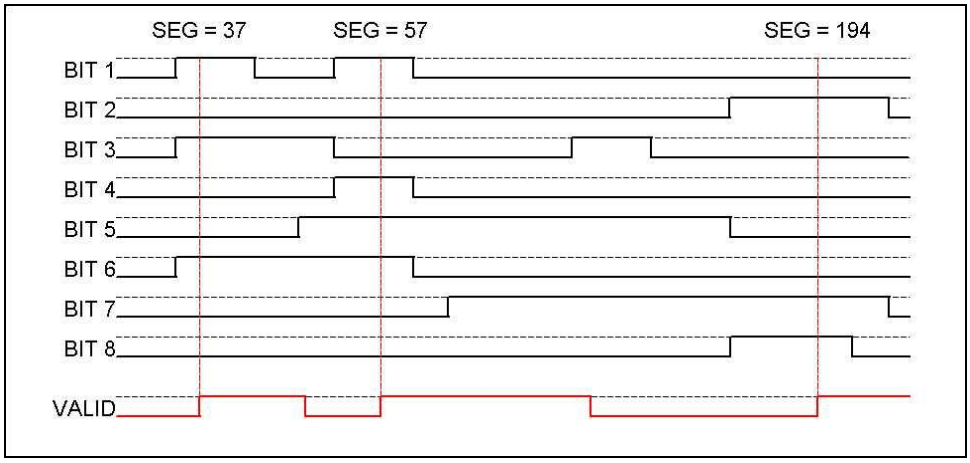

Figure 3-22, Typical Dynamic Control Scenario

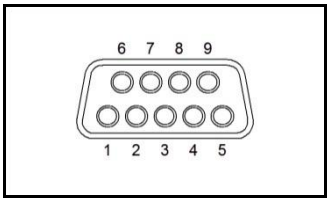

Figure 3-23, Sequence / Segment control connector

Table 3-7, Sequence / Segment control connector pin assignments

Pin Number	Assignment
1	Bit 1
2	Bit 3
3	Bit 5
4	Bit 7
5	Valid
6	Bit 2
7	Bit 4
8	Bit 6
9	Bit 8
Case	Ground

Arbitrary Waveforms and Run Mode Options

Run Mode options define if the arbitrary waveforms will be generated continuously, triggered, or gated. Refer to the Selecting Run Modes section of this manual to learn more about these functions. Table 3-9 provides information on the arbitrary waveform function in conjunction with the run mode options. This table is useful, as it summarizes all the controls that start and stop the waveforms. The Glossary defines the terms that were used in this table.

Table 3-8, Arbitrary Waveforms in Various Run Mode Options

Run Mode	Waveform	Arm Options	Idle Waveform	Enable Signal	Abort Signal	Initiate Signal	Wave Loops	Smart Trig
Continuous	Arbitrary	Self Armed	Wave	- (*)	-	-	-	-
		Armed	DC	Trigger | BUS | Event	BUS | F.P.	-	-	-
Triggered	Arbitrary	Self Armed	-	-	-	-	-	-
		Armed	DC	-	BUS	F.P.| BUS | Trigger | Event	1-1M	Yes
Gated	Arbitrary	Self Armed	-	-	-	-	-	-
		Armed	DC	-	BUS	Trigger | Event	-	-

- (*) – Not Relevant

Glossary of Terms

Run Mode – defines basic instrument run mode options:

Continuous **–** waveform is generated continuously.

Triggered – output waits for trigger, waveform is generated once.

Gated – output waits for gating signal, waveform is generated as long as the gating signal is true.

Waveform – arbitrary waveform, waveform coordinates must be downloaded to the instrument to generate these waveforms.

Arm Options – defines if a waveform is self-enabled or requires a control signal to initiate.

Self Armed **–** the output does not require a control signal to generate waveforms.

Armed – a control signal is required to enable waveform generation.

Idle Waveform – defines the state of the output waveform before an event (enable, trigger or gate) initiates a waveform.

DC – first waveform point.

Wave **–** first waveform in a sequence.

Enable Signal – defines the source from where an enable signal is expected. Affects the output only when Arm Option is set to Armed.

BUS – a remote command enables the output.

Event – a transition at the event input enables the output.

Trigger – a valid signal at the trigger input enables the output

Abort Signal – defines the source from where an abort signal is expected.

BUS – a remote command aborts all output activities. This signal does not affect the generator when set to operate in self- armed mode.

F.P. – a front panel push button is used to abort all outputs activities

Initiate Signal – defines the wait for trigger entry port to initiate a waveform. Relevant for trigger and gated run modes only.

F.P. **–** a front panel push button is used to initiate a waveform or a sequence.

BUS – a remote command initiates a waveform or a sequence. Selecting this option automatically disables front panel operation.

Trigger – a valid signal at the trigger input initiates a waveform.

Event – a transition at the event input initiates a waveform.

Wave Loops – defines how many times the waveform repeats after a valid Initiate Signal.

Smart Trigger – defines if smart trigger features are available for a specific function.

Generating Sequenced Waveforms

In general, the SE5082 cannot create sequenced waveforms by itself. If you want to use sequenced waveforms, you must first load them into the instrument. The SE5082 is supplied with a waveform creation and editing utility. Besides waveform creation, this utility has instrument control features and sequence table generator panel. To generate a sequence, you must first download waveforms to the instrument, generate a sequence table and download the sequence table to the instrument. Sequences are easily generated using the Waveform Studio as demonstrated in Figure 3-23 and displayed on the SE5082 as shown in Figure 3-24. Note that different sequences can be generated for each channel.

Detailed information on the structure of the arbitrary waveform and the commands that are needed to download arbitrary waveforms to the SE5082 is given in the Programming Reference Chapter. There, you can also find information how to create and download sequence tables using SCPI programming commands. Information in this chapter will give you a general idea what sequenced waveforms are all about.

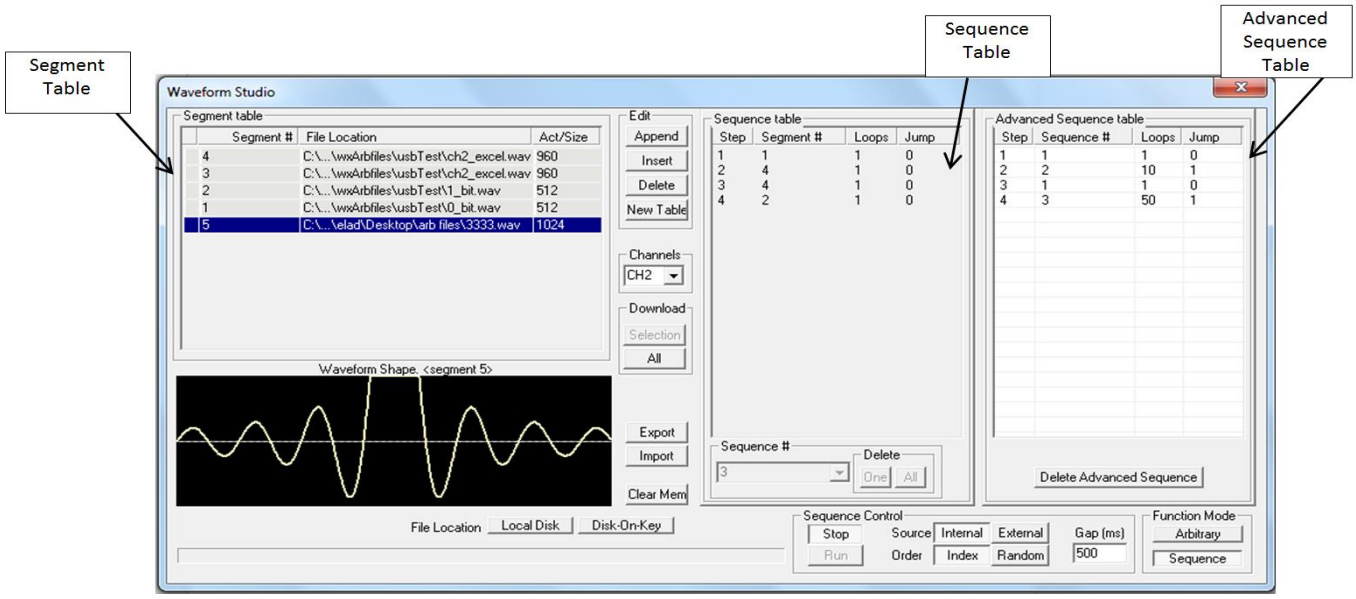
Segment Table
Sequence Table
Advanced Sequence Table
Waveform Studio
Segment table
Segment # | File Location | Act/Size
4 | C:\...\wxArbfiles\usbTest\ch2_excel.wav | 960
3 | C:\...\wxArbfiles\usbTest\ch2_excel.wav | 960
2 | C:\...\wxArbfiles\usbTest\1_bit.wav | 512
1 | C:\...\wxArbfiles\usbTest\0_bit.wav | 512
5 | C:\...\elad\Desktop\arb files\3333.wav | 1024
Edit
Append
Insert
Delete
New Table
Channels
CH2
Download
Selection
All
Waveform Shape, <segment 5>
Export
Import
Clear Mem
File Location
Local Disk
Disk-On-Key
Sequence table
Step | Segment # | Loops | Jump
1 | 1 | 1 | 0
2 | 4 | 1 | 0
3 | 4 | 1 | 0
4 | 2 | 1 | 0
Sequence #
3
Delete
One
All
Advanced Sequence table
Step | Sequence # | Loops | Jump
1 | 1 | 1 | 0
2 | 2 | 10 | 1
3 | 1 | 1 | 0
4 | 3 | 50 | 1
Delete Advanced Sequence
Sequence Control
Stop
Run
Source
Internal
External
Order
Index
Random
Gap (ms)
500
Function Mode
Arbitrary
Sequence

Figure 3-24, Using the Waveform Editor to Generate Sequences

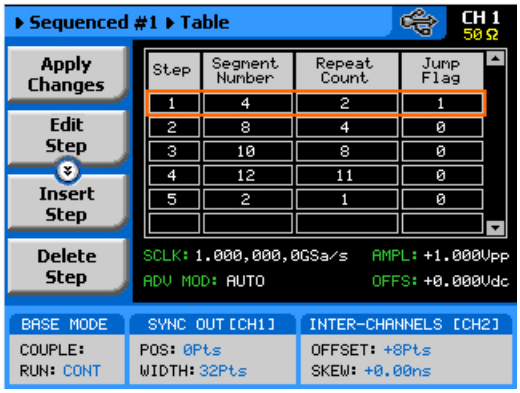

Figure 3-25, Sequence Table Display Example

What are Sequenced Waveforms?

Sequenced waveforms are constructed from two or more arbitrary waveforms, which are linked and looped in any way you can imagine, as long as you observe the limitations set forth in the specification section of this manual.

The first thing to do before you can generate sequenced waveforms, is download waveforms to the SE5082. You may use any application to create waveform segments. Then, you can build your sequence table. An example of how sequenced waveforms work with three different waveforms is demonstrated in Chapter 1, Figures 1-7 through 1-10. Refer to Figure 3-25 and use the following description to learn how to output sequenced waveforms as well as how to program sequence parameters.

Press the Sequenced button in the Functions group on the right side of the generator to select Sequenced waveforms as the output function type. The screen as shown in Figure 3-25 displays the sequence function screen. Providing that waveforms were loaded and a table created, it will immediately commence with generating arbitrary waveforms. Note the channel you are currently programming and make sure the icon on the upper right corner agrees with your required programming sequence.

Use the following procedure to modify the parameters that are associated with the Sequenced waveforms function:

1. Press the soft key next to the required parameter to display the edit field.
2. Punch in the value using the numeric keypad. Be careful not to exceed parameter limits while you key the numbers.
3. Select and press a suffix.
4. Press Enter to lock in the new value.

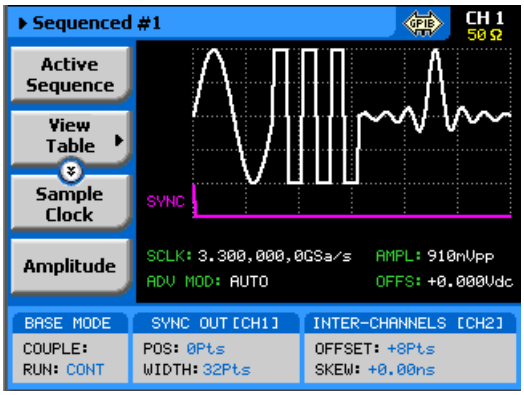

Figure 3-26, Sequence Function Parameters

Alternatively, after you display the edit field, you may use the dial and/or the arrow keys to modify the field, then press Enter to lock in the new value. If you didn't make programming errors and didn't make any mistakes while downloading your waveform segment(s), then the output should generate your desired waveform. There are six parameters that are available for programming in this window: Active Sequence, View Table, Sample Clock, Amplitude, Offset and Config. These parameters are explained as follows:

Active Sequence – selects the active sequence to be generated at the output terminals. 1000 different sequence scenarios can be defined and replayed one at a time.

View Table – provides access to a new menu that has sequence table control, as shown in Figure 3-24. If no table was yet defined, you can define the sequence table from this menu. You can also edit an existing sequence table from this command. Information on editing the sequence table is given later.

Sample Clock – programs the sample clock frequency for the sequenced waveform. The final period of the complete sequence can be extracted from the following relationship:

Sequence Duration = 1 / (SCLK / n)

n = the number of waveform points in the sequence, including looped waveforms.

Amplitude – defines the amplitude of the sequenced waveform. Might be replaced by Power, in case AC coupled has been selected for this output.

Offset – defines the offset of the sequenced waveform. If AC couple mode is selected for this output, offset will not be available.

Config – provides access to a new menu that has sequence configuration controls. The configuration menu is shown in Figure 3-26 and its controls are described below. There are 6 control parameters in this menu: Sequence Type, Advance Mode, Once Count, Select Source, Select Timing and Jump Event.

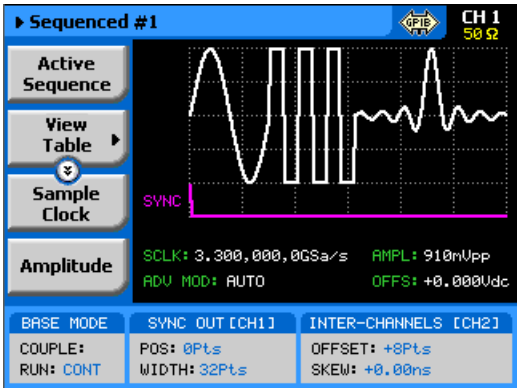

Figure 3-27, Sequence Configuration Parameters

Sequence Type – defines if the sequence is normal or advanced. The advanced mode will be discussed in detail in the following paragraphs.

Advance Mode – defines the advance mode for the sequence. There are three advance mode options you can select from: Automatic, Once and Stepped. These are described in detail later in the chapter.

Once Count – programs a count value for the Once advance mode. Additional information on this parameter will be given in the sequence advance description section. Note that this button appears only when the Once advance mode has been selected.

Select Source – defines the source from where one of the sequences is selected: remote command or rear panel external control. The rear-panel dynamic control is explained separately in one of the following paragraphs.

Select Timing – When a new sequence is selected, the generator is presented with a choice: switch at the end of the sequence – coherent switch, or switch immediately without waiting for the sequence to complete. The select timing menu provides access to this selection. Coherent is the default choice.

Jump Event – defines the source from where the generator will expect a jump signal to flag that it can advance to the next step. This parameter will show only for the Auto and Step advance modes.

Sync Lock – Programs sync output lock on a specific waveform or sequence.

Sequence Table Elements

The sequence table, as shown in Figure 3-24, specifies: Step, Segment Number, Repeat Count and Jump Bit. Description of the various elements within the sequence table is given below. These are explained in the following paragraphs.

Step - This parameter defines an index array for the sequence generator. When generating sequences, the instrument links the steps in descending order. Therefore, make sure that you enter your waveform segments in exactly the order you would like them at

the output.

Segment Number - This parameter associates waveform segments with a step index. You can use different waveforms for different steps or you can use the same waveform for a number of steps. There are no limitations how you associate steps to waveforms, except you cannot program in the sequence table waveforms that were not defined earlier.

Repeat Count – This parameter defines how many times the waveform will loop for the selected step. For example, if you program 2, the waveform will cycle twice on the same step before transitioning to the next step.

Jump Flag – This field is a special code that is used in conjunction with the automatic advance mode. When the sequence generator encounters this bit, it will stop and dwell on the step until a jump event is asserted to the relevant input. Information on the jump bit is given later. “0” flags continuous, “1” flags wait-for-an-event signal.

Assuming that you already downloaded waveforms, created and downloaded the corresponding sequence table, you can now proceed with the following instructions on how to set the SE5082 to output sequenced waveforms.

Editing the Sequence Table

If you select the View Table option as was described above, the sequence table will display as shown in Figure 3-27. If you already have a sequence table in place, you can edit the steps and modify the table per your new requirements. If you do not have a sequence table, you can construct the table from this screen. However, you first must make sure that the segments you intend to use are loaded with waveforms.

Please see Figure 3-27, noting the commands that are available for editing and creating a sequence table.

Apply Changes – After you make modifications to the sequence table, you must use this command to update the internal registers with the new table settings and output updates immediately with the new settings. Changes, if made in the table, will be updated automatically when you exit the Edit Table screen. However, the output will change to the new settings only after you re-enter the sequence function. Due to hardware considerations, please note a valid sequence table has to contain at least three rows.

Edit Step – Provides entry point to the table. You may scroll between the fields using the arrow keys. If you want to edit a specific step, place the cursor on the step and press Enter. Edit the field as required and press Enter again to lock in the new value.

Insert Step – Allows adding another step to the sequence table. You have a choice of adding the step above or below the cursor line or at the end of the sequence table.

Delete Step – Use this command to delete a specific step from the

sequence. You'll be asked to confirm if you really want to delete the step, before the final execution.

Delete Table – Use this command to delete the entire sequence table. You'll be asked to confirm if you really want to delete the step before the final execution.

Tip

Use the arrow keys or the dial to scroll through the edit parameters. The Apply Changes will remain at the top, while the others may be selectively accessed.

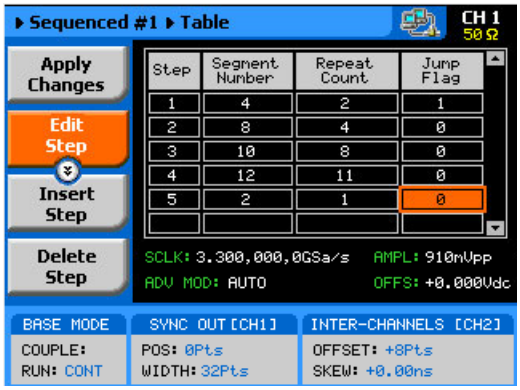

Figure 3-28, Editing the Sequence Table

Selecting Sequence Advance Mode

As was explained above, the SE5082 goes through an index of steps. It may loop a few times on a designated step and eventually, after the last step in the table, the process repeats itself. Going from step-to-step through the sequence table is done automatically by the instrument. However, there are applications requiring control of when and how the link is stepped. The SE5082 has a number of sequence advance options: Auto, Once and Stepped. Table 3-8 summarizes the various advance modes in reference to the run mode options. The various options on how to advance sequencer steps are described as follows:

Auto – specifies continuous advance where the generator steps continuously and automatically through the links to the end of the sequence table and then repeats the sequence from the beginning. For example, if a sequence is made of three segments – 1, 2, and 3, and AUTO mode is used, the sequence will generate an infinite number of 1, 2, 3, 1, 2, 3, 1, 2, 3…waveforms. Of course, each step waveform can be assigned a repeat counter that will cause the waveform to loop as many times as is specified in the Repeat count field.

The above description is true as long as the Jump Bits in all of the steps is set to "0". When a Jump Bit flag = "1" has been assigned to one or more steps, the waveforms dwells on this specific step until a valid jump signal event is asserted and only then the step is

allowed to complete the waveform and move to the next step in the sequence. AUTO is the default sequence advance mode.

Once – using this advance mode, the SE5082 initiates a single sequence scenario and resumes its idle state. The idle waveform depends on the run mode, as shown in Table 3-5. In continuous run mode, the sequence is enabled and in triggered and gated run mode the sequence is initiated. In both cases, if the Jump Bit flag is set to "1", the generator will dwell on the selected waveform until released to the next step from a valid jump signal.

The once advance mode has a repeat loop that can be used to generate a counted burst of single sequence scenarios. This counter can be programmed from 1 to 1,000,000 loops.

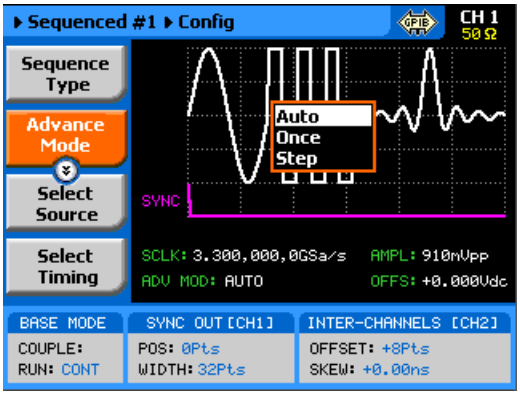

Figure 3-29, Sequence Advance Options

Stepped – this is similar to auto advance mode, except the generator places all jump bit flags to "1" and therefore, each step is advanced to the next only after a valid jump event signal has been asserted to the proper input.

To access the advance mode parameter, select Sequence from the Function group, then select the Config soft button. The advance mode options, as shown in Figure 3-28, will display. Use the dial or arrow keys to scroll down to the required mode and press Enter to lock in the selected mode.

Sequences and Run Mode Options

Run Mode options define if the sequenced waveforms will be generated continuously, triggered, or gated. Refer to the Selecting Run Modes section of this manual to learn more about these functions. Table 3-8 provides information on the sequenced waveform function, in conjunction with the run mode options. This table is useful, as it summarized all the controls that start and stop the waveforms. The Glossary defines the terms used in this table.

Table 3-9, Sequence Advance in Various Run Mode Options

Run Mode	Waveform	Advance Mode	Arm Options	Idle Waveform	Enable Signal	Abort Signal	Initiate Signal	Wave Loops	Seq Loops	Jump Flag	Jump Signal	Smart Trig
Continuous	Sequenced	Auto	Self Armed	Sequence	-	-	- (*)	1-1M	-	Bit (0 | 1)	Event	-
		-	Armed	Wave	Trigger | BUS | Event	F.P.| BUS	-	1-1M	-	Bit (0 | 1)	Event	-
		Once	Armed	Wave	Trigger | BUS | Event	F.P.| BUS	-	1-1M	1-1M	Bit (0 | 1)	-	-
		Stepped	Armed	Wave	Trigger | BUS | Event	F.P.| BUS	-	-	-	-	Event	-
	Adv'd Seq.	Auto	Self Armed	-		-	-	1-1M	-	Bit (0 | 1)	Event	-
		-	Armed	Sequence	Trigger | BUS | Event	F.P.| BUS	-	1-1M	-	Bit (0 | 1)	Event	-
		Once	Armed	Sequence	Trigger | BUS | Event	F.P.| BUS	-	1-1M	1-1M	Bit (0 | 1)	-	-
		Stepped	Armed	Sequence	Trigger | BUS | Event	F.P.| BUS	-	-	-	-	Event	-
Triggered	Sequenced	Auto	Self Armed	-	-	-	-	-	-	-	-	-	-
			Armed	DC	-	BUS	F.P.| BUS|Trigger|Event	1-1M	-	Bit (0 | 1)	Event	Yes
		Once	Armed	DC	-	BUS	F.P.| BUS|Trigger|Event	1-1M	1-1M	Bit (0 | 1)	Event	-
		Stepped	Armed	DC	-	BUS	F.P.| BUS|Trigger|Event	-	-	-	Event	Yes
	Adv'd Seq.	Auto	Self Armed	-	-	-	-	-	-	-	-	-	-
			Armed	DC	-	BUS	F.P.| BUS|Trigger|Event	1-1M	-	Bit (0 | 1)	Event	Yes
		Once	Armed	DC	-	BUS	F.P.| BUS|Trigger|Event	1-1M	1-1M	Bit (0 | 1)	Event	-
		Stepped	Armed	DC	-	BUS	F.P.| BUS|Trigger|Event	-	-	-	Event	Yes
Gated	Sequenced	Auto	Self Armed	-	-	-	-	-	-	-	-	-	-
			Armed	DC	-	BUS	Trigger|Event	1-1M	-	Bit (0 | 1)	Event	-
		Once	Armed	-	-	-	-	-	-	-	-	-	-
		Stepped	Armed	-	-	-	-	-	-	-	-	-	-
	Adv'd Seq.	Auto	Self Armed	-	-	-	-	-	-	-	-	-	-
			Armed	DC	-	BUS	Trigger|Event	1-1M	1-1M	Bit (0 | 1)	Event	-
		Once	-	-	-	-	-	-	-	-	-	-	-
		Stepped	-	-	-	-	-	-	-	-	Event	-

- (*) – denotes not relevant

Glossary of Terms

Run Mode – defines basic instrument run mode options:

Continuous – waveform is generated continuously.

Triggered – output waits for trigger, waveform is generated once.

Gated – output waits for gating signal, waveform is generated as long as the gating signal is true.

Waveform – Sequenced or advanced sequencing waveforms, waveform coordinates must be downloaded to the instrument and sequence tables updated to generate these waveforms.

Advance Mode – basic sequence advance modes, relevant for sequenced waveforms and sequenced sequences only:

Auto – a sequence is generated continuously.

Once – a sequence is generated once.

Stepped – *the* sequence waits for an event to jump to the next step.

Arm Options – defines if a waveform is self-enabled or requires a control signal to initiate.

Self Armed – the output does not require a control signal to generate waveforms.

Armed – a control signal is required to enable waveform generation.

Idle Waveform – defines the state of the output waveform before an event (enable, trigger or gate) initiates a waveform.

DC – first waveform point.

Wave – first waveform in a sequence.

Enable Signal – defines the source from where an enable signal is expected. Affects the output only when Arm Option is set to Armed.

BUS – a remote command enables the output.

Event – a transition at the event input enables the output.

Trigger – a valid signal at the trigger input enables the output.
Abort Signal – defines the source from where an abort signal is expected.

BUS – a remote command aborts all output activities. This signal does not affect the generator when set to operate in self armed mode.

F.P. – a front panel push button is used to abort all outputs activities

Initiate Signal – defines the wait for trigger entry port to initiate a waveform. Relevant for trigger and gated run modes only.

F.P. – a front panel push button is used to initiate a waveform or a sequence.

BUS – a remote command initiates a waveform or a sequence. Selecting this option automatically disables front panel operation.

Trigger – a valid signal at the trigger input initiates a waveform

or a sequence.
Event – a transition at the event input initiates a waveform or sequence.

Wave Loops – defines how many times the waveform repeats after a valid Initiate Signal.

Seq Loops – defines how many times the sequence and sequenced sequences repeat after a valid Initiate Signal.

Jump Flag – marks a step in a sequence as either wait for an event to jump or unconditional jump.
Bit = 0 – unconditional jump to the next step upon completion of the waveform loops.
.***Bit = 1*** – dwell on current step and wait for an event to jump to the next step.

Jump Signal – defines the entry port for a signal to cause a jump
Event – assigns the event input as the jump signal.

Smart Trigger – defines if smart trigger features are available for a specific function.

Using Advanced Sequencing

The advanced sequencing is, in fact, very similar to the standard sequence operation except, instead of waveforms, the sequence generator steps through complete sequences. Of course, it is a bit tedious to prepare so many waveforms and build so many sequence scenarios, and it would be much easier to do it from a remote computer but, you can also do simple advance sequencing from the front panel, as well.

The first time you select the sequenced waveforms function, the generator defaults to Normal sequencing. To change the mode to advanced sequencing, select the Config soft key and then the Sequence Type soft key from the configuration controls. Then select the Advanced option, as shown in Figure 3-29. Finally, when you select the View Table soft key, you'll notice that the table shows sequence instead of waveforms. The rest is pretty straightforward, as it operates exactly as described for normal sequence scenarios.

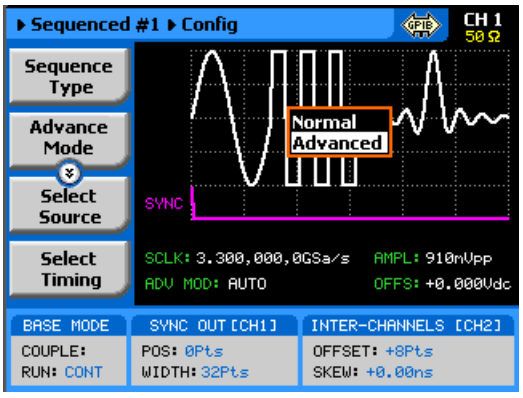

Figure 3-30, Selecting the Advanced Sequencing Option

Using the Dynamic Sequence Control

A control input is available on the SE5082 rear panel that provides control over the replay of a specific sequence or segment. The control connector has 9 pins and therefore allows replay of one of 256 sequences. A valid line must be asserted to validate sequence change. Having the dynamic control feature, in effect, can serve as replacement of the advanced sequence table where the real-time application can decide when and for how long a sequence will be generated.

The dynamic control input accepts TTL level signals. Bit 1 is connected to pin 1; it weighs 2^0 binary (decimal 1) and bit 8 is connected to pin 9 and weighs 2^7 binary (decimal 128) so if all lines are low, the output generates segment number 1 (the firmware counts the sequences from 0 to 255) and if all line are high, the output generates sequence 256. Any combination from 0 to 255 can be produced at the inputs of this connector but the output will change only when a valid signal is asserted at pin 9 of the same connector. Figure 3-30 shows a typical example of external sequence control; the first valid transition selects sequence 37, the second selects sequence 57 and the third transitions selects sequence 194.

Two dynamic control inputs are available on the rear panel, one for channel 1 and one for channel 2, and each channel can be modified to have a specific output sequence. Note, however, that if you intend to synchronize the outputs, the sequences that are selected with this control must have exactly the same length.

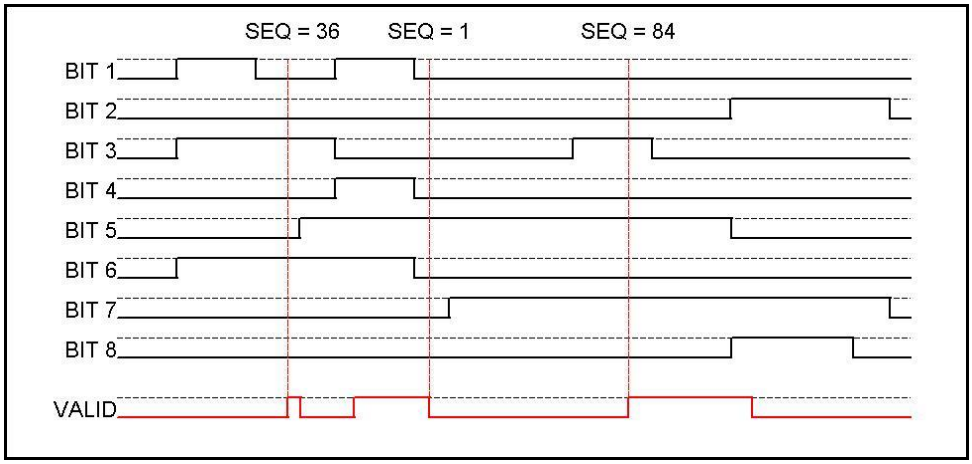

Figure 3-31, Typical Dynamic Sequence Control Scenario

Generating Modulated Waveforms

The SE5082**,** being a digital instrument, has the advantage of being able to generate almost any waveform through its DAC, provided that there are enough waveform points to compute the function. Normally, complex modulation schemes will be computed on an external computer and downloaded to the generator as waveform points, but for simpler modulation schemes, the equation already resides in the instrument. The SE5082 can generate AM, FM, Sweep, Chirp, FSK, ASK, Frequency hop and Amplitude hop. The advantage of these waveforms is that they are commonly used and when produced, they are extremely accurate.

The main point to keep in mind is that the modulation waveforms are computed and recomputed, every time a parameter is modified. This is a bit different from analog instruments in the way that the output must stop to re-compute the waveform, and then it comes back to life. The next point to consider is the modulation barriers, because the longer and finer the modulation is, the more waveform points are required to generate the function and therefore, it takes more time to compute the waveform and load it to the memory. Normal computation time is around a few milliseconds, but for lower frequencies, it might take a few seconds.

The modulation function is turned on immediately after you press the Modulation button in the Functions group on the right side of the generator. By default, modulation is turned off and therefore, the output generates carrier wave (CW). The first modulation screen is shown in Figure 3-31. The modulation functions and their parameters are described as follows:

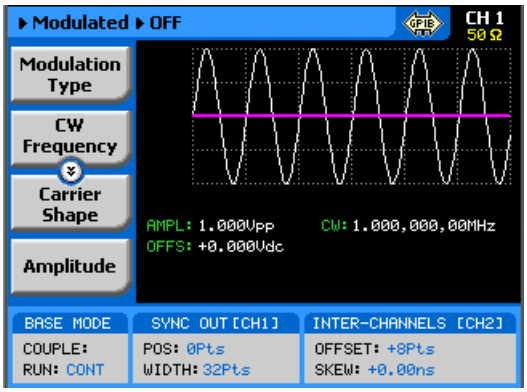

Figure 3-32, The Modulation Function Main Screen

Tip

The values of the three parameters shown in the Modulation Off display are duplicated in all modulation panels.

Off

The Modulation Off is a special instance of the modulation function, where the output is not modulated, but generates carrier waveform (CW) frequency only. CW can be sine, triangle, or square waveforms. When placed in Modulation Off, the CW is generated from the main outputs continuously. The CW parameter does not change when you switch from one modulation function to another. Figure 3-32 shows the Modulation Off menus.

While in the Off option, there are some parameters that can be programmed for the carrier waveform:

Modulation Type – defines the type of modulation scheme that will modulate the carrier waveform: Off, FM, FSK, Frequency Hop, Sweep, Chirp, AM, ASK, Amplitude Hop. The selected modulation scheme is generated immediately at the output terminals.

CW Frequency – defines the frequency of the carrier waveform. The CW parameter, as programmed in this menu is shared by all other modulation options.

Carrier Shape – defines the shape of the carrier waveform. The carrier shape can be selected from Sine, Triangle and Square. Note that the Carrier Shape parameter is used in all of the modulation displays, so if you change the shape for one function, it affects the other modulation displays as well.

Amplitude – defines the carrier amplitude level. The same level is used throughout the instrument when you move from waveform shape to another. The Amplitude parameter, as programmed in this menu, is shared by all other waveform options.

Offset – defines the offset level for the carrier waveforms. The same level is used throughout the instrument when you move from one waveform shape to another. The Offset parameter, as programmed in this menu, is shared by all other waveform options.

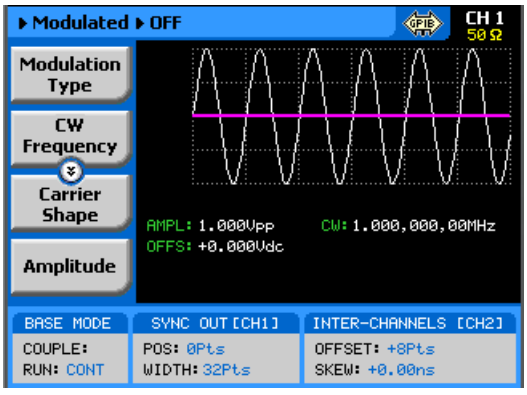

Figure 3-33, Modulation OFF Parameters

FM

The FM function allows frequency modulation of a carrier waveform (CW). The carrier waveform is modulated by an internal waveform, normally referred to as modulating waveform. The shape of the modulating waveform can be selected from sine, triangle, square or arbitrary waveforms.

The FM function has a number of menus that control the modulation parameters. These are shown in Figure 3-33 and described in the following paragraphs:

Modulation Type – defines the type of modulation scheme that will modulate the carrier waveform: Off, FM, FSK, Frequency Hop, Sweep, Chirp, AM, ASK and Amplitude Hop. The selected modulation scheme is generated immediately at the output terminals.

Modulation Shape – Defines the shape and type of the modulating waveform: Sine Triangle, Square and Ramp. The Modulation Shape menu that provides access to the selection of the envelope waveform is shown in Figure 3-34.

CW Frequency – defines the frequency of the carrier waveform. Using this standard FM function, the shape of the carrier waveform is always sine.

Frequency Deviation – defines the range of frequencies that the modulation will go through. The peak value is symmetrical around the value of the carrier waveform frequency.

Modulation Frequency – defines the frequency of the modulating waveform. The modulating waveform is programmed from 100Hz to 250MHz.

Marker– programs a unique frequency where the SYNC output generates a pulse to mark this frequency.

Carrier Shape – defines the shape of the carrier waveform. The carrier shape can be selected from Sine, Triangle and Square. Note that the Carrier Shape parameter is used in all of the modulation displays, so if you change the shape for one function, it affects the other modulation displays as well.

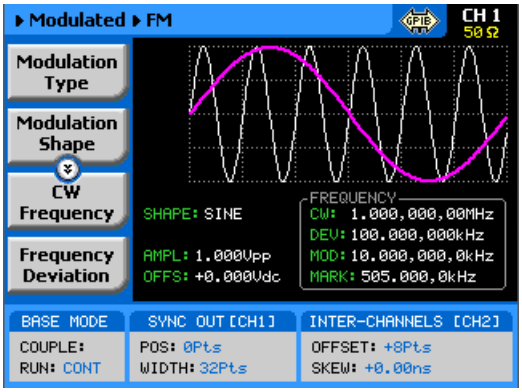

Figure 3-34, FM Menus

Amplitude – defines the carrier amplitude level. The same level is used throughout the instrument, when you move from one waveform shape to another.

Offset – defines the offset level for the carrier waveforms. The same level is used throughout the instrument when you move from one waveform shape to another.

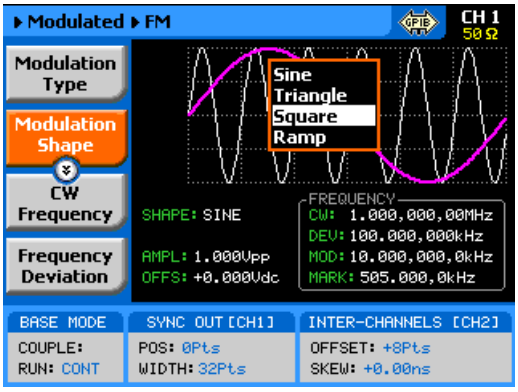

Figure 3-35, Modulation Waveform Shapes

FSK

FSK (Frequency Shift keying) modulation allows frequency hops between two pre-programmed frequencies: Carrier Waveform Frequency and Shifted Frequency. Note that CW is sinewave only and that the switch between two frequencies is always coherent.

The CW and shifted frequencies can be programmed with 10 digits throughout the entire frequency range of the instrument. The FSK sequence is designed in an FSK table that can either be loaded from the front panel or downloaded from a remote interface from a utility such as ArbConnection. An example of the FSK table, as created in ArbConnection, is shown in Figure 3-35.

When you select FSK modulation, the parameters, as shown in Figure 3-36 and described in the following paragraphs, will be

available for modification:

Modulation Type – defines the type of modulation scheme that will modulate the carrier waveform: Off, FM, FSK, Frequency Hop, Sweep, Chirp, AM, ASK and Amplitude Hop. The selected modulation scheme is generated immediately at the output terminals.

FSK Data – defines the sequence of which the frequencies will toggle. FSK data is stored in an external table. The FSK Data table contains a list of "0"s and "1"s which determine the sequence. "0" defines CW and "1" defines shifted frequency.

CW Frequency – defines the frequency of the carrier waveform. In this case, the CW frequency will also be used as the idle frequency. Using this standard FSK function, the shape of the carrier waveform is always sine.

Shifted Frequency – defines the frequency to which the generator will shift when logic level "1" is sensed at the trigger input.

Baud – defines the rate at which the frequencies are toggled. The rate can be programmed within the range of 0.1 bits/s to 500 Mbits/s.

Marker – defines an index point in the FSK sequence where the SYNC output will generate a marker pulse.

Carrier Shape – defines the shape of the carrier waveform. The carrier shape can be selected from Sine, Triangle and Square. Note that the Carrier Shape parameter is used in all of the modulation displays, so if you change the shape for one function, it affects the other modulation displays as well.

Amplitude – defines the carrier amplitude level. The same level is used throughout the instrument when you move from waveform shape to another.

Offset – defines the offset level for the carrier waveforms. The same level is used throughout the instrument when you move from waveform shape to another.

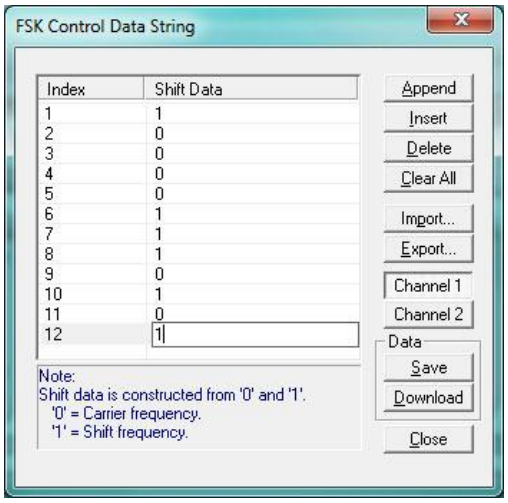

Figure 3-36, ArbConnection FSK Control Data String Example

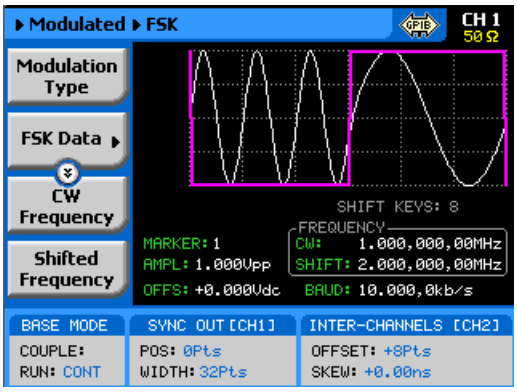

Figure 3-37, FSK Menus

Frequency Hop

In frequency hop mode, the output waveform (sinewave) hops from frequency to frequency in a sequence defined by the hop table. Frequencies can be programmed with 11 digits resolution throughout the entire range of the instrument. There are two frequency hop types:

1. Frequency hops with fixed dwell time
2. Frequency hops with variable dwell time

Dwell time defines the time that the frequency will remain stable and will change at the end of this interval. With the fixed dwell time, the waveform hops from frequency to frequency at constant intervals, defined by the fixed dwell time parameter. When the variable dwell time is selected, each hop can be programmed to

have a unique dwell interval.

When you select frequency hop modulation, the menus, as shown in Figure 3-37 and described in the following paragraphs, will be available for modification:

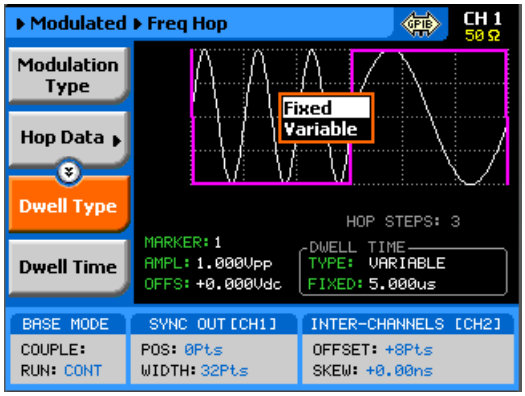

Figure 3-38, Frequency Hop Menus

Modulation Type – defines the type of modulation scheme that will modulate the carrier waveform: Off, FM, FSK, Frequency Hop, Sweep, Chirp, AM, ASK and Amplitude Hop. The selected modulation scheme is generated immediately at the output terminals.

Hop Data – allows programming and editing of the frequency hop table. The hop data table contains a list of frequencies and the generator will hop through these frequencies in the same order and at a rate defined by the dwell time setting. Hop data length and frequency range as specified in Appendix A.

The frequency hop table depends on the Dwell Type selection and will display the dwell time parameter only if the variable dwell time option has been selected. The hop table as shown in Figure 3-38 is an example how such table are programmed from ArbConnection.

Dwell Type – defines if each hop step will have constant or variable dwell times. Using the variable time option, each step can be programmed to have a unique dwell time value.

Dwell Time – defines the lapse of time for a hop step, when the variable dwell time option has been selected.

Marker– programs a unique index point where the SYNC output generates a pulse to mark a specific hop step.

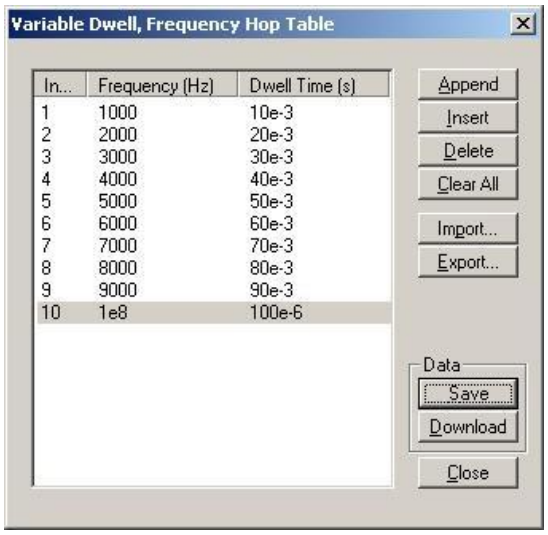

Figure 3-39, Variable Dwell Time Frequency Hop Table Example

Carrier Shape – defines the shape of the carrier waveform. The carrier shape can be selected from Sine, Triangle and Square. Note that the Carrier Shape parameter is used in all of the modulation displays, so if you change the shape for one function, it affects the other modulation displays as well.

Amplitude – defines the carrier amplitude level. The same level is used throughout the instrument when you move from waveform shape to another.

Offset – defines the offset level for the carrier waveforms. The same level is used throughout the instrument when you move from waveform shape to another.

Sweep

Sweep modulation allows carrier waveform (CW) to sweep from one frequency, as defined by the sweep start parameter, to another frequency, as defined by the sweep stop parameter.

When you select sweep modulation, the menus, as shown in Figure 3-39 and described in the following paragraphs, will be available for modification.

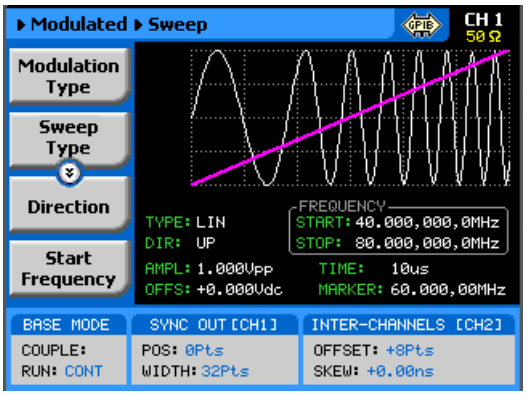

Figure 3-40, Sweep Menus

Modulation Type – defines the type of modulation scheme that will modulate the carrier waveform: Off, FM, FSK, Frequency Hop, Sweep, Chirp, AM, ASK and Amplitude Hop. The selected modulation scheme is generated immediately at the output terminals.

Sweep Type – defines the sweep steps of which the frequency increments or decrements from start to stop frequencies. A choice is provided between linear and logarithmic. If you select linear sweep, the carrier waveform frequency steps through the frequencies within a time interval, which is set by the sweep time parameter. Likewise, using the logarithmic sweep type, the frequency span between the start and stop frequencies is stepped through using logarithmic steps.

Direction – defines the sweep direction. UP sets sweep direction from start frequency to stop frequency. DOWN reverses the sweep direction, so the output sweeps from stop frequency to start frequency.

Start Frequency – defines the frequency value of which the generator will start its sweep. Note that the sweep start can be at a higher frequency value, depending on the sweep direction setting.

Stop Frequency – defines the frequency value of which the generator will stop its sweep. Note that the sweep stop can be at a lower frequency value, depending on the sweep direction setting.

Sweep Time – defines the time that will lapse from sweep start to sweep stop frequencies.

Marker – defines a frequency of which, when transitioned through, will output a marker pulse at the SYNC output connector.

Carrier Shape – defines the shape of the carrier waveform. The carrier shape can be selected from Sine, Triangle and Square. Note that the Carrier Shape parameter is used in all of the modulation displays so if you change the shape for one function, it affects the other modulation displays as well.

Amplitude – defines the carrier amplitude level. The same level is

used throughout the instrument when you move from waveform shape to another.

Offset – defines the offset level for the carrier waveforms. The same level is used throughout the instrument when you move from waveform shape to another.

Chirp

Chirp modulation allows carrier waveform (CW) to sweep from one frequency, as defined by the sweep start parameter, to another frequency, as defined by the sweep stop parameter while, at the same time, swing through an amplitude range, as defined by the amplitude and AM Depth index.

When you select chirp modulation, the menus, as shown in Figure 3-40 and described in the following paragraphs, will be available for modification.

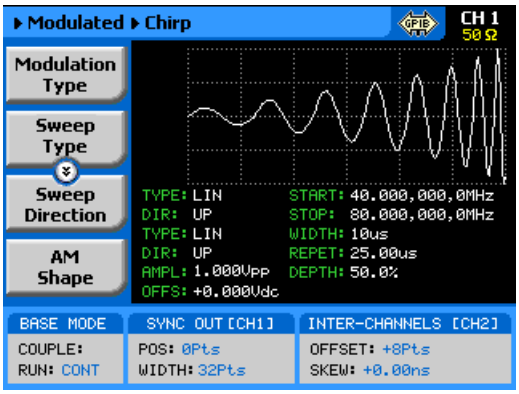

Figure 3-41, Chirp Menus

Modulation Type – defines the type of modulation scheme that will modulate the carrier waveform: Off, FM, FSK, Frequency Hop, Sweep, Chirp, AM, ASK and Amplitude Hop. The selected modulation scheme is generated immediately at the output terminals.

Sweep Type – defines the sweep steps of which the frequency increments or decrements from start to stop frequencies. A choice is provided between linear and logarithmic. If you select linear sweep, the carrier waveform frequency steps through the frequencies within a time interval, which is set by the sweep time parameter. Likewise, using the logarithmic sweep type, the frequency span between the start and stop frequencies is stepped through using logarithmic steps.

Sweep Direction – defines the sweep direction. UP sets sweep direction from start frequency to stop frequency. DOWN reverses the sweep direction, so the output sweeps from stop frequency to start frequency.

AM Shape – defines the amplitude steps of which the amplitude

increments or decrements from amplitude setting to the depth index value. A choice is provided between linear and logarithmic. If you select linear step, the amplitude steps linearly through the amplitude range. Likewise, using the logarithmic step type, the amplitude steps through the amplitude span using logarithmic steps.

AM Direction – defines the sweep direction. UP sets sweep direction from index value to amplitude setting. DOWN reverses the process.

Start Frequency – defines the frequency value of which the generator will start its sweep. Note that the sweep start can be at a higher frequency value, depending on the sweep direction setting.

Stop Frequency – defines the frequency value of which the generator will stop its sweep. Note that the sweep stop can be at a lower frequency value, depending on the sweep direction setting.

Pulse Width – defines the time that will lapse from sweep start to sweep stop frequencies. In other words, it defines the width of the entire chirp pulse.

Pulse Repetition – defines the time that will lapse from the start of one chirp sequence to the start of the next chirp sequence.

AM Depth – defines an index number of which references to the amplitude level value. 100% defines full amplitude span and 0% defines no signal (dc value).

Carrier Shape – defines the shape of the carrier waveform. The carrier shape can be selected from Sine, Triangle and Square. Note that the Carrier Shape parameter is used in all of the modulation displays so if you change the shape for one function, it affects the other modulation displays as well.

Amplitude – defines the carrier amplitude level. The same level is used throughout the instrument when you move from waveform shape to another.

Offset – defines the offset level for the carrier waveforms. The same level is used throughout the instrument when you move from waveform shape to another.

AM

The AM function enables amplitude modulation of a carrier waveform (CW). The carrier waveform is modulated by an internal waveform, normally referred to as envelope waveform. The envelope waveform can be selected from sine, triangle square or ramp shapes. When AM is selected, the menus that are associated with AM become accessible. These are shown in Figure 3-41.

There are other parameters that control how the CW is amplitude modulated. These are:

Modulation Type – defines the type of modulation scheme that will modulate the carrier waveform: Off, FM, FSK, Frequency Hop,

Sweep, Chirp, AM, ASK and Amplitude Hop. The selected modulation scheme is generated immediately at the output terminals.

Modulation Shape – defines the envelope function. There are four shapes that can be used: Sine, Triangle, Square and Ramp. The Modulation Shape menu that provides access to the selection of the envelope waveform is shown in Figure 3-42.

Modulation depth – programmed in units of % and defines the depth of the modulating envelope. Modulation depth is programmed from 0% to 200%.

Modulation Frequency – defines the frequency of the modulating waveform. The modulating waveform is programmed from 100Hz to 1MHz.

CW Frequency – defines the frequency of the carrier waveform.

Carrier Shape – defines the shape of the carrier waveform. The carrier shape can be selected from Sine, Triangle and Square. Note that the Carrier Shape parameter is used in all of the modulation displays, so if you change the shape for one function, it affects the other modulation displays as well.

Amplitude – defines the carrier amplitude level. The same level is used throughout the instrument when you move from waveform shape to another.

Offset – defines the offset level for the carrier waveforms. The same level is used throughout the instrument when you move from one waveform shape to another.

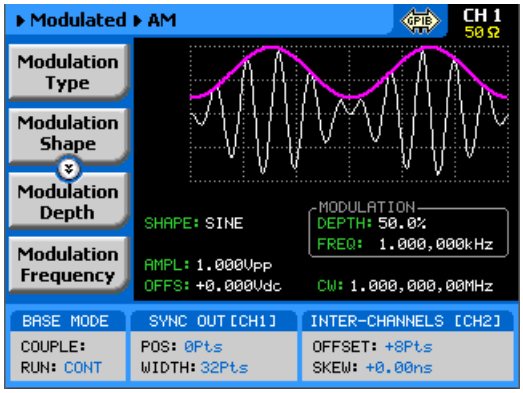

Figure 3-42, AM Menus

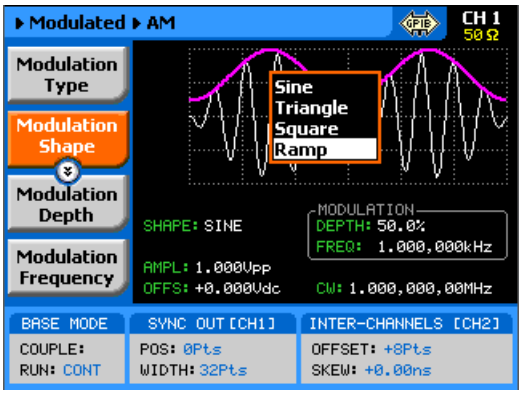

Figure 3-43, Modulating Waveform Shapes

ASK

ASK (Amplitude Shift keying) modulation allows amplitude hops between two pre-programmed amplitude levels. Note that sine wave CW only is hopped. The signal level can hop between two amplitude levels throughout the entire amplitude range without crossing range or relay ranges. Amplitude is programmed with 4 digits of resolution.

The ASK sequence is designed in an ASK table that can either be loaded from the front panel or downloaded from a remote interface from a utility such as ArbConnection. An example of the ASK table, as created in ArbConnection, is shown in Figure 3-43.

When you select ASK modulation, the parameters, as shown in Figure 3-44 and described in the following paragraphs, will be available for modification:

Modulation Type – defines the type of modulation scheme that will modulate the carrier waveform: Off, FM, FSK, Frequency Hop, Sweep, Chirp, AM, ASK and Amplitude Hop. The selected modulation scheme is generated immediately at the output terminals.

ASK Data – defines the sequence of which the amplitudes will toggle. ASK data is stored in an external table. The length of the table is limited from 1 to 4096 toggle sequences. The ASK Data table contains a list of “0”s and “1”s which determine the sequence. “0” defines the initial amplitude and “1” the shifted amplitude.

CW Frequency – defines the frequency of the carrier waveform. In this case, the CW frequency will also be used as the idle frequency. Using this standard ASK function, the shape of the carrier waveform is always sine.

Start Amplitude – defines the initial amplitude level. Note that the start amplitude does not necessarily define lower value than the stop amplitude.

Shifted Amplitude – defines the amplitude level of which the generator will shift when logic level “1” is sensed at the trigger input.

Note that the stop amplitude does not necessarily define higher value than the start amplitude.

Baud – defines the rate of which the amplitude is toggled. The rate can be programmed within the range 0.1 bits/s to 500Mbits/s.

Marker – defines an index point in the ASK sequence where the SYNC output will generate a marker pulse.

Offset – defines the offset level for the carrier waveforms. The same level is used throughout the instrument when you move from waveform shape to another.

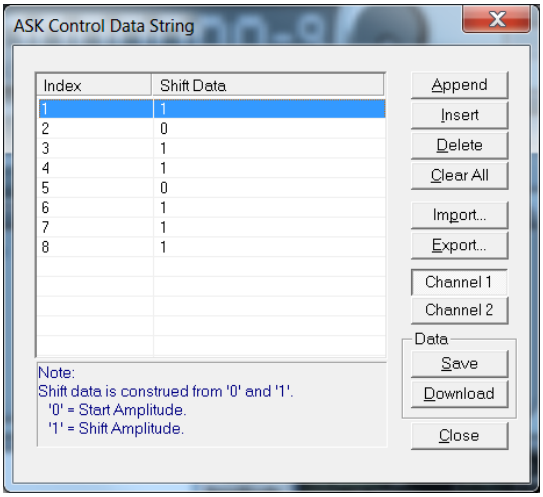

Figure 3-44, ArbConnection ASK Control Data String Example

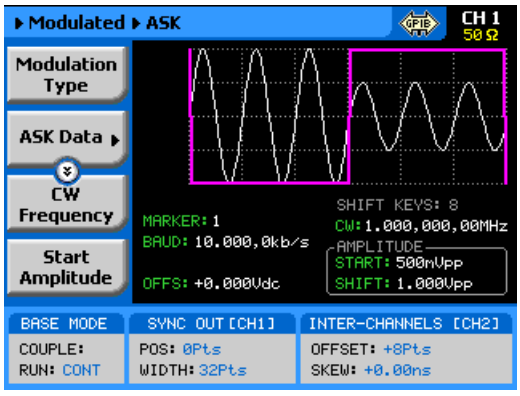

Figure 3-45, ASK Menus

Amplitude Hop

In Amplitude hop mode, the output waveform (sinewave) hops from amplitude to amplitude in a sequence defined by the hop table. Amplitude hop modulation allows amplitude hops throughout the entire range of the instrument. The base signal is always CW (sine waveform).

There are two amplitude hop types:

1. Amplitude hops with fixed dwell time and
2. Amplitude hops with variable dwell time

Dwell time defines the time that the amplitude will remain stable and will change at the end of this interval. With the fixed dwell time, the waveform hops from amplitude to amplitude at constant intervals, defined by the fixed dwell time parameter. When the variable dwell time is selected, each hop can be programmed to have a unique dwell interval.

When you select the amplitude hop modulation, the menus, as shown in Figure 3-45 and described in the following paragraphs, will be available for modification.

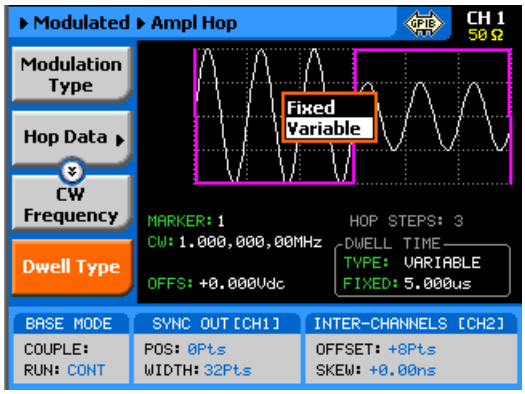

Figure 3-46, Amplitude Hop Menus

Modulation Type – defines the type of modulation scheme that will modulate the carrier waveform: Off, FM, FSK, Frequency Hop, Sweep, Chirp, AM, ASK and Amplitude Hop. The selected modulation scheme is generated immediately at the output terminals.

Hop Data – allows programming and editing of the amplitude hop table. The hop data table contains a list of amplitudes and the generator will hop through these amplitudes in the same order and at a rate defined by the dwell time setting.

The amplitude hop table depends on the Dwell Type selection and will display the dwell time parameter only if the variable dwell time option has been selected. The hop table as shown in Figure 3-46 is an example how such table are programmed from ArbConnection.

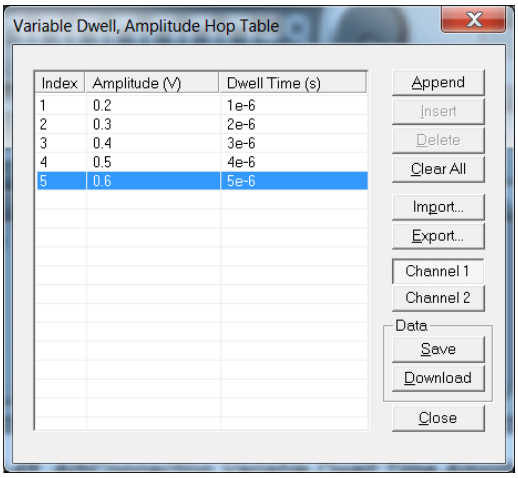

Figure 3-47, ArbConnection Variable Dwell Time Amplitude Hop Table Example

CW Frequency – defines the frequency of the carrier waveform. The shape of the carrier waveform is always sine.

Dwell Type – defines if each hop step will have constant or variable dwell times. Using the variable time option, each step can be programmed to have a unique dwell time value.

Dwell Time – defines the lapse of time for a hop step, when the variable dwell time option has been selected. Dwell time is programmable in steps of 1 ns from 2 ns to 10 seconds.

Carrier Shape – defines the shape of the carrier waveform. The carrier shape can be selected from Sine, Triangle and Square. Note that the Carrier Shape parameter is used in all of the modulation displays so if you change the shape for one function, it affects the other modulation displays as well.

Marker– programs a unique index point where the SYNC output generates a pulse to mark a specific hop step.

Amplitude – defines the carrier amplitude level. The same level is used throughout the instrument when you move from waveform shape to another.

Offset – defines the offset level for the carrier waveforms. The same level is used throughout the instrument when you move from waveform shape to another.

Modulated Waveforms and Run Mode Options

Run Mode options define if the modulated waveforms will be generated continuously, triggered, or gated. Refer to the Selecting Run Modes section of this manual to learn more about these functions. Table 3-9 provides information on the modulated waveform function in conjunction with the run mode options. This table is useful, as it summarizes all the controls that start and stop the waveforms. The Glossary defines the terms used in this table.

Table 3-10, Modulated Waveforms in Various Run Mode Options

Run Mode	Waveform	Arm Options	Idle Waveform	Enable Signal	Abort Signal	Initiate Signal	Wave Loops	Smart Trig
Continuous	Modulated	Self Armed	Wave	- (*)	-	-	-	-
		Armed	DC	Trigger | BUS | Event	F.P. | BUS	-	-	-
Triggered	Modulated	Self Armed	-	-	-	-	-	-
		Armed	DC	-	BUS	F.P.| BUS | Trigger | Event	1-1M	Yes
Gated	Modulated	Self Armed	-	-	-	-	-	-
		Armed	DC	-	BUS	Trigger | Event	-	-

- (*) – Not Relevant

Glossary of Terms

Run Mode – defines basic instrument run mode options:

Continuous – waveform is generated continuously.

Triggered – output waits for trigger, waveform is generated once.

Gated – output waits for gating signal, waveform is generated as long as the gating signal is true.

Waveform – modulated waveform is generated at the output.

Arm Options – defines if a waveform is self-enabled or requires a control signal to initiate.

Self Armed – the output does not require a control signal to generate waveforms.

Armed – a control signal is required to enable waveforms.

Idle Waveform – defines the state of the output waveform before an event (enable, trigger or gate) initiates a waveform.

DC – first waveform point.

Wave – first waveform in a sequence.

Enable Signal – defines the source from where an enable signal is expected. Affects the output only when Arm Option is set to Armed.

BUS – a remote command enables the output.

Event – a transition at the event input enables the output.

Trigger – a valid signal at the trigger input enables the output.

Abort Signal – defines the source from where an abort signal is expected.

BUS – a remote command aborts all output activities. This signal

does not affect the generator, when set to operate in self- armed mode.

F.P. – a front panel push button is used to abort all outputs activities.

Initiate Signal – defines the wait for trigger entry port to initiate a waveform. Relevant for trigger and gated run modes only.

F.P. – a front panel push button is used to initiate a waveform or a sequence.

BUS – a remote command initiates a waveform or a sequence. Selecting this option automatically disables front panel operation.

Trigger – a valid signal at the trigger input initiates a waveform or a sequence.

Event – a transition at the event input initiates a waveform or sequence.

Wave Loops – defines how many times the waveform repeats after a valid Initiate Signal.

Smart Trigger – defines if smart trigger features are available for a specific function.

Generating Pulse/Pattern Waveforms

The SE5082 employs a complete set of menus that are dedicated to the pulse/pattern function. Press on the Pulse in the Function group and the output will immediately be updated with the pulse waveform and the display will be modified to show pulse parameters, as shown in Figure 3-55.

The pulse function provides the means for designing pulses and their associated parameters, exactly as would be done on a stand-alone analog pulse generator. Note, however, that the pulse is built in the same memory as the arbitrary waveforms are being stored and therefore, changing from arbitrary to digital pulse mode and vice versa may overwrite waveforms that were downloaded to the memory.

If this is the first time that you enter this menu, the output will generate a pulse with the following properties:

Pulse Mode: Single
Period = 10 ms
Pulse Width = 2 ms
High Level = 1 V
Low Level = 0 V
Polarity = Normal
Transitions = Fast

By default, pulse timing is set in units of time (seconds). However, this can be changed so that some parameters are set as ratios between the parameter to the pulse period. This option will be discussed later in this section of the manual.

Note

The pulse shape on the SE5082 LCD display is an icon only. The actual waveform at the output terminals may look entirely different.

The pulse generator menus provide access to all pulse parameters, just as they would be programmed on an analog pulse generator. To access the pulse parameters, use one of the soft keys. If you do not see a required parameter on the screen, press the key up or down to scroll through the menus.

There are many pulse parameters that can be displayed on the screen. However, most applications do not require that all pulse parameters be displayed and therefore, the menus were built in a way that show all of the parameters that are absolutely necessary to program the selected pulse mode. For example, if single pulse mode with fast transitions is required, the delay, rise and fall time parameters will not be shown on the screen. Information on how to program simple and complex pulse modes is given in the following paragraphs.

The technique of changing parameter values is exactly the same as you are using to modify standard waveform parameters. Simply press the soft key that is associated with the parameter, then punch in the numbers using the numeric keyboard and complete the modification by assigning a suffix and pressing the Enter button.

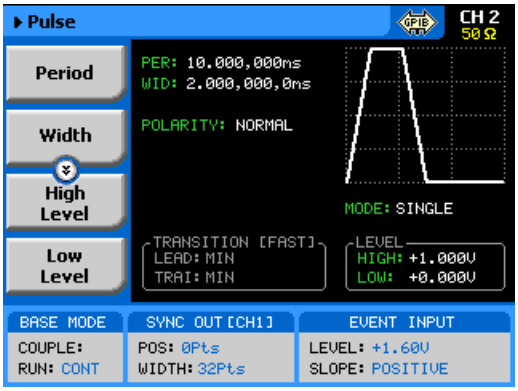

Figure 3-48, Pulse Function Menus

Adjusting the pulse shape with the required characteristics can only be done if all of its parameters can be adjusted both in the time and amplitude domains. The SE5082 provides all the necessary controls to do just that. However, always keep in mind that some parameters may interfere during the programming sequence and cause setting conflicts. These setting conflicts will be discussed later in this chapter. Below you will find a list of all pulse parameters that you can access though the soft key menus.

The main menu, as shown in Figure 3-47, provides access to five soft key buttons: Period, Width, High Level, Low level and Config. Pressing the TOP soft key from anywhere in the pulse menus will restore the display to the one shown in Figure 3-47.

Period – programs the pulse period in unit of seconds.

Width – programs the pulse width in units of seconds. The pulse width can be set as a percent of the period when the Pulse Parameters option in the Config menu has been modified from Time to Percent.

High Level – programs the pulse high-level value in units of volts.

Low Level – programs the pulse low-level value in units of volts.

Config – provides access to a new menu that has pulse configuration controls. The configuration menu is shown in Figure 3-48 and its controls are described below. There are five control parameters in this menu: Pulse Parameters, Pulse Mode, Polarity, Transition Type and Level Control. Each of these controls has a major impact on pulse.

Pulse Composer - opens a table that programs pulse patterns that have mixed transition times.

Pattern – provide access to a new menu that enables generating different pattern scenarios.

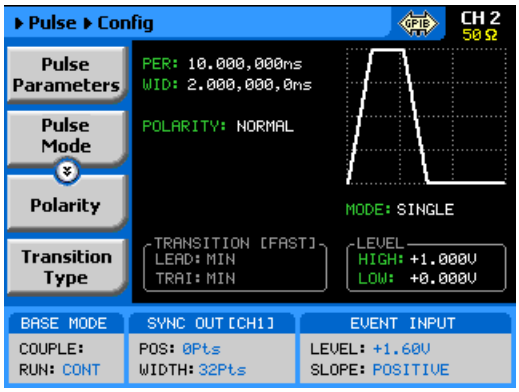

Figure 3-49, Pulse Configuration Menus

Pulse Parameters

The Pulse Parameters control lets you select if you want the pulse parameters programmed in units, time or as percent ratios to the pulse period. Pulse parameters are normally programmed in units of time and this is the default condition after reset or the first time you access this function. The advantage of using ratio is that the instrument computes the timing and keeps a fixed ratio of period-to-width whenever you change the period. The parameters available in this mode are: pulse width, pulse delay and pulse transition times. Amplitude and offset values are not affected when you select the ratio mode.

Understanding Pulse Parameters

Standard definitions of pulse parameters are not within the scope of this manual, but it is important to understand how the SE5082 interprets these definitions and how you should expect to see the pulse shape on monitoring devices, such as oscilloscopes.

Figure 3-49 depicts a common pulse shape as understood by most hardware engineers. The standard interpretation implies that the pulse width is measured between the edges, at the middle of the amplitude level and transitions are often specified from 10% to 90%. However, this is not always the case and some engineers, for example, prefer the 20% to 80% value as transitions and measure the high time of the pulse as the pulse width.

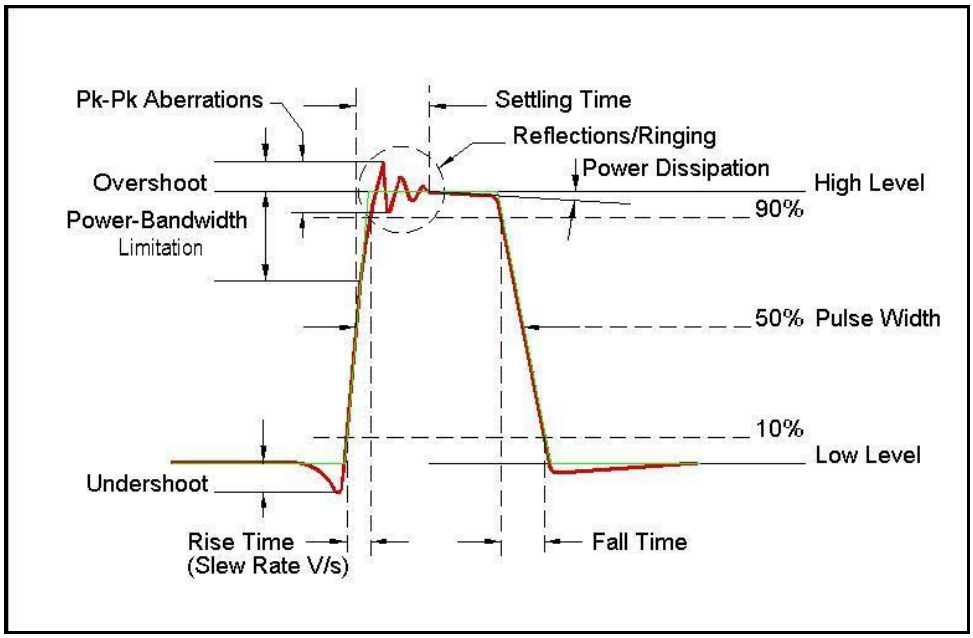

Figure 3-50, Standard Interpretation of Pulse Parameters

As mentioned above, the pulse function is computed and generated from the arbitrary memory and therefore, some of the parameters are interpreted differently, because it will take too long to compute and build digital waveforms the same way that analog components do. Standard or SE5082 interpretation, once the commonality and differences are defined, will be equally simple to design pulse characteristics with the pulse generator that is available in the SE5082.

Figure 3-50 shows how the SE5082 interprets the programmed pulse parameters. The differences are painted blue to highlight them. One major difference from standard analogy, is that the pulse width includes the time it take to transition from low level to the time it take to transition from high level.

The rest of the parameters were derived from these characteristics and should present little difficulty, especially when most of the pulse applications do not contain linear transitions, anyway.

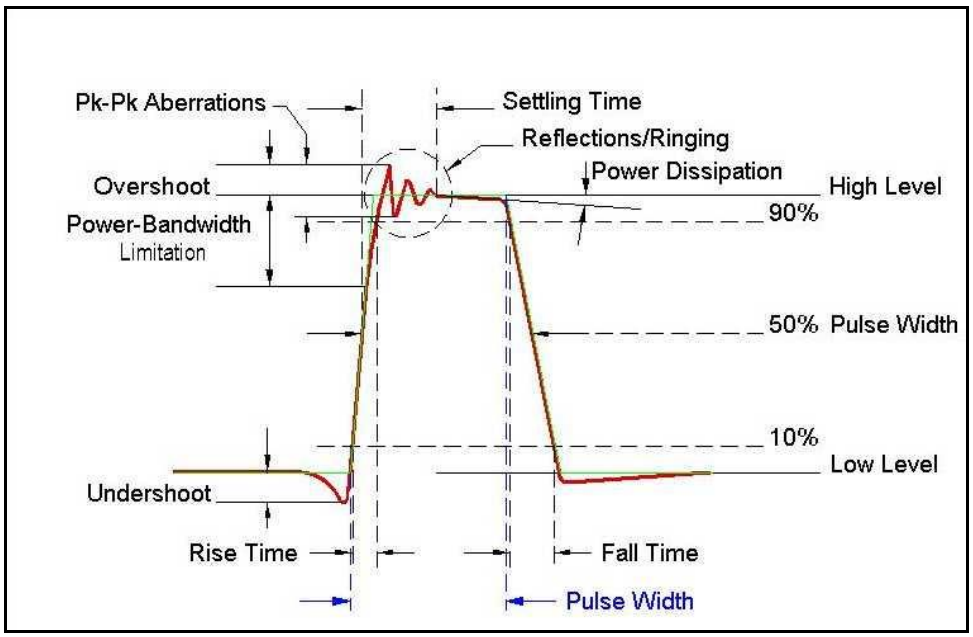

Figure 3-51, SE5082 Interpretation of Pulse Parameters

Pulse Modes

The pulse mode menu provides access to modifying the basic shape of the pulse waveform. These are: Single Pulse, Double Pulse and Delayed Pulse. These modes and their associated parameters are discussed below. To access the Pulse Modes options, press the soft key next to this parameter. The pulse mode options will become available, as shown in Figure 3-51. Scroll up or down the list and press enter to complete the selection process.

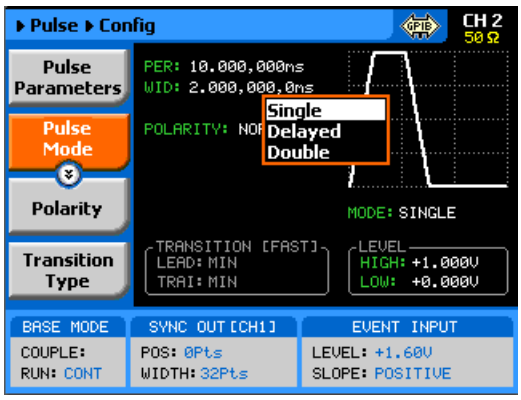

Figure 3-52, Pulse Mode Options

Single Pulse Mode

The basic pulse mode is the Single Pulse. Single pulse defines the shape of a single pulse only. In continuous operating mode, it may appear as a string of pulses that have constant period, width and amplitude. However, the meaning of this mode is that if you place the instrument in triggered run mode then only one pulse is initiated with every trigger. The menus that are associated with the simplest configuration of the single pulse mode are: Period, Width, High and Low Levels. These parameters are discussed below. With more complex configuration, you can select linear transitions and width and amplitude modes. However, these are described separately.

Figure 3-52 shows a typical single pulse shape and highlights all of its relevant parameters. While most of the parameters that are shown in Figure 3-52 can be programmed and adjusted for a specific application, some characteristics of the pulse are derived from the quality of the generator and its output stage. These are specified and can be found in Appendix A.

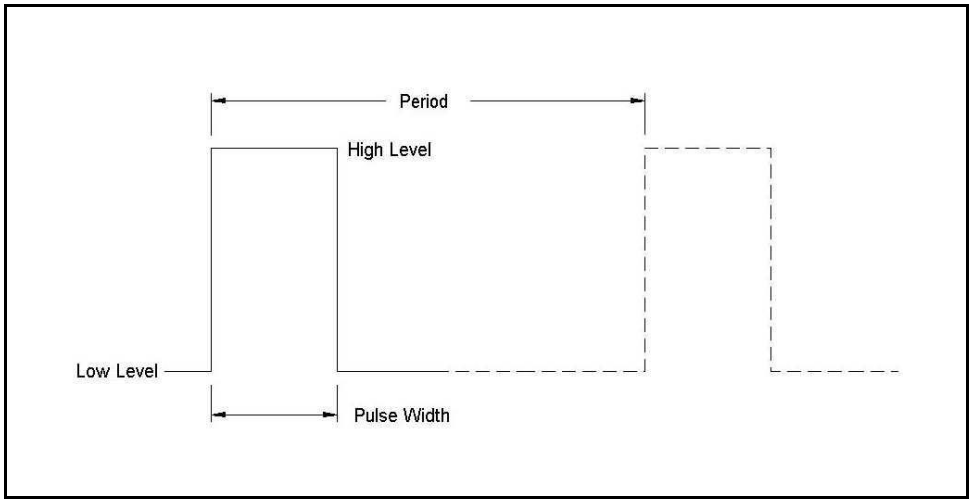

Figure 3-53, Single Pulse Parameters Summary

Note

Before you start programming pulse parameters, keep in mind that incorrect planning of your entries may cause setting conflicts that may, in turn, cause the instrument to ignore some parameters. A common example is if you try to program pulse width that is wider than the period. Therefore, always program the pulse parameters in increasing order. A list of setting conflicts is given later in this chapter.

When Single Pulse mode option has been selected, the main pulse menus will allow access to the following pulse parameters:

Period – programs the pulse period in unit of seconds.

Width – programs the pulse width in units of seconds. The pulse width can be set as a ratio to the period when the Pulse Parameters option in the Config menu has been modified from Time to Percent.

High Level – programs the pulse high-level value in units of volts.

Low Level – programs the pulse low-level value in units of volts.

Delayed Pulse Mode

The Delayed Pulse mode is a special mode that delays the pulse output after a trigger is issued. To modify the pulse mode to delayed pulse, use the procedure as given above under the Pulse Modes heading.

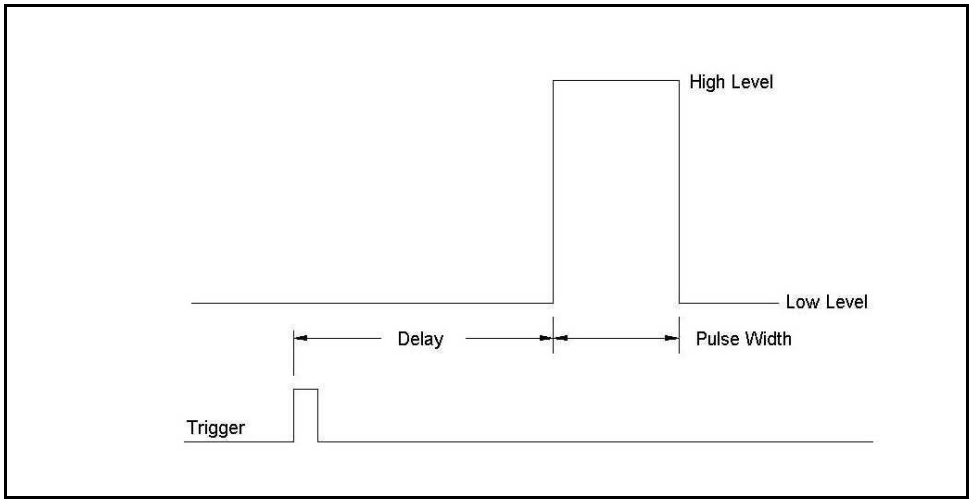

Figure 3-54, Delayed Pulse Mode

The menus that are associated with the simplest delayed pulse configuration are: Period, Width, High and Low Levels and Pulse Delay. The pulse delay is measured from the trigger edge to the first transition of the pulse leading edge, as shown in Figure 3-53.

The Pulse Delay parameter becomes available in the main pulse parameters screen only after you select the delayed pulse mode.

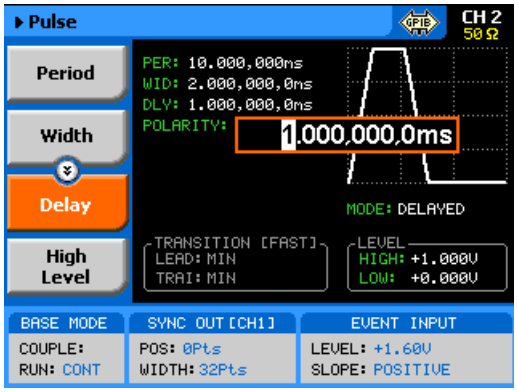

Figure 3-55, Programming the Single Pulse Delay Parameter

Double Pulse Mode

The Double Pulse mode is a special mode that doubles pulse output into a pair of two pulses at a time. In continuous run mode, the output will appear as a string of pulse pairs, separated by an interval set by the period parameter. If you place the instrument in triggered run mode, a pair of pulses is initiated with every trigger. To modify the pulse mode to double pulse, use the procedure as given above under the Pulse Modes heading.

The menus that are associated with the simplest double pulse configuration are: Period, Width, High and Low Levels and Double Pulse Delay, which sets the delay between the pairs of pulses. The double pulse delay is measured from the first pulse transition to the next pulse transition, as shown in Figure 3-55.

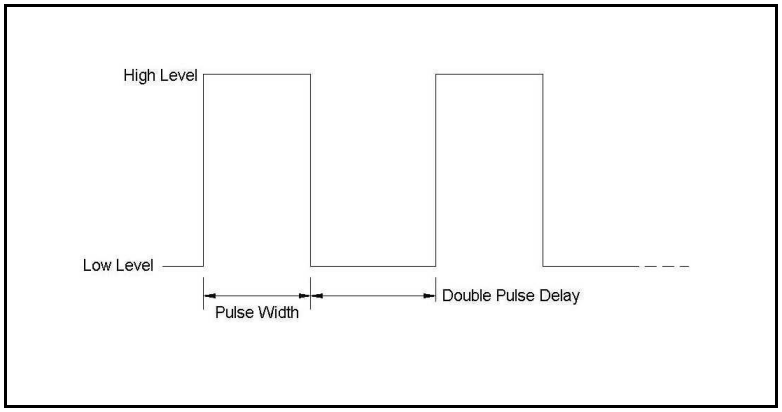

Figure 3-56, Double Pulse Mode

The double pulse delay parameter becomes available in the main pulse parameters screen, as shown in Figure 3-56, only after you select the double pulse mode. Note that the delay is programmed from pulse-end to pulse-start.

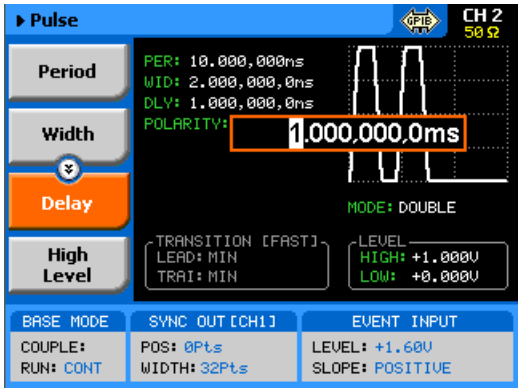

Figure 3-57, Programming the Double Pulse Delay

Programming Pulse Polarity

The pulse polarity parameter determines if the pulse is generated In Normal, Complemented, or Inverted shape. Pulse polarity can be selected in conjunction with any of the pulse modes. The various polarity options are shown in Figure 3-57.

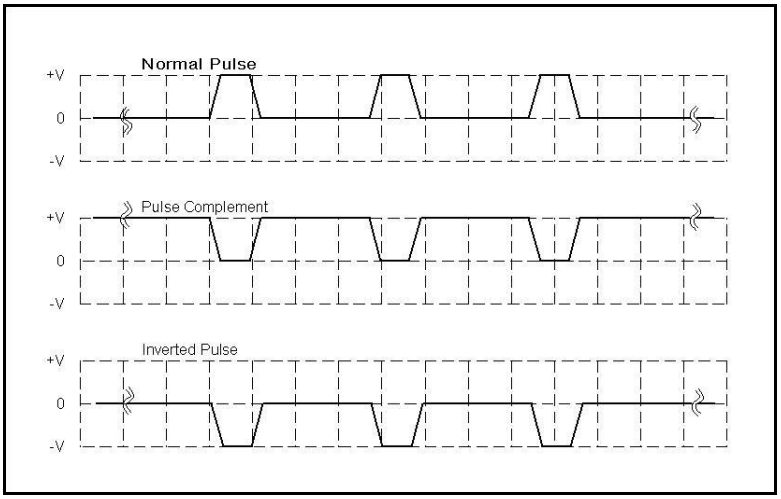

Figure 3-58, Pulse Polarity Options

As shown in this figure, the complemented shape is mirrored around the horizontal axis in a way that the high level becomes low and similarly, the low level becomes high. In complemented mode, the inversion process is symmetrical to about 50% of the value of the pulse amplitude.

In Inverted mode, the normal pulse is mirrored about the 0 V horizontal axis, positive values are converted to negative and negative values are converted to positive.

Use Figure 3-58 to access and modify the pulse polarity parameter. Note that the pulse polarity can be programmed separately for each channel and therefore, before you modify this parameter, make sure that you program the correct channel.

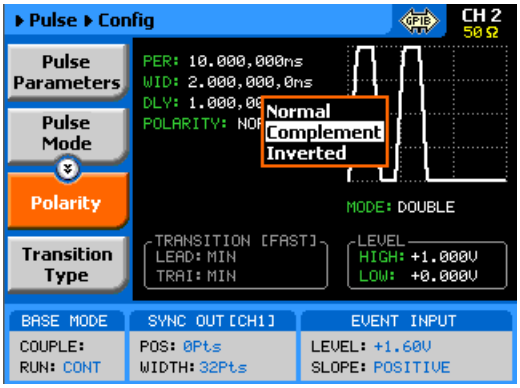

Figure 3-59, Programming the Pulse Polarity

Applying Linear Transitions

Most of the applications that use pulse generation require that the transitions from low-to-high and from high-to-low be as fast as possible. Such transitions are normally created as a by-product of the output amplifier. General-purpose amplifiers that can drive 50 Ω loads with high amplitudes are rare, and usually the products that you can buy off the shelf either have poor drive capabilities or uncontrolled aberration capability.

For a pulse generator that generates pulses with fast transitions only, the problem is simpler, because the designer can use a switching amplifier at the output amplifier stage. It takes a very different approach to design an output stage that slows the transitions of the leading and trailing edges. To this extent, the SE5082 has a unique output amplifier stage that allows full control over pulse transitions over a very high dynamic range of amplitudes and offsets, without the slightest degradation of the integrity of the signal. A comparison between pulses with fast and linear transitions is shown in Figure 3-59. As you can see, the top train has fast transitions. These are normally in the range of <1 ns and are expected to be very fast and without aberrations. The bottom pulse has linear transitions that are expected to have good linearity and slew-rate accuracy. Observe that the pulse width on the pulses that have linear transitions is measured from the first transition to high. Also note that the leading and the trailing edges can be programmed to have different transitions times.

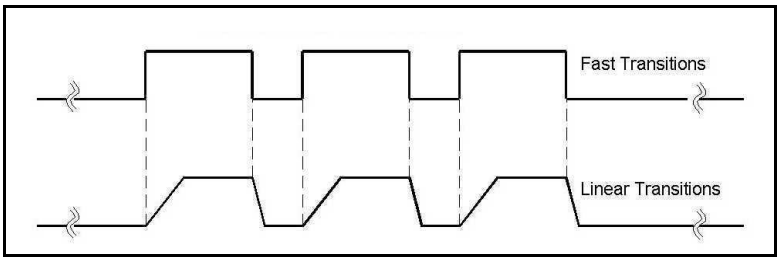

Figure 3-60, Fast and Linear Transitions, Compared

The menu that selects between fast and linear transitions is shown in Figure 3-60. The application of linear transitions affects all pulse modes. Note that you may select symmetrical transitions and in this case, programming the lead time automatically adjusts the fall time to the same value and vice versa: programming the fall time will automatically adjust the rise time value.

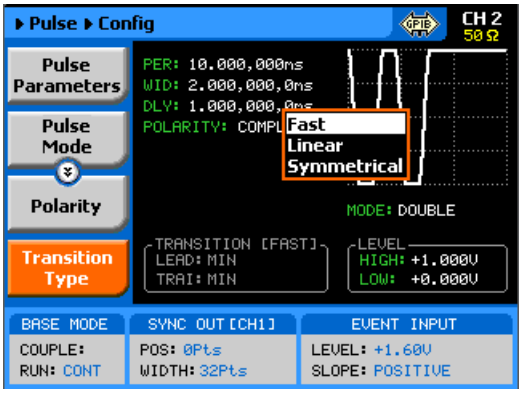

Figure 3-61, Programming the Transition Type

After you select the linear transition option, you'll probably want to program the transition times for the leading and trailing edges. There are some considerations to observe before you program the transitions. First, note the limits, as specified in Appendix A. Then you have to make sure that the transition settings do not conflict with the pulse width settings. For example, if you set the pulse width to 100 ns and the leading edge transition time to 120 ns, the instrument will not allow you to do the change. A list of setting conflicts is given later in this chapter.

Finally, you have to bear in mind that the transitions are programmed digitally, so there is no limitation on ranges, but if you select long transition cycles, there is danger that the generator will not have enough waveform memory to complete the waveform. These limitations are summarized in Appendix A and some are listed as errors in the following paragraphs.

TIP

Although not a design limitation, setting conflict may occur if the leading and trailing edges are not programmed within the same range.

After you configure the SE5082 to have linear transitions, the Leading Edge and the Trailing Edge soft buttons will be visible on the main pulse display, as shown in Figure 3-61.

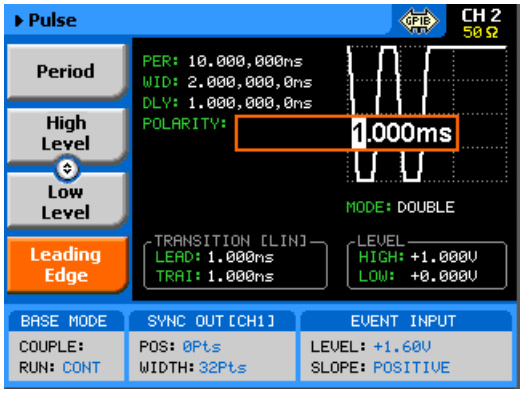

Figure 3-62, Leading Edge Programming Example

Programming the Amplitude Level Mode

Common waveform generators allow adjustment of the peak-to-peak amplitude plus offset, in cases where the required level is none symmetrical about the 0 V level. While this is normally acceptable for generating standard waveforms, pulsed waveform applications require additional flexibility in the way that the levels are set. For example, one may want the baseline to stay on 0V, regardless of the top or the low peaks settings. Other applications could require separate adjustments of each level and so on.

Using the SE5082 as your pulse source, you have four options to set up the pulse levels, as shown in Figure 3-62 and explained below:

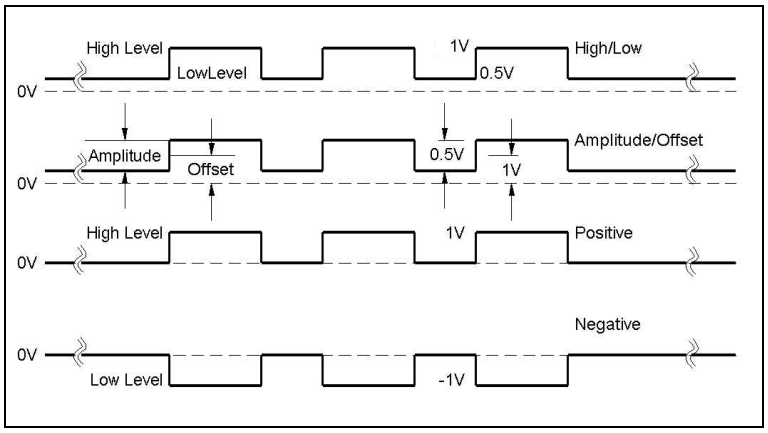

Figure 3-63, Amplitude Level Modes Options

1. ***High/Low Levels.*** This is the default option. Using this mode, you can program the high and the low levels directly, while allowing the instrument to set the internal amplitude and offset according to your selection. The maximum amplitude that can be programmed depends on the installed output module. Note that the amplitude level is specified when the pulse is applied on a 50Ω load impedance. The only way to exceed the peak-to-peak rail limitation is by increasing the load impedance, but then expect degradation of the pulse shape. The example in Figure 3-62 shows the high level set at 1 V and the low level set at 0.5 V and therefore, the resulting peak-to-peak level is 0.5 V.
2. ***Amplitude/Offset.*** This option allows separate modification of the amplitude and the offset. This option is very useful when you want to vary the vertical position of the pulse, but hold the peak-to-peak amplitude level fixed. The example in Figure 3-62 shows the amplitude set at 0.5 V and the low level set at 1 V and therefore, the resulting waveform is exactly as was programmed in the high/low level programming example.
3. ***Positive.*** This option is very useful when you want to vary the positive position of the pulse, but hold the low level fixed at the 0 V level. The example in Figure 3-62 shows the high level set at 1 V and the low level fixed at 0 V. Modification of the high level will not alter the low level position.
4. ***Negative.*** This option is very useful when you want to vary the negative position of the pulse, but hold the high level fixed at the 0 V level. The example in Figure 3-64 shows the low level set at -1 V and the high level fixed at 0 V. Modification of the low level will not alter the high level position.

The amplitude level options are selected in the configuration menu, after you select the Level Control tab, as shown in Figure 3-63.

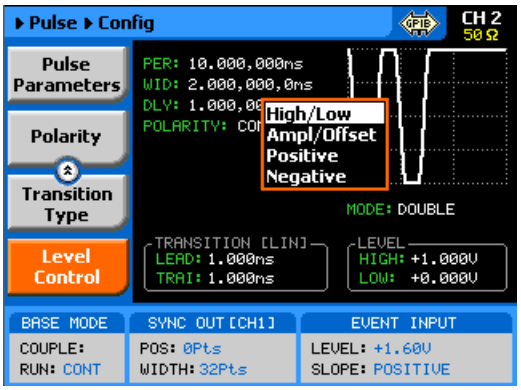

Figure 3-64, Programming the Amplitude Level Mode

Pulse Design Limitations

Keeping in mind that the pulse is created digitally with the use of memory points, one should understand that there are limitations to creating such pulses that evolve from this system. These limitations are summarized below.

1. Step increment defines resolution and period
The pulse is being created digitally, using a sample clock generator that clocks memory points. The rate of the sample clock defines the incremental resolution. Consider that you want to generate 100 us pulse rates with 1 us high time pulse and the rest of the period low. In this case, the generator can select the 100 MSa/s to 1 GSa/s clock rates, because this is enough for generating a high signal of 1 us using just 10000 to 100000 memory points. However, when you want to define much smaller pulse widths at larger rep rates, the number of points that are used for the generation increases as a function of the period. The limitation is set by the number of memory points. With the standard memory model, the incremental resolution is 1 in 32,000,000. This increases to 1 in 64,000,000 if you purchased the SE5082-1.

2. Sum of pulse parameters cannot exceed the period
While designing a pulse shape, keep in mind that the generator will automatically detect if you are trying to “mess with” the mathematics. Therefore, remember that the sum of all parameters cannot exceed the period. Always start your pulse design by assigning the required period first and only then work your way down the parameters list.

3. Pulse peaks cannot exceed the supply voltage rails.
The maximum peak-to-peak amplitude span for a given pulse design is the specified max amplitude of the installed output module and a further limitation is that the positive and negative settings cannot exceed the output amplifier rails.

The paragraphs below summarize possible settings conflicts and devise options to undo the lockup. In case you try to program a parameter that will cause a setting conflict, the instrument will

automatically detect the problem and issue an error message. In this case, the output may appear distorted and generate uncontrolled signals. Under no circumstances should you use the output when an error is displayed, even if the signal appears to look somewhat as expected, because there is no control over accuracy and signal integrity when an error message is displayed.

Settings Conflict Errors

Settings conflict errors may involve one or more parameters and there is also a chance that more than one error is embedded in the settings. For simplicity, the SE5082 displays the first error it detects. In this case, the output is updated, but a red asterisk is displayed on the message bar. This symbol will remain on the top message bar until the error is cleared. A detailed description for each of the conflicting settings is given below.

Note

The following abbreviations are used throughout the following setting conflicts descriptions:

PER – Period setting

HIL – High-level setting

LOL – Low-level setting

WID – Pulse-width setting

DEL – Delay-edge setting

TRE – Trailing-edge setting

ITRG – Period-of-internal-trigger setting

Amplitude Errors

Amplitude errors occur when attempting to program amplitude outside the specified window level. Note that this depends on which output module is installed. This error will occur in all run mode options. The generator will detect the following conditions:

HIL - LOL < Min. Amplitude
HIL + LOL > max. Amplitude
HIL > Max Window
LOL < Min Window

The minimum and maximum amplitude levels are absolute values that the SE5082 can accept. The same will occur if you reverse the high and low levels, because the instrument will sense it as a negative voltage, which is obviously less than the minimum 50 mV limit.

Corrective Actions
Evaluate the required amplitude level and change the high and low levels accordingly.

Pulse Timing Errors

Pulse timing errors occur when attempting to program the period value outside the specified limits or when there is a setting conflict between the period and the leading edge transition values. The generator will detect the following conditions:

PER > 1.6 s
PER < 800 ps
LEE > WID

While the first two conditions are easy to grasp, as they are simply a matter of how the pulse parameters are specified, the last condition evolves from the way that the SE5082 interprets the pulse width parameter. If you look at earlier sections of the pulse generator description, you'll notice that the pulse width includes the leading edge transition time and therefore, if you fail to take this into consideration, the generator will issue an error flag.

Corrective Actions
1. Make sure the period limits are observed
2. Reduce the leading-edge value

Pulse Linear Transition Errors

Pulse linear transition errors occur when attempting to program the leading and trailing edge values outside the specified limits. The generator will detect the following conditions:

LEE/TRE < 1 ns
LEE/TRE > 100 ms
WID+TRE > 1.6 s

While the first two conditions are easy to grasp, as they are simply a matter of how the pulse parameters are specified, the last condition evolves from the way that the SE5082 interprets the pulse width parameter. If you look at earlier sections of the pulse generator description, you'll notice that the pulse width includes the leading edge transition time and therefore, if you fail to take this into consideration, the generator will issue an error flag.

Corrective Actions
1. Make sure the leading- and trailing-edge limits are observed
2. Reduce the leading-edge value
3. Reduce the trailing-edge value

Pulse Delay Errors

Pulse delay errors occur when attempting to program pulse or double pulse delay values outside the specified limits. The generator will detect the following conditions:

DEL < 200 ps
DEL > 1.6-0.2e-9 s
DEL+WID+TRE > 1.6 s (single pulse mode)

DEL+2*WID+2*TRE > 1.6s (Double pulse mode)

While the first two conditions are easy to grasp, as they are simply a matter of how the pulse parameters are specified, the last condition evolves from the way that the SE5082 interprets the pulse width parameter. If you look at earlier sections of the pulse generator description, you'll notice that the pulse width includes the leading-edge transition time and therefore, if you fail to take this into consideration, the generator will issue an error flag.

Corrective Actions

1. Make sure the delay limits are observed
2. Reduce the delay value
3. Reduce the leading-edge value
4. Reduce the trailing-edge value

General Pulse Errors

Bearing in mind that the pulse is generated mathematically and replayed as an arbitrary waveform, it is very easy to get into tight corners that do not let the generator compute the waveform efficiently. This may come from the following conditions:

Required waveform resolution is < 200 ps
Required memory length > 32,000,000
The computed SCLK > 5 GSa/s
System memory allocation limitations

The general pulse errors are hard to define. However, if reasonable parameters range and ratio-to-period are kept, the SE5082 should have no problem computing and generating the required pulse shape.

Pulse Patterns

The SE5082 pattern functionality is implemented as an extension of the Pulse Mode. In addition to RZ pulses, it allows the generation of pattern sequences that are using NRZ formatting with adjustable transition. The pattern generation itself allows the definition of pattern sequences with 2, 3, 4 or 5 different level settings, which allows the emulation of electrical idle sequences as being required in several serial data protocols.

There are two patterns that can be generated by the SE5082: PRBS (Pseudo Random Bit Sequence) and user-composed pulse patterns. The patterns are created digitally. However, they closely simulate an analog pulse generator, so pulse pattern parameters are programmed just as they would be programmed on a dedicated pattern generator instrument. Just keep in mind that since this is a digital instrument, there are some limitations to the pattern design that evolve from the fact that the best resolution is one sample clock interval. Also, remember that the patterns are created digitally in the arbitrary memory and therefore, their smallest incremental step has a maximum value limitation, depending on the memory length, as specified in Appendix A.

Information how to create and generate PRBS patterns and user-composed pulse patterns is given below.

Generating PRBS Patterns

Pseudo Random Bit Sequences (PRBS) are commonly used for bit error rate measurements. PRBS sequences are extremely beneficial for many applications that generate sequences in a transmitter side and later evaluated for their error contents in the receiver side.

When using PRBS, certain number of bits are required for synchronization (preamble) and only after bit synchronization is achieved, received sequence is compared with the reference and errors are detected.

PRBS is a sequence of nearly random distributed 1's and 0's which is repeated after a defined number of bits. It is usually generated using a linear feedback shift register (LFSR). The LFSR has two basic parameters determining properties of the generated sequence: length of shift register and feedback configuration. The length of the shift register determines maximum achievable length (period) of non-repetitive PRBS sequence according to the following equation:

$$L_{PRBS} = 2^n - 1$$

Where n is the length of LFSR register (number of bits).

The maximum length of PRBS is called M-sequences and is commonly used in many applications like spread spectrum systems, scramblers, bit error rate testers etc.

The maximum length of generated PRBS for certain length of the shift register can be achieved by proper configuration of the feedback. There are basically two possible realizations of LFSR – Fibonacci (many-to-one) and Galois (one-to-many). Both versions can be based either on XOR or XNOR gates using various number and combination of feedback taps – outputs of particular registers of the shift register. By changing the feedback configuration (number of taps and their position) it is possible to find more different M sequences for certain length of the shift register.

To access the PRBS generator menus, press on the Pulse button in the Function group, scroll down the menus and push the Pattern button. The display will be modified to show pattern parameters, as shown in Figure 3-64.

Note

The Pattern shape on the SE5082 LCD display is an icon only. The actual waveform at the output terminals may look entirely different.

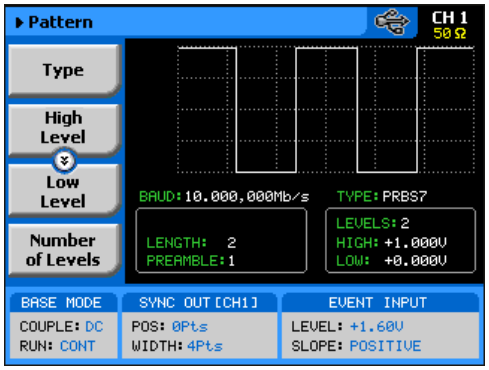

Figure 3-65, Pattern Menus

The technique of changing parameter values is exactly the same as you are using to modify standard waveform parameters. Simply press the soft key that is associated with the parameter, then punch in the numbers using the numeric keyboard and complete the modification by assigning a suffix and pressing the Enter button.

When PRBS mode option has been selected, the main Pattern tabs, as shown in Figure 3-64, will allow access to the following parameters:

Type – programs the PRBS type sequence that the instrument calculates and generates at the output connector. There are 6 PRBS types: PRBS 2^7, PRBS 2^9, PRBS 2^{11}, PRBS 2^{15}, PRBS 2^{23}, PRBS 2^{31} and an additional user-defined PRBS type. In continuous run mode, the sequence is repeated continuously but in triggered mode, the sequence is repeated until the loop number has been reached. Information on the user defined PRBS is given in the programming chapter; note however that user PRBS cannot be defined from the front panel menus.

High Level – programs the pattern high-level value in units of volts.

Low Level – programs the pattern low-level value in units of volts.

Number of Levels – programs the level configuration of the PRBS pattern. The effect of levels 2 to 5 are demonstrated in Figures 3-65 to 3-76. The high and low levels provide references for the level setting. In all cases, the maximum and minimum amplitude levels are programmed in the high and low level menus.

Baud – programs the pulse pattern sample sequence in in units of bits per second.

Length – programs the pulse length of the PRBS pattern in units of bits.

Preamble – programs a synchronization interval in units of bits. The generator will repeat the preamble length continuously until a trigger signal is received. After trigger, the generator will jump to the PRBS patterns that follow the preamble and will repeat the sequences continuously without branching back to the preamble. If predetermined number of PRBS loops is required, modify the SE5082 run mode setting to triggered and you'll be allowed access to the loop counter.

Loop – This menu tab appears only when selecting the trigger run mode. The loop number determines how many times the PRBS pattern repeats itself following a trigger command. The loop number does not affect the generator in continuous run mode.

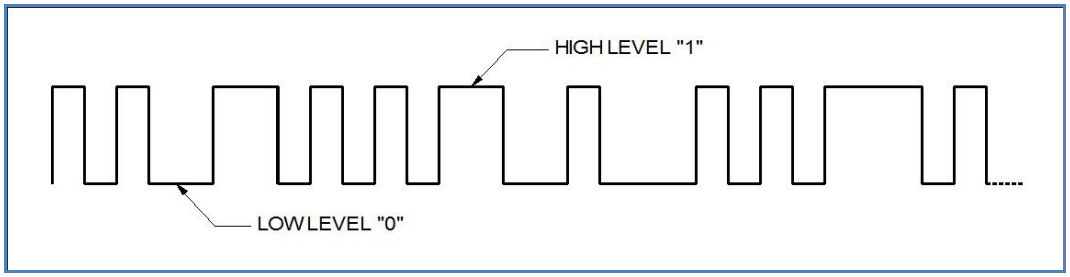

Figure 3-66, PRBS 2 Level Pattern Example

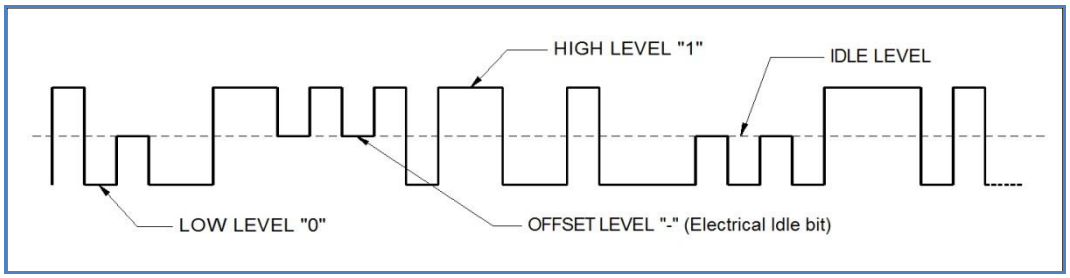

Figure 3-67, PRBS 3 Level Pattern Example

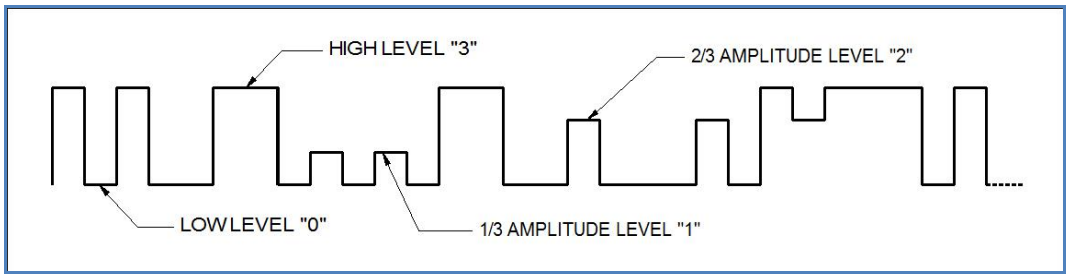

Figure 3-68, PRBS 4 Level Pattern Example

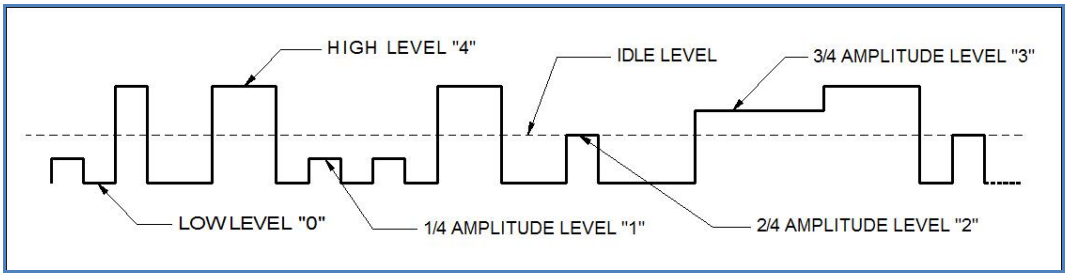

Figure 3-69, PRBS 5 Level Pattern Example

Generating User Composed Patterns

Composed pattern is a rubber-band method of generating patterns that have mixed transition times. One can generate a pattern that has simple and fast transitions from high level to low level or, generate a very complex pattern that has mixture of fast and linear transitions plus control the amplitude level for each transition. From the front panel, pulses are created by stretching a line from last point to the next. It is of course much simpler when programming such a pulse using the available commands for composed pulse waveforms, as described in the programming chapter of this manual.

The SE5082 offers two methods of creating user-defined pulse patterns, the first is pulse patterns that transition from high level to low level using the fastest transition time that the generator offers. This type is nomenclature as “Fast”. The second option is the “Linear” pulse pattern that allows slower transitions from high to low amplitude levels. In both cases, the high and low amplitude levels can be designed as unique values for each transition as will be explained in the following paragraphs.

Do the following to access the pulse composer menus:

1. Push the Pulse button in the Function group keys. The pulse dialog box, as shown in Figure 3-69 will be displayed.

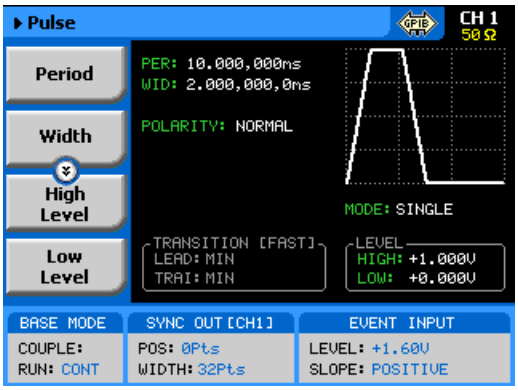

Figure 3-70, User-composed Pulse Patterns Menus

2. Using the down key, scroll down with the menus to access the Pulse Composer menu tab, as shown in Figure 3-70.
3. Press the Pulse Composer soft key and observe the Pulse Composer editing fields as demonstrated in Figure 3-71.

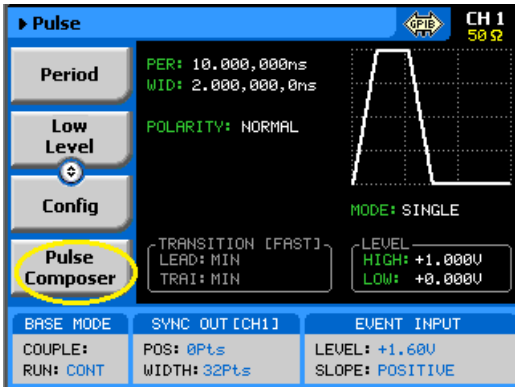

Figure 3-71, Access the Pulse Composer Menus

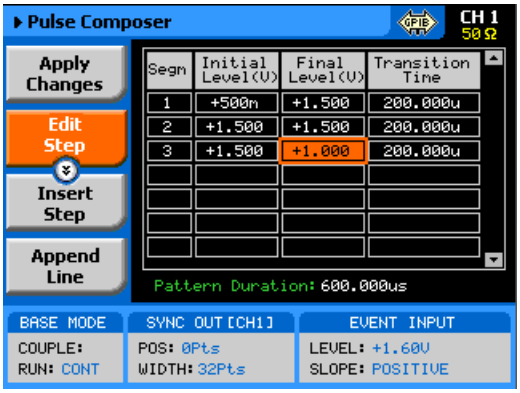

Figure 3-72, Pulse Composer Menus Example

The following parameters can be accessed and modified from the pulse editor menus, Step, Level and Segment Duration. The Pattern Duration is automatically cumulated while building the pulse pattern and is displayed at the bottom of the table. As explained above, there are two types of transitions that the pulse composer can build and generate: Fast and Linear. Description how to use the composer to build these two pulse patterns is given below.

Always, as the first step before you start building your patterns, make sure that you are building the correct configuration, fast or linear. Scroll down to the Transition Type, as shown in Figure 3-72 and select the proper pulse composer configuration setting. The default setting is “Fast” so you need not do any action if this is what you intend to create.

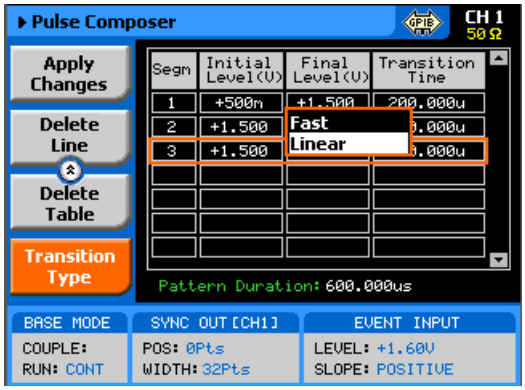

Figure 3-73, Selecting the Proper Pulse Composer Configuration

Building “Fast” Transitions Patterns

Patterns that have “fast” transitions are those that use the fastest transition times (rise and fall times) that the generator can deliver. Transition times are specified in Appendix A, but are usually around 500 ps. It is recommended that when building a pulse pattern that those transition times are taken into account because the duration of the segment, as will be explained later, includes the transitions.

Figure 3-72 demonstrates a simple fast transition pulse pattern. Notice that the pattern is built from segments, each having level and duration. Levels are specified in volts and duration is specified in seconds. Each level can have a unique amplitude level and duration as long as these reside within the range that is specified in Appendix A. Also note that the table entries in Figure 3-71 were not used to build the pulse pattern as shown in Figure 3-72.

Pattern pulses are computed and generated in the arbitrary memory and therefore, refrain from using extreme values. For example, a single 2 ns pulse and a period of 1000 seconds will present a huge problem for the algorithm because the composer will attempt to use

the smallest sample clock increments, say 1 GSa/s and will have to use the same increments for the rest of the duration. When running out of memory space, the generator will automatically revert to the sequence mode and hence will take lots of memory space and lots of sequence resource.

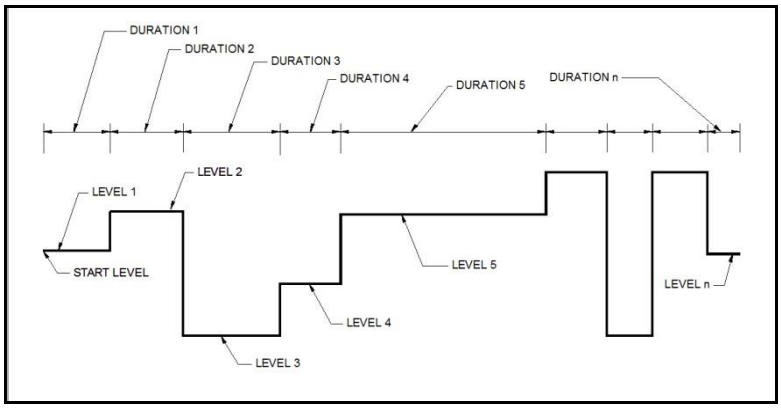

Figure 3-74, "Fast" Transition Pulse Example

To access the pulse composer tab menus, simply press one of the tabs and modify or select the parameters that are associated with this specific tab. When the pulse composer has been selected, the composer tabs, as shown in Figure 3-71, will allow access to the following parameters:

Apply Changes – this button must be pressed to complete the process of building the pulse pattern table. Modifications made to the composer table will not be computed unless this tab has been depressed.

Edit Step – if you already entered steps but made an error in one of its parameters, you can access this specific step instantly and modify the required value. Select the step to edit, press the enter button to highlight the parameter, modify the value and press enter again to lock the new parameter value. While in the table, you can scroll up or down with the cursor keys to access other steps. Remember to press the Apply Changes soft key button to complete the process.

Insert Step – lets you add a pulse segment above or below the highlighted line. If you want to add lines at the end of the table, use the Append Line soft key instead. Remember to press the Apply Changes soft key button to complete the process.

Append Line – will add a line at the end of the table and will automatically highlight this line. Proceed to modify the parameters as explained in the Edit Step paragraph above. Remember to press the Apply Changes soft key button to complete the process.

Delete Line – in case you just want to remove pattern segments, use this soft key to select and remove the wanted line item.

Remember to press the Apply Changes soft key button to complete the process.

Delete Table – removes all table entries and clears the fields for new entries.

Building "Linear" Transitions Patterns

Patterns that have "linear" transitions are those that use the slow transition times (rise and fall times). Figure 3-72 demonstrates a simple linear transition pulse pattern. The pattern is built from segments, each having an initial and final level and a transition time. Levels are specified in volts and transition times are specified in seconds. Each segment can have a unique amplitude level and transition time as long as these reside within the range that is specified in Appendix A. Note that the table entries in Figure 3-75 were not used to build the pulse pattern as shown in Figure 3-74.

When you start building a pattern table, only the first segment allows access to the Initial level. The final level of step 1 will automatically become the start level of the second segment and will not be accessible when you edit the second segment and so on for the rest of the segments. Therefore, if you look at the composer table in Figure 3-75, the Final Level value is the same as the Initial Level of the following segment transition.

Pattern pulses are computed and generated in the arbitrary memory and therefore, refrain from using extreme values. For example, a single 50 ns pulse that has transitions of 10 ns and a period of 1000 seconds will present a huge problem for the algorithm because the composer will attempt to use the smallest sample clock increments, say 1 nSa/s and will have to use the same increments for the rest of the duration. When running out of memory space, the generator will automatically revert to the sequence mode and hence will take lots of memory space and lots of sequence resource. Note that the default option for the pulse composer is "Fast" so before you start adding values to the pulse composer table, make sure that the selected option is "Linear", as explained earlier in this section.

To select the linear transition option, scroll down to the Transition Type and select the "Linear" option, as shown in Figure 3-74. The linear pulse composer table is shown in Figure 3-77.

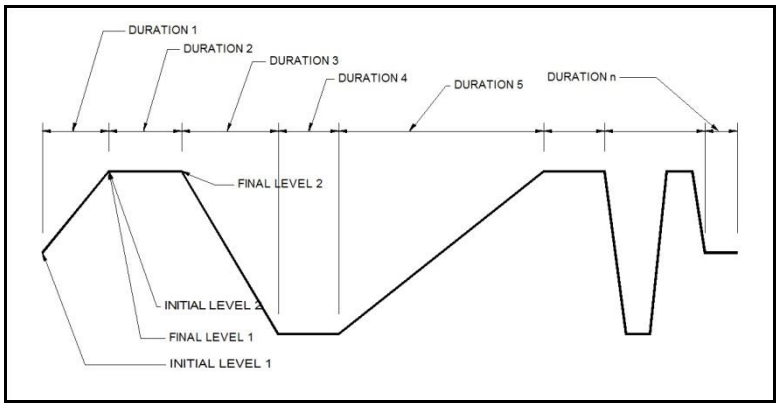

Figure 3-75, "Linear" Transition Pulse Example

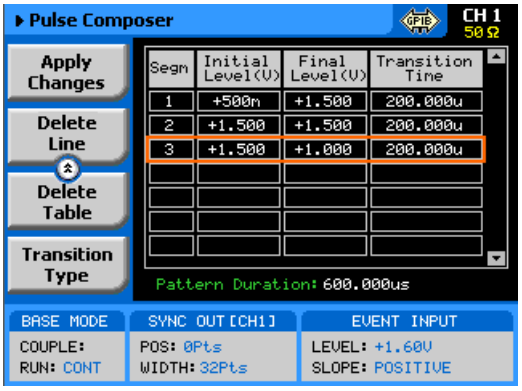

Figure 3-76, Linear Transition Pulse Composer Example

To access the pulse composer tab menus, simply press one of the tabs and modify or select the parameters that are associated with this specific tab. When the pulse composer has been selected, the composer tabs, as shown in Figure 3-75, will allow access to the following parameters:

Apply Changes – this button must be pressed to complete the process of building the pulse pattern table. Modifications made to the composer table will not be computed unless this tab has been depressed.

Edit Step – if you already entered steps but made an error in one of its parameters, you can access this specific step instantly and modify the required value. Select the step to edit, press the enter button to highlight the parameter, modify the value and press enter again to lock the new parameter value. Note that the only values that can be modified are the Transition Time and the Final Level because the Initial level is updated automatically with the amplitude level of the previous segment. While in the table, you can scroll up or down with the cursor keys to access other steps. Remember to

press the Apply Changes soft key button to complete the process.

Insert Step – lets you add a pulse segment above or below the highlighted line. If you want to add lines at the end of the table, use the Append Line soft key instead. Remember to press the Apply Changes soft key button to complete the process.

Append Line – will add a line at the end of the table and will automatically highlight this line. Proceed to modify the parameters as explained in the Edit Step paragraph above. Remember to press the Apply Changes soft key button to complete the process.

Delete Line – in case you just want to remove pattern segments, use this soft key to select and remove the wanted line item. Remember to press the Apply Changes soft key button to complete the process.

Delete Table – removes all table entries and clears the fields for new entries.

Using the Markers

There are two marker outputs available for each channel. The markers are output differentially through rear panel SMB connectors. The purpose of these markers is to be used as auxiliary outputs that are fully synchronous with the output waveforms. Although the markers are part of the waveform data, when used, they do not reduce the number of DAC bits that are available for the arbitrary function.

The markers can be placed anywhere along the waveforms and their width adjusted, sort of adjustable pulse width. Low- and high-voltage controls are also available separately for each marker output, and since the markers are part of the waveform data, jitter between the markers and the output waveform is minimized.

Bench operation of the SE5082 allows programming of a single marker transition per channel. However, if you are controlling the product from remote, you can download multiple markers as part of the waveform data and in fact, use the marker terminals to output strings of pulse data. Using the full 5 GSa/s speed of the sample clock, the markers can clock data out at the rate of over 2.5 Gbit/s.

Note that there is a constant system delay between the output start and the marker outputs, but this can be adjusted to have the same phase using the delay and offset features of the output waveform. The system delay value is given in Appendix A.

The display as shown in Figure 3-76 will provide access to the marker parameters as soon as you hit the Markers button in the Control group on the left of the instrument. The screen is divided into two sections: Marker 1 and Marker 2. The Markers screen and parameters are described below.

State – toggles both marker outputs ON and OFF. Programming the markers from the front panel always generates markers from the start edge of the waveform.

CH1

Press the CH1 soft key button to access parameters that controls channel 1 markers. These are: State, Delay, Pos, Width, High and Low. These parameters are explained as follows:

Delay – programs the marker delay value. The delay is measured from the waveform start position and is programmable with resolution of 10 ps from 0.00 ns to 3.00 ns.

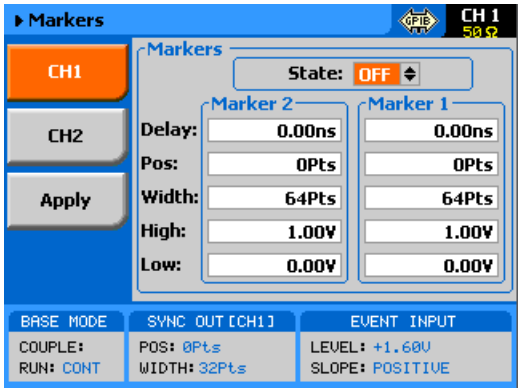

Figure 3-77, Marker Menus

Pos – programs the position of the marker start in reference to the waveform start point. The position is programmed in units of waveform points (or sample clock periods) from 0 to the full length of the waveform, in increments of 4 waveform points. The marker-position value does not affect the marker-delay value and vice versa.

Width – programs the width of the marker in units of waveform points. Minimum width is 4 points and maximum is the full wavelength of the waveform. The width is programmed in increments of 4 points. The marker delay and position have no effect on the marker, except the sum of the marker-position value and marker-width value cannot exceed the length of the waveform.

High – programs the high level of the marker pulse. The level is valid when terminated into a proper load. The high level is programmable from 0.5 V to 1.25 V.

Low – programs the low level of the marker pulse. The level is valid when terminated into a proper load. The low level is programmable from 0 V to 0.8 V.

CH2

Press the CH2 soft key button to access parameters that control channel 2 markers. These are the same as were described above, for channel 1.

Apply

As was mentioned above, the marker output data is part of the arbitrary waveform data and therefore, modifying the marker parameters do not change the marker output, until the Apply soft key button has been pressed. Every timing change of the marker output initiates an internal cycle that reads and clears the current position, and then updates the waveform memory with the new marker setting. The process is very quick if small changes are made to the markers, but may be longer if many waveform points need to be read and updated.

Using Store/Recall

The store and recall functions allow you to save important waveforms and front panel settings and recall them later, when they are needed for a specific application. If you are using external utilities and control the instrument remotely, then it will be best to store all instrument settings and waveforms on the computer's media. However, if local operation is desired, you may still store settings, but bear in mind that the internal memory is limited on what can be saved. Instrument settings and waveforms can also be stored on a disk-on-key that is connected to a front-panel host USB connector (type 1).

The front-panel store/recall function allows access to 9 memory cells. One or more cells can be used for this operation, but you need to observe the limitations. The internal memory has a capacity to store a total of 4G of waveform points. However, storage of waveforms larger than 1 million points is rather slow and therefore, if you need to store longer waveforms, these should be stored on the hard-drive of your computer.

Press the Store/Recall button in the Control group on the right side of the instrument. This will cause the display to show the Store/Recall panel, as shown in Figure 3-77. There are four soft-key buttons in this display: Store, Recall, List and Update. Information on how to use the keys to store and recall setups on different media options is given below.

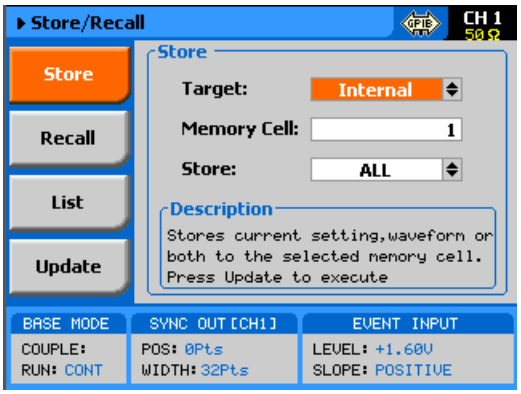

Figure 3-78, Store/Recall Menus

Storing Instrument Setups and Waveforms

The Store display is shown in Figure 3-78. This is the first display you enter after pressing the Store/Recall button in the Control group on the right side of the SE5082. The Store display has three fields that you can program: Target, Memory Cell and Store. The operation of these is described below. Note that the selected memory cell will be updated with the information that is shown on the display only after you hit the Update soft key button.

Target – will provide the path to the selected memory target. There are two options: Internal and USB. When you select the Internal option, setups and waveform data will be stored on an internal flash memory. The USB option will route the setups and waveform data to the front panel USB host connector.

Memory Cell – use this field to select the active memory cell. There are 9 memory cells that you may access and each of these cells can be used to store different setups and different waveforms.

Store – use this field to select what you want to store. You may select between Setup, Waveform and All. Setup stores the current front panel settings, including parameters, as well as operating and output modes. Waveform will store just waveform segments and All will store both the front panel setting, sequence tables and all waveform segments.

Figure 3-79, Store Menus

Regardless of what you select for each field, the operation will be completed only after you press the Update Soft Key button.

Note that storing waveforms larger than 1 M points to either the internal flash memory or the external USB is rather slow and therefore, expect long delays while storing large waveform banks when using one of these options. It is highly recommended that waveforms above 1 M points be stowed away on a faster media such as a computer hard drive and downloaded to the instrument as required for the test.

Recalling Instrument Setups and Waveforms

The Recall display is shown in Figure 3-79. Just like the Store display, the Recall display has three fields that you can program: Source, Memory Cell and Store. The operation of these is described below. Note that the front panel will be updated with the information that is shown on the display only after you hit the Update soft key button.

Source – will provide the path to the selected source. There are two options: Internal and USB stick. When you select the Internal option, setups and waveform data will be recalled from an internal flash memory. The USB option will route the setups and waveform data from the front-panel USB host- connector. There you may connect a USB stick that can store very large waveforms.

Memory Cell – use this field to select the active memory cell. There are 9 memory cells that you may access and each of these cells can be used to recall different setup and different waveforms.

Recall –this field shows what you are about to recall. The field is grayed-out and does not allow access to the selected option since the contents of the memory cell is programmed in the store menu.

Regardless of what you select for each field, the operation will be completed only after you press the Update Soft Key button.

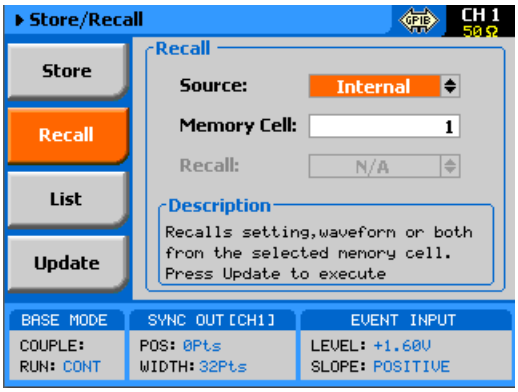

Figure 3-80, Recall Menus

Note that recalling waveforms larger than 1 M points from either the internal flash memory or the external USB is rather slow and therefore, expect long delays while recalling large waveform banks when using one of these options. It is highly recommended that waveforms above 1 M points be stowed away on a faster media such as a computer hard drive and downloaded to the instrument as required for the test.

Listing Storage Memory Contents

If you are not sure what is stored in each of the memory cells, you can press the List Soft Key button. The list has no function other than showing what is stored in the memory cells. Press the List Soft Key button and the panel as shown in Figure 3-80 will be displayed. Note that the List display provides access to the storage media. You may select between USB and Internal and the panel will show the parameters that are stored on this specific media.

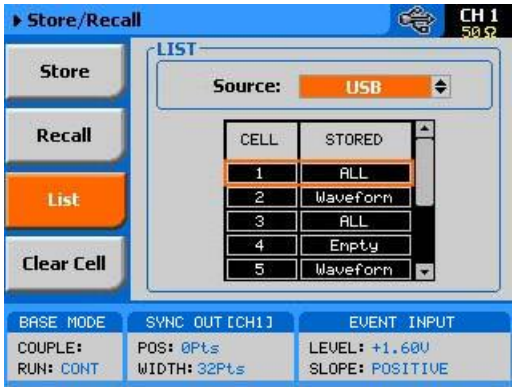

Figure 3-81, Stored Memory Cells Listed

Note that the dialog box can display the contents of five cells only. To see the rest, use the dial or the cursor up and down buttons to navigate to the required cell. To modify the Source media, press the cursor up button till the Source field is selected, press the enter key and again use the cursor up or down key to select the required storage media; then, press Enter again to select and the display will be updated with the contents of the source.

From the List dialog box you may also clear memory cells that you want to discard of. Press the Clear Cell soft key button, using the cursor up and down, or the dial, select the cell number you want to clear and press Enter. In the example above, Cell number 4 has been cleared and shows “Empty” in the stored options.

Store/Recall Considerations

The SE5082 is using Linux as its operating system. Although the displays and menus have the feeling of Windows operating system, some of the privileges that are associated with Windows OS are not available in the SE5082. For example, do not expect the same behavior as plugging a stick to a Windows based operating system; There are no messages and no confirmation when a USB has been mounted or removed.

The following are some considerations that you should bear in mind when using the USB stick to store waveforms and setups out of the SE5082.

When plugging a USB stick to the SE5082 allow between 5 to 10 seconds for the operating system to mount the device. An attempt to access the stick before it has been mounted generates an error.

When downloading setups and waveforms on a USB stick, the setups and waveforms can be used reliably on another SE5082 instrument only if the firmware revision is the same. The reason is because an instrument image is never the same across firmware versions.

Waveform data is stored in a working memory that is extremely fast because it feeds the DAC however, in contrary, the flash memory that is used for storing waveforms is much slower and therefore, moving large chunks of waveforms from the working memory to the flash memory is a slow process because the data is transferred one word at a time. For this reason, refrain from storing or recalling large waveforms regardless if you use the internal memory or an external USB stick; Keeping your large waveforms on a hard drive is a much faster method to recall waveforms to the working memory.

It is possible to copy files that were stored on a USB stick to a host computer however, before copying files from a computer to another USB stick make sure that the files have names like setup1 and wave1 with no added characters or extensions. Sometimes, Linux adds strange characters to file names but these can be renamed in Windows environment to remove the extra characters.

When powered down, the SE5082 can be programmed to power on with its last setup or with all parameters set to the factory defaults. The Power ON State, as shown in Figure 3-1 provides access to this selection. When the Setup option is selected, just before the instrument turns off, it performs an automatic storage to its internal flash memory of its last state, along with the entire bank of waveforms and sequence tables. If you downloaded to the SE5082 short waveform segments that have a combined length of less than 1 M should not be a problem however, above 1 M, the storage process may take long. It is therefore recommended that you avoid using the Setup option in the System menu.

Store/Recall Messages

Some messages may appear when you perform store and recall operations on the SE5082. These messages are described below. The following messages will appear after performing a successful store or recall operations using the internal flash memory:

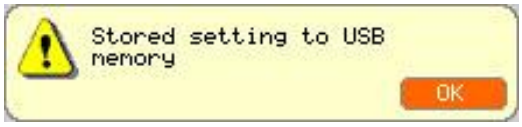

Figure 3-82, Stored setting to USB memory

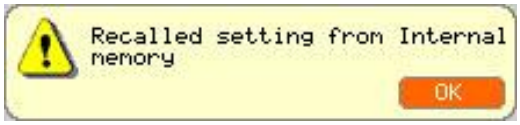

Figure 3-83, Recalled setting from USB memory

There are also some USB error messages that may appear in case an illegal operation has been performed. A general error such as unrecognized file structure or file contents will generate the following error:

Figure 3-84, Unrecognized file structure or file contents

The following error message will display in the event that one attempts to use the external USB stick but has not yet plugged it to the instrument or the device is not yet mounted on the system,

Figure 3-85, Unplugged USB error

And finally, if you pull the USB out of its socket before the store or recall operation has been fully executed, expect the following message to appear. In most case, recovery from this error is possible only after recycling the SE5082 power.

Figure 3-86, Missing USB memory stick error

This page was intentionally left blank

Chapter 4
Programming Reference

Title **Page**

What's in This Chapter

This Chapter lists and describes the set of SCPI-compatible (Standard Commands for Programmable Instruments) remote commands used to operate the Tabor SE5082**.** To provide familiar formatting for users who have previously used the SCPI reference documentation, the command descriptions are dealt with in a similar manner. In particular, each sub-system's documentation starts with a short description, followed by a table showing the complete set of commands in the sub-system. In conclusion, the effects of individual keywords and parameters are described. The complete listing of all commands used for programming the SE5082 is given in Table 4-1.

Introduction to SCPI

Commands to program the instrument over the GPIB are defined by the SCPI 1993.0 standard. The SCPI standard classifies a common language protocol. It goes one step further than IEEE-STD-488.2 and defines a standard set of commands to control every programmable aspect of the instrument. It also defines the format of command parameters and the format of values returned by the instrument.

SCPI is an ASCII-based instrument command language designed for test and measurement instruments. SCPI commands are based on a hierarchical structure, known as a tree system. In this system, associated commands are grouped together under a common node or root, consequently forming subsystems.

Part of the INITiate subsystem is shown below to illustrate the tree system:

```
:INITiate
    :CONTinuous
        :ENABle SELF|ARMed
```

INITiate is the root keyword of the command; CONTinuous is a second level keyword. ENABle is third level keyword. A colon (:) separates a command keyword from a lower level keyword.

Command Format

The format used to show commands in this manual is shown below:

```
FREQuency {<frequency>|MINimum|MAXimum}
```

The command syntax shows most commands (and some parameters) as a mixture of upper and lowercase letters. The uppercase letters indicate the abbreviated spelling for the command. For shorter program lines, send the abbreviated form. For better program readability, use the long form.

For example, in the above syntax statement, FREQ and FREQUENCY are both acceptable forms. Use upper or lowercase letters. Therefore, FREQ, FREQUENCY, freq, and Freq are all acceptable. Other forms such as FRE and FREQUEN will generate an error.

The above syntax statement shows the frequency parameter enclosed in triangular brackets. The brackets are not sent with the command string. A value for the frequency parameter (such as "FREQ 50e+6") must be specified.

Some parameters are enclosed in square brackets ([ ]). The brackets indicate that the parameter is optional and can be omitted. The brackets are not sent with the command string.

Command Separator

A colon (:) is used to separate a command keyword from a lower level keyword as shown below:

```
SOUR:FUNC:SHAP SIN
```

A semicolon (;) is used to separate commands within the same subsystem, and can also minimize typing. For example, sending the following command string:

```
TRIG:FILT:HPASS 1;WIDT 1
```

is the same as sending the following two commands:

```
TRIG:FILT:HPASS 1
TRIG:FILT:HPASS:WIDT 1
```

Use the colon and semicolon to link commands from different subsystems. For example, in the following command string, an error is generated if both the colon and the semicolon are not used.

```
OUTP:STATE ON;:TRIG:BURS ON
```

The MIN and MAX Parameters

Substitute MINimum or MAXimum in place of a parameter for some commands. For example, consider the following command:

```
FREQuency {<frequency>|MINimum|MAXimum}
```

Instead of selecting a specific frequency, substitute MIN to set the frequency to its minimum value or MAX to set the frequency to its maximum value.

Querying Parameter Setting

Query the current value of most parameters by adding a question mark (?) to the command. For example, the following command sets the output function to square:

```
SOUR:FUNC:SHAP SQR
```

Query the output function by executing:

```
SOUR:FUNC:SHAP?
```

Query Response Format

The response to a query depends on the format of the command. In general, a response to a query contains current values or settings of the generator. Commands that set values can be queried for their current value. Commands that set modes of operation can be queried for their current mode. IEEE-STD-488.2 common queries generate responses, which are common to all IEEE-STD-488.2 compatible instruments.

SCPI Command Terminator

A command string sent to the function generator must be terminated with a <new line = '\n'> character. Command string termination always resets the current SCPI command path to the root level.

IEEE-STD-488.2 Common Commands

The IEEE-STD-488.2 standard defines a set of common commands that perform functions like reset, trigger and status operations. Common commands begin with an asterisk (*), are four to five characters in length, and may include one or more parameters. The command keyword is separated from the first parameter by a blank space. Use a semicolon (;) to separate multiple commands as shown below:

```
*RST; *STB?; *IDN?
```

SCPI Parameter Type

The SCPI language defines four different parameter types to be used in program messages and response messages: numeric, discrete, Boolean, binary block.

Numeric Parameters

Commands that require numeric parameters will accept all commonly used decimal representations of numbers including optional signs, decimal points, and scientific notation. Special values for numeric parameters like MINimum and MAXimum are also accepted.

Engineering units using numeric parameters (e.g., MHz or kHz) can also be sent. If only specific numeric values are accepted, the waveformgenerator will ignore values which are not allowed and will generate an error message. The following command is an example of a command that uses a numeric parameter:

```
VOLT:AMPL <amplitude>
```

Discrete Parameters

Discrete parameters are used to program settings that have a limited number of values (i.e., FIXed, USER and SEQuence). They have short and long form command keywords. Upper and lowercase letters can be mixed. Query responses always return the short form in all uppercase letters. The following command uses discrete parameters:

```
SOUR:FUNC:MODE {FIXed | USER | SEQuence}
```

Boolean Parameters

Boolean parameters represent a single binary condition that is either true or false. The generator accepts "OFF" or "0" for a false condition. The generator accepts "ON" or "1" for a true condition. The instrument always returns "0" or "1" when a boolean setting is queried. The following command uses a boolean parameter:

```
OUTP:FILT { OFF | ON }
```

The same command can also be written as follows:

```
OUTP:FILT {0 | 1 }
```

Binary Block Parameters

Binary block parameters are used for transferring data blocks to the generator, for example, waveforms, segment table, sequence table etc. The binary block parameter format is

```
#<header><binary data>
```

Where the header, holds the data size, followed by the data itself. For example, the following command uses the binary block parameter `#42048<binary data>` to transfer a 1024 points waveform to the generator

```
TRAC:DATA#42048<binary_block>
```

Information on commands using binary blocks is given later in this chapter.

SCPI Syntax and Styles

Where possible, the syntax and styles used in this section follow those defined by the SCPI consortium. The commands on the following pages are broken into three columns; the Keyword, the Parameter Form, Default and HS command equivalent.

The Keyword column provides the name of the command. The actual command consists of one or more keywords, since SCPI are based on a hierarchical structure, also known as the tree system. Square brackets ([ ]) are used to enclose a keyword that is optional when programming the command. Therefore, the SE5082 will process the command to have the same effect whether the optional node is omitted by the programmer, or not. Letter case in tables is used to differentiate between the accepted short form (upper case) and the long form (upper and lower case).

The Parameter Form column indicates the number and order of a parameter in a command and their legal value. Parameter types are distinguished by enclosing the type in angle brackets (< >). If parameter form is enclosed by square brackets ([ ]) these are then optional (pay attention to be sure that optional parameters are consistent with the intention of the associated keywords). The vertical bar (|) can be read as "or" and is used to separate alternative parameter options.

SE5082 Commands

Table 4-1 lists all of the SE5082 SCPI commands. The commands are arranged in logical groups that provide similar functionality, even if they branch from different hierarchy headers, and make it easier to understand the various commands.

The SE5082 commands are described in 16 different groups, each describing a different type of operation. The groups are:

- Channel and group control commands
- Run mode commands
- Analog output control commands
- Marker output commands
- Standard waveforms commands
- Arbitrary waveforms commands
- Sequenced waveforms commands
- Advanced sequencing commands
- Modulated waveforms commands
- Pulse waveforms commands
- Pulse Pattern Commands
- LAN configuration commands
- Store/Recall Commands
- System commands

Detailed descriptions of each of the various SE5082 commands are given as follows. Note that Table 4-1 lists all of the commands that control two channels. For a single-channel version, ignore all of the commands that are irrelevant for a single-channel type.

Table 4-1, Model SE5082 Commands List Summary

Keyword	Parameter Form	Default	Notes
1. Channel and Group Control Commands			
:INSTrument			
[:SELect]	CH1 | CH2 | 1 | 2	CH1	Select channel for program
:COUPle			
:OFFSet	0 to ±(n-128) (n = waveform length)	0	Course offset adjustment
:SKEW	-3e-9 to 3e-9	0	Fine skew adjustment
:STATe	OFF | ON | 0 | 1	0	Couple all channels
:XINStrument			
:MODE	MASTer | SLAVe | MSLave	MAST	System configuration
:OFFSet	0 to n (n = waveform length)	0	Multi-instrument offset
:STATe	OFF | ON | 0 | 1	0	
:SKEW	-5e-9 to 5e-9	0	
2. Run Mode Commands			
:ABORt			Unconditional abort
:ARM			Applies to Event Input
[:SEQuence]			
:ECL			Sets ECL level
:LEVel	-5.0 to +5.0	1.6	
:SLOPe	POSitive | NEGative | EITHer	POS	
:TTL			Sets TTL level
:ENABle			Unconditional enable
:INITiate			
:CONTinuous			
:ENABle	SELF | ARMed	SELF	
:SOURce	BUS | EVENt | TRIGger	BUS	Defines enable source
[:STATe]	OFF | ON | 0 | 1	1	
:GATE			
[:STATe]	OFF | ON | 0 | 1	0	
:TRIGger			Applies to trigger input
[:IMMediate]			Same as *trg
[:SEQuence]			
:COUNt	1 to 16,777,216	1	Counted bursts
:DELay	0 to 8e6 (integers divisible by 8)	0	0 = OFF
:ECL			Sets ECL level
:FILTer			
:HPASs			Sets high pass width
:WIDTh	10e-9 to 2	100e-3	
[:STATe]	OFF | ON | 0 | 1	0	
:LPASs			Sets low pass width
:WIDTh	10e-9 to 2	1e-3	
[:STATe]	OFF | ON | 0 | 1	0	
:HOLDoff	100e-9 to 2	100e-9	

Table 4-2, Model SE5082 Commands List Summary (Continued)

Keyword	Parameter Form	Default	Notes
:STATe	OFF | ON | 0 | 1	0	
:LEVel	-5.0 to +5.0	1.6	
:INPut			
:IMPedance	50 | 10K	50	
:MODE	NORMal | OVERride	NORM	Normal or retriggerable
:SLOPe	POSitive | NEGative | EITHer	POS	
:SOURce			
[:ADVance]	EXTernal | BUS | TIMer | EVENt	EXT	Selects trigger source
:TIMer			
:MODE	TIME | DELay	TIME	
:DELay	152 to 8e6 (integers divisible by 8)	152	Stop to start delay
:TIME	200e-9 to 20	15e-6	
:TTL			Sets TTL level
3. Analog Output Control Commands			
:OUTPut			
[:STATe]	OFF | ON | 0 | 1	0	Toggles output(s) on/off
:SYNC			
:FUNCtion	PULSe | WCOMplete	PULS	Selects sync out shape
:POSition			
[:POINt]	0 to 32e6-32 (0 to 64e6 with option -1, divided by 32)	0	Programs sync position
:WIDTh	32 to n-32 (divided by 32)	32	Sync width (Pulse only)
:SOURce	CH1 | CH2 | 1 | 2	CH1	Sync source
[:STATe]	OFF | ON | 0 | 1	0	Toggles sync on/off
:SAMPle			
:FORMat	NRZ | NRTZ | RTZ | RF	NRZ	
[:SOURce]			
:FREQuency			
[:CW]	1.0 to 2.5e9 | MINimum | MAXimum	10e6	Standard Sine Wave Freq.
	1.0 to 1.25e9 | MINimum | MAXimum	10e6	Standard Square Wave Freq
	1.0 to 300e6 | MINimum | MAXimum	10e6	Other Standard Waves Freq
:RASTer	10e6 to 6e9 | MINimum | MAXimum	1e9	Arb waveforms frequency
:FIX?			Query SCLK of std. wave
:PULSe?			Query SCLK of Pulse wave
:SOURce	INTernal | EXTernal	INT	Selects SCLK source
:DIVider	1 to 64 (integers, 2^n only)	1	External clock divider
:FUNCtion			
:MODE	FIXed | USER | SEQuence | ASEQuence | MODulation | PULSe | PATTern	FIX	Selects function type
:ROSCillator			
:SOURce	INTernal | EXTernal	INT	
:EXTernal			
:FREQuency	10M | 20M | 50M | 100M	100M	

Table 4-3, Model SE5082 Commands List Summary (Continued)

Keyword	Parameter Form	Default	Notes
:VOLTage			DC coupled output
[:LEVel]			
[:AMPLitude]	50e-3 to 2.0 MINimum | MAXimum	0.5	HV (HV option)
	100e-3 to 1.2 | MINimum | MAXimum	0.5	DC (DC option)
	0.4 to 0.54 | MINmium | MAXimum	0.54	Direct (DR option)
:OFFSet			
[:COMMon]	-1.0 to 1.0 | MINimum | MAXimum	0	HV
	-0.5 to 0. 5 | MINimum | MAXimum	0	DC
4. Marker Output Commands			
[:SOURce]			
:MARKer[1|2]			Selects active marker
:DELay	0 to 3e-9	0	Delay from SYNC
[:STATe]	OFF | ON | 0 | 1	0	Toggles marker on/off
:POSition	0 to n-4 (n = segment length)	0	Position from start, int divided by 4
:WIDTh	0 to n-4 (n = segment length)	64	Marker width int divided by 4
:VOLTage			
[:LEVel]			
:HIGH	0.5 to 1.25	1.0	Marker high level
:LOW	0 to 0.8	0	Marker low level
5. Standard Waveforms Commands			
[:SOURce]			
:FUNCtion			
:SHAPe	SINusoid | TRIangle | SQUare | RAMP | SINC | GAUSsian | EXPonential | NOISe | DC	SIN	Standard function shape
:SINusoid			
:PHASe	-360.0 to 360.00 (degrees)	0	
:TRIangle			
:PHASe	-360.0 to 360.00 (degrees)	0	
:SQUare			
:DCYCle	0 to 99.99 (percent)	50	
:RAMP			
:DELay	0 to 99.999 (percent)	10	
:TRANsition			
[:LEADing]	0 to 99.999 (percent)	60.0	
:TRAiling	0 to 99.999 (percent)	30.0	
:SINC			
:NCYCle	4 to 100	10	
:GAUSsian			
:EXPonent	1 to 200	10	

Table 4-4, Model SE5082 Commands List Summary (Continued)

Keyword	Parameter Form	Default	Notes
:EXPonential			
:EXPonent	-100 to 100	-10	
:DC			
[:OFFSet]	-1.0 to 1.0	0	
6. Arbitrary Waveforms Commands			
:TRACe			
[:DATA]	#<header><binary_block>		Write waveform binary data to the ***active-segment*** of the ***active channel***
:DEFine	<segment-number>,<segment-length> <1 to 32,000><384 to Arbitrary-Memory-Size >		Define the length (divided by 32) of the segment.
:DEFine<n>?	<n = 1 to 32,000>		Query length of seg <n>
:DELete			
[:NAME]	1 to 32,000		Delete one segment
:ALL			Delete all segments
:POINts?			Queries active segment waveform length
:SELect	<segment number> between 1 and 32000	1	Select the active segment (of the active channel)
:SOURce	BUS | EXTernal	BUS	Toggle control source
:TIMing	COHerent | IMMediate	IMM	Select timing
:SEGMent			
[:DATA]	#<header><binary_block>		Write the segment-table of the active channel
7. Sequenced Waveforms Commands			
[:SOURce]			
:SEQuence			
:ADVance	AUTOmatic | ONCE | STEPped	AUTO	Sequence advancing mode
[:DATA]	#<header><binary_block>		Write the data of the active sequence.
:DEFine	<step_no>,<segment_no>,<loops>,<jump_flag> <1 to 49152><1 to 32000><1 to 16777215><0 or 1>.		Define the n'th step of the active sequence. Note that length (in steps) of each sequence is at least 3, and the lengths sum of all sequences must not exceed 49152.
:DELete			
[:NAME]	1 to 1,000		Delete the specified sequence.
:ALL			Delete all sequences.
:JUMP			
[:EVENt]	BUS | EVENt	BUS	Toggle jump source
:LENGth	Query only		Query the length (in steps) of the active sequence.
:SELect	1 to 1,000		Select the active sequence.
:SOURce	BUS | EXTernal	BUS	Toggle control source

Table 4-5, Model SE5082 Commands List Summary (Continued)

Keyword	Parameter Form	Default	Notes
:TIMing	COHerent | IMMediate	IMM	Jump timing
:PREStep	WAVE | DC	WAVE	DC is active in continuous and BUS source only
:ONCe			
:COUNt	1 to 16,777,216		The number of times the active sequence will be repeated
:SYNC			
:LOCK	<step_number>	1	Sync position
:RESet			Force the sequence to its first step
:TYPE	NORMal | ADVanced		Select the active sequencer type
8. Advanced Sequencing Commands			
[:SOURce]			
:ASEQuence			
:ADVance	AUTOmatic | ONCE | STEPped	AUTO	The advancing-mode of the advanced sequencer.
:DEFine	<step_no>,<sequence_no>,<loops>,<jump_flag> <1 to 1000><1 to 1000><1 to 1048575><0 or 1>		Define the n'th step of the advanced-sequencer. Note that the length of the advanced-sequencer should be between 3 and 1000.
:DELete			Deletes the advanced-sequencer's table
:LENGth	Only Query		Query the length (in steps)
:ONCe			
:COUNt	1 to 1,048,575		Set the number of times the advanced sequence will be repeated
:SYNC			
:LOCK	1 to 1,000	1	Sync position
:RESet			Force the Aseq to its first step
[:DATA]	#<header><binary-block> 4-bytes with the number of loops (1 to 1048575) 2-byts with the sequence number (1 to 1000) 1-byte with the jump-flag (either 0 or 1) 1-byte spare		Write the advanced-sequencer's table. Note that the length (in steps) of the advanced sequencer's table should be between 3 and 1000.
9. Modulated Waveforms Commands			
[:SOURce]			
:MODulation			
:TYPE	OFF | AM | FM | SWEep | CHIRp | FSK | ASK | FHOPping | AHOPping	OFF	
:CARRier			
[:FREQuency]	10e3 to 2.5e+9	1e6	
:FUNCtion	SINusoid | TRIangle | SQUare	SIN	
[:SOURce]			
:AM			
:FUNCtion			
[:SHAPe]	SINusoid | TRIangle | SQUare | RAMP	SIN	

Table 4-6, Model SE5082 Commands List Summary (Continued)

Keyword	Parameter Form	Default	Notes
:INTernal			
:FREQuency	100.0 to 100.0e6	1e3	The ratio: MODulation:CARRier:FREQ / AM:INTernal:FREQ Must be between 10.0 and 40.0e+3
:DEPTh	0 to 200	50	
:FM			
:DEViation	10e-3 to 1.25e+9	500e3	0.5 * Max Carrier Frequency
:FUNCtion			
[:SHAPe]	SINusoid | TRIangle | SQUare | RAMP	SIN	
:FREQuency	100.0 to 250.0e+6	10e3	0.1 * Max carrier frequency The ratio: MODulationCARRier:FREQ / FM:FREQ Must be between 10.0 and 40.0e+3
:MARKer			
[:FREQuency]	10e3 to 2.5e+9	505e3	
:SWEep			
:FREQuency			
[:STARt]	10e3 to 2.5e+9	40e6	
:STOP	10e3 to 2.5e+9	80e6	
:TIME	0.5e-6 to 0.01 (seconds)	10e-6	The product of: Max(SWEep:FREQ:START, SWEep:FREQ:STOP) * SWEep:TIME Must be between 10.0 and 40.0e+3
:DIRection	UP | DOWN	UP	
:SPACing	LINear | LOGarithmic	LIN	
:MARKer			
[:FREQuency]	10e3 to 2.5e+9	60e6	
:CHIRp			
:WIDTh	0.5e-6 to 0.01 (seconds)	10e-6	The product of: Max(CHIRp:FREQ:START, CHIRp:FREQ:STOP) * CHIRp:WIDTh Must be between 10.0 and 40.0e+3
:REPetition	200e-9 to 20	100e-6	
:FREQuency			
[:STARt]	10e3 to 2.5e+9	40e6	
:STOP	10e3 to 2.5e+9	80e6	
:DIRection	UP | DOWN	UP	
:SPACing	LINear | LOGarithmic	LIN	
:MARKer			
[:FREQuency]	10e3 to 2.5e+9	60e6	
:AMPLitude			
:DEPTh	0 to 100%	50%	
:DIRection	UP | DOWN	UP	
:SPACing	LINear | LOGarithmic	LIN	

Table 4-7, Model SE5082 Commands List Summary (Continued)

Keyword	Parameter Form	Default	Notes
:FSK			
:FREQuency			
:SHIFted	10e3 to 2.5e9	2e6	
:BAUD	0.1 to 1.0e9	10e3	
:MARKer	1 to FSK symbols list length	1	
:DATA	#<header><binary-data> Symbols-list length 2 to 1000.		Download the FSK symbols list.
:ASK			
[:AMPLitude]			
[:STARt]	100e-3 to 1.2 (v) for DC amplifier	0.5	
	50e-3 to 2.0 (v) for HV amplifier		
	50e-3 to 0.54 (v) for DR (DAC output)		
:SHIFted	100e-3 to 1.2 (v) for DC amplifier	1	
	50e-3 to 2.0 (v) for HV amplifier		
	50e-3 to 0.54 (v) for DR (DAC output)		
:BAUD	0.1 to 1.0e9	10e3	
:MARKer	1 to ASK symbols list length	1	
:DATA	#<header><binary-data> Symbols-list length 2 to 1000.		Download the ASK symbols list.
:FHOPping			
:DWELl			
:MODE	FIXed | VARiable	VAR	
[:TIMe]	1e-9 to 10 (seconds)	5e-6	The "fixed" dwell time. It should not exceed 10 / (length of the "fixed" FHOP list)
:FIXed			
:DATA	#<header><binary-data> Array of (2 and 256) double-float values		Download the frequencies list Total dwell-time must not exceed 10 seconds.
:VARiable			
:DATA	#<header><binary-data> Array of (2 to 256) double-float values (frequency, dwell-time) pairs		Download the list of (frequency, dwell-time) pairs Total dwell-time must not exceed 10 seconds.
:MARKer	1 to (active) FHOP list length	1	
:AHOPping			
:DWELl			
:MODE	FIXed | VARiable	VAR	
[:TIMe]	1e-9 to 10 (seconds)	5e-6	Should not exceed 10 / (length of the "fixed" AHOP list)

Table 4-8, Model SE5082 Commands List Summary (Continued)

Keyword	Parameter Form	Default	Notes
:FIXed			
:DATA	#<header><binary-data> Array of (2 to 256) double-float values 100e-3 to 1.2(v) for DC amplifier 50e-3 to 2.0 (v) for HV amplifier 50e-3 to 0.54 (v) for DR (DAC output)		Download the amplitudes list Total dwell-time (=fixed_dwell_time * list_length) must not exceed 10 seconds.
:VARiable			
:DATA	#<header><binary-data> Array of (2 to 256) double-float values(amplitude, dwell-time) pairs 100e-3 to 1.2(v) for DC amplifier 50e-3 to 2.0 (v) for HV amplifier 50e-3 to 0.54 (v) for DR (DAC output)		Download the list of (amplitude, dwell-time) pairs Total dwell-time (=sum of the variable dwell-times) must not exceed 10 seconds.
:MARKer	1 to (active) AHOP list length	1	
10. Pulse Commands			
:PULSe			
:CONFigure	TIME | PERCent	TIME	
:DELay	0, 0.2e-9 to 1.6-0.2e-9	1e-6	Delay in seconds.
:PERCent	0 .01 to 99.99	10	Delay in %
:DOUBle			
:DELay	0, 0.2e-9 to 1.6-0.2e-9	1e-6	Double delay in seconds
:PERCent	0 .01 to 99.99	10	Delay in %
:LEVel			
[:CONTrol]	HLOW | AOFFset | POSitive | NEGative	HLOW	
:HIGH	-0.75 to +0.75 for DC amplifier	0.5	
	-1.5 to +1.5 for HV amplifier	0.5	
	-0.27 to +0.27 for DR (DAC output)	0.25	
:LOW	-0.75 to +0.75 for DC amplifier	0	
	-1.5 t0 +1.5 for HV amplifier	0	
	-0.27 to +0.27 for DR (DAC output)	-0.25	
:AMPLitude	100e-3 to 1.0 for DC amplifier	0.5	Amplitude in volts. Also accepts MINimum | MAXimum
	50e-3 to 2.0 for HV amplifier	0.5	
	50e-3 to 0.54 for DR (DAC output)	0.5	
:OFFSet	-0.5 to 0.5 for DC amplifier	0.25	DC offset in volts Also accepts MINimum | MAXimum
	-1.0 to 1.0 for HV amplifier	0.2	
	0 for DR (DAC output)	0	
:MODE	SINGle | DELayed | DOUBle	SING	
:POLarity	NORMal | COMPlement | INVerted	NORM	
:PERiod	0.8e-9 to 1.6	10e-6	Total pulse period in seconds
:TRANsition			
:STATe	FAST | LINear | SYMMetrical	FAST	Transition type
[:LEADing]	Between Ɛ and 1.6-Ɛ, Ɛ=0.2e-9	1e-6	Rise time in sec
:PERCent	0 .01 to 99.99	10	Rise time in %

Table 4-9, Model SE5082 Commands List Summary (Continued)

Keyword	Parameter Form	Default	Notes
:TRAiling	Between Ɛ and 1.6-Ɛ, Ɛ=0.2e-9	1e-6	Fall time in sec
:PERCent	0 .01 to 99.99	10	Fall time in %
:WIDTh	Between Ɛ and 1.6-Ɛ, Ɛ=0.2e-9	2e-6	Width in sec
:PERCent	0 .01 to 99.99	20	Width in %
11.Pattern			
:PATTern			
:MODE	PRBS | COMPoser		
[:PRBS]			
:TYPE	PRBS7 | PRBS9 | PRBS11 | PRBS15 | PRBS23 | PRBS31 | USER	PRBS7	
:BAUD	1 to 1e+9	10e6	
:LEVel	2 | 3 | 4 | 5	2	
:HIGH	-0.75 to 0.75 for DC amplifier	0.5	
	-1.5 to 1.5 for HV amplifier	1.0	
	-0.27 to +0.27 for DR (DAC output)		
:LOW	-0.75 to 0.75 for DC amplifier	-0.5	
	-1.5 to 1.5 for HV amplifier	-1.0	
	-0.27 to +0.27 for DR (DAC output)		
:LOOPs	1 to 1e6	1	
:PREamble	1 to 16e6	1	
:LENGth	1 to 16e6	32	
:DATA	#<header><binary-data> The binary data is of N bytes that hold the user-defined symbols-list of the PRBS composer. Each byte holds a single ASCII character from '0', '1', '2', '3' or '-', that defines a single symbol.		Download the symbols list of the USER-PRBS pattern. Maximal length of the symbols list is 16e+6
:COMPoser			
:TRANsition			
:TYPe	FAST | LINear	LINear	Transition type
:FAST			
[:DATA]	#<header><binary-data> The binary-data is array of N*12 bytes holding the definitions of the N flat-intervals of the Fast-Pattern, Where each 12-bytes sub-block consists: DC-Level as float value (4-bytes) between (-maxV_{offs} - maxV_{pp}/2) and (+maxV_{offs} +maxV_{pp}/2) Dwell-Time as double-float (8-bytes)		Download the definitions of the N flat-intervals of the Fast-Pattern. Difference between the highest and lowest levels should not exceed maxV_{pp} The G.C.D of all dwell-times should be at least 0.2e-9 (seconds).
:LINear			
:STARt	(-maxV_{offs} - maxV_{pp}/2) to (+maxV_{offs} + maxV_{pp}/2)	0.25 (volt)	The start-level (in volts) of the first linear-interval in the Linear-Pattern.

Table 4-10, Model SE5082 Commands List Summary (Continued)

Keyword	Parameter Form	Default	Notes
:RESolution	0.2e-9 to 100e-9	0.2e-9 sec	The time-resolution (the sampling time of each waveform point)
:TYPE	AUTO | USER	AUTO	AUTO means that the time resolution is automatically selected. USER means that it is set manually by the user . In the later case, it should divide the G.C.D of all the pattern's dwell-times.
12. LAN Configuration Commands			
:SYSTem			
:IP			
[:ADDRess]	<IP_address>		
:MASK	<mask>		
:GATeway	<gate_way>		
:BOOTp	OFF | ON | 0 | 1	0	
:HOSTname:	<host_name>		
:MAC?			
:KEEPalive			
:STATe	OFF | ON | 0 | 1	1	
:TIMEout	2 to 300	45	
:PROBes	2 to 10	2	
:FIRMware			
:PORT?			
:TELNet			
:PORT?			
13. Store/Recall Commands			
:SYSTem			
:STORe			
:CELL	1 to 9	1	
:CLEar			Clear cell
:CONFig	SETup | WAVE | ALL	ALL	
:TARGet	INTernal | USB	INT	
:UPDate			
:RECall			
:CELL	1 to 9	1	
:TARGet	INTernal | USB	INT	
:UPDate			
14. System Commands			
:RESet			Same as *RST
:SYSTem			
:ERRor?			
:TEMPerature?			
:POWerup	DEFault | SETup	SET	
:VERSion?			

Table 4-11, Model SE5082 Commands List Summary (Continued)

Keyword	Parameter Form	Default	Notes
:INFormation			
:CALibration?			
:MODel?			
:SERial?			
:HARDware?			
:FIRMware			
:DATE?			
:SVN?			
:VISA			
:TCPIP?			
*CLS			
*OPC?			
*RST			
*TRG			
*IDN?			<vendor-name>,<instrument model name>,<serial-number>,<firmware version>
*STB?			
*TST?			
*ESE			
*SRE			
*ESR?			
*OPT?	# Channels, Installed Memory, Output Board		1 character for Channels: 1, 2 2 character for Memory: 32, 64 2 character for Output: HV, DC, DR

Channel & Group Control Commands

Commands that are listed in Table 4-2 control parameters that group and synchronize two or more channels, as well as their relative offset and skew parameters. Factory defaults after *RST are shown in the Default column. Parameter range and low and high limits are listed, where applicable.

Note the command *inst ch1* / *ch2.* The default parameter is *ch1*, which means that commands that are sent to the SE5082 affect channel 1 settings only. Select the *inst ch2* parameter if you want to program channel 2 parameters.

Tip

The Model SE5081 has only one output channel and therefore, the command *inst:sel* is not required for correct programming sequences however, for the Model SE5082, it is mandatory to select the proper channel before commands are being streamed to the product. The SE5082 has two completely separate channels and therefore, each of the commands that are listed in the following tables should be preceded by the *inst:sel 1* or *inst:sel 2* commands.

There are some commands that affect both channels simultaneously and do not require that a specific channel is selected; these are LAN configuration commands, Store/recall commands, System Commands and common commands.

Note that the two channels can operate in synchronized mode where both channels share the same output waveform type, the same run mode option and a common sample clock feed. In this case, the sample clock value should be programmed for channel 1 only but the other commands still require the use of *inst:sel* command.

Table 4-12, Channel & Group Control Commands Summary

Keyword	Parameter Form	Default	Notes
:INSTrument			
[:SELect]	CH1 | CH2 | 1 | 2	CH1	Select channel for program
:COUPle			
:STATe	OFF | ON | 0 | 1	0	Couple all channels
:OFFSet	0 to ±(n-128) (n = waveform length)	0	Course offset adjustment
:SKEW	-3e-9 to 3e-9	0	Fine skew adjustment
:XINStrument			
:MODE	MASTer | SLAVe | MSLave	MAST	System configuration
:STATe	OFF | ON | 0 | 1	0	
:OFFSet	0 to n (n = waveform length)	0	Multi-instrument offset
:SKEW	-5e-9 to 5e-9	0	

:INSTrument{CH1|CH2|1|2}(?)

Description

This command will set the active channel for future programming command sequences. Subsequent commands affect the selected channel only.

Parameters

Range	Type	Default	Description
CH1-CH2 or 1-2	Discrete	CH1	Sets the active channel for programming from remote.

Response

The SE5082 will return 1 or 2 depending on the present active channel setting.

Example

Command `:INST 1`

Query `:INST?`

:INSTrument:COUPle:STATe{OFF|ON|0|1}(?)

Description

Sets or queries the couple state of the synchronized channels. Use this command to cause the two channels to synchronize. Following this command, the sample clock of channel 1 will feed the channel 2 and the start phase of the channel 2 channels will lock to the channel 1 waveform.

Parameters

Range	Type	Default	Description
0-1	Discrete	0	Sets the couple mode on and off

Response

The SE5082 will return 1 if the couple state is ON, or 0 if the couple state is OFF.

Example

Command `:INST:COUP:STATE ON`

Query `:INST:COUP:STATE?`

:INSTrument:COUPle:OFFSet{<ch_offset>}(?)

Description

When couple state is ON, this command sets or queries the offset between the start phase of the master channel (CH1) and the start phase of the slave channel (CH2).

Parameters

Name	Range	Type	Default	Description
<ch_offset>	0 to n-128	Numeric (integer only)	0	Defines a coarse phase offset between two channels. When offset is applied to one channel it is always in reference to the other channel. For example, offsetting channel 2 by 1024 points and then offsetting channel 1 by 2048 points will cause channel 2 waveform to lag channel 1 by 1024 points. Offset can be programmed in increments of 8 sample clock periods. The two channels must have the same waveform length in order for the phase offset parameter to be meaningful. Offset range is programmable in units of waveform points from 0 to the maximum length of the channel waveform, less 128 waveform points. Fine adjustment of phase offset between channels is achieved using the *inst:coup:skew* command. Note that this parameter is operating in conjunction with the continuous run mode and only when the two channels are synchronized.

Response

The SE5082 will return the present value of the coarse offset setting in units of waveform points.

Example

Command `:INST:COUP:OFFS 32`

Query `:INST:COUP:OFFS?`

:INSTrument:COUPle:SKEW{<ch_skew>}(?)

Description

When couple state is ON, this command sets or queries the skew between the two channels. Skew defines fine offset between channels in units of time. Only channel 2 has the skew computed in reference to channel 1. The inst:coupl:skew command is applied automatically to channel 2 and does not require that you use the inst:sel 2 command.

Parameters

Name	Range	Type	Default	Description
<ch_skew>	-3e-9 to 3e-9	Numeric	0	Defines channel 2 skew in reference to channel 1.

	The two channels must have the same waveform length in order for the phase skew parameter to be meaningful. The skew range is programmable in units of seconds throughout the range of ±3 ns. Coarse adjustment of phase offset between channels is achieved using the *inst:coup:offs* command. Note that this parameter is operating in conjunction with the continuous run mode and only when two or more channels are synchronized.

Response

The *SE5082* will return the present value of the skew setting in units of seconds.

Example

Command `:INST:COUP:SKEW 1e-9`

Query `:INST:COUP:SKEW?`

:XINStrument:MODE{MASTer|SLAVe}

Description

Use this command to define master and slave instruments. This command affects the system only when a synchronization cable is attached to the rear panel of both instruments and is intended to operate in synchronized mode.

Parameters

Name	Type	Default	Description
<MASTer>	String	MAST	Defines the active instrument as the controlling master in a two-instrument system. Only channel 1 in each instrument can be defined as master; the rest of the channels are defined as slaves.
<SLAVe>	String		Defines the active instrument as a slave in a two-instrument system.

Response

The SE5082 will return MAST or SLAV depending on the present mode setting.

Example

Command `:XINST:MODE SLAV`

Query `:XINST:MODE?`

:XINSTrument:STATe{OFF|ON|0|1}(?)

Description

Sets or queries the couple state of the synchronized instruments. Use this command to synchronize two instruments to a single sample clock reference. This command can be used only after defining one instrument as master (xins:mode mast) and the other as slave (xins:mode slav). Use the xins:stat 1 on the master instrument only. The reference sample clock is generated by the master instrument

Parameters

Range	Type	Default	Description
0-1	Discrete	0	Sets the two-instruments couple on and off

Response

The SE5082 will return 1 if the couple is ON, or 0 if the couple is OFF.

Example

Command `:XINST:STAT ON`

Query `:XINST:STAT?`

:XINStrument:OFFSet{<instrument_offset>}(?)

Description

When couple state is ON, this command sets or queries the offset between the start phase of the slave instrument in reference to the master instrument.

Parameters

Name	Range	Type	Default	Description
<instrument_offset>	0 to n	Numeric (integer only)	0	Defines a coarse phase offset between two instruments. When offset is applied to one instrument it is always in reference to the other instrument. For example, offsetting the slave instrument by 1024 points and then offsetting master instrument by 2048 points will cause slave waveform to lag the master waveform by 1024 points. Offset can be programmed in increments of 8 sample clock periods.

Response

The SE5082 will return the present value of the coarse offset setting in units of waveform points (SCLK periods).

Example

Command `:XINST:OFFS 8`

Query `:XINST:OFFS?`

:XINSTrument:SKEW{<Inst_skew>}(?)

Description

When couple state is ON, this command sets or queries the skew between the start phase of the slave instrument in reference to the master instrument.

Parameters

Name	Range	Type	Default	Description
<Inst_skew>	-5e-9 to 5e-9	Numeric	0	Defines a phase offset between two instruments. When offset is applied to one instrument it is always in reference to the other instrument.

Response

The *SE5082* will return the present value of the skew setting in units of seconds.

Example

Command `:XINST:SKEW 1e-9`

Query `:XINST:SKEW?`

Run Mode Commands

The Run Mode Commands group is used to synchronize device actions with external or internal events.

The SE5082 can operate in two basic modes: self-armed and armed.

Self-armed mode is the default option where waveforms are generated at the output connector, immediately after the output function has been selected.

In armed mode, the SE5082 requires an enable command or an external analog event to cause the output to start generating waveforms and when already armed, a remote abort command will cease the generation of the signal and the output will return to a known idle state. This mode is very useful to control how and when the waveform will start and stop for systems that require precise control of waveform timing.

Other commands in this group control the basic run modes of the waveform generator. The available run modes are:

- continuous, where waveforms are generated continuously at the output connector and triggered and gated.
- conditional, where waveforms are generated on conditional events, regardless if they are generated internally from a built-in trigger generator or applied externally to the trigger and event inputs.

Also use the commands in this group to control the sensitivity, the polarity and other conditions of which external signals will affect the trigger and event inputs.

A built-in counter is available to control a precise number of cycles for applications requiring a burst of waveforms that follows a trigger event.

Additional information on the run mode options and how the generator behaves under the various run mode options is given in Chapter 3. Factory defaults after *RST are shown in the default column. Parameter low and high limits are given where applicable. Use the commands in Table 4-3 to set up the SE5082 run mode and for setting up the input conditions for the various trigger inputs.

Table 4-13, Run Mode Commands Summary

Keyword	Parameter Form	Default	Notes
:ABORt			Unconditional abort
:ARM			Applies to Event Input
[:SEQuence]			
:ECL			Sets ECL level
:LEVel	-5.0 to +5.0	1.6	
:SLOPe	POSitive | NEGative | EITHer	POS	
:TTL			Sets TTL level
:ENABle			Unconditional enable
:INITiate			
:CONTinuous			
:ENABle	SELF | ARMed	SELF	
:SOURce	BUS | EVENt | TRIGger	BUS	Defines enable source
[:STATe]	OFF | ON | 0 | 1	1	
:GATE			
[:STATe]	OFF | ON | 0 | 1	0	
:TRIGger			Applies to trigger input
[:IMMediate]			Same as *trg
[:SEQuence]			
:COUNt	1 to 16,777,216	1	Counted bursts
:DELay	0 to 8e6 (integers divisible by 8)	0	0 = OFF
:ECL			Sets ECL level
:FILTer			
:HPASs			Sets high pass width
:WIDTh	10e-9 to 2	100e-3	
[:STATe]	OFF | ON | 0 | 1	0	
:LPASs			Sets low pass width
:WIDTh	10e-9 to 2	1e-3	
[:STATe]	OFF | ON | 0 | 1	0	
:HOLDoff	100e-9 to 2	100e-9	
:STATe	OFF | ON | 0 | 1	0	
:LEVel	-5.0 to +5.0	1.6	
:INPut			
:IMPedance	50 | 10K	50	
:MODE	NORMal | OVERride	NORM	Normal or retriggerable
:SLOPe	POSitive | NEGative | EITHer	POS	
:SOURce			
[:ADVance]	EXTernal | BUS | TIMer | EVENt	EXT	Selects trigger source
:TIMer			
:MODE	TIME | DELay	TIME	
:DELay	152 to 8e6 (integers divisible by 8)	152	Stop to start delay
:TIME	200e-9 to 20	15e-6	
:TTL			Sets TTL level

:ABORt

Description

Use this command for an immediate and unconditional termination of the output waveform. A prerequisite condition that makes this command effective is to place the SE5082 in continuous and armed run mode and then enable the output using the *enab* command. This command is also effective in all triggered run mode options. Following the abort command, the SE5082 stops generating waveforms and the output starts generating an idle waveform that could be one of: dc, first waveform in a sequence or first sequence in an advanced sequence scheme. The abort command ignores the trac:sel:tim and the sequ:sel:tim settings. The resulting scenarios of the abort command are summarized in the Run Modes Summary table in Chapter 3.

Example

Command `:ABOR`

:ARM:ECL

Description

Use this command to set the event input to accept ecl (negative) signals. The threshold level is automatically set to -1.3 V, which is the mid-level for negative ecl logic. Other related commands are: *arm:ttl* to set the threshold level for TTL signals and *arm:lev* to program a threshold level between -5 V to 5 V. Note that commands that start with *ARM* affect the conditions for the event input only.

Example

Command `:ARM:ECL`

:ARM:LEVel<level>(?)

Description

This command programs the threshold level for the event input signals. Other related commands are: *arm:ttl* to set the threshold level for TTL signals and *arm:ecl* to program a threshold level for ECL signals. Note that commands that start with *ARM* affect the conditions for the event input only.

Parameters

Name	Range	Type	Default	Description
<level>	-5 to 5	Numeric	1.6	Programs the threshold level for the invent input.

Response

The SE5082 will return the present threshold level setting in units of volts.

Example

Command `:ARM:LEV 3.3`

Query `:ARM:LEV?`

:ARM:SLOPe{POSitive|NEGative|EITHer}(?)

Description

Use this command to define the edge that will affect the event input. Positive going transitions will affect the event input when the POS option is selected. Negative transitions will affect the event input when the NEG option is selected. Both transitions will affect the event input when the EITH option is selected. Note that commands that start with *ARM* affect the conditions for the event input only.

Parameters

Name	Type	Default	Description
POSitive	Discrete	POS	Selects the positive going edge.
NEGative	Discrete		Selects the negative going edge.
EITHer	Discrete		Selects both positive and negative going edges.

Response

The SE5082 will return POS, NEG, or EITH depending on the current slope setting.

Example

Command `:ARM:SLOP NEG`

Query `:ARM:SLOP?`

:ARM:TTL

Description

Use this command to set the event input to accept ttl signals. The threshold level is automatically set to 1.6 V, which is the mid-level for ttl logic. Other related commands are: *arm:ecl* to set the threshold level for ECL signals and *arm:lev* to program a threshold level between -5 V to 5 V. Note that commands that start with *ARM* affect the conditions for the event input only.

Example

Command `:ARM:TTL`

:ENABle

Description

Use this command for an immediate and unconditional generation of the selected output waveform. Prerequisite condition that makes this command effective is to place the SE5082 in continuous and armed run mode. This command has no effect in triggered run mode options. Generation of the output waveform can be terminated abruptly using the *abor* command. Following the abort command, the SE5082 stops generating waveforms and the output starts generating an idle waveform that could be one of: dc, first waveform in a sequence or first sequence in an advanced sequence scheme. The resulting scenarios of the enable and abort commands are summarized in the Run Modes Summary table in Chapter 3.

Example

Command `:ENAB`

:INITiate:CONTinuous:ENABle{SELF|ARMed}(?)

Description

Use this command to set or query the state of the enable mode. This command is effective in continuous mode only and has no effect when the *init:cont 0* or *init:gate 1* (triggered or gated modes) were executed.

Parameters

Name	Type	Default	Description
SELF	Discrete	SELF	In continuous run mode, waveforms are generated at the output connector as soon as they are selected.
ARMed	Discrete		The SE5082 generates waveforms at the output connector only after an *enab* command is executed. For an immediate and unconditional termination of the output waveform use the *abor* command.

Response

The SE5082 will return SELF, or ARM depending on the current enable mode setting.

Example

Command `:INIT:CONT:ENAB ARM`

Query `:INIT:CONT:ENAB?`

:INITiate:CONTinuous{1|0|ON|OFF}(?)

Use this command to set or query the run mode status.

Parameters

Range	Type	Default	Description
1-0	Discrete	1	“1” selects the continuous run mode. “0” disables the continuous operation and forces the triggered run mode. Trigger signal is applied to the trigger input only and output waveforms will be generated only when the trigger signal is valid and true. The slope and level of the trigger input are programmable.

Response

The SE5082 will return 1 or 0 depending on the current run mode setting.

Example

Command `:INIT:CONT OFF`

Query `:INIT:CONT?`

:INITiate:CONTinuous:ENABle:SOURce{BUS|EVENt|TRIGger}(?)

Description

Use this command to set or query the source of the enable signal. This command is effective in continuous mode only and has no effect when the *init:cont 0* or *init:gate 1* (triggered or gated modes) were executed.

Parameters

Name	Type	Default	Description
BUS	Discrete	BUS	Defines the source of the enable signal as a remote command sent over one of the interfacing controllers (USB, LAN or GPIB). Signals at the event input will be ignored. In continuous run mode, waveforms are generated at the output connector as soon as a remote enable command is executed. For an immediate and unconditional termination of the output waveform use the *abor* command.
EVENt	Discrete		Defines the source of the enable signal as the event input connector. Remote enable commands will be ignored. In continuous run mode, waveforms are generated at the output connector as soon as a valid event signal is sensed at the event input connector. For an immediate and unconditional termination of the output waveform use the *abor* command.
TRIGger	Discrete		Defines the source of the enable signal as the front panel trigger input connector. Remote enable commands will be ignored. In continuous run mode, waveforms are generated at the output connector as soon as a valid event signal is sensed at the trigger input connector. For an immediate and unconditional termination of the output waveform use the *abor* command.

Response

The SE5082 will return BUS, EVEN, or TRIG depending on the current enable source setting.

Example

Command `:INIT:CONT:ENAB:SOUR BUS`

Query `:INIT:CONT:ENAB:SOUR?`

:INITiate:GATE{0|1|OFF|ON}(?)

Description

Use this command to set or query the gated run mode status.

Parameters

Range	Type	Default	Description
0-1	Discrete	0	"0" run mode setting is unchanged. "1" selects the gated run mode. The gated run mode should only be selected if continuous run mode is off otherwise it has no effect on the current run mode. Gating signal is applied to the trigger input only and output waveforms will be generated only when the gate signal is valid and true. The slope and level of the gating entry are programmable.

Response

The SE5082 will return 1 or 0, depending on the current run mode setting.

Example

Command `:INIT:GATE 1`

Query `:INIT:GATE?`

:TRIGger

Description

Use this command to trigger the SE5082 from a remote computer. You may also use the common command *trg that will have the same effect. This command will affect the SE5082 only after you program the instrument to operate in triggered run mode (*init:cont 0*) and select the trigger source BUS option. Note that commands that start with *TRIG* affect the conditions for the trigger input only.

Example

Command `:*TRG`

:TRIGger:COUNt<burst>(?)

Description

Use this command to set or query the burst counter setting. This command is effective only when the SE5082 is programmed to operate in triggered run mode (*init:cont 0*).

Parameters

Name	Range	Type	Default	Description
<burst>	1 to 16,777,216	Numeric (integer only)	1	Programs the burst count. Following a valid trigger signal, the SE5082 generates a pre-programmed number of waveform cycles and then resumes an idle state. The counted burst can be initiated using one of the following: front panel MANUAL push button, remote command such as *trg, or a transition at the trigger input connector.

Response

The SE5082 will return the present burst count value.

Example

Command `:TRIG:COUN 1000`

Query `:TRIG:COUN?`

:TRIGger:DELay<interval>(?)

Description

Use this command to set or query the trigger delay setting. The trigger delay parameter defines the interval that will elapse from a valid trigger signal to the initiation of the first output waveform. Trigger delay is turned off using the *trig:del 0* command. The trigger delay command affects the generator only after it has been programmed to operate in triggered run mode. Modify the SE5082 to triggered run mode using the *init:cont 0* command. The delay interval is programmed in sample clock period increments.

Parameters

Name	Range	Type	Default	Description
<interval>	0 to 8e6	Numeric (integer only)	0	"0" turns OFF the delayed trigger function. Delay is programmed in sample clock period increments, so expect the delay time to change if you modify your sample clock setting. Program the delay interval using integer numbers divisible by 8 only.

Response

The SE5082 will return the present trigger delay interval value.

Example

Command `:TRIG:DEL 1000`

Query `:TRIG:DEL?`

:TRIGger:ECL

Description

Use this command to set the trigger input to accept ecl (negative) signals. The threshold level is automatically set to -1.3 V, which is the mid-level for negative ecl logic. Other related commands are: trig*:ttl* to set the threshold level for TTL signals and trig*:lev* to program a threshold level between -5 V to 5 V.

Example

Command `:TRIG:ECL`

:TRIGger:FILTer:HPASs:WIDTh<width>(?)

Description

Use this command to set or query the trigger high pass filter value. Trigger signal having pulse width below the programmed settings will not trigger the generator. The trigger filter has three options: high pass filter, which will trigger the SE5082 only if the width is larger than the programmed value, low pass filter, which will trigger the generator only if the width is smaller than the programmed value, and window pass filter, which will trigger the generator only if the width is within a certain range specified by the high and low pass filters.

The *trig:fil:hpas:wid* command sets a high pass threshold for the trigger signal and the *trig:fil:lpas:wid* command sets a low pass threshold for the trigger signal. If both the high and low pass filters are turned on, signals having a pulse width smaller than the low pass setting and larger that the high pass setting will trigger the generator.

Parameters

Name	Range	Type	Default	Description
<time>	10e-9 to 2	Numeric	100e-3	Programs the high pass pulse width value in seconds with 2ns resolution.

Response

The SE5082 will return the present high pass value in units of seconds.

Example

Command `:TRIG:FILT:HPAS:WIDT 200e-9`

Query `:TRIG:FILT:HPAS:WIDT?`

:TRIGger:FILTer:HPASs{OFF|ON|0|1}(?)

Description

Use this command to set or query the status of the high pass filter.

Range	Type	Default	Description
0-1	Discrete	0	Turns the high pass filter on and off.

Response

The SE5082 will return 0, or 1 depending on the present high pass filter state.

Example

Command `:TRIG:FILT:HPAS ON`

Query `:TRIG:FILT:HPAS?`

:TRIGger:FILTer:LPASs:WIDTh<width>(?)

Description

Use this command to set or query the trigger low pass filter value. Trigger signal having pulse width above the programmed setting will not trigger the generator. The trigger filter has three options: high pass filter, which will let trigger the SE5082 only if the width is larger than the programmed value, low pass filter, which will trigger the generator only if the width is smaller than the programmed value, and window pass filter, which will trigger the generator only if the width is within a certain range specified by the high and low pass filters.

The *trig:fil:hpas:wid* command sets a high pass threshold for the trigger signal and the *trig:fil:lpas:wid* command sets a low pass threshold for the trigger signal. If both the high and low pass filters are turned on, signals having pulse width smaller than the low pass setting and larger that the high pass setting will trigger the generator.

Parameters

Name	Range	Type	Default	Description
<time>	10e-9 to 2	Numeric	1e-3	Programs the high pass pulse width value in units of seconds with 2ns resolution.

Response

The SE5082 will return the present high pass value in units of seconds.

Example

Command `:TRIG:FILT:LPAS:WIDT 200e-3`

Query `:TRIG:FILT:LPAS:WIDT?`

:TRIGger:FILter:LPASs{OFF|ON|0|1}(?)

Description

Use this command to set or query the status of the low pass filter.

Range	Type	Default	Description
0-1	Discrete	0	Turns the low pass filter on and off.

Response

The SE5082 will return 0 or 1 depending on the current low pass filter state.

Example

Command `:TRIG:FILT:LPAS 1`

Query `:TRIG:FILT:LPAS?`

:TRIGger:HOLDoff<holdoff>(?)

Description

Use this command to set or query the trigger holdoff period. The trigger holdoff filter defines a period that starts with the first valid trigger input and ends with the holdoff setting of which all triggers within this range, valid or not, are ignored, but the first valid signal after the holdoff range will cause the SE5082 to generate a waveform.

Parameters

Name	Range	Type	Default	Description
<time>	100e-9 to 2	Numeric	100e-9	Programs the trigger holdoff period in units of second.

Response

The SE5082 will return the present holdoff value in units of second.

Example

Command `:TRIG:HOLD 100e-6`

Query `:TRIG:HOLD?`

:TRIGger:HOLDoff:STATe{OFF|ON|0|1}(?)

Description

Use this command to set or query the status of the holdoff filter.

Range	Type	Default	Description
0-1	Discrete	0	Turns the holdoff filter on and off.

Response

The SE5082 will return 0 or 1 depending on the present holdoff filter state.

Example

Command `:TRIG:HOLD:STAT ON`

Query `:TRIG:HOLD:STAT?`

:TRIGger:INPut:IMPedance{50|10K}

Description

Use this command to select the EXT trigger input impedance.

Name	Type	Default	Description
50	Discrete	50	Selects 50Ω trigger input impedance.
10K	Discrete		Selects 1KΩ trigger input impedance

Response

The SE5082 will return 50 or 1K depending on the present trigger input impedance.

Example

Command `:TRIG:INP:IMP 1K`

:TRIGger:LEVel<level>(?)

Description

Use this command to program or query the threshold level for the trigger input signals. Other related commands are: trig*:ttl* to set the threshold level for TTL signals and trig*:ecl* to program a threshold level for ECL signals. Note that commands that start with trig affect the conditions for the trigger input only.

Parameters

Name	Range	Type	Default	Description
<level>	-5 to 5	Numeric	1.6	Programs the threshold level for the trigger input.

Response

The SE5082 will return the present threshold level setting in units of volts.

Example

Command `:TRIG:LEV 2.5`

Query `:TRIG:LEV?`

:TRIGger:MODE{NORMal|OVERride}(?)

Description

Use this command to define or query the trigger mode. In normal mode, the first trigger activates the output and consecutive triggers are ignored for the duration of the output waveform. In override mode, the first trigger activates the output and consecutive triggers restart the output waveform, regardless if the current waveform has been completed or not.

Parameters

Name	Type	Default	Description
NORMal	Discrete	NORM	Selects the normal trigger mode.
OVERride	Discrete		Selects the override trigger mode.

Response

The SE5082 will return NORM or OVER, depending on the current trigger mode setting.

Example

Command `:TRIG:MODE OVER`

Query `:TRIG:MODE?`

:TRIGger:SLOPe{POSitive|NEGative|EITher}(?)

Description

Use this command to define or query the edge that will affect the trigger input. Positive going transitions will affect the trigger input when the POS option is selected. Negative transitions will affect the trigger input when the NEG option is selected and both positive and negative transitions will affect the trigger input when the EIT option is selected. Note that commands that start with *ARM* affect the conditions for the event input only.

Parameters

Name	Type	Default	Description
POSitive	Discrete	POS	Selects the positive going edge.
NEGative	Discrete		Selects the negative going edge.
EITher	Discrete		Selects both positive and negative going edges.

Response

The SE5082 will return POS, NEG, or EIT depending on the current slope setting.

Example

Command `:TRIG:SLOP NEG`

Query `:TRIG:SLOP?`

:TRIGger:SOURce:ADVance{EXTernal|BUS|TIMer|EVENt}(?)

Description

Use this command to set or query the source of the trigger event that will stimulate the SE5082 to generate waveforms. The source advance command will affect the generator only after it has been programmed to operate in trigger run mode. Modify the SE5082 to trigger run mode using the *init:cont off* command.

Parameters

Name	Type	Default	Description
EXTernal	Discrete	EXT	Selects the TRIG IN connector as the input source. The front panel MANUALcan be used in case external triggers are not available. All other inputs are ignored.
BUS	Discrete		Selects the remote controller as the trigger source. Only software commands are accepted; TRIG IN, Event IN and manual triggers are ignored.
TIMer	Discrete		Activates the built in internal trigger generator. BUS

		and external trigger are ignored. The period of the internal trigger is programmable and can be used to replace an external trigger source.
EVENt	Discrete	Selects the Event IN connector as the input source. All other inputs are ignored.

Response

The SE5082 will return EXT, BUS, TIM, or EVEN depending on the selected trigger source advance setting.

Example

Command `:TRIG:SOUR:ADV BUS`

Query `:TRIG:SOUR:ADV?`

:TRIGger:TIMer:MODE{TIME|DELay}(?)

Description

Use this command to set or query the mode that the internal trigger generator will operate. Timed defines start-to-start triggers and Delayed defines end-to-start triggers. The timer commands will affect the generator only after it has been programmed to operate in timer mode. Modify the SE5082 to trigger run mode using the *init:cont off* command and program the internal timer using the *trig:tim* command.

Parameters

Name	Type	Default	Description
TIME	Discrete	TIME	Selects the timed internal trigger generator. The generator will automatically issue periodical triggers that stimulate the output to generate single cycle waveforms. The periods are programmed in units of seconds from waveform start to waveform start.
DELay	Discrete		Selects the delayed internal trigger generator. The generator will automatically issue periodical triggers that stimulate the output, to generate single cycle waveforms. The periods are programmed in units of waveform points from waveform end to waveform start.

Response

The SE5082 will return TIME or DEL depending on the selected internal trigger timer mode setting.

Example

Command `:TRIG:TIM:MODE DEL`

Query `:TRIG:TIME:MODE?`

:TRIGger:TIMer:DELay<timer>(?)

Description

Use this command to set or query the delay setting of the internal delayed trigger generator. This value is associated with the internal trigger run mode only and has no effect on other trigger modes. The internal delayed trigger generator is a free-running oscillator, asynchronous with the frequency of the output waveform. The timer intervals are measured from waveform stop to waveform start.

Parameters

Name	Range	Type	Default	Description
<timer>	152 to 8e6	Numeric	152	Programs the internal delayed trigger generator period in units of waveform points. Program the value using integers divisible by 8.

Response

The SE5082 will return the present internal delayed trigger period value in units of waveform points.

Example

Command `:TRIG:TIM:DEL 320`

Query `:TRIG:TIM:DEL?`

:TRIGger:TIMer:TIME<timer>(?)

Description

Use this command to set or query the period of the internal timed trigger generator. This value is associated with the internal trigger run mode only and has no effect on other trigger modes. The internal trigger generator is a free-running oscillator, asynchronous with the frequency of the output waveform. The timer intervals are measured from waveform start to waveform start.

Parameters

Name	Range	Type	Default	Description
<timer>	200e-9 to 20	Numeric	15e-6	Programs the internal timed trigger generator period in units of seconds.

Response

The SE5082 will return the present internal timed trigger period value in units of seconds.

Example

Command `:TRIG:TIM:TIME 100e-6`

Query `:TRIG:TIM:TIME?`

:TRIGger:TTL

Description

Use this command to set the trigger input to accept ttl signals. The threshold level is automatically set to 1.6 V, which is the mid-level for ttl logic. Other related commands are: *trig:ecl* to set the threshold level for ECL signals and *trig:lev* to program a threshold level between -5 V to 5 V.

Example

Command `:TRIG:TTL`

Analog Output Control Commands

The Analog Output Control Commands group is used for programming the characteristics of the output waveform. Notice that there are two main subsystems that control output functions: OUTPut and SOURce. The output subsystem commands control parameters that are related directly to the output terminals (main and sync outputs) and the source subsystem commands program parameters that control waveform shape, frequency and level.

The SE5082 has several output modules to choose from when ordering the instrument. There are 2 dc coupled output modules, DC and HV that differ in bandwidth, amplitude and offset, and an AC coupled Direct DAC module. The various modules differ in how the outputs behave in time and frequency domains. The DC module offers 3GHz bandwidth with 1.2Vpp amplitude. HV is optimized for best pulse response at higher amplitude levels, but at the cost of reduced bandwidth. The direct path is mainly used for RF applications where output bandwidth and flatness are optimized, but the control of dc offset bias is not available.

Other commands in this group control the SYNC parameters: type, position and width. Also use the commands in this group to control the shape of the output waveform, its frequency, its output level (or power) and the source of the clock reference.

Additional information on the various output functions and how the generator generates the different waveforms is given in Chapter 3. Factory defaults after *RST are shown in the default column. Parameter low and high limits are given where applicable. Use the commands in Table 4-4 to set up the SE5082 output waveforms, and their associated characteristics.

Table 4-14, Analog Output Commands Summary

Keyword	Parameter Form	Default	Notes
:OUTPut			
[:STATe]	OFF | ON | 0 | 1	0	Toggles output(s) on/off
:SYNC			
:FUNCtion	PULSe | WCOMplete	PULS	Selects sync out shape
:POSition			
[:POINt]	0 to 32e6-32 (0 to 64e6 with option -1, divided by 32)	0	Programs sync position
:WIDTh	32 to n-32 (divided by 32)	32	Sync width (Pulse only)
:SOURce	CH1 | CH2 | 1 | 2	CH1	Sync source
[:STATe]	OFF | ON | 0 | 1	0	Toggles sync on/off
:SAMPle			
:FORMat	NRZ | NRTZ | RTZ | RF	NRZ	
[:SOURce]			
:FREQuency			
[:CW]	1.0 to 2.5e9 | MINimum | MAXimum	10e6	Standard Sine Wave Freq.
	1.0 to 1.25e9 | MINimum | MAXimum	10e6	Standard Square Wave Freq
	1.0 to 300e6 | MINimum | MAXimum	10e6	Other Standard Waves Freq
:RASTer	10e6 to 6e9 | MINimum | MAXimum	1e9	Arb waveforms frequency
:FIX?			Query SCLK of std. wave
:PULSe?			Query SCLK of Pulse wave
:SOURce	INTernal | EXTernal	INT	Selects SCLK source
:DIVider	1 to 256 (integers, 2^n only)	1	External clock divider
:FUNCtion			
:MODE	FIXed | USER | SEQuence | ASEQuence | MODulation | PULSe | PATTern	FIX	Selects function type
:ROSCillator			
:SOURce	INTernal | EXTernal	INT	
[:EXTernal]			
:FREQuency	10M | 20M | 50M | 100M	100M	
:VOLTage			DC coupled output
[:LEVel]			
[:AMPLitude]	50e-3 to 2.0 MINimum | MAXimum	0.5	HV (HV option)
	100e-3 to 1.2 | MINimum | MAXimum	0.5	DC (DC option)
	0.4 to 0.54 | MINimum | MAXimum	0.54	Direct (DR option)
:OFFSet			
[:COMMon]	-1.0 to 1.0 | MINimum | MAXimum	0	HV (HV option)
	-0.75 to 0.75 | MINimum | MAXimum	0	DC(DC option)

:OUTPut{OFF|ON|0|1}(?)

Description

This command will set or query the output state of the SE5082. Note that for safety, the outputs always default to off, even if the last instrument setting before power down was on. Also note that the offsetting leaves the output connector connected to the amplifier path but no signal is being generated while in the off state.

Parameters

Range	Type	Default	Description
0-1	Discrete	0	Sets the output on and off

Response

The SE5082 will return 1 if the output is on, or 0 if the output is off.

Example

Command `:OUTP ON`

Query `:OUTP?`

:OUTPut:SYNC:FUNCtion{PULSe|WCOMplete}(?)

Description

Use this command to set or query the shape of the sync pulse. Pulse output can be programmed for position and width and the WCOM (wave complete) as fixed and cannot be moved from its origin.

Parameters

Name	Type	Default	Description
PULSe	Discrete	PULS	Selects the pulse shape as the output waveform for the sync output. The minimum pulse width is 16 sample clock periods. However, the width can be expanded to the full length of the waveform in increments of 16 points. Program the sync pulse width using the *outp:sync:wid* command and its relative position to the start of the waveform using the *outp:sync:pos* command.
WCOMplete	Discrete		This will select the waveform complete pulse option. The sync output will transition high at the beginning of the waveform and will return to low after the waveform cycle has been completed. Width and position control of the sync pulse is not available when this option is selected.

Response

The SE5082 will return PULS or WCOM, depending on the selected SYNC waveform function.

Example

Command `:OUTP:SYNC:FUNC WCOM`

Query `:OUTP:SYNC:FUNC?`

:OUTPut:SYNC:POSition<position>(?)

Description

This command will program the SE5082 SYNC position. This command is active in arbitrary (USER) mode only.

Parameters

Name	Range	Type	Default	Description
<position>	0 to 32e6-32	Numeric (Integer only)	0	Will set the SYNC position in waveform points. The sync position can be programmed in increments of 32 points to the maximum length of the waveform providing that the number is divisible by 32. The range is extended 64e6-32 when option-1 is installed; 32e6 memory size is standard.

Response

The SE5082 will return the present SYNC position value.

Example

Command `:OUTP:SYNC:POS 1024`

Query `:OUTP:SYNC:POS?`

:OUTPut:SYNC:WIDTh<width>(?)

Description

This command will program the SE5082 SYNC width. This command is active in arbitrary (USER) mode only.

Parameters

Name	Range	Type	Default	Description
<width>	32 to n-32	Numeric (Integer only)	32	Will set the SYNC width in waveform points. The sync width can be programmed in increments of 16 points minimum. “n” designates the length of the segment.

Response

The SE5082 will return the present SYNC width value.

Example

Command `:OUTP:SYNC:WIDT 2048`

Query `:OUTP:SYNC:WIDT?`

:OUTPut:SYNC:SOURce{CH1|CH2|1|2}(?)

Description

Use this command to set or query the source of the sync pulse. There is only one sync output connector on the front panel and this connector can be programmed to source from channel 1 or channel 2 waveforms. This command does not affect the single channel version. Channel 1 position and width and channel 2 position and width are programmed individually for each channel.

Parameters

Name	Type	Default	Description
CH1 | 1	Discrete	CH1	This will select channel 1 as the source for generating the sync pulse.
CH2 | 2	Discrete		Defines channel 2 as the source for generating the sync pulse.

Response

The SE5082 will return CH1 or CH2 depending on the selected SYNC source option.

Example

Command `:OUTP:SYNC:SOUR CH1`

Query `:OUTP:SYNC:SOUR?`

:OUTPut:SYNC{OFF|ON|0|1}(?)

Description

This command will set or query the state of the sync output. Note that for safety, the outputs always default to off, even if the last instrument setting before power down was on. Also note that the offsetting leaves the sync output connector connected to the amplifier path, but no signal is being generated while in the off state.

Parameters

Range	Type	Default	Description
0-1	Discrete	0	Sets the output on and off

Response

The SE5082 will return 1 if the sync output is on, or 0 if the sync output is off.

Example

Command `:OUTP:SYNC ON`

Query `:OUTP:SYNC?`

:OUTPut:SAMPle:FORMat{NRZ|NRTZ|RTZ|RF}(?)

Description

Use this command to set or query the DAC's sampling mode. It embeds different outputs modes (NRZ, RTZ, NRTZ, RF) that allows performances optimizations depending on the working Nyquist zone.

Parameters

Name	Type	Default	Description
<NRZ>	Discrete	NRZ	The default sampling mode of the DAC, Non Return to Zero, optimized for first Nyquist Zone.
<NRTZ>	Discrete		Narrow Return To Zero sampling mode, optimized for first & second Nyquist Zones
<RTZ>	Discrete		Return To Zero sampling mode, optimized for second Nyquist zone.
<RF>	Discrete		Radio Frequency sampling mode, optimized for second and third Nyquist zone.

Response

The SE5082 will return NRZ/NRTZ/RTZ/RF depending on the present sampling mode.

Example

Command `:OUTP:SAMP:FORM RF`

Query `:OUTP:SAMP:FORM?`

:FREQuency{<freq>|MINimum|MAXimum}(?)

Description

Use this command to set or query the frequency of the standard waveforms in units of hertz (Hz). This parameter has no effect on arbitrary waveforms.

Parameters

Name	Range	Type	Default	Description
<freq>	1 to 2.5e9	Numeric	10e6	Will set the frequency of the standard waveform in units of Hz. The frequency command can be used with resolutions up to 8 digits. Note that the maximum frequency varies depending on the selected waveform. For Sine waveform max frequency is 2.5GHz, for Square waveform max frequency is 1.25GHz and for all other waveform max frequency is 300MHz.
<MINimum>		Discrete		Will set the frequency of the standard waveform to the lowest possible frequency (10e3).
<MAXimum>		Discrete		Will set the frequency of the standard waveform to the highest possible frequency (2e9).

Response

The SE5082 will return the present frequency value. The returned value will be in standard scientific format (for example: 100mHz would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:FREQ 100e6`

Query `:FREQ?`

:FREQuency:RASTer{<sclk>|MINimum|MAXimum}(?)

Description

Use this command to set or query the sample clock frequency of the arbitrary waveform in units of samples per second (Sa/s). This parameter has no effect on standard waveforms.

Parameters

Name	Range	Type	Default	Description
<sclk>	10e6 to 6e9	Numeric	1e9	Will set the sample clock frequency of the arbitrary and sequenced waveform in units of Sa/s. The sample clock command can be programmed with resolutions up to 8 digits.
<MINimum>		Discrete		Will set the sample clock frequency to the lowest possible frequency (10e6).
<MAXimum>		Discrete		Will set the frequency of the standard waveform to the highest possible frequency (6e9).

Response

The SE5082 will return the present sample clock frequency value. The returned value will be in standard scientific format (for example: 1 GHz would be returned as 1e9 – positive numbers are unsigned).

Example

Command `:FREQ:RAST 2.5e9`

Query `:FREQ:RAST?`

:FREQuency:RASTer:FIX?

Description

Use this command to query the sample clock frequency of the standard waveform, when in Standard mode, in units of samples per second (Sa/s).

Response

The SE5082 will return the present sample clock frequency value. The returned value will be in standard scientific format (for example: 1 GHz would be returned as 1e9 – positive numbers are unsigned).

Example

Query `:FREQ:RAST:FIX?`

:FREQuency:RASTer:PULS?

Description

Use this command to query the sample clock frequency of the Pulse waveform, when in Pulse mode, in units of samples per second (Sa/s).

Response

The SE5082 will return the present sample clock frequency value. The returned value will be in standard scientific format (for example: 1 GHz would be returned as 1e9 – positive numbers are unsigned).

Example

Query `:FREQ:RAST:PULS?`

:FREQuency:RASTer:SOURce{INTernal|EXTernal}(?)

Description

Use this command to select or query the source of the sample clock generator. This command affects all of the waveforms, as the internal clock is removed and external clock is applied. Make sure that a valid clock is applied to the external clock input before you change the option to external, because the generator can not generate waveforms without a valid source of sample clock generator. Note that the internal sample clock generator is unique for each channel however, when an external clock source is selected, the same source is applied to both channels. Use the *freq:rast:sour:div* command for applications that require different sample clock frequencies per channel.

Parameters

Name	Type	Default	Description
INTernal	Discrete	INT	Selects the internal clock generator as the main clock source.
EXTernal	Discrete		Activates the external sample clock input. A valid signal must be applied from the external signal to the SE5082 for the generator to continue generating waveforms. Observe the input level and limitations before connecting an external signal to the external sample clock input.

Response

The SE5082 will return INT, or EXT depending on the current sample clock source setting.

Example

Command `:FREQ:RAST:SOUR EXT`

Query `:FREQ:RAST:SOUR?`

:FREQuency:RASTer:DIVider<divider>(?)

Description

Use this command to set or query the sample clock frequency divider. The same external sample clock signal is fed to both channels, so if different frequencies are required, use the sample clock divider to get different sample clocks to each channel.

Parameters

Name	Range	Type	Default	Description
<sclk>	1 to 64	Numeric (integer only)	1	Will set the external sample clock frequency divider. Except 1 (no divider), all divider values must be integers - (2^n). Each channel may be programmed separately to have a different divider value.

Response

The SE5082 will return the present external clock divider value in integer number.

Example

Command `:FREQ:RAST:DIV 4`

Query `:FREQ:RAST:DIV?`

:FUNCTion:MODE{FIXed|USER|SEQuence|ASEQuence|MODulation|PULSe|PATTern}(?)

Description

Use this command to set or query the type of waveform that will be available at the output connector.

Parameters

Name	Type	Default	Description
FIXed	Discrete	FIX	Selects the standard waveform shapes. There is an array of waveforms that is built into the program. You can find these waveform shapes in the standard waveforms section.
USER	Discrete		Selects the arbitrary waveform shapes. Arbitrary waveforms must be loaded to the SE5082 memory before they can be replayed. You can find information on arbitrary waveforms in the appropriate sections in this manual.
SEQuenced	Discrete		Selects the sequenced waveform output. To generate a sequence, you must first download waveform coordinates to different segments and then build a sequence table to generate a complex waveform that is using these segments.
ASEQuenced	Discrete		Selects the advanced sequencing waveform output. To generate an advanced sequences, you must first download waveform coordinates to different segments, use these waveforms to design sequences and then use

		these sequences to build an advanced sequence table where sequences are sequenced to create an extremely complex waveform.
MODulated	Discrete	Selects the modulated waveforms. There is an array of built-in modulation schemes. However, you can also build custom modulation schemes using the arbitrary function.
PULSe	Discrete	Selects the digital pulse function. The digital pulse function behaves and reacts to the programming sequence as a regular pulse generator, except the waveforms are digitally constructed and generated from the arbitrary memory.
PATTern	Discrete	Selects the pulse pattern function. The pulse pattern function behaves and reacts to the programming sequence as a regular pattern generator, except the waveforms are digitally constructed and generated from the arbitrary memory.

Response

The SE5082 will return FIX, USER, SEQ, ASEQ, MOD, PULS, or PATT depending on the present output function mode setting.

Example

Command `:FUNC:MODE USER`

Query `:FUNC:MODE?`

:ROSCillator:SOURce{INTernal|EXTernal}(?)

Description

Use this command to set or query the source of the 10 MHz reference. This source defines the accuracy and stability of the clock generator. The internal reference has an accuracy and stability of 1 ppm; applications requiring higher accuracy or stability can use the external reference option and apply there an improved 10 MHz signal.

Parameters

Name	Type	Default	Description
INTernal	Discrete	INT	Selects an internal source. The internal source is a TCXO (temperature compensated crystal oscillator) device that has 1ppm accuracy and stability over the operating temperature range.
EXTernal	Discrete		Reroutes the 10 MHz source to the external reference input. An external reference must be connected to the SE5082 for it to continue with its normal operation.

Response

The SE5082 will return INT, or EXT depending on the present 10 MHz clock reference source setting.

Example

Command `:ROSC:SOUR EXT`

Query `:ROSC:SOUR?`

:ROSCillator:EXTernal:FREQuency{10M|20M|50M|100M}(?)

Description

Use this command to set or query the frequency range that will be applied to the reference oscillator input. The frequency value must be close to the value of the external frequency, because it sets up the PLL's for the reference oscillator to accept and lock on the correct external frequency value.

Parameters

Name	Type	Default	Description
10M	Discrete	100M	Sets the the frequency of the external reference to 10MHz
20M	Discrete		Sets the the frequency of the external reference to 20MHz
50M	Discrete		Sets the the frequency of the external reference to 50MHz
100M	Discrete		Sets the the frequency of the external reference to 100MHz

Response

The SE5082 will return the present reference frequency value.

Example

Command `:ROSC:EXT:FREQ 10M`

Query `:ROSC:EXT:FREQ?`

:VOLTage{<amplitude>|MINimum|MAXimum}(?)

Description

Use this command to set or query the amplitude of the waveform. The SE5082 displays a calibrated value when on load impedance of 50 Ω. Offset and amplitude settings are independent providing that the |offset + amplitude/2| value does not exceed the specified voltage window. This command does not apply for AC output module.

Parameters

Name	Range	Type	Default	Description

<amplitude>	As per installed option. Refer to table 4-4a.	Numeric	500e-3	Will set the amplitude of the output waveform in units of volts. The display shows the correct amplitude level only when the output cable is terminated into 50Ω. The range varies depending on the output module installed.
<MINimum>		Discrete		Will set the amplitude to the lowest possible level.
MAXimum>		Discrete		Will set the amplitude to the highest possible level.

Response

The SE5082 will return the present DAC amplitude value. The returned value will be in standard scientific format (for example: 100 mV would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:VOLT 0.8`

Query `:VOLT?`

Table 4-4a, Output options summary table

Option	Amplitude	Offset	Window	Notes
Direct DAC (into 50 Ω) Range, single-ended	400mVpp to 540mVpp	N/A	N/A	AC coupled
Range, differential	800mVpp to 1080mVpp			
DC (into 50 Ω) Range, single-ended	100mVpp to 1.2Vpp	-0.5V to 0.5V	-0.75 V to 0.75 V	DC coupled
Range, differential	200mVpp to 2.4Vpp		-1.5 V to 1.5 V	
HV (into 50 Ω) Range, single-ended	50mVpp to 2Vpp	-1.0V to +1.0V	-2.0 V to 2.0 V	DC coupled
Range, differential	100mVpp to 4Vpp		-4.0 V to 4.0 V	

:VOLTage:OFFSet:[COMMon]{<offset>|MINimum|MAXimum}(?)

Description

Use this command to set or query the dc offset of the output waveform. The SE5082 displays a calibrated value when on load impedance of 50 Ω. Offset and amplitude settings are independent providing that the |offset + amplitude| value does not exceed the specified amplitude window. This command does not apply for AC output module.

Parameters

Name	Range	Type	Default	Description
<offset>	As per installed option. Refer to table 4-4a.	Numeric	0	Will set the offset of the output waveform in units of volts. The display shows the correct amplitude level only when the output cable is terminated into 50 Ω. The range varies depending on the output module installed.
<MINimum>		Discrete		Will set the offset to the lowest possible level.
<MAXimum>		Discrete		Will set the offset to the highest possible level.

Response

The SE5082 will return the present dc offset value. The returned value will be in standard scientific format (for example: 100 mV would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:VOLT:OFFS 0.5`

Query `:VOLT:OFFS?`

Marker Output Control Commands

The Marker Output Control Commands group is used for programming the characteristics of the marker outputs. Each channel has two differential marker outputs. These are located on the rear panel. Each marker can be programmed to have unique properties such as: delay, position, width and level. Simple front panel programming allows creation of one marker for each marker output however, since markers are programmed as bits D12 and D13 of the waveform data (see Figure 4-6), one can use this function to program arbitrary patterns on the marker outputs.

Additional information on the various marker output functions is given in Chapter 3. Factory defaults after *RST are shown in the default column. Parameter low and high limits are given where applicable. Use the commands in Table 4-5 to set up the SE5082 marker outputs and their associated parameters.

Tip

When markers are programmed from the front panel, the SE5082 generates one marker transition only for each marker output. The user has the freedom to turn markers on and off and modify marker attributes such as delay, position, width and levels. Internally, markers are programmed as part of the waveform and, when modified from the front panel, the firmware sets and remembers where markers were placed so every time that marker attributes are modified, the firmware knows where these attributes are stored and re-writes the entire waveform memory with the new attributes.

It is different when markers are programmed from remote; internal attributes are not visible to the external user and therefore, if you programmed marker attributes from the front panel and then did the same from remote, there is danger that marker transitions are present at unexpected locations. To avoid such errors, always set the mark:widt to 0 for all markers and only then download waveforms that have marker attributes on bits D12 and D13.

Table 4-15, Marker Output Control Commands Summary

Keyword	Parameter Form	Default	Notes
[:SOURce]			
:MARKer[1|2]			Selects active marker
:DELay	0 to 3e-9	0	Delay from SYNC
[:STATe]	OFF | ON | 0 | 1	0	Toggles marker on/off
:POSition	0 to n-4 (n = segment length)	0	Position from start, int divided by 4
:WIDTh	0 to n-4 (n = segment length)	64	Marker width int divided by 4
:VOLTage			
[:LEVel]			
:HIGH	0.5 to 1.25	1.0	Marker high level
:LOW	0 to 0.8	0	Marker low level

:MARKer[1|2]:DELay<delay>(?)

Description

Use this command to set or query the delay of the marker output. The delay is measured from the sync output in units of seconds. The marker has an initial delay of 0 sample clock periods, not including initial skew, and an amplitude swing of 0 to 1 V.

Parameters

Name	Range	Type	Default	Description
<delay>	0 to 3e-9	Numeric	0	Will set marker delay value in units of seconds. Each channel has two separate markers that can be programmed to have unique delays and amplitude levels. Note that you can program D12 and D13 to create multiple markers along the waveform length however, in this case, you must remove the default marker from the waveform map by setting the width parameter *mark:widt 0*.

Response

The SE5082 will return the present marker delay value. The returned value will be in standard scientific format (for example: 1 ns would be returned as 1e-9 – positive numbers are unsigned).

Example

Command `:MARK1:DEL 1e-9`

Query `:MARK1:DEL?`

:MARKer[1|2]{OFF|ON|0|1}(?)

Description

This command will set or query the state of the marker outputs. Note that for safety, the outputs always default to off, even if the last instrument setting before power down was on. The on/off setting affects both markers simultaneously on each channel.

Parameters

Range	Type	Default	Description
0-1	Discrete	0	Sets the marker outputs on and off

Response

The SE5082 will return 1 if the marker outputs are ON, or 0 if the marker outputs are OFF.

Example

Command `:MARK1 ON`

Query `:MARK1?`

:MARKer[1|2]:POSition<position>(?)

Description

Use this command to set or query the position of the marker output. The position is defined from the waveform first point in units of waveform points (sample clock periods). The marker has an initial position of 0 points and an amplitude swing of 0 to 1 V.

Parameters

Name	Range	Type	Default	Description
<position>	0 to n-4	Numeric	0	Will set marker position relative to the waveform start point in units of waveform points. The position range is from 0 to the last point of the waveform, minus 4. You can program the position with increments of 4 points. Note that you can program D12 and D13 to create multiple markers along the waveform length however, in this case, you must remove the default marker from the waveform map by setting the width parameter *mark:widt 0*.

Response

The SE5082 will return the present marker position value in units of waveform points.

Example

Command `:MARK1:POS 320`

Query `:MARK1:POS?`

:MARKer[1|2]:WIDTh<width>(?)

Description

Use this command to set or query the width of the marker output. The width is defined in units of waveform points (sample clock periods). The marker has an initial width of 64 points and an amplitude swing of 0 to 1 V.

Parameters

Name	Range	Type	Default	Description
<width>	0 to n-4	Numeric	64	Will set marker width in units of waveform points. The width range is from 0 to the last point of the waveform less 4. You can program the width in increments of 4 points. Note that you can program D14 and D15 to create multiple markers along the waveform length however, in this case, you must remove the default marker from the waveform map by setting the width parameter *mark:widt 0*.

Response

The SE5082 will return the present marker width value in units of waveform points.

Example

Command `:MARK1:WIDT 128`

Query `:MARK1:WIDT?`

:MARKer[1|2]:VOLTage:HIGH<hi_level>(?)

Description

Use this command to set or query the high level of the marker output. The high level is defined in units of volts. High and low levels can be programmed independently as long as the amplitude window remains within 1.25 V.

Parameters

Name	Range	Type	Default	Description
<hi_level>	0.5 to 1.25	Numeric	0.5	Will program the marker high level in units of volts. Each marker can be programmed to a different low- and high-level setting. The high level is calibrated when the cable is terminated into a 50 Ω load.

Response

The SE5082 will return the present marker high-level value. The returned value will be in standard scientific format (for example: 0.1 V would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:MARK1:VOLT:HIGH 0.75`

Query `:MARK1:VOLT:HIGH?`

:MARKer[1|2]:VOLTage:LOW<lo_level>(?)

Description

Use this command to set or query the low level of the marker output. The high level is defined in units of volts. High and low levels can be programmed independently as long as the amplitude window remains within 1.25 V.

Parameters

Name	Range	Type	Default	Description
<lo_level>	0 to 0.8	Numeric	0	Will program the marker low level in units of volts. Each marker can be programmed to a different low- and high- level setting. The low level is calibrated when the cable is terminated into a 50 Ω load.

Response

The SE5082 will return the present marker low-level value. The returned value will be in standard scientific format (for example: 0.1 V would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:MARK1:VOLT:LOW 0.1`

Query `:MARK1:VOLT:LOW?`

Standard Waveforms Control Commands

Use this group to control the shape and parameters of the standard waveforms functions. The commands in this group will affect the output only when the SE5082 has been programmed to generate standard waveforms. In standard waveform mode, some waveform coordinates are stored in tables and some are computed every time a waveform is being selected or modified. Expect small delays after the commands have been sent, because the waveform is recomputed and refreshed with every command.

Factory defaults after *RST are shown in the Default column. Parameter range and low and high limits are listed, where applicable. Use the commands in Table 4-6 to set up the SE5082 standard waveforms and their associated parameters.

Table 4-6, Standard Waveforms Control Commands Summary

Keyword	Parameter Form	Default	Notes
[:SOURce]			
:FUNCtion			
:SHAPe	SINusoid | TRIangle | SQUare | RAMP | SINC | GAUSsian | EXPonential | NOISe | DC	SIN	Standard function shape
:SINusoid			
:PHASe	-360.0 to 360.00 (degrees)	0	
:TRIangle			
:PHASe	-360.0 to 360.00 (degrees)	0	
:SQUare			
:DCYCle	0 to 99.99 (percent)	50	
:RAMP			
:DELay	0 to 99.999 (percent)	10	
:TRANsition			
[:LEADing]	0 to 99.999 (percent)	60.0	
:TRAiling	0 to 99.999 (percent)	30.0	
:SINC			
:NCYCle	4 to 100	10	
:GAUSsian			
:EXPonent	1 to 200	10	
:EXPonential			
:EXPonent	-100 to 100	-10	
:DC			
[:OFFSet]	-1.0 to 1.0	0	

:FUNCtion:SHAPe{SINusoid|TRIangle|SQUare|RAMP|SINC|GAUSsian|EXPonential|DC|NOISe}(?)

Description

Use this command to set or query the type of waveform that will be available at the output connector. This command will affect the SE5082 only when the standard waveforms output has been programmed. Select the standard waveforms using the *func:mode fix* command.

Parameters

Name	Type	Default	Description
SINusoid	Discrete	SIN	Selects the built-in sine waveform.
TRIangle	Discrete		Selects the built-in triangular waveform.
SQUare	Discrete		Selects the built-in square waveform.
RAMP	Discrete		Selects the built-in ramp waveform.
SINC	Discrete		Selects the built-in sinc waveform.
EXPonential	Discrete		Selects the built-in exponential waveform.
GAUSsian	Discrete		Selects the built-in gaussian waveform.
DC	Discrete		Selects the built-in DC waveform.
NOISe	Discrete		Selects the built-in noise waveform.

Response

The SE5082 will return SIN, TRI, SQU, SPUL, RAMP, SINC, GAUS, EXP, DC, or NOIS depending on the selected waveform setting.

Example

Command `:FUNC:SHAP TRI`

Query `:FUNC:SHAP?`

:SINusoid:PHASe<phase>(?)

Description

Use this command to set or query the start phase for the standard sine waveform.

Parameters

Name	Range	Type	Default	Description
<phase>	-360 to 360	Numeric	0	Programs the start phase parameter in units of degrees. Sine phase resolution is 0.01° limited however at high frequencies, depending on the number of waveform points that are used to create the sine shape.

Response

The SE5082 will return the present start phase value in units of degrees.

Example

Command `:SIN:PHAS 90`

Query `:SIN:PHAS?`

:TRIangle:PHASe<phase>(?)

Description

Use this command to set or query the start phase for the standard triangular waveform.

Parameters

Name	Range	Type	Default	Description
<phase>	-360 to 360	Numeric	0	Programs the start phase parameter in units of degrees. Triangle phase resolution is 0.01° limited however at high frequencies, depending on the number of waveform points that are used to create the shape.

Response

The SE5082 will return the present start phase value in units of degrees.

Example

Command `:TRI:PHAS 180`

Query `:TRI:PHAS?`

:SQUare:DCYCle<duty_cycle>(?)

Description

Use this command to set or query the duty cycle of the standard square waveform.

Parameters

Name	Range	Type	Default	Description
<duty_cycle>	0 to 99.99	Numeric	50	Programs the duty cycle of the standard square waveform in units of percent. Duty cycle setting resolution is limited to 0.01% of the square wave period.

Response

The SE5082 will return the present duty cycle value in units of percent.

Example

Command `:SQU:DCYC 75`

Query `:SQU:DCYC?`

:RAMP:DELay<delay>(?)

Description

Use this command to set or query the delay of the standard ramp waveform. The delay parameter defines the time that will lapse from the waveform start to the first transition of the ramp period. Please note that the (delay + leading-transition + trailing-transition) <= 100

Parameters

Name	Range	Type	Default	Description
<delay>	0 to 99.99	Numeric	10	Programs the ramp delay parameter in units of percent

Response

The SE5082 will return the present ramp delay value in units of percent.

Example

Command `:RAMP:DEL 50`

Query `:RAMP:DEL?`

:Ramp:TRANsition<rise>(?)

Description

Use this command to set or query the ramp transition time from low to high. The rise time parameter defines the time that will lapse from the first transition to high, until the ramp reaches its high-level value. Please note that the (delay + leading-transition + trailing-transition) <= 100

Parameters

Name	Range	Type	Default	Description
<rise>	0 to 99.99	Numeric	60	Programs the ramp rise time parameter in units of percent.

Response

The SE5082 will return the present rise time value in units of percent.

Example

Command `:RAMP:TRAN 20`

Query `:RAMP:TRAN?`

:RAMP:TRANsition:TRAiling<fall>(?)

Description

Use this command to set or query the ramp transition time from high to low. The fall time parameter defines the time that will lapse from the first transition to low, until the ramp reaches its low level value. Please note that the (delay + leading-transition + trailing-transition) <= 100

Parameters

Name	Range	Type	Default	Description
<fall>	0 to 99.99	Numeric	30	Programs the ramp fall time parameter in units of percent

Response

The SE5082 will return the present fall time value in units of percent.

Example

Command `:RAMP:TRAN:TRA 20`

Query `:RAMP:TRAN:TRA?`

:SINC:NCYCle<N_cycles>(?)

Description

Use this command to set or query the number of “0-crossings” of the standard SINC pulse waveform.

Parameters

Name	Range	Type	Default	Description
<N_cycle>	4 to 100	Numeric (Integer only)	10	Programs the number of zero-crossings parameter.

Response

The SE5082 will return an integer number depending on the current number of zero-crossing value.

Example

Command `:SINC:NCYC 8`

Query `:SINC:NCYC?`

:GAUSsian:EXPonent<exp>(?)

Description

Use this command to set or query the exponent for the standard Gaussian pulse waveform.

Parameters

Name	Range	Type	Default	Description
<exp>	1 to 200	Numeric (Integer only)	10	Programs the exponent parameter.

Response

The SE5082 will return an integer number depending on the present exponent value.

Example

Command `:GAUS:EXP 20`

Query `:GAUS:EXP?`

:EXPonential:EXPonent<exp>(?)

Description

Use this command to set or query the exponent for the standard exponential pulse waveform.

Parameters

Name	Range	Type	Default	Description
<exp>	-100 to 100	Numeric (Integer only)	-10	Programs the exponent parameter.

Response

The SE5082 will return an integer number, depending on the present exponent value.

Example

Command `:EXP:EXP 20`

Query `:EXP:EXP?`

:DC<offset>(?)

Description

Use this command to set or query the offset of the DC function in units of volts.

Parameters

Name	Range	Type	Default	Description
<offset>	-0.75 to 0.75	Numeric	0	Programs the DC offset parameter in units of volts. Please note limits change depending on amplifier option installed.

Response

The SE5082 will return the present DC offset value in units of volts.

Example

Command `:DC 0.3`

Query `:DC?`

Arbitrary Waveforms Control Commands

This group is used to control the arbitrary waveforms and their respective parameters. This will allow you to create segments and download waveforms. Using these commands, you can also define segment size and delete some or all unwanted waveforms from your memory.

Generating Arbitrary Waveforms

Arbitrary waveforms are generated from digital data points, which are stored in a dedicated waveform memory. Each data point has a vertical resolution of 12 bits (4,096 points), i.e., each sample is placed on the vertical axis with a precision of 1/4,096. The SE5082 has the following waveform memory capacity:

32M – standard memory option

64M – optional memory expansion

Each horizontal point has a unique address - the first being 00000 and the last depending on the memory option. In cases where smaller waveform lengths are required, the waveform memory can be divided into smaller segments.

When the instrument is programmed to output arbitrary waveforms, the clock samples the data points (one at a time) from address 0 to the last address. The rate at which each sample is replayed is defined by the sample clock rate parameter.

Unlike the built-in standard waveforms, arbitrary waveforms must first be loaded into the instrument's memory. Correct memory management is required for best utilization of the arbitrary memory. An explanation of how to manage the arbitrary waveform memory is given in the following paragraphs.

Arbitrary Memory Management

The arbitrary memory in comprised of a finite length of words. The maximum size arbitrary waveform that can be loaded into memory depends on the option that is installed in your instrument. The various options are listed in Chapter 1 of this manual. If you purchased the SE5082 within its basic configuration, you should expect to have 32 Meg words in each channel to load waveforms.

Waveforms are created using small sections of the arbitrary memory. The memory can be partitioned into smaller segments (up to 32k) and different waveforms can be loaded into each segment, each having a unique length. Minimum segment size is 384 points and can be increased by increments of 32 points. Information on how to partition the memory, define segment length and download waveform data to the SE5082 is given in the following paragraphs. The arbitrary waveform commands are listed in Table 4-7. Factory defaults after *RST are shown in the Default column. Parameter range and low and high limits are listed, where applicable.

Table 4-7, Arbitrary Waveforms Commands Summary

Keyword	Parameter Form	Default	Notes
:TRACe			
[:DATA]	#<header><binary_block>		Write waveform binary data to the active-segment of the active channel
:DEFine	<segment-number>,<segment-length> <1 to 32,000><384 to Arbitrary-Memory-Size >		Define the length (divided by 32) of the segment.
:DEFine<n>?	<n = 1 to 32,000>		Query length of seg <n>
:DELete			
[:NAME]	1 to 32,000		Delete one segment
:ALL			Delete all segments
:POINts?			Queries active segment waveform length
:SELect	<segment number> between 1 and 32000	1	Select the active segment (of the active channel)
:SOURce	BUS | EXTernal	BUS	Toggle control source
:TIMing	COHerent | IMMediate	IMM	Select timing
:SEGMent			
[:DATA]	#<header><data_array>		Write the segment-table of the active channel

:TRACe#<header><binary_block>

Description

This command will download waveform data to the SE5082 waveform memory. Waveform data is loaded to the SE5082 using high-speed binary data transfer. High-speed binary data transfer allows any 8-bit bytes to be transmitted in a message. This command is particularly useful for sending large quantities of data. As an example, the next command will download to the generator an arbitrary block of data of 1,024 points

```
TRACe#42048<binary_block>
```

This command causes the transfer of 2,048 bytes of data (1,024 waveform points) into the active memory segment. The <header> is interpreted this way:

- The ASCII "#" ($23) designates the start of the binary data block.
- "4" designates the number of digits that follow representing the binary data block size in bytes.
- "2,048" is the number of bytes to follow.

The generator accepts waveform samples as 16-bit integers, which are sent in two-byte words. Therefore, the total number of bytes is always twice the number of data points in the waveform. For example, 20000 bytes are required to download a waveform with 10,000 points. The IEEE-STD-488.2 definition of Definite Length Arbitrary Block Data format is demonstrated in Figure 4-4.

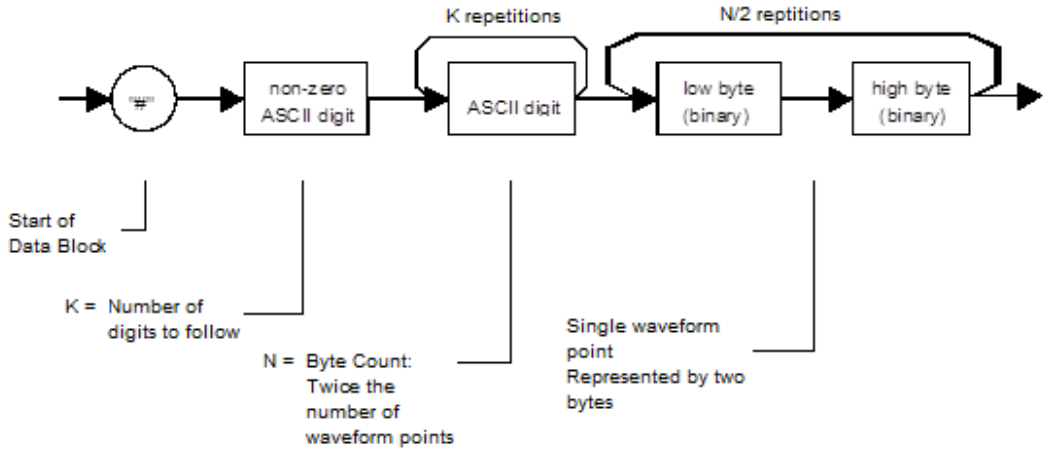

Figure 4-1, Definite Length Arbitrary Block Data Format

- <binary_block> Represents waveform data.

The waveform data is made of 16-bit words however, programmers may choose to prepare the data in two bytes and arrange to download these two bytes in a sequence (low byte followed by high byte). Figure 4-5 shows a waveform word that is acceptable for the SE5082. There are a number of points you should be aware of before you start preparing the data:

1. Waveform data points have 12-bit values - 0x000 to 0xFFF.
2. Data point range is 0 to 4,095 decimal for the SE5082. 0x000 corresponds to -Amplitude/2 +offset and 0xFFF corresponds to Amplitude/2 +offset. Point 2,047 corresponds to offset setting.

Figure 4-5 shows how to prepare the 16-bit word for a data point representation and Figure 4-6 shows how this data represents waveform and marker data and stop bit control.

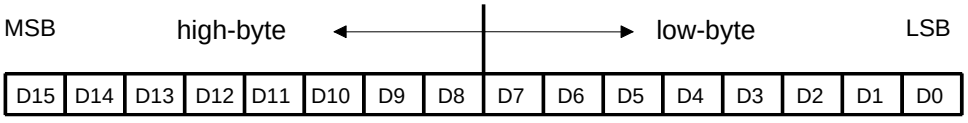

Figure 4-2, 16-bit Data Point Representation

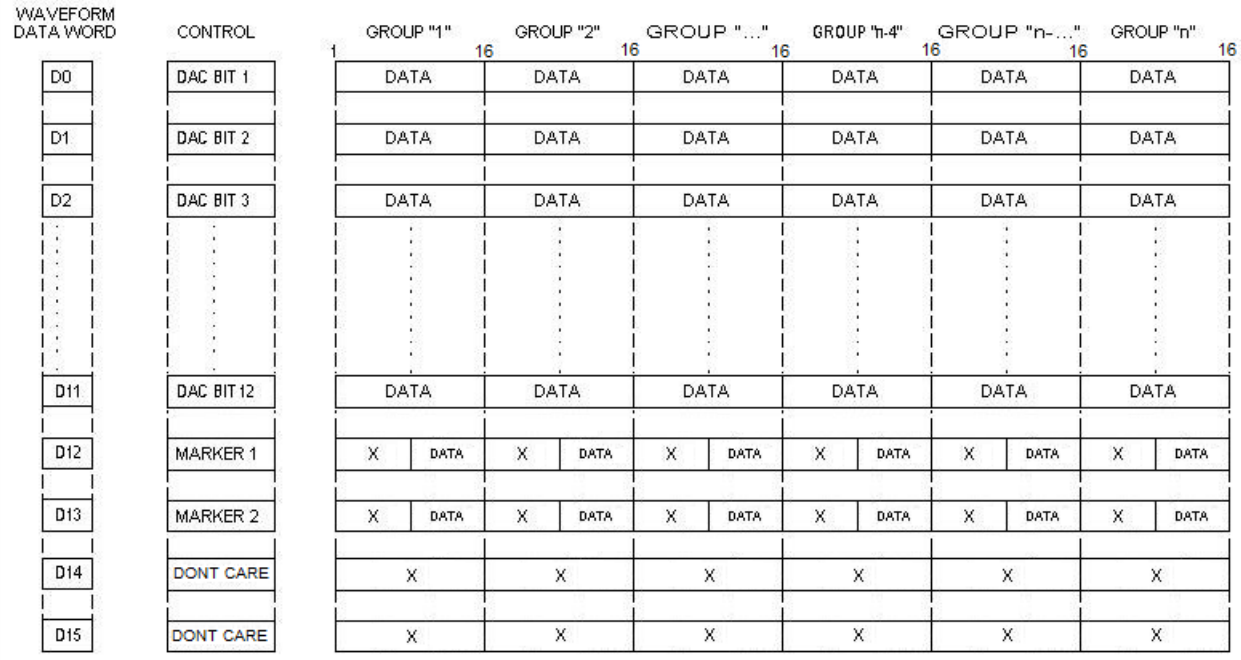

Figure 4-3, Waveform Data Word Representation

Notes:

1. Each group of data contains 32 words and therefore, waveform data is programmed in multiples of 16 words.
2. "X" data in D12 and D13 implies that the first 24 words you can place are "don't care" data but the last 8 words contain marker position data. You may place markers at any position along the waveform, but the data can be programmed in the last 8 words of a group of 16 words. Note: Marker resolution is 4 sample clock cycles. Marker position is programmed in groups of 4 waveform points, therefore programming "1" at the last 8 words in D12 and D13 will automatically set markers 1 and 2 to "1" over 32 waveform points.
3. Use the following programming example to position the markers:

Conditions:

1. mark_pos%4 = 0; mark_width%4 = 0;

2. mark_pos+mark_width < wave length

Calculations:

marker_points = mark_width/4;

```
for(ii=0; ii<marker_points; ii++)
{
wave_index = 24*((integer)(mark_pos/32) + 1) + mark_pos/4;
mark_pos = mark_pos+4;
}
```

Parameters

Name	Type	Description
<header>	Discrete	Contains information on the size of the binary block that contains waveform coordinates.
<binary_block>	Binary	Block of binary data that contains waveform data points (vertical coordinates), as explained above.

Example

Command `:TRAC#42048<binary_block>`

:TRACe:DEFine<segment_#>,<length>

Description

Use this command to define the size of a specific memory segment. The final size of the arbitrary memory is 16,000,000 points (32,000,000 points or 64,000,000 points optional). The memory can be partitioned to smaller segments, up to 32,000 segments. The total length of memory segments cannot exceed the size of the waveform memory.

 ***NOTE***

Segment size can be programmed in numbers divisible by 32 only. For example, 2112 bytes is an acceptable length for a binary block. 2110 is not a multiple of 32 and therefore the generator will generate an error message if this segment length is used.

Parameters

Name	Range	Type	Default	Description
<segment_#>	1 to 32k	Numeric (integer only)	1	Selects the segment number of which will be programmed using this command.
<length>	384 to n	Numeric (integer only)		Programs the size of the selected segment. Minimum segment length is 384 points. The maximum (n) is limited by the size of the installed memory. Segment size can be programmed with increments of 32 points.

Example

Command `:TRAC:DEF 1, 1024`

:TRACe:DELete<segment_#>

Description

This command will delete a predefined segment from the working memory. The memory space that is being freed will be available for new waveforms, as long as the new waveform will be equal or smaller in size to the deleted segment. If the deleted segment is the last segment, then the size of another waveform written to the same segment is not limited. For example, let's consider two segments: the first being a 1024-point waveform and the second with 640 points. If you delete segment 1, you can reprogram another waveform to segment 1 with size to 1024 points. If you reprogram segment 1 with more than 1024 points, the instrument will generate an error and will not accept this waveform. On the other hand, if you delete segment 2, which was the last segment you programmed, you can reprogram this segment with waveforms having length limited only by the size of the remaining memory space.

Parameters

Name	Range	Type	Default	Description
<segment_#>	1 to 32k	Numeric (integer only)	1	Selects the segment number of which will be deleted

Example

Command `:TRAC:DEL 1`

:TRACe:DELete:ALL

Description

This command will delete all predefined segments and will clear the entire waveform memory space. This command is particularly important in case you want to defragment the entire waveform memory and start building your waveform segments from scratch.

Example

Command `:TRAC:DEL:ALL`

:TRACe:POINts?

Description

This command will query the number of points that were used for building up the active waveform.

Response

The SE5082 will return the active waveform length as an integer number.

Example

Query `:TRAC:POIN?`

:TRACe:SELect<segment_#>(?)

Description

Use this command to set or query the active waveform segment at the output connector. By selecting the active segment, you are performing two functions:

1. Successive: TRAC commands will affect the selected segment.
2. The SYNC output will be assigned to the selected segment. This behavior is especially important for sequence operation, where multiple segments form a large sequence. In this case, you can synchronize external devices exactly to the segment of interest.

Parameters

Name	Range	Type	Default	Description
<segment_#>	1 to 32k	Numeric (Integer only)	1	Selects the active segment number.

Response

The SE5082 will return the active waveform segment number.

Example

Command `:TRAC:SEL 4`

Query `:TRAC:SEL?`

:TRACe:SELect:SOURce{BUS|EXTernal}(?)

Description

Use this command to set or query the source of the segment select command. This defines from where the select command is expected to be received, causing a waveform segment change. Using the BUS option, waveforms can be selected using remote commands only. The EXT option transfers the control to a rear panel connector that allows dynamic selection of the active waveform segment. The rear panel connector has 9 pins of which 8 are used for parallel control bits and one line is used for data validation. Using the external waveform control, one can dynamically select a waveform from a preprogrammed list of 256 waveforms. The transition characteristics from waveform segment to another is programmed using the *trac:sel:tim* command.

Parameters

Name	Type	Default	Description
BUS	Discrete	BUS	Defines that waveform segments will be switched only when a remote command has been received.
EXTernal	Discrete		Defines that the segment control is transferred to a rear panel connector. The connector has 8 bits of parallel control lines that can switch between up to 256 segments.

Response

The SE5082 will return BUS, or EXT depending on the present segment jump setting.

Example

Command `:TRAC:SEL:SOUR BUS`

Query `:TRAC:SEL:SOUR?`

:TRACe:SELect:TIMing{COHerent|IMMediate}(?)

Description

Use this command to set or query the timing characteristics of the trace select command. This defines how the generator transitions from waveform to waveform. Use the coherent option to let the waveform complete before it jumps to the next waveform. Applications that require an unconditional jump can use the immediate option, where the generation of the current waveform is aborted and the new waveform is started immediately thereafter. This command affects the segment transition timing, regardless if the segment control is from remote or from the rear panel connector.

Parameters

Name	Type	Default	Description
COHerent	Discrete		Defines that when a new waveform segment is selected, the transition to the new waveform will occur only when the current waveform has reached its end point.
IMMediate	Discrete	IMM	Defines that when a new waveform segment is selected, the current waveform will be aborted and the transition to the new waveform will occur immediately, without waiting for the current waveform to reach its end point.

Response

The SE5082 will return COH, or IMM depending on the present segment jump timing setting.

Example

Command `:TRAC:SEL:TIM IMM`

Query `:TRAC:SEL:TIM?`

:SEGMent#<header><data_array>

Description

This command will partition the waveform memory to smaller segments hence speeding up memory segmentation. The idea is that waveform segments can be built as one long waveform and then just use this command to split the memory to the appropriate memory segments. In this way, there is no need to define

and download waveforms to individual segments.

Using this command, segment table data is loaded to the SE5082 using high-speed binary transfer in a similar way to downloading waveform data with the trace command. High-speed binary transfer allows any 8-bit bytes to be transmitted in a message. This command is particularly useful for large number of segment. As an example, the next command will generate three segments with 12 bytes of data that contains segment size information.

SEGMent#212<data_array>

This command causes the transfer of 12 bytes of data (3 segments) into the segment table buffer. The <header> is interpreted this way:

- The ASCII "#" ($23) designates the start of the binary data block.
- "2" designates the number of digits that follow.
- "12" is the number of bytes to follow. This number must divide by 4.

The data array contains a list of the segments lengths where each segment length is represented using 4 bytes. Therefore, the total number of bytes is always 4 times the number of segments. For example, 36 bytes are required to download 9 segments to the segment table. The IEEE-STD-488.2 definition of Definite Length Arbitrary Block Data format is demonstrated in Figure 4-4.

Figure 4-7 shows how to prepare the 32-bit data aligned for the segment table buffer. Only segment size is required because the segments are automatically numbered and their relative start address is automatically placed at the end of the last memory segment.

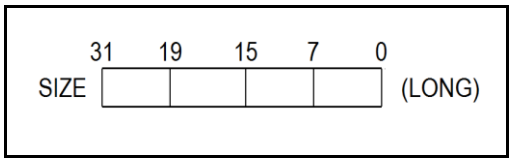

Figure 4-4, Segment Size Array Entry Example

There are a number of points you should be aware of before you start preparing the data:

1. Each channel has its own segment table buffer. Therefore, make sure you selected the correct active channel (with the INST:SEL command) before you download segment table data to the generator.
2. SEGM# command overrides segment definitions, similar to the TRAC:DEL:ALL command.
3. Minimum number of segments is 1; maximum number of segments is 32,000
4. Maximum segment size depends on your installed option. With the basic SE5082 you can program maximum 32M in one segment.
5. Segment table data has 32-bit values which are used for segment size. Therefore, Data for each segment must have 4 bytes.
6. The number of bytes in a complete segment table must divide by 4. The SE5082 has no control over data sent to its segment table during data transfer. Therefore, wrong data and/or incorrect number of bytes will cause erroneous memory partition and will require to perform a reboot.
7. The sample program below demonstrates the usage of the SEGM# command. This example generates a segment table with four entries: Segment 1 = 640 points, segment 2 = 1,024 points, segment 3 = 2,048 and segment 4 = 20,480 points. Regardless if you previously had any segments defined previously, the segment index will start from 1 to n.

   UINT_32 seg_data_inf[4], total_buff_bytes_num;

```
seg_data_inf[0] = 640;
seg_data_inf[1] = 1024;
seg_data_inf[2] = 2048;
seg_data_inf[3] = 20480;
total_buff_bytes_num = 4 * num_of_segments;
digits_num = (UINT_32)(log10((double) total_buff_bytes_num) +1.0);
sprintf(cmd_str,":segm #%d%d", digits_num, total_buff_bytes_num);
sendCmdNoNL (cmd_str);
h_vi.dwlData((unsigned char *)seg_data_inf, total_buff_bytes_num);
```

8. SEGM# command is extremely useful in case you need to define and download waveforms to a large number of segments otherwise, you would be using the TRAC:DEF command to define individual segments. In this case, you can further save programming sequences and download time by combining and downloading all of your waveform segments in one download command. Since segment definition does not modify memory contents, you can either download the waveform and then slice the memory segments or reverse the procedure; Either way, the process of downloading one waveform and then define the entire segment table with one command makes this process extremely efficient.
9. Unlike downloading individual waveforms to individual segments, one has to prepare a combined waveform for this one-shot download of multiple waveform segments. Information and considerations are provided in the trac#.

Consider the three segments as shown in the next figure, marked as segment 1, 4000 points long, segment 2, 3200 points long and segment 3, 5600 points long. You can define each segment separately using the trac:def command and then, download waveform data to each defined memory segment.

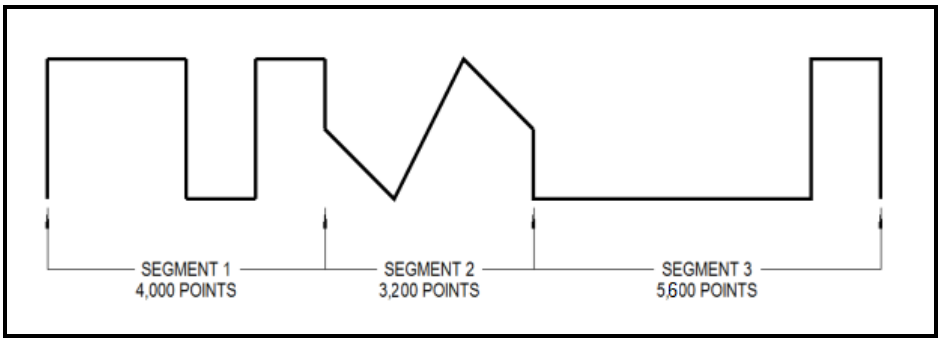

Figure 4-5, Multiple Segment Download Array Example

Now consider that you intend to use the SEGM# command and therefore you'd like to combine the three waveform segments into one single waveform. The following figure demonstrates how the new waveform looks like:

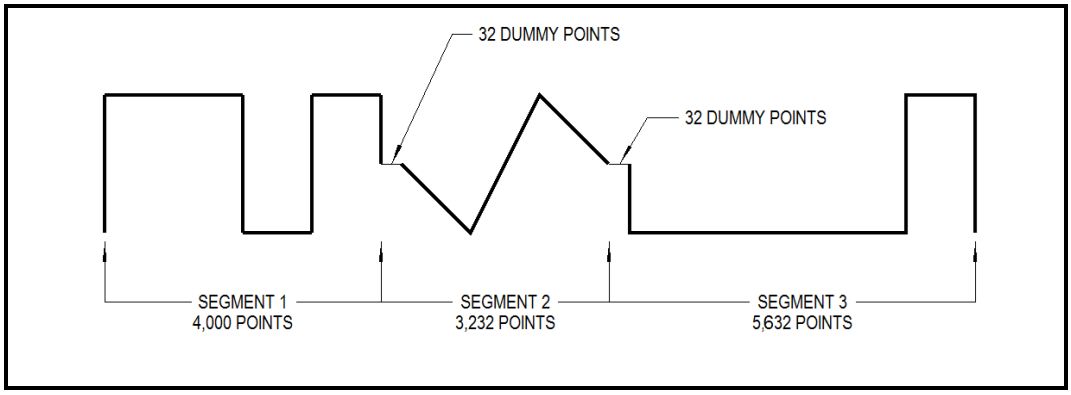

Figure 4-6, Single Segment Download Array Example

Notice that besides the first waveform, each of the following waveforms is artificially expanded by 32 dummy points right at the beginning of the segment. The value of these points should be the same as the value of the first point of the waveform segment. So now, the combined length of the three waveform segments is 4,000+3,200+5,600+64 = 12,864 points. And the header for the data download command should look as follows:

TRACe#525728<binary_block>

Where 5 defines that 5 digits will follow and 24128 is the number of bytes that will be downloaded using the binary download process. Explanation on the TRACe# command is given in the Arbitrary Waveform Commands section of this manual.

Parameters

Name	Type	Description
<data_array>	Binary	Block of binary data that contains information on the segment table.

Example

Command `:SEGM#212<data_array>`

Sequenced Waveforms Control Commands

This group is used to control the sequenced waveforms and their respective parameters. This will allow you to create multiple sequence tables and modify segment loops and links. Also, use these commands to add or delete sequences from your instrument.

Generating Sequenced Waveforms

Sequenced waveforms are made of a number of arbitrary waveforms, which can be linked and looped in user-programmable order. Sequenced waveforms are generated from waveforms stored in the SE5082 as memory segments. Therefore, before a sequence can be used, download waveform segments to the arbitrary memory. Information on how to partition the memory and how to download waveforms is given in this chapter

An example of how sequenced waveforms work is demonstrated in figures 1-8 through 1-10. The sequence generator lets you link and loop segments in user-defined order. Figure 1-11 shows a sequence of waveforms that were stored in three different memory segments.

In general, sequences can be built one step at a time, using the *seq:def* command. The one-step method is slow and tedious. However, it allows better control for one who just begins his first sequence programming. Advanced users can download a complete sequence table using the binary sequence download option. The latter is much faster for applications requiring large sequence tables. Use the information below to understand sequence commands and how to implement them in your application.

The sequenced waveforms commands are listed in Table 4-8. Factory defaults after *RST are shown in the Default column. Parameter range and low- and high-limits are listed, where applicable.

Table 4-8, Sequence Control Commands

Keyword	Parameter Form	Default	Notes
[:SOURce]			
:SEQuence			
:TYPE	NORMal | ADVanced		Select the active sequencer type
:ADVance	AUTOmatic | ONCE | STEPped	AUTO	Sequence advancing mode
[:DATA]	#<header><binary_block>		Write the data of the active sequence.
:DEFine	<step_no>,<segment_no>,<loops>,<jump_flag> <1 to 49152><1 to 32000><1 to 16777215><0 or 1>.		Define the n'th step of the active sequence. Note that length (in steps) of each sequence is at least 3, and the lengths sum of all sequences must not exceed 49152.
:DELete			
[:NAME]	1 to 1,000		Delete the specified sequence.
:ALL			Delete all sequences.
:JUMP			
[:EVENt]	BUS | EVENt	BUS	Toggle jump source
:LENGth	Query only		Query the length (in steps) of the active sequence.
:SELect	1 to 1,000		Select the active sequence.
:SOURce	BUS | EXTernal	BUS	Toggle control source
:TIMing	COHerent | IMMediate	COH	Jump timing
:PREStep	WAVE | DC	WAVE	DC is active in continuous and BUS source only
:ONCe			
:COUNt	1 to 16,777,216		The number of times the active sequence will be repeated
:SYNC			
:LOCK	<step_number>	1	Sync position
:RESet			Force the sequence to its first step

:SEQuence:TYPE<NORMal|ADVanced>(?)

Description

Use this command to select between normal sequence type and the advanced sequence type where sequences, and not segments, are sequenced. By default the when selecting sequence mode the sequence type is normal. The advance sequence type has its own set of commands as detailed in the next section.

Parameters

Name	Type	Default	Description
NORMAL	Discrete	NORMAL	This is the default option when working in sequence mode where segments are sequenced according to predefined sequence tables.
ADVANCED	Discrete		This is a special incidence of the sequence generator where sequences, and not segments, are sequenced.

Response

The SE5082 will return NORMAL or ADVANCED depending on the current sequence type setting.

Example

Command `:SEQ:TYPE ADV`

Query `:SEQ:TYP?`

:SEQuence:ADVance{AUTOmatic|ONCE|STEPped}(?)

Description

This command will select the sequence advance mode. It defines how the output advances through the sequence steps. There are three advance modes: automatic, once and stepped.

In automatic advance mode, the routine goes through the steps automatically and if there are no jump flag, the sequence will end and then start over automatically. If a loop counter other than 1 is programmed for a specific step in the sequence, the step will loop n times and then automatically advance to the next step. The jump flag inhibits the progression to the next step until a valid signal at the event input releases the step to jump.

In once advance mode, the routine goes through the steps automatically and if there are no jump flags, the sequence will end and idle on a specific waveform, depending on the selected run mode. If the once counter is programmed to a value other than 1, the sequence will repeat itself n times. If a loop counter other than 1 is programmed for a specific step in the sequence, the step will loop n times and then automatically advance to the next step. The jump flag inhibits the progression to the next step, until a valid signal at the event input releases the step to jump.

In stepped advance mode, the routine goes through the steps only after a valid event signal. When the sequence is complete, the sequence repeats itself with valid event signals. Jump flags are ignored in stepped mode, but a loop counter other than 1 will repeat the step for n times with each event, before advancing to the next step.

Parameters

Name	Type	Default	Description
AUTOmatic	Discrete	AUTO	Specifies continuous advance where the generator steps continuously to the end of the sequence table and repeats the sequence from the start. For example, if a sequence is made of three segments 1, 2 and 3, the sequence will generate an infinite number of 1,2,3,1,2,3,1,2,3…waveforms. Of course, each link (segment) can be programmed with its associated loop (repeat) number and jump flag to inhibit advancement until an event signal has been received.
ONCE	Discrete		This selects the once sequence advance mode, where the generator steps through the sequence table once automatically, except if the *seq:once:coun*

		programmed a value greater than 1.
STEPped	Discrete	Specifies the stepped sequence advance mode, where the generator steps to the next waveform only when a valid event signal has been received.

Response

The SE5082 will return the AUTO, ONCE, or STEP depending on the present sequence advance mode setting.

Example

Command `:SEQ:ADV ONCE`

Query `:SEQ:ADV?`

:SEQuence#<header><data_array>

Description

This command will build a complete sequence table in one binary download. In this way, there is no need to define and download individual sequencer steps. Using this command, sequence table data is loaded to the SE5082 using high-speed binary transfer in a similar way to downloading waveform data with the trace command. High-speed binary transfer allows any 8-bit bytes to be transmitted in a message. This command is particularly useful for long sequences that use a large number of segments and sequence steps. As an example, the next command will generate three-step sequence with 24 bytes of data that contains segment number, loops and jump flag option.

```
SEQuence#224<data_array>
```

This command causes the transfer of 24 bytes of data (3-step sequence) to the sequence table buffer. The <header> is interpreted this way:

- The ASCII "#" ($23) designates the start of the binary data block.
- "2" designates the number of digits that follow.
- "24" is the number of bytes to follow. This number must divide by 8.

The data array contains a list of sequence table entries where each entry is represented by 8 bytes. Therefore, the total number of bytes is always eight times the number of sequence entries. For example, 24 bytes are required to download 3 sequence entries to the sequence table.

The code below demonstrates the data structure and the below figure shows how to prepare the 64-bit entry for the sequence step, number of loops and jump flag option.

```
typedef struct
{
UINT_32 loops;
  UINT_16 segment_number;
  UINT_16   jump_flag;
}
seqTableEntry_t;
```

There are a number of points you should be aware of before you start preparing the data:

1. Each channel has its own sequence table buffer. Therefore, make sure you select the correct active channel (with the INST:SEL command) before you download sequence table data to the generator
2. Minimum number of sequencer steps is 3; maximum number is 49,152
3. The number of bytes in a complete sequence table must divide by 8. The Model SE5082 has no control over data sent to its sequence table during data transfer. Therefore, wrong data and/or incorrect number of bytes will cause erroneous sequence partition
4. Step numbers are assigned automatically in the same order as the structure is built. The first 8-byte structure forms step number 1 and the last, form step number n. As an example, for 100 steps sequence, one should build an array of 100 seqTableEntry_t entries.

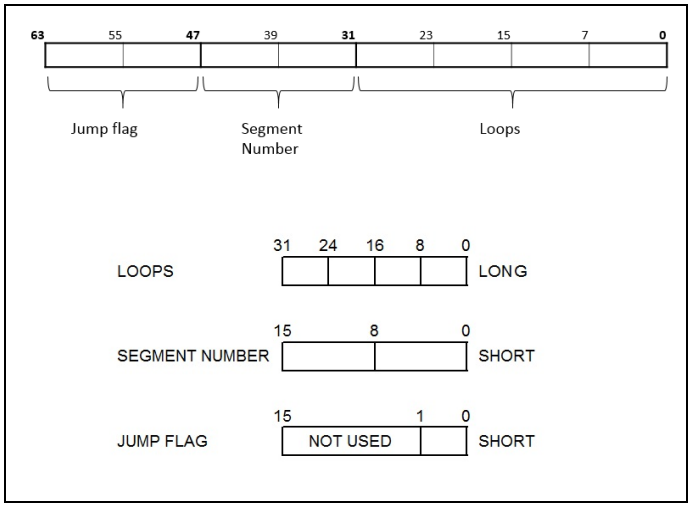

Figure 4-7, 64-bit Sequence Table Entry Format

Parameters

Name	Type	Default	Description
<header>	Discrete		Sequence table header, defines the length of the binary <data_array>.
<data_array>	Binary		Block of binary data that contains information on the sequence table.

Example

Command `:SEQ#224<data_array>`

:SEQuence:DEFine<step>,<segment_#>,<loops>,<jump_flag>(?)

Description

Use this command to create a sequence table. It defines all of the parameters that are associated with the sequence step such as: step number, segment number, loops and jump flag. Each step in a sequence table must be programmed separately, except if the seq:data command is used to download a complete table in binary code. The seq:leng command can be used to first predefine the length of the sequence table. However, the length is adjusted automatically to the number of entries when the seq:def commands program each step.

Parameters

Name	Range	Type	Description
<step>	1 to 49,152	Numeric (integer only)	Programs the step in the sequence table. Steps are indexed from 1 to 49,152 and must be programmed in an ascending order. Empty step locations in a sequence table are not permitted. Minimum number of steps required to create a sequence is 3.
<segment_#>	1 to 32,000	Numeric (integer only)	Assigns a waveform segment to a specific step number. When encountered in the sequence table, the waveform that is associated with the step will be generated.
<loops>	1 to 1M	Numeric (integer only)	Programs the repeat number of loops that a specific step will play, before advancing to the next step in the sequence.
<jump_flag>	0-1	Boolean	This flag specifies if the sequence will advance or hold on a specific step. “0” flag specifies normal advance, “1” flag specifies that the output will dwell on the current step and will advance to the next step only after a valid event has been received.

Example

Command `:SEQ:DEF 1,5,100,0`

TIP

The SE5082 attempts to rebuild the sequence table and restart the sequence every time you use the *seq:def* command, while your generator is in sequenced operating mode. Therefore, sending this command in sequenced mode will slow the programming process and the operation of the generator. Using the *seq:def* command in FIX or USER mode will greatly speed up programming time. Sequencer steps are programmed from 1 to 49,152. However, minimum number of steps to create a sequence is 3.

:SEQuence:DELete<sequence_#>

Description

The SE5082 can store up to 1,000 different sequences for each channel. Use this command to delete unwanted sequences and to free memory space for fresh sequence tables. You can remove individual sequences regardless of their indexed order. For example, you can delete sequences 4, 7 and 15. Restoring deleted sequences is not a problem except their length must be equal or less than the deleted size. If you are not sure what to do, use the seq:del:all command and reprogram the entire sequence list.

Parameters

Name	Range	Type	Default	Description
<sequence_#>	1 to 1,000	Numeric (integer only)	1	Selects a specific sequence name that will be deleted.

Example

Command :SEQ:DEL 3

:SEQuence:DELete:ALL

Description

Use this command to delete all sequences and to free space for fresh sequence tables.

Example

Command :SEQ:DEL ALL

:SEQuence:JUMP{BUS|EVENt}(?)

Description

Use this command to set or query the source of the jump signal. The jump signal is required for Auto and Stepped advance sequence modes. In Auto advance mode, the sequence will advance through the steps automatically until a jump bit = “1” is encountered, which will cause the generator to dwell on the step until a valid jump signal is asserted. In Stepped mode, the jump signal is required for every step to advance to the next waveform in the sequence ladder.

Parameters

Name	Type	Default	Description
BUS	Discrete	BUS	Defines that the sequence will advance to the next step only when a remote trigger command such as *trg has been received.
EVENt	Discrete		Defines that the sequence will advance to the next step only when a valid signal has been asserted to the Event input.

Response

The SE5082 will return BUS, or EVEN depending on the present sequence jump setting.

Example

Command :SEQ:JUMP EVEN

Query :SEQ:JUMP?

:SEQuence:LENGth?

Description

Use this command to query the number of steps programmed in the active sequence table.

Response

The SE5082 will return the present length value of the active sequence.

Example

Query `:SEQ:LENG?`

:SEQuence:SELect<sequence_#>(?)

Description

Use this command to query or select an active sequence number to be generated at the output connector. Each channel can store up to 1,000 different sequence tables. By selecting the active sequence, successive *seq:def* commands will affect the selected sequence only.

Parameters

Name	Range	Type	Default	Description
<sequence_#>	1 to 1,000	Discrete (Integer only)	1	Selects an active sequence number. Subsequent programming will affect the active sequence only.

Response

The SE5082 will return the active sequence number.

Example

Command `:SEQ:SEL 10`

Query `:SEQ:SEL?`

:SEQuence:SELect:SOURce{BUS|EXTernal}(?)

Description

Use this command to set or query the source of the sequence select command. This defines from where the select command is expected to be received, causing a sequence change. Using the BUS option, sequences can be selected using remote commands only. The EXT option transfers the control to a rear panel connector that allows dynamic selection of the active sequence. The rear panel connector has 9 pins, of which 8 are used for parallel control bits, and one line is used for data validation. Using the external sequence control, one can dynamically select a sequence from a preprogrammed list of 256 sequences. The transition characteristics from sequence to another is programmed using the *seq:sel:tim* command.

Parameters

Name	Type	Default	Description
BUS	Discrete	BUS	Defines that sequences will be switched only when a remote command has been received.
EXTernal	Discrete		Defines that the sequence control is transferred to a rear panel connector. The connector has 8 bits of parallel control lines that can switch between up to 256 sequences.

Response

The SE5082 will return BUS, or EXT depending on the present sequence select setting.

Example

Command `:SEQ:SEL:SOUR EXT`

Query `:SEQ:SEL:SOUR?`

:SEQuence:SELect:TIMing{COHerent|IMMediate}(?)

Description

Use this command to set or query the timing characteristics of the sequence select command. This defines how the generator transitions from sequence to sequence. Use the coherent option to let the sequence complete, before it jumps to the next sequence. Applications that require an unconditional jump can use the immediate option, where the generation of the current sequence is aborted and the new sequence is started immediately thereafter. This command affects the sequence transition timing, regardless if the sequence control is from remote or from the rear panel connector.

Parameters

Name	Type	Default	Description
COHerent	Discrete		Defines that when a new sequence is selected, the transition to the new sequence will occur only when the current sequence has reached its end point.
IMMediate	Discrete	IMM	Defines that when a new sequence is selected, the current sequence will be aborted and the transition to the new sequence will occur immediately, without waiting for the current sequence to reach its end point.

Response

The SE5082 will return COH, or IMM depending on the present sequence select timing setting.

Example

Command `:SEQ:SEL:TIM COH`

Query `:SEQ:SEL:TIM?`

:SEQuence:PREStep{WAVE|DC}(?)

Description

This command is valid only for armed continuous operation and where the selected arming event is BUS. This command is available from remote only and will not operate from exercising front panel controls. This command modifies the normal operation of the sequencer in such a way that it places a blank DC segment in front of the sequence table so, the first time the sequence is selected, a DC signal is present at the output and when triggered for the first time, the sequences steps to its first active waveform in the sequence table.

At the end of the sequence, the sequence repeats itself without using the blank DC pre-step until aborted by the user.

Parameters

Name	Type	Default	Description
WAVE	Discrete	WAVE	This is the default option where immediately when a sequence function is selected, the first waveform in the sequence table is replayed till a legal event occurs that increments the waveform to the next step in the sequence table.
DC	Discrete		This is a special option that places a blank DC waveform in front of the sequence table but is not considered part of the sequence table.

Response

The SE5082 will return WAVE, or DC depending on the present sequence pre-step setting.

Example

Command `:SEQ:PRES DC`

Query `:SEQ:PRES?`

:SEQuence:ONCE:COUNt<loops>(?)

Description

Use this command to set or query the number of loops that a sequence will execute when its advance mode is programmed to ONCE. If a value other than 1 is programmed in the once counter database, then the sequence will execute to its last waveform and then return to an idle state. The sequence repeats n times, depending on the setting of the *seq:once:coun* command.

Parameters

Name	Range	Type	Default	Description
<loops >	1 to 16,777,216	Numeric (integer only)	1	Programs the loop counter for the once advance mode.

Response

The SE5082 will return the active once loop counter value.

Example

Command `:SEQ:ONCE:COUN 33`

Query `:SEQ:ONCE:COUN?`

:SEQuence:SYNC:LOCK<step_number>(?)

Description

By default, the sequence generates a sync pulse on step number 1 of the sequence table. Use this command to move the sync output pulse to different steps. The width of the sync pulse is equal to the length of the waveform that is associated with it.

Parameters

Name		Type	Default	Description
<step_number>	1 to 49,152	Numeric (integer only)	1	Programs the sync pulse to a required position along the sequence table steps.

Response

The SE5082 will return an integer number depending on the current sync lock setting.

Example

Command `:SEQ:SYNC:LOCK 101`

Query `:SEQ:SYNC:LOCK?`

:SEQuence:RESet

Description

Use this command to force the sequence to its first step.

Example

Command `:SEQ:RES`

Advanced Sequencing Control Commands

This group is used to control a special incidence of the sequence generator where sequences, and not segments, are sequenced. Unlike the standard sequence generator that can store up to 1000 different sequence scenarios, the advanced sequencing generator can store only one sequence.

Generating an Advanced Sequence

The advanced sequence generator is similar to the standard sequence generator, except the sequence is made of sequences that were preloaded and stored in individual sequence tables. The advanced sequence generator has the ability to link and loop these sequences in user-programmable order.

If you already built and used standard sequence tables then the procedure of building an advanced sequencing table is very similar. There are a number of tools that you can use to build an advanced sequencing table.

In general, the advanced sequence table is built one step at a time using the a*seq:def* command. The one-step method is slow and tedious. However, it allows better control for someone who is just beginning their first sequence programming. Use the information below to understand the advanced sequencing commands and how to implement them in your application.

The advanced sequencing commands are listed in Table 4-9. Factory defaults after *RST are shown in the Default column. Parameter range and low-and high-limits are listed, where applicable.

Table 4-9, Advanced Sequence Control Commands

Keyword	Parameter Form	Default	Notes
[:SOURce]			
:ASEQuence			
:ADVance	AUTOmatic | ONCE | STEPped	AUTO	The advancing-mode of the advanced sequencer.
:DEFine	<step_no>,<sequence_no>,<loops>,<jump_flag> <1 to 1000><1 to 1000><1 to 1048575><0 or 1>		Define the n'th step of the advanced-sequencer. Note that the length of the advanced-sequencer should be between 3 and 1000.
:DELete			Deletes the advanced-sequencer's table
:LENGth	Only Query		Query the length (in steps)
:ONCe			
:COUNt	1 to 1,048,575		Set the number of times the advanced sequence will be repeated
:SYNC			
:LOCK	1 to 1,000	1	Sync position
:RESet			Force the Aseq to its first step
[:DATA]	#<header><binary-block> 4-bytes with the number of loops (1 to 1048575) 2-bytes with the sequence number (1 to 1000) 1-byte with the jump-flag (either 0 or 1) 1-byte spare		Write the advanced-sequencer's table. Note that the length (in steps) of the advanced sequencer's table should be between 3 and 1000.

:ASEQuence:ADVance{AUTOmatic|ONCE|STEPped}(?)

Description

This command will select the sequence advance mode, which defines how the output advances through the sequencer steps. There are three advance modes: automatic, once and stepped.

In automatic advance mode, the routine goes through the steps automatically and if there are no jump flag, the sequence will end and then start over automatically. If a loop counter other than 1 is programmed for a specific step in the sequence, the step will loop n times and then automatically advance to the next step. The jump flag inhibits the circulation to the next step, until a valid signal at the event input releases the step to jump.

In once advance mode, the routine goes through the steps automatically and if there are no jump flag, the sequence will end and idle on a specific sequence, depending on the selected run mode. If the once counter is programmed to a value other than 1, the sequence will repeat itself n times. If a loop counter other than 1 is programmed for a specific step in the sequence, the step will loop n times and then automatically advance to the next step. The jump flag inhibits the propagation to the next step, until a valid signal at the event input releases the step to jump.

In stepped advance mode, the routine goes through the steps only after a valid event signal. When the sequence is complete, the sequence repeats itself with valid event signals. Jump flags are ignored in stepped mode, but loop counter other than 1 will repeat the step for n times with each event, before advancing to the next step.

Parameters

Name	Type	Default	Description
AUTOmatic	Discrete	AUTO	Specifies continuous advance where the generator steps continuously to the end of the sequence table and repeats the sequence from the start. For example, if a sequence is made of three sequences 1, 2 and 3, the sequence will generate an infinite number of 1,2,3,1,2,3,1,2,3…sequences. Of course, each link (sequence) can be programmed with its associated loop (repeat) number and jump flag to inhibit circulation until an event signal has been received.
ONCE	Discrete		This selects the once sequence advance mode, where the generator steps through the sequence table automatically once, except if the *aseq:once:coun* programmed a value greater than 1.
STEPped	Discrete		Specifies the stepped sequence advance mode, where the generator steps to the next sequence only when a valid event signal has been received.

Response

The SE5082 will return the AUTO, ONCE, or STEP, depending on the present sequence advance mode setting.

Example

Command `:ASEQ:ADV ONCE`

Query `:ASEQ:ADV?`

:ASEQuence#<header><data_array>

Description

This command will build a complete advanced sequence table in one binary download. In this way, there is no need to define and download individual sequence steps. Using this command, sequence table data is loaded to the SE5082 using high-speed binary transfer in a similar way to downloading waveform data with the trace command. High-speed binary transfer allows any 8-bit bytes to be transmitted in a message. This command is particularly useful for long sequences that use a large number of sequences. As an example, the next command will generate three-step sequence with 24 bytes of data that contains sequence number, loops and jump flag option.

```
ASEQuence#224<data_array>
```

This command causes the transfer of 24 bytes of data (3-step sequence) to the advanced sequence table buffer. The <header> is interpreted this way:

- The ASCII "#" ($23) designates the start of the binary data block.
- "2" designates the number of digits that follow.
- "24" is the number of bytes to follow. This number must divide by 8.

The data array contains a list of advanced sequence table entries where each entry is represented by 8 bytes. Therefore, the total number of bytes is always eight times the number of advance sequence steps. For example, 24 bytes are required to download 3 advanced sequence steps to the advanced sequence table.

The code below demonstrates the data structure and the below figure shows how to prepare the 64-bit word for the advanced sequence step, number of loops and jump flag option.

```
typedef struct
{
UINT_32 loops;
  UINT_16 sequence_number;
  UINT_16   jump_flag;
}
AseqTableEntry_t;
```

There are a number of points you should be aware of before you start preparing the data:

1. Each channel has its own advanced sequence table buffer. Therefore, make sure you select the correct active channel (with the INST:SEL command) before you download the advanced sequence table data to the generator
2. Minimum number of advanced sequencer steps is 3; maximum number is 1,000
3. The number of bytes in a complete sequence table must divide by 8. The Model SE5082 has no control over data sent to its sequence table during data transfer. Therefore, wrong data and/or incorrect number of bytes will cause erroneous sequence partition
4. Step numbers are assigned automatically in the same order as the structure is built. The first 8-byte structure forms step number 1 and the last, form step number n. As an example, for 100 steps sequence, one should build an array of 100 AseqTableEntry_t entries.

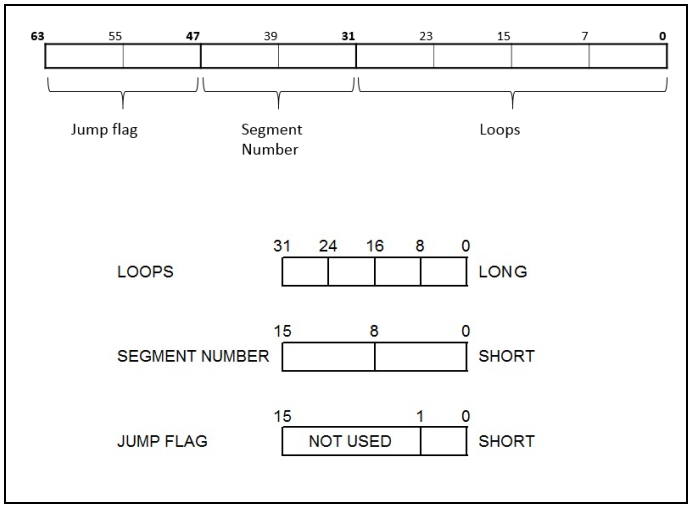

Figure 4-8, 64-bit Advanced Sequence Table Download Format

Parameters

Name	Type	Default	Description
<header>	Discrete		Advanced sequence table header, defines the length of the binary <data_array>.
<data_array>	Binary		Block of binary data that contains information on the sequence table.

Example

Command `:ASEQuence#224<data_array>`

:ASEQuence:DEFine<step>,<sequence_#>,<loops>,<jump_flag>(?)

Description

Use this command to create an advanced sequencing table. It defines all of the parameters that are associated with the sequence step such as: step number, sequence number, loops and jump flag. Each step in a sequence table must be programmed separately, except if the *aseq:data* command is used to download a complete table in binary code. The *aseq:leng* command can be used to first predefine the length of the sequence table. However, the length is adjusted automatically to the number of entries when the *aseq:def* commands program each step.

Parameters

Name	Range	Type	Description
<step>	1 to 1,000	Numeric (integer only)	Programs the step in the sequence table. Steps are indexed from 1 to 1,000 and must be programmed in an ascending order. Empty step locations in a sequence table are not permitted. Minimum number of steps required to create an advanced sequence table is 3.
<sequence_#>	1 to 1,000	Numeric (integer only)	Assigns a sequence to a specific step number. When encountered in the sequence table, the sequence number that is associated with the step will be generated.
<loops>	1 to 1M	Numeric integer only)	Programs the repeat number of loops that a specific step will play before advancing to the next step in the sequence.
<jump_flag>	0-1	Boolean	This flag specifies if the sequence will advance or hold on a specific step. “0” flag specifies normal advance, “1” flag specifies that the output will dwell on the current step and will circulate to the next step only after a valid event has been received.

TIP

The SE5082 attempts to rebuild the sequence table and restart the sequence every time you use the *aseq:def* command and while your generator is sequencing. Therefore, sending this command in sequenced mode will slow the programming process and the operation of the generator. Using the *aseq:def* command in FIX or USER mode will greatly speed up programming time.

Example

Command `:ASEQ:DEF 13,5,52,0`

:ASEQuence:DELete

Description

Use this command to delete and reset the contents of the advanced sequencing table. No variables are associated with this command, because there is only one table available in the system.

Example

Command `:ASEQ:DEL`

:ASEQuence:LENGth?

Description

Use this command to query the number of steps programmed in the active advance sequence table.

Response

The SE5082 will return the present length value of the active advance sequence.

Example

Query `:ASEQ:LENG?`

:ASEQuence:ONCE:COUNt<loops>(?)

Description

Use this command to set or query the number of loops that the advanced sequence will execute when its advance mode is programmed to ONCE. If a value other than 1 is programmed in the once counter database, then the sequence will execute to its last waveform and return to an idle state. The sequence repeats n times, depending on the setting of the *aseq:once:coun* command.

Parameters

Name	Range	Type	Default	Description
<loops >	1 to 1,048,575	Numeric (integer only)	1	Programs the loop counter for the once advance mode.

Response

The SE5082 will return the active once loop counter value.

Example

Command `:ASEQ:ONCE:COUN 12`

Query `:ASEQ:ONCE:COUN?`

:ASEQuence:LOCK<seq_number>(?)

Description

By default, the advanced sequence generates a sync pulse on step number 1 of the advanced sequence table. Use this command to move the sync output pulse to different steps. The width of the sync pulse is equal to the length of the sequence that is associated with it.

Parameters

Name		Type	Default	Description
<seq_number>	1 to 1,000	Numeric (integer only)	1	Programs the sync pulse to a required position along the sequence table steps.

Response

The SE5082 will return an integer number depending on the current sync lock setting.

Example

Command `:ASEQ:LOCK 3`

Query `:ASEQ:LOCK?`

:ASEQuence:RESet

Description

Use this command to force the advanced sequence to its first step.

Example

Command `:ASEQ:RES`

Modulated Waveforms Global Control Commands

This group is used to set up the instrument in modulated waveforms mode and to select the general parameters that control all modulation functions. Note that the modulation can be turned off to create continuous carrier waveforms (CW). The following modulation schemes can be selected and controlled: AM, FM, Sweep, Chirp, FSK, ASK, Frequency Hopping and Amplitude Hopping. The modulated waveforms global control commands are summarized in Table 4-10. Factory defaults after *RST are shown in the Default column. Parameter range and low and high limits are listed, where applicable.

Table 4-10, Modulated Waveforms Global Commands

Keyword	Parameter Form	Default	Notes
[:SOURce]			
:MODulation			
:TYPE	OFF | AM | FM | SWEep | CHIRp | FSK | ASK | FHOPping | AHOPping	OFF	
:CARRier			
[:FREQuency]	10e3 to 2.5e9	1e6	
:FUNCtion	SINusoid | TRIangle | SQUare	SIN	

:MODulation:TYPE{OFF|AM|FM|SWEep|CHIRp|FSK|ASK|FHOPping|AHOPping}(?)

Description

This command selects the modulation type. All modulation types are internal, so external signals are not required for producing modulation.

Parameters

Name	Type	Default	Description
OFF	Discrete	OFF	Modulation off is a special mode where the output generates continuous, non-modulated sinusoidal carrier waveforms (CW).
AM	Discrete		This turns on the AM function. Program the AM parameters to fine-tune the function for your application.
FM	Discrete		This turns on the FM function. Program the FM parameters to fine-tune the function for your application.
SWEep	Discrete		This turns on the sweep function. Program the sweep parameters to fine-tune the function for your application.
CHIRp	Discrete		This turns on the chirp function. Program the chirp parameters to fine-tune the function for your application.

FSK	Discrete	This turns on the FSK function. Program the FSK parameters to fine tune the function for your application.
ASK	Discrete	This turns on the ASK function. Program the ASK parameters to fine tune the function for your application.
FHOPping	Discrete	This turns on the frequency hopping function. Program the hop parameters to fine tune the function for your application.
AHOPping	Discrete	This turns on the amplitude hopping function. Program the amplitude hopping parameters to fine tune the function for your application.

Response

The SE5082 will return OFF, FM, AM, SWE, FSK, ASK, HOP, AHOP depending on the present modulation type setting.

Example

Command `:MOD:TYPE AM`

Query `:MOD:TYPE?`

:MODulation:CARRier<frequency>(?)

Description

This command programs the CW frequency. Note that the CW waveform can be sine, square, triangle or ramp and its frequency setting is separate from the standard waveform. However, please note that the frequency range of the CW waveform depends on the shape as detailed in the standard waveform section. The CW frequency setting is valid for all modulation types.

Parameters

Name	Range	Type	Default	Description
<frequency>	10e3 to 2.5e9	Numeric	1e6	Programs the frequency of the carrier waveform in units of Hz. Note that the CW frequency should correspond to the CW shape.

Response

The SE5082 will return the current carrier frequency value.

Example

Command `:MOD:CARR 100e6`

Query `:MOD:CARR?`

:MODulation:CARRier:FUNCtion{SINusoid|TRIangle|SQUare}(?)

Description

This specifies the carrier function. There are three functions that can be modulated: Sine, Triangle and Square. The sine, triangle and the square are computed and placed in the memory as complete waveforms and the modulation schemes are computed and replayed as arbitrary waveforms.

Parameters

Name	Type	Default	Description
SINusoid	Discrete	SIN	Selects sine as the modulated waveform
TRIangle	Discrete		Selects triangle as the modulated waveform
SQUare	Discrete		Selects square as the modulated waveform

Response

The SE5082 will return SIN, TRI, SQU or RAMP depending on the selected waveform setting.

Example

Command `:MOD:CARR:FUNC TRI`

Query `:MOD:CARR:FUNC?`

Modulation Control Commands

This group is used to control parameters for individual modulation schemes. The modulation control commands are summarized in Table 4-11. Factory defaults after *RST are shown in the Default column. Parameter range and low and high limits are listed, where applicable.

Table 4-11, Modulated Waveforms Control Commands

Keyword	Parameter Form	Default	Notes
[:SOURce]			
:AM			
:FUNCtion			
[:SHAPe]	SINusoid | TRIangle | SQUare | RAMP	SIN	
:INTernal			
:FREQuency	100.0 to 100.0e6	1e3	The ratio: MODulation:CARRier:FREQ/ AM:INTernal:FREQ Must be between 10.0 and 40.0e+3
:DEPTh	0 to 200	50	

Table 4-11, Modulated Waveforms Control Commands (continued)

Keyword	Parameter Form	Default	Notes
:FM			
:DEViation	10e-3 to 1.25e+9	500e3	0.5 * Max Carrier Frequency
:FUNCtion			
[:SHAPe]	SINusoid | TRIangle | SQUare | RAMP	SIN	
:FREQuency	100.0 to 250.0e+6	10e3	0.1 * Max carrier frequency The ratio: MODulationCARRier:FREQ / FM:FREQ Must be between 10.0 and 40.0e+3
:MARKer			
[:FREQuency]	10e3 to 2.5e+9	505e3	
:SWEep			
:FREQuency			
[:STARt]	10e3 to 2.5e+9	40e6	
:STOP	10e3 to 2.5e+9	80e6	
:TIME	0.5e-6 to 0.01 (seconds)	10e-6	The product of: Max(SWEep:FREQ:START, SWEep:FREQ:STOP) * SWEep:TIME Must be between 10.0 and 40.0e+3
:DIRection	UP | DOWN	UP	
:SPACing	LINear | LOGarithmic	LIN	
:MARKer			
[:FREQuency]	10e3 to 2.5e+9	60e6	
:CHIRp			
:WIDTh	0.5e-6 to 0.01 (seconds)	10e-6	The product of: Max(CHIRp:FREQ:START, CHIRp:FREQ:STOP) * CHIRp:WIDTh Must be between 10.0 and 40.0e+3
:REPetition	200e-9 to 20	100e-6	
:FREQuency			
[:STARt]	10e3 to 2.5e+9	40e6	
:STOP	10e3 to 2.5e+9	80e6	
:DIRection	UP | DOWN	UP	
:SPACing	LINear | LOGarithmic	LIN	
:MARKer			
[:FREQuency]	10e3 to 2.5e+9	60e6	
:AMPLitude			
:DEPTh	0 to 100%	50%	
:DIRection	UP | DOWN	UP	
:SPACing	LINear | LOGarithmic	LIN	
:FSK			
:FREQuency			
:SHIFted	10e3 to 2.5e9	2e6	
:BAUD	0.1 to 1.0e9	10e3	
:MARKer	1 to FSK symbols list length	1	
:DATA	#<header><binary-data> Symbols-list length 2 to 1000.		Download the FSK symbols list.

Table 4-11, Modulated Waveforms Control Commands (Continued)

Keyword	Parameter Form	Default	Notes
:ASK			
[:AMPLitude]			
[:STARt]	100e-3 to 1.2 (v) for DC amplifier	0.5	
	50e-3 to 2.0 (v) for HV amplifier		
	50e-3 to 0.54 (v) for DR (DAC output)		
:SHIFted	100e-3 to 1.2 (v) for DC amplifier	1	
	50e-3 to 2.0 (v) for HV amplifier		
	50e-3 to 0.54 (v) for DR (DAC output)		
:BAUD	0.1 to 1.0e9	10e3	
:MARKer	1 to ASK symbols list length	1	
:DATA	#<header><binary-data> Symbols-list length 2 to 1000.		Download the ASK symbols list.
:FHOPping			
:DWELl			
:MODE	FIXed | VARiable	VAR	
[:TIMe]	1e-9 to 10 (seconds)	5e-6	The "fixed" dwell time. It should not exceed 10 / (length of the "fixed" FHOP list)
:FIXed			
:DATA	#<header><binary-data> Array of (2 and 256) double-float values		Download the frequencies list Total dwell-time must not exceed 10 seconds.
:VARiable			
:DATA	#<header><binary-data> Array of (2 to 256) double-float values (frequency, dwell-time) pairs		Download the list of (frequency, dwell-time) pairs Total dwell-time must not exceed 10 seconds.
:MARKer	1 to (active) FHOP list length	1	
:AHOPping			
:DWELl			
:MODE	FIXed | VARiable	VAR	
[:TIMe]	1e-9 to 10 (seconds)	5e-6	Should not exceed 10 / (length of the "fixed" AHOP list)
:FIXed			
:DATA	#<header><binary-data> Array of (2 to 256) double-float values 100e-3 to 1.2(v) for DC amplifier 50e-3 to 2.0 (v) for HV amplifier 50e-3 to 0.54 (v) for DR (DAC output)		Download the amplitudes list Total dwell-time (=fixed_dwell_time * list_length) must not exceed 10 seconds.
:VARiable			
:DATA	#<header><binary-data> Array of (2 to 256) double-float values(amplitude, dwell-time) pairs 50e-3 to 1.2(v) for HB amplifier 50e-3 to 2.0 (v) for HV amplifier 50e-3 to 0.54 (v) for DR (DAC output)		Download the list of (amplitude, dwell-time) pairs Total dwell-time (=sum of the variable dwell-times) must not exceed 10 seconds.
:MARKer	1 to (active) AHOP list length	1	

AM Programming

Use the following command for programming the AM parameters. AM control is internal. The commands for programming the amplitude modulation function are described below. Note that the carrier waveform frequency (CW) setting is common to all modulation schemes.

:AM:FUNCtion:SHAPe(SINusoid|TRIangle|SQUare|RAMP}(?)

Description

This command will select one of the waveform shapes as the active modulating waveform.

Parameters

Name	Type	Default	Description
SINusoid	Discrete	SIN	Selects the sine shape as the modulating waveform
TRIangle	Discrete		Selects the triangular shape as the modulating waveform
SQUare	Discrete		Selects the square shape as the modulating waveform
RAMP	Discrete		Selects the ramp shape as the modulating waveform

Response

The SE5082 will return SIN, TRI, SQU, or RAMP depending on the selected function shape setting.

Example

Command `:AM:FUNC:SHAP SQU`

Query `:AM:FUNC:SHAP?`

:AM:INTernal:FREQuency<am_freq>(?)

Description

This command will set the modulating wave frequency for the built-in standard modulating waveform library. Please not that the ratio: MODulation:CARRier:FREQ / AM:INTernal:FREQ must be between 10.0 and 40.0e+3.

Parameters

Name	Range	Type	Default	Description
<am_freq>	100 to 100e6	Numeric	1e3	Programs the frequency of the modulating waveform in units of Hz. The frequency of the built-in standard modulating waveforms only is affected. Note that maximum carrier-to-internal-frequency-ratio is 1e6.

Response

The SE5082 will return the present modulating waveform frequency value. The returned value will be in standard scientific format (for example: 100kHz would be returned as 100e3 – positive numbers are unsigned).

Example

Command `:AM:INT:FREQ 100e3`

Query `:AM:INT:FREQ?`

:AM:DEPTh<depth>(?)

Description

This command will set the modulating wave frequency for the built-in standard modulating waveform library.

Parameters

Name	Range	Type	Default	Description
<depth>	0 to 200	Numeric	50	Programs the depth of the modulating waveform in units of percent.

Response

The SE5082 will return the present modulating depth value.

Example

Command `:AM:DEPT 100`

Query `:AM:DEPT?`

FM Programming

Use the following command for programming the FM parameters. FM control is internal. The commands for programming the frequency modulation function are described below. Note that the carrier waveform frequency (CW) setting is common to all modulation schemes.

:FM:DEViation<deviation>(?)

Description

This programs the deviation range around the carrier frequency. The deviation range is always symmetrical about the carrier frequency. Please note that the deviation frequency can be up to 0.5 * Max Carrier Frequency.

Parameters

Name	Range	Type	Default	Description
<deviation>	10e-3 to 1.25e9	Numeric	500e3	Programs the deviation range around the carrier frequency in units of Hz.

Response

The SE5082 will return the present deviation frequency value. The returned value will be in standard scientific format (for example: 100mHz would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:FM:DEV 10e6`

Query `:FM:DEV?`

:FM:FUNCtion:SINusoid|TRIangle|SQUare|RAMP}(?)

Description

This command will select one of the waveform shapes as the active modulating waveform.

Parameters

Name	Type	Default	Description
SINusoid	Discrete	SIN	Selects the sine shape as the modulating waveform
TRIangle	Discrete		Selects the triangular shape as the modulating waveform
SQUare	Discrete		Selects the square shape as the modulating waveform
RAMP	Discrete		Selects the ramp shape as the modulating waveform

Response

The SE5082 will return SIN, TRI, SQU, or RAMP depending on the selected function shape setting.

Example

Command `:FM:FUNC:SHAP TRI`

Query `:FM:FUNC:SHAP?`

:FM:FREQuency<fm_freq>(?)

Description

This command will set the modulating wave frequency for the built-in standard modulating waveform library. Please note that the ratio MODulation:CARRier:FREQ / FM:FREQ must be between 10.0 and 40.0e+3.

Parameters

Name	Range	Type	Default	Description
<fm_freq>	100 to 50e6	Numeric	10e3	Programs the frequency of the modulating waveform in units of Hz.

Response

The SE5082 will return the present modulating waveform frequency value. The returned value will be in standard scientific format (for example: 100mHz would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:FM:FREQ 1e6`

Query `:FM:FREQ?`

:FM:MARKer<frequency>(?)

Description

This function programs marker frequency position. An FM marker can be placed inside the following range: (carrier frequency ± deviation frequency). The marker pulse is output from the SYNC output connector. Please note that the FM marker is not available for function shape square.

Parameters

Name	Range	Type	Default	Description
<frequency>	10e3 to 2.5e9	Numeric	505e3	Programs the marker frequency position in units of Hz.

Response

The SE5082 returns the present marker frequency value. The returned value will be in standard scientific format (for example: 100mHz would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:FM:MARK 12.3e6`

Query `:FM:MARK?`

Sweep Modulation Programming

Use the following command for programming the sweep parameters. Sweep control is internal. The frequency will sweep from start to stop frequencies at an interval determined by the sweep time value and controlled by a step type determined by the sweep step parameter.

There are two sweep modes: Linear, where the step of the generator increments from start-to-stop frequency is linear; and Logarithmic, where the step of the generator increments from start-to-stop frequency is logarithmic.

The commands for programming the frequency sweep function are described below.

:SWEep:FREQuency<start_freq>(?)

Description

This specifies the sweep start frequency. The SE5082 will normally sweep from start-to-stop frequencies. However, if the sweep direction is reversed, the output will sweep from stop-to-start frequencies. The start-and-stop frequencies may be programmed freely throughout the frequency of the CW frequency range.

Parameters

Name	Range	Type	Default	Description
<start_freq>	10e3 to 2.5e9	Numeric	40e3	Programs the sweep start frequency. Sweep start is programmed in units of Hz.

Response

The SE5082 will return the present sweep start frequency value. The returned value will be in standard scientific format (for example: 100mHz would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:SWE:FREQ 10e6`

Query `:SWE:FREQ?`

:SWEep:FREQuency:STOP<stop_freq>(?)

Description

This specifies the sweep stop frequency. The SE5082 will normally sweep from start-to-stop frequencies. However, if the sweep direction is reversed, the output will sweep from stop-to-start frequencies. The start-and-stop frequencies may be programmed freely throughout the frequency range however, please note that for best results, the max frequency of the selected function shape should not exceed its max frequency as specified in the standard waveform frequency range. For example Ramp max frequency is 300MHz.

Parameters

Name	Range	Type	Default	Description
<stop_freq>	10e3 to 2.5e9	Numeric	80e6	Programs the sweep stop frequency. Sweep stop is programmed in units of Hz.

Response

The SE5082 will return the present sweep stop frequency value. The returned value will be in standard scientific format (for example: 100mHz would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:SWE:FREQ:STOP 100e6`

Query `:SWE:FREQ:STOP?`

:SWEep:TIMe<time>(?)

Description

This specifies the time that will take the SE5082 to sweep from start-to-stop frequencies. The time does not depend on the sweep boundaries, as it is automatically adjusted by the software to the required interval. At the end of the sweep cycle, the output waveform maintains the sweep stop frequency setting, except if the SE5082 is in continuous run mode, where the sweep repeats itself continuously. Please note the product of:

Max(SWEep:FREQ:START, SWEep:FREQ:STOP) * SWEep:TIME must be between 10.0 and 40.0e+3

Parameters

Name	Range	Type	Default	Description
<time>	0.5e-6 to 0.01	Numeric	10e-6	Programs the sweep time. Sweep time is programmed in units of s.

Response

The SE5082 will return the present sweep time. The returned value will be in standard scientific format (for example: 100ms would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:SWE:TIM 1e-6`

Query `:SWE:TIM?`

:SWEep:DIRection(UP|DOWN}(?)

Description

This specifies if the SE5082 output will sweep from start-to-stop (UP) or from stop-to-start (DOWN) frequencies. Sweep time does not affect the sweep direction and frequency limits. At the end of the sweep cycle, the output waveform normally maintains the sweep stop frequency setting, but will maintain the start frequency if the DOWN option is selected, except if the SE5082 is in continuous run mode where the sweep

repeats itself continuously.

Parameters

Name	Type	Default	Description
UP	Discrete	UP	Selects the sweep up direction
DOWN	Discrete		Select the sweep down direction

Response

The SE5082 will return UP, or DOWN depending on the selected direction setting.

Example

Command `:SWE:DIR UP`

Query `:SWE:DIR?`

:SWEep:SPACing(LINear|LOGarithmic}(?)

Description

This specifies the sweep step type. Two options are available: logarithmic or linear. In linear, the incremental steps between the frequencies are uniform throughout the sweep range. Logarithmic type defines logarithmic spacing throughout the sweep start and stop settings.

Parameters

Name	Type	Default	Description
LINear	Discrete	LIN	Selects the linear sweep spacing
LOGarithmic	Discrete		Select the logarithmic sweep spacing

Response

The SE5082 will return LIN or LOG, depending on the selected spacing setting.

Example

Command `:SWE:SPAC LOG`

Query `:SWE:SPAC?`

:SWEep:MARKer<frequency>(?)

Description

This function programs marker frequency position. Sweep marker can be placed in between the start and the stop frequencies. The marker pulse is output from the SYNC output connector.

Parameters

Name	Range	Type	Default	Description
<frequency>	10e3 to 2.5e9	Numeric	60e6	Programs the marker frequency position in units of Hz.

Response

The SE5082 will return the present marker frequency value. The returned value will be in standard scientific format (for example: 100mHz would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:SWE:MARK 50e6`

Query `:SWE:MARK?`

Chirp Modulation Programming

Use the following command for programming the chirp parameters. Chirp control is internal. Within a single chirp, one can sweep the frequency from start to stop settings and modulate the amplitude from 0% to 100% simultaneously. Other chirp parameters are fully controlled by these commands

The commands for programming the chirp function are described below.

:CHIRp:WIDTh<width>(?)

Description

Use this command to set or query the time that will lapse from chirp start to chirp end. Please note that the product of:

Max(CHIRp:FREQ:START, CHIRp:FREQ:STOP) * CHIRp:WIDTh must be between 10.0 and 40.0e+3

Parameters

Name	Range	Type	Default	Description
<width>	0.5e-6 to 0.01	Numeric	10e-6	Programs the width of the chirp in units of seconds.

Response

The SE5082 will return the present chirp width value. The returned value will be in standard scientific format. (for example: 100ms would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:CHIR:WIDT 10e6`

Query `:CHIR:WIDT?`

:CHIRp:REPetition<interval>(?)

Description

Use this command to set or query the time interval between consecutive chirp cycles. The time is measured between two adjacent chirp starts.

Parameters

Name	Range	Type	Default	Description
<interval>	200e-9 to 2	Numeric	25e-6	Programs the intervals between adjacent chirp cycles. Chirp repetition is programmed in units of seconds.

Response

The SE5082 will return the present chirp repetition value. The returned value will be in standard scientific format (for example: 100ms would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:CHIR:REP 100e-6`

Query `:CHIR:REP?`

:CHIRp:FREQuency[:STARt]<start_freq>(?)

Description

Use this command to set or query the start frequency within the chirp cycle. Start and stop frequencies can be identical for non-swept chirps but could also be different if frequency sweep is required within a single chirp cycle.

Parameters

Name	Range	Type	Default	Description
<start_freq>	10e3 to 2.5e9	Numeric	40e6	Programs the chirp start frequency. Chirp start frequency is programmed in units of seconds.

Response

The SE5082 will return the present chirp start frequency value. The returned value will be in standard scientific format (for example: 100ms would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:CHIR:FREQ 100e6`

Query `:CHIR:FREQ?`

:CHIRp:FREQuency:STOP<stop_freq>(?)

Description

Use this command to set or query the stop frequency within the chirp cycle. Start and stop frequencies can be identical for non-swept chirps but could also be different if frequency sweep is required within a single chirp cycle. The start-and-stop frequencies may be programmed freely throughout the frequency range however, please note that for best results, the max frequency of the selected function shape should not exceed its max frequency as specified in the standard waveform frequency range. For example Ramp max frequency is 300MHz.

Parameters

Name	Range	Type	Default	Description
<start_stop>	10e3 to 2.5e9	Numeric	80e6	Programs the chirp stop frequency. Chirp start frequency is programmed in units of seconds.

Response

The SE5082 will return the present chirp stop frequency value. The returned value will be in standard scientific format (for example: 100ms would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:CHIR:FREQ:STOP 300e6`

Query `:CHIR:FREQ:STOP?`

:CHIRp:DIRection(UP|DOWN}(?)

Description

Use this command to set or query the chirp frequency direction. This specifies if the SE5082 output will sweep from start-to-stop (UP) or from stop-to-start (DOWN) frequencies. Chirp width does not affect the sweep direction and frequency limits.

Parameters

Name	Type	Default	Description
UP	Discrete	UP	Selects the sweep up direction
DOWN	Discrete		Select the sweep down direction

Response

The SE5082 will return UP, or DOWN depending on the selected direction setting.

Example

Command `:CHIR:DIR UP`

Query `:CHIR:DIR?`

:CHIRp:SPACing(LINear|LOGarithmic}(?)

Description

Use this command to set or query the chirp frequency steps. Two options are available: logarithmic or linear. In linear, the incremental steps between the frequencies are uniform throughout the sweep range. Logarithmic type defines logarithmic spacing throughout the sweep start and stop settings.

Parameters

Name	Type	Default	Description
LINear	Discrete	LIN	Selects the linear sweep spacing
LOGarithmic	Discrete		Select the logarithmic sweep spacing

Response

The SE5082 will return LIN or LOG, depending on the selected spacing setting.

Example

Command `:CHIR:SPAC LOG`

Query `:CHIR:SPAC?`

:CHIRp:MARKer<frequency>(?)

Description

This function programs marker frequency position. Chirp marker can be placed in between the start and the stop frequencies. The marker pulse is output from the SYNC output connector.

Parameters

Name	Range	Type	Default	Description
<frequency>	10e3 to 2.5e9	Numeric	60e6	Programs the marker frequency position in units of Hz.

Response

The SE5082 will return the present marker frequency value. The returned value will be in standard scientific format (for example: 100mHz would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:CHIR:MARK 50e6`

Query `:CHIR:MARK?`

:CHIRp:AMPLitude:DEPTh<index>(?)

Description

Use this command to set or query the chirp amplitude modulation index. Chirp amplitude can be modulated up or down, depending on the chirp direction setting.

Parameters

Name	Range	Type	Default	Description
<index>	0 to 100	Integer	50	Programs the chirp amplitude modulation index in units of %.

Response

The SE5082 will return the present amplitude modulation index value. The returned value will be an integer.

Example

Command `:CHIR:AMPL:DEPT 75`

Query `:CHIR:AMPL:DEPT?`

:CHIRp:AMPLitude:DIRection(UP|DOWN}(?)

Description

Use this command to set or query the chirp amplitude direction. The start and stop amplitude settings is determined by the chirp amplitude modulation depth.

Parameters

Name	Type	Default	Description
UP	Discrete	UP	Selects the chirp amplitude modulation up direction
DOWN	Discrete		Select the chirp amplitude modulation down direction

Response

The SE5082 will return UP or DOWN, depending on the selected chirp amplitude direction setting.

Example

Command `:CHIR:AMPL:DIR DOWN`

Query `:CHIR:AMPL:DIR?`

:CHIRp:AMPLitude:SPACing(LINear|LOGarithmic}(?)

Description

Use this command to set or query the chirp amplitude spacing. Two options are available: logarithmic or linear. In linear, the incremental steps between the amplitudes are uniform throughout the chirp range. Logarithmic type defines logarithmic amplitude spacing throughout the chirp start and stop settings.

Parameters

Name	Type	Default	Description
LINear	Discrete	LIN	Selects the linear chirp amplitude spacing
LOGarithmic	Discrete		Select the logarithmic chirp amplitude spacing

Response

The SE5082 will return LIN or LOG, depending on the selected chirp amplitude spacing setting.

Example

Command `:CHIR:AMPL:SPAC LOG`

Query `:CHIR:AMPL:SPAC?`

FSK Modulation Programming

Use the following command for programming the FSK parameters. FSK control is internal. The frequency will shift from carrier to shifted frequency setting at a rate determined by the baud value and controlled by a sequence of bits in the FSK data table. The commands for programming the frequency shift keying function are described below. Note that the carrier waveform frequency (CW) setting is common to all modulation schemes.

:FSK:FREQuency:SHIFted<shift_freq>(?)

Description

This programs the shifted frequency. The frequency shifts when the pointer in the data array points to “1”.

Parameters

Name	Range	Type	Default	Description
<shift_freq>	10e3 to 2.5e9	Numeric	2e6	Programs the shifted frequency value in units of Hz.

Response

The SE5082 will return the present shifted frequency value. The returned value will be in standard scientific format (for example: 100mHz would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:FSK:FREQ:SHIF 100e6`

Query `:FSK:FREQ:SHIF?`

:FSK:FREQuency:BAUD<baud>(?)

Description

This allows the user to select FSK word rate. The word rate is the interval of which the bit streams in the FSK data array are clocked causing the output frequency to hop from carrier to shifted frequency values and vice versa.

Parameters

Name	Range	Type	Default	Description
<baud>	0.1 to 1e9	Numeric	10e3	Programs the rate of which the frequency shifts from carrier to shifted frequency in units of Hz.

Response

The SE5082 will return the present baud value. The returned value will be in standard scientific format (for example: 100mHz would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:FSK:FREQ:BAUD 10e6`

Query `:FSK:FREQ:BAUD?`

:FSK:FREQuency:MARKer<index>(?)

Description

Programs where on the data stream the SE5082 will generate a pulse, designated as FSK marker, or index point. The marker pulse is generated at the SYNC output connector. Note that if you intend to program marker position, you must do it before you load the FSK data list and the marker position can be between 1 and the FSK synbols list length.

Parameters

Name	Range	Type	Default	Description
<index>	1 to Symbols list length	Numeric (integer only)	1	Programs a marker pulse at an index bit position.

Response

The SE5082 will return the present marker position.

Example

Command `:FSK:FREQ:MARK 19`

Query `:FSK:FREQ:MARK?`

:FSK:DATA#<header><fsk_data>

Description

Loads the data stream that will cause the SE5082 to hop from carrier to shifted frequency and vice versa. Data format is a string of "0" and "1" which define when the output generates carrier frequency and when it shifts frequency to the FSK value. "0" defines carrier frequency,"1" defines shifted frequency. Note that if you intend to program marker position, you must do it before you load the FSK data list.

Below you can see how an FSK data table is constructed. The sample below shows a list of 10 shifts. The SE5082 will step through this list, outputting either carrier or shifted frequencies, depending on the data list: Zero will generate carrier frequency and One will generate shifted frequency. The binary-data is array of N bytes holding the symbols-list, where each byte holds single symbol-value (0 or 1).

Sample FSK Data Array

0 1 1 1 0 1 0 0 0 1

Parameters

Name	Type	Description
<header>	Discrete	FSK header, defines the length of the binary <data_array> in bytes.
<fsk_data>	Binary	Block of binary data that contains information for the generator when to shift from carrier to shifted frequency and vice versa. The symbols-list length should be between 2 and 1000.

Example

Command `:FSK:DATA#210`
`<data_aray>`

ASK Modulation Programming

Use the following command for programming the ASK parameters. ASK control is internal. The amplitude will toggle between two amplitude settings at a rate determined by the baud value and controlled by a sequence of bits in the ASK data table. The commands for programming the amplitude shift keying function are described below. Note that the carrier waveform frequency (CW) setting is common to all modulation schemes.

:ASK<amplitude>(?)

Description

This programs the normal amplitude setting. The amplitude shifts when the pointer in the data array points to “1”.

Parameters

Name	Range	Type	Default	Description
<amplitude>	DC -100e-3 to 1.2 HV - 50e-3 to 2.0 DR - 50e-3 to 0.54	Numeric	0.5 0.5 0.25	Programs the amplitude setting in units of volt. Range depends on installed module option

Response

The SE5082 will return the present amplitude value. The returned value will be in standard scientific format (for example: 100mV would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:ASK 0.75`

Query `:ASK?`

:ASK:SHIFted<shift_ampl>(?)

Description

This programs the shifted amplitude. The amplitude shifts when the pointer in the data array points to “1”.

Parameters

Name	Range	Type	Default	Description
<shift_ampl>	DC -100e-3 to 1.2 HV - 50e-3 to 2.0 DR - 50e-3 to 0.54	Numeric	1 1 0.5	Programs the shifted amplitude setting in units of volt.

Response

The SE5082 will return the present shifted amplitude value. The returned value will be in standard scientific format (for example: 100mHz would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:ASK:SHIF 0.25`

Query `:ASK:SHIF?`

:ASK:BAUD<rate>(?)

Description

This allows the user to select ASK word rate. The word rate is the interval of which the bit streams in the ASK data array are clocked causing the output amplitude to hop from one level to shifted amplitude level values and visa versa.

Parameters

Name	Range	Type	Default	Description
<rate>	0.1 to 1e9	Numeric	10e3	Programs the rate of which the frequency shifts from carrier to shifted frequency in units of Hz.

Response

The SE5082 will return the present baud value. The returned value will be in standard scientific format (for example: 100kHz would be returned as 100e3 – positive numbers are unsigned).

Example

Command `:ASK:BAUD 25e6`

Query `:ASK:BAUD?`

:ASK:FREQuency:MARKer<index>(?)

Description

Programs where on the data stream the SE5082 will generate a pulse, designated as ASK marker, or index point. The marker pulse is generated at the SYNC output connector. Note that if you intend to program marker position, you must do it before you load the ASK data list.

Parameters

Name	Range	Type	Default	Description
<index>	1 to Symbols list length	Numeric (integer only)	1	Programs a marker pulse at an index bit position.

Response

The SE5082 will return the present marker position.

Example

Command `:ASK:FREQ:MARK 41`

Query `:ASK:FREQ:MARK?`

:ASK:DATA#<header><ask_data>

Description

Loads the data stream that will cause the SE5082 to hop from one amplitude level to shifted amplitude level and vice versa. Data format is a string of "0" and "1" which define when the output generates base level and when it shifts amplitude to the ASK value. "0" defines base level amplitude,"1" defines shifted amplitude level. Note that if you intend to program marker position, you must do it before you load the ASK data list.

Below you can see how an ASK data table is constructed. The sample below shows a list of 10 shifts. The SE5082 will step through this list, outputting either base or shifted amplitudes, depending on the data list: Zero will generate base level and One will generate shifted amplitude. The binary-data is array of N bytes holding the symbols-list, where each byte holds single symbol-value (0 or 1).

Sample ASK Data Array

0 1 1 1 0 1 0 0 0 1

Parameters

Name	Type	Description
<header>	Discrete	ASK header, defines the length of the binary <data_array> in bytes.
<ask_data>	Binary	Block of binary data that contains information for the generator when to shift from base to shifted amplitude and vice versa. The symbols-list length should be between 2 and 1000.

Example

Command `:ASK:DATA#210`
`<data_array>`

Frequency Hopping Modulation Programming

Use the following command for programming the frequency hop parameters. Hop control is internal. The frequency will hop from frequency to frequency at a rate determined by the dwell time value and controlled by a sequence of frequencies in the HOP data table.

There are two hop modes: Fixed Dwell, where the rate of which the generator hops from frequency to frequency is constant and Variable Dwell, where the rate of which the generator hops from frequency to frequency is programmable for each hop.

The commands for programming the frequency hopping function are described below. Note that the carrier waveform frequency (CW) setting is common to all modulation schemes.

:FHOP:DWELl:MODE(FIXed|VARiable}(?)

Description

This selects between fixed or variable dwell-time for the frequency hops. Select the fixed option if you want each frequency to dwell equally on each step. The variable option lets you program different dwell times for each frequency hop. The SE5082 output hops from one frequency to the next according to a sequence given in a hop table. The variable dwell time table contains dwell time data for each step however, the fixed dwell time table does not contain any dwell time information and therefore, if you select the fixed option, make sure your dwell time is programmed as required.

Parameters

Name	Type	Default	Description
FIXed	Discrete	VAR	Selects the fixed dwell time frequency hops mode
VARiable	Discrete		Select the variable dwell time frequency hops mode

Response

The SE5082 will return FIX, or VAR depending on the selected dwell setting.

Example

Command `:FHOP:DWEL:MODE FIX`

Query `:FHOP:DWEL:MODE?`

:FHOP:DWELl<dwell_time>(?)

Description

This selects the dwell time for frequency hops when the selected mode is Fixed dwell time hops. The dwell time table in this case does not contain the dwell time per step parameters and therefore, the value which is programmed with this command remains constant for the entire hop sequence.

Parameters

Name	Range	Type	Default	Description
<dwell_time>	1e-9 to 10	Numeric	5e-6	Programs dwell time for the fixed dwell-time frequency hop function. The same dwell time will be valid for each frequency hop. Dwell time is programmed in units of s and should not exceed 10 / (length of the "fixed" FHOP list).

Response

The SE5082 will return the present dwell time value. The returned value will be in standard scientific format (for example: 100 ms would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:FHOP:DWEL 1e-3`

Query `:FHOP:DWEL?`

:FHOP:FIX:DATA#<header><fix_hop_data>

Description

This command will download the data array that will cause the instrument to hop through the frequency list. The dwell time for each frequency list item is fixed and can be programmed using the HOP:DWEL command. Note that if you intend to program marker position, you must do it first and then load the frequency hops list.

Below you can see how a hop table is constructed. The file sample below shows a list of 10 frequencies. The SE5082 will hop through this list, outputting the next frequency each time it hops. Note that the last cycle is always completed even if the dwell time is shorter than the period of the waveform. For example, if you program dwell time of 1ms and the frequency step has frequency of 1Hz (1s period), the frequency step will last 1 second although the dwell time is 1ms and the total dwell-time (fixed_dwell_time * list_length) must not exceed 10 seconds.

Sample Frequency Hops Data Array

1e+6 2e+6 3e+3 4e+6 5e+5 6e+2 7e+1 8e+6 9e+3 10e+5

Parameters

Name	Type	Description
<header>	Discrete	FHOP data header, defines the length of the binary <data_array> in bytes.
<fix_hop_data>	Double	Block of binary data array of (2 to 256) double-float values that contains information of frequency values.

Example

Command `:FHOP:FIX:DATA#210`
`<data_array>`

:FHOP:VARiable:DATA#<header><var_hop_data>

Description

This command will download the data array that will cause the instrument to hop through the frequency list. The dwell time for each frequency list item is variable and is supplied in the variable hop table data array. Note that the HOP:DWEL command has no effect on this sequence. Also note that if you intend to program marker position, you must do it first and then load the frequency hops list.

Below you can see how a hop table is constructed. The file sample below shows a list of 10 frequencies and their associated dwell times. The SE5082 will hop through this list, outputting the next frequency each time it hops. Note that the last cycle is always completed even if the dwell time is shorter than the period of the waveform. For example, if you program dwell time of 1 ms and the frequency step has frequency of 1Hz (1s period), the frequency step will last 1 second although the dwell time is 1 ms. Total dwell-time must not exceed 10 seconds.

Sample Frequency Hops Data Array

1e+6 100e-6 2e+6 200e-6 3e+3 3e-4 4e+6 40e-2 5e+5 5e-3 6e+2 600e-6 7e+1 0.7 8e6 1e-6 9e+3 90e-6 10e+5 100e-6

In the above example, the first number is the frequency value and the second number is its dwell time. Therefore, only even number of sets can be located in this table.

Parameters

Name	Type	Description
<header>	Discrete	FHOP data header, defines the length of the binary <data_array> in bytes.
<var_hop_data>	Double	Block of binary data array of (2 to 256) double-float value pairs that contains information of frequency hop values and their respective dwell time.

Example

Command `:FHOP:VAR:DATA#210`
`<data_array>`

:FHOP:MARKer<index>(?)

Description

Programs where on the frequency list the SE5082 will generate a pulse, designated as Hop marker, or index point. The marker pulse is generated at the SYNC output connector.

Parameters

Name	Range	Type	Default	Description
<index>	1 to FHOP list length	Numeric (integer only)	1	Programs a marker pulse at an index frequency hop position.

Response

The SE5082 will return the present marker position.

Example

Command `:FHOP:MARK 16`

Query `:FHOP:MARK?`

Amplitude Hopping Modulation Programming

Use the following command for programming the amplitude hop parameters. Hop control is internal. The amplitude will hop from amplitude level to amplitude level at a rate determined by the dwell time value and controlled by a sequence of amplitudes in the HOP data table.

There are two hop modes: Fixed Dwell, where the rate of which the generator hops from amplitude level to amplitude level is constant and Variable Dwell, where the rate of which the generator hops from amplitude level to amplitude level is programmable for each hop.

The commands for programming the amplitude hopping function are described below. Note that the carrier waveform frequency (CW) setting is common to all modulation schemes.

:AHOP:DWELl:MODe(FIXed|VARiable}(?)

Description

This selects between fixed or variable dwell-time for the amplitude hops. Select the fixed option if you want each amplitude level to dwell equally on each step. The variable option lets you program a different dwell time value for each amplitude hop. The SE5082 output hops from one amplitude level to the next according to a sequence given in a hop table. The variable dwell time table contains dwell time data for each step however, the fixed dwell time table does not contain any dwell time information and therefore, if you select the fixed option, make sure your dwell time is programmed as required.

Parameters

Name	Type	Default	Description
FIXed	Discrete		Selects the fixed dwell time amplitude hops mode
VARiable	Discrete	VAR	Select the variable dwell time amplitude hops mode

Response

The SE5082 will return FIX, or VAR depending on the selected dwell setting.

Example

Command `:AHOP:DWEL:MODE FIX`

Query `:AHOP:DWEL:MODE?`

:AHOP:DWELl<dwell_time>(?)

Description

This selects the dwell time for amplitude hops when the selected mode is Fixed dwell time hops. The dwell time table in this case does not contain the dwell time per step parameters and therefore, the value which is programmed with this command remains constant for the entire hop sequence.

Parameters

Name	Range	Type	Default	Description
<dwell_time>	1e-9 to 10	Numeric	5e-6	Programs dwell time for the fixed dwell-time amplitude hop function. The same dwell time will be valid for each amplitude hop. Dwell time is programmed in units of s and should not exceed 10 / (length of the "fixed" AHOP list).

Response

The SE5082 will return the present dwell time value. The returned value will be in standard scientific format (for example: 100 ms would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:AHOP:DWEL 1e-3`

Query `:AHOP:DWEL?`

:AHOP:FIX:DATA#<header><fix_hop_data>

Description

This command will download the data array that will cause the instrument to hop through the amplitude list. The dwell time for each amplitude list item is fixed and can be programmed using the HOP:DWEL command. Note that if you intend to program marker position, you must do it first and then load the amplitude hops list.

Below you can see how a hop table is constructed. The file sample below shows a list of 10 amplitudes. The SE5082 will hop through this list, outputting the next amplitude each time it hops. Note that the last cycle is always completed even if the dwell time is shorter than the period of the waveform. For example, if you program dwell time of 1ms and the amplitude step has frequency of 1Hz (1s period), the frequency step will last 1 second although the dwell time is 1ms. The total dwell-time (fixed_dwell_time * list_length) must not exceed 10 seconds. Also note, that the amplitude limits are different depending on the installed output module.

Sample Amplitude Hops Data Array

0 1 1.2 1.3 1.4 1.5 100e-3 200e-3 300e-3 400e-3 500e-3

Parameters

Name	Type	Description
<header>	Discrete	AHOP header, defines the length of the binary <data_array> in bytes.
<fix_hop_data>	Double	Block of binary data array of (2 to 256) double-float values that contains information of amplitude values.

Example

Command `:AHOP:FIX:DATA#210`

`<data_array>`

:AHOP:VARiable:DATA#<header><var_hop_data>

Description

This command will download the data array that will cause the instrument to hop through the amplitude list. The dwell time for each amplitude list item is variable and is supplied in the variable hop table data array. Note that the HOP:DWEL command has no effect on this sequence. Also note that if you intend to program marker position, you must do it first and then load the amplitude hops list.

Below you can see how a hop table is constructed. The file sample below shows a list of 10 amplitudes and their associated dwell times. The SE5082 will hop through this list, outputting the next amplitude each time it hops. Note that the last cycle is always completed even if the dwell time is shorter than the period of the waveform. For example, if you program dwell time of 1ms and the amplitude step has frequency of 1Hz (1s period), the amplitude step will last 1 second although the dwell time is 1ms. Also note, that the amplitude limits are different depending on the installed output module.

Sample Amplitude Hops Data Array

0.1 5e-6 1.2 10e-6 1.3 20e-6 1.4 50e-6 1.5 1e-6 100e-3 100e-9 200e-3 200e-9 300e-3 300e-9 400e-3 400e-9 500e-3 500e-9

In the above example, the first number is the amplitude value and the second number is its dwell time. Therefore, only even number of sets can be located in this table.

Parameters

Name	Type	Description
<header>	Discrete	AHOP header, defines the length of the binary <data_array> in bytes.
<var_hop_data>	Double	Block of binary data array of (2 to 256) double-float value pairs that contains information of amplitude hop values and their respective dwell time.

Example

Command `:AHOP:VAR:DATA#210`
`<data_array>`

:AHOP:MARKer<index>(?)

Description

Programs where on the amplitude list the SE5082 will generate a pulse, designated as Hop marker, or index point. The marker pulse is generated at the SYNC output connector.

Parameters

Name	Range	Type	Default	Description
<index>	1 to	Numeric	1	Programs a marker pulse at an index amplitude hop

AHOP list length	(integer only)	position.

Response

The SE5082 will return the present marker position.

Example

Command `:AHOP:MARK 22`

Query `:AHOP:MARK?`

Pulse Waveform Programming

Use the following commands for programming pulse waveforms and pulse parameters. The pulse is created digitally. However, it closely simulates an analog pulse generator, so pulse parameters are programmed just as they would be programmed on a dedicated pulse generator instrument. Just keep in mind that since this is a digital instrument, there are some limitations to the pulse design that evolve from the fact that the best resolution is one sample clock interval. Also, remember that the pulse is created digitally in the arbitrary memory and therefore, its smallest incremental step has a maximum value limitation as specified in Appendix A. Figures 4-12 through 4-14 show how the pulse commands affect the shape of the pulse when placed in Single, Delayed and Double pulse modes. The pulse commands are summarized in Table 4-12.

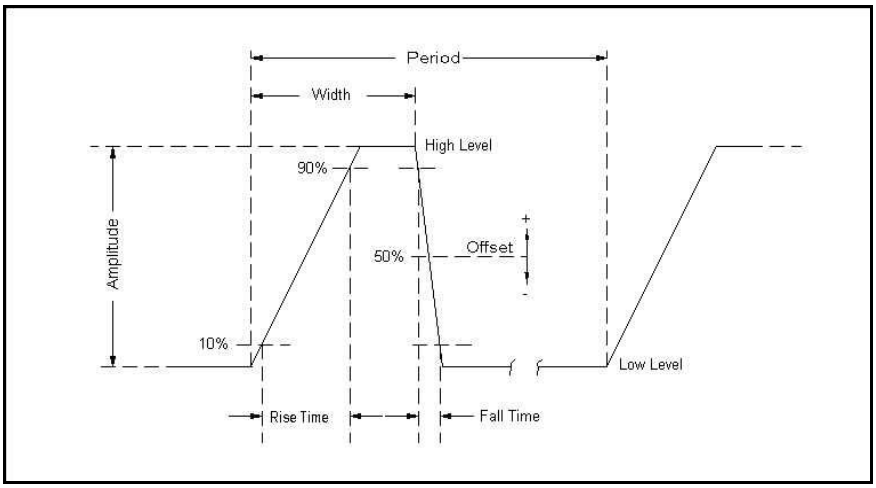

Figure 4-9, Single Pulse Parameters

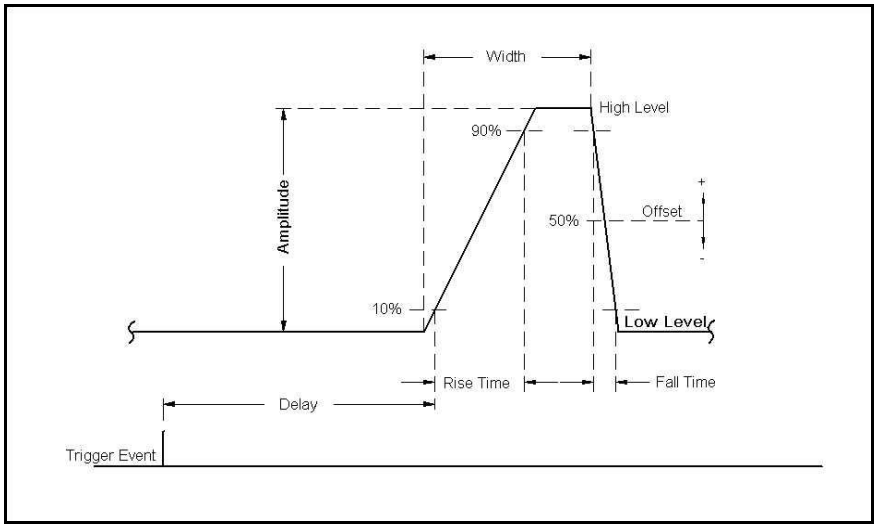

Figure 4-10, Delayed Pulse Parameters

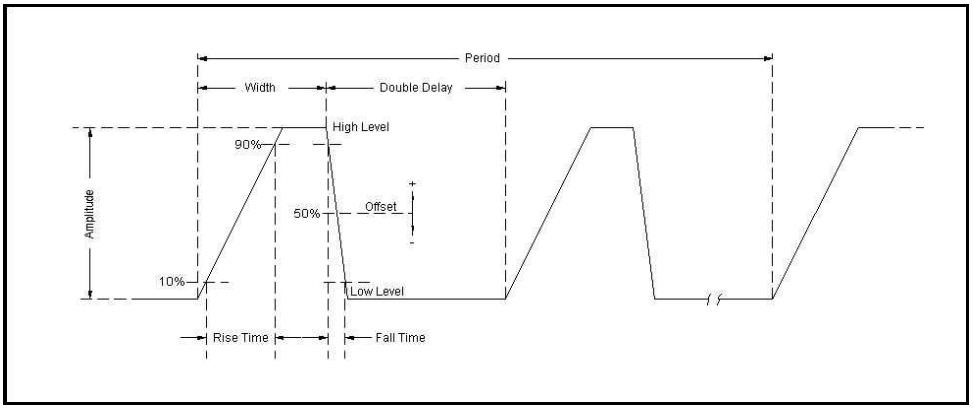

Figure 4-11, Double Pulse Parameters

Table 4-12, Pulse Waveform Commands Summary

Keyword	Parameter Form	Default	Notes
:PULSe			
:CONFigure	TIME | PERCent	TIME	
:DELay	0, 0.2e-9 to 1.6-0.2e-9	1e-6	Delay in seconds.
:PERCent	0 .01 to 99.99	10	Delay in %
:DOUBle			
:DELay	0, 0.2e-9 to 1.6-0.2e-9	1e-6	Double delay in seconds
:PERCent	0 .01 to 99.99	10	Delay in %
:LEVel			
[:CONTrol]	HLOW | AOFFset | POSitive | NEGative	HLOW	
:HIGH	-0.75 to +0.75 for DC amplifier	0.5	
	-1.5 to +1.5 for HV amplifier	0.5	
	-0.27 to +0.27 for DR (DAC output)	0.25	
:LOW	-0.75 to +0.75 for DC amplifier	0	
	-1.5 t0 +1.5 for HV amplifier	0	
	-0.27 to +0.27 for DR (DAC output)	-0.25	
:AMPLitude	100e-3 to 1.0 for DC amplifier	0.5	Amplitude in volts. Also accepts MINimum | MAXimum
	50e-3 to 2.0 for HV amplifier	0.5	
	50e-3 to 0.54 for DR (DAC output)	0.5	
:OFFSet	-0.5 to 0.5 for DC amplifier	0.25	DC offset in volts Also accepts MINimum | MAXimum
	-1.0 to 1.0 for HV amplifier	0.2	
	0 for DR (DAC output)	0	
:MODE	SINGle | DELayed | DOUBle	SING	
:POLarity	NORMal | COMPlement | INVerted	NORM	
:PERiod	0.8e-9 to 1.6	10e-6	Total pulse period in seconds
:TRANsition			
:STATe	FAST | LINear | SYMMetrical	FAST	Transition type
[:LEADing]	Between Ɛ and 1.6-Ɛ, Ɛ=0.2e-9	1e-6	Rise time in sec
:PERCent	0 .01 to 99.99	10	Rise time in %
:TRAiling	Between Ɛ and 1.6-Ɛ, Ɛ=0.2e-9	1e-6	Fall time in sec
:PERCent	0 .01 to 99.99	10	Fall time in %
:WIDTh	Between Ɛ and 1.6-Ɛ, Ɛ=0.2e-9	2e-6	Width in sec
:PERCent	0 .01 to 99.99	20	Width in %

:PULSe:CONFigure{TIME|PERCent}(?)

Description

In applications where an absolute setting of a certain pulse parameter is required, it is recommended to select the time option. This means that the parameter can be programmed individually in units of time and will retain the programmed value, regardless of the pulse period setting. The percent option is useful for applications that require a certain relationship between the period and the individual parameter, such as width-to-period or rise-time-to-period. Use this command to configure the generator for one of these options.

Parameters

Name	Type	Default	Description
TIME	Discrete	TIME	Enables the programming of pulse parameters in units of time (seconds)
PERCent	Discrete		Enables the programming of certain pulse parameters in units of percent. The ratio is computed in reference to the period frequency. For example, if the period is set to 100 ms and the width is programmed to 10%, the width will remain 10% of the programmed period, (10 ms in this example) but will be automatically adjusted to 15 ms if the period is changed to 150 ms.

Response

The SE5082 will return TIME or PERC depending on the present pulse configuration setting.

Example

Command `:PULS:CONF PERC`

Query `:PULS:CONF?`

:PULSe:DELay<delay>(?)

Description

This command programs the delayed interval of which the output idles on the low level, until the first transition to high level. The delay is measured from the trigger transition to the first pulse transition. Note that this delay does not include the system delay error that is specified in Appendix A. Also note that the only case where the delay can exceed the value of the period setting is in triggered mode, where the external trigger intervals determine the period of the pulse.

Parameters

Name	Range	Type	Default	Description
<delay>	0, 0.2e-9 to (1.6-0.2e-9)	Numeric	1e-3	Will set the delay time interval in units of seconds. As shown in Figure 4-13, the delay is measured from trigger transition to the first pulse transition. PLUS:DEL 0 implies that delay is off.

Response

The SE5082 will return the pulse delay value in units of seconds.

Example

Command `:PULS:DEL 1e-3`

Query `:PULS:DEL?`

:PULSe:DELay:PERCent<delay>(?)

Description

This command will program the delayed interval of which the output idles on the low level, until the first transition to high level. The delay is measured from the first external trigger transition to the first pulse transition. Note that this delay does not include the system delay error that is specified in Appendix A. Also note that the only case where the delay can exceed the value of the period setting is in triggered mode, where the external trigger intervals determine the period of the pulse.

Parameters

Name	Range	Type	Default	Description
<delay>	0.01 to 99.99	Numeric	10	Will set the delay time interval in units of percent. As shown in Figure 4-13, the delay is measured from trigger transition to the first pulse transition.

Response

The SE5082 will return the pulse delay value in units of %.

Example

Command `:PULS:DEL:PERC 10`

Query `:PULS:DEL:PERC?`

:PULSe:DOUBle:DELay<delay>(?)

Description

This command will program the delay between two adjacent pulses when the double mode is selected. Otherwise, the double pulse delay has no effect on the pulse structure. Note that the only case where the delay can exceed the value of the period setting is in triggered mode, where the trigger interval determines the period of the pulse. Double pulse building block parameters are shown in Figure 4-14.

Parameters

Name	Range	Type	Default	Description
<delay>	0, 0.2e-9 to (1.6-0.2e-9)	Numeric	1e-3	Will set the delay between two adjacent pulses for the double pulse mode only. The delay is

	programmed in units of seconds and is measured from the last transition of the first pulse to the first transition of the second pulse. PULS:DOUB:DEL 0 implies that delay is off.

Response

The SE5082 will return the present double pulse delay value in units of seconds.

Example

Command `:PULS:DOUB:DEL 10e-3`

Query `:PULS:DOUB:DEL?`

:PULSe:DOUBle:DELay:PERCent<delay>(?)

Description

This command will program the delay between two adjacent pulses when the double mode is selected. Otherwise, the double pulse delay has no effect on the pulse structure. Note that the only case where the delay can exceed the value of the period setting is in triggered mode, where the trigger interval determines the period of the pulse. Double pulse building block parameters are shown in Figure 4-14.

Parameters

Name	Range	Type	Default	Description
<delay>	0.01 to 99.99	Numeric	10	Will set the delay between two adjacent pulses for the double pulse mode only. The delay is programmed in units of percent and is measured from the last transition of the first pulse to the first transition of the second pulse.

Response

The SE5082 will return the present double pulse delay value in units of percent.

Example

Command `:PULS:DOUB:DEL:PERC 15`

Query `:PULS:DOUB:DEL:PERC?`

:PULSe:LEVel{HLOW|AOFFset|POSitive|NEGative} (?)

Description

This command will determine how the pulse levels are programmed. Pulse level mode options are: High/Low, Amplitude/Offset, Positive and Negative.

Parameters

Name	Type	Default	Description
HLOW	Discrete	HLOW	Programs pulse level using high level and low level parameters.
AOFFset	Discrete		Programs pulse level using the amplitude and offset parameters. When the offset is left at 0 V settings (default), amplitude changes modifies the pulse level symmetrically about the 0 V level.
POSitive	Discrete		Programs pulse level using the high level parameters. The low level remains at 0 V at all times.
NEGative	Discrete		Programs pulse level using the low level parameters. The high level remains at 0 V at all times.

Response

The SE5082 will return HLOW, AOFF, POS or NEG depending on the present pulse level mode setting.

Example

Command `:PULS:LEV AOFF`

Query `:PULS:LEV?`

:PULSe:LEVel:HIGH<high_level>(?)

Description

This command programs the high level amplitude of the pulse waveform. The high level is calibrated when the load impedance is 50 Ω and the range varies depending on output module installed.

Parameters

Name	Range (per module)	Type	Default	Description
<high_level>	DC – -0.75 to +0.75	Numeric	0.5	Will set the high level of the pulse waveform in units of volts.
	HV - -1.5 to +1.5		0.5	
	DR – -0.27 to +0.27		0.25	

Response

The SE5082 will return the present high-level value. The returned value will be in standard scientific format (for example: 100mV would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:PULS:LEV:HIGH 1`

Query `:PULS:LEV:HIGH?`

:PULSe:LEVel:LOW<low_level>(?)

Description

This command programs the low level amplitude of the pulse waveform. The low level is calibrated when the load impedance is 50 Ω.

Parameters

Name	Range	Type	Default	Description
<low_level>	DC – -0.75 to +0.75	Numeric	0	Will set the low level of the pulse waveform in units of volts.
	HV - -1.5 to +1.5		0	
	DR - -0.27 to +0.27		-0.25	

Response

The SE5082 will return the present low-level value. The returned value will be in standard scientific format (for example: 100mV would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:PULS:LEV:LOW 0.2`

Query `:PULS:LEV:LOW?`

:PULSe:LEVel:AMPLitude{<ampl>|MINumum|MAXimum}(?)

Description

This command programs the peak-to-peak amplitude of the pulse waveform. The amplitude is calibrated when the source impedance is 50 Ω. Note that this value is a duplication of the volt:ampl parameter and therefore, modifying this parameter in the pulse menu will automatically modify the amplitude setting for the other instrument functions.

Parameters

Name	Range	Type	Default	Description
<ampl>	DC – 100e-3 to +1.2	Numeric	0.5	Will set the amplitude of the pulse waveform in units of volts. Amplitude setting is always peak-to-peak. Offset and amplitude settings are independent, providing that the offset + amplitude do not exceed the amplitude window, as specified in Appendix A.
	HV – 50e-3 to +2		0.5	
	DR – 0.05 to 0.54		0.5	
<MINimum>	Discrete			Will set the amplitude to the lowest possible amplitude.
<MAXimum>	Discrete			Will set the amplitude to the highest possible amplitude level.

Response

The SE5082 will return the present pulse amplitude value. The returned value will be in standard scientific format (for example: 100mV would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:PULS:LEV:AMPL 0.6`

Query `:PULS:LEV:AMPL?`

:PULSe:LEVel:OFFset{<offs>|MINimum|MAXimum}(?)

Description

This command programs the offset amplitude of the pulse waveform. The offset is calibrated when the source impedance is 50Ω. Note that this value is a duplication of the volt:offs parameter and therefore, modifying this parameter in the pulse menu will automatically modify the offset setting for the other instrument functions.

Parameters

Name	Range	Type	Default	Description
<offs>	DC – -0. 5 to +0.5 HV - -1 to +1 DR - 0	Numeric	0 0 0	Will set the offset of the pulse waveform in units of volts. Offset and amplitude settings are independent providing that the offset + amplitude do not exceed the specified window.
<MINimum>	Discrete			Will set the amplitude to the lowest possible offset.
<MAXimum>	Discrete			Will set the amplitude to the highest possible offset level.

Response

The SE5082 will return the present pulse offset value. The returned value will be in standard scientific format (for example: 100mV would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:PULS:LEV:OFFS 0.5`

Query `:PULS:LEV:OFFS?`

:PULSe:MODE{SINGle|DELayed|DOUBle}(?)

Description

This command will program the mode of the pulse. Pulse mode options are: Single pulse, Delayed pulse and Double pulse. These modes are depicted in Figures 4-12 through 4-14.

Parameters

Name	Type	Default	Description
SINGle	Discrete	SING	Programs single pulse output, which generates normal pulses
DELayed	Discrete		Programs a delayed pulse mode, which generates normal pulses that are delayed from the SYNC output

DOUBle	Discrete	Programs a double pulse mode, which generates a pair of single pulses that are displaced by a double delay period

Response

The SE5082 will return SING, DEL or DOUB depending on the present pulse mode setting.

Example

Command `:PULS:MODE DOUB`

Query `:PULS:MODE:?`

:PULSe:POLarity{NORMal|COMPlement|INVerted}(?)

Description

This command will program the polarity of the pulse in reference to the baseline level. The polarity options are: Normal - where the pulse is generated exactly as programmed, Inverted - where the pulse is inverted about the 0 level base line and Complemented - where the pulse is inverted about its mid-amplitude axis.

Parameters

Name	Type	Default	Description
NORMal	Discrete	NORM	Programs normal pulse output
COMPlement	Discrete		Programs complemented pulse output
INVerted	Discrete		Programs an inverted pulse output

Response

The SE5082 will return NORM, COMP or INV depending on the present polarity setting.

Example

Command `:PULS:POL COMP`

Query `:PULS:POL?`

:PULSe:PERiod<period>(?)

Description

This command will program the pulse repetition rate (period). Note that with the sum of all parameters, including the pulse width and fall time cannot exceed the programmed pulse period and therefore, it is recommended that the pulse period be programmed first, before all other pulse parameters. Note that by selecting the double pulse mode, the pulse period remains unchanged.

Parameters

Name	Range	Type	Default	Description

<period>	0.8e-9 to 1.6	Numeric	10e-6	Programs the period of the pulse waveform in units of seconds.

Response

The SE5082 will return the present pulse period value in units of seconds.

Example

Command `:PULS:PER 1e-3`

Query `:PULS:PER?`

:PULSe:TRANsition:STATe{FAST|LINear|SYMMetrical} (?)

Description

This command will place the pulse output in one of three transition options: 1) Fast - the level transitions from high-to-low or low-to-high at the fastest rate that the generator can produce 2) Linear - the level transitions linearly at a programmed rate from low-to-high and from high-to-low and each transition can be programmed to a different rate and 3) Symmetrical - the leading and trailing edges transition exactly at the same rate.

Parameters

Name	Type	Default	Description
FAST	Discrete	FAST	Programs the fast transitions mode. In this mode, the leading and trailing edges will transition as fast as the instrument allows.
LINear	Discrete		Selects linear transitions. Note that unlike analog pulse generators, the SE5082 can be programmed freely within the specified boundaries, without the limitation of transition ranges commonly attributed to analog pulse generators.
SYMMertical	Discrete		Programs a special mode where the transitions are symmetrical for both the leading and trailing edges (lead time = trail time). In this case, the value of the lead-time is automatically attributed to the trail time and the value of the trail time has no effect on the pulse shape.

Response

The SE5082 will return FAST, LIN or SYMM depending on the present transition mode setting.

Example

Command `:PULS:TRAN:STAT LIN`

Query `:PULS:TRAN:STAT?`

:PULSe:TRANsition<l_edge>(?)

Description

This command will program the interval it will take the leading edge of the pulse to transition from its low- to high-level settings. The parameter is programmed in units of seconds. Unlike analog pulse generators, the SE5082 can be programmed freely within the specified boundaries, without the limitation of transition ranges that are commonly attributed to analog pulse generators. Note that this parameter will affect the instrument only when the pulse transition mode is set to linear.

Parameters

Name	Range	Type	Default	Description
<l_edge>	0.2e-9 to (1.6-0.2e-9)	Numeric	1e-6	Will set the leading-edge transition time parameter in units of seconds. Note that the sum of all parameters, including transition times, must not exceed the programmed pulse period.

Response

The SE5082 will return the present leading-edge transition time value in units of seconds.

Example

Command `:PULS:TRAN 100e-9`

Query `:PULS:TRAN?`

:PULSe:TRANsition:PERCent<l_edge>(?)

Description

This command will program the interval it will take the leading edge of the pulse to transition from its low- to high-level settings. The parameter is programmed in units of percent of the pulse period. Note that this parameter will affect the instrument only when the pulse transition mode is set to linear and when the pulse configuration is programmed to percent.

Parameters

Name	Range	Type	Default	Description
<l_edge>	0.01 to 99.99	Numeric	10	Will set the leading-edge transition time parameter in units of %. Note that the sum of all parameters, including transition times must not exceed 100%.

Response

The SE5082 will return the present leading-edge transition time value in units of %.

Example

Command `:PULS:TRAN:PERC 1`

Query `:PULS:TRAN:PERC?`

:PULSe:TRANsition:TRAiling<t_edge>(?)

Description

This command will program the interval it will take the trailing edge of the pulse to transition from its high- to low-level settings. The parameter is programmed in units of seconds. Unlike analog pulse generators, the SE5082 can be programmed freely within the specified boundaries, without the limitation of transition ranges that are commonly attributed to analog pulse generators. Note that this parameter will affect the instrument only when the pulse transition mode is set to linear.

Parameters

Name	Range	Type	Default	Description
<T_edge>	0.2e-9 to (1.6-0.2e-9)	Numeric	1e-6	Will set the trailing-edge transition time parameter in units of seconds. Note that the sum of all parameters, including transition times, must not exceed the programmed pulse period.

Response

The SE5082 will return the present trailing edge transition time value in units of seconds.

Example

Command `:PULS:TRAN:TRA 10e-6`

Query `:PULS:TRAN:TRA?`

:PULSe:TRANsition:TRAiling:PERCent<t_edge>(?)

Description

This command will program the interval it will take the trailing edge of the pulse to transition from its high- to low-level settings. The parameter is programmed in units of percentages of the pulse period. Note that this parameter will affect the instrument only when the pulse transition mode is set to linear and when the pulse configuration is programmed to percent.

Parameters

Name	Range	Type	Default	Description
<l_edge>	0.01 to 99.99	Numeric	10	Will set the trailing-edge transition time parameter in units of %. Note that the sum of all parameters, including transition times, must not exceed 100%.

Response

The SE5082 will return the present leading edge transition time value in units of %.

Example

Command `:PULS:TRAN:TRA:PERC 10`

Query `:PULS:TRAN:TRA:PERC?`

:PULSe:WIDTh<width>(?)

Description

This command will program the pulse width value. Figures 4-12 through 4-14 show how the pulse width affects the shape of the pulse. Note that the only case where the pulse width can exceed the value of the period setting is in triggered mode, where the trigger determines the period of the pulse.

Parameters

Name	Range	Type	Default	Description
<width>	0.2e-9 to (1.6-0.2e-9)	Numeric	2e-6	Will set the width of pulse in units of seconds. Note that the sum of all parameters, including the pulse width, must not exceed the programmed pulse period.

Response

The SE5082 will return the present pulse width value in units of seconds.

Example

Command `:PULS:WIDT 500e-6`

Query `:PULS:WIDT?`

:PULSe:WIDTh:PERCent<width>(?)

Description

This command will program the pulse width value. Figures 4-12 through 4-14 show how the pulse width affects the shape of the pulse. Note that this parameter will affect the instrument only when the pulse transition mode is set to linear and when the pulse configuration is programmed to percent.

Parameters

Name	Range	Type	Default	Description
<l_edge>	0.01 to 99.99	Numeric	20	Will set the pulse width parameter in units of %. Note that the sum of all parameters, including transition times, must not exceed 100%.

Response

The SE5082 will return the present leading edge transition time value in units of %.

Example

Command `:PULS:WIDT:PERC 20`

Query `:PULS:WIDT:PERC?`

Pulse Pattern Programming

Use the following commands for programming pulse pattern waveforms and their associated parameters. The patterns are created digitally. However, they closely simulate an analog pulse generator, so pulse pattern parameters are programmed just as they would be programmed on a dedicated pattern generator instrument. Just keep in mind that since this is a digital instrument, there are some limitations to the pattern design that evolve from the fact that the best resolution is one sample clock interval. Also, remember that the patterns are created digitally in the arbitrary memory and therefore, their smallest incremental step has a maximum value limitation, depending on the memory length, as specified in Appendix A.

Major consideration has been given to build in the capability to design special and complex pulse patterns, such as user-defined pulse patterns and Pseudo Random Bit Sequence.

The SE5082 pattern functionality is implemented as an extension of the Pulse Mode. In addition to RZ pulses, it allows the generation of pattern sequences that are using NRZ formatting with adjustable transition times. The pattern generation itself allows the definition of pattern sequences with 2, 3, 4 or 5 different level settings, which allows the emulation of electrical idle sequences as being required in several serial data protocols.

PRBS pattern is sequence of random patterns that can be used for testing and stressing bit receivers. PRBS is “pseudo” because it is deterministic and after N elements it starts to repeat itself, unlike real random sequences, such as sequences generated by radioactive decay or by white noise. PRBS's are used in telecommunication, encryption, simulation, correlation techniques and time-of-flight spectroscopy.

Figure 4-15 show an example of a digital pulse pattern that employs mixed fast and linear transitions. Figures 4-16 through 4-19 show the effect of level settings on the PRBS pattern. Commands and their associated definitions are given in the following. Note that the number of levels that are used during the pattern generation is being defined when creating a new pattern and cannot be changed afterwards.

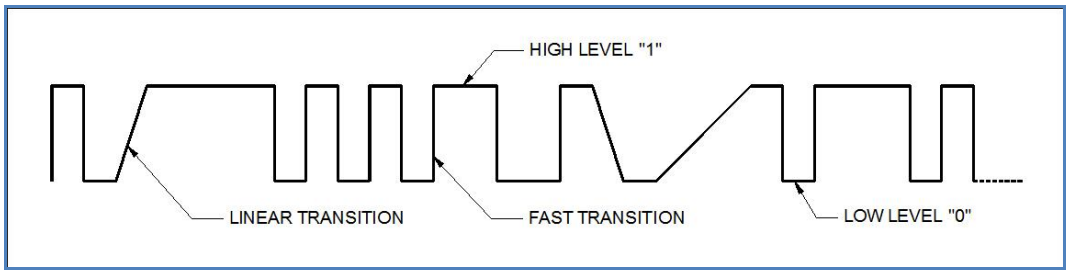

Figure 4-12, Composed Pulse Pattern with Mixed Fast and Linear transitions

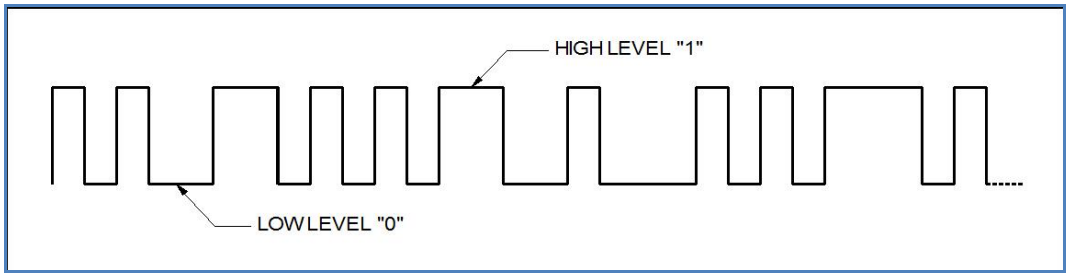

Figure 4-13, PRBS 2 Level Pattern Example

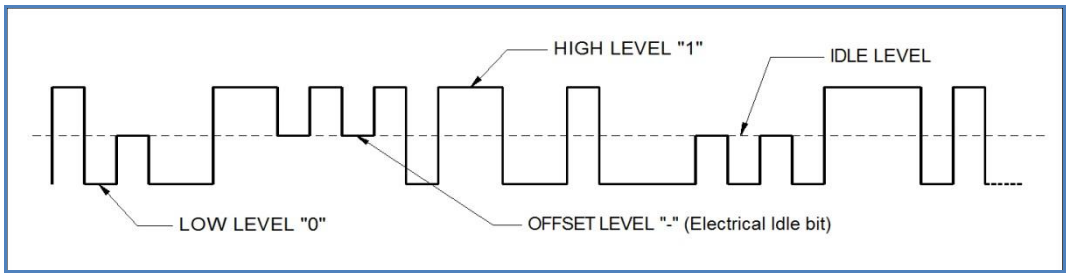

Figure 4-14, PRBS 3 Level Pattern Example

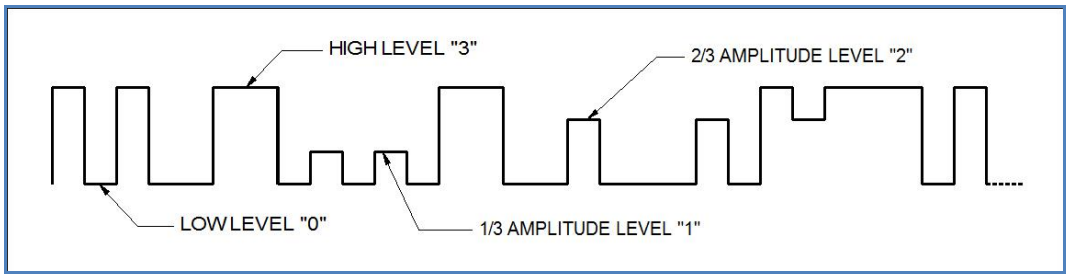

Figure 4-15, PRBS 4 Level Pattern Example

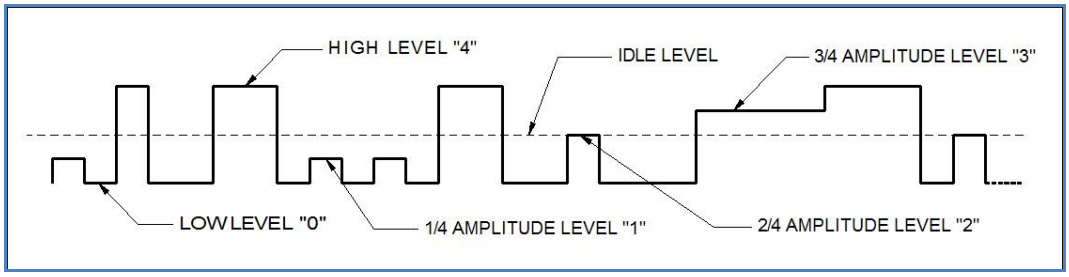

Figure 4-16, PRBS 5 Level Pattern Example

Table 4-13, Pulse Pattern Commands Summary

Keyword	Parameter Form	Default	Notes
:PATTern			
:MODE	PRBS | COMPoser		
[:PRBS]			
:TYPE	PRBS7 | PRBS9 | PRBS11 | PRBS15 | PRBS23 | PRBS31 | USER	PRBS7	
:BAUD	1 to 1e+9	10e6	
:LEVel	2 | 3 | 4 | 5	2	
:HIGH	-0.75 to 0.75 for DC amplifier	0.5	
	-1.5 to 1.5 for HV amplifier	0.5	
	-0.27 to +0.27 for DR (DAC output)	0.25	
:LOW	-0.75 to 0.75 for DC amplifier	0	
	-1.5 to 1.5 for HV amplifier	0	
	-0.27 to +0.27 for DR (DAC output)	-0.25	
:LOOPs	1 to 1e6	1	
:PREamble	1 to 16e6	1	
:LENGth	1 to 16e6	32	
:DATA	#<header><binary-data> The binary data is of N bytes that hold the user-defined symbols-list of the PRBS composer. Each byte holds a single ASCII character from '0', '1', '2', '3' or '-', that defines a single symbol.		Download the symbols list of the USER-PRBS pattern. Maximal length of the symbols list is 16e+6
:COMPoser			
:TRANsition			
:TYPe	FAST | LINear	LINear	Transition type
:FAST			
[:DATA]	#<header><binary-data> The binary-data is array of N*12 bytes holding the definitions of the N flat-intervals of the Fast-Pattern, Where each 12-bytes sub-block consists: DC-Level as float value (4-bytes) between (-maxV_{offs} - maxV_{pp}/2) and (+maxV_{offs} +maxV_{pp}/2) Dwell-Time as double-float (8-bytes)		Download the definitions of the N flat-intervals of the Fast-Pattern. Difference between the highest and lowest levels should not exceed maxV_{pp} The G.C.D of all dwell-times should be at least 0.2e-9 (seconds).

Table 4-16, Pulse Pattern Commands Summary (Continued)

Keyword	Parameter Form	Default	Notes
:LINear			
:STARt	(-maxV_{offs} - maxV_{pp}/2) to (+maxV_{offs} + maxV_{pp}/2)	0.25 (volt)	The start-level (in volts) of the first linear-interval in the Linear-Pattern.
[:DATA]	#<header><binary-data> The binary-data is array of N*12 bytes holding the definitions of the N linear-intervals of the Piecewise-Linear-Pattern, where each 12-bytes sub-block consists: Ending-Voltage-Level as float value (4-bytes) between (-maxV_{offs} - maxV_{pp}/2) and (+maxV_{offs} +maxV_{pp}/2) Difference between the highest and lowest levels should not exceed maxV_{pp} Dwell-Time as double-float (8-bytes)		Download the definitions of the N linear-intervals of the Piecewise-Linear-Pattern. The start voltage level of each interval is ending voltage level of the previous interval (the start level of the first interval is set separately). The G.C.D of all –non-zero- dwell-times should be at least 0.2e-9 (seconds). Note that zero dwell-time is allowed and can be used for level jumps)
:RESolution	0.2e-9 to 100e-9	0.2e-9 sec	The time-resolution (the sampling time of each waveform point)
:TYPE	AUTO | USER	AUTO	AUTO means that the time resolution is automatically selected. USER means that it is set manually by the user . In the later case, it should divide the G.C.D of all the pattern's dwell-times.

:PATTern:MODE{PRBS|COMPosed}(?)

Description

Use this command to set or query the type of pulse pattern that will be generated by the SE5082 output. There are 6 PRBS types and an additional user-defined PRBS type. The composed option provides access to special commands that lets you create any pattern and mix fast and linear transitions.

Parameters

Name	Type	Default	Description
PRBS	Discrete	PRBS	Selects one of 6 internal and one user-defined PRBS sequences. Patt:type selects the required PRBS type.
COMPoser	Discrete		Selects the free-programming pattern mode. Use this option if you intend to create complex digital patterns. These patterns can have fast or linear transitions, depending on the patt:comp:tran:type setting.

Response

The SE5082 will return PRBS or COMP depending on the present pattern mode setting.

Example

Command `:PATT:MODE PRBS`

Query `:PATT:MODE?`

:PATTern[:PRBS]:TYPE{ PRBS7 | PRBS9 | PRBS11 | PRBS15 | PRBS23 | PRBS31 | USER }(?)

Description

Use this command to set or query the type of PRBS sequence that will be generated by the SE5082 output. There are 6 PRBS types and an additional user-defined PRBS type. The sequence is generated by an internal PRBS generator. In continuous run mode, the sequence is repeated continuously but in triggered mode, the sequence is repeated until the patt:loop number has been reached.

Parameters

Name	Type	Default	Description
PRBS7	Discrete	PRBS7	Selects pseudo-random bit sequence that is generated internally from a polynomial series of 2e7 order.
PRBS9	Discrete		Selects pseudo-random bit sequence that is generated internally from a polynomial series of 2e9 order.
PRBS11	Discrete		Selects pseudo-random bit sequence that is generated internally from a polynomial series of 2e11 order.
PRBS15	Discrete		Selects pseudo-random bit sequence that is generated internally from a polynomial series of 2e15 order.
PRBS23	Discrete		Selects pseudo-random bit sequence that is generated internally from a polynomial series of 2e23 order.
PRBS31	Discrete		Selects pseudo-random bit sequence that is generated internally from a polynomial series of 2e31 order.
USER	Discrete		User programmable pseudo-random bit sequence that is generated from an external data file.

Response

The SE5082 will return PRBS7, PRBS9, PRBS11, PRBS15, PRBS23, PRBS31, or USER depending on the present pattern type setting.

Example

Command `:PATT:TYPE PRBS11`

Query `:PATT:TYPE?`

:PATTern[:PRBS]:BAUD<baud >(?)

Description

Use this command to set or query the PRBS baud. PRBS baud is the rate of which each bit is sampled.

Parameters

Name	Range	Type	Default	Description
<baud>	1 to 1e9	Numeric	10e6	Will set the PRBS baud in units of bits per second.

Response

The SE5082 will return the present baud value. The returned value will be in standard scientific format (for example: 100 s would be returned as 100 – positive numbers are unsigned).

Example

Command `:PATT:BAUD 100e6`

Query `:PATT:BAUD?`

:PATTern[:PRBS]:LEVel<level>(?)

Description

Use this command to set or query the PRBS voltage level setting. The effect of the level setting is demonstrated in Figures 4-16 to 4-19. The PRBS high and low levels determine the maximum amplitude swing while the PRBS levels parameter determine interim symbol levels.

Parameters

Name	Range	Type	Default	Description
<level>	2 to 5	Numeric (integer only)	2	Will set the symbol level, as demonstrated in Figures 4-16 to 4-19.

Response

The SE5082 will return the present PRBS symbol level value.

Example

Command `:PATT:LEV 4`

Query `:PATT:LEV?`

:PATTern[:PRBS]:LEVel:HIGH<high_level>(?)

Description

Use this command to set or query the high level voltage for the PRBS pattern. The pattern amplitude will swing from high to low level setting. Please note the setttings range and default value vary depending on the installed output module

Parameters

Name	Range	Type	Default	Description
<high_level>	DC – -0.75 to +0.75 HV - -1.5 to +1.5 DR – 0.27 to +0.27	Numeric	0.5 0.5 0.25	Will set the high level of the PRBS pattern in units of volts.

Response

The SE5082 will return the present PRBS high-level value. The returned value will be in standard scientific format (for example: 100 mV would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:PATT:LEV:HIGH 1`

Query `:PATT:LEV:HIGH?`

:PATTern[:PRBS]:LEVel:LOW<low _level>(?)

Description

Use this command to set or query the low level voltage for the PRBS pattern. The pattern amplitude will swing from high to low level setting.

Parameters

Name	Range	Type	Default	Description
<low_level>	DC – -0.75 to +0.75 HV - -1.5 to +1.5 DR – -0.27 to +0.27	Numeric	0 0 -0.25	Will set the low level of the PRBS pattern in units of volts.

Response

The SE5082 will return the present PRBS low-level value. The returned value will be in standard scientific format (for example: 100 mV would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:PATT:LEV:LOW 0.25`

Query `:PATT:LEV:LOW?`

:PATTern[:PRBS]:LOOPs<cycle>(?)

Description

Use this command to set or query the number of PRBS patterns. Note that in continuous run mode, this parameter has no effect. In triggered run mode, the PRBS will start and repeat itself n times, depending on the loops value and then resume idle position.

Parameters

Name	Range	Type	Default	Description
<cycle>	1 to 1e6	Numeric (integer only)	1	Will set the number of cycles that the PRBS pattern will repeat following a trigger command.

Response

The SE5082 will return the present PRBS loops value.

Example

Command `:PATT:LOOP 1000`

Query `:PATT:LOOP?`

:PATTern[:PRBS]:PREamble<count>(?)

Description

Use this command to set or query the number of PRBS patterns that will be used as preamble. This parameter is useful in triggered run mode where, after triggered, the PRBS preamble sequence will replay once only, followed by PRBS patterns that will repeat continuously or stopped by the count value. Normally, the preamble sequence is used for synchronization purpose and therefore, it must be created and controlled in a user-defined data file.

Parameters

Name	Range	Type	Default	Description
<count>	1 to 16e6	Numeric (integer only)	1	Will set the number of preamble patterns at the beginning of the PRBS sequence.

Response

The SE5082 will return the present PRBS preamble count value.

Example

Command `:PATT:PRE 1e3`

Query `:PATT:PRE?`

:PATTern[:PRBS]:LENGth<length>(?)

Description

Use this command to set or query the length of the PRBS patterns that will be allocated for downloading PRBS data using the patt:data command.

Parameters

Name	Range	Type	Default	Description
<length>	1 to 16e6	Numeric (integer only)	32	Will set the number of memory locations that will be allocated for the PRBS data.

Response

The SE5082 will return the present PRBS memory allocation value.

Example

Command `:PATT:LENG 1e3`

Query `:PATT:LENG?`

:PATTern:DATA#<header><data_array>

Description

This command will download an array of data for the user-defined PRBS pattern. The data is supplied to the generator in a form of binary characters, representing levels only. For example, 0,1,1,1,0,0,1,0 represents level 2 PRBS pattern; similarly, 1,2,0,3,3,-,1,-,2,-,4,0,0 contains all level which are associated with PRBS level 5 pattern. The data is provided as CHAR binary bytes that translate to ASCII levels 2, 3, 4 and 5. The translation to PRBS levels is done automatically by the generator so that users do not have to worry about setting levels as long as high and low levels are programmed prior to downloading the PRBS user data.

Parameters

Name	Type	Description
<header>	Discrete	Provides information on the size of the binary block that follows. Additional information on IEEE-488.2 arbitrary data downloads is available in earlier descriptions of data downloads. Contains PRBS pattern data strings.
<data_array>	Binary	The binary data is of N bytes that hold the user-defined symbols-list of the PRBS composer. Each byte holds a single ASCII character from '0', '1', '2', '3' or '-', that defines a single symbol.

Example

Command `:PATT:DATA#<header><data_array>`

:PATTern:COMPosed:TRANsition:TYPE{FAST|LINear}(?)

Description

Use this command to set or query the type of transitions that the pulse composer will generate. You cannot mix fast and linear transitions if you select the fast mode however, if you select the linear mode, you can still generate pulses that have fast transitions. Note that this command has an effect on the pulse pattern only when the pattern mode option is COMP.

Parameters

Name	Type	Default	Description
FAST	Discrete	FAST	Using this option, the pulse composer can generate pulse patterns that have fast transitions only.
LINear	Discrete		Selects this option if you want to mix fast and linear transitions in the pulse patter that you want to compose.

Response

The SE5082 will return FAST or LIN depending on the present pattern transition setting.

Example

Command `:PATT:COMP:TRAN:TYPE LIN`

Query `:PATT:COMP:TRAN:TYPE?`

:PATTern:COMPosed:FAST#<header><data_array>

Description

This command will download an array of data for the built-in pulse composer. The composed data is made of 12 bytes. Figure 4-20 shows a pulse data parameter sequence that is acceptable for the SE5082.

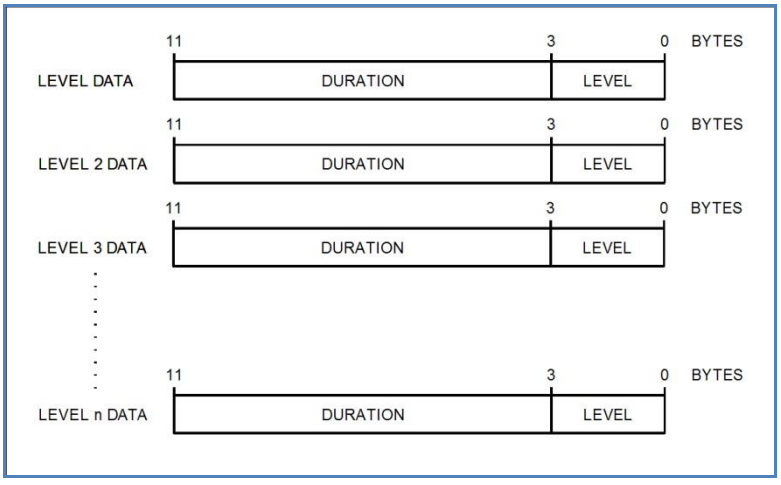

Figure 4-17, Composed "FAST" Pulse - Level Data Representation

Note in the above:

Pulse level data is provided in terms of level and duration only. The pulse will be built by defining the levels and the duration that the level dwells on itself.

Transitions between levels are FAST however, note the fastest transition that the output can generate is around 400 ps

Parameters

Name	Type	Description
<header>	Discrete	Provides information on the size of the binary block that contains pattern parameters data.
<data_array>	Binary	Block of binary data that contains level and duration data, as explained and shown in Figure 4-20 above. The binary-data is array of N*12 bytes holding the definitions of the N flat-intervals of the Fast-Pattern, where each 12-bytes sub-block consists: DC-Level as float value (4-bytes) between (-maxVoffs - maxVpp/2) and (+maxVoffs +maxVpp/2) Dwell-Time as double-float (8-bytes) Difference between the highest and lowest levels should not exceed maxVpp The G.C.D of all dwell-times should be at least 0.2e-9 (seconds).

Example

Command `:PATT:COMP#<data_array>`

:PATTern:COMPosed:LINear:STARt<level>(?)

Description

Use this command to set or query the first level for the composed linear pulse. The first level is mandatory to place the first pulse level. Consequent data set the next pulse level and the duration that it will take for the level to transition from the first to the next level.

Parameters

Name	Range	Type	Default	Description
<level>	DC– -0.6 to +0.6 HV - -1.5 to +1.5 DR – -0.27 to +0.27	Numeric	0	Will set the start level of the pulse waveform in units of volts.

Response

The SE5082 will return the present low-level value. The returned value will be in standard scientific format (for example: 100 mV would be returned as 100e-3 – positive numbers are unsigned).

Example

Command `:PATT:COMP:LIN:STAR 0.5`

Query `:PATT:COMP:LIN:STAR?`

:PATTern:COMPosed:LINear#<header><data_array>

Description

This command will download an array of data for the built-in pulse composer. The composed data is made of 12 bytes. Figure 4-21 shows a pulse data parameter sequence that is acceptable for the SE5082. Figure 4-22 demonstrates how the pulse composer interprets the level and duration to generate linear transitions.

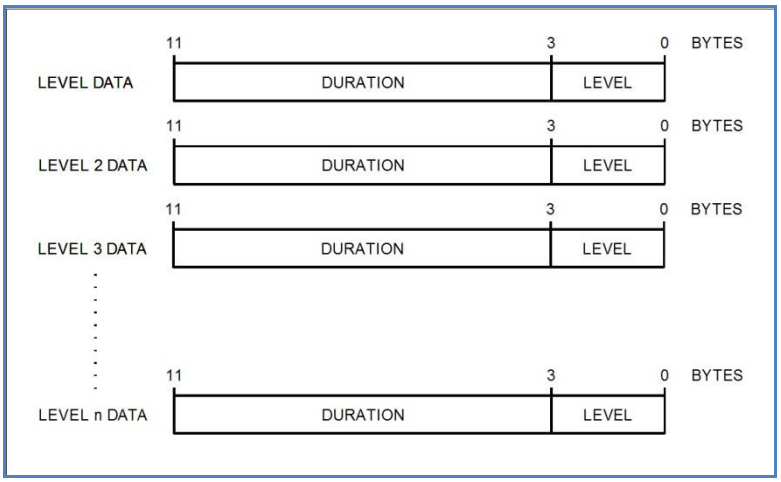

Figure 4-18, Composed Linear Pulse - Data Representation

Note in the above:

Pulse level data is provided in terms of level and duration only. The pulse will be built by defining the levels and the duration that the level transitions from last to next.

Note the fastest transition that the output can generate is around 400 ps

Note in Figure 4-22:

Initial level defines from what amplitude level the first transition will occur.

Final level 1 defines the slope of which the pulse will transition linearly and Duration 1 defines how long it will take for the amplitude to reach the end level.

The end level is automatically selected for the start level of the next transition.

Duration set to 0 generates fast transition. The fastest transition that the output can generate is around 400 ps

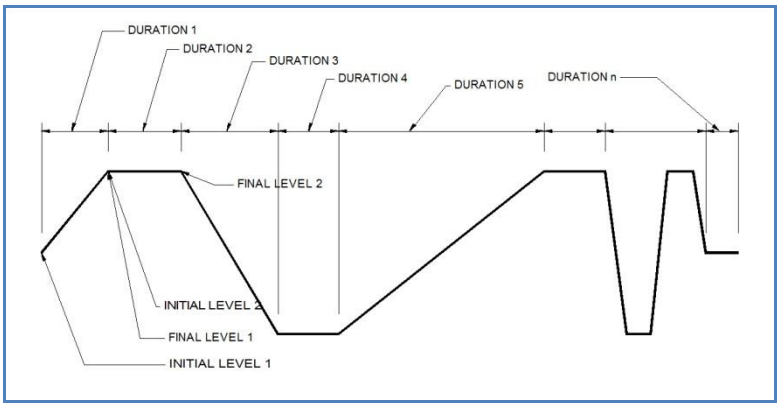

Figure 4-19, Composed Linear Pulse - Level Data Representation

Parameters

Name	Type	Description
<header>	Discrete	Provides information on the size of the binary block that contains pattern parameters data.
<data_array>	Binary	Block of binary data that contains level and duration data, as explained and shown in Figure 4-21 above. The binary-data is array of N*12 bytes holding the definitions of the N linear-intervals of the Piecewise-Linear-Pattern, where each 12-bytes sub-block consists: - Ending-Voltage-Level as float value (4-bytes). Difference between the highest and lowest levels should not exceed maxVpp. The start voltage level of each interval is ending voltage level of the previous interval (the start level of the first interval is set separately). - Dwell-Time as double-float (8-bytes). The G.C.D of all non-zero dwell-times should be at least 0.2e-9 (seconds). Note that zero dwell-time is allowed and can be used for level jumps)

Example

Command `:PATT:COMP:LIN#<header><data_array>`

:PATTern:COMPosed:RESolution<resolution>(?)

Description

If patt:comp:res:type is set to user, you use this command to set or query the resolution that you prefer to build your linear waveform. Notice however, that fine resolution dictates many waveform data points and therefore, if you have slow transitions, make sure that your resolution allows for sufficient number of data points to build the waveform. When patt:comp:res:type is set to auto, the resolution will be set automatically by the machine and this command will not affect the output.

Parameters

Name	Range	Type	Default	Description
<resolution>	200e-12 to 100e-9	Numeric	0.2e-9	Will set the number of memory locations that will be allocated for the PRBS data.

Response

The SE5082 will return the present resolution value in units of seconds.

Example

Command `:PATT:COMP:RES 10e-9`

Query `:PATT:COMP:RES?`

:PATTern:COMPosed:RESolution:TYPE{AUTO|USER}(?)

Description

Use this command to set or query the resolution type. If you do not want to bother with setting resolutions, select the auto option. Use the user option when you fully understand the implications of available waveform points and the amount of points that you require to build your linear transitions.

Parameters

Name	Type	Default	Description
AUTO	Discrete	AUTO	Using this option, the pulse composer will automatically select the best resolution to build the linear waveform.
USER	Discrete		Selects this option if you want to select the resolution that the linear transitions will increment when you build your required pulse.

Response

The SE5082 will return AUTO or USER depending on the present linear resolution type setting.

Example

Command `:PATT:COMP:RES:TYPE USER`

Query `:PATT:COMP:RES:TYPE?`

LAN System Configuration Commands

Use these commands to configure IP addresses and other LAN parameters. It is strongly recommended that this be done with a computer specialist, because incorrect programming may place the instrument in an unknown configuration which will not communicate as expected with the network. The LAN commands are summarized in Table 4-14.

Note

Last LAN configuration settings will remain as long as the instrument is turned on. New LAN configuration settings will take effect only after the instrument has been powered off and on.

Table 4-14, LAN Commands Summary

Keyword	Parameter Form	Default	Notes
:SYSTem			
:IP			
[:ADDRess]	<IP_address>		
:MASK	<mask>		
:GATeway	<gate_way>		
:BOOTp	OFF | ON | 0 | 1	0	
:HOSTname	<host_name>		
:MAC?			
:KEEPalive			
:STATe	OFF | ON | 0 | 1	1	
:TIMEout	2 to 300	45	
:PROBes	2 to 10	2	
:FIRMware			
:PORT?			
:TELNet			
:PORT?			

SYSTem:IP<ip_adrs>(?)

Description

This command programs the IP address for LAN operation. The same command is performed from the front panel Utility -> Remote Interface -> LAN menu. Note that after you use this command the power to the SE5082 must be recycled before the new address takes precedence.

Parameters

Name	Range	Type	Description
<ip_adrs>	0 to 255	String	Programs the IP address for LAN operation. The current IP address can be observed on the TCP/IP Network Properties display.

Response

The SE5082 will return the present IP address value similar to the following: 192.168.0.245

Example

Command `:SYST:IP<192.168.0.200>`

Query `:SYST:IP?`

SYSTem:IP:MASK<mask_adrs>(?)

Description

This command programs the subnet mask address for LAN operation. The programming is performed from the front panel Utility -> Remote Interface -> LAN menu.

Parameters

Name	Range	Type	Description
<mask_adrs>	0 to 255	String	Programs the subnet mask address for LAN operation. Programming must be performed from the front panel. The current IP address can be observed on the TCP/IP Network Properties display.

Response

The SE5082 will return the present IP address value similar to the following: 255.255.255.0

Example

Command `:SYST:IP:MASK<>`

Query `:SYST:IP:MASK?`

SYSTem:IP:BOOTp{OFF|ON|0|1}(?)

Description

Use this command to toggle BOOTP mode on and off.

Parameters

Range	Type	Default	Description
0-1	Discrete	0	Toggles BOOTP mode on and off. When on, the IP address is administrated automatically by the system.

Response

The SE5082 will return 0 or 1, depending on the present BOOTP setting.

Example

Command `:SYST:IP:BOOT ON`

Query `:SYST:IP:BOOT?`

SYSTem:IP:GATeway<gate_adrs>(?)

Description

This command programs the gateway address for LAN operation. The programming is performed from the front panel Utility -> Remote Interface -> LAN menu.

Parameters

Name	Range	Type	Description
<gate_adrs>	0 to 255	String	Programs the gateway address for LAN operation. Programming must be performed from the front panel. The current IP address can be observed on the TCP/IP Network Properties display.

Response

The SE5082 will return the present gateway address value similar to the following: 0.0.0.0

Example

Command `:SYST:IP:GAT<255.255.255.0>`

Query `:SYST:IP:GAT?`

SYSTem:IP:HOSTname<name>(?)

Description

This command programs the host name address for LAN operation. The programming is performed in the factory and it is highly suggested that users do not change the host name without first consulting an Tabor customer service person.

Parameters

Name	Type	Description
<name>	String	Programs the host name for LAN operation.

Response

The SE5082 will return a string containing the host name. String length is 16 characters.

Example

Command `:SYST:IP:HOST<PC1>`

Query `:SYST:IP:HOST?`

SYSTem:IP:MAC(?)

Description

Use this command to query the MAC address of the instrument

Response

The SE5082 will return the MAC address of the instrument.

Example

Query `:SYST:IP:MAC?`

SYSTem:KEEPalive:STATe{OFF|ON|0|1}(?)

Description

Use this command to toggle the keep-alive mode on and off. The keep-alive mode assures that LAN connection remains uninterrupted throughout the duration of the LAN interfacing.

Parameters

Range	Type	Default	Description
0-1	Discrete	1	Toggles the keep-alive mode on and off. When on, the SE5082 constantly checks for smooth LAN connection at intervals programmed by the syst:keep:time command. The LAN will be probed as many times as programmed by syst:keep:prob parameter to check if there is an interruption in the LAN communication. When communication fails, the SE5082 reverts automatically to local (front panel) operation.

Response

The SE5082 will return 0 or 1, depending on the present keep-alive setting.

Example

Command `:SYST:KEEP:STAT ON`

Query `:SYST:KEEP:STAT?`

SYSTem:KEEPalive:TIMeout<time_out>(?)

Description

This command programs the keep-alive time-out. The keep-alive mode assures that LAN connection remains uninterrupted throughout the duration of the LAN interfacing.

Parameters

Name	Range	Type	Default	Description
<time_out>	2 to 300	Numeric	45	Programs the keep-alive time out in units of seconds. The time-out period is initiated when the LAN is idle for more than the time-out period. The LAN will be probed as many times as programmed by syst:keep:prob parameter to check if there is an interruption in the LAN communication. When communication fails, the SE5082 reverts automatically to local (front panel) operation.

Response

The SE5082 will return the present keep-alive time-out value.

Example

Command `:SYST:KEEP:TIM 100`

Query `:SYST:KEEP:TIM?`

SYSTem:KEEPalive:PROBes<probes>(?)

Description

This command programs the number of probes that are used by the keep-alive sequence. The keep-alive mode assures that LAN connection remains uninterrupted throughout the duration of the LAN interfacing.

Parameters

Name	Range	Type	Default	Description
<probes>	2 to 10	Numeric	2	Programs the number of probes that are used by the keep-alive sequence. The time-out period is initiated when the LAN is idle for more than the time-out period and the LAN will be probed as many times as programmed by this parameter to check if there is an interruption in the LAN communication. When communication fails, the SE5082 reverts automatically to local (front panel) operation.

Response

The SE5082 will return the present keep-alive number of probes.

Example

Command `:SYST:KEEP:PROB 3`

Query `:SYST:KEEP:PROB?`

SYSTem:FIRMware:PORT?

Description

This command queries the LAN port used to send commands to the instruments.

Response

The SE5082 will return the LAN port number, 5025.

Example

Query `:SYST:FIRM:PORT?`

SYSTem:TELNet:PORT?

Description

This command queries the telnet port used to send commands to the instruments.

Response

The SE5082 will return the telnet port number, 5024.

Example

Query `:SYST:TELN:PORT?`

Store/Recall Commands

Use these commands to store instrument settings. The store command collects the current front panel setting and parameters and stores them in a dedicated memory cell. The store operation can include front panel settings and current waveforms but, due to memory limitations, you may select to keep settings only without waveforms or waveforms only without settings. The store/recall commands are summarized in Table 4-16.

Table 4-16, Store/Recall Commands Summary

Keyword	Parameter Form	Default	Notes
:SYSTem			
:STORe			
:CELL	1 to 9	1	
:CLEar			Clears memory cell
:CONFig	SETup | WAVE | ALL	ALL	
:TARGet	INTernal | USB	INT	
:UPDate			
:RECall			
:CELL	1 to 9	1	
:TARGet	INTernal | USB	INT	
:UPDate			

SYSTem:STORe:CELL<cell_number>(?)

Description

This command selects a memory cell. The selected memory cell will become the target for the store operation. You may select to store front panel settings, waveforms or both.

Parameters

Name	Range	Type	Description
<cell_number>	1 to 9	Numeric (integer only)	Selects an active memory cell number. Consequent store commands will affect this cell only.

Response

The SE5082 will return the active memory cell value.

Example

Command `:SYST:STOR:CELL 1`

Query `:SYST:STOR:CELL?`

SYSTem:STORe:CLEar

Description

Use this command to clear the content of a specific memory cell. This will prepare the memory cell to accept new setup and waveform data.

Example

Command `:SYST:STOR:CLE`

SYSTem:STORe:CONFig{SETup|WAVE|ALL}(?)

Description

Use this command to select what you intend to store in the active memory cell. The default setting is ALL. If you plan to save anything, make sure that the default setting is changed before you do a cell memory update.

Parameters

Name	Type	Default	Description
SETup	Discrete		Selects front-panel configuration to be stored in the active memory cell.
WAVE	Discrete		Selects the current arbitrary waveform to be stored in the active memory cell.
ALL	Discrete	ALL	Will store both front-panel configuration and waveform in the active memory cell.

Response

The SE5082 will return SET, WAVE, or ALL depending on the present configuration setting.

Example

Command `:SYST:STOR:CONF WAVE`

Query `:SYST:STOR:CONF?`

SYSTem:STORe:TARGet{INTernal|USB}(?)

Description

Use this command to select the target of your store operation. You may select between an internal flash memory that is limited in size and disk-on-key flash that you can attach to the front panel input. Selecting the internal option, you may store waveforms up to 100k long but external memory has no limitations except the physical size of the flash on the disk-on-key.

Parameters

Name	Type	Default	Description
INTernal	Discrete	INT	Selects the internal flash memory as the target for the store operation. Waveform size for this option is limited to 100k points.
USB	Discrete		Selects the front panel USB input as the target for the store operation. Waveform size for this option is limited only by the size of the disk-on-key flash.

Response

The SE5082 will return INT or USB, depending on the present store target setting.

Example

Command `:SYST:STOR:TARG INT`

Query `:SYST:STOR:TARG?`

SYSTem:STORe:UPDate

Description

Use this command to update the active memory cell with the front panel settings, waveforms, or both.

Example

Command `:SYST:STOR:UPD`

SYSTem:RECall:CELL<cell_number>(?)

Description

This command selects a memory cell. The selected memory cell will become the source for the recall operation. You may select to recall front panel settings, waveforms, or both.

Parameters

Name	Range	Type	Description
<cell_number>	1 to 9	Numeric (integer only)	Selects an active memory cell number. Consequent recall commands will affect this cell only.

Response

The SE5082 will return the active memory cell value.

Example

Command `:SYST:REC:CELL 1`

Query `:SYST:REC:CELL?`

SYSTem:RECall:TARGet{INTernal|USB}(?)

Description

Use this command to select the source of your recall operation. You may select between an internal flash memory that is limited in size and disk-on-key flash that you can attach to the front panel input. Selecting the internal option, you may recall waveforms up to 100k long, but external memory has no limitations except the physical size of the flash on the disk-on-key.

Parameters

Name	Type	Default	Description
INTernal	Discrete	INT	Selects the internal flash memory as the source for the recall operation. Waveform size for this option is limited to 100 k points.
USB	Discrete		Selects the front panel USB input as the source for the recall operation. Waveform size for this option is limited only by the size of the disk-on-key flash.

Response

The SE5082 will return INT or USB, depending on the present recall source setting.

Example

Command `:SYST:REC:TARG INT`

Query `:SYST:REC:TARG?`

SYSTem:RECall:UPDate

Description

Use this command to update the front panel and arbitrary memory with the information stored in the active memory cell.

Example

Command `:SYST:REC:UPD`

The Store/Recall Folder Structure

Bear in mind that the flexibility of the embedded operating system within the SE5082 is not the same as the Windows OS and therefore, strict adherence to the folder and file structure are required for the store and recall operation. The following paragraphs explain how to create and maintain the correct folder and file images; this is done automatically when a USB stick is used and the store operation is done from the front panel.

The simplest option is of course to start with an empty stick. The folders must be placed in the root directory. Each setup is stored in a separate folder and each setup folder must have a unique name that associates itself with the cell number. The folder names are of the structure Setup'X' where X can be a number from 1 to 9. For example, folder name Setup1 is legal and corresponds to cell 1 but folder name Setup5_created_May_2011 is illegal. Figure 4-23 shows legal folder names for using the stored data on a USB stick.

In Figure 4-23, note that each setup is stored in a separate folder. Folders are named Setup1 through Setup2. Setup1 contains several files, some examples are Setup.txt, which includes instrument settings, w00001c1.wav through w00003c1 which contain waveforms coordinated in binary format for generating arbitrary waveforms, ahp_fix2, which contains the fixed amplitude hopping values list of channel 2. Instructions how to use setup file names is given in the following paragraphs.

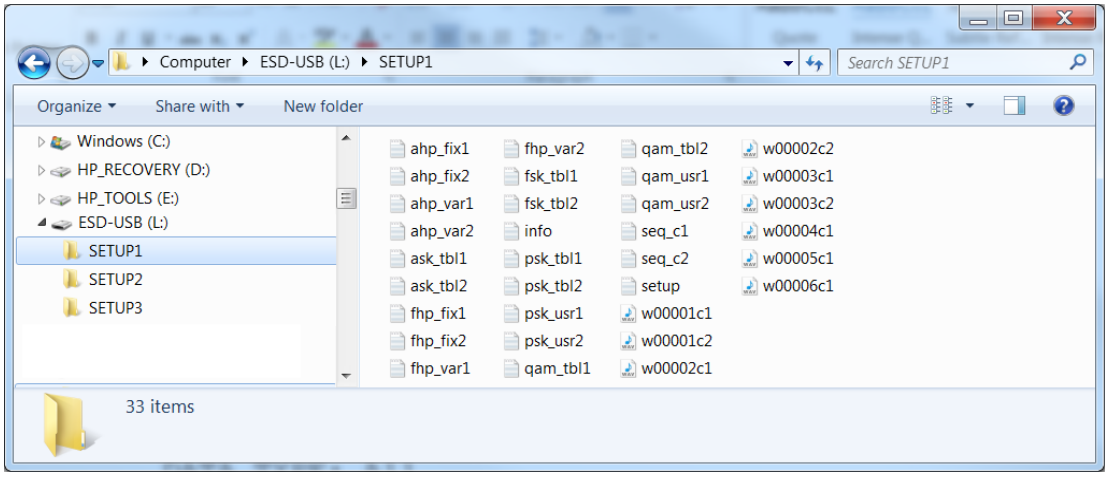

Figure 4-20, Store/Recall Folders Structure Example

The Store/Recall File Names

The setup folder contains files that are translated to instrument settings when downloaded to the SE5082. These files have specific and unique file names for the OS to be able to read them properly. Failure to name the files as requested and understood by the SE5082 will result in false expectations that the settings were loaded to the instrument. Therefore, use the names and extensions exactly as given below. Note that files containing parameter values contain the name of the parameter and the corresponding channel.

setup.txt - is the main image file name (case sensitive!). It contains the entire settings of the generator written as SCPI commands. The file can be written and edited using MS word or any other word processor but it must be saved as a text file without attributes and characters that are not recognized by the SE5082 OS. An example of this file can be seen in the following pages.

w0000Xc1.wav – is a binary file that contains arbitrary waveform coordinates in binary format. The “X” represents the segment number where this file is stored and “c1” corresponds to the instrument channel 1. Information how to create this file is given in this chapter. Another option to create this file easily is using the waveform composer in ArbConnection. Information how to use ArbConnection to generate an arbitrary waveform is given in a separate ArbConnection manual. Note in Figure 4-23 that six arbitrary waveform files are shown; each is downloaded to a different segment in channel 1.

info.txt - is a meta information text file. It contains information such as creation date, device model and device model id. Furthermore the file consists of the type of data to be restored, the ‘Endianness’

and 'Wave Data Format'. An example of this file can be seen below.

ahp_fix1.txt – is a text file that contains a list of amplitude hopping values for channel 1. Only one list is allowed per setup.

ahp_var1.txt – is a text file that contains a list of amplitude hopping values and their corresponding dwell time for channel 1. Only one list is allowed per setup.

ask_tbl1.txt – is a text file that contains a list of ASK values for channel 1. Only one list is allowed per setup.

fhp_fix1.txt – is a text file that contains a list of frequency hopping values for channel 1. Only one list is allowed per setup.

fhp_var1.txt – is a text file that contains a list of frequency hopping values and their corresponding dwell time for channel 1. Only one list is allowed per setup.

fsk_tbl1.txt – is a text file that contains a list of FSK values for channel 1. Only one list is allowed per setup.

seq_c1 – is a text file that contains the sequencer data and SCPI commands for channel 1. Only one list is allowed per setup.

Note that for channel 2 all parameter list files and waveform files names will be identical but will end with the channel number. For example seq_c2.

The Store/Recall File Structure

If you perform a store operation, the file shown below shows the entire configuration of the SE5082. Of course, the values may look differently but the contents of the file will be similar. You may use the file examples below to start exerting and compiling your external file. There are some guidelines that you can use, which will help you minimize the amount of work that you have to do; note the following:

1. The Setup file contains all of the commands that control the generator. The file below shows the parameters and configuration of the example setup. If you do not intend to change parameter values, reset the generator before the recall operation and then recall the stored cell. The instrument will change those parameters that are listed in the file and will not modify default values that need not be touched.
2. Colon (:) designates a command must follow. The pound signs designate remarks and are being ignored.
3. If the commands look familiar, this is because these are exactly

the SCPI commands that are being used for remote programming. If you are familiar with the SCPI concept then it should be very easy for you to comprehend the usage of these commands.

4. Channel dependent commands are separated from commands that are common for both channels. It is recommended that the order of programming remain as in the Setup file but it is up to the user's discretion to rearrange the commands in an order more familiar or friendlier for his application.
5. The Setup file does not include the arbitrary and sequence configuration, this is specified in the seq_c1 or seq_c2 files. Each arbitrary waveform is specified in the setup directory, bearing exactly the name format as specified above. The names of the file are defined so as to include the segment where they are to be located and the channel.
6. Following SCPI rules, commands can be used in short or long format.

Recall Setup1 Example

The Removable Disk folder structure is shown in Figure 4-24 and the setup, info, seq_c1 files are given below. Note that there are three arbitrary waveform files for channel 1, w00001c1, w00002c1 and w00003c1.

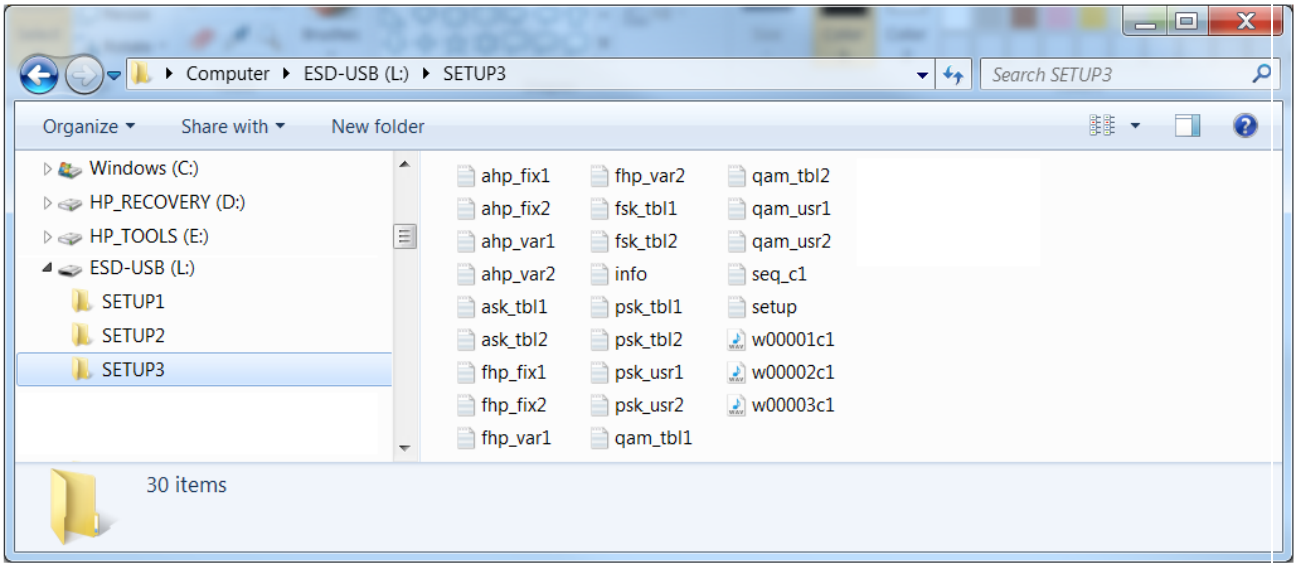

Figure 4-21, Setup3 Example Folder

Setup.txt

```
######################################################################
# Setup Config Script
# (each line is either comment line or SCPI command).
#
# Config parameters read from: TABOR SE5082 ver: 0.04
# Storing date: 2017/02/21 14:17:16
######################################################################


# Common Parameters:
:SYST:POW DEF
:INST:COUP:OFFS 0
:INST:COUP:SKEW 0.000000e+00
:INST:COUP:STAT OFF
:OUTP:SYNC:SOUR CH1
:XINS:MODE MAST
:XINS:OFFS 0
:XINS:STAT OFF
:XINS:SKEW -0.000000000e+00


########### Configure Channel 2 ###########

:INST:SEL 2

# ARM Config:
:ARM:SEQ:LEV 1.600000e+00
:ARM:SEQ:SLOP POS
:INIT:CONT:ENAB SELF
:INIT:CONT:ENAB:SOUR BUS

# TRIG Config:
:INIT:CONT:STAT ON
:INIT:GATE:STAT OFF
:TRIG:SEQ:COUN 1
:TRIG:SEQ:DEL 0
```

```
:TRIG:SEQ:FILT:HPAS:WIDT 1.000000e-01
:TRIG:SEQ:FILT:HPAS:STAT OFF
:TRIG:SEQ:FILT:LPAS:WIDT 1.000000e-03
:TRIG:SEQ:FILT:LPAS:STAT OFF
:TRIG:SEQ:HOLD 1.000000e-07
:TRIG:SEQ:HOLD:STAT OFF
:TRIG:SEQ:LEV 1.600e+00
:TRIG:SEQ:MODE NORM
:TRIG:SEQ:SLOP POS
:TRIG:SEQ:SOUR:ADV EXT
:TRIG:SEQ:TIM:MODE TIME
:TRIG:SEQ:TIM:DEL 152
:TRIG:SEQ:TIM:TIME 1.500000e-05

# Freq-Raster Config:
:SOUR:FREQ:RAST 1.000000000e+09
:SOUR:FREQ:RAST:SOUR INT
:SOUR:FREQ:RAST:DIV 1
:SOUR:ROSC:SOUR INT
:SOUR:ROSC:EXT:FREQ 1.000000000e+08

# Common Train Config:
:SOUR:VOLT:LEV:AMPL 5.000e-01
:SOUR:VOLT:LEV:OFFS:COMM 0.000e+00

# Marker Config:
:SOUR:MARK1:DEL 0.000000e+00
:SOUR:MARK1:POS 0
:SOUR:MARK1:WIDT 64
:SOUR:MARK1:VOLT:LEV:HIGH 1.000e+00
:SOUR:MARK1:VOLT:LEV:LOW 0.000e+00
:SOUR:MARK2:DEL 0.000000e+00
:SOUR:MARK2:POS 0
:SOUR:MARK2:WIDT 64
:SOUR:MARK2:VOLT:LEV:HIGH 1.000e+00
:SOUR:MARK2:VOLT:LEV:LOW 0.000e+00
```

```
# Standard-Wave Config:
:SOUR:FREQ:CW 1.000000000e+07
:SOUR:FUNC:SHAP SIN
:SOUR:SIN:PHAS 0.000
:SOUR:TRI:PHAS 0.000
:SOUR:SQU:DCYC 50.000
:SOUR:RAMP:DEL 10.000
:SOUR:RAMP:TRAN:LEAD 60.000
:SOUR:RAMP:TRAN:TRA 30.000
:SOUR:SINC:NCYC 10
:SOUR:GAUS:EXP 10
:SOUR:EXP:EXP -10
:SOUR:DC:OFFS 0.000

# Digital-Pulse Config:
:PULS:CONF TIME
:PULS:DEL 1.000000000e-06
:PULS:DEL:PERC 1.00e+01
:PULS:DOUB:DEL 1.000000000e-06
:PULS:DOUB:DEL:PERC 1.00e+01
:PULS:LEV:CONT HLOW
:PULS:LEV:HIGH 0.500
:PULS:LEV:LOW 0.000
:PULS:LEV:AMPL 0.500
:PULS:LEV:OFFS 0.000
:PULS:MOD SING
:PULS:POL NORM
:PULS:PER 1.000000000e-05
:PULS:TRAN:LEAD 1.000000000e-06
:PULS:TRAN:LEAD:PERC 1.00e+01
:PULS:TRAN:TRA 1.000000000e-06
:PULS:TRAN:TRA:PERC 1.00e+01
:PULS:TRAN:STAT FAST
:PULS:WIDT 2.000000000e-06
:PULS:WIDT:PERC 2.00e+01

# Pattern Generator Config:
```

```
:PATT:MODE COMP
:PATT:PRBS:TYPE USER
:PATT:PRBS:BAUD 1.00000000e+07
:PATT:PRBS:LEV 2
:PATT:PRBS:LEV:HIGH 0.000e+00
:PATT:PRBS:LEV:LOW 0.000e+00
:PATT:PRBS:LOOP 1
:PATT:PRBS:PRE 0
:PATT:PRBS:LENG 8
:PATT:COMP:TRAN:TYPE LIN
:PATT:COMP:LIN:STAR 5.000e-01
:PATT:COMP:RES 5.000000000e+09
:PATT:COMP:RES:TYP AUTO

# Arbitrary Modulations Config:
:SOUR:MOD:TYPE OFF
:SOUR:MOD:CARR:FREQ 1.000000000e+06
:SOUR:MOD:CARR:FUNC SIN
:SOUR:AM:FUNC:SHAP SIN
:SOUR:AM:INT:FREQ 1.000000000e+03
:SOUR:AM:DEPT 50.000
:SOUR:FM:DEV 5.000000000e+05
:SOUR:FM:FUNC:SHAP SIN
:SOUR:FM:FREQ 1.000000000e+04
:SOUR:FM:MARK:FREQ 5.050000000e+05
:SOUR:SWE:FREQ:STAR 4.000000000e+07
:SOUR:SWE:FREQ:STOP 8.000000000e+07
:SOUR:SWE:TIME 1.000000000e-05
:SOUR:SWE:DIR UP
:SOUR:SWE:SPAC LIN
:SOUR:SWE:MARK:FREQ 6.000000000e+07
:SOUR:CHIR:WIDT 1.000000000e-05
:SOUR:CHIR:REP 2.500000000e-05
:SOUR:CHIR:FREQ:STAR 4.000000000e+07
:SOUR:CHIR:FREQ:STOP 8.000000000e+07
:SOUR:CHIR:FREQ:DIR UP
:SOUR:CHIR:FREQ:SPAC LIN
```

```
:SOUR:CHIR:AMPL:DIR UP
:SOUR:CHIR:AMPL:SPAC LIN
:SOUR:CHIR:AMPL:DEPT 5.000e+01
:SOUR:CHIR:MARK:FREQ 6.000000000e+07
:SOUR:FSK:FREQ:SHIF 2.000000000e+06
:SOUR:FSK:BAUD 1.000000000e+04
:SOUR:FSK:MARK 1
:SOUR:ASK:AMPL:STAR 0.500000
:SOUR:ASK:AMPL:SHIF 1.000000
:SOUR:ASK:BAUD 1.000000000e+04
:SOUR:ASK:MARK 1
:SOUR:FHOP:DWEL:MODE VAR
:SOUR:FHOP:DWEL:TIME 5.000000000e-06
:SOUR:FHOP:MARK 1
:SOUR:AHOP:DWEL:MODE VAR
:SOUR:AHOP:DWEL:TIME 5.000000000e-06
:SOUR:AHOP:MARK 1

# Active Segment:
:TRAC:SEL 1

# HOP Config:
:TRAC:SEL:SOUR BUS
:TRAC:SEL:TIM IMM

# Sync Config:
:OUTP:SYNC:FUNC PULS
:OUTP:SYNC:POS:POIN 0
:OUTP:SYNC:WIDT 32

# Function Mode:
:SOUR:FUNC:MODE FIX


########### Configure Channel 1 ###########

:INST:SEL 1
```

```
# ARM Config:
:ARM:SEQ:LEV 1.600000e+00
:ARM:SEQ:SLOP POS
:INIT:CONT:ENAB SELF
:INIT:CONT:ENAB:SOUR BUS

# TRIG Config:
:INIT:CONT:STAT OFF
:INIT:GATE:STAT OFF
:TRIG:SEQ:COUN 1
:TRIG:SEQ:DEL 0
:TRIG:SEQ:FILT:HPAS:WIDT 1.000000e-01
:TRIG:SEQ:FILT:HPAS:STAT OFF
:TRIG:SEQ:FILT:LPAS:WIDT 1.000000e-03
:TRIG:SEQ:FILT:LPAS:STAT OFF
:TRIG:SEQ:HOLD 1.000000e-07
:TRIG:SEQ:HOLD:STAT OFF
:TRIG:SEQ:LEV 1.600e+00
:TRIG:SEQ:MODE NORM
:TRIG:SEQ:SLOP POS
:TRIG:SEQ:SOUR:ADV EXT
:TRIG:SEQ:TIM:MODE TIME
:TRIG:SEQ:TIM:DEL 152
:TRIG:SEQ:TIM:TIME 1.500000e-05

# Freq-Raster Config:
:SOUR:FREQ:RAST 5.000000000e+09
:SOUR:FREQ:RAST:SOUR INT
:SOUR:FREQ:RAST:DIV 1
:SOUR:ROSC:SOUR INT
:SOUR:ROSC:EXT:FREQ 1.000000000e+08

# Common Train Config:
:SOUR:VOLT:LEV:AMPL 5.000e-01
:SOUR:VOLT:LEV:OFFS:COMM 0.000e+00
```

```
# Marker Config:
:SOUR:MARK1:DEL 0.000000e+00
:SOUR:MARK1:POS 0
:SOUR:MARK1:WIDT 64
:SOUR:MARK1:VOLT:LEV:HIGH 1.000e+00
:SOUR:MARK1:VOLT:LEV:LOW 0.000e+00
:SOUR:MARK2:DEL 0.000000e+00
:SOUR:MARK2:POS 0
:SOUR:MARK2:WIDT 64
:SOUR:MARK2:VOLT:LEV:HIGH 1.000e+00
:SOUR:MARK2:VOLT:LEV:LOW 0.000e+00

# Standard-Wave Config:
:SOUR:FREQ:CW 5.000000000e+07
:SOUR:FUNC:SHAP SIN
:SOUR:SIN:PHAS 0.000
:SOUR:TRI:PHAS 0.000
:SOUR:SQU:DCYC 50.000
:SOUR:RAMP:DEL 10.000
:SOUR:RAMP:TRAN:LEAD 60.000
:SOUR:RAMP:TRAN:TRA 30.000
:SOUR:SINC:NCYC 10
:SOUR:GAUS:EXP 10
:SOUR:EXP:EXP -10
:SOUR:DC:OFFS 0.000

# Digital-Pulse Config:
:PULS:CONF TIME
:PULS:DEL 1.000000000e-06
:PULS:DEL:PERC 1.00e+02
:PULS:DOUB:DEL 1.000000000e-06
:PULS:DOUB:DEL:PERC 1.00e+02
:PULS:LEV:CONT HLOW
:PULS:LEV:HIGH 0.500
:PULS:LEV:LOW 0.000
:PULS:LEV:AMPL 0.500
:PULS:LEV:OFFS 0.250
```

```
:PULS:MOD SING
:PULS:POL NORM
:PULS:PER 1.000000100e-06
:PULS:TRAN:LEAD 1.000000000e-06
:PULS:TRAN:LEAD:PERC 1.00e+02
:PULS:TRAN:TRA 1.000000000e-06
:PULS:TRAN:TRA:PERC 1.00e+02
:PULS:TRAN:STAT FAST
:PULS:WIDT 5.000000000e-07
:PULS:WIDT:PERC 5.00e+01

# Pattern Generator Config:
:PATT:MODE COMP
:PATT:PRBS:TYPE USER
:PATT:PRBS:BAUD 1.00000000e+07
:PATT:PRBS:LEV 2
:PATT:PRBS:LEV:HIGH 0.000e+00
:PATT:PRBS:LEV:LOW 0.000e+00
:PATT:PRBS:LOOP 1
:PATT:PRBS:PRE 0
:PATT:PRBS:LENG 8
:PATT:COMP:TRAN:TYPE LIN
:PATT:COMP:LIN:STAR 5.000e-01
:PATT:COMP:RES 5.000000000e+09
:PATT:COMP:RES:TYP AUTO

# Arbitrary Modulations Config:
:SOUR:MOD:TYPE OFF
:SOUR:MOD:CARR:FREQ 1.000000000e+06
:SOUR:MOD:CARR:FUNC SIN
:SOUR:AM:FUNC:SHAP SIN
:SOUR:AM:INT:FREQ 1.000000000e+03
:SOUR:AM:DEPT 50.000
:SOUR:FM:DEV 5.000000000e+05
:SOUR:FM:FUNC:SHAP SIN
:SOUR:FM:FREQ 1.000000000e+04
:SOUR:FM:MARK:FREQ 5.050000000e+05
```

```
:SOUR:SWE:FREQ:STAR 4.000000000e+07
:SOUR:SWE:FREQ:STOP 8.000000000e+07
:SOUR:SWE:TIME 1.000000000e-05
:SOUR:SWE:DIR UP
:SOUR:SWE:SPAC LIN
:SOUR:SWE:MARK:FREQ 6.000000000e+07
:SOUR:CHIR:WIDT 1.000000000e-05
:SOUR:CHIR:REP 2.500000000e-05
:SOUR:CHIR:FREQ:STAR 4.000000000e+07
:SOUR:CHIR:FREQ:STOP 8.000000000e+07
:SOUR:CHIR:FREQ:DIR UP
:SOUR:CHIR:FREQ:SPAC LIN
:SOUR:CHIR:AMPL:DIR UP
:SOUR:CHIR:AMPL:SPAC LIN
:SOUR:CHIR:AMPL:DEPT 5.000e+01
:SOUR:CHIR:MARK:FREQ 6.000000000e+07
:SOUR:FSK:FREQ:SHIF 2.000000000e+06
:SOUR:FSK:BAUD 1.000000000e+04
:SOUR:FSK:MARK 1
:SOUR:ASK:AMPL:STAR 0.500000
:SOUR:ASK:AMPL:SHIF 1.000000
:SOUR:ASK:BAUD 1.000000000e+04
:SOUR:ASK:MARK 1
:SOUR:FHOP:DWEL:MODE VAR
:SOUR:FHOP:DWEL:TIME 5.000000000e-06
:SOUR:FHOP:MARK 1
:SOUR:AHOP:DWEL:MODE VAR
:SOUR:AHOP:DWEL:TIME 5.000000000e-06
:SOUR:AHOP:MARK 1

# Active Segment:
:TRAC:SEL 2

# HOP Config:
:TRAC:SEL:SOUR BUS
:TRAC:SEL:TIM IMM
```

```
# Sync Config:
:OUTP:SYNC:FUNC PULS
:OUTP:SYNC:POS:POIN 0
:OUTP:SYNC:WIDT 32

# Function Mode:
:SOUR:FUNC:MODE PULS


########### Select Channel 1 ###########

:INST:SEL 1
```

Seq_c1.txt

```
#####################################################
# Sequencer Data of Channel 1
# Storing date: 2017/02/21 14:17:16.
# Each line consists of SCPI command.
#####################################################

# Select channel 1
:INST:SEL 1

# Delete the sequencer table:
:SOUR:SEQ:DEL:ALL


####### Define Sequence no. 1 #######

# Select the sequence:
:SOUR:SEQ:SEL 1

# Define the sequencer steps:
:SOUR:SEQ:DEF 1, 1, 1, 0
:SOUR:SEQ:DEF 2, 2, 2, 0
:SOUR:SEQ:DEF 3, 3, 3, 0
```

```
####### Define Sequence no. 2 #######

# Select the sequence:
:SOUR:SEQ:SEL 2

# Define the sequencer steps:
:SOUR:SEQ:DEF 1, 3, 1, 0
:SOUR:SEQ:DEF 2, 1, 3, 0
:SOUR:SEQ:DEF 3, 2, 1, 0

# Sequence Step Mode:
:SOUR:SEQ:PRES WAVE

# Active Sequence:
:SOUR:SEQ:SEL 2

##########################################
# Advanced Sequence Info:
##########################################
# Delete the advanced-sequence table:
:SOUR:ASEQ:DEL


# Advanced sequence meta-info:
:SOUR:SEQ:ADV AUTO
:SOUR:ASEQ:ADV AUTO
:SOUR:ASEQ:ONC:COUN 1
:SOUR:ASEQ:SYNC:LOCK 1
```

Info.txt

```
# SETUP2 meta information
# Creation date: 2017/02/21 14:17:16

DEVICE_VENDER: Tabor Electronics
DEVICE_MODEL: SE5082
DEVICE_MODEL_ID: 0x168E5082
DEVICE_SW_VER: 0.04
```

```
# The 'DATA_TYPE' is the type of data to restored.
# The choices are:
#  - 'Config' (configuration data).
#  - 'Wave'   (wave data).
#  - 'All'    (both config data and wave data).
#  - 'None'   (none).
# The default is: 'All'.
DATA_TYPE: All


# The 'ENDIANNESS' refers to the wave-data and digital-data.
# It can be either 'LITTLE_ENDIAN' ("Intel-format") or 'BIG_ENDIAN' ("Motorola-
format").
# The default is: 'LITTLE_ENDIAN')
ENDIANNESS: LITTLE_ENDIAN


# The 'WAVE_DATA_FORMAT' is the format of wave-samples in the (binary) '.wav'
files
# (those files consists of 16-bits-words, where each word contains single wave-
sample).
# The choices are: 'U16', 'S16', 'U14', 'S14', 'U12', 'S12'
# Where U/S means Signed/Unsigned, and 12/14/16 is the number of bits (out of
16).


# For example, 'S14' means that the 14 less-significant bits of each
# 16-bits-word represents wave-sample's value as singed integer
# (the 14th bit is the sign-bit!).
# The default value is: 'U14'.
WAVE_DATA_FORMAT: U14
```

The above Setup3 file when recalled from the front panel of the unit will do the following:

Channel1 – Will be set to Sequence mode with an SCLK of 5GS/s and a sequence table where in step 1 segment3 is repeated once, step 2 segment1 is repeated three times, and step 3 segment2 is repeated once.

Channel2 – Will be set to standard mode with a sine wave with 0

degrees starting phase and 10MHz frequency.

Please note that the setup file contains the values of all parameteres of the unit.

System Commands

The system-related commands are not related directly to waveform generation but are an important part of operating the SE5082. Use these commands to reset the instrument and query its system settings.

Table 4-16: System Commands Summary

Keyword	Parameter Form	Default
:RESet (*RST)		
:SYSTem		
:ERRor?		
:LOCal		
:VERSion?		
:INFormation		
:CALibration?		
:MODel?		
:SERial?		
:HARDware?		
:POWerup	DEFault | SETup	DEF
*RST		
*TRG		
*OPC?		
*STB?		
*IDN?		
*OPT?		

RESet, or *RST

Description

This command will reset the SE5082 to its factory defaults.

Example

Command `:*RST`

SYSTem:ERRor?

Description

Query only. This query will interrogate the SE5082 for programming errors.

Response

The SE5082 will return error code. Error messages are listed below.

Example

Query `:SYST:ERR?`

Error Messages

In general, whenever the SE5082 receives an invalid SCPI command, it automatically generates an error. Errors are stored in a special error queue and may be retrieved from this buffer one at a time. Errors are retrieved in first-in-first-out (FIFO) order. The first error returned is the first error that was stored. When you have read all errors from the queue, the generator responds with a 0,"No error" message.

If more than 30 errors have occurred, the last error stored in the queue is replaced with -350, “Queue Overflow”. No additional errors are stored until you remove errors from the queue. If no errors have occurred when you read the error queue, the generator responds with 0,"No error".

The error queue is cleared when power has been shut off or after a *CLS command has been executed. The *RST command does not clear the error queue. Use the following command to read the error queue:

SYSTem:ERRor?

Errors have the following format (the error string may contain up to 80 characters):

102,"Syntax error".

A complete listing of the errors that can be detected by the generator is given below.

100,"Command error". When the generator cannot detect more specific errors, this is the generic syntax error used.

101,"Invalid Character". A syntactic element contains a character, which is invalid for that type.

102,"Syntax error". Invalid syntax found in the command string.

103,"Invalid separator". An invalid separator was found in the command string. A comma may have been used instead of a colon or a semicolon. In some cases where the generator cannot detect a specific separator, it may return error -100 instead of this error.

104,"Data type error". The parser recognized a data element different than allowed.

108,"Parameter not allowed". More parameters were received than expected for the header.

109,"Missing parameter". Too few parameters were received for the command. One or more parameters that were required for the command were omitted.

128,"Numeric data not allowed". A legal numeric data element was received, but the instrument does not accept one in this position.

131,"Invalid suffix". A suffix was incorrectly specified for a numeric parameter. The suffix may have been misspelled.

148,"Character data not allowed". A character data element was encountered where prohibited by the instrument.

200,"Execution error". This is the generic syntax error for the instrument when it cannot detect more specific errors. Execution error as defined in IEEE-488.2 has occurred.

221,"Setting conflict". Two conflicting parameters were received which cannot be executed without generating an error.

222,”Data out of range”. Parameter data, which followed a specific header, could not be used because its value is outside the valid range defined by the generator.

224,”Illegal parameter value”. A discrete parameter was received which was not a valid choice for the command. An invalid parameter choice may have been used.

250,"Mass storage error". General error related to mass memory storage. Mass memory errors relate to the internal flash, working memory and external USB stick.

251,"Missing mass storage". This error will generate when attempting to store settings on an external USB stick but the media is not mounted.

252,"Write buffer overflow". This will occur when the writable buffer exceeds its limitation. Error relates to mass storage only.

253,"Cannot create a directory". Will be generated when the external media is incapable of creating a specific directory.

254,"Cannot delete file or directory". An attempt to delete a file or directory has failed.

255,"File extension not found". Store and recall files are expected to have certain extensions. Files that do not have the correct extension will generate this error.

256,"File name not found". An attempt to read a file name that does not exist from an external media.

257,"File type not supported". Store and recall files are expected to have certain format. Files that do not have the correct format will generate this error.

258,"Invalid specified path". An unexpected file path encountered.

300,”Device-specific-error”. This is the generic device-dependent error for the instrument when it cannot detect more specific errors. A device- specific error as defined in IEEE-488.2 has occurred.

301,”Digital Pulse-specific-error”. This is the generic device-dependent error for the instrument when it is attempting to construct a digital pulse but cannot detect more specific errors.

311,”Memory error”. Indicates that an error was detected in the instrument’s memory.

350,”Queue Overflow”. The error queue is full because more than

30 errors have occurred. No additional errors are stored until the errors from the queue are removed. The error queue is cleared when power has been shut off, or after a *CLS command has been executed.

410,"Query INTERRUPTED". A command was received which sends data to the output buffer, but the output buffer contained data from a previous command (the previous data is not overwritten). The output buffer is cleared when power is shut off or after a device clear has been executed.

SYSTem:LOCal

Description

This command will deactivate the active interface and will restore the SE5082 to local (front panel) operation.

Example

Command `:SYST:LOC`

SYSTem:VERSion?

Description

Query only. This query will interrogate the SE5082 for its current firmware version. The firmware version is automatically programmed to a secure location in the flash memory and cannot be modified by the user except when performing firmware update.

Response

The SE5082 will return the current firmware version code in a format similar to the following: 1.15

Example

Query `:SYST:VERS?`

SYSTem:INFormation:CALibration?

Description

Query only. This query will interrogate the instrument for its last calibration date.

Response

The generator will return the last calibration date in a format similar to the following: 24 Sep 2010 (10 characters maximum).

Example

Query `:SYST:INF:CAL?`

SYSTem:INFormation:MODel?

Description

Query only. This query will interrogate the instrument for its model number in a format similar to the following: SE5082-132. The model number is programmed to a secure location in the flash memory and cannot be modified by the user.

Response

The generator will return its model number SE5082-132, SE5082-164, SE5082-232, or SE5082-264.

Example

Query `:SYST:INF:MOD?`

SYSTem:INFormation:SERial?

Description

Query only. This query will interrogate the instrument for its serial number. The serial number is programmed to a secure location in the flash memory and cannot be modified by the user.

Response

The generator will return its serial number in a format similar to the following: 2xxxxx

Example

Query `:SYST:INF:SER?`

SYSTem:INFormation:HARDware?

Description

Query only. This query will interrogate the instrument for its hardware revision level. The hardware revision includes: PCB revision, FPGA revision and FPLD revision. It is programmed to a secure location in the flash memory and cannot be modified by the user.

Response

The generator will return its hardware revisions in a format similar to the following: E, D, A

Example

Query `:SYST:INF:HARD?`

SYSTem:POWerup{DEFault|SETup}(?)

Description

Use this command to define how the SE5082 powers up. Normally, the instrument powers up with its parameters rest to factory default setting however, if one requires that the instrument powers up with its last setting, this could be changed using this command. Using the “SET” option is not recommended in conjunction with long waveform segments because the instrument powers down and then up only after the waveforms have been stored in its internal flash memory; an operation that may last very long.

Parameters

Name	Type	Default	Description
DEFault	Discrete	DEF	Selects the factory reset as the power up setting of the instrument.
SETup	Discrete		Selects the last setup before power down as the power up setting for the instrument.

Response

The SE5082 will return DEF or SET, depending on the present power up setting.

Example

Command `:SYST:POW SET`

Query `:SYST:POW?`

IEEE-STD-488.2 Common Commands and Queries

Since most instruments and devices in an ATE system use similar commands that perform similar functions, the IEEE-STD-488.2 document has specified a common set of commands and queries that all compatible devices must use. This avoids situations where devices from various manufacturers use different sets of commands to enable functions and report status.

The IEEE-STD-488.2 treats common commands and queries as device dependent commands. For example, *TRG is sent over the bus to trigger the instrument. Some common commands and queries are optional, but most of them are mandatory.

The following is a complete listing of all common-commands and queries, which are used by the SE5082

***RST** - Resets the generator to its default state.

***TRG** - Triggers the generator from the remote interface. This command affects the generator if it is first placed in the Trigger or Burst mode of operation and the trigger source is set to "BUS".

***CLS** - Clear the Status Byte summary register and all event registers.

***OPC?** - Returns "1" to the output buffer after all the previous commands have been executed. *OPC? is used for synchronization between a controller and the instrument using the MAV bit in the Status Byte or a read of the Output Queue. Reading the response to the *OPC? query has the advantage of removing the complication of dealing with service requests and multiple polls to the instrument. However, both the system bus and the controller handshake are in a temporary hold-off state while the controller is waiting to read the *OPC? query response.

***STB?** - Query the Status Byte summary register. The *STB? command is similar to a serial poll but is processed like any other instrument command. The *STB? command returns the same result as a serial poll, but the "request service" bit (bit 6) is not cleared if a serial poll has occurred.

***IDN?** - Query the generator’s identity. The returned data is organized into four fields, separated by commas. The generator responds with its manufacturer and model number in the first two fields, and may also report its serial number and options in fields three and four. If the latter information is not available, the device must return an ASCII 0 for each.

***OPT?** - Returns the number of channels: 1 or 2, memory size: 32 or 64.

Instrument Programming Over Remote Interface

The Model SE5082 can be programmed with SCPI commands from one of three remote interfaces: LAN USB and GPIB. Only a single remote interface can be used at a time. These interfaces not only differ by how they mechanically interface to the instrument but also they are very different in the way that the instrument reacts to their commands. For example, the generator does not require any LAN drivers to connect to a host computer but without a proper USB driver installed on the computer, the generator will not link and will not receive commands. There are also hardware issues that differ from interface to interface, for example, there is one standard in the market for the USB interface and driving hardware but, on the other hand, the GPIB standard being developed back in the early 80's of the last decade, triggered competition between vendors that lead to multiple and parallel development of GPIB interfaces and drivers and, by that, although there is one common standard (IEEE-488.2), each of the interfacing hardware reacts a bit different to commands and has its own peculiarities when it comes to timing, response and handshaking principals.

When the generator receives waveform coordinates or table data, the amount of data that is being transferred to the instrument is huge and can reach 128,000,000 words of data, if both channels are being written with waveform data. Waveform coordinates are being downloaded using special technique, which is described earlier in the manual so, it is important to remember that, compared to SCPI commands that may take a few milliseconds to program, downloading time of waveforms and table data is in the order of seconds.

Some of the interfaces are not so forgiving when it comes to such differences in timing, especially when control commands are chained with waveforms and table data and therefore, one must install the proper control and suitable handshake that inhibits the flow of data till the generator is fully ready to accept the next command, query or waveform coordinates.

In general, the model SE5082 should always be ready to accept commands but bear in mind that commands are parsed, interpreted by the firmware and then distributed to various registers and internal busses that control the instrument functions. Some of the commands trigger just one event but some commands have to access and set a huge bank of registers. For example, a *trg command flips one port but *rst has to access and reset hundreds of registers so, obviously, there will be timing issue if one expects that *trg and *rst take the same amount of time to execute.

The bottom line is that one must place proper delays and inhibits before and after each command or data download otherwise, the generator will miss data or generate errors that will interfere with its normal operation.

There is also the problem of timeout. Different programs expect that certain operations do not exceed time limit that is set by the system. Often, programmers set this value vaguely without really investigating what is the real timeout that is required to execute a command and, in some case, when an operation exceeds the programmed time out value, the instruments simply reports that it timed out and the reason is not always obvious to the programmer.

So, the recommendation for sending SCPI commands is to keep them short as possible. Also, bear in mind that control commands should not be mixed with waveform data because data is transferred to the generator using special binary format that is routed directly to the arbitrary working memory and hence there is danger that, if interrupted unexpectedly, the transmission will either time out or, in extreme cases will lock out the interface and only power recycling will revive the generator.

An exception to the above recommendation is the *rst command. This command is usually sent at the beginning of a program to make sure that the instrument is programmed from its default setting. It is possible that a program modified just a few parameters but a hard reset does not care and has to access and process hundreds of settings and therefore the generator is not ready to accept any commands till all of its internal registers were reset. A fixed delay of around 1 to 2 seconds is sufficient to avoid hiccups but, if timing is crucial, it is also possible to use the STB register to discover when the generator has recovered all parameters and is ready for the next set of commands.

Using Telnet for Interactive Text-Oriented Communication

When the LAN remote interface is active, one can communicate with the instru- ment from command-line shell using Telnet1.

The instrument uses port no. 5024 for Telnet communication. So if for example the instrument's TCP-IP Address is 192.168.0.170, then Telnet communication can be opend from command-line shell by typing:

```
telnet 192.168.0.170 5024
```

After that a welcome message is displayed, and SCPI commands can be typed in interactive manner:

```
Welcome to Tabor Electronics SE5082 (0000458752)
>>*IDN?
Tabor Electronics,SE5082,0000458752,4.10
>>:OUTPut ON
```

>>

However, while Telnet is good for interactive text-oriented communication, it does not support binary-data transmission, and does not suit a real programming like one would do with (for example) C, C++, C#, Python, MatLab or LabView.

Note

On Windows, Telnet Client is disabled by default, and it can be enabled as explained in: https://social.technet.microsoft.com/wiki/contents/articles/910.windows-7-enabling-telnet-client.aspx.

Key Aspects of Remote Instrument Programming

There are three main alternatives when selecting the communication infrastructure:

Instrument's IVI-Driver

The IVI driver implements the standard interface of Arbitrary Waveform/Function Generator (as defined by the IVI-Foundation) plus other useful functions. It comes in two formats: COM DLL and .NET DLL so it can be used by any .NET application and also by LabView, MatLab and Python (on Windows). In addition the driver is based on NI-VISA DLL and requires some installations.

Standard VISA library

This is the most common approach, because it gives high flexibility and ease of coding however, it does require some installations. We recommend on the NI-VISA library that can be downloaded from *www.ni.com/downloads/ni-drivers/* (registration required). It can be used from various platforms including LabView, C++, C#, MatLab and Python (check PyVisa). We supply open source modules (for C++, MatLab and Python) that implement common utilities (using the standard VISA library). It is recommended to work with those modules or at least use them as a coding reference (the code examples below give the main guide lines).

Note

There are versions of the NI-VISA library for Windows MAC and some Linux distributions (although not formally supported, installation instructions for Ubuntu can be found on the web).

User Implemented Communication functions

Users can implement their own communication functions depending on the interface used. For LAN Interface one only needs to open a standard TCPIP socket and for USB Interface USBTMC is a good starting point. Tabor offers a Python module that implements LAN and USB communication without VISA. In addition, Octave has its own instrument-control package that does not use VISA library.

Commands Execution Synchronization

The instrument executes the SCPI commands it receives from the remote controller in a serial manner (one after the other). Furthermore, each SCPI statement (with one or more commands separated by semi-colons) is parsed as an atomic unit and the answers to all the queries in the statement are sent back together in a single response after the instrument executes all the commands in the statement. There are several things to be aware of :

- A SCPI statement ends either with a new-line character or with a binary- data header followed by binary-data.
- The maximal length of a SCPI statement is 256 ASCII characters (including the binary-data header, but not including the binary data itself).
- If a statement contains one or more queries and ends with binary-data, then the response will be sent back after receiving all the data.
- The response also ends with a new-line character, and if it contains multiple answers they are separated by semi-colons.

The crucial point is that even though the instrument buffers the incoming SCPI statements in internal queue, the remote controller should wait until the instrument finishes executing the current command before it sends the next one.

The best way to do this is by adding *OPC? to any SCPI statement that contains no other query, and then read back the response (which shall contain the character ‘1’ followed by a new-line

character). In this way the controller can be sure that the former command was completed, and the instrument is ready for the next command.

Note that depending on the time it takes to execute the command and on the controller communication timeout, one may also need to add some delay after the composed statement is sent and before the response is read back.

In most cases the composed statement can be treated as a single query. For example, send ":INSTrument:SELect 1; :OUTPut ON; *OPC?\n" as a single query. When sending binary data to instrument, the *OPC? should be added before the binary header, and the response should be read back separately after sending all data. The following is an example of how to use the *OPC? When sending binary data:

1. Define segment 1 of 1024 points, by sending the query: `:TRACe:DEF 1,1024; *OPC?\n`

2. Select segment 1 as the active one, by sending the query: `:TRACe:SEL 1;*OPC?\n`

3. Send binary header (for 2048 binary-data bytes) with OPC:

`*OPC?;:TRACe:DATA #42048`

4. Send the 2048 binary-data bytes (no new line and no read operation between the header and the data).

5. Read the response to the OPC query that was sent together with the binary-data header.

This approach (together with additional useful utilities) is fully implemented in the IVI Driver and in all the other function-libraries (C++, Python, MatLab) that can be found in www.taborelec.com. It is highly recommended to use one of those libraries, or at least take them as a programming reference. The following code snippets however give the guide lines.

Note

In case of GPIB interface, the procedure for sending binary-data is a bit different (for details please refer to the code examples below).

VISA based Programming Examples

The following examples give guide lines for programming the instrument using VISA library. There are three examples, one in C\C++, one in Python and one in Matlab. There are several important points to remember:

1. When working with LAN Interface, one should use VISA Interface of type TCPIP-SOCKET rather than TCPIP-INSTRUMENT. The instrument uses socket no. 5025 for LAN communication, so if for example the instrument's TCPIP Address is `192.168.0.170`, then the corresponding VISA resource name is `TCPIP::192.168.0.170::5025::SOCKET`
2. The following VISA attributes should be initialized:

- Set the `VI_ATTR_WR_BUF_OPER_MODE` attribute to: `VI_FLUSH_ON_ACCESS`.
- Set the `VI_ATTR_RD_BUF_OPER_MODE` attribute to: `VI_FLUSH_ON_ACCESS`.

If the interface type is `VI_INTF_TCPIP` also:

- Set the `VI_ATTR_TERMCHAR_EN` attribute to: `VI_TRUE`.
- Set the `VI_ATTR_TERMCHAR` attribute to: '`\n`'.

C / C++ Example

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdarg.h>
#include <math.h>
#include <visa.h>

#ifdef __GNUC__
typedef __const char *__restrict msgfmt_t;

#define _printf_format_(n1, n2) \
    __attribute__ ((__format__ (__printf__, n1, n2)))

// forward declaration
ViStatus SendCmd(ViSession vi, msgfmt_t cmd_format, ...) _printf_format_(2,3);
#else
typedef const char * msgfmt_t;
#endif // __GNUC__

/*****************************************************************************//**
// @brief Paranoia Level (0: low, 1: normal, 2: high)
// ****************************************************************************/
int g_paranoiaLevel = 1;

/*****************************************************************************//**
// @brief Check if the VISA status code is okay.
// ****************************************************************************/
#define VI_STATUS_OK(viStat) ((viStat)>=0)

/*****************************************************************************//**
// @brief The maximal length of SCPI statement
// ****************************************************************************/
#define MAX_SCPI_STATEMENT_LEN 256

/*****************************************************************************//**
// @brief VISA timeout in milliseconds
// ****************************************************************************/
#define VISA_TIMEOUT_MSEC      10000 //VI_INFINITE

/*****************************************************************************//**
// @brief VISA read-buffer size (in bytes)
// ****************************************************************************/
#define VISA_READ_BUFF_SIZE    4096

/*****************************************************************************//**
// @brief VISA write-buffer size (in bytes)
// ****************************************************************************/
#define VISA_WRITE_BUFF_SIZE   30000

/*****************************************************************************//**
// @brief The chunk size in bytes when writing binary-data to LAN/USB Interface
// ****************************************************************************/
#define CHUNK_SIZE_LAN_USB   30000
```

```
/*****************************************************************************//**
// @brief The chunk size in bytes when writing binary-data to GPIB Interface.
// ****************************************************************************/
#define CHUNK_SIZE_GPIB      30000

/*****************************************************************************//**
// @brief Max Read-Status-Byte trials on writing binary-data to GPIB Interface.
// ****************************************************************************/
#define GPIB_STB_TRIALS      2000

/*****************************************************************************//**
// @brief The PI constant
// ****************************************************************************/
#define MATH_PI              (3.14159265)

/*****************************************************************************//**
// @brief The maximal wave-point value
// ****************************************************************************/
#define MAX_WAV_VAL          ((1<<12)-1)

/*****************************************************************************//**
// @brief The minimal segment length (in points)
// ****************************************************************************/
#define MIN_SEGMENT_LEN      (384)

/*****************************************************************************//**
// @brief Segment quantum (in points)
// ****************************************************************************/
#define SEGMENT_QUANTUM      (32)

/*****************************************************************************//**
// @brief Trace error message (with printf format).
// ****************************************************************************/
#define TRACE_ERR_MSG(msgFmt, ...) do {\
    fprintf(stderr, "ERROR: " msgFmt "\n", ##__VA_ARGS__); fflush(NULL); \
} while(0)

/*****************************************************************************//**
// @brief Trace warning message (with printf format).
// ****************************************************************************/
#define TRACE_WRN_MSG(msgFmt, ...) do {\
    fprintf(stdout, "WARN: " msgFmt "\n", ##__VA_ARGS__); fflush(NULL); \
} while(0)

/*****************************************************************************//**
// @brief Trace info message (with printf format).
// ****************************************************************************/
#define TRACE_INF_MSG(msgFmt, ...) do {\
    fprintf(stdout, "INFO: " msgFmt "\n", ##__VA_ARGS__); fflush(NULL); \
} while(0)

/*****************************************************************************//**
// @brief Get description of the specified VISA error code.
// @param vi    VISA session handle.
// @param viErr VISA error code.
// @return the description of the specified VISA error code.
```

```
// ****************************************************************************/
const char* GetVisaErrDesc(ViSession vi, ViStatus viErr)
{
    static char desc[256];

    viErr = viStatusDesc(vi, viErr, desc);
    if (VI_SUCCESS != viErr)
    {
        snprintf(desc, sizeof(desc), "??");
    }

    return desc;
}

/*****************************************************************************//**
// @brief Initialize I/O attributes of the given VISA session.
//
// @param vi       VISA session handle.
// @param intfType VISA Interface type (zero if unknown).
// @return VISA status code (negative upon error).
// ****************************************************************************/
ViStatus InitVisaAttributes(ViSession vi, long intfType = 0)
{
    ViStatus viErr = VI_SUCCESS;

    if (intfType <= 0)
        viErr = viGetAttribute(vi, VI_ATTR_INTF_TYPE, &intfType);

    if (VI_STATUS_OK(viErr))
        viErr = viSetAttribute (vi, VI_ATTR_TMO_VALUE, VISA_TIMEOUT_MSEC);
    if (VI_STATUS_OK(viErr))
        viErr = viSetBuf (vi, VI_READ_BUF, VISA_READ_BUFF_SIZE);
    if (VI_STATUS_OK(viErr))
        viErr = viSetBuf (vi, VI_WRITE_BUF, VISA_WRITE_BUFF_SIZE);
    if (VI_STATUS_OK(viErr))
        viErr = viSetAttribute (vi,
                    VI_ATTR_WR_BUF_OPER_MODE, VI_FLUSH_ON_ACCESS);
    if (VI_STATUS_OK(viErr))
        viErr = viSetAttribute (vi,
                    VI_ATTR_RD_BUF_OPER_MODE, VI_FLUSH_ON_ACCESS);
    if (VI_INTF_TCPIP == intfType)
    {
        if (VI_STATUS_OK(viErr))
            viErr = viSetAttribute (vi, VI_ATTR_TERMCHAR_EN, VI_TRUE);
        if (VI_STATUS_OK(viErr))
            viErr = viSetAttribute (vi, VI_ATTR_TERMCHAR, '\n');
    }

    if (!VI_STATUS_OK(viErr))
    {
        TRACE_ERR_MSG("%s failed with viErr=0x%04lx (%s).",
            __FUNCTION__, (unsigned long)viErr, GetVisaErrDesc(vi, viErr));
    }

    return viErr;
}
```

```
/*****************************************************************************//**
// @brief Send SCPI command (in printf-format) to instrument.
//
// If the paranoia-level is 1 then "*OPC?" is appended to the SCPI command,
// and the composed SCPI statement is sent to instrument as a single query.
// This way the caller can be sure that the command was completed.
//
// If the paranoia-level is 2 (or above) then ":SYST:ERR?" is appended
// to the SCPI command, the composed SCPI statement is sent to instrument
// as a single query, the response is checked and a warning is printed if
// it contains error-code other than 0 (which means no error).
// It is useful for debug.
//
// @param vi          VISA Session handle.
// @param cmd_format the command's (printf like) format
//                   followed by the format arguments.
// @return VISA status code (negative upon error).
// ****************************************************************************/
ViStatus SendCmd(ViSession vi, msgfmt_t cmd_format, ...)
{
    va_list vl;
    long cmdLen;
    ViStatus viErr = VI_SUCCESS;
    char cmdBuf[MAX_SCPI_STATEMENT_LEN + 1];

    va_start(vl, cmd_format);
    cmdLen = vsnprintf(cmdBuf, sizeof(cmdBuf), cmd_format, vl);
    va_end(vl);

    // Remove trailing whitespace (including new-line character):
    for (; cmdLen > 0 && isspace(cmdBuf[cmdLen-1]); --cmdLen)
    {
        cmdBuf[cmdLen-1] = '\0';
    }

    if (cmdLen > 0 && (unsigned long)cmdLen < sizeof(cmdBuf))
    {
        long n;
        if (g_paranoiaLevel == 1)
        {
            // Append "; *OPC?\n"
            n = snprintf(&cmdBuf[cmdLen],
                sizeof(cmdBuf) - cmdLen, "; *OPC?\n");
        }
        else if (g_paranoiaLevel > 1)
        {
            // Append "; :SYST:ERR?\n"
            n = snprintf(&cmdBuf[cmdLen],
                sizeof(cmdBuf) - cmdLen, "; :SYST:ERR?\n");
        }
        else
        {
            // Append "\n"
            n = snprintf(&cmdBuf[cmdLen], sizeof(cmdBuf) - cmdLen, "\n");
        }

        cmdLen = (n < 0 ? n : cmdLen + n);
```

```
    }

    if (cmdLen < 0 || cmdLen >= MAX_SCPI_STATEMENT_LEN)
    {
        viErr = VI_ERROR_INV_LENGTH; // buffer overflow
    }
    else if (g_paranoiaLevel == 1)
    {
        int resp_val = 0;
        // Send the composed statement as query:
        viErr = viQueryf(vi, (ViString)cmdBuf, (ViString)"%d", &resp_val);
    }
    else if (g_paranoiaLevel > 1)
    {
        char respBuf[256];
        respBuf[0] = 0;
        // Send the composed statement as query:
        viErr = viQueryf(vi, (ViString)cmdBuf, (ViString)"%t", respBuf);

        // The expected response is "<error-code>,<error-description>"
        // (where error-code zero means no error)
        if (VI_STATUS_OK(viErr) && respBuf[0] != '0')
        {
            unsigned long n;
            for (n = strlen(cmdBuf); n > 0 && isspace(cmdBuf[n-1]); --n)
            {
                cmdBuf[n-1] = '\0';
            }
            for (n = strlen(respBuf); n > 0 && isspace(respBuf[n-1]); --n)
            {
                respBuf[n-1] = '\0';
            }
            TRACE_WRN_MSG("%s error \'%s\' on command \"%s\".",
                    __FUNCTION__, respBuf, cmdBuf);

            // Clear the instrument's errors-queue:
            snprintf(cmdBuf, sizeof(cmdBuf), "*CLS; *OPC?\n");
            viErr = viQueryf(vi, (ViString)cmdBuf, (ViString)"%t", respBuf);
        }
    }
    else
    {
        // Send the (bare) command:
        viErr = viPrintf(vi, cmdBuf);
    }

    if (!VI_STATUS_OK(viErr))
    {
        TRACE_ERR_MSG("%s failed with viErr=0x%04lx (%s).",
            __FUNCTION__, (unsigned long)viErr, GetVisaErrDesc(vi, viErr));
    }

    return viErr;
}

/*****************************************************************************//**
// @brief Make binary-data header.
```

```
//
// Helping function used by ```DownloadBinDat()```.
//
// @param prefix1   the 1st part of the binary-data-header prefix.
// @param prefix2   the 2nd part of the binary-data-header prefix.
// @param binDatSz  the binary-data size in bytes.
// @param outBuf    buffer for the output string.
// @param outBufLen the length of the \c outBuf buffer.
// @return the length of the binary-data header (negative on error).
// ****************************************************************************/
long MakeBinDatHeader(
    const char*   prefix1,
    const char*   prefix2,
    unsigned long binDatSz,
    char*         outBuf,
    unsigned long outBufLen)
{
    long n = 0;
    char datSizeStr[32] = { 0 };

    if (NULL == outBuf) { outBufLen = 0; }
    if (outBufLen > 0) { outBuf[0] = '\0'; }
    if (NULL == prefix1) { prefix1 = ""; }
    if (NULL == prefix2) { prefix2 = ""; }

    n = snprintf(datSizeStr, sizeof(datSizeStr), "%lu", binDatSz);
    if (n < 0 || n > 9 || (unsigned long)n >= sizeof(datSizeStr))
    {
        n = -1; // the data-size in bytes must contain up to 9 decimal digits
    }
    else if (*prefix1 != '\0' && *prefix2 != '\0')
    {
        n = snprintf(outBuf, outBufLen,
                "%s;%s#%ld%lu", prefix1, prefix2, n, binDatSz);
    }
    else
    {
        n = snprintf(outBuf, outBufLen,
                "%s%s#%ld%lu", prefix1, prefix2, n, binDatSz);
    }

    return n;
}

/*****************************************************************************//**
// @brief Write binary-data block to instrument.
//
// Helping function used by ```DownloadBinDat()```.
//
// @param vi          VISA session handle.
// @param binDat      the binary-data block.
// @param binDatLen   the length in bytes of the ```binDat``` block.
// @param maxChunkLen the maximal length in bytes of written chunk.
// @return VISA status code (negative upon error).
// ****************************************************************************/
ViStatus WriteBinaryBlock(
    ViSession           vi,
```

```
    const unsigned char* binDat,
    unsigned long        binDatLen,
    unsigned long        maxChunkLen)
{
    ViStatus viErr = VI_SUCCESS;

    if (NULL != binDat && binDatLen > 0)
    {
        unsigned long offset = 0;
        if (0 == maxChunkLen) { maxChunkLen = 1; }

        while (offset < binDatLen)
        {
            if (offset + maxChunkLen > binDatLen)
            {
                maxChunkLen = binDatLen - offset;
            }
            viErr = viWrite(
                vi, (ViBuf)&binDat[offset], (ViUInt32)maxChunkLen, VI_NULL);
            if (!VI_STATUS_OK(viErr))
            {
                break;
            }

            offset += maxChunkLen;
        }
    }
    return viErr;
}

/*****************************************************************************//**
// @brief Prepare for downloading binary-data (helping function).
//
// Helping function used by ```DownloadBinDat()```.
//
// @param vi       VISA session handle.
// @param intfType VISA Interface Type (zero if unknown).
// @return VISA status code (negative upon error).
// *****************************************************************************/
ViStatus PreDownloadBinaryDat(ViSession vi, long intfType = 0)
{
    ViStatus viErr = VI_SUCCESS;

    // Query the VISA Interface Type (if it is not given):
    if (intfType <= 0)
    {
        intfType = 0;
        viErr = viGetAttribute(vi, VI_ATTR_INTF_TYPE, (void *)&intfType);
    }

    // The following is needed for GPIB Interface:
    if (VI_INTF_GPIB == intfType && VI_STATUS_OK(viErr))
    {
        long stbLoops = 0;
        ViUInt16 stb = 0x10;
        char opc_query[] = { "*OPC?\n" };
```

```
        viErr = viWrite(
            vi, (ViBuf)opc_query, (ViUInt32)(strlen(opc_query)), VI_NULL);

        // Wait till the status-byte changes from 0x10.
        for ( ; stbLoops < GPIB_STB_TRIALS && viErr >= 0; ++stbLoops)
        {
            viErr = viReadSTB (vi, &stb);
            if ((stb & 0x10) != 0x10) { break; }
        }

        if ( VI_STATUS_OK(viErr))
        {
            ViUInt32 bActual;
            unsigned char bufRead[256];
            viErr = viRead(vi, bufRead, (long)sizeof(bufRead), &bActual);
        }
    }

    return viErr;
}

/*****************************************************************************//**
// @brief Download binary data to instrument.
//
// @param prefix     the binary-header prefix (i.e. ":TRAC:DATA").
// @param binData    the binary data.
// @param binDatSize the binary-data size (in bytes).
// @param intfType   the VISA Interface type (zero if unknown).
// @return VISA status code (negative upon error).
// *****************************************************************************/
ViStatus DownloadBinDat(
    ViSession            vi,
    const char*          prefix,
    const unsigned char* binData,
    unsigned long        binDatSize,
    long                 intfType = 0)
{
    ViStatus viErr = VI_SUCCESS;

    long headerLen = 0;
    char binDatHeader[MAX_SCPI_STATEMENT_LEN];

    if (NULL == binData) { binDatSize = 0; }

    if (NULL == prefix) { prefix = ""; }

    // Query the VISA Interface Type (if it is not given):
    if (intfType <= 0)
    {
        intfType = 0;
        viErr = viGetAttribute(vi, VI_ATTR_INTF_TYPE, (void *)&intfType);
    }

    // Prepare binary-data header:
    if ( VI_STATUS_OK(viErr))
    {
        if (VI_INTF_GPIB != intfType && g_paranoiaLevel > 0)
```

```
    {
        // Prepare binary-data header with "*OPC?" in its prefix:
        headerLen = MakeBinDatHeader("*OPC?", prefix,
                    binDatSize, binDatHeader, sizeof(binDatHeader));
    }
    else
    {
        // Prepare binary-data header without "*OPC?" in its prefix:
        headerLen = MakeBinDatHeader(NULL, prefix,
                    binDatSize, binDatHeader, sizeof(binDatHeader));
    }

    if (headerLen < 0 ||
        (unsigned long)headerLen >= sizeof(binDatHeader))
    {
        headerLen = 0;
        viErr = VI_ERROR_INV_LENGTH; // buffer overflow
    }
}

// Prepare for downloading binary-data:
if ( VI_STATUS_OK(viErr))
    viErr = PreDownloadBinaryDat(vi, intfType);

if ( VI_STATUS_OK(viErr))
{
    ViStatus rc;
    ViInt32 tmo1 = 0, tmo2 = 0;
    // Read current timeout
    rc = viGetAttribute(vi, VI_ATTR_TMO_VALUE, &tmo1);
    if  ((tmo2 = tmo1) < (long)(binDatSize / 20) &&  VI_STATUS_OK(rc))
    {
        // Increase the timeout:
        tmo2 = binDatSize / 20;
        rc = viSetAttribute (vi, VI_ATTR_TMO_VALUE, tmo2);
    }

    // Write the binary-data header:
    viErr = viWrite(vi,
            (ViBuf)binDatHeader, (ViUInt32)headerLen, VI_NULL);

    // Write the binary data:
    if ( VI_STATUS_OK(viErr))
    {
        if (VI_INTF_GPIB == intfType)
        {
            viErr = WriteBinaryBlock(
                vi, binData, binDatSize, CHUNK_SIZE_GPIB);
        }
        else
        {
            viErr = WriteBinaryBlock(
                vi, binData, binDatSize, CHUNK_SIZE_LAN_USB);

            if (g_paranoiaLevel > 0 &&  VI_STATUS_OK(viErr))
            {
                // Read the response to the "*OPC?" query
```

```
                    // that was sent with the binary-data header:
                    ViUInt32 bActual;
                    unsigned char bufRead[256];
                    viErr = viRead(vi, bufRead,
                                (long)sizeof(bufRead), &bActual);
                }
            }
        }

        // Restore the original timeout:
        if (tmo2 != tmo1)
        {
            rc = viSetAttribute (vi, VI_ATTR_TMO_VALUE, tmo1);
            if (!VI_STATUS_OK(rc)) { viErr = rc; }
        }
    }

    if (VI_STATUS_OK(viErr) && g_paranoiaLevel > 1)
    {
        // Query instrument's internal error:
        char respBuf[256];
        char cmdBuf[MAX_SCPI_STATEMENT_LEN] = { ":SYST:ERR?\n" };

        respBuf[0] = '\0';
        viErr = viQueryf(vi, (ViString)cmdBuf, (ViString)"%t", respBuf);

        // The expected response is "<error-code>,<error-description>"
        // (where error-code zero means no error)
        if (VI_STATUS_OK(viErr) && respBuf[0] != '0')
        {
            unsigned long n;
            for (n = strlen(respBuf); n > 0 && isspace(respBuf[n-1]); --n)
            {
                respBuf[n-1] = '\0';
            }
            TRACE_WRN_MSG(
                "%s error \'%s\' while writing binary-data: \"%s\".",
                __FUNCTION__, respBuf, binDatHeader);

            // Clear the instrument's errors-queue:
            snprintf(cmdBuf, sizeof(cmdBuf), "*CLS; *OPC?\n");
            viErr = viQueryf(vi, (ViString)cmdBuf, (ViString)"%t", respBuf);
        }
    }

    if (!VI_STATUS_OK(viErr))
    {
        TRACE_ERR_MSG("%s failed with viErr=0x%04lx (%s).",
            __FUNCTION__, (unsigned long)viErr, GetVisaErrDesc(vi, viErr));
    }

    return viErr;
}

/*****************************************************************************//**
// @brief The main function.
//
```

```
// Connect to instrument and download sine-wave of 1024 points to channels 1&2.
// ***************************************************************************/
int main(int argc, char* argv[])
{
    ViStatus viErr;
    ViSession vi = 0;
    ViSession defaultRM = 0;

    long intfType = 0;
    char rscName[256] = { "TCPIP::192.168.0.170::5025::SOCKET" };
    if (argc > 1)
    {
        snprintf(rscName, sizeof(rscName), "%s", argv[1]);
    }

    g_paranoiaLevel = 2; // high paranoia level (for debug)

    // Open VISA session ..
    viErr = viOpenDefaultRM(&defaultRM);
    if (VI_STATUS_OK(viErr))
        viErr = viOpen(defaultRM, rscName, VI_NULL,VI_NULL, &vi);

    if (VI_STATUS_OK(viErr))
        viErr = viGetAttribute(vi, VI_ATTR_INTF_TYPE, &intfType);

    if (VI_STATUS_OK(viErr))
        viErr = InitVisaAttributes(vi, intfType);

    if (!VI_STATUS_OK(viErr))
    {
        TRACE_ERR_MSG("%s failed to connect to: \"%s\", viErr=0x%04lx (%s).",
            __FUNCTION__, rscName, (unsigned long)viErr,
            GetVisaErrDesc(defaultRM, viErr));
    }
    else
    {
        char respBuf[256];

        viErr = viQueryf(vi, (ViString)"*IDN?\n", (ViString)"%t", respBuf);
        if (VI_STATUS_OK(viErr))
            TRACE_INF_MSG("Connected to: %s", respBuf);

        // Clear error-queue and reset the instrument:
        if (VI_STATUS_OK(viErr))
            viErr = SendCmd(vi, "*CLS; *RST");

        if (VI_STATUS_OK(viErr))
        {
            // Create sine-wave of 1024 points
            double x;
            unsigned n, segLen = 1024;
            unsigned short waveDat[1024];

            TRACE_INF_MSG("Downloading sine-wave "
                "of %u points to channels 1&2 ...", segLen);

            for (n = 0; n < segLen; ++n)
```

```
            {
                x = sin(2.0 * MATH_PI * n / segLen);
                x = 0.5 * (x + 1.0) * MAX_WAV_VAL;

                waveDat[n] = (unsigned short)(x + 0.5);
            }

            for (n = 0; n < 2 && VI_STATUS_OK(viErr); ++n)
            {
                // Select channel and User-Mode:
                viErr = SendCmd(vi, ":INST:SEL %d; :FUNC:MODE USER", n+1);

                // Define segment 1 and select it:
                if (VI_STATUS_OK(viErr))
                    viErr = SendCmd(vi,
                            ":TRAC:DEF 1,%d; :TRAC:SEL 1", segLen);

                // Download the binary wave data:
                if (VI_STATUS_OK(viErr))
                {
                    unsigned long datSz = segLen * sizeof(unsigned short);
                    const unsigned char* dat = (const unsigned char*)waveDat;
                    viErr = DownloadBinDat(
                            vi, ":TRAC:DATA", dat, datSz, intfType);
                }

                // Turn output on:
                if (VI_STATUS_OK(viErr))
                    viErr = SendCmd(vi, ":OUTP ON");
            }
        }

        if (VI_STATUS_OK(viErr))
        {
            // Query and print the instrument's last error:
            respBuf[0] = 0;
            viErr = viQueryf(
                vi, (ViString)":SYST:ERR?\n", (ViString)"%t", respBuf);
            if (VI_STATUS_OK(viErr))
            {
                TRACE_INF_MSG("%s", respBuf);
            }
        }

        if (!VI_STATUS_OK(viErr))
        {
            TRACE_ERR_MSG(
                "%s failed with viErr=0x%04lx (%s).",
                __FUNCTION__, (unsigned long)viErr,
                GetVisaErrDesc(vi, viErr));
        }
    }

    // Close VISA session ..
    if (vi) { viClose(vi); }
    if (defaultRM) { viClose(defaultRM); }
```

```
    return 0;
}
```

Python Example

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import math
import sys
import numpy as np
import warnings
import ctypes
import visa
import pyvisa.constants as vc

class Consts(object):
    '''Constants.'''

    ## The maximal length of SCPI statement
    MAX_SCPI_STATEMENT_LEN = 256

    ## VISA timeout in milliseconds
    VISA_TIMEOUT_MSEC = 10000

    ## VISA read-buffer size (in bytes)
    VISA_READ_BUFF_SIZE = 4096

    ## VISA write-buffer size (in bytes)
    VISA_WRITE_BUFF_SIZE = 30000

    ## Chunk size in bytes when writing binary-data to LAN/USB Interface.
    CHUNK_SIZE_LAN_USB = 30000

    ## Chunk size in bytes when writing binary-data to GPIB Interface.
    CHUNK_SIZE_GPIB = 30000

    ## Max Read-Status-Byte trials on writing binary-data to GPIB.
    GPIB_STB_TRIALS = 2000

    ## The maximal wave-point value.
    MAX_WAV_VAL = 2**12 - 1

    ## The minimal segment length (in points).
    MIN_SEGMENT_LEN = 384

    ## Segment quantum (in points).
    SEGMENT_QUANTUM = 32

## Paranoia Level (0: low, 1: normal, 2: high)
g_paranoia_level = 1

def get_paranoia_level():
    '''Get the paranoia-level (0: low, 1: normal, 2: high).'''
    return g_paranoia_level

def set_paranoia_level(paranoia_level):
    '''Set the paranoia-level (0: low, 1: normal, 2: high).'''
```

```
    global g_paranoia_level
    g_paranoia_level = int(g_paranoia_level)

def is_vi_status_ok(vi_status):
    '''Check if the given VISA status code is okay.

    :param vi_status: the VISA status code to test.
    :returns: `True` if the given VISA status code is okay; otherwise `False`.
    '''
    return bool(int(vi_status) >= 0)

def init_visa_attributes(vi, intf_type = 0):
    '''Initialize I/O attributes of the given VISA session.

    :param vi: `pyvisa` instrument.
    :param intf_type: VISA Interface type (zero if unknown).
    :returns: VISA status code (negative upon error).
    '''

    if intf_type <= 0:
        intf_type = vi.get_visa_attribute(vc.VI_ATTR_INTF_TYPE)

    vi.timeout = int(Consts.VISA_TIMEOUT_MSEC)

    vi_err = vi.visalib.set_buffer(
        vi.session, vc.VI_READ_BUF, Consts.VISA_READ_BUFF_SIZE)

    if is_vi_status_ok(vi_err):
        vi_err = vi.visalib.set_buffer(
            vi.session, vc.VI_WRITE_BUF, Consts.VISA_WRITE_BUFF_SIZE)

    if is_vi_status_ok(vi_err):
        vi.read_termination = '\n'
        vi.write_termination = '\n'

    if is_vi_status_ok(vi_err):
        vi.set_visa_attribute(
            vc.VI_ATTR_WR_BUF_OPER_MODE, vc.VI_FLUSH_ON_ACCESS)

    if is_vi_status_ok(vi_err):
        vi.set_visa_attribute(
            vc.VI_ATTR_RD_BUF_OPER_MODE, vc.VI_FLUSH_ON_ACCESS)

    if intf_type == vc.VI_INTF_TCPIP and is_vi_status_ok(vi_err):
        vi.set_visa_attribute(vc.VI_ATTR_TERMCHAR_EN, vc.VI_TRUE)

    if is_vi_status_ok(vi_err):
        vi.clear()

    return vi_err

def send_cmd(vi, cmd_str, paranoia_level=None):
    '''Send SCPI command to instrument.

    If `paranoia_level == 1` then '*OPC?' is appended to the SCPI command,
    and the whole SCPI statement is sent to instrument as a single query.
    This way the caller can be sure that the command was completed.
```

```
    If `paranoia_level > 1` then ':SYST:ERR?' is appended to the SCPI
    command, the whole SCPI statement is sent to instrument as a single query,
    the response is checked and a warning is printed if it contains error-code
    other than 0 (which means no error). This mode is useful for debug.

    If `paranoia_level` is `None` then the global paranoia-level is used.

    :param vi: `pyvisa` instrument.
    :param paranoia_level: if not `None` overrides the global paranoia-level.
    :returns: VISA status code (negative upon error).
    '''
    vi_err = vc.VI_SUCCESS
    if paranoia_level is None:
        paranoia_level = get_paranoia_level()

    if paranoia_level == 1:
        ask_str = cmd_str.rstrip()
        if len(ask_str) > 0:
            ask_str += '; *OPC?'
        else:
            ask_str = '*OPC?'
        _ = vi.ask(ask_str)
    elif paranoia_level >= 2:
        ask_str = cmd_str.rstrip()
        if len(ask_str) > 0:
            ask_str += '; :SYST:ERR?'
        else:
            ask_str = ':SYST:ERR?'
        syst_err = vi.ask(ask_str)
        if not syst_err.startswith('0'):
            syst_err = syst_err.rstrip()
            wrn_msg = 'WARNING: "{0}" after command: "{1}"'
            wrn_msg = wrn_msg.format(syst_err, cmd_str)
            warnings.warn(wrn_msg)
            _ = vi.ask('*CLS; *OPC?') # clear the error-list
    else:
        vi_err = vi.write(cmd_str)

    return vi_err

def make_bin_dat_header(bin_dat_size, header_prefix=None):
    '''Make Binary-Data Header

    :param bin_dat_size: the binary-data total size in bytes.
    :param header_prefix: the header prefix (e.g. ':TRACe:DATA').
    :returns: binary-data header (string)
    '''
    bin_dat_size = int(bin_dat_size)
    dat_sz_str = "{0:d}".format(bin_dat_size)

    if header_prefix is None:
        header_prefix = ''

    bin_dat_header = '{0:s}#{1:d}{2:s}'.format(
        header_prefix, len(dat_sz_str), dat_sz_str)
    return bin_dat_header
```

```
def write_raw_string(vi, wr_str):
    '''Write raw string to device (no termination character is added).

    :param vi: `pyvisa` instrument.
    :param wr_str:  the string to write.
    :returns: VISA status code (negative upon error).
    '''
    count = 0L
    str_len = len(wr_str)
    p_dat = ctypes.cast(wr_str, ctypes.POINTER(ctypes.c_byte))
    ul_sz = ctypes.c_ulong(str_len)
    p_ret = ctypes.cast(count, ctypes.POINTER(ctypes.c_ulong))
    vi_err = vi.visalib.viWrite(vi.session, p_dat, ul_sz, p_ret)

    return vi_err

def write_raw_bin_dat(vi, bin_dat, dat_size, max_chunk_size = 1024):
    '''Write raw binary data to instrument.

    The binary data is sent in chunks of up to `max_chunk_size` bytes.

    :param vi: `pyvisa` instrument.
    :param bin_dat: the binary data buffer.
    :param dat_size: the data-size in bytes.
    :param max_chunk_size: maximal chunk-size (in bytes).
    :returns: VISA status code (negative upon error).
    '''
    vi_err = vc.VI_SUCCESS
    wr_offs = 0
    count = 0L

    tmo1 = vi.timeout
    if tmo1 < dat_size // 20:
        vi.timeout = dat_size // 20

    try:
        if isinstance(bin_dat, np.ndarray):
            p_dat = bin_dat.ctypes.data_as(ctypes.POINTER(ctypes.c_byte))
        else:
            p_dat = ctypes.cast(bin_dat, ctypes.POINTER(ctypes.c_byte))

        p_cnt = ctypes.cast(count, ctypes.POINTER(ctypes.c_ulong))

        if dat_size <= max_chunk_size:
            ul_sz = ctypes.c_ulong(dat_size)
            vi_err = vi.visalib.viWrite(vi.session, p_dat, ul_sz, p_cnt)
        else:
            while wr_offs < dat_size:
                chunk_sz = min(max_chunk_size, dat_size - wr_offs)
                ul_sz = ctypes.c_ulong(chunk_sz)
                ptr = ctypes.cast(ctypes.addressof(p_dat.contents)
                                  + wr_offs, ctypes.POINTER(ctypes.c_byte))
                vi_err = vi.visalib.viWrite(vi.session, ptr, ul_sz, p_cnt)
                if not is_vi_status_ok(vi_err):
                    break
                wr_offs = wr_offs + chunk_sz
```

```
        finally:
            vi.timeout = tmo1

        return vi_err

    def _pre_download_binary_data(vi, intf_type = 0):
        '''Prepare for downloading binary-data (helping function).

        :param vi: `pyvisa` instrument.
        :param intf_type: VISA Interface Type (zero if unknown).
        :returns: VISA status code (negative upon error).
        '''
        if intf_type <= 0:
            intf_type = vi.get_visa_attribute(vc.VI_ATTR_INTF_TYPE)

        if intf_type == vc.VI_INTF_GPIB:
            _ = vi.write("*OPC?")
            for _ in range(Consts.GPIB_STB_TRIALS):
                status_byte = vi.stb
                if (status_byte & 0x10) == 0x10:
                    break
            _ = vi.read()

        return vc.VI_SUCCESS

    def download_binary_data(vi, pref, bin_dat, dat_size,
                             intf_type=0, paranoia_level=None):
        '''Download binary data to instrument.

        If `paranoia_level >= 1` then '*OPC?' query is added to the prefix
        and the response is read back after sending all the binary data*.
        In that way the caller can be sure that all data was received and
        processed by the instrument.

        (*) In case of GPIB interface the flow is a bit different.

        If `paranoia_level >= 2` the a separate ':SYST:ERR?' query is sent
        to instrument (after reading the '*OPC?' response) and a warning is
        printed if the response starts with error-code other than 0.
        This mode is useful for debug.

        If `paranoia_level` is `None` then the global paranoia-level value is used.

        :param vi: `pyvisa` instrument.
        :param pref: the header prefix (e.g. ':TRACe:DATA').
        :param bin_dat: the binary data buffer.
        :param dat_size: the data-size in bytes.
        :param intf_type: the interface-type (zero if unknown).
        :param paranoia_level: if not `None` overrides the global paranoia-level.
        :returns: VISA status code (negative upon error).
        '''
        if intf_type <= 0:
            intf_type = vi.get_visa_attribute(vc.VI_ATTR_INTF_TYPE)

        if paranoia_level is None:
            paranoia_level = get_paranoia_level()
```

```
    if pref is None:
        pref = ''
    else:
        pref = pref.rstrip()

    add_opc = paranoia_level >= 1 and intf_type != vc.VI_INTF_GPIB
    if add_opc:
        if len(pref) > 0:
            pref = '*OPC? ;' + pref
        else:
            pref = '*OPC? '

    # Make binary-data header:
    dat_header = make_bin_dat_header(dat_size, pref)
    # Send the binary-data header:
    vi_err = write_raw_string(vi, dat_header)
    # Prepare to download binary-data:
    if is_vi_status_ok(vi_err):
        vi_err = _pre_download_binary_data(vi, intf_type)

    # Download binary-data:
    if is_vi_status_ok(vi_err):
        if intf_type == vc.VI_INTF_GPIB:
            max_chunk_size = Consts.CHUNK_SIZE_GPIB
        else:
            max_chunk_size = Consts.CHUNK_SIZE_LAN_USB

        vi_err = write_raw_bin_dat(vi, bin_dat, dat_size, max_chunk_size)

    # Read the response to the *OPC? query that was sent with the header
    if add_opc and is_vi_status_ok(vi_err):
        _ = vi.read()

    # Query the instrument's last internal error:
    if paranoia_level >= 2 and is_vi_status_ok(vi_err):
        syst_err = vi.ask(':SYST:ERR?')
        if not syst_err.startswith('0'):
            syst_err = syst_err.rstrip()
            wrn_msg = 'WARNING: "{0}" after sending binary data'
            wrn_msg = wrn_msg + ' (pref="{1}", dat_size={2})'
            wrn_msg = wrn_msg.format(syst_err, pref, dat_size)
            warnings.warn(wrn_msg)
            _ = vi.ask('*CLS; *OPC?') # clear the error-list

    return vi_err

def get_vi_err_desc(vi, vi_err):
    '''Get VISA Error description.

    :param vi: `pyvisa` instrument.
    :param vi_err: VISA status code.
    :returns: description (string) of the given VISA error code.
    '''
    return vi.visalib.status_description(vi.session, vi_err)

# ----------------------------------------------------------------------------
# Connect to instrument and download sine-wave of 1024 points to channels 1&2.
```

```
# ------------------------------------------------------------------------
if __name__ == '__main__':

    vi = None
    vi_def_rm = None

    try:
        if len(sys.argv) > 1:
            inst_addr = sys.argv[1]
        else:
            inst_addr = 'TCPIP::192.168.0.180::5025::SOCKET'

        set_paranoia_level(2) # high paranoia-level (for debug)

        vi_err = -1
        intf_type = 0

        # Open VISA Session ..
        vi_def_rm = visa.ResourceManager()
        if vi_def_rm is not None:
            vi = vi_def_rm.open_resource(inst_addr)

        if vi is not None:
            intf_type =  vi.get_visa_attribute(vc.VI_ATTR_INTF_TYPE)
            vi_err = init_visa_attributes(vi, intf_type)

        if not is_vi_status_ok(vi_err):
            err_desc = get_vi_err_desc(vi_def_rm, vi_err)
            err_msg = 'ERROR: Failed to open "{0}" ("{1}")'
            err_msg = err_msg.format(inst_addr, err_desc)
            warnings.warn(err_msg)
        else:
            tmp_str = vi.ask('*IDN?')
            print 'Connected to: {0}\n'.format(tmp_str)

            # Clear the internal error-queue and reset the instrument:
            vi_err = send_cmd(vi, '*CLS;*RST')

            if is_vi_status_ok(vi_err):
                # Create sine-wave of 1024 points
                seg_len = 1024

                print 'Downloading sine-wave of ' \
                    + repr(seg_len) + ' points to channels 1&2 ..'

                x = np.linspace(
                    start=0, stop=2*math.pi, num=seg_len, endpoint=False)
                zero_val = Consts.MAX_WAV_VAL / 2.0
                amplitude = Consts.MAX_WAV_VAL / 2.0
                wav = np.sin(x) * amplitude + zero_val
                wav = np.round(wav)
                wav = np.clip(wav, 0, Consts.MAX_WAV_VAL)
                wav = wav.astype(np.uint16)

                for n in range(2):
                    # Select channel and User-Mode:
                    if is_vi_status_ok(vi_err):
```

```
                    vi_err = send_cmd(vi,
                        ':INST:SEL {0}; :FUNC:MODE USER'.format(n+1))

                # Define segment 1 and select it:
                if is_vi_status_ok(vi_err):
                    vi_err = send_cmd(vi,
                        ':TRACe:DEF 1,{0}; :TRACe:SEL 1'.format(seg_len))

                # Download the binary wave data:
                if is_vi_status_ok(vi_err):
                    vi_err = download_binary_data(vi,
                        ':TRACe:DATA', wav, wav.nbytes, intf_type)

                # Turn output on:
                if is_vi_status_ok(vi_err):
                    vi_err = send_cmd(vi,':OUTPut ON');

            # Query and print the instrument's last error:
            if is_vi_status_ok(vi_err):
                tmp_str = vi.ask(':SYST:ERR?')
                print '{0}\n'.format(tmp_str)

        if not is_vi_status_ok(vi_err):
            err_desc = get_vi_err_desc(vi_def_rm, vi_err)
            err_msg = 'ERROR: {0}'.format(err_desc)
            warnings.warn(err_msg)
finally:
    if vi is not None:
        vi.close()
    if vi_def_rm is not None:
        vi_def_rm.close()
```

Matlab Example

```
%%
% ============================================================================
%> @file SE5082.m
%> @brief The \c SE5082 class declaration.
% ============================================================================


% ============================================================================
%> @brief SE5082 Controller.
% ============================================================================
classdef SE5082 < handle

    properties
        %> Paranoia level (0:low, 1:normal, 2:high).
        ParanoiaLevel = 1;
    end

    properties (SetAccess=private)
        %> The instrument's address (VISA resource name).
        InstAddr = '';
        %> The VISA session handle.
        ViSessn = 0;
    end

    properties (Constant=true)
        %> Maximal SCPI-statment length (in characters).
        MAX_SCPI_STATEMENT_LEN = 255;
        %> VISA timeout (seconds).
        VISA_TIMEOUT_SECONDS = 10;
        %> VISA input-buffer size (bytes).
        VISA_IN_BUFF_SIZE = 4096;
        %> VISA output-buffer size (bytes).
        VISA_OUT_BUFF_SIZE = 30000;
        %> binary chunk size (bytes).
        BINARY_CHUNK_SIZE = 30000;
        %> Waiting pause (seconds).
        WAIT_PAUSE_SEC = 0.02;
        %> Maximal wave-point value (2^12 - 1).
        MAX_WAV_VAL = 4095;
        %> Minimal segment length (in points).
        MIN_SEGMENT_LEN = 384;
        %> Segment quantum (in points).
        SEGMENT_QUANTUM = 32;
    end

     methods % public

          % ======================================================================
          %> @brief Class constructor.
          %>
          %> @param instAddr instrument-address (VISA resource name).
          %> @param paranoiaLevel (optional) initialize the paranoia-level.
          %>
```

```
%> @retval obj instance of the \c SE5082 class.
% ==========================================================================
function obj = SE5082(instAddr, paranoiaLevel)

    assert(nargin == 1 || nargin == 2);

    obj.InstAddr = instAddr;

    if ~exist('paranoiaLevel','var'),
        obj.ParanoiaLevel = 1;
    elseif paranoiaLevel < 1,
        obj.ParanoiaLevel = 0;
    elseif paranoiaLevel > 1,
        obj.ParanoiaLevel = 2;
    else
        obj.ParanoiaLevel = fix(paranoiaLevel);
    end

    obj.ViSessn = visa('NI', instAddr);
    set(obj.ViSessn, 'OutputBufferSize', obj.VISA_OUT_BUFF_SIZE);
    set(obj.ViSessn, 'InputBufferSize', obj.VISA_IN_BUFF_SIZE);
    obj.ViSessn.Timeout = obj.VISA_TIMEOUT_SECONDS;
end;

% ==========================================================================
%> @brief Class destructor.
% ==========================================================================
function delete(obj)
    obj.Disconnect();
    delete(obj.ViSessn);
    obj.ViSessn = 0;
end;

% ==========================================================================
%> @brief Open connection to remote instrument.
%>
%> The instrument's address is set in the constructor.
%>
%> @retval ok \c true if succeeded; otherwise, \c false.
% ==========================================================================
function ok = Connect(obj)
   ok = false;
   try
       if strcmp(obj.ViSessn.Status, 'open'),
           ok = true;
       else
           fopen(obj.ViSessn);
           pause(obj.WAIT_PAUSE_SEC);
           ok = strcmp(obj.ViSessn.Status, 'open');
       end
   catch ex,
       msgString = getReport(ex);
       warning('fopen failed:\n%s',msgString);
   end
end;
```

```
% ====================================================================
%> @brief Close the connection to remote instrument.
% ====================================================================
function Disconnect(obj)
    if strcmp(obj.ViSessn.Status, 'open'),
        stopasync(obj.ViSessn);
        flushinput(obj.ViSessn);
        flushoutput(obj.ViSessn);
        fclose(obj.ViSessn);
    end
end;

% ====================================================================
%> @brief Query the instrument's last internal error.
%>
%> If \c bSendCls is set to \c true, and the instrument's last
%> internal-error is not zero, then a "*CLS" command is sent to
%> the instrument in order to clear the instrument's errors queue.
%>
%> @param bSendCls (optional) should send "*CLS" ?
%> @retval errNb the instrument's error number (0 means no-error).
%> @retval errDesc (optional) the instrument's error description.
% ====================================================================
function [errNb, errDesc] = QuerySysErr(obj, bSendCls)
    if ~exist('bSendCls', 'var'),
        bSendCls = false;
    end;

    obj.waitTransferComplete();
   [answer, count, errmsg] = query(obj.ViSessn, 'SYST:ERR?');
   obj.waitTransferComplete();

   if ~isempty(errmsg),
       error('querySysErr() failed: %s', errmsg);
   end;

   sep = find(answer == ',');
   if (isempty(sep) || count <= 0 || answer(count) ~= char(10)),
       warning('querySysErr() received invalid answer: "%s"',...
               answer);
       flushinput(obj.ViSessn);
   end

   if ~isempty(sep) && isempty(errmsg),
       errNb = str2double(answer(1:sep(1) - 1));
       errmsg = answer(sep(1):end);
       if 0 ~= errNb && nargin > 1 && bSendCls,
           query(obj.ViSessn, '*CLS; *OPC?');
       end
   else
       errNb =  -1;
       if isempty(errmsg)
           errmsg = answer;
       end;
   end;
```

```
    if nargout > 1
        errDesc = errmsg;
    end
end;

% ====================================================================
%> @brief Send SCPI command (in printf format) to instrument.
%>
%> If the paranoia-level is 1, then a "*OPC?" query is appended
%> to the SCPI command, and the combined SCPI statement is sent
%> as a single query. In that way the caller can be sure that the
%> command execution was completed.
%>
%> If the paranoia-level is 2, then a ":SYST:ERR?" query is
%> appended to the SCPI command, the combined SCPI statement is
%> sent to the instrument as a single query, and if the response
%> contains error-number other than 0, then a warning is prtinted.
%> This mode is useful for debug.
%>
%> @param cmdFmt the command (printf) format optionally followed
%>               by the format arguments.
% ====================================================================
function SendCmd(obj, cmdFmt, varargin)
    obj.waitTransferComplete();

    if nargin > 2,
        cmdFmt = sprintf(cmdFmt, varargin{1:end});
    end

    resp = '';
    errMsg = '';
    respLen = 0;

    if obj.ParanoiaLevel == 0,
        assert(length(cmdFmt) <= obj.MAX_SCPI_STATEMENT_LEN);
        fprintf(obj.ViSessn, cmdFmt);
        obj.waitTransferComplete();
    elseif obj.ParanoiaLevel == 1,
        cmdFmt = strcat(cmdFmt, ';*OPC?');
        assert(length(cmdFmt) <= obj.MAX_SCPI_STATEMENT_LEN);
        [resp, respLen, errMsg] = query(obj.ViSessn, cmdFmt);
    elseif obj.ParanoiaLevel >= 2,
        cmdFmt = strcat(cmdFmt, ';:SYST:ERR?');
        assert(length(cmdFmt) <= obj.MAX_SCPI_STATEMENT_LEN);
        [resp, respLen, errMsg] = query(obj.ViSessn, cmdFmt);
    end

    if (obj.ParanoiaLevel > 0 && ~isempty(errMsg)),
        error('query(''%s\'') failed\n %s', cmdFmt, errMsg);
    elseif (obj.ParanoiaLevel >= 2 && respLen > 0),
        resp = deblank(resp);
        sep = find(resp == ',');
        if ~isempty(sep),
            errNb = str2double(resp(1:sep(1) - 1));
            if 0 ~= errNb,
                query(obj.ViSessn, '*CLS; *OPC?');
```

```
                warning('System Error #%d after ''%s'' (%s).', ...
                    errNb, cmdFmt, resp);
            end
        end
    end
end;

% ====================================================================
%> @brief Send SCPI query (in printf format) to instrument.
%>
%> @param qformat the SCPI query (printf) format optionally
%>                followed by the format arguments.
%> @retval resp the instrument's response (string).
% ====================================================================
function resp = SendQuery(obj, qformat, varargin)
    obj.waitTransferComplete();
    if nargin == 2,
        [resp, respLen, errMsg] = query(obj.ViSessn, qformat);
    elseif nargin > 2,
        qformat = sprintf(qformat, varargin{1:end});
        [resp, respLen, errMsg] = query(obj.ViSessn, qformat);
    else
        resp = '';
        errMsg = '';
        respLen = 0;
    end

    if ~isempty(errMsg),
        error('SendQuery(''%s\'') failed\n %s', qformat, errMsg);
    end

    if respLen > 0,
        % remove trailing blanks
        resp = deblank(resp);
    end
end;

% ====================================================================
%> @brief Send binary data to instrument.
%>
%> The \c elemType is the element-type in the \c datArray data-
%> array. the supported types are: 'int8', 'uint8' 'char',
%> 'int16', 'uint16', 'int32', 'uint32', 'single', 'int64',
%> 'uint64' and 'double' (the default value is: 'uint8').
%>
%> If the paranoia-level is positive, then a "*OPC?" query is
%> added to the prefix of the binary-data header, and the response
%> is read back after sending all the binary-data. In that way the
%> caller can be sure that all data was received and processed by
%> by the instrument.
%>
%> If the paranoia-level is above 1 then a separate ":SYST:ERR?"
%> query is sent to the instrument after sending all the binary
%> data and reading the "*OPC?" response, and a warning is printed
%> if the response to the ":SYST:ERR?" query contains error other
%> than zero. This mode is useful for debug.
```

```
%>
%> @param pref the binary-header prefix (e.g. ":TRAC:DATA").
%> @param datArray array with the binary-data to send.
%> @param elemType the element-type in \c datArray (e.g. 'uint16')
% ================================================================
function SendBinaryData(obj, pref, datArray, elemType)
    obj.waitTransferComplete();

   if ~exist('pref', 'var'),
       pref = '';
   end
   if ~exist('datArray', 'var')
       datArray = [];
   end
   if ~exist('elemType', 'var'),
       elemType = 'uint8';
       datArray = typecast(datArray, 'uint8');
   end

   numItems = length(datArray);
   switch elemType,
       case { 'int8', 'uint8' 'char' }
           itemSz = 1;
       case { 'int16', 'uint16' }
           itemSz = 2;
       case { 'int32', 'uint32', 'single' }
           itemSz = 4;
       case { 'int64', 'uint64', 'double' }
           itemSz = 8;
       otherwise
           error('unsopported element-type ''%s''', elemType);
   end

   assert(itemSz >= 1 && itemSz <= obj.BINARY_CHUNK_SIZE);

   getChunk = @(offs, len) datArray(offs + 1 : offs + len);

   % make binary-data header
   szStr = sprintf('%lu', numItems * itemSz);
   pref = sprintf('*OPC?;%s#%u%s', pref, length(szStr), szStr);
   % send it (without terminating new-line!):
   fwrite(obj.ViSessn, pref, 'char');
   obj.waitTransferComplete();

   % send the binary-data (in chunks):
   offset = 0;
   chunkLen = fix(obj.BINARY_CHUNK_SIZE / itemSz);
   while offset < numItems,
       if offset + chunkLen > numItems,
           chunkLen = numItems - offset;
       end
       dat = getChunk(offset, chunkLen);
       fwrite(obj.ViSessn, dat, elemType);
       obj.waitTransferComplete();
       offset = offset + chunkLen;
   end
```

```
            % read back the response to that *OPC? query:
            fscanf(obj.ViSessn, '%d');

            if obj.ParanoiaLevel >= 2,
                [errNb, errDesc] = obj.QuerySysErr(1);
                if 0 ~= errNb,
                    warning('System Error #%d (%s) after ' ...
                        + 'sending ''%s ..''.', errNb, errDesc, pref);
                end
            end
        end;

    end % public methods

    methods (Access = private) % private methods

        % ====================================================================
        %> @brief Wait till transfer status is 'idle'.
        % ====================================================================
        function waitTransferComplete(obj)
            while ~strcmp(obj.ViSessn.TransferStatus,'idle')
                pause(obj.WAIT_PAUSE_SEC);
            end
        end;

    end % private methods
end

%%
% ============================================================================
%> @file SE5082_Example.m
%> @brief Usage example (script).
% ============================================================================

inst_addr='TCPIP0::192.168.0.170::5025::SOCKET';
paranoiaLevel = 2; % for debug ..


inst = SE5082(inst_addr, paranoiaLevel);

try
    ok = inst.Connect();
    if ok,
        resp = inst.SendQuery('*IDN?');
        fprintf('Connectred to %s\n', resp);

        % Reset the instrument and clear its internal error-queue:
        inst.SendCmd('*CLS; *RST');

        % Create sine-wave of 1024 points:
        segLen = 1024;

        fprintf(['Downloading sine-wave of %d ', ...
            'points to channels 1&2 ...\n'], segLen);
```

```
        x = linspace(0, 2 * pi, segLen + 1);
        amp = inst.MAX_WAV_VAL / 2.0;
        mid = inst.MAX_WAV_VAL / 2.0;

        x = sin(x(1:segLen)) * amp + mid;
        x = round(x);
        x = min(max(x, 0), inst.MAX_WAV_VAL);
        wav = uint16(x);

        for n = 1:2,
            % Select channel and User-Mode:
            inst.SendCmd(':INST:SEL %d; :FUNC:MODE USER', n);

            % Define segment 1 and select it:
            inst.SendCmd(':TRAC:DEF 1,%d; :TRAC:SEL 1', segLen);

            % Download the binary wave data:
            inst.SendBinaryData(':TRAC:DATA', wav, 'uint16');

            % Turn output on:
            inst.SendCmd(':OUTP ON');
        end

        % Query last instrument internal error:
        resp = inst.SendQuery(':SYST:ERR?');
        fprintf('%s\n', resp);
    end
catch ex
    errStr = getReport(ex);
    warning('\nException:\n%s\n', errStr);
end

clear inst
```

This page was intentionally left blank

Appendices

Appendix A
Specifications

Electrical Specifications

Instrument configuration

Characteristics	Description
SE5082	5GS/s arbitrary microwave generator, with multi-Nyquist RF DAC and memory based architecture with direct DAC output path for time and frequency domain applications
SE5081	Single channel instrument with 32,000,000 waveform points
SE5082	Dual channel instrument with 32,000,000 waveform points
Option -1	64,000,000 waveform points per channel option
Option -2	Reference clock output
Module-HV	High Voltage Amplifier (HV)
Module-DC	High Bandwidth Amplifier (DC)
SE Rack Mount	Rack mounting kit assembly
SE Synchronisation Cable	Synchronization cable to synchronize multiple units

Inter-channel offset control (Coarse tuning – Dual-channel versions only)

Characteristics		Description
Initial skew		<200 ps
Control		
	Range	0 to segment length; 0 to 80 points with external segment control
	Resolution	
	Up to 300 MS/s	16 points
	From 300 MS/s	8 points
Accuracy		Same as sample clock accuracy

Inter-channel skew control (Fine Tuning – Dual-channel versions only)

Characteristics		Description
Initial skew		<200 ps
Control		(Skew is added to offset)
	Range	-3 ns to +3 ns
	Resolution	10 ps
Accuracy		±(10% of setting + 20 ps)

Waveform type

Characteristics	Description
Standard	A waveform is selected from a built in library. The standard waveform parameters are programmable
Arbitrary	Arbitrary waveform coordinates are downloaded and stored in memory segments. The arbitrary waveform parameters are programmable
Sequenced	Arbitrary waveforms are downloaded and stored in memory segments. The segments are arranged in a sequence table that step, loop, jump and nest on segments in a user-defined configuration. Conditional jump and nest pending an event signal
Advanced Sequences	Same functionality as described for sequenced waveforms except sequences are arranged in the sequence table
Modulated	A modulated waveform is calculated from a built in library of modulation schemes
Pulse	A pulse waveform is calculated and downloaded to the arbitrary waveform memory.

Sampling modes

Characteristics	Description
DC operational mode	
NRZ (Non Return to Zero)	The legacy operational mode. This mode gives good results at the beginning of the 1st Nyquist zone
RTZ (Return to Zero)	This mode enable the operation in the 2nd Nyquist zone
NRTZ (Narrow Return to Zero)	Optimized power in the 1st Nyquist zone and beginning of the 2nd Nyquist zone also possible operation in the 4th and 5th Nyquist zones
RF	Optimized for operations over the second half of the 2nd Nyquist zone or over the 3rd Nyquist zone and possible operation in the first half of the 4th Nyquist zone

Run mode

Characteristics	Description
Continuous	A selected output function shape is output continuously
Self armed	No start commands are required to generate waveforms
Armed	The output dwells on a dc level and waits for an enable command and then the output waveform is output continuously; An abort command turns off the waveform
Triggered	A trigger signal activates a single-shot or counted burst of output waveforms and then the instrument waits for the next trigger signal
Normal mode	The first trigger signal activates the output; consecutive triggers are ignored for the duration of the output waveform
Override mode	The first trigger signal activates the output; consecutive triggers restart the output waveform regardless if the current waveform has been completed or not
Gated	A waveform is output when a gate signal is asserted. The waveform is repeated until the gate signal is de-asserted. Last period is always completed

Standard Waveforms

Characteristics	Description
General	Waveforms are computed and generated every time a standard waveform is selected.
Standard waveform library	Built-in, auto computed waveforms: sine, triangle, square, ramp, pulse, sinc, exponential rise, exponential decay, gaussian, noise and dc.
Standard waveform control	The standard waveform parameters can be adjusted to specific requirements. The waveform is re-computed with each parameter change.

Standard waveforms frequency Control

Characteristics		Description
Range		
	HV	10 kHz to 1 GHz
	DC	10 kHz to 2.5 GHz (1st Nyquist – 2.5 GHz instantaneous BW)
	DAC	10 kHz to 2.5 GHz (1st Nyquist – 2.5 GHz instantaneous BW)
Resolution		12 digits
Accuracy		
	Internal reference	≤1 ppm from 19°C to 29°C; 1ppm/°C below 19°C or above 29°C; ≤1 ppm/year aging rate
	External reference	Same as accuracy and stability of the external reference. Reference is applied to the reference input

Arbitrary Waveforms

Characteristics	Description
General	Arbitrary Waveforms are created on a remote computer and downloaded to the arbitrary waveform memory through one of the available remote interfaces. The frequency of the waveform is calculated from its programmed sample clock value and the number of waveform points that were used for creating the waveform. The waveform can be displayed in 4 Nyquist zones, enabling direct IF/RF output without the need for an up convertor.
Waveform length	384 to 32,000,000 points (64,000,000 with memory option), in multiples of 32 points
Number of waveforms	1 to 32,000
Dynamic waveform control	Software command or rear panel segment control input (D-sub, 8 bit lines)
Waveform jump timing	Coherent or asynchronous, selectable
DAC resolution	12 bits

Sequenced Waveforms

Characteristics	Description
General	Segments are grouped in a sequence table that links, loops and jumps to next in user-defined scenarios. Sequence steps are advanced on trigger events or remote commands. Each channel has its own sequence scenario
Sequence scenario	1 to 1,000 unique scenarios, programmed in sequence tables
Sequence table length	3 to 49,152 steps.
Step advance control	Auto, once (x “N”) and stepped
Loop counter	
Segment loops	1 to 16,000,000 cycles, each segment
Sequence loops	1 to 1,000,000 (applies to “Once” sequence advance mode only)

Advanced Sequencing

Characteristics	Description
General	Enables the grouping of sequences into scenarios in a way that is similar to how segments are grouped in a sequence table. Each channel has its own advance sequencing generator
ASequence scenario	1 scenario, programmed in advanced sequence table
Dynamic advance sequence control	Software command or rear panel sequence control input (D-sub, 8 bit lines)
ASequence table length	3 to 1,000 steps
Step advance control	Auto, once and stepped
Once loop counter	1 to 1,000,000 cycles, each sequence

Arbitrary/sequenced waveforms Sample Clock Control

Characteristics	Description
Range	50 MSa/s to 5 GS/s (Typically 6 GS/s), common or separate for each channel
Resolution	12 digits
Accuracy	
Internal reference	≤1 ppm from 19°C to 29°C; 1ppm/°C below 19°C or above 29°C; 1 ppm/year aging rate
External reference	Same as accuracy and stability of the external reference. Reference is applied to the reference input or sample clock input

Analog output - High Voltage Amplifier module

Characteristics	Description
Type of output	Single-ended [1] or differential
Impedance	50 Ω, typical
Amplitude control[2]	Specified into 50 Ω, levels double into high impedance
Amplitude control (into 50 Ω)	
Window,single-ended	-2.25 V to 2.25 V
Window, differential	-4.5 V to 4.5 V
Range, single-ended	50 mVp-p to 2 Vp-p
Range, differential	100 mVp-p to 4 Vp-p
Resolution	4 digits
Accuracy, offset = 0V	±(2% +2 mV)
Offset control	Common mode, specified into 50 Ω, levels double into high impedance
Range	-1.0 V to + 1.0 V
Resolution	4 digits
Accuracy	±(2% +15 mV)
Rise/fall time (20% to 80%)	500 ps, typical
Bandwidth	600 MHz, typical (calculated)
Overshoot	5%, typical
Harmonic distortion [3]	-42 dBc
Non harmonic distortion [3]	-70 dBc, 1 Vp-p, DC to 600 MHz
Phase Noise [4]	-115 dBc/Hz, 1 Vp-p, 10 kHz offset

[1] The unused output must be terminated with 50 Ω to ground
[2] Exceeding the amplitude window is allowed but may cause excessive signal distortion
[3] Amplitude=1 Vp-p , offset=0 V, SCLK=4 GSa/s, 40 points sine waveform (100 MHz output frequency)
[4] Amplitude=1 V, offset=0 V, SCLK=4.5 GS/s, arbitrary 32 points sine waveforms, typical values

Analog output – DC Amplifier module

Characteristics	Description
Type of output	Single-ended [1] or differential
Impedance	50 Ω, typical
Amplitude control[1] (into 50 Ω)	
Window, single-ended	-0.75 V to 0.75 V
Window, differential	-1.5 V to 1.5 V
Range, single-ended	100 mV to 1.2 V
Range, differential	200 mV to 2.4 Vp-p
Resolution	4 digits
Accuracy, offset = 0 V	±(1% +5 mV)
Offset control	Common mode, specified into 50 Ω, levels double into high impedance
Range	-500 mV to + 500 mV
Resolution	4 digits
Accuracy	±(5% +5 mV)
Rise/fall time (20% to 80%)	<100 ps, typical @ 0.6 Vpp - 1.2 Vpp
Calculated Bandwidth (3 dB) for NRZ mode	3 GHz
Overshoot	6%, typical @ 0.6 Vpp - 1.2 Vpp
Harmonics [3] (typical)	
Up to 650 MHz	-60 dBc
650 MHz to 1.6 GHz	-55 dBc
1.6 GHz to 2.5 GHz	-45 dBc
Phase Noise [4]	-120 dBc/Hz, 10 kHz offset
SFDR (typical)	Spurious Free Dynamic Range (NRZ Mode)
Up to 650 MHz	-80 dBc
650 MHz to 1.5 GHz	-70 dBc
1.5 GHz to 2.5 GHz	-58 dBc

[1] The unused output must be terminated with 50 Ω to ground
[2] Exceeding the amplitude window is allowed but may cause excessive signal distortion
[3] 1 Vp-p, Offset=0 V, SCLK=5 GS/s, NRZ mode, sine waveform, typical values measured using balun
[4] Amplitude=1 V, offset=0 V, SCLK=4.5 GS/s, arbitrary 32 points sine waveforms, typical values

Analog output - Direct DAC Output

Characteristics	Description
Type of output	Single-ended [1] or differential
Impedance	50 Ω, typical
Amplitude control (into 50 Ω)	
Range, single-ended	400 mV to 540 mV
Range, differential	800 mV to 1080 mV
Resolution	4 digits
Accuracy, offset = 0 V	±(1% +5 mV)
Calculated Bandwidth (3 dB) for NRZ mode	4 GHz
Harmonics [2] (typical)	
Up to 650 MHz	-65 dBc
650 MHz to 2.5 GHz	-55 dBc
Phase Noise [3]	-120 dBc/Hz, 10 kHz offset
SFDR (typical)	Spurious Free Dynamic Range (NRZ Mode)
Up to 625 MHz	-70 dBc
625 MHz to 1.5 GHz	-60 dBc
1.5 GHz to 2.5 GHz	-60 dBc

[1] The unused output must be terminated with 50 Ω to ground
[2] 1 Vp-p, Offset=0 V, SCLK=5 GS/s, NRZ mode, sine waveform, typical values measured using balun
[3] Amplitude=1 V, offset=0 V, SCLK=4.5 GS/s, arbitrary 32 points sine waveforms, typical values

SYNC output

Characteristics	Description
Connector type	SMA
Type of output	Single ended
Source	Channel 1 or channel 2
Waveform	Pulse (32 points width), WCOM (waveform duration pulse)
Impedance	50 Ω, typical
Amplitude	1.2 V, typical; doubles into high impedance
Variable position control	
Range	0 to segment length
Resolution	32 points
Rise/fall time	2 ns, typical
Variable Width control	
Range	32 points to segment length
Resolution	32 points

Reference clock output (option)

Characteristics	Description
Connector type	Rear panel BNC
Frequency	100 MHz if using internal reference, 10MHz or 100MHz if using external reference
Output impedance	50 Ω, typical
Output voltage	1 Vp-p

Event input

Characteristics	Description
General	Branching in or out from a sequence loop and enabling/disabling output in armed mode
Connector type	Rear panel BNC
Input impedance	10 kΩ, typical
Polarity	Positive, negative or either, selectable
Damage level	± 20 Vdc
Frequency range	0 to 15 MHz
Trigger level control	
Range	-5 V to 5 V
Resolution	12 bit (2.5 mV)
Accuracy	± (5% of setting + 2.5 mV)
Sensitivity	200 mVp-p
Pulse width, minimum	10 ns

Trigger input

Characteristics	Description
Connector type	SMA
Drive	Channel 1, channel 2, or both
Input impedance	10 kΩ, 50Ω selectable
Polarity	Positive, negative, or both, selectable
Damage level	± 20 Vdc
Frequency range	0 to 15 MHz
Trigger level control	
Range	-5 V to 5 V
Resolution	12 bit (2.5 mV)
Accuracy	± (5% of setting + 2.5 mV)
Sensitivity	200 mVp-p
Pulse width, minimum	10 ns
System delay [4)]	200 sample clock periods + 50 ns, typical
Trigger delay	Separate for each channel
Range	0 to 8,000,000 sample clock periods
Resolution	8 points
Accuracy	Same as sample clock accuracy
Smart Trigger	Detects a unique pulse width range
Conditioned trigger	<pulse width, >pulse width, <>pulse width
Pulse width range	50 ns to 2 s
Resolution	2 ns
Accuracy	±(5% of setting +20 ns)
Trigger Holdoff	Ignores triggers for a holdoff duration
Holdoff range	100 ns to 2 s
Resolution	2 ns
Accuracy	±(5% of setting +20 ns)
Trigger jitter[1)]	8 sampling periods
Delay accuracy between channel 1 and channel 2 after a Trigger event [1)]	8 sampling periods

1) Trigger input to analog output.

Internal Trigger Generator

Characteristics	Description
Source	Common or separate for each channel
Mode	Timer (waveform start to waveform start); Delayed (waveform stop to waveform start)
Timer	
Range	200 ns to 2 s
Resolution	3 digits
Accuracy	100 ppm
Delayed	
Range	152 to 8,000,000 sample clock periods
Resolution	Integer numbers, divisible by 8

Sequence / Segment Control input

Characteristics	Description
Connector type	D-sub, 8 bit lines (12 bit TBD)
Number of input connectors	1ch instrument: 8 bit bus + valid line 2ch instrument: (8 bit bus + valid line) per channel
Switching rate	20 ns + waveform duration minimum
Input impedance	10 kΩ, typical
Input level	TTL

External reference clock input

Characteristics	Description
Connector type	Rear panel BNC
Input frequency	10 MHz, 20 MHz, 50 MHz or 100 MHz, programmable
Input impedance	50 Ω, typical
Input voltage swing	-5 dBm to 5 dBm
Damage Level	10 dBm

External sample clock input

Characteristics	Description
General	External signal is fed to a frequency splitter. Same frequency is applied to both channel
Connector type	Rear panel SMA
Input impedance	50 Ω, typical
Input voltage swing	0 dBm to 10 dBm
Input frequency range	50 MHz to 5 GHz
Clock divider	1/1, 1/2, 1/4, … 1/64, separate for each channel
Damage Level	15 dBm

Two Instruments Synchronization

Characteristics	Description
General	Two instruments are synchronized via dedicated synchronization cable. Master instrument controls waveform generation of slave instrument
Initial offset + skew between instruments	20 ns + 0 to 16 SCLK periods
Offset Control Range	0 to waveform length; 0 to 80 points with external segment control
Offset Resolution	8 SCLK periods increments
Skew Control Range	-5 ns to 5 ns (skew is added to offset)
Skew Resolution	10 ps (1ps or 5ps TBD)
Clock Source	Master sample clock generator
Trigger Source	Master trigger input

Mechanical, Environmental and Maintenance Specifications

Display

Characteristics	Description
Type	TFT LCD, back-lit
Size	4 “
Resolution	320 x 240 pixels

Peripheral devices

Characteristics	Description
USB port	1 x front, USB host, standard A; 1 x rear, USB device, standard B
LAN port	1000/100/10 BASE-T
GPIB port	IEEE 488.2 standard interface, 24 pin
Segment control port	2 x D-sub, 9 pin

Power supply

Characteristics	Description
Source voltage and frequency	
Rating range	100 VAC to 240 VAC
Frequency range	50 Hz to 60 Hz
Power consumption	100 VA

Mechanical

Characteristics	Description
Dimensions	
With feet	315 x 102 x 395 mm (W x H x D)
Without feet	315 x 88 x 395 mm (W x H x D)
Weight	
Without package	4.5 kg
Shipping weight	6 kg

Environmental

Characteristics	Description
Operating temperature	0°C to 40°C
Storage temperature	-40°C to 70°C
Humidity	85% RH, non condensing

Certifications and compliances

Characteristics	Description
Safety	IEC61010-1:2010
EMC	IEC 61326-1:2013

Maintenance

Characteristics	Description
General	Periodical recalibration is required to maintain accuracy of output characteristics
Recalibration Period	2 years