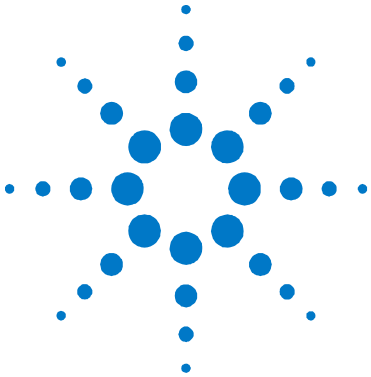


Agilent Arbitrary Waveform Generator

M8190A-B02

User's Guide

Agilent Technologies

Notices

Manual Part Number

M8190-91010

Edition

Fourth edition, November 2011

Agilent Technologies, Deutschland GmbH

Herrenberger Str. 130

71034 Böblingen, Germany

For Assistance and Support

http://www.agilent.com/find/assist

Limitation of Warranty

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance. No other warranty is expressed or implied. Agilent Technologies specifically disclaims the implied warranties of Merchantability and Fitness for a Particular Purpose.

Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Safety Notices

CAUTION

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a **CAUTION** notice until the indicated conditions are fully understood and met.

WARNING

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

Safety Summary

General Safety Precautions

The following general safety precautions must be observed during all phases of operation of this instrument. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the instrument.

For safe operation the general safety precautions for the M9502A and M9505A AXIe chassis, must be followed. See: http://www.agilent.com/find/M9505A

Agilent Technologies Inc. assumes no liability for the customer's failure to comply with these requirements.

Before operation, review the instrument and manual for safety markings and instructions. You must follow these to ensure safe operation and to maintain the instrument in safe condition.

Initial Inspection

Inspect the shipping container for damage. If there is damage to the container or cushioning, keep them until you have checked the contents of the shipment for completeness and verified the instrument both mechanically and electrically. The Performance Tests give procedures for checking the operation of the instrument. If the contents are incomplete, mechanical damage or defect is apparent, or if an instrument does not pass the operator's checks, notify the nearest Agilent Technologies Sales/Service Office.

WARNING To avoid hazardous electrical shock, do not perform electrical tests when there are signs of shipping damage to any portion of the outer enclosure (covers, panels, etc.).

General

This product is a Safety Class 3 instrument. The protective features of this product may be impaired if it is used in a manner not specified in the operation instructions.

Environment Conditions

This instrument is intended for indoor use in an installation category II, pollution degree 2 environment. It is designed to operate within a temperature range of 5 – 40 °C (40 – 105 °F) at a maximum relative humidity of 80% and at altitudes of up to 2000 meters.

This module can be stored or shipped at temperatures between -40°C and +70°C. Protect the module from temperature extremes that may cause condensation within it.

Before Applying Power

Verify that all safety precautions are taken including those defined for the mainframe.

Line Power Requirements

The Agilent M8190A operates when installed in an Agilent AXIe mainframe.

Do Not Operate in an Explosive Atmosphere

Do not operate the instrument in the presence of flammable gases or fumes.

Do Not Remove the Instrument Cover

Operating personnel must not remove instrument covers. Component replacement and internal adjustments must be made only by qualified personnel.

Instruments that appear damaged or defective should be made inoperative and secured against unintended operation until they can be repaired by qualified service personnel.

Symbols on Instruments

Indicates warning or caution. If you see this symbol on a product, you must refer to the manuals for specific Warning or Caution information to avoid personal injury or damage to the product.

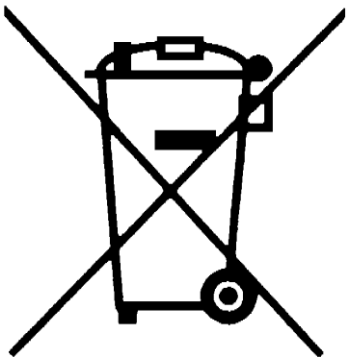
CE Marking to state compliance within the European Community: This product is in conformity with the relevant European Directives.

General Recycling Mark

N10149

C-Tick Conformity Mark of the Australian ACA for EMC compliance.

Environmental Information



This product complies with the WEEE Directive (2002/96/EC) marketing requirements. The affixed label indicates that you must not discard this electrical/electronic product in domestic household waste.

Product category: With reference to the equipment types in the WEEE Directive Annexure I, this product is classed as a "Monitoring and Control instrumentation" product.

Do not dispose in domestic household waste.

To return unwanted products, contact your local Agilent office, or see **www.agilent.com/environment/product/** *for more information.*

Contents

Contents 5

1 Introduction 9

1.1 The Front Panel 11

1.2 M8190A Software Installation 12

1.2.1 Pre-Requisites 12

1.2.2 Installation Process 13

1.2.3 Post Installation Steps 19

1.2.4 How to use the Instrument 20

2 M8190A User Interface 21

2.1 Introduction 21

2.2 Launching the M8190A User Interface 21

2.3 M8190A User Interface Overview 24

2.3.1 Title Bar 24

2.3.2 Menu Bar 24

2.3.3 Status Bar 27

2.3.4 Clock / Output Tabs 27

2.3.5 Control and Status Area 27

2.4 Driver Call Log 28

2.5 Report Error Window 29

2.6 Clock Tab 30

2.7 Output Tab 31

3 General Programming 33

3.1 Introduction 33

3.2 IVI-COM Programming 34

3.3 SCPI Programming 35

3.4 Programming Recommendations 35

3.5 Arbitary Waveform Commands (TRACe Subsystem) 37

3.5.1 TRACe[1|2]:DEFine <segment-id >,<length>[,<DAC-Value>]37

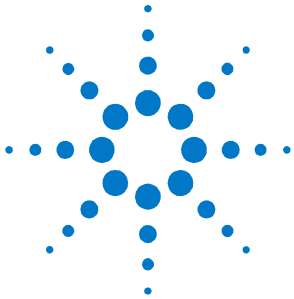
3.5.2 TRACe[1|2][:DATA] <segment-id>,<offset>,(<block>|<numeric-values>) 38

3.5.3 TRACe[1|2]:DELete <segment-id> 39
3.5.4 TRACe[1|2]:DELete:ALL 39
3.5.5 TRACe[1|2]:CATalog? 39
3.5.6 TRACe[1|2]:FREE? 39
3.6 Level Commands ([SOURce]:VOLTage Subsystem) 39
3.6.1 [:SOURce]:VOLTage[1|2] [:LEVel][:IMMediate][:AMPLitude][?]40
3.6.2 [:SOURce]:VOLTage[1|2] [:LEVel][:IMMediate]:OFFSet[?]..... 40
3.7 Sampling Frequency Commands 40
3.7.1 [SOURce:]FREQuency:RASTer[?] [frequency] 40
3.7.2 [SOURce:]FREQuency:RASTer:EXTernal[?] [frequency] 40
3.7.3 [SOURce:]FREQuency:RASTer:SOURce[1|2] [?] [INTernal|EXTernal] 41
3.8 Output Commands (OUTPut Subsystem) 41
3.8.1 OUTPut[1|2][:NORMal][:STATe][?] [0|1|ON|OFF] 41
3.9 Status reporting Commands (STATus Subsystem) 41
3.9.1 STATus:QUEStionable 41
3.9.2 STATus:OPERation 43
3.10 System Related Commands (SYSTem Subsystem) 45
3.10.1 SYSTem:ERRor[:NEXT]? 45
3.10.2 SYSTem:HELP:HEADers? 45
3.11 Triggering Commands 46
3.11.1 ABORt 46
3.11.2 ARM[:SEQuence][:STARt][:LAYer]:DELay[1|2][?] <delay>..... 46
3.11.3 INITiate:IMMediate 46
3.12 Common Command List 46
3.12.1 *CLS 46
3.12.2 *ESE 47
3.12.3 *ESR? 47
3.12.4 *OPC 47
3.12.5 *OPC? 47
3.12.6 *OPT? 48
3.12.7 *RST 48
3.12.8 *SRE[?] 48
3.12.9 *STB ? 48
3.12.10 *TST? 48

3.13 Status Model 49
3.13.1 Status Byte Register 51
3.13.2 Status Commands 51
3.13.3 Status Questionable Data Register Command Subsystem 52
3.13.4 Status Operation Status Subsystem 54
3.14 Example Programs 56
3.14.1 C++ Example 56
3.14.2 C# Example 58

4 Characteristics 61

4.1 Performance Specification 61
4.1.1 DIRECT OUT 1 / DIRECT OUT 2 61
4.1.2 Sample Clock 62
4.1.3 Internal Synthesizer Clock Characteristics 62
4.1.4 Sample Clock Input 63
4.1.5 Sample Clock Output 63
4.1.6 General 64
4.1.7 Definitions 65
4.2 Sequencing 66



1 Introduction

Introduction

The Agilent M8190A ensures accuracy and repeatability with 14-bit resolution, up to 8 GSa/s sampling rate and up to 80 dBc SFDR. High dynamic range and excellent vertical resolution gives you confidence that you are testing your device, not the signal source.

As an example, a test setup that exhibits a high error vector magnitude (EVM) reading might prevent you from seeing problems within your device under test (DUT). The level of reality possible with the M8190A minimizes problems like this.

Features and Benefits

- Precision AWG with 14-bit resolution up to 8 GSa/s
- Spurious-free-dynamic range (SFDR) up to 80 dBc typical
- Harmonic distortion (HD) up to -72 dBc typical
- 1610 MSa arbitrary waveform memory per channel
- Analog bandwidth 3.5 GHz
- Transition times 75 ps (20 % to 80 %)
- Differential output
- Form-factor: 2-slot AXIe module, controlled via external PC or embedded AXIe system controller
- Supported operating system: Windows 7 / 64 bit

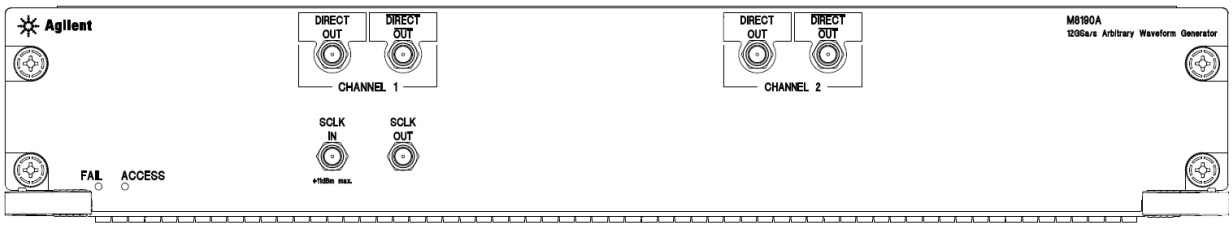
Additional Documents

Additional documentation can be found at:

- http://www.agilent.com/find/M9505A for 5-slot chassis related documentation.
- http://www.agilent.com/find/M9502A for 2-slot chassis related documentation.
- http://www.agilent.com/find/M9045A for PCIe laptop adapter card related documentation.
- http://www.agilent.com/find/M9047A for PCIe desktop adapter card related documentation.
- http://www.agilent.com/find/M9536A for embedded AXIe controller related documentation.
- http://www.agilent.com/find/M8190A for AXIe based AWG module related documentation.

1.1 The Front Panel

The Front Panel of M8190A is shown in the figure below.



Front Panel of M8190A

Inputs/Outputs

The inputs and outputs of the instrument are available at the front panel:

- The analog output signal of channel 1 and channel 2 (**DIRECT OUT 1 & DIRECT OUT 2**) are differential outputs of the two Digital to Analog Converter **(DAC).** The Outputs can be switched ON or OFF with an internal relays.
- The Sample Clock Input (**SCLK IN**) can be used, if an external clocjk source shall be used to clock the Digital to Analog Converters.
- The Sample Clock Output (**SCLK OUT**) outputs the clock signal that is used to internally clock both DAC.

Status LED

Two LED are available at the front panel to indicate the status of the AWG module:

- The green **'Access'** LED indicates that the controlling PC exchanges data with the AWG module.
- The red **'Fail'** LED has following functionality:
  - It is 'ON' for about 30 seconds after powering the AXIe chassis.
  - After about 30 seconds the LED is switched 'OFF'. If an external PC is used to control the AXIe chassis, this PC can be powered after this LED has switched OFF.
  - During normal operation of the module this LED is 'OFF'. In case of an error condition such as e.g. a self test error, the LED is switch 'ON'.

1.2 M8190A Software Installation

This section will show you the steps required to install M8190A software package.

1.2.1 Pre-Requisites

The following are the pre-requisites for installing Agilent M8190A software:

- The supported Operating System are:
  - Windows XP SP3 (32 bit)
  - Windows Vista (32 bit)
  - Windows Vista (64 bit)
  - Windows 7 (32 bit)
  - Windows 7 (64 bit)

- Ensure that you have Agilent IO Libraries Suite Version 16.1 or higher installed on your system. The Agilent IO Libraries Suite can be found on the CD that is part of shipment content or at http://www.agilent.com/find/iosuite.

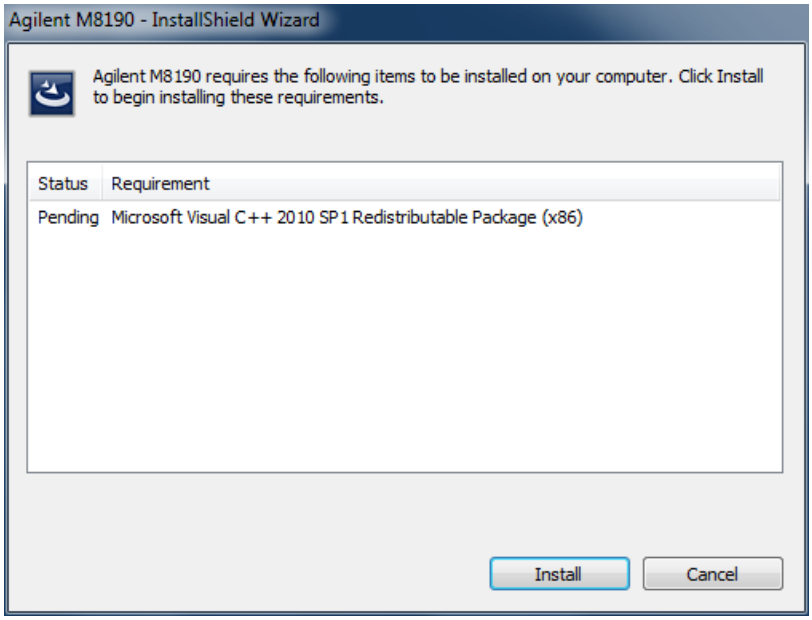
  Note: even if already a non-Agilent I/O library is installed, it is still necessary to install the Agilent I/O library. Agilent I/O library will install as "secondary" I/O library in this case. This use case is fully supported. There only needs to be some additional considerations when doing SCPI remote programming. This is explained in chapter 3.3 SCPI Programming.

1.2.2 Installation Process

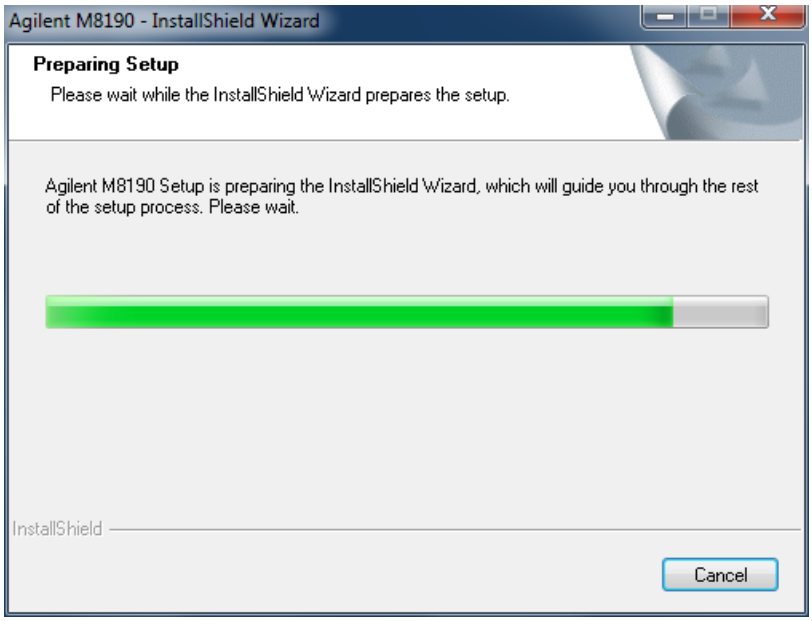
Follow the given steps to install **Agilent M8190A** software on your system.

1. Double-click the executable (M8190_Setup.exe). This executable file will be available either on CD, USB drive or Web.

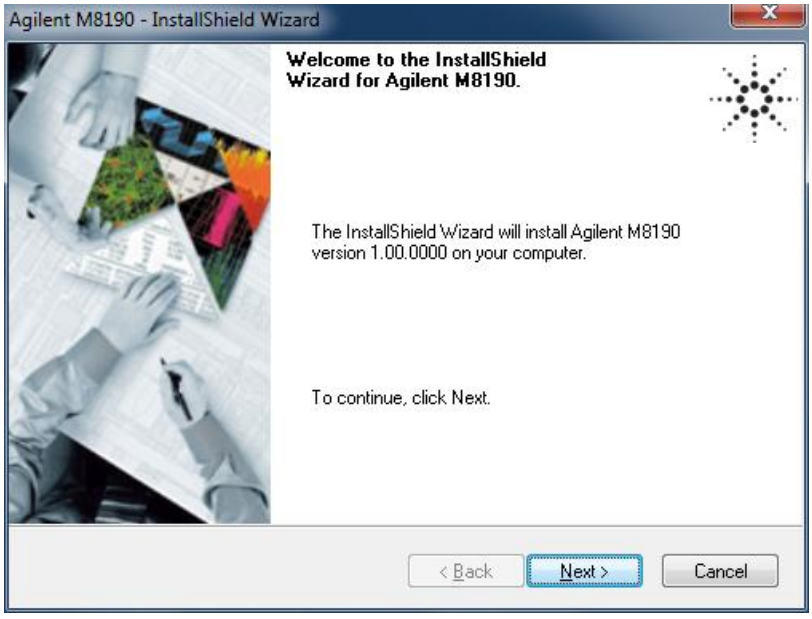
Note: The installer will first check for some pre-requisites and install them, if necessary. It is possible that your PC requests a reboot during this step. Reboot your PC, if requested.



2. The Agilent **M8190A Setup** will prepare the **InstallShield Wizard** for the installation process. The following window will appear.

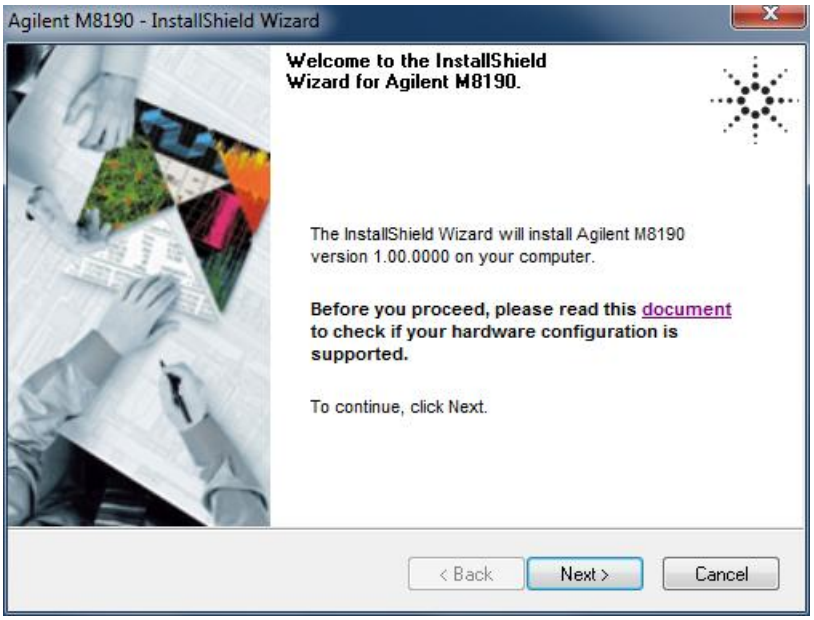


3. Follow the onscreen instructions to begin the installation process.

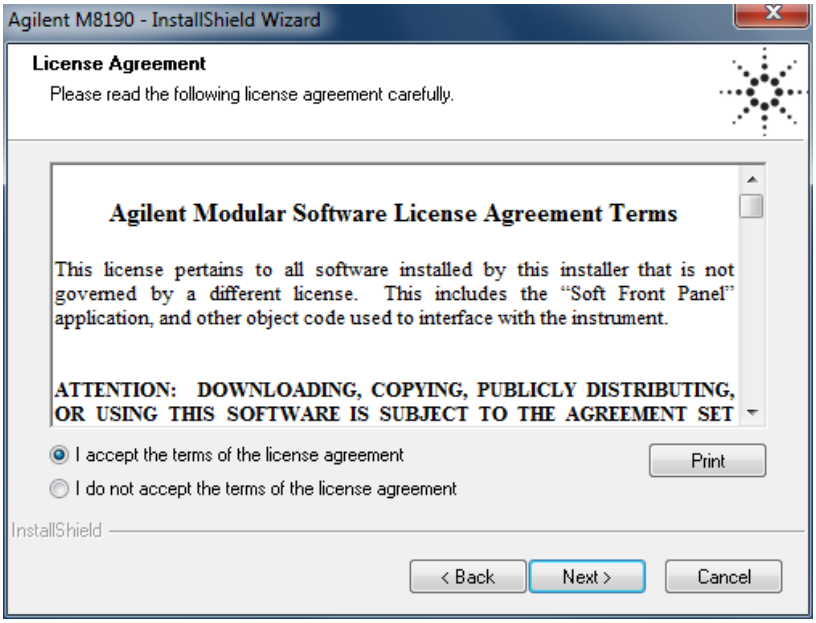


4. We recommend you to read the document to check if your hardware configuration is supported.

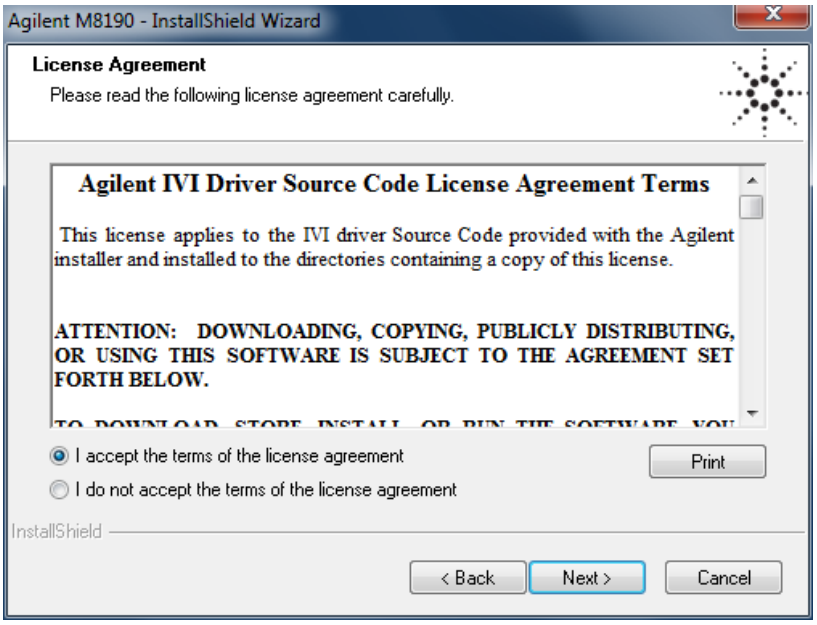
   Click **Next** to begin the installation process.



5. Accept the terms of Agilent Modular Software license agreement. Click **Next** to begin the installation process.

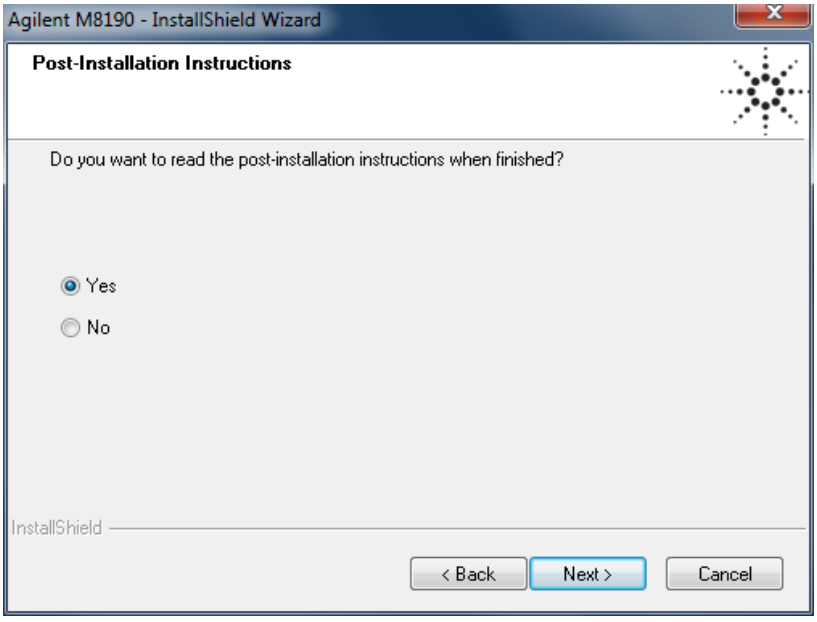


6. Accept the terms of Agilent IVI Driver Source Code license agreement. Click **Next** to begin the installation process.

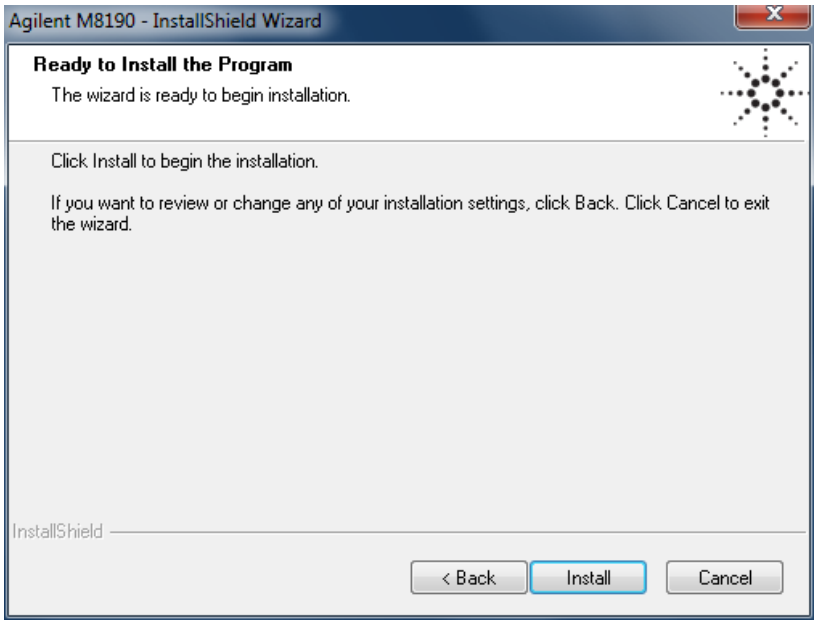


7. Select **Yes** if you want to read the post-installation instructions when finished.

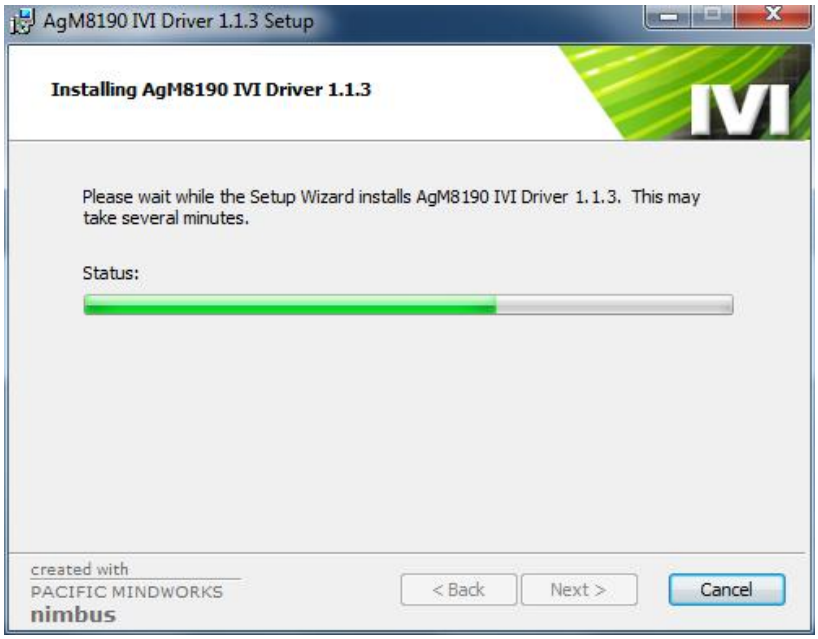
   Click **Next** to begin the installation process.



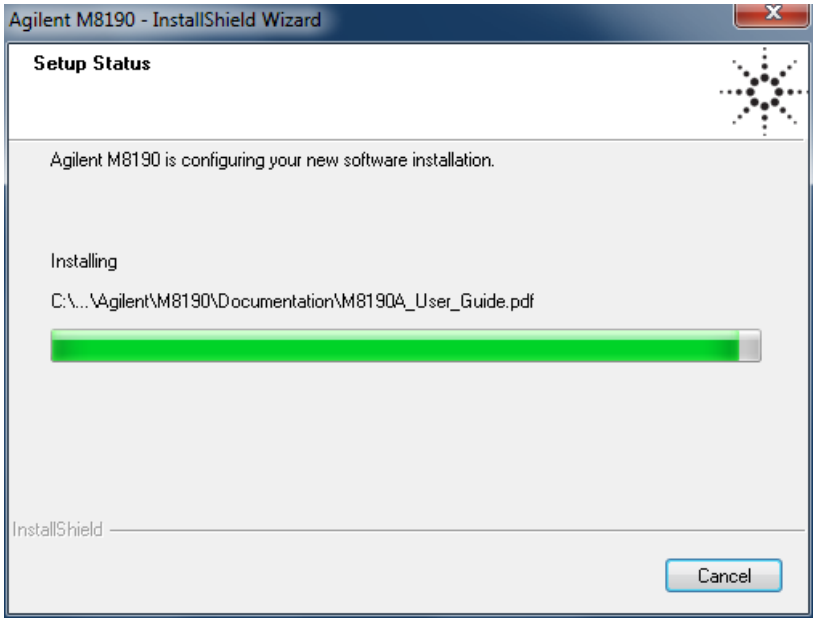
8. Follow the onscreen instructions to begin the installation process.



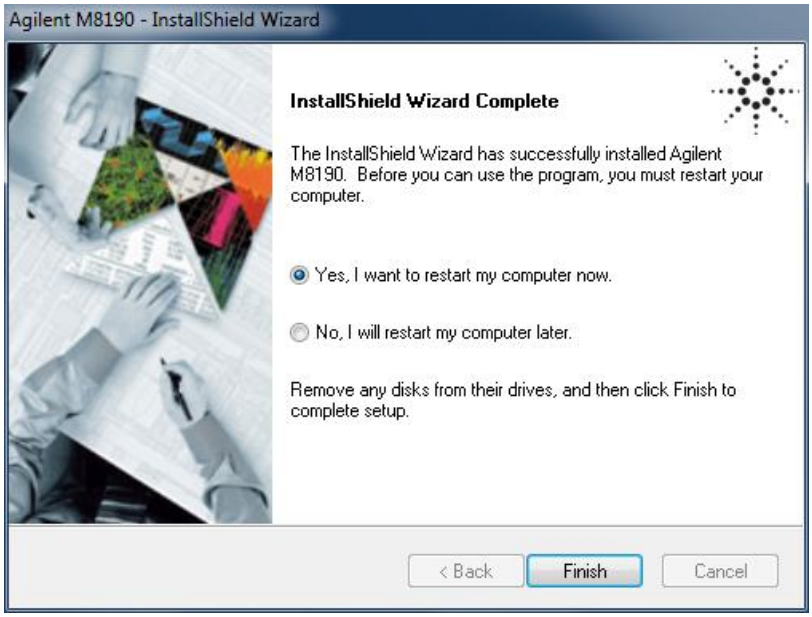
9. The **Setup Wizard** will now install AgM8190 IVI Drivers.



The following screen will appear that shows the M8190A installation progress.



10. The following screen will appear once the **Agilent M8190A** software is successfully installed on your system. Click **Finish** to restart your system. Do not connect the AXIe chassis to your system using the PCIe cable during this reboot.



This completes the Agilent M8190A software installation.

1.2.3 Post Installation Steps

Follow the post installation steps as shown below:

1. Shut down PC and instrument.
2. Connect the instrument to the PC using the PCIe cable.
3. Switch on the instrument. Wait until the 'Access' LED of the M8190A has switched from red to green.
4. Switch on the PC.
5. The PC should automatically recognize the instrument.
   Check this in the device manager; e.g. via *Start – Control Panel – Device Manager*, or *Computer – Manage – Device Manager*:
   The instrument should be visible as *Agilent – M8190*.
   Note: Your PC might request a reboot. Reboot your PC, if requested.
6. Check if the M8190 is also visible in the Agilent Connection Expert (e.g. via *Start – Agilent IO Libraries Suite – Agilent Connection Expert*, or using the small blue IO symbol in the task bar).
   If the Instrument is not shown in the PXI section you can either (a) reboot or (b) restart the "Agilent IO Libraries Service" (e.g. via *Administrative Tools – Services* or *Computer – Manage – Services* and *Applications – Services*).
7. Start the firmware (*Start – Agilent – M8190 – M8190*).
8. Install Intel Network Drivers on Windows XP: The AXIe chassis contains an Intel 82573L NIC as a PCIe endpoint. Refer to the AXIe chassis User's Guide at http://www.agilent.com/find/M9505A for instructions how to install this driver.
9. Add the M8190A as a LAN instrument using HiSLIP or Socket protocol in the Agilent Connection Expert. Note: VXI-11 is currently not supported.

1.2.4 How to use the Instrument

In order to use the instrument:

1. It must always be connected and switched on before the PC is started.
2. The firmware must be started (*Start – Agilent – M8190 – M8190*).
3. You can then connect to the firmware to control the instrument.

2 M8190A User Interface

2.1 Introduction

This chapter describes the different tools and their features provided by the M8190A graphical user interface.

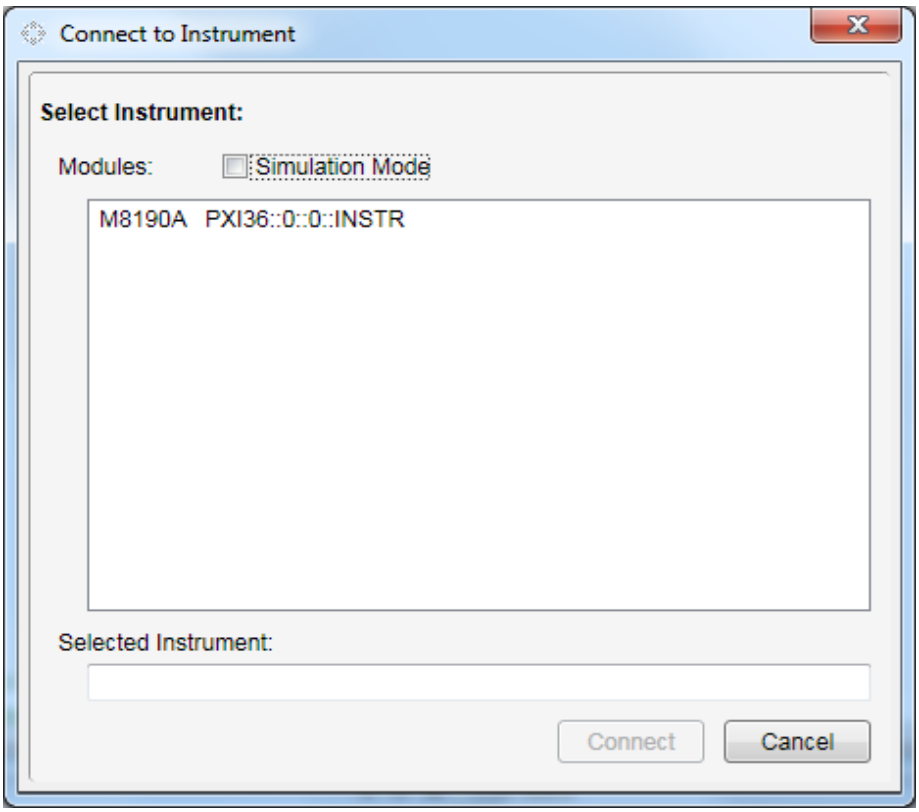
2.2 Launching the M8190A User Interface

There are two ways to launch the M8190A user interface:

1. Select Start – All Programs – Agilent – M8190 – Soft Front Panel from the Start Menu.

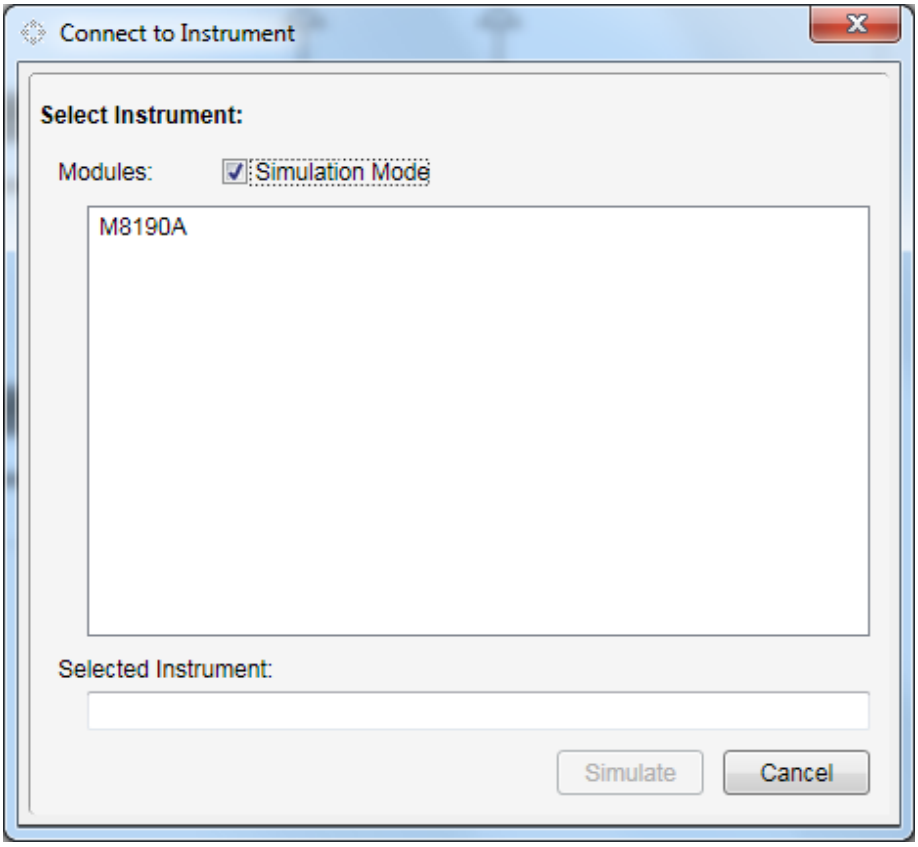
2. From the Agilent Connection Expert select the discovered M8190 module, press the right mouse key to open the context menu and select “Send Commands To This Instrument”.

The following screen will appear:



The instrument selection dialog shows the addresses of the discovered M8190A modules. Select a module from the list and press “Connect”.

If no M8190A module is connected to your PC, you can check “Simulation Mode” to simulate an M8190A module.



2.3 M8190A User Interface Overview

The M8190A user interface includes the following GUI items:

- Title Bar
- Menu Bar
- Status Bar
- Clock / Output Tabs
- Control and Status Area

The detailed information on these GUI items is given in the sections that follow.

2.3.1 Title Bar

The title bar contains the standard Microsoft Windows elements such as the window title and the icons for minimizing, maximizing, or closing the window.

2.3.2 Menu Bar

The menu bar consists of various pull down menus that provide access to the different functions and launch interactive GUI tools.

The menu bar includes the following pull down menu:

- File
- View
- Utility
- Tools
- Help

Each pull down menu and its options are described in the following sections.

2.3.2.1 File Menu

The **File** menu includes the following selections:

- **File** – **Connect...**
  Opens the instrument selection dialog.

- **File** – **Exit**
  Exits the user interface.

2.3.2.2 View Menu

The **View** menu includes the following selections:

- **View** – **Refresh**
  Reads the instrument state and updates all fields.

2.3.2.3 Utility Menu

The **Utility** menu includes the following selections:

- **Utility** – **Reset**
  Resets the instrument, reads the state and updates all fields.

- **Utility** – **Errors**
  Opens a window to display the errors reported by the instrument.

- **Utility** – **Self Test**
  Opens a window to start the self test and display the result after completion.

2.3.2.4 Tools Menu

The **Tools** menu includes the following selections:

- **Tools** – **Monitor Driver Calls...**
  Opens the Driver Call Log.

2.3.2.5 Help Menu

The **Help** menu includes the following selections:

- **Help** – **Driver Help...**
  Opens the IVI driver online help.

- **Help** – **Online Support...**
  Opens the instrument's product support web page.

- **Help** – **About...**
  Displays revision information for hardware, software and firmware.
  Displays the serial number of the connected module.

2.3.3 Status Bar

The status bar contains three fields from left to right:

- **Connection state**
  "Not Connected" – No instrument is connected.
  "Connected: <Instrument resource string>" – An instrument is connected. The resource string, for example PXI36::0::0::INSTR is displayed.
  "Simulation Mode" – No real instrument is connected. The user interface is in simulation mode.
  Click this field to open the Instrument Selection Dialog.

- **Instrument status**
  Displays the instrument status, for example "Reset complete" after issuing a reset command. In case of error it displays additional error information.

- **Error status**
  "Error" – The connected instrument reported an error.
  "No Error" – No errors occurred.
  Click this field to open the Report Error Window.

2.3.4 Clock / Output Tabs

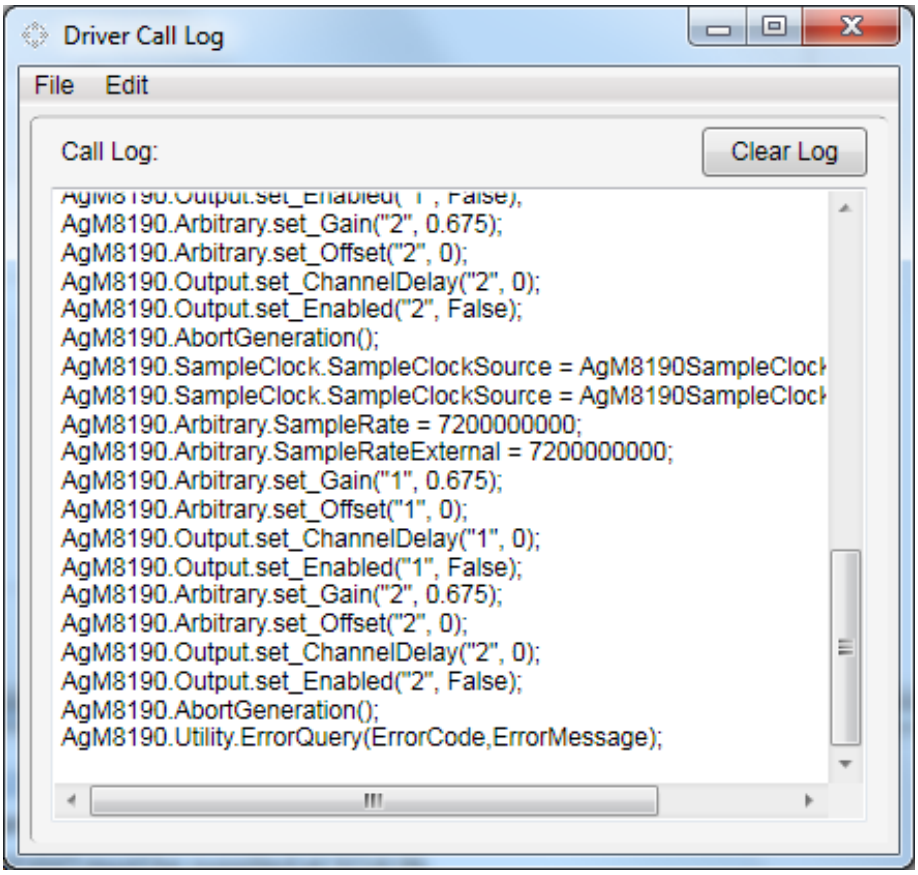
These tabs are used to configure all parameters of the M8190A module. They are described in detail in the following sections.

2.3.5 Control and Status Area

This area provides a **Run/Stop** button to start and stop signal generation.

2.4 Driver Call Log

Use this window to inspect the sequence of IVI driver calls used to configure the M8190A module.



It has the following GUI items:

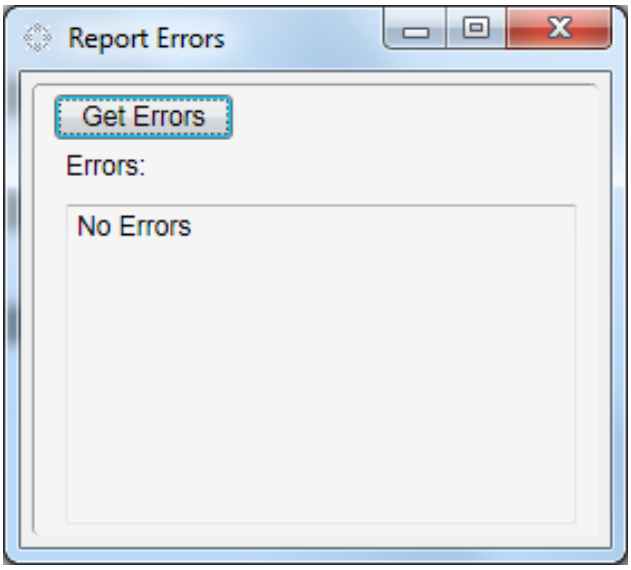
- **The File** menu has the following item:
  File – **Save As**
  Saves the Driver Call Log as a text file.

- The **Edit menu** has the following items:
  Edit – **Select All**
  Selects the contents of the Driver Call Log.
  Edit – **Copy**
  Copies the selected contents of the Driver Call Log to the clipboard.

- **Clear Log**
  Clears the Driver Call Log.

2.5 Report Error Window

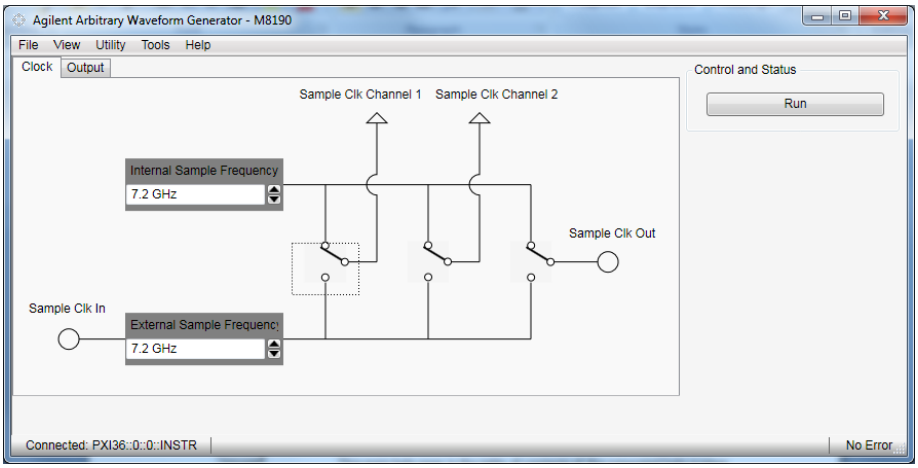
Use this window to read out and display the error queue of the M8190A module.



- **Get Errors**
  Queries the instruments error queue and displays the errors, if any.

2.6 Clock Tab

Use this tab to configure the sample clock of the M8190A module. It contains switches for internal/external sample clock selection and input fields to configure the relevant frequencies.



- Sample clock selection switch for channel 1 and 2

  These switches select between internal and external sample clock for channel 1 and 2. This instrument revision does not support different clock sources for the two channels. The channels either use both internal or both external clock.

- Sample clock output configuration switch

  This switch shows which sample clock is present at the sample clock output. This instrument revision always outputs the sample clock – internal or external, which is selected for the two channels.

- Internal sample frequency

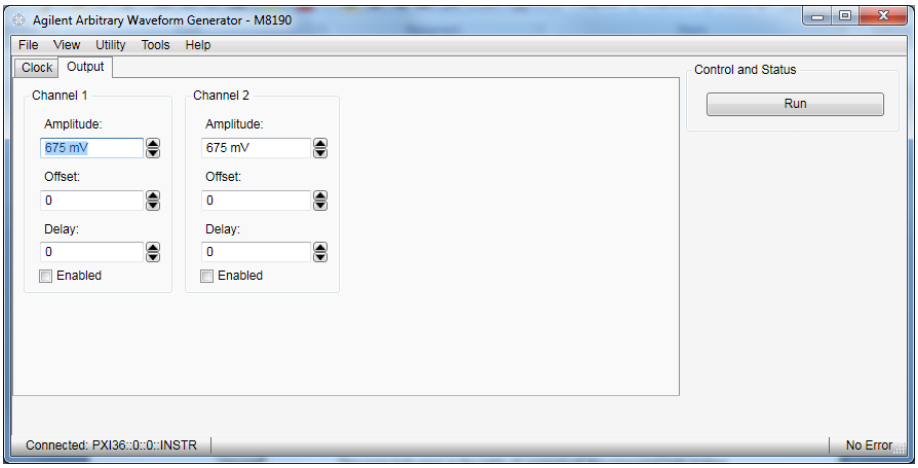
  If internal sample clock is selected, this field specifies the frequency of the internally generated sample clock.

- External sample frequency

  If external sample clock is selected, this field specifies the frequency of the used external clock source.

2.7 Output Tab

Use this tab to configure the outputs of the M8190A module.



The two channels have the following input fields.

- Amplitude

  Specifies the amplitude of the output signal.
- Offset

  Specifies the offset of the output signal.
- Delay

  Specifies data out delay relative to the sample clock output.
- Enabled

  If checked, the generated signal is present at the output.

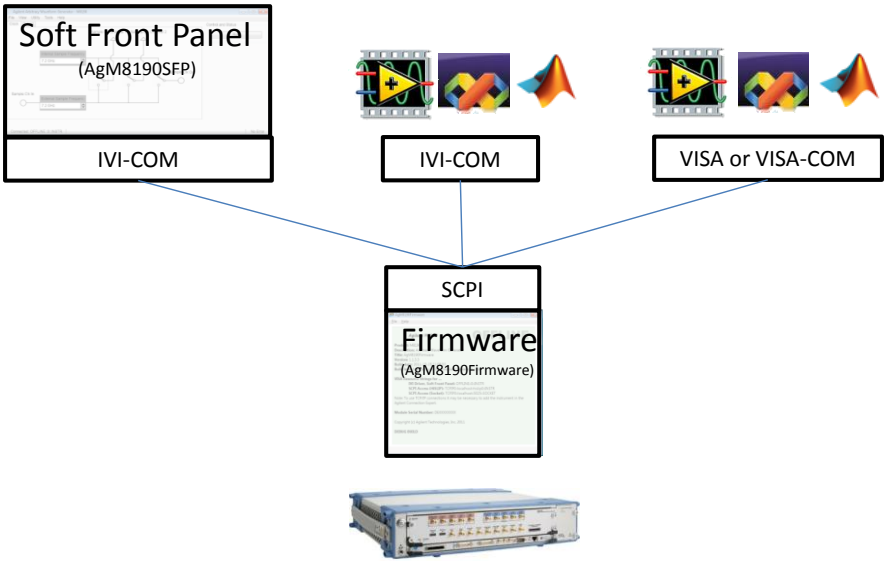
3 General Programming

3.1 Introduction

Introduction

M8190A can be programmed like other modular instruments using IVI-COM driver. In addition, classic instrument programming using SCPI commands is supported.

The following picture gives an overview about how things work together:



Firmware talks to the actual M8190A module using a PCI express connection. I/O to the module is done using VISA library of Agilent I/O library. Addressing is done with PXI resource strings, e.g. "PCI36::0::0::INSTR". The purpose of the firmware is to provide a classic instrument like SCPI interface that is exposed via LAN.

IVI-COM wraps the SCPI commands into API based programming model. To select what Module is programmed, the PXI resource string of the module is used. The IVI-driver will automatically locate an already running Firmware that is handling the module. If no such firmware exists, it is started automatically. This way it is completely hidden that the IVI driver actually needs the Firmware for programming the M8190A module.

VISA or VISA-COM are libraries from an installed I/O library such as the Agilent I/O library to program the Firmware "instrument" using SCPI command strings. Firmware must be already running to connect to it.

Soft Front Panel is providing the user interface. It is used for interactively changing settings. In addition it can log what IVI calls need to be done when changing a setting. This can be activated with **Tools > Monitor Driver calls...**. Also you can verify changes done from a remote program with **View > Refresh**.

3.2 IVI-COM Programming

The recommended way to program the M8190A module is to use the IVI drivers. See documentation of the IVI drivers how to program using IVI drivers. The connection between the IVI-COM driver and the Ag8190Firmware is hidden. To address a module therefore the PXI resource string of the module, such as "PCI36::0::0::INSTR" is used. The IVI driver will connect to an already running Ag8190Firmware. If Ag8190Firmware is not running, it will automatically start it.

3.3 SCPI Programming

Introduction

In addition to IVI programming SCPI programming using a LAN connection is also supported. Two LAN protocols are supported:

- **HiSLIP:** this protocol is recommended. It offers the functionality of VXI-11 protocol with better performance that is near socket performance. Visa Resource String looks like "TCPIP0::localhost::hislip0::INSTR". The correct resource string is shown in the Ag8190Firmware main window. To use HiSlip protocol an I/O library such as the Agilent I/O library. Since the protocol is new it might not be supported by the installed I/O library. Agilent I/O library 16.1 and above supports it. However Agilent I/O library might be installed as secondary I/O library. In this case check if the primary I/O library supports HiSLIP. If it does not, the socket protocol must be used.
- **Socket:** this protocol can be used with any I/O library or using standard operating system socket functionality connecting to port 5025. This protocol must be used if the used I/O library is not supporting HiSLIP protocol. Visa Resource string looks like "TCPIP0::localhost::5025::SOCKET", the exact resource string can be seen in the Ag8190Firmware main window.

VXI-11 protocol is currently not supported. It will be supported in a future release.

3.4 Programming Recommendations

This section lists some recommendations for programming the instrument.

- Start programming from the default setting. The common command for setting the default setting is:
  `*RST`
- Use the binary data format when transferring waveform data.
- The SCPI standard defines a long and a short form of the commands. For fast programming speed it is recommended to use the short forms. The short forms of the commands are represented by upper case letters. For example the short form of the command to set 10mV offset is:
  `:VOLT:OFFS 0.01`

- To improve programming speed it is also allowed to skip optional subsystem command parts. Optional subsystem command parts are depicted in square brackets, e.g.: Set amplitude
  `[:SOURce]:VOLTage[1|2]`
  `[:LEVel][:IMMediate][:AMPLitude]`
  Sufficient to use:
  `:VOLT`
- M81190A is a real 2 channel instrument. Parameters have to be specified for output 1 and output 2. If there is no output specified the command will set the default output 1. So, for setting a offset of 10mV for output 1 and output 2 the commands are:
  `:VOLT:OFFS 0.01` # sets offset of 10mV at output 1
  `:VOLT1:OFFS 0.01` # sets offset of 10mV at output 1
  `:VOLT2:OFFS 0.01` # sets offset of 10mV at output 2
- If it is important to know whether the last command is completed then send the common query:
  `*OPC?`
- It is recommended to test the new setting which will be programmed on the instrument by setting it up manually. When you have found the correct setting, then use this to create the program.
- In the program it is recommended to send the command for starting data generation (`:INIT:IMM`) as the last command. This way intermediate stop/restarts (e.g. when changing sample rate or loading a waveform) are avoided and optimum execution performance is achieved.

```
*RST            # set default settings
...             # other commands to set modes
...             # and parameters
:OUTP1 ON       # enable the output 1
:INIT:IMM       # start data generation.
```

3.5 Arbitary Waveform Commands (TRACe Subsystem)

Download and administrate waveforms in module memory (as “segments”). First a segment must be defined (:TRAC[1|2]:DEF). Then it can be filled with DAC values (:TRAC[1|2]). After starting signal generation with :INIT:IMM the last defined segment is generated.
DAC values have 14 bit resolution and are signed. Valid range is -8191 to +8191. -8191 represents the Low Level and +8191 represents the High Level. 0 represents the Offset Voltage. They are stored in 16bit signed integers using the most significant bits. This is achieved by shifting the 14 bit values 2 bits to the left. The lower 2 bits are reserved for future use and must be set to 0:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB13	DB12	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	0	0

3.5.1 TRACe[1|2]:DEFine <segment-id >,<length>[,<DAC-Value>]

< segment-id > 1..256K

<length> Number of samples. Granularity is 48, minimum value 192, maximum value is 2147483616.

<DAC-Value> A DAC value in the above described 16 bit format.

Defines a segment with length <length> If <DAC-Value> is specified, all values are initialized to <DAC-Value>. If <DAC-Value> is not specified, memory is only allocated but not initialized. If segment already exists an error is generated.

3.5.2 TRACe[1|2][:DATA] <segment-id>,<offset>,(<block>|<numeric-values>)

< segment-id > 1..256K

<offset> Offset in samples. This allows splitting the transfer in smaller sections. Offset must fulfill the granularity requirement (multiple of 48). Also this is necessary to overcome the SCPI restriction that only allows transferring up to 999999999Bytes at once.

<block> Binary data in IEEE-488.2 binary block format with samples in the above described 16 bit format (byte order: little endian). Granularity must be obeyed, so the number of samples transmitted must be multiple of 48. This means that a multiple of 96 bytes must be transmitted.

<numeric-values> Comma separated values in above described 16 bit format, each representing one sample.

If segment does not exist, an error is generated. If length of segment is exceeded error -223 (too much data) is reported.

Illustration of how DAC bits are encoded, and the multiple of 48 sample requirements:

Waveform Data Word	Definition	Block 1 (48 Samples)				Block 2 (48 Samples)				...				Block n (48 Samples)			
D15	DAC Bit 13	Block 1, Sample 1	Block 1, Sample 2	...	Block 1, Sample 48	Block 2, Sample 1	Block 2, Sample 2	...	Block 2, Sample 48	...	...	...	...	Block n, Sample 1	Block n, Sample 2	...	Block n, Sample 48
D14	DAC Bit 12																
D13	DAC Bit 11																
D12	DAC Bit 10																
D11	DAC Bit 9																
D10	DAC Bit 8																
D9	DAC Bit 7																
D8	DAC Bit 6																
D7	DAC Bit 5																
D6	DAC Bit 4																
D5	DAC Bit 3																
D4	DAC Bit 2																
D3	DAC Bit 1																
D2	DAC Bit 0																
D1	reserved	0	0	...	0	0	0	...	0	0	0	...	0	0	0	...	0
D0	reserved	0	0	...	0	0	0	...	0	0	0	...	0	0	0	...	0

Example trac:data 1,0,#29612345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678

3.5.3 TRACe[1|2]:DELete <segment-id>

< segment-id > 1..256K

Delete a segment.

3.5.4 TRACe[1|2]:DELete:ALL

Delete all segments

3.5.5 TRACe[1|2]:CATalog?

The query returns a comma-separated list of segment-ids that are defined + the length of the segments. So first number is a segment id, next length ...

If no segment is defined, 0, 0 is returned.

3.5.6 TRACe[1|2]:FREE?

The query returns the amount of user memory space available for traces in the following form <bytes available>, <bytes in use>, < contiguous bytes available>.

3.6 Level Commands ([SOURce]:VOLTage Subsystem)

The following SCPI command subsystem specifies all the commands to change amplitude and offset of a signal to be output.

3.6.1 [:SOURce]:VOLTage[1|2] [:LEVel][:IMMediate][:AMPLitude][?]

Set output Amplitude. Valid Range 0.65 to 0.7.

3.6.2 [:SOURce]:VOLTage[1|2] [:LEVel][:IMMediate]:OFFSet[?]

Set output Offset. Valid Range -0.02 to +0.02

3.7 Sampling Frequency Commands

3.7.1 [SOURce:]FREQuency:RASTer[?] [frequency]

Set or query the sample frequency (arb waveform frequency).

3.7.2 [SOURce:]FREQuency:RASTer:EXTernal[?] [frequency]

Set or query the external sample frequency (arb waveform frequency) provided at SCLK IN.

3.7.3 [SOURce:]FREQuency:RASTer:SOURce[1|2] [?] [INTernal|EXTernal]

Set or query the selected clock source for a channel. When switching to EXTernal a stable sample clock that matches the frequency set with FREQ:RAST:EXT must be supplied at SCLK IN.

3.8 Output Commands (OUTPut Subsystem)

3.8.1 OUTPut[1|2][:NORMal][:STATe][?] [0|1|ON|OFF]

Switch output on or off.

3.9 Status reporting Commands (STATus Subsystem)

3.9.1 STATus:QUEStionable

The Questionable Data register group provides information about the quality or integrity of the instrument. Any or all of these conditions can be reported to the Questionable Data summary bit through the enable register.

Bit 0 is used to indicate output protection has fired.

Bit 5 is used to indicate instable external clock.

3.9.1.1 STATus:QUEStionable:CONDition?

Reads the condition register in the questionable status group. It's a read-only register and bits are not cleared when you read the register. A query of the register returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register.

3.9.1.2 STATus:QUEStionable:ENABle[?]

Sets or queries the enable register in the questionable status group. The selected bits are then reported to the Status Byte. A *CLS will not clear the enable register but it does clear all bits in the event register. To enable bits in the enable register, you must write a decimal value which corresponds to the binary-weighted sum of the bits you wish to enable in the register.

3.9.1.3 STATus:QUEStionable[:EVENt]?

Reads the event register in the questionable status group. It's a read-only register. Once a bit is set, it remains set until cleared by this command or *CLS command. A query of the register returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register.

3.9.1.4 STATus:QUEStionable:NTRansition

Sets or queries the negative-transition register in the questionable status group. A negative transition filter allows an event to be reported when a condition changes from true to false. Setting both positive/negative filters true allows an event to be reported anytime the condition changes. Clearing both filters disables event reporting. The contents of transition filters are unchanged by *CLS and *RST.

3.9.1.5 STATus:QUEStionable:PTRansition

Set or queries the positive-transition register in the questionable status group. A positive transition filter allows an event to be reported when a condition changes from false to true. Setting both positive/negative filters true allows an event to be reported anytime the condition changes. Clearing both filters disables event reporting. The contents of transition filters are unchanged by *CLS and *RST.

3.9.2 STATus:OPERation

The operation status register contains conditions which are part of the instrument's normal operation.

Bit 8 is used to indicate system is started (init:imm has been done).

3.9.2.1 STATus: OPERation:CONDition?

Reads the condition register in the operation status group. It's a read-only register and bits are not cleared when you read the register. A query of the register returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register.

3.9.2.2 STATus: OPERation:ENABle[?]

Sets or queries the enable register in the operation status group. The selected bits are then reported to the Status Byte. A *CLS will not clear the enable register but it does clear all bits in the event register. To enable bits in the enable register, you must write a decimal value which corresponds to the binary-weighted sum of the bits you wish to enable in the register.

3.9.2.3 STATus: OPERation [:EVENt]?

Reads the event register in the condition status group. It's a read-only register. Once a bit is set, it remains set until cleared by this command or *CLS command. A query of the register returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register.

3.9.2.4 STATus: OPERation:NTRansition

Sets or queries the negative-transition register in the operation status group. A negative transition filter allows an event to be reported when a condition changes from true to false. Setting both positive/negative filters true allows an event to be reported anytime the condition changes. Clearing both filters disables event reporting. The contents of transition filters are unchanged by *CLS and *RST.

3.9.2.5 STATus: OPERation:PTRansition

Set or queries the positive-transition register in the operation status group. A positive transition filter allows an event to be reported when a condition changes from false to true. Setting both positive/negative filters true allows an event to be reported anytime the condition changes. Clearing both filters disables event reporting. The contents of transition filters are unchanged by *CLS and *RST.

3.10 System Related Commands (SYSTem Subsystem)

3.10.1 SYSTem:ERRor[:NEXT]?

Read and clear one error from the instrument's error queue.

A record of up to 30 command syntax or hardware errors can be stored in the error queue. Errors are retrieved in first-in-first-out (FIFO) order. The first error returned is the first error that was stored. Errors are cleared as you read them.

If more than 30 errors have occurred, the last error stored in the queue (the most recent error) is replaced with "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 instrument responds with 0,"No error".

The error queue is cleared by the *CLS command or when the power is cycled.

The error queue is not cleared by a reset (*RST) command.

The error messages have the following format (the error string may contain up to 255 characters).

-113,"Undefined header".

3.10.2 SYSTem:HELP:HEADers?

The HEADers? query shall return all SCPI commands and queries and IEEE 488.2 common commands and common queries implemented by the instrument. The response shall be a <DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA> element. The full path for every command and query shall be returned separated by linefeeds. The syntax of the response is defined as: The <nonzero digit> and sequence of <digit> follow the rules in IEEE 488.2, Section 8.7.9. An <SCPI header> is defined as: It shall contain all the nodes from the root. The <SCPI program mnemonic> contains the node in standard SCPI format. The short form shall use uppercase characters while the additional characters for the long form shall be in lowercase characters. Default nodes shall be surrounded by square brackets ([]).

3.11 Triggering Commands

3.11.1 ABORt

Stop signal generation.

3.11.2 ARM[:SEQuence][:STARt][:LAYer]:DELay[1|2][?] <delay>

Set or query trigger to data out delay. Valid range 0 to 30e-12.

3.11.3 INITiate:IMMediate

Start signal generation.

3.12 Common Command List

3.12.1 *CLS

Clear the event register in all register groups. This command also clears the error queue and cancels a *OPC operation. It doesn't clear the enable register.

3.12.2 *ESE

Enable bits in the Standard Event Status Register to be reported in the Status Byte. The selected bits are summarized in the “Standard Event” bit (bit 5) of the Status Byte Register. The *ESE? query returns a value which corresponds to the binary-weighted sum of all bits enabled decimal by the *ESE command. These bits are not cleared by a *CLS command. Value Range: 0–255.

3.12.3 *ESR?

Query the Standard Event Status Register. Once a bit is set, it remains set until cleared by a *CLS (clear status) command or queried by this command. A query of this register returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register.

3.12.4 *OPC

Set the “Operation Complete” bit (bit 0) in the Standard Event register after the previous commands have been completed.

3.12.5 *OPC?

Return “1” to the output buffer after the previous commands have been completed. Other commands cannot be executed until this command completes.

3.12.6 *OPT?

Read the installed options. The response consists of any number of fields separated by commas.

3.12.7 *RST

Reset instrument to its factory default state.

3.12.8 *SRE[?]

Enable bits in the Status Byte to generate a Service Request. To enable specific bits, you must write a decimal value which corresponds to the binary-weighted sum of the bits in the register. The selected bits are summarized in the “Master Summary” bit (bit 6) of the Status Byte Register. If any of the selected bits change from “0” to “1”, a Service Request signal is generated. The *SRE? query returns a decimal value which corresponds to the binary-weighted sum of all bits enabled by the *SRE command.

3.12.9 *STB ?

Query the summary (status byte condition) register in this register group. This command is similar to a Serial Poll but it is processed like any other instrument command. This command returns the same result as a Serial Poll but the “Master Summary” bit (bit 6) is not cleared by the *STB? command.

3.12.10 *TST?

Execute Self tests. If self-tests pass, a 0 is returned. A number lager than 0 indicates the number of failed tests. .

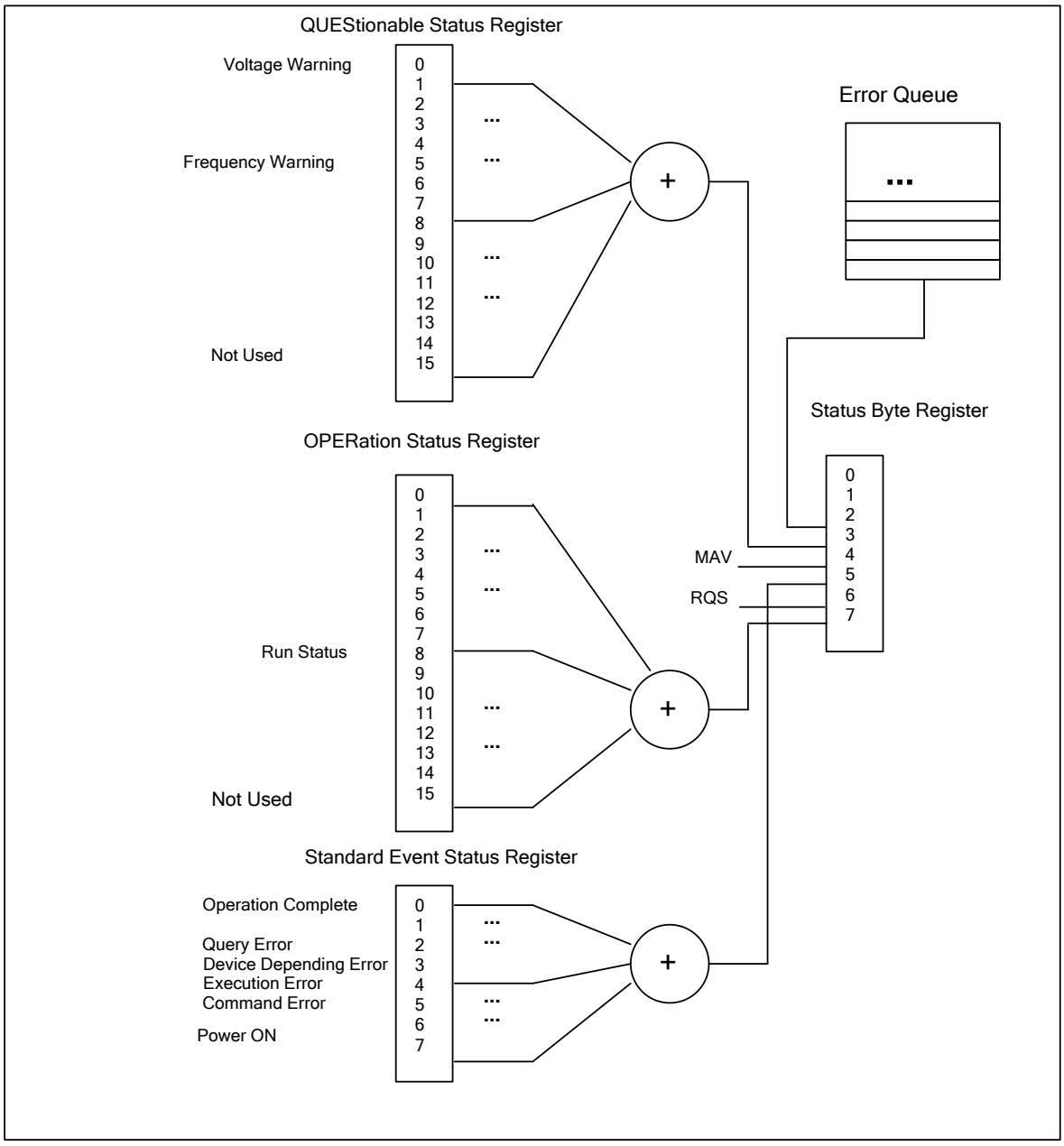
3.13 Status Model

Introduction

This section describes the structure of the SCPI status system used by the 81150A / 81160A. The status system records various conditions and states of the instrument in several register groups as shown on the following pages. Each of the register groups is made up of several low level registers called Condition registers, Event registers, and Enable registers which control the action of specific bits within the register group.

These groups are explained below:

- A condition register continuously monitors the state of the instrument. The bits in the condition register are updated in real time and the bits are not latched or buffered. This is a read-only register and bits are not cleared when you read the register. A query of a condition register returns a decimal value which corresponds to the binary-weighted sum of all bits set in that register.
- An event register latches the various events from changes in the condition register. There is no buffering in this register; while an event bit is set, subsequent events corresponding to that bit are ignored. This is a read only register. Once a bit is set, it remains set until cleared by query command (such as STAT:QUES:EVEN?) or a *CLS (clear status) command. A query of this register returns a decimal value which corresponds to the binary-weighted sum of all bits set in that register.
- An enable register defines which bits in the event register will be reported to the Status Byte register group. You can write to or read from an enable register. A *CLS (clear status) command will not clear the enable register but it does clear all bits in the event register. A STAT:PRES command clears all bits in the enable register. To enable bits in the enable register to be reported to the Status Byte register, you must write a decimal value which corresponds to the binary weighted sum of the corresponding bits.
- Transition Filters are used to detect changes of the state in the condition register and set the corresponding bit in the event register. You can set transition filter bits to detect positive transitions (PTR), negative transitions (NTR) or both. Transition filters are read/write registers. They are not affected by *CLS.



Status Register Structure

3.13.1 Status Byte Register

The Status Byte summary register reports conditions from the other status registers. Data that is waiting in the instrument's output buffer is immediately reported on the "Message Available" bit (bit 4) for example. Clearing an event register from one of the other register groups will clear the corresponding bits in the Status Byte condition register. Reading all messages from the output buffer, including any pending queries, will clear the "Message Available" bit. To set the enable register mask and generate an SRQ (service request), you must write a decimal value to the register using the *SRE command.

Bit Number		Decimal Value	Definition
0	Not used	1	Not Used. Returns "0"
1	Not used	2	Not Used. Returns "0"
2	Error Queue	4	One or more error are stored in the Error Queue
3	Questionable Data	8	One or more bits are set in the Questionable Data Register (bits must be enabled)
4	Message Available	16	Data is available in the instrument's output buffer
5	Standard Event	32	One or more bits are set in the Standard Event Register
6	Master Summary	64	One or more bits are set in the Status Byte Register
7	Operational Data	128	One or more bits set in the Operation Data Register (bits must be enabled)

3.13.2 Status Commands

The PRESet command is an event that configures the SCPI and device-dependant status data structures. The mandatory mechanism is defined in part by the IEEE 488.2.

3.13.2.1 :STATus:PRESet

It clears all status group event registers. Presets the status group, enables PTR and NTR registers as follows:
ENABle = 0x0000, PTR = 0xffff, NTR = 0x0000

3.13.3 Status Questionable Data Register Command Subsystem

The Questionable Data register group provides information about the quality or integrity of the instrument. Any or all of these conditions can be reported to the Questionable Data summary bit through the enable register.

Bit Number		Decimal Value	Definition
0	Voltage warning	1	Output has switched off (to protect itself)
1	Not used	2	Returns “0”
2	Not used	4	Returns “0”
3	Not used	8	Returns “0”
4	Not used	16	Returns “0”
5	Frequency warning	32	Output signal is invalid
6	Not used	64	Returns “0”
7	Not used	128	Returns “0”
8	Not used	256	Returns “0”
9	Not used	512	Returns “0”
10	Not used	1024	Returns “0”
11	Not used	2048	Returns “0”
12	Not used	4096	Returns “0”
13	Not used	8192	Returns “0”
14	Not used	16384	Returns “0”
15	Not used	32768	Returns “0”

The following commands access the questionable status group.

3.13.3.1 :STATus:QUEStionable[:EVENt]?

Reads the event register in the questionable status group. It's a read-only register. Once a bit is set, it remains set until cleared by this command or *CLS command. A query of the register returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register.

3.13.3.2 :STATus:QUEStionable:CONDition?

Reads the condition register in the questionable status group. It's a read-only register and bits are not cleared when you read the register. A query of the register returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register.

3.13.3.3 :STATus:QUEStionable:ENABle[?]

Sets or queries the enable register in the questionable status group. The selected bits are then reported to the Status Byte. A *CLS will not clear the enable register but it does clear all bits in the event register. To enable bits in the enable register, you must write a decimal value which corresponds to the binary-weighted sum of the bits you wish to enable in the register.

3.13.3.4 :STATus:QUEStionable:NTRansition[?]

Sets or queries the negative-transition register in the questionable status group. A negative transition filter allows an event to be reported when a condition changes from true to false. Setting both positive/negative filters true allows an event to be reported anytime the condition changes. Clearing both filters disable event reporting. The contents of transition filters are unchanged by *CLS and *RST.

3.13.3.5 :STATus:QUEStionable:PTRansition[?]

Set or queries the positive-transition register in the questionable status group. A positive transition filter allows an event to be reported when a condition changes from false to true. Setting both positive/negative filters true allows an event to be reported anytime the condition changes. Clearing both filters disable event reporting. The contents of transition filters are unchanged by *CLS and *RST.

3.13.4 Status Operation Status Subsystem

The Operation status register contains conditions which are part of the instrument's normal operation.

Bit Number		Decimal Value	Definition
0	Not used	1	Returns "0"
1	Not used	2	Returns "0"
2	Not used	4	Returns "0"
3	Not used	8	Returns "0"
4	Not used	16	Returns "0"
5	Not used	32	Returns "0"
6	Not used	64	Returns "0"
7	Not used	128	Returns "0"
8	Run Status	256	Indicates if system is running (:INIT:IMM) is done and signals are generated.
9	Not used	512	Returns "0"
10	Not used	1024	Returns "0"
11	Not used	2048	Returns "0"
12	Not used	4096	Returns "0"
13	Not used	8192	Returns "0"
14	Not used	16384	Returns "0"
15	Not used	32768	Returns "0"

The following commands access the operation status group.

3.13.4.1 :STATus:OPERation[:EVENt]?

Reads the event register in the operation status group. It's a read-only register. Once a bit is set, it remains set until cleared by this command or *CLS command. A query of the register returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register.

3.13.4.2 :STATus:OPERation:CONDition?

Reads the condition register in the operation status group. It's a read-only register and bits are not cleared when you read the register. A query of the register returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register.

3.13.4.3 :STATus:OPERation:ENABle[?]

Sets or queries the enable register in the operation status group. The selected bits are then reported to the Status Byte. A *CLS will not clear the enable register but it does clear all bits in the event register. To enable bits in the enable register, you must write a decimal value which corresponds to the binary-weighted sum of the bits you wish to enable in the register.

3.13.4.4 :STATus:OPERation:NTRansition[?]

Sets or queries the negative-transition register in the operation status group. A negative transition filter allows an event to be reported when a condition changes from true to false. Setting both positive/negative filters true allows an event to be reported anytime the condition changes. Clearing both filters disable event reporting. The contents of transition filters are unchanged by *CLS and *RST.

3.13.4.5 :STATus:OPERation:PTRansition[?]

Set or queries the positive-transition register in the operation status group. A positive transition filter allows an event to be reported when a condition changes from false to true. Setting both positive/negative filters true allows an event to be reported anytime the condition changes. Clearing both filters disable event reporting. The contents of transition filters are unchanged by *CLS and *RST.

3.14 Example Programs

Both example programs generate a Sample Frequency / 32 sinusoidal waveform. They demonstrate how to connect, recommended programming sequence, how to use binary waveform download.

3.14.1 C++ Example

```
#define _USE_MATH_DEFINES
#include <math.h>
#include <visa.h.>
#include <sstream>
#include <vector>
#include <string>

using namespace std;

static string ScpiBlockPrefix(size_t blocklen)
{
  ostringstream blockLenSStr;
  blockLenSStr << blocklen;
  ostringstream resultSStr;
  resultSStr << "#" << blockLenSStr.str().size() << blockLenSStr.str();
  return resultSStr.str();
}

static void GenerateWaveformCommands(int patternLength,
  int& sampleCount, vector<ViByte>& buffer1, vector<ViByte>& buffer2)
{
```

```
  // generate a sine wave, ensure granularity (48) and minimum samples (192)
  sampleCount = 0;
  vector<ViChar> binaryValues;
  do
  {
    for (int i = 0; i < patternLength; ++i)
    {
      const short dac = (short)(8191 * sin(2 * M_PI * i / patternLength));
      const short value = (short)(dac << 2);
      binaryValues.push_back((ViChar)value);
      binaryValues.push_back((ViChar)(value >> 8));
    }
    sampleCount += patternLength;
  } while (((sampleCount % 48) != 0) || (sampleCount < 192));

  // encode the binary commands to download the waveform
  string bytes1 = string(":trac1:data 1,0,") +
ScpiBlockPrefix(binaryValues.size());
  string bytes2 = string(":trac2:data 1,0,") +
ScpiBlockPrefix(binaryValues.size());

  buffer1.resize(bytes1.size() + binaryValues.size());
  memcpy(&buffer1[0], bytes1.c_str(), bytes1.size());
  memcpy(&buffer1[bytes1.size()], &binaryValues[0], binaryValues.size());

  buffer2.resize(bytes2.size() + binaryValues.size());
  memcpy(&buffer2[0], bytes2.c_str(), bytes2.size());
  memcpy(&buffer2[bytes2.size()], &binaryValues[0], binaryValues.size());
}

int main()
{
  int error;
  ViSession session, vi;

  const int patternLength = 32;

  int sampleCount;
  vector<ViByte> buffer1;
  vector<ViByte> buffer2;
  GenerateWaveformCommands(patternLength, sampleCount, buffer1, buffer2);

  error = viOpenDefaultRM(&session);
  error = viOpen(session, "TCPIP0::127.0.0.1::HISLIP0::INSTR",
    VI_NO_LOCK, 10000, &vi);

  // ensure instrument is stopped
  error = viPrintf(vi, ":abor\n");

  // set up new waveforms, use binary transfer for waveforms
  error = viPrintf(vi, ":trac1:del:all;:trac2:del:all\n");
  error = viPrintf(vi, ":trac1:def 1,%d\n", sampleCount);
  error = viPrintf(vi, ":trac2:def 1,%d\n", sampleCount);
  ViUInt32 writtenCount;
  error = viWrite(vi, &buffer1[0], (ViUInt32)buffer1.size(), &writtenCount);
  error = viFlush(vi, VI_WRITE_BUF);
  error = viWrite(vi, &buffer2[0], (ViUInt32)buffer2.size(), &writtenCount);
  error = viFlush(vi, VI_WRITE_BUF);
```

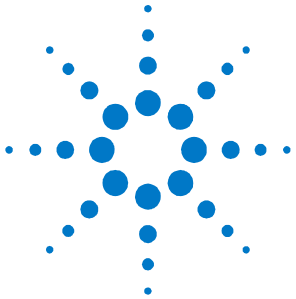
```
 // switch on outputs
  error = viPrintf(vi, ":outp1 on;:outp2 on\n");

  // finally start instrument, now samples are generated
  error = viPrintf(vi, ":init:imm\n");

  // wait until all commands have been executed
  error = viPrintf(vi, "*opc?\n");
  ViChar buffer[5000];
  error = viScanf(vi, "%t", buffer);

  error = viClose(vi);
  error = viClose(session);

  return 0;
}
```

3.14.2 C# Example

```
using System;
using System.Collections.Generic;
using System.Text;

namespace CsExample
{
  class Sine_32
  {
    public static int Main()
    {
      int error;
      int session, vi;
      const int timeout = 10000;

      const int patternLength = 32;

      int sampleCount;
      byte[] buffer1;
      byte[] buffer2;
      GenerateWaveformCommands(patternLength,
        out sampleCount, out buffer1, out buffer2);

      error = visa32.viOpenDefaultRM(out session);
      error = visa32.viOpen(session, "TCPIP0::127.0.0.1::HISLIP0::INSTR",
        visa32.VI_NO_LOCK, timeout, out vi);
      error = visa32.viSetAttribute(vi, visa32.VI_ATTR_TMO_VALUE, timeout);

      // ensure instrument is stopped
      error = visa32.viPrintf(vi, ":abor\n");
```

```
      // set up new waveforms, use binary transfer for waveforms
      error = visa32.viPrintf(vi, ":trac1:del:all;:trac2:del:all\n");
      error = visa32.viPrintf(vi, ":trac1:def 1," + sampleCount + "\n");
      error = visa32.viPrintf(vi, ":trac2:def 1," + sampleCount + "\n");
      int writtenCount;
      error = visa32.viWrite(vi, buffer1, buffer1.Length, out writtenCount);
      error = visa32.viFlush(vi, visa32.VI_WRITE_BUF);
      error = visa32.viWrite(vi, buffer2, buffer2.Length, out writtenCount);
      error = visa32.viFlush(vi, visa32.VI_WRITE_BUF);

      // switch on outputs
      error = visa32.viPrintf(vi, ":outp1 on;:outp2 on\n");

      // finally start instrument, now samples are generated
      error = visa32.viPrintf(vi, ":init:imm\n");

      // wait until all commands have been executed
      error = visa32.viPrintf(vi, "*opc?\n");
      StringBuilder sb = new StringBuilder(5000);
      error = visa32.viScanf(vi, "%t", sb);

      error = visa32.viClose(vi);
      error = visa32.viClose(session);

      return 0;
    }

    static void GenerateWaveformCommands(int patternLength,
      out int sampleCount, out byte[] buffer1, out byte[] buffer2)
    {
      // generate a sine wave, ensure granularity (48) and minimum samples
(192)
      sampleCount = 0;
      List<byte> binaryValues = new List<byte>();
      do
      {
        for (int i = 0; i < patternLength; ++i)
        {
          short dac = (short)(8191 * Math.Sin(2 * Math.PI * i /
patternLength));
          short value = (short)(dac << 2);
          binaryValues.Add((byte)value);
          binaryValues.Add((byte)(value >> 8));
        }
        sampleCount += patternLength;
      } while (((sampleCount % 48) != 0) || (sampleCount < 192));
      int binaryLength = binaryValues.Count;

      // encode the binary commands to download the waveform
      Byte[] bytes1 = ASCIIEncoding.ASCII.GetBytes(
        ":trac1:data 1,0," + ScpiBlockPrefix(binaryLength));
      Byte[] bytes2 = ASCIIEncoding.ASCII.GetBytes(
        ":trac2:data 1,0," + ScpiBlockPrefix(binaryLength));

      buffer1 = new byte[bytes1.Length + binaryLength];
      bytes1.CopyTo(buffer1, 0);
      binaryValues.CopyTo(buffer1, bytes1.Length);
```

```
      buffer2 = new byte[bytes2.Length + binaryLength];
      bytes2.CopyTo(buffer2, 0);
      binaryValues.CopyTo(buffer2, bytes2.Length);
    }

    static string ScpiBlockPrefix(int blocklen)
    {
      string blockLenString = blocklen.ToString();
      string result = "#" + blockLenString.Length.ToString() +
blockLenString;
      return result;
    }
  }
}
```



4 Characteristics

4.1 Performance Specification

4.1.1 DIRECT OUT 1 / DIRECT OUT 2

Type of output	Single-ended[1] or differential, DC-coupled
Skew between normal and complement outputs	5 ps (typ)
Impedance	50 Ω (nom)
Amplitude control	Specified into 50 Ω, levels double into high impedance
Range, single-ended	650 mV_{p-p} to 700 mV_{p-p}
Resolution	100 µV
Offset	-20 mV to +20 mV
Rise/fall time (20% to 80%)	75 ps (typ)
Bandwidth (3 dB) [2]	3.5 GHz (typ), excluding sin(x)/x roll-off
Bandwidth (5 dB) [2]	4.5 GHz (typ), excluding sin(x)/x roll-off
Delay: DIRECT OUT 1 to DIRECT OUT 2	
Fix delay	0 ns
Fix delay accuracy	± 20 ps
Variable delay	Independent for DIRECT OUT 1 and DIRECT OUT 2
Delay range	0 ps to 30 ps, variable
Resolution	50 fs, 4 digits
Accuracy	± 10 ps (typ)
Harmonic distortion[2] [3]	-72 dBc (typ), f_{out}=100 MHz -60 dBc (typ), f_{out}=10 MHz ... 3000 MHz
SFDR[2] (excluding harmonic distortion, $f_{Sa}/4 \pm f_{out}$, $f_{Sa}/2 - f_{out}$, $3f_{Sa}/4 - f_{out}$)	-80 dBc (typ), f_{out}=100 MHz, measured DC to 1 GHz -75 dBc (typ), f_{out}=100 MHz, measured DC to 3 GHz -63 dBc (typ), f_{out}=10 MHz ... 2000 MHz, measured DC to 3 GHz

$f_{Sa}/4 \pm f_{out}$ Spur[2)]	f_{out} = 100 MHz: - 75 dBc (typ) f_{out} = 1 GHz: - 60 dBc (typ) f_{out} = 2 GHz: - 52 dBc (typ)
$f_{Sa}/2-f_{out}$ Spur[2)]	f_{out} = 100 MHz: - 74 dBc (typ) f_{out} = 1 GHz: - 53 dBc (typ) f_{out} = 2 GHz: - 44 dBc (typ)
$3f_{Sa}/4-f_{out}$ Spur[2)]	f_{out} = 100 MHz: - 80 dBc (typ) f_{out} = 1 GHz: - 73 dBc (typ) f_{out} = 2 GHz: - 64 dBc (typ)
Two-Tone IMD[2)]	TTIMD = -73 dBc (typ), f_{out1} = 999.5 MHz, f_{out2} = 1000.5 MHz

[1)] Unused output must be terminated with 50 Ω to GND
[2)] SCLK = 7.2 GSa/s, amplitude = 700 mV_{p-p}
[3)] Measured with a balun

4.1.2 Sample Clock

There are two selectable sources for the sample clock:

1. Internal Synthesizer
2. SCLK IN: Sample clock input – SMA connector on the front panel

Both channels get the sample clock from the same clock source (internal or external). The channels operate synchronously.

4.1.3 Internal Synthesizer Clock Characteristics

Frequency range	6.5 GHz to 8 GHz (spec)
Frequency accuracy	± 10 ppm
Frequency stability	±1 ppm per year, ±1 ppm 0 °C to 40 °C

4.1.4 Sample Clock Input

A sample clock input (SCLK IN) is provided on the front panel.

Frequency range	6.5 GHz to 8 GHz
Input voltage range	+0 dBm to +11 dBm
Damage level	+15 dBm
Input impedance	50 Ω (nom), AC coupled
Transition time	< 1 ns
Connector	SMA

4.1.5 Sample Clock Output

The source for the Sample Clock Output can be either the Internal Synthesizer or the Sample Clock Input. The source for the Sample Clock Output is the same as the sample clock that is internally used to clock the DAC.

Source	Selectable, Internal Synthesizer or Sample Clock Input
Frequency range	6.5 GHz to 8 GHz
Output amplitude	350 mVpp (typ), fix
Transition time	20 ps (typ), (20% / 80 %)
Phase noise	< -110 dBc/Hz (typ) at 10 kHz offset, f_{Sa} = 8 GSa/s, f_{out} = 1 GHz
Impedance	50 Ω (nom), AC coupled
Connector	SMA

4.1.6 General

Power consumption	170 W (nom)
Operating temperature	5°C to 40°C
Storage temperature	-40°C to 70°C
Operating humidity	5 % to 80 % relative humidity, non-condensing
Operating altitude	up to 2000 m
Safety designed to	IEC 61010-1, UL 61010-1, CAN/CSA-C22.2 No. 61010-1
EMC tested to	IEC61326-1
Warm-up time	30 min
Calibration interval	1 year recommended
Warranty	1 year standard

Cooling requirements: When operating the M8190A choose a location that provides at least 80 mm of clearance at rear, and at least 30 mm of clearance at each side for the AXIe chassis.

4.1.7 Definitions

Specification (spec)

The warranted performance of a calibrated instrument that has been stored for a minimum of 2 hours within the operating temperature range of 0°C to – 40°C and after a 45-minute warm up period. Within ±10°C after autocal. All specifications include measurement uncertainty and were created in compliance with ISO-17025 methods.

Data published in this document are specifications (spec) only where specifically indicated.

Typical (typ)

The characteristic performance, which 80% or more of manufactured instruments will meet. This data is not warranted, does not include measurement uncertainty, and is valid only at room temperature (approximately 23°C).

Nominal (nom)

The mean or average characteristic performance, or the value of an attribute that is determined by design such as a connector type, physical dimension, or operating speed. This data is not warranted and is measured at room temperature (approximately 23°C).

Measured (meas)

An attribute measured during development for purposes of communicating the expected performance. This data is not warranted and is measured at room temperature (approximately 23°C).

Accuracy

Represents the traceable accuracy of a specified parameter. Includes measurement error and timebase error, and calibration source uncertainty.

4.2 Sequencing

The M8190A offers following modes of operation:

1. **Continuous, self armed:**
   The signal generation starts immediately after the waveform download and repeats the segment until the instrument is stopped.
2. **Continuous, armed:**
   After the waveform download the instruments starts after a 'Run' command has been received either by a remote command or from the GUI. After the start, the segment until repeats the instrument is stopped.

Sample memory size 1,610,612,736 samples per channel (~1610 MSa per channel)

Segment granularity 48 samples

Segments
Number of segments: 1
Minimum segment length: 240 samples
Maximum segment size: 1,610,612,736 samples per channel (~1610 MSa per channel)

Sections A single segment can consist of multiple sections that are downloaded individually to the instrument and are linked inside the M8190A to form a segment.

Fourth edition, November 2011
Printed in Germany

M8190-91010

Agilent Technologies