Keysight M8190A Arbitrary Waveform Generator

User's Guide

KEYSIGHT TECHNOLOGIES

Notices

Manual Part Number

M8190-91030

Edition

Edition 11, August 2014

Keysight Technologies, Deutschland GmbH

Herrenberger Str. 130

71034 Böblingen, Germany

For Assistance and Support

http://www.keysight.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. Keysight Technologies specifically disclaims the implied warranties of Merchantability and Fitness for a Particular Purpose.

ESD sensitive device

All front-panel connectors of the M8190A are sensitive to Electrostatic discharge (ESD).

We recommend to operate the instrument in an electrostatic safe environment.

There is a risk of instrument malfunction when touching a connector.

Please follow this instruction:

Before touching the front-panel connectors discharge yourself by touching the properly grounded mainframe.

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, Keysight 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. Keysight 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 Keysight 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 Keysight 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.keysight.com/find/M9505A*

Keysight Technologies 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 Keysight 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 0 °C – 40 °C (32 °F – 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 Keysight M8190A operates when installed in a Keysight 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

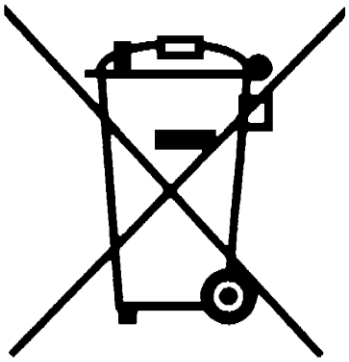
Indicates warning or caution. If you see this symbol on a product, you must refer to the manuals for specific Warning or Caution information to avoid personal injury or damage to the product.

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

General Recycling Mark

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 Keysight office, or see ***www.keysight.com/environment/product/*** *for more information.*

Contents

Contents 7

1 Introduction 18

1.1 Document History 20

1.2 Options 21

1.3 The Front Panel of the Two Channel Instrument 24

1.4 The Front Panel of the One Channel Instrument 26

1.5 Modes of Operation 27

1.5.1 Block Diagram 27

1.5.2 NRZ / DNRZ / Doublet 29

1.5.3 Delay Adjust 32

1.6 Installing Licenses 33

2 M8190A User Interface 35

2.1 Introduction 35

2.2 Launching the M8190A Soft Front Panel 35

2.3 M8190A User Interface Overview 38

2.3.1 Title Bar 38

2.3.2 Menu Bar 38

2.3.3 Status Bar 41

2.3.4 Clock/Output/Aux/Standard Waveform/ Multi-Tone/Complex Modulation/Import Waveform/Status/Control Tabs 41

2.3.5 Numeric Control Usage 41

2.4 Driver Call Log 43

2.5 Errors Window 44

2.6 Clock Tab 45

2.7 Output Tab 47

2.8 Aux Tab 50

2.9 Standard Waveform Tab 52

2.10 Multi-Tone Tab 58

2.11 Complex Modulation Tab 65
2.12 Import Waveform Tab 75
2.13 Status/Control Tab 81
2.14 Correction File Format 84
2.15 M8190 Soft Front Panel and M8190 Firmware 86

3 Sequencing 87

3.1 Introduction 87
3.1.1 Option Sequencing 87
3.1.2 Sequence Table 87
3.1.3 Sequencer Granularity 88
3.2 Sequencing Hierarchy 89
3.2.1 Segment 89
3.2.2 Sequence 89
3.2.3 Scenario 90
3.3 Trigger Modes 90
3.3.1 Continuous 90
3.3.2 Triggered 90
3.3.3 Gated 91
3.4 Arm Mode 91
3.4.1 Self Armed 91
3.4.2 Armed 91
3.5 Advancement Modes 92
3.5.1 Auto 92
3.5.2 Conditional 92
3.5.3 Repeated 92
3.5.4 Single 93
3.6 Sequencer Controls 93
3.6.1 External Inputs 93
3.6.2 Logical Functions 98
3.6.3 Internal Trigger Generator 99
3.6.4 Mapping External Inputs to Logical Functions 99
3.7 Sequencer Execution Flow 102
3.8 Sequencer Modes 103
3.8.1 Arbitrary Mode 103
3.8.2 Sequence Mode 108

3.8.3 Scenario Mode 115
3.9 Dynamic Sequencing 119
3.9.1 Dynamic Continuous 120
3.9.2 Dynamic Triggered 121
3.10 Idle Command Segments 122
3.11 Limitations 124
3.11.1 Segment Length and Linear Playtime 124

4 Digital Up-Conversion 128

4.1 Introduction 128
4.1.1 Direct Mode 129
4.1.2 Digital Up-Conversion Modes 129
4.1.3 Configuration at Run-Time 130
4.2 IQ Modulation 130
4.2.1 Interpolated Modes 130
4.2.2 Markers in Interpolated Modes 131
4.3 Configuration at Run-Time 131
4.3.1 Configuration using Standard Data Segments 132
4.3.2 Configuration using Configuration Segments 134
4.3.3 Sample-Aligned Configuration Using Markers 145
4.4 Amplitude Scaling 146
4.4.1 Scaling by using the amplitude table 146
4.4.2 Scaling by using the action table 147
4.4.3 Scaling by using the SCPI command or the Soft Front Panel 147
4.5 Coarse Delay and Digital Up-Conversion 148
4.6 Doublet Mode and Digital Up-Conversion 148

5 Streaming 149

5.1 Introduction 149
5.2 Streaming Implementation Using Dynamic Modes 149
5.3 Streaming Implementation Using the Ring Buffer Mechanism 150
5.3.1 Requirements 151
5.3.2 Theory of Operation 154
5.3.3 Examples 155

6 Markers 157

6.1 Introduction 157

6.2 Sample Markers 157

6.2.1 Pause Between Multiple Marked Samples 159

6.2.2 Sample Marker in Looped Segments 159

6.2.3 Sample Marker in Segments which are Addressed Offset Based 160

6.3 Sync Markers 160

7 General Programming 161

7.1 Introduction 161

7.2 IVI-COM Programming 162

7.3 SCPI Programming 163

7.3.1 AgM8190Firmware.exe 163

7.4 Programming Recommendations 167

7.5 System Related Commands (SYSTem Subsystem) 169

7.5.1 :SYSTem:ERRor[:NEXT]? 169

7.5.2 :SYSTem:HELP:HEADers? 170

7.5.3 :SYSTem:LICense:EXTended:LIST? 170

7.5.4 :SYSTem:SET[?] 171

7.5.5 :SYSTem:VERSion? 172

7.5.6 :SYSTem:COMMunicate:*? 172

7.6 Common Command List 177

7.6.1 *IDN? 177

7.6.2 *CLS 177

7.6.3 *ESE 177

7.6.4 *ESR? 177

7.6.5 *OPC 178

7.6.6 *OPC? 178

7.6.7 *OPT? 178

7.6.8 *RST 178

7.6.9 *SRE[?] 178

7.6.10 *STB? 179

7.6.11 *TST? 179

7.6.12 *LRN? 179

7.6.13 *WAI? 179

7.7 Status Model 180
7.7.1 :STATus:PRESet 182
7.7.2 Status Byte Register 182
7.7.3 Questionable Data Register Command Subsystem 183
7.7.4 Operation Status Subsystem 185
7.7.5 Voltage Status Subsystem 187
7.7.6 Frequency Status Subsystem 188
7.7.7 Sequence Status Subsystem 188
7.7.8 Run Status Subsystem 189
7.8 :ARM/TRIGger Subsystem 190
7.8.1 :ABORt[1|2] 190
7.8.2 :ARM[:SEQuence][:STARt][:LAYer]:DELay[1|2][?] <delay>|MINimum|MAXimum 190
7.8.3 :ARM[:SEQuence][:STARt][:LAYer]:CDELay[1|2][?] <coarse_delay>|MINimum|MAXimum 192
7.8.4 :ARM[:SEQuence][:STARt][:LAYer]:RNOisefloor[1|2][?] OFF|ON|0|1 193
7.8.5 :INITiate:CONTinuous[1|2]:ENABle[?] SELF|ARMed 193
7.8.6 :INITiate:CONTinous[1|2][:STATe][?] OFF|ON|0|1 194
7.8.7 :INITiate:GATE[1|2][:STATe][?] OFF|ON|0|1 195
7.8.8 :INITiate:IMMediate[1|2] 196
7.8.9 :ARM[:SEQuence][:STARt][:LAYer]:TRIGger:LEVel[?] <level>|MINimum|MAXimum196
7.8.10 :ARM[:SEQuence][:STARt][:LAYer]:TRIGger:IMPedance[?] LOW|HIGH 197
7.8.11 :ARM[:SEQuence][:STARt][:LAYer]:TRIGger:SLOPe[?] POSitive|NEGative|EITHer .. 198
7.8.12 :ARM[:SEQuence][:STARt][:LAYer]:TRIGger:SOURce[?] EXTernal|INTernal 199
7.8.13 :ARM[:SEQuence][:STARt][:LAYer]:TRIGger:FREQuency[?] <frequency>|MINimum|MAXimum 199
7.8.14 :ARM[:SEQuence][:STARt][:LAYer]:EVENt:LEVel[?] <level>|MINimum|MAXimum 200
7.8.15 :ARM[:SEQuence][:STARt][:LAYer]:EVENt:IMPedance[?] LOW|HIGH 201
7.8.16 :ARM[:SEQuence][:STARt][:LAYer]:EVENt:SLOPe[?] POSitive|NEGative|EITHer 201
7.8.17 :ARM[:SEQuence][:STARt][:LAYer]:DYNPort:WIDTh[?] LOWerbits|ALLBits 202
7.8.18 :TRIGger[:SEQuence][:STARt]:SOURce:ENABle[?] TRIGger|EVENt 203
7.8.19 :TRIGger[:SEQuence][:STARt]:ENABle[1|2]:HWDisable[:STATe][?] 0|1|OFF|ON ... 204
7.8.20 :TRIGger[:SEQuence][:STARt]:BEGin[1|2]:HWDisable[:STATe][?] 0|1|OFF|ON 205
7.8.21 :TRIGger[:SEQuence][:STARt]:ADVance[1|2]:HWDisable[:STATe][?] 0|1|OFF|ON 205
7.9 TRIGger - Trigger Input 207
7.9.1 :TRIGger[:SEQuence][:STARt]:SOURce:ADVance[?] TRIGger|EVENt|INTernal 207
7.9.2 :TRIGger[:SEQuence][:STARt]:ENABle[1|2][:IMMediate] 207
7.9.3 :TRIGger[:SEQuence][:STARt]:BEGin[1|2][:IMMediate] 208
7.9.4 :TRIGger[:SEQuence][:STARt]:BEGin[1|2]:GATE[:STATe][?] OFF|ON|0|1 208
7.9.5 :TRIGger[:SEQuence][:STARt]:ADVance[1|2][:IMMediate] 209

7.10 :FORMat Subsystem 210
7.10.1 :FORMat:BORDer NORMal|SWAPped 210
7.11 :INSTrument Subsystem 211
7.11.1 :INSTrument:COUPle:STATe[1|2][?] OFF|ON|0|1 211
7.11.2 :INSTrument:SLOT[:NUMBer]? 212
7.11.3 :INSTrument:IDENtify [<seconds>] 212
7.11.4 :INSTrument:IDENtify:STOP 213
7.11.5 :INSTrument:MMODule:CONFig? 213
7.11.6 :INSTrument:MMODule:MODE? 214
7.12 :MMEMory Subsystem 216
7.12.1 :MMEMory:CATalog? [<directory_name>] 216
7.12.2 :MMEMory:CDIRectory [<directory_name>] 217
7.12.3 :MMEMory:COPY <string>,<string>[,<string>,<string>] 218
7.12.4 :MMEMory:DELete <file_name>[,<directory_name>] 218
7.12.5 :MMEMory:DATA <file_name>, <data> 219
7.12.6 :MMEMory:DATA? <file_name> 220
7.12.7 :MMEMory:MDIRectory <directory_name> 220
7.12.8 :MMEMory:MOVE <string>,<string>[,<string>,<string>] 221
7.12.9 :MMEMory:RDIRectory <directory_name> 221
7.12.10 :MMEMory:LOAD:CSTate <file_name> 222
7.12.11 :MMEMory:STORe:CSTate <file_name> 223
7.13 :OUTPut Subsystem 223
7.13.1 :OUTPut[1|2][:NORMal][:STATe][?] OFF|ON|0|1 223
7.13.2 :OUTPut[1|2]:COMPlement[:STATe][?] OFF|ON|0|1 224
7.13.3 :OUTPut:SCLK:SOURce[?] INTernal|EXTernal 225
7.13.4 :OUTPut[1|2]:ROUTe[:SELect][?] DAC|DC|AC 226
7.13.5 :OUTPut[1|2]:DIOFfset[?] <value>|MINimum|MAXimum 226
7.14 Sampling Frequency Commands 228
7.14.1 [:SOURce]:FREQuency:RASTer[?] <frequency>|MINimum|MAXimum 228
7.14.2 [:SOURce]:FREQuency:RASTer:EXTernal[?] <frequency>|MINimum|MAXimum 228
7.14.3 [:SOURce]:FREQuency:RASTer:SOURce[1|2][?] INTernal|EXTernal 229
7.15 Reference Oscillator Commands 230
7.15.1 [:SOURce]:ROSCillator:SOURce[?] EXTernal|AXI|INTernal 230
7.15.2 [:SOURce]:ROSCillator:SOURce:CHECk? EXTernal|AXI|INTernal 231
7.15.3 [:SOURce]:ROSCillator:FREQuency[?] <frequency>|MINimum|MAXimum 231

7.16 :DAC|DC|AC Subsystem ... 232
7.16.1 [:SOURce]:DAC|DC|AC[1|2]:FORMat[?] RZ|DNRZ|NRZ|DOUBlet ... 232
7.16.2 Voltage Ranges ... 233
7.16.3 [:SOURce]:DAC|DC|AC[1|2]:VOLTage[:LEVel][:IMMediate]:AMPLitude[?] <level> 234
7.16.4 [:SOURce]:DAC|DC[1|2]:VOLTage[:LEVel][:IMMediate]:OFFSet[?] <level> ... 234
7.16.5 [:SOURce]:DAC|DC[1|2]:VOLTage[:LEVel][:IMMediate]:HIGH[?] <level> ... 235
7.16.6 [:SOURce]:DAC|DC[1|2]:VOLTage[:LEVel][:IMMediate]:LOW[?] <level> ... 236
7.16.7 [:SOURce]:DC[1|2]:VOLTage[:LEVel][:IMMediate]:TERMination[?] <level> ... 236
7.17 :VOLTage Subsystem ... 237
7.17.1 [:SOURce]:VOLTage[1|2][:LEVel][:IMMediate][:AMPLitude][?] <level> ... 237
7.17.2 [:SOURce]:VOLTage[1|2][:LEVel][:IMMediate]:OFFSet[?] <level> ... 238
7.17.3 [:SOURce]:VOLTage[1|2][:LEVel][:IMMediate]:HIGH[?] <level> ... 238
7.17.4 [:SOURce]:VOLTage[1|2][:LEVel][:IMMediate]:LOW[?] <level> ... 239
7.18 :MARKer Subsystem ... 240
7.18.1 Ranges ... 240
7.18.2 [:SOURce]:MARKer[1|2]:SAMPle:VOLTage[:LEVel][:IMMediate]:AMPLitude[?] <level> ... 240
7.18.3 [:SOURce]:MARKer[1|2]:SAMPle:VOLTage[:LEVel][:IMMediate]:OFFSet[?] <level> 241
7.18.4 [:SOURce]:MARKer[1|2]:SAMPle:VOLTage[:LEVel][:IMMediate]:HIGH[?] <level> . 241
7.18.5 [:SOURce]:MARKer[1|2]:SAMPle:VOLTage[:LEVel][:IMMediate]:LOW[?] <level> .. 242
7.18.6 [:SOURce]:MARKer[1|2]:SYNC:VOLTage[:LEVel][:IMMediate]:AMPLitude[?] <level> 243
7.18.7 [:SOURce]:MARKer[1|2]:SYNC:VOLTage[:LEVel][:IMMediate]:OFFSet[?] <level> 243
7.18.8 [:SOURce]:MARKer[1|2]:SYNC:VOLTage[:LEVel][:IMMediate]:HIGH[?] <level> 244
7.18.9 [:SOURce]:MARKer[1|2]:SYNC:VOLTage[:LEVel][:IMMediate]:LOW[?] <level> 245
7.19 [:SOURce]:FUNCtion[1|2]:MODE ARBitrary|STSequence|STSCenario ... 246
7.20 :SEQuence Subsystem ... 247
7.20.1 [:SOURce]:SEQuence[1|2]:DEFine:NEW <length> ... 247
7.20.2 [:SOURce]:SEQuence[1|2]:DATA[?] <sequence_id>,<step>[,<length>|,<value>,<value>...|,<data block>] ... 248
7.20.3 [:SOURce]:SEQuence[1|2]:DELete <sequence_id> ... 250
7.20.4 [:SOURce]:SEQuence[1|2]:DELete:ALL ... 250
7.20.5 [:SOURce]:SEQuence[1|2]:CATalog? ... 251
7.20.6 [:SOURce]:SEQuence[1|2]:FREE? ... 251
7.20.7 [:SOURce]:SEQuence[1|2]:ADVance[?] <sequence_id>,AUTO|CONDitional|REPeat|SINGle ... 252
7.20.8 [:SOURce]:SEQuence[1|2]:COUNt <sequence_id>,<count> ... 253
7.20.9 [:SOURce]:SEQuence[1|2]:NAME[?] <sequence_id>,<name> ... 253
7.20.10 [:SOURce]:SEQuence[1|2]:COMMent[?] <sequence_id>,<comment> ... 254

7.21 :STABle Subsystem 255
7.21.1 Generating arbitrary waveforms in dynamic mode 255
7.21.2 Generating sequences in non-dynamic mode 255
7.21.3 Generating sequences in dynamic mode 256
7.21.4 Generating scenarios 256
7.21.5 [:SOURce]:STABle[1|2]:RESet 256
7.21.6 [:SOURce]:STABle[1|2]:DATA[?] <sequence_table_index>,(<length>|<block>|<value>,<value>...) 257
7.21.7 [:SOURce]:STABle[1|2]:DYNamic:HWDisable[:STATe][?] OFF|ON|0|1 264
7.21.8 [:SOURce]:STABle[1|2]:SEQuence:SELect[?] <sequence_table_index> 265
7.21.9 [:SOURce]:STABle[1|2]:SEQuence:STATe? 266
7.21.10 [:SOURce]:STABle[1|2]:DYNamic:[STATe][?] OFF|ON|0|1 267
7.21.11 [:SOURce]:STABle[1|2]:DYNamic:SELect[?] <sequence_table_index> 268
7.21.12 [:SOURce]:STABle[1|2]:SCENario:SELect[?] <sequence_table_index> 268
7.21.13 [:SOURce]:STABle[1|2]:SCENario:ADVance[?] AUTO|CONDitional|REPeat|SINGle 269
7.21.14 [:SOURce]:STABle[1|2]:SCENario:COUNt[?] <count> 270
7.21.15 [:SOURce]:STABle[1|2]:STReaming[:STATe][?] 0|1|OFF|ON 270

7.22 :TRACe Subsystem 272
7.22.1 Direct and Interpolated Mode 272
7.22.2 Waveform Memory Capacity 272
7.22.3 Waveform Granularity and Size 273
7.22.4 Waveform Data Format in Direct Mode 274
7.22.5 Waveform Data Format in Interpolated Mode 274
7.22.6 Arbitrary Waveform Generation 275
7.22.7 :TRACe[1|2]:DEFine <segment_id>,<length>[,<init_value1>[,<init_value2>]] 275
7.22.8 :TRACe[1|2]:DEFine:NEW? <length> [,<init_value1>[,<init_value2>]] 277
7.22.9 :TRACe[1|2]:DEFine:WONLy <segment_id>,<length>[,<init_value1>[,<init_value2>]] 278
7.22.10 :TRACe[1|2]:DEFine:WONLy:NEW? <length>[,<init_value1>[,<init_value2>]] 279
7.22.11 :TRACe[1|2]:DWIDth[?] WSPeed|WPRecision|INTX3|INTX12|INTX24|INTX48 ... 280
7.22.12 :TRACe[1|2][:DATA][?] <segment_id>,<offset>,(<length>|<block>|<numeric_values>) 281
7.22.13 :TRACe[1|2]:IQIMport <segment_id>,<file_name>,TXT|TXT14|BIN|IQBIN|BIN6030|BIN5110|LICensed|MAT89600|DSA90000|CSV,IONLy|QONLy|BOTH,ON|OFF|1|0,[,ALENgth|FILL|[,<init_valueI>[,<init_valueQ> [,<ignore_header_parameters>]] 282
7.22.14 :TRACe[1|2]:IMPort <segment_id>,<file_name>,TXT|TXT14|TXTD|BIN|BIN6030[,ALENgth|FILL|[,<dac_value>]] 290
7.22.15 :TRACe[1|2]:DELete <segment_id> 292
7.22.16 :TRACe[1|2]:DELete:ALL 293
7.22.17 :TRACe[1|2]:CATalog? 293

7.22.18 :TRACe[1|2]:FREE? 294
7.22.19 :TRACe[1|2]:SELect[?] <segment_id> 295
7.22.20 :TRACe[1|2]:NAME[?] <segment_id>,<name> 295
7.22.21 :TRACe[1|2]:COMMent[?] <segment_id>,<comment> 296
7.22.22 :TRACe[1|2]:ADVance[?] AUTO|CONDitional|REPeat|SINGle 297
7.22.23 :TRACe[1|2]:COUNt[?] <count> 298
7.22.24 :TRACe[1|2]:MARKer[:STATe][?] OFF|ON|0|1 298

7.23 :TEST Subsystem 300
7.23.1 :TEST:PON? 300
7.23.2 :TEST:TST? 300

7.24 CARRier Subsystem 302
7.24.1 [:SOURce]:CARRier[1|2]:FREQuency[?] <frequency_integral>[,<frequency_fractional>]|DEFault 302
7.24.2 [:SOURce]:CARRier[1|2]:FREQuency:INTegral[?] <frequency_integral>|MIN|MAX|DEFault 303
7.24.3 [:SOURce]:CARRier[1|2]:FREQuency:FRACtional[?] <frequency_fractional>|MIN|MAX|DEFault 304
7.24.4 [:SOURce]:CARRier[1|2]:SCALe[?] <scale>|MINimum|MAXimum|DEFault 304
7.24.5 [:SOURce]:CARRier[1|2]: POFFset [?] <phase- offset> |MINimum|MAXimum|DEFault 305

7.25 :FTABle Subsystem 306
7.25.1 [:SOURce]:FTABle[1|2]:DATA[?] <frequency_table_index>, (<length>|<block>|<frequency_integral_part>,<frequency_fractional_part >...) 306
7.25.2 [:SOURce]:FTABle[1|2]:RESet 307

7.26 :ATABle Subsystem 308
7.26.1 [:SOURce]:ATABle[1|2]:DATA[?] <amplitude_table_index>, (<length>|<block>|<amplitude_scale>,<amplitude_scale>...) 308
7.26.2 [:SOURce]:ATABle[1|2]:RESet 309

7.27 :ACTion Subsystem 310
7.27.1 [:SOURce]:ACTion[1|2]:DEFine[:NEW]? 310
7.27.2 [:SOURce]:ACTion[1|2]:APPend <action_sequence_index>,<action>[,<value> [,<value>]] 311
7.27.3 [:SOURce]:ACTion[1|2]:DELete <action_sequence_id> 312
7.27.4 [:SOURce]:ACTion[1|2]:DELete:ALL 313

7.28 Example Programs 313

8 Characteristics 314

8.1 Performance Specification 314

8.2 General 314

Appendix 315

A.1 Resampling Algorithms for Waveform Import 315

A.1.1 Resampling Requirements 315

A.1.2 Resampling Methodology 316

1 Introduction

Introduction

The Keysight M8190A is a 12 GSa/s Arbitrary Waveform Generator. It combines excellent signal fidelity with highest sampling rates. It offers up to 2 GSa waveform memory per channel and three different high bandwidth output paths to ideally address applications.

Features and Benefits

- Precision AWG with
  - 14-bit resolution up to 8 GSa/s with Option -14B
  - 12-bit resolution up to 12 GSa/s with Option -12G
  - 14-bit resolution up to 7.2 GSa/s with Option -DUC
- Spurious-free-dynamic range (SFDR) up to 80 dBc typical
- Harmonic distortion (HD) up to -72 dBc typical
- Standard arbitrary waveform memory size 128 MSa per channel with Option -14B and Option -12G, 64 MSa IQ sample pairs with Option -DUC
  - 2 GSa = 2048 MSa = 2,147,483,648 Sa arbitrary waveform memory per channel with Option -02G in 12 bit mode
  - 1.5 GSa = 1,536 MSa = 1,610,612,736 Sa arbitrary waveform memory per channel with Option -02G in 14 bit mode
  - 0.75 GSa = 768 MSa = 805,306,368 Sa IQ sample pair arbitrary waveform memory per channel with Option -02G in DUC mode. I.e. with Option -02G and Option -DUC, the M8190A offers 768 MSa I data and 768 MSa Q data per channel
- Analog bandwidth 3.5 GHz; (5 GHz with Option - AMP)
- Transition times 50 ps (20 % to 80 %) with Option - AMP
- Differential output
- Form-factor: 2-slot AXIe module
- Controlled via external PC or embedded AXIe system controller M9536A

Supporting Operating System

The Keysight M8190A supports the following operating systems:

- Windows Vista (32 bit)
- Windows Vista (64 bit)
- Windows 7 (32 bit)
- Windows 7 (64 bit)
- Windows 8

NOTE

The Keysight M8190A V3.0 or later does not support Windows® XP® operating system.

Additional Documents

Additional documentation can be found at:

- *http://www.keysight.com/find/M9505A* for 5-slot chassis related documentation.
- *http://www.keysight.com/find/M9502A* for 2-slot chassis related documentation.
- *http://www.keysight.com/find/M9045A* for PCIe laptop adapter card related documentation.
- *http://www.keysight.com/find/M9047A* for PCIe desktop adapter card related documentation.
- *http://www.keysight.com/find/M9536A* for embedded AXIe controller related documentation.
- *http://www.keysight.com/find/M8190A* for AXIe based AWG module related documentation.
- *http://www.keysight.com/find/M8192A* for AXIe based synchronization module related documentation.

1.1 Document History

First Edition (March, 2012)	The first edition of the User's Guide describes the functionality of firmware version V2.0.
Second Edition (May, 2012)	The second edition of the User's Guide describes the functionality of firmware version V2.1.
Third Edition (June, 2012)	The third edition of the User's Guide describes the functionality of firmware version V2.2.
Fourth Edition (September, 2012)	The fourth edition of the User's Guide describes the functionality of firmware version V2.3.
Fifth Edition (March, 2013)	The fifth edition of the User's Guide describes the functionality of firmware version V2.4.
Sixth Edition (October, 2013)	The sixth edition of the User's Guide describes the functionality of firmware version V3.0.
Seventh Edition (December, 2013)	The seventh edition of the User's Guide describes the functionality of firmware version V3.0.
Eighth Edition (February, 2014)	The eighth edition of the User's Guide describes the functionality of firmware version V3.1.
Ninth Edition (April, 2014)	The ninth edition of the User's Guide describes the functionality of firmware version V3.2.

1.2 Options

The AWG has a modular product structure and requires AXIe chassis.

Table 1-1: Options provided by M8190A

M8190A	Option	Available as SW upgrade?	Comment
1 channel	-001	No	Must order either -001 or -002
2 channel	-002	No	
14 bit / 8 GSa/s	-14B	Yes	Must order either -14B or -12G or both
12 GSa /s / 12 bit	-12G	Yes	
Addition DC and AC Amplifier	-AMP	Yes	Optional options
Upgrade to 2 GSa memory per channel	-02G	Yes	
Sequencer	-SEQ	Yes	
Digital Up-Conversion	-DUC	Yes	
Fast switching	-FSW	Yes	Fast switching for 12 GSa/s requires export control license
ISO17025	-1A7	No	Calibration options
Z540	-Z54	No	
As a standard configuration, the one and two channel versions contain 128 MSa of memory per channel.			

Option -001 or -002

With this option the number of channels is selected. The M8190A is available as a one channel (-001) or 2 channel (-002) instrument. There is no upgrade possible from option -001 to option -002.

Option -14B, -12G

These options define the resolution and the maximum sample rate of the M8190A:

- -12G : The High Speed Mode offers a maximum sample frequency of 12 GSa/s and a DAC resolution of 12 bits
- -14B : The High Precision Mode offers a maximum sample frequency of 8 GSa/s and a DAC resolution of 14 bits

Every M8190A must have minimum one of the two option (-12G or -14B). Both options can be installed in parallel on the M8190A. If both options are installed, it is possible to switch between the 12 bit and 14 bit mode by software configuration. Also, a software upgrade for the options is possible.

Option -AMP

This option enables two additional output amplifiers. This gives a total of three selectable output paths.

1. Direct Output: Optimized for optimum Spurious Free Dynamic Range (SFDR), Harmonic Distortions (HD)
2. DC Output: Optimized for Serial Data Applications. This amplifier offers up to 1V amplitude, a voltage window of -1 V to +3.3 V and an external termination voltage window of -1 V to +3.3 V.
3. AC Output: Optimized for RF applications. This amplifier offers up to 2 V amplitude, 5 GHz bandwidth, flatness correction – including sinx/x compensation at 12 GSa/s.

Refer to the data sheet for detailed specification.

Without -AMP, the M8190A offers a Direct Output path.

Option -AMP is software upgradeable.

Option -02G

This option offers 2048 MSa waveform memory per channel with Option -12G, 1,536 MSa with Option -14B and 768 MSa IQ sample pairs with Option -DUC.

Without -02G, the M8190A offers 128 MSa waveform memory per channel with Option -12G and Option -14B and 64 MSa IQ sample pairs with Option -DUC.

Option -02G is software upgradeable.

Option -SEQ

This option offers extensive sequencing capabilities. For more details, refer to the chapter *Sequencing*.

Option -SEQ is software upgradeable.

Option -DUC

This option provides 15 bit wide IQ sample data at a low speed, which is interpolated to the sample rate of the instrument and digitally up-converted to a carrier frequency using an IQ Modulator. Parts of the functionality described in the chapter *Digital Up-Conversion* uses sophisticated sequencing capability of the M8190A. As a result, the functionality described in the sections *Configuration at Run-Time* and *Amplitude Scaling* requires:
Option -SEQ. For more details, refer to the chapter *Digital Up-Conversion*.

Option -DUC is software upgradeable.

NOTE

In DUC mode, the DAC always operates with 14 bit resolution.

- Option -12G is installed and option -14B is not installed=> The M8190A operates with 14 bit resolution, if DUC mode is selected.
- Option -12G is not installed and option -14B is installed=> The M8190A operates with 14 bit resolution, if DUC mode is selected.
- Option -12G is installed and option -14B is installed=> The M8190A operates with 14 bit resolution, if DUC mode is selected.
- Option -12G is not installed and option -14B is not installed=> This configuration cannot be ordered.

Option -FSW

This option enables the M8190A to externally select or step through segments or sequences faster than every 100 µs.

Option -FSW is export controlled and is software upgradeable.

Option -1A7, -Z54

Calibration options

1.3 The Front Panel of the Two Channel Instrument

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

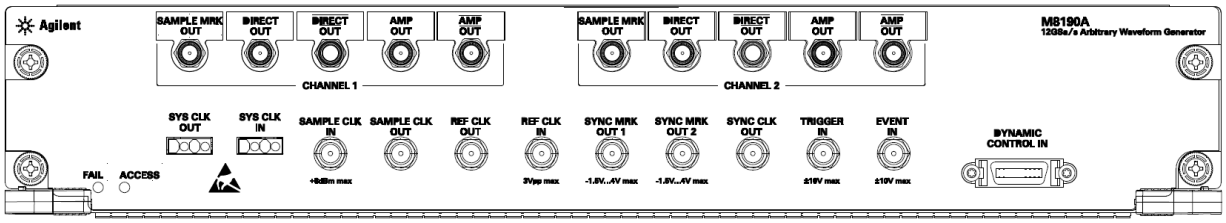


Figure 1-1: Front Panel of M8190A (2 Channels)

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 Converters (DAC). The Outputs can be switched ON or OFF with internal relays.
- The analog output signals of channel 1 and channel 2 (AMP OUT 1 & AMP OUT 2) are amplified output signals of the Digital to Analog Converters (DAC). The Outputs can be switched ON or OFF with internal relays. Two different amplifiers can be selected. One amplifier is optimized in performance for time domain applications (DC amplifier) and has differential outputs. The second amplifier is optimized for RF applications, maximum bandwidth and high output level (AC amplifier) and has a single-ended output. These two amplifiers are available, if option -AMP is installed. Option -AMP is SW upgradeable.
- The Sample Marker Output (SAMPLE MRK OUT 1 & SAMPLE MRK OUT 2) can be used to mark the beginning of a certain segment or a certain position in the analog signal to e.g. trigger an external device. It has a timing resolution of one sample clock period.
- The Synchronization Marker Output (SYNC MRK OUT 1 & SYNC MRK OUT 2) has a similar functionality as the Sample Marker Output. It has a timing resolution of 64 sample clock periods in 12 bit mode (48 sample clock periods in 14 bit mode and 24 IQ sample pairs in interpolated modes when using Option -DUC).

- The Trigger Input (TRIGGER IN) has a combined functionality as Trigger or Gate and is used to start the M8190A by an external signal. This input is defined in detail in the chapter *Sequencing*.
- The Event Input (EVENT IN) is used to e.g. step through segments or scenarios by an external signals. This input is defined in detail in the chapter *Sequencing*.
- The Sample Clock Input (SAMPLE CLK IN) can be used, if an external clock source shall be used to clock the Digital to Analog Converters.
- The Sample Clock Output (SAMPLE CLK OUT) can be used to output the clock signal from the internal sample clock or the sample clock input.
- The Reference Clock Input (REF CLK IN) can be used to synchronize to an external clock. The input frequency can vary between 1 MHz and 200 MHz.
- The Reference Clock Output (REF CLK OUT) can be used to synchronize a DUT to the M8190A. The output frequency is 100 MHz.
- The Sync Clock Output (SYNC CLK OUT) can be used to synchronize a DUT to the M8190A. E.g. the set-up and hold timings of the Trigger Input, Event Input or Dynamic Control Input signals refer to the Synch Clock Output.
- The Dynamic Control Input (DYNAMIC CONTROL IN) offers a 19 bit wide parallel interface to select the 512 k segments externally. This input is defined in detail in the chapter *Sequencing*.
- The System Clock Input (SYS CLK IN) and System Clock Output (SYS CLK OUT) are reserved for future use. Do not connect!

Status LED

Two LEDs 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'.
- In case the output relay has shut-off because of an external overload condition, this LED flashes.

1.4 The Front Panel of the One Channel Instrument

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

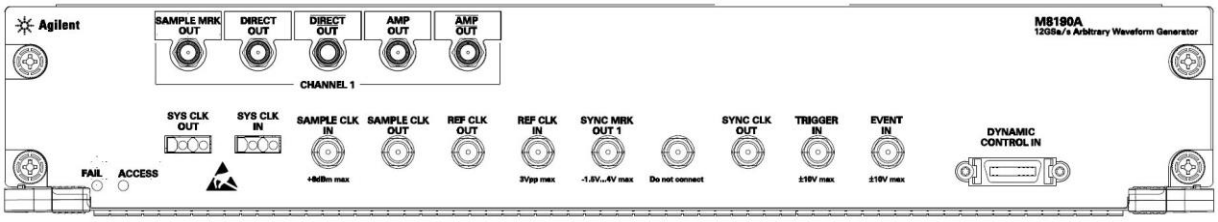


Figure 1-2: Front Panel of M8190A (1 Channel)

Inputs/Outputs

The description of the inputs and outputs of is given in the section *Inputs/Outputs*.

The 1 channel M8190A has following differences compared to the 2 channel M8190A:

- The SAMPLE MRK OUT, DIRECT OUT and AMP OUT of channel 2 is not available.
- The SYNC MRK OUT 2 of channel 2 is not available.
- There is one SMA connector labeled 'Do not connect'.
- Input (SYS CLK IN) and System Clock Output (SYS CLK IN) are reserved for future use. Do not connect!
- There is one SMA connector labeled 'Do not connect'. This connector is for future use.

Status LED

The functionality of the LED is identical with the 2 channel M8190. Refer to the section *Status LED* for a description of the status LED.

1.5 Modes of Operation

1.5.1 Block Diagram

The drawing below shows a block diagram of the instrument.

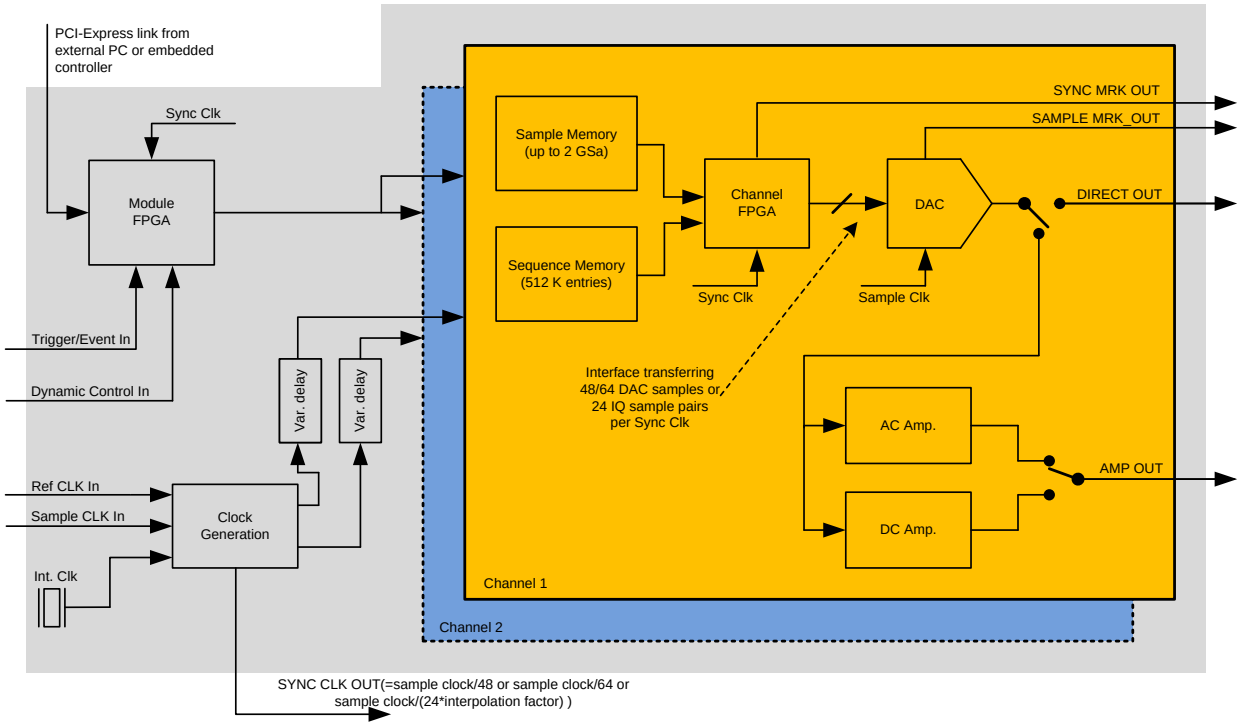


Figure 1-3: M8190A Block Diagram

The level of detail is chosen to provide a general high level understanding of how the instrument is working. Therefore, not all of ports are shown in the above diagram.

Channel

The data generation consists of a Sample Memory which contains the sample data, a Sequence Memory which contains the required information for sequencing like the sequence structure or loop counter values and a Channel FPGA which combines both into a sequence controlled sample stream. These parts cannot run at sample clock rate and therefore multiple samples must be executed in parallel at a lower clock speed. Depending on the selected direct mode, the Sync Clock, also called Sequencer Clock is the sample clock divided by 64 (high speed mode) or by 48 (high precision mode). When using the interpolated modes (refer to the chapter *Digital Up-Conversion*), one Sync Clock cycle covers 24 delivered (not interpolated) IQ sample pairs. Therefore, a sync clock cycle consists of "24*interpolation factor" DAC samples.

The DAC converts the parallel sample stream to sample clock granularity. The analog output of the DAC can either be used directly at the DIRECT OUT pin or can be routed through two different available amplifier paths.

Clock Generation

The clock generator can work on three clock sources: External Reference Clock (REF CLK IN), the SAMPLE CLK IN or an internally generated clock. The two channels can be delayed independently. The SYNC CLK OUT can be used to source the user's environment to provide synchronous signals at the TRIGGER IN and at the EVENT IN.

The Module FPGA

The Module FPGA is the common point of the instrument. The PCIe link is connected to it as well as the Trigger and Event input and the port used for dynamic sequencing. The whole chain from the Trigger/Event input to the Channel FPGA is sourced with the Sync Clock. This allows providing an exact output delay from the inputs to the DAC output for all cases where triggers and events are generated based on the Sync Clock output and where the corresponding setup and hold time conditions are met.

1.5.2 NRZ / DNRZ / Doublet

DAC Format Modes

Each channel uses two DACs (A & B) which are internally used. Each DAC usually contributes to signal 50% of a sample period.

The following three DAC format modes are available:

- NRZ (Non Return to Zero)
- DNRZ, (Double NRZ)
- Doublet

In addition, the following format mode is also available:

- RZ (Return to Zero)

It is not specified but available as over programming.

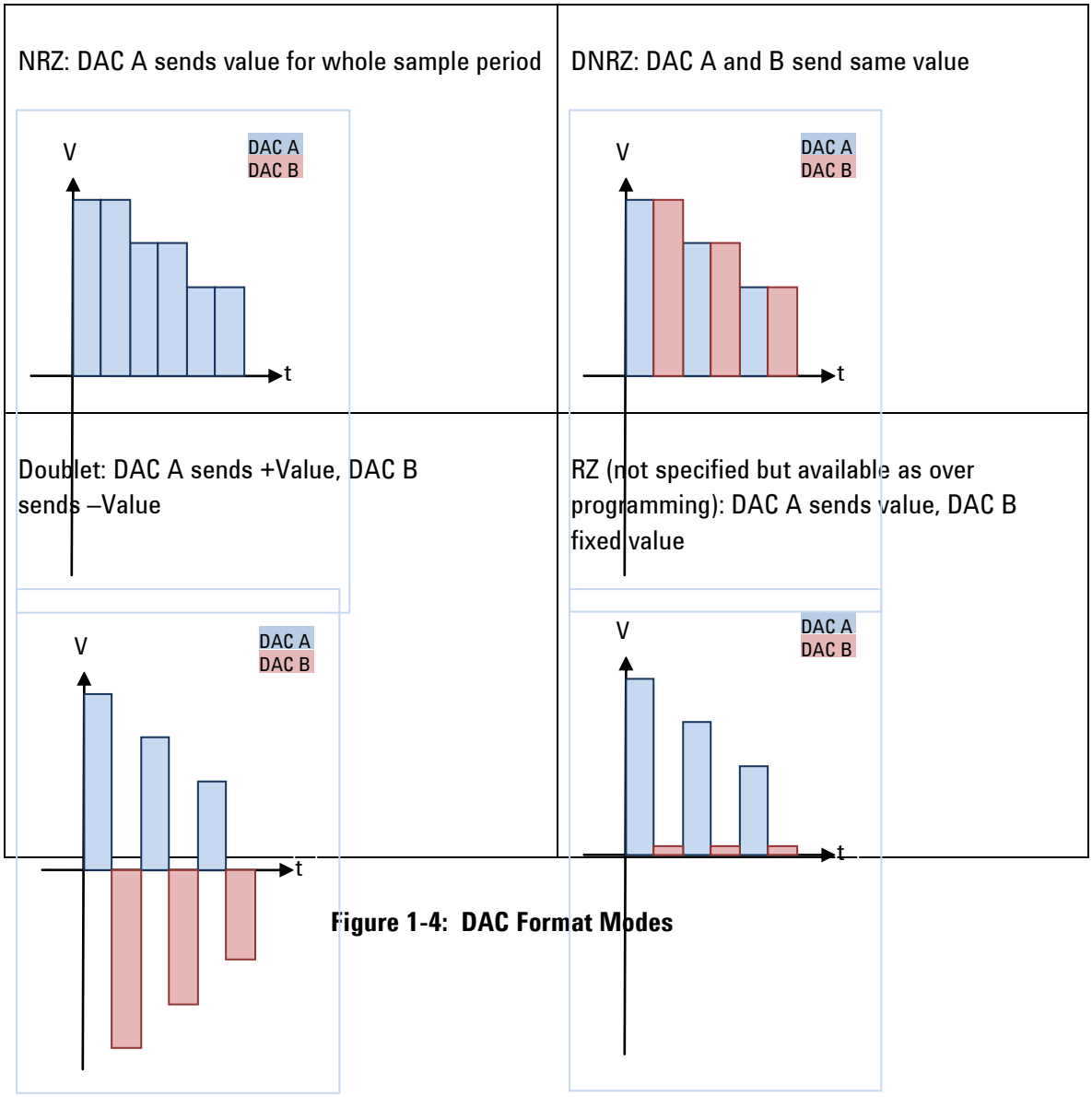


Figure 1-4: DAC Format Modes

The four modes have different amplitude vs. frequency responses, which are shown in the figure below.

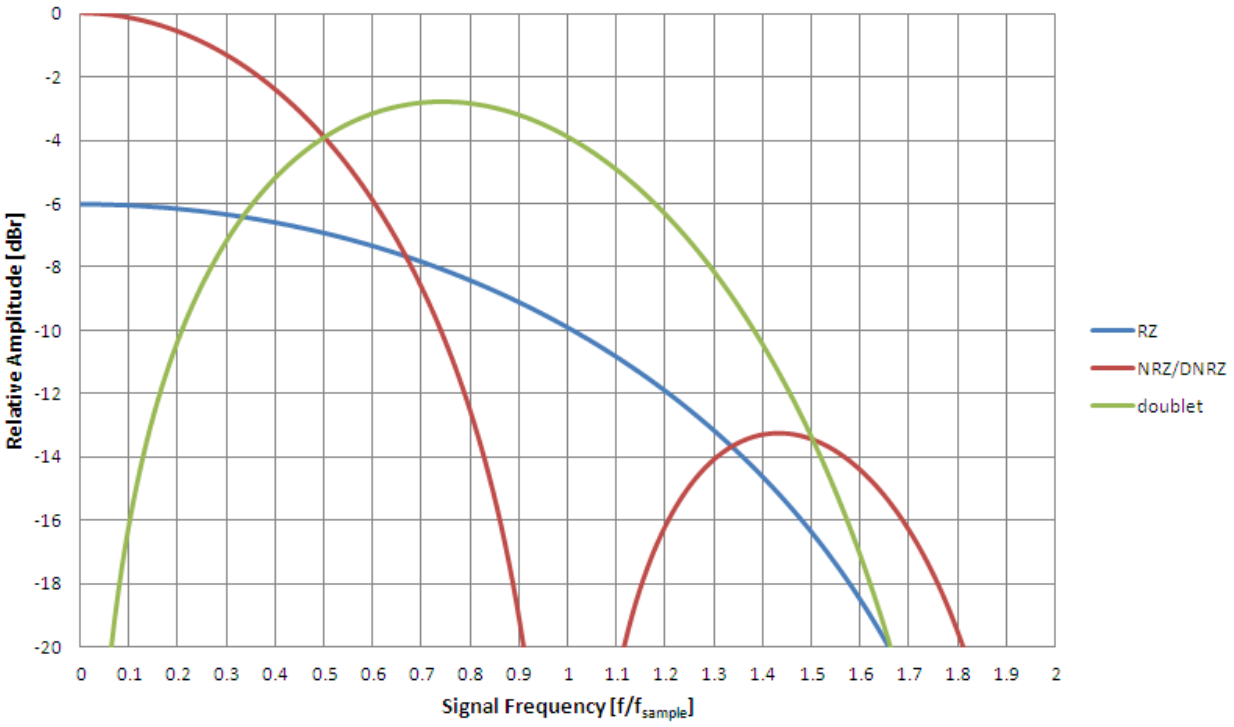


Figure 1-5: DNRZ/NRZ/RZ/Doublet Frequency Response

NOTE This is only comparing the idealized response of the modes, and ignores the roll off due to finite switching risetimes, output RC time constant and other losses

---

DNRZ — Best signal performance in 1st Nyquist region

$$ideal\ frequency\ response = \frac{Sin\ (x)}{x}$$

NRZ — For time domain applications, less ripple at 2 f_{sample}, more distortions than DNRZ

$$ideal\ frequency\ response = \frac{Sin\ (x)}{x}$$

Doublet	More output power and flatter frequency response in 2nd Nyquist region limited to f_{sample} > 1 GSa/s $ideal\ frequency\ response = \frac{Sin\ (\frac{x}{2})^2}{(\frac{x}{2})}$
RZ	Flatter frequency response in 2nd Nyquist region than DNRZ/NRZ (but less power than Doublet mode). Limited to f_{sample} > 1 GSa/s Attention!: does not return to zero single ended. (Both signals of the differential output return to a level slightly above the positive fullscale level, this is removed when used with AC path or Balun.)

1.5.3 Delay Adjust

In order to compensate for e.g. external cable length differences as well as the initial skew, channel 1 and channel 2 can be independently delayed with a very high timing resolution.

Setting the variable delay of channel 1 to 10 ps has following effect:

- Direct Out1 (or Amp Out1, if selected) is delayed by 10 ps with respect to following signals: Sample Clock Out, Sync Marker Out1, Sync Marker Out2, Direct Out2 (or Amp Out2, if selected), Trigger / Gate Input, Event Input
- Sample Marker Out1 is delayed by 10 ps with respect to following signals: Sample Clock Out, Sync Marker Out1, Sync Marker Out2, Direct Out2 (or Amp Out2, if selected), Trigger / Gate Input, Event Input

NOTE Modifying the variable delay of one channel always affects the delay of the Analog Output AND the Sample Marker of that channel.

The variable delay is split into two delay elements:

1. Fine delay
2. Coarse delay

The Variable Delay is the sum of Fine Delay and Coarse Delay. If a de-skew between channel 1 and channel 2 is needed, adjust in the first step the coarse delay to the optimum position. In the second step, use the fine delay to perfectly align both channels.

1.6 Installing Licenses

After you purchase a license and you acquire the corresponding license file, you need to install the license on M8190A.

You can install the new license in the following ways:

1. In Keysight License Manager, click the **File** menu, and then select **Install...**. An **Install License File(s)** window appears. In this window, browse to the location where you saved the license file. Select the license file, and then click the **Open** button.

2. To manually install a license by entering the appropriate license file information, click the **Tools** menu, click **Enter License Text...**. The **License Text Entry and Installation** dialog box appears.
   Type in the license data exactly as you received from Keysight. Click the **Install** button to install the license.

3. On Windows-based systems, you can install the license by copying the license file into the license directory
   `C:\Program Files\Keysight\licensing`.

Once the licenses are installed, you can use the Keysight License Manager to view all licenses for the local system as depicted in the following figure.

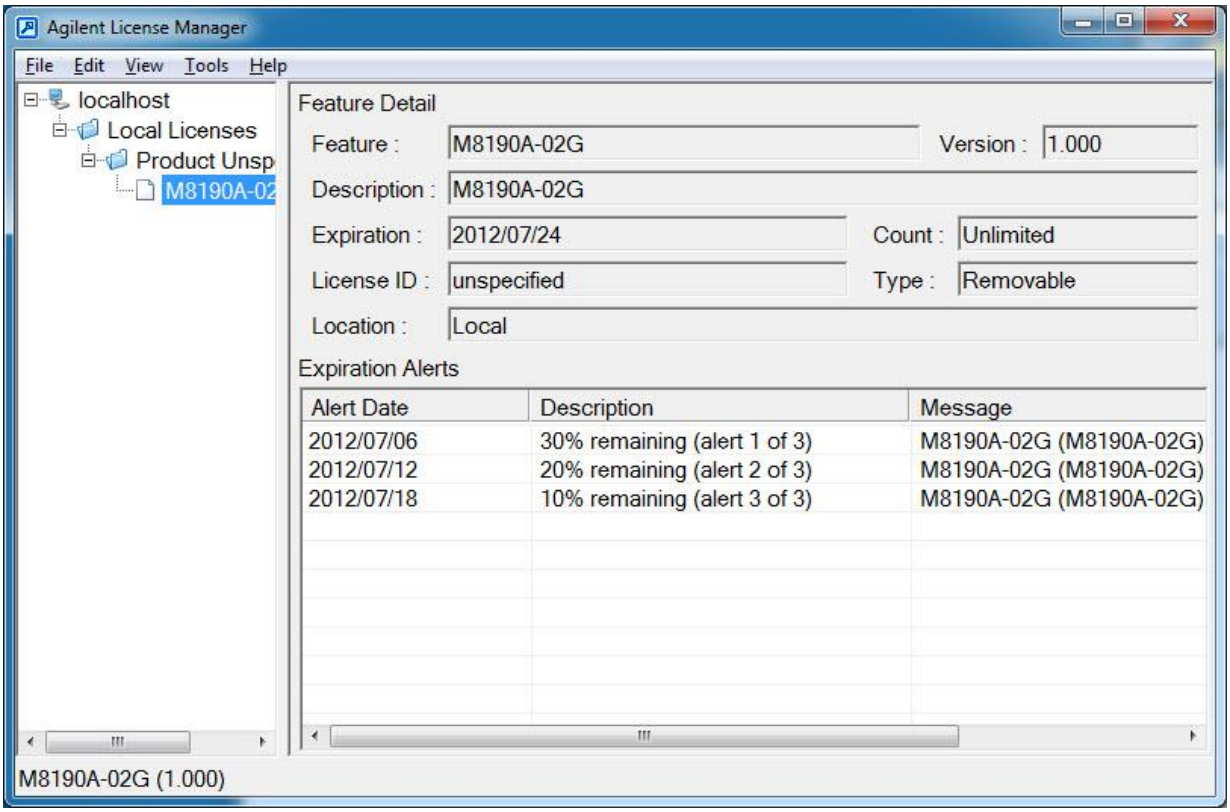


Figure 1-6: Using Keysight License Manager to view installed licenses

NOTE Licenses for instrument options are transferred to the M8190A module. They are later no longer visible in the Keysight License Manager.

2 M8190A User Interface

2.1 Introduction

This chapter describes the M8190A Soft Front Panel.

2.2 Launching the M8190A Soft Front Panel

There are three ways to launch the M8190A Soft Front Panel:

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

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

3. From the Keysight Connection Expert select the discovered M8190 module, select the “Installed Software” tab and press the “Start SFP” button.

The following screen will appear:

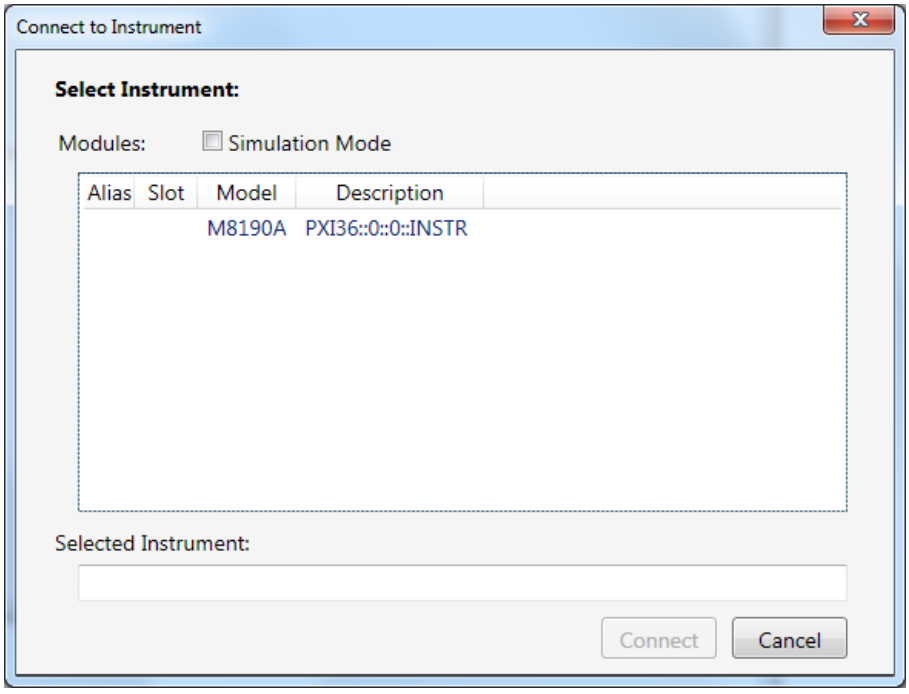


Figure 2-1: M8190A connected to PC

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.

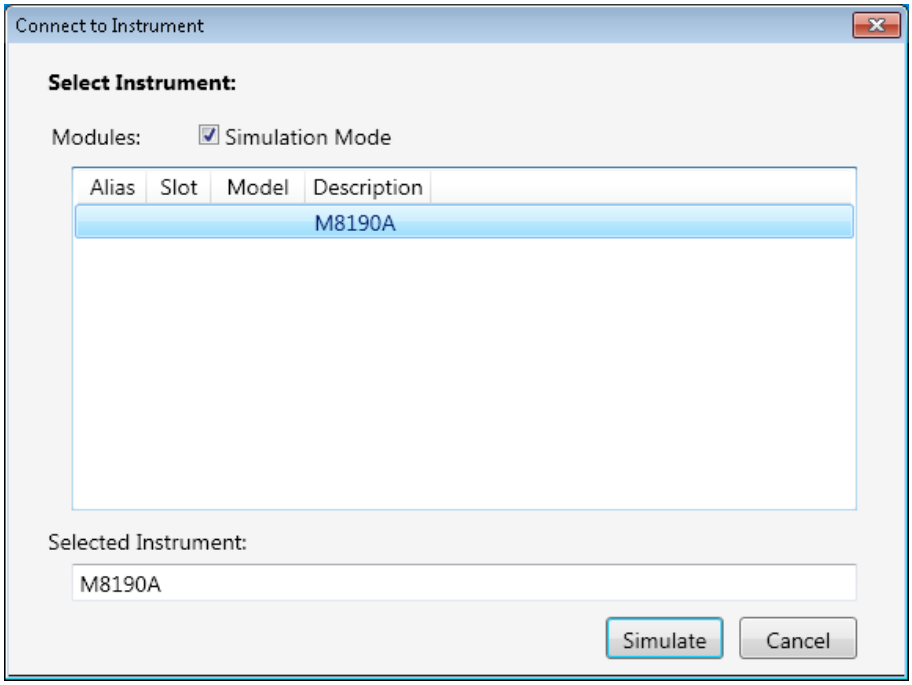


Figure 2-2: M8190A connected in stimulation mode

2.3 M8190A User Interface Overview

The M8190A user interface includes the following GUI items:

- Title Bar
- Menu Bar
- Status Bar
- Tabs (Clock, Output, Aux, Standard Waveform, Multi-Tone, Complex Modulation, Import Waveform and Status/Control)

The detailed information on these GUI items is described 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
- Utilities
- 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 – Save Configuration As...
  Saves configuration as a text file.

- File – Load Configuration...
  Load the previously saved configuration file.

- 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.

- View – Auto Refresh
  Reads the instrument state periodically 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 the *Errors 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* window.

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/Aux/Standard Waveform/ Multi-Tone/Complex Modulation/Import Waveform/Status/Control Tabs

These tabs are used to configure the most important parameters of the M8190A module. They are described in detail in the sections that follow.

2.3.5 Numeric Control Usage

The numeric control is used to adjust the value and units. Whenever you bring the mouse pointer over the numeric control, a tooltip appears which shows the possible values in that range.

Figure 2-3: Tooltip showing possible values in the range

The numeric controls can be used in the following ways:

- Use the up/down arrows to change the value. The control automatically stops at the maximum/minimum allowed value.
- You can increase or decrease the value starting at a specific portion of the value. To do this, place the cursor to the right of the targeted digit and use the up/down arrows. This is especially useful when changing a signal characteristic that is immediately implemented, and observing the result in another instrument. For example, you can change the signal generator's frequency by increments of 10 MHz and observe the measured result in a signal analyzer:

Figure 2-4: Typing directly into the field

- Type directly into the field and press the Enter key. If you enter a value outside the allowed range, the control automatically limits the entered value to the maximum or minimum allowed value.
- When you type the value, you can type the first letter of the allowed unit of measure to set the units. For example, in the Frequency control you can use "H", "K", "M", or "G" to specify hertz, kilohertz, megahertz, or gigahertz, respectively. (The control is not case sensitive.)

The controls allow scientific notation if it is appropriate to the allowed range. Type the first decimal number, enter an "E", and omit any trailing zeroes. For example, in the Frequency control you can type 2.5e+9 and press Enter to set the frequency to 2.5 GHz. (The plus sign is automatically inserted if it is omitted.)

2.4 Driver Call Log

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

Figure 2-5: Driver Call Log Window

It has the following buttons:

- Save...

  Saves the Driver Call Log as a text file.
- Clear History

  Clears the Driver Call Log.
- Close

  Exits the window.

2.5 Errors Window

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

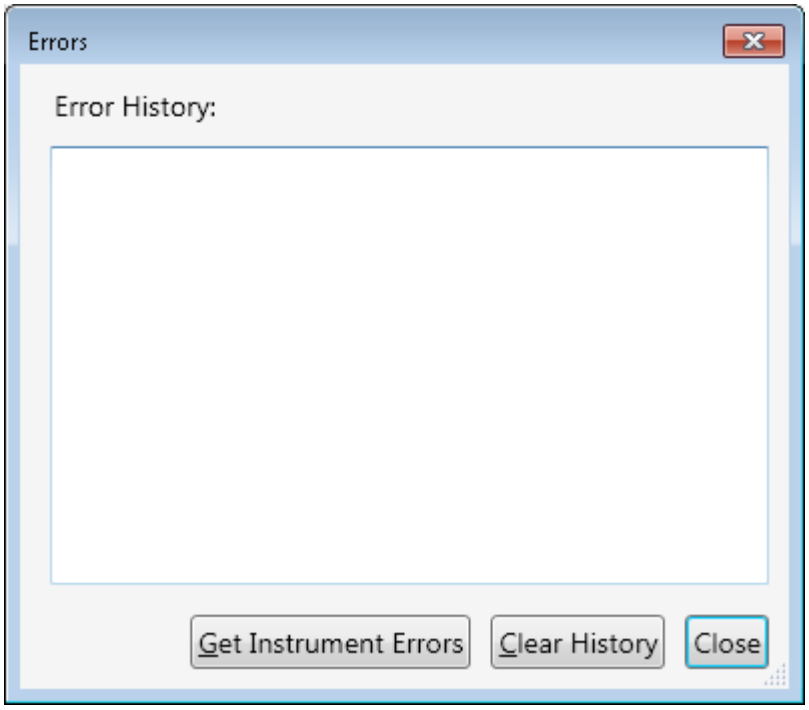


Figure 2-6: Errors Window

It has the following buttons:

- Get Instrument Errors

  Queries the instruments error queue and displays the errors, if any.
- Clear History

  Clears the error history.
- Close

  Exits the window.

2.6 Clock Tab

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

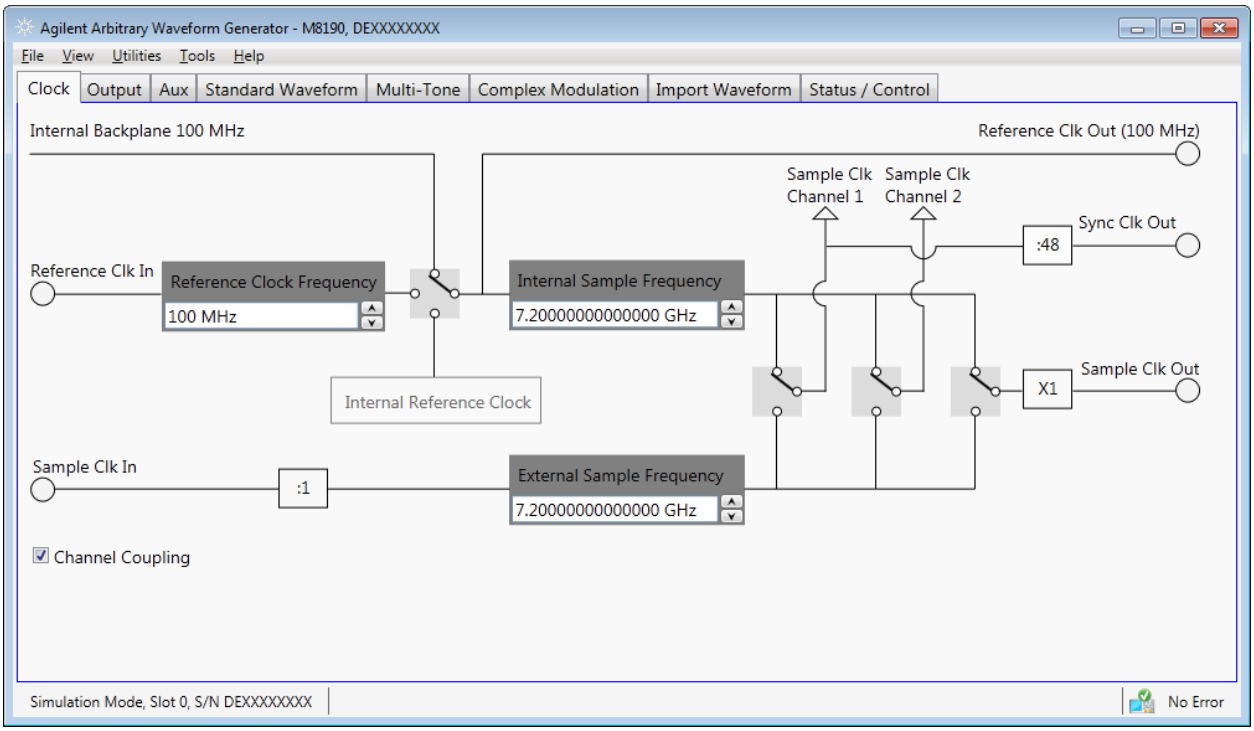


Figure 2-7: Clock Tab

- Reference clock selection switch

  This switch selects between the different reference clock sources.

  - Internal: Reference from internal oscillator
  - Internal Backplane 100 MHz: Reference from AXI Backplane
  - External: Reference from Ref Clock In

- Sample clock selection switch for channel 1 and channel 2

  These switches select between internal and external sample clock for channel 1 and channel 2.

- Sample clock output configuration switch

  This switch selects which sample clock - either internal or external - is present at the sample clock output.

- 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. For sample frequencies below 1 GHz the frequency at the Sample Clk In must be a multiple of the desired sample frequency (see command *[:SOURce]:FREQuency:RASTer:EXTernal[?]*). The ratio is shown on the divider symbol.

- External reference frequency

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

- Channel Coupling

  Use this check box to select the coupling mode of the two channels. The channels can operate in coupled or uncoupled mode.

2.7 Output Tab

Use this tab to configure the outputs (Channel 1 and Channel 2) of the M8190A module.

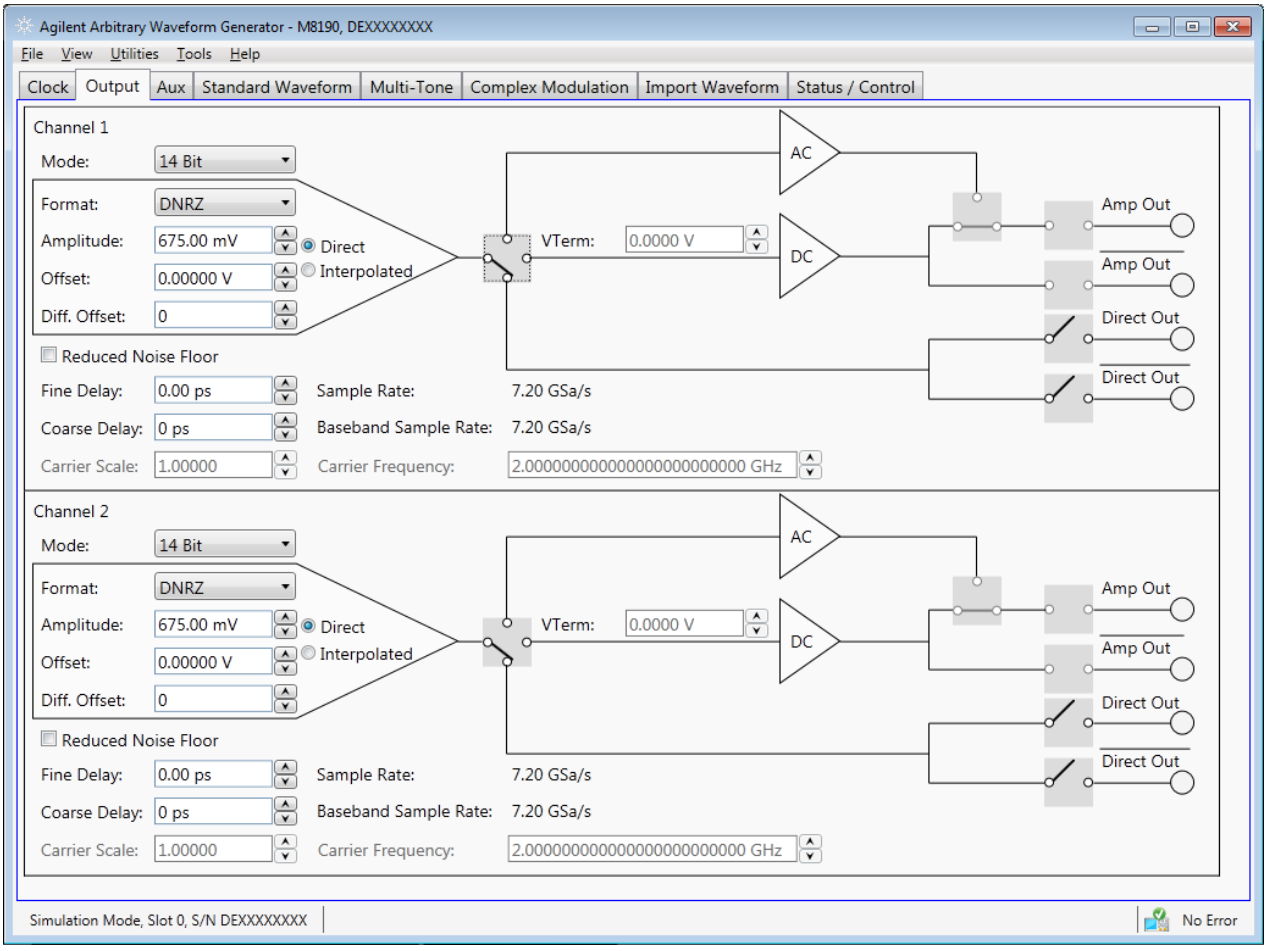


Figure 2-8: Output Tab

The two channels have the following input fields.

- Mode

  Selects the waveform output modes. Available selections depend on the selection made by the direct/interpolated radio buttons. Direct mode: 12 bit, 14 bit. Interpolated mode: INTX3/12/24/48.

  - WSPeed: Speed mode, 12 bit DAC resolution
  - WPRecision: Precision mode, 14 bit DAC resolution
  - INTX3: Interpolation x3 mode
  - INTX12: Interpolation x12 mode
  - INTX24: Interpolation x24 mode
  - INTX48: Interpolation x48 mode

  The interpolated modes INTX3, INTX12, INTX24 and INTX48 are only available when the DUC option is installed.

- DAC format mode

  Selects among the *DAC Format Modes*. Internally two DACs are used (A, B). Each usually contributes to signal 50% of a sample period.

  - RZ: DAC A sends value for whole sample period.
  - DNRZ: DAC A and B send same value.
  - NRZ: DAC A sends value for whole sample period.
  - DOUBlet: DAC A sends +Value, DAC B sends –Value

- Direct/Interpolated radio buttons

  Switch between interpolated and direct modes. The interpolated modes INTX3, INTX12, INTX24 and INTX48 are only available when the DUC option is installed.

- Output path selection switch

  Selects among the three output paths. Depending on this setting, the input fields for offset and termination voltage are applicable or not.

  - DAC: Direct DAC output
  - DC: Amplified differential output
  - AC: Single ended AC coupled output with up to 10 dBm output level

- Sample Rate

  Displays the sample rate depending on the clock source (external or internal).

- Baseband Sample Rate

  Displays baseband sample rate for interpolated mode i.e. sample rate divided by the interpolation factor (x3, x12, x24 or x48).

- Amplitude

  Specifies the amplitude of the output signal.
- Offset

  Specifies the offset of the output signal.
- Differential offset

  Specifies an offset adjustment to compensate for small offset differences between normal and complement output (see *Differential Offset*).
- Reduced Noise Floor

  Sets the "Reduced Noise Floor" feature.
- VTerm

  Specifies the termination voltage when the DC output path is selected.
- Fine Delay

  Specifies the fine delay portion of the *variable delay* (see available *Range*).
- Coarse Delay

  Specifies the coarse delay portion of the *variable delay*. See available *Range* and *Resolution*.
- Carrier Scale

  Sets the carrier amplitude scale for interpolated modes.
- Carrier Frequency

  Set the carrier frequency for interpolated modes.
- Output enable switches

  If set to closed position, the generated signal is present at the output.

2.8 Aux Tab

Use this tab to configure the external trigger and event inputs and the marker outputs of the M8190A module.

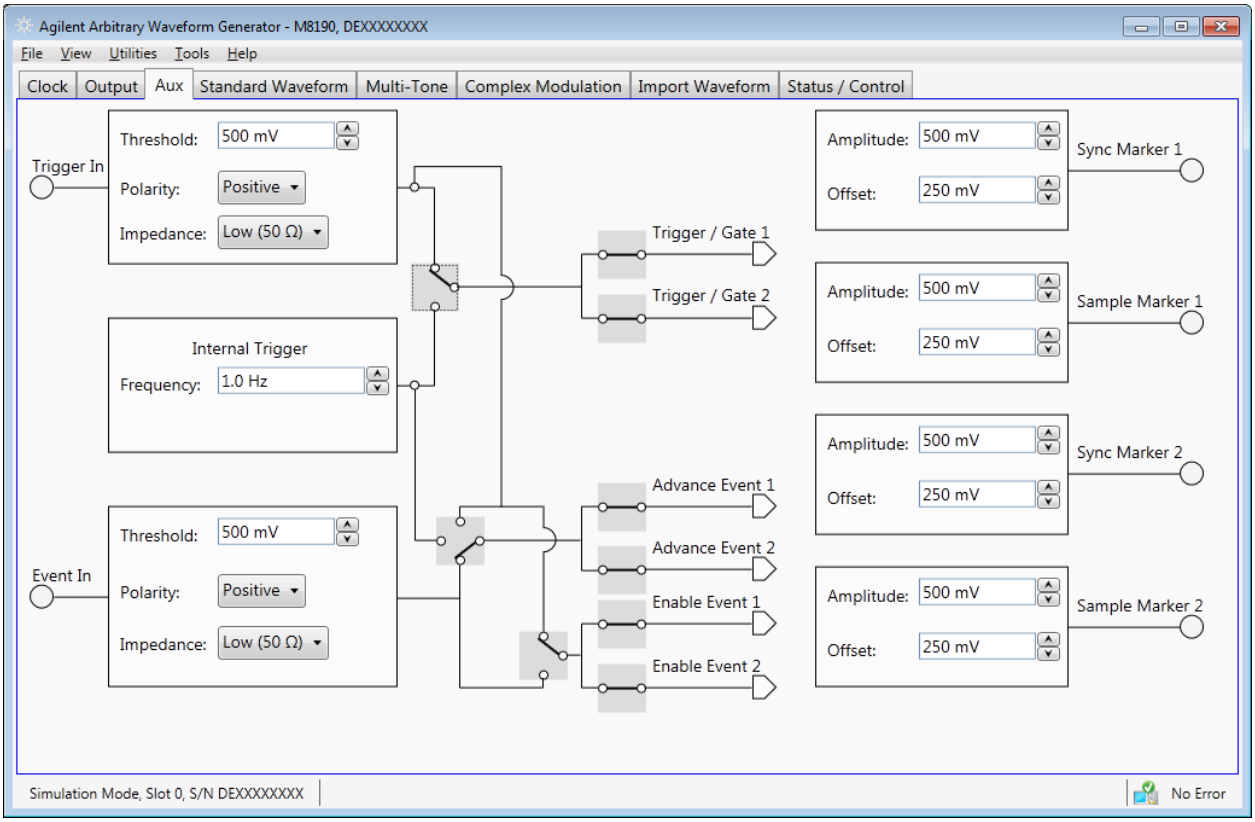


Figure 2-9: Aux Tab

This tab has the following input fields.

- Threshold for trigger and event input

  Sets threshold level.

- Polarity for trigger and event input

  Sets the polarity for the input.

  Positive – rising edge

  Negative – falling edge

  Either - both

- Impedance for trigger and event input

  Sets the impedance of the input.

  Low – 50 Ohm

  High – 1k Ohm

- Internal trigger frequency

  Sets the frequency of the internal trigger generator.

- Trigger /Gate selection switch

  Selects between external trigger input and internal trigger generator.

- Enable Event selection switch

  Selects the source for the enable event, either trigger input or event input.

- Advancement Event selection switch

  Selects the source for the advancement event, either trigger input, event input or internal trigger generator.

- Trigger /Gate hardware input switch

  Enable or disable the trigger hardware input for channel 1 and channel 2.

- Enable Event hardware input switch

  Enable or disable the event hardware input for channel 1 and channel 2.

- Advancement Event hardware input switch

  Enable or disable the advancement event hardware input for channel 1 and channel 2.

- Amplitude for marker outputs

  Specifies the amplitude of the marker output signal.

- Offset for marker outputs

  Specifies the offset of the marker output signal.

2.9 Standard Waveform Tab

Use this tab to create a variety of standard waveform types. It provides the controls which allow the complete definition of signal generation parameters for the following waveform shapes:

- Sinusoidal
- Square with linear transitions
- Square with cosine-shaped transitions
- Triangle
- Sinc (Sin x/x)
- Bandwidth-limited Gaussian noise

The standard waveform tab allows you to generate signals for both direct and up-converter (IQ) modes. It also provides a graphic waveform preview functionality, which can be used to validate created signals before sending them to the instrument or can also be stored in a file for later use. The application takes care of handling the requirements and limits of the target hardware in aspects such as maximum and minimum record lengths and sampling rate and record length granularity. As a result, the signals designed in this tab will be always feasible to be generated by the instrument and free of distortions such as wrap-around or timing artifacts, even if the signal is generated in looped mode.

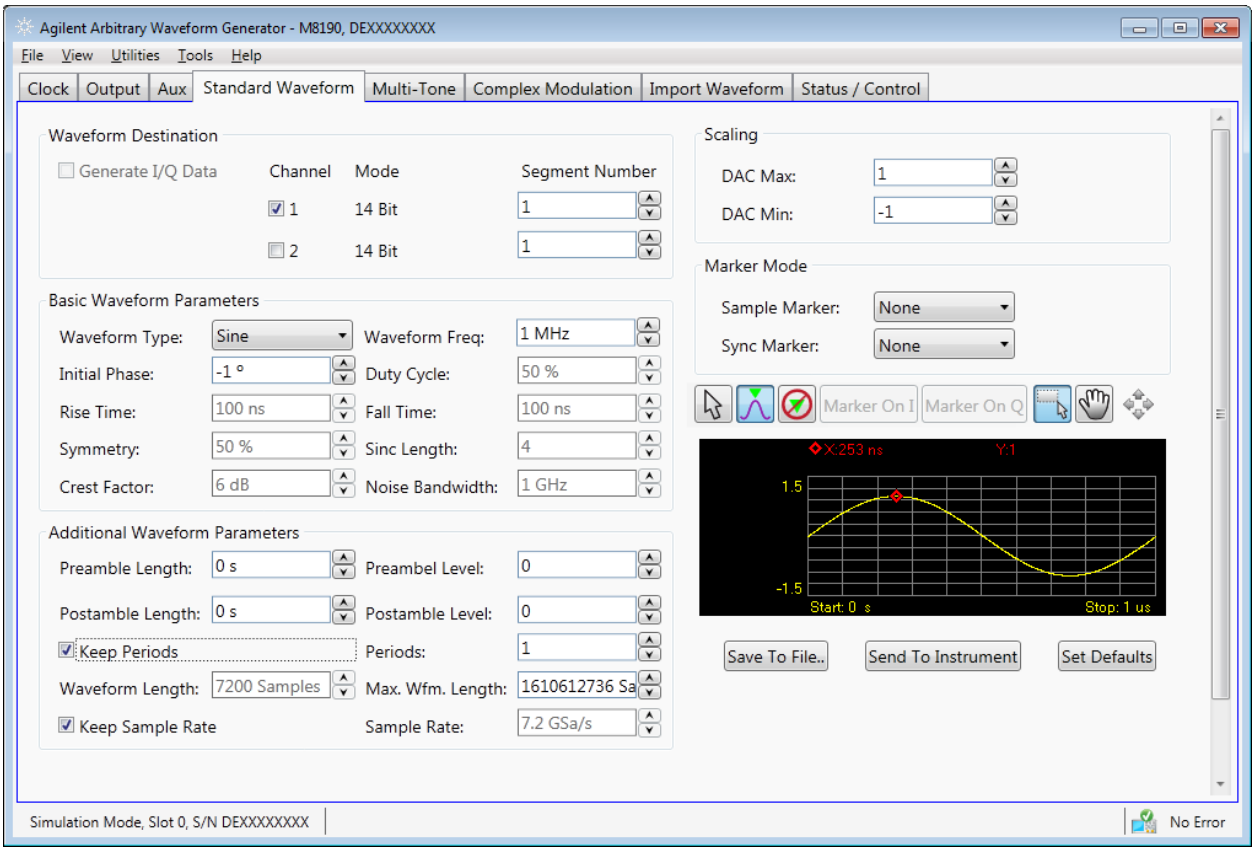


Figure 2-10: Standard Waveform Tab

This tab has the following controls:

Waveform Destination Section

- Generate I/Q Data

  Always checked while in up-converter modes. While in direct mode, it will assign the I (In-phase) component to Channel 1 and the Q (Quadrature) component to Channel 2.

- Channel

  Independent checkboxes allow the definition of standard waveforms for Channel 1, Channel 2, or both. One of the boxes will be always checked and checking both is only possible when both channels are coupled (see *Channel Coupling* control in Clock Tab).

- Segment Number

  Target segment for each channel can be defined independently.

Basic Waveform Parameters Section

- Waveform Type:

  The following waveform types are available:

  - Sine: Sinusoidal waveform. Frequency and Initial Phase parameters can be defined for this waveform type using the corresponding controls. If the Generate I/Q checkbox is checked, two sinewaves with a 90º phase difference will be assigned to the I and Q components.
  - Square_Linear: Square signal with linear transitions. Frequency, Rise Time, Fall Time, Duty Cycle, and Initial Phase parameters can be defined for this waveform type using the corresponding controls.
  - Square_Cos: Square signal with cosine shaped transitions. Frequency, Rise Time, Fall Time, Duty Cycle, and Initial Phase parameters can be defined for this waveform type using the corresponding controls.
  - Triangle: Triangular waveform with linear transitions. Frequency, Symmetry, and Initial Phase parameters can be defined for this waveform type using the corresponding controls.
  - Sinc: Sin x/x waveform. Frequency, Symmetry, Sinc Length, and Initial Phase parameters can be defined for this waveform type using the corresponding controls.
  - Noise: Gaussian noise with limited bandwidth. Frequency, Crest Factor, and Noise Bandwidth parameters can be defined for this waveform type using the corresponding controls. If the Generate I/Q checkbox is checked, two uncorrelated noise waveforms will be assigned to the I and Q components.

- Waveform Frequency

  Repetition rate for one cycle of the standard waveform. It is always a positive number except when Signal Type is set to Sine and the Generate I/Q Data checkbox is checked. In this case, frequency may be negative so the resulting SSB (Single-Side Band) will be located over or below the carrier frequency.

- Initial Phase

  The phase within a normalized cycle of the standard waveform for the first sample in the segment.

- Duty Cycle

  The relative width as a percentage of the mark and the space sections of square waves.

- Rise Time

  The transition time (10%-90%) for the rising edge in square waveforms.

- Fall Time

  The transition time (10%-90%) for the falling edge in square waveforms.

- Symmetry

  For both triangular and sinc waveforms, it marks the location as a percentage of the positive highest peak within a period of the basic signal.

- Sinc Length

  The number of zero crossings in a single period for the sinc waveform type.

- Crest Factor

  The peak-to-average power ratio in dBs for Noise samples before low-pass filtering. Actual crest factor in the final signal after filtering will be higher.

- Noise Bandwidth

  Baseband noise bandwidth for Noise waveforms. For IQ modes, noise bandwidth around the carrier frequency will be twice this parameter.

Additional Waveform Parameters Section

- Preamble Length

  The duration of a DC section before the defined Standard waveform starts.

- Preamble Level

  The level for the DC section before the defined Standard waveform starts. Acceptable range for this parameter is -1/+1, being the full dynamic range of the instrument' DAC.

- Postamble Length

  The duration of a DC section after the defined Standard waveform stops.

- Postamble Level

  The level for the DC section after the defined Standard waveform stops. Acceptable range for this parameter is -1/+1, being the full dynamic range of the instrument' DAC.

- Keep Periods

  This checkbox is only available when "Keep Sample Rate" is selected. When this option is selected, the waveform calculation algorithm preserves the user-defined number of periods.

- Periods

  The number of repetition of single periods of the standard waveform within the target segment. This parameter is set automatically when Frequency is changed and preamble and postamble lengths are set to zero in order to

obtain the best timing accuracy and meet the record length granularity requirements.

- Waveform Length

  The length in samples of the resulting segment. It may be set within acceptable limits and it may be calculated automatically to properly implement other signal and instrument parameters such as sampling rate.

- Max. Wfm. Length

  Maximum waveform length must be used to force the resulting waveform to be shorter or equal that a user-set limit.

- Keep Sample Rate

  This check box preserves the sampling rate to a user-defined value no matter how any other signal parameters may be defined. Keeping the sampling rate to a fixed value may be necessary when multiple waveforms are created to be used in a sequence or scenario.

- Sample Rate

  Final DAC' conversion rate for the resulting signal. It may be set by the user or automatically calculated depending on other signal parameters.

Scaling Section

- DAC Max

  Standard waveforms may occupy a limited range of the DAC' full scale. This parameter sets the maximum level. If set to a lower level than DAC Min, this will be automatically set to the same level. Acceptable range for this parameter is -1/+1, being the full dynamic range of the instrument' DAC.

- DAC Min

  Standard waveforms may occupy a limited range of the DAC' full scale. This parameter sets the minimum level. If set to a higher level than DAC Max, this will be automatically set to the same level. Acceptable range for this parameter is -1/+1, being the full dynamic range of the instrument' DAC.

Marker Mode Section

- Sample Marker

  Sample marker signaling the beginning of each segment may be activated (Segment selection) and deactivated (None selection).

- Sync Marker

  Sync marker signaling the beginning of each segment may be activated (Segment selection) and deactivated (None selection).

Preview Section

- Waveform Preview Toolbar

  The waveform preview toolbar includes the icons to preview the waveform. The following icons are available:

	Uses the mouse to control the marker. The respective position of marker at X and Y axis are displayed on the top of waveform.
	Takes the marker to the peak position
	Sets the marker on the I data part of the waveform
	Sets the marker on the Q data part of the waveform
	Turns off the marker
	Provides zoom functionality. Use the mouse pointer to select the area on waveform that you want to zoom. Once done, you can click Auto scale icon to zoom out the waveform.
	Uses the mouse pointer to move the waveform around. You can also use the pan tool when the waveform is zoomed in.
	Auto scale the waveform

- Save To File...

  Signals can be stored in files in whether BIN (for non IQ modes) or IQBIN (for IQ modes) formats. These files may be reused within the Import Waveform tab.

- Send To Instrument

  Signal will be transferred to the selected segments of the selected channels. The previous running status for the target instrument will be preserved but sampling rate may be modified depending on the waveform requirements.

- Set Default

  All the standard waveform parameters are set automatically to their corresponding default values.

2.10 Multi-Tone Tab

Use this tab to create signals made-up of multiple tones, either equally or arbitrarily spaced. It also allows for the definition of a frequency interval without tones (or notch) for NPR (Noise Power Ratio) testing. Amplitudes and phases of the individual tones can be corrected through correction factor files defined by the user. The Multi-Tone tab allows you to generate signals for both direct and up-converter (IQ) modes. It also provides a graphic waveform preview functionality, which can be used to validate the location and amplitudes of the tones in the signal before sending it to the instrument or be stored in a file for later use. The signal's crest factor or Peak-to-Average Power Ratio (PAPR) is also shown. The application handles requirements and limits of the target hardware in aspects such as maximum and minimum record lengths, sampling rate, and record length granularity. As a result, generation of signals designed in this tab will always be feasible through the instrument, and they will be free of distortions such as wrap-around or timing artifacts, even if they are generated in looped mode.

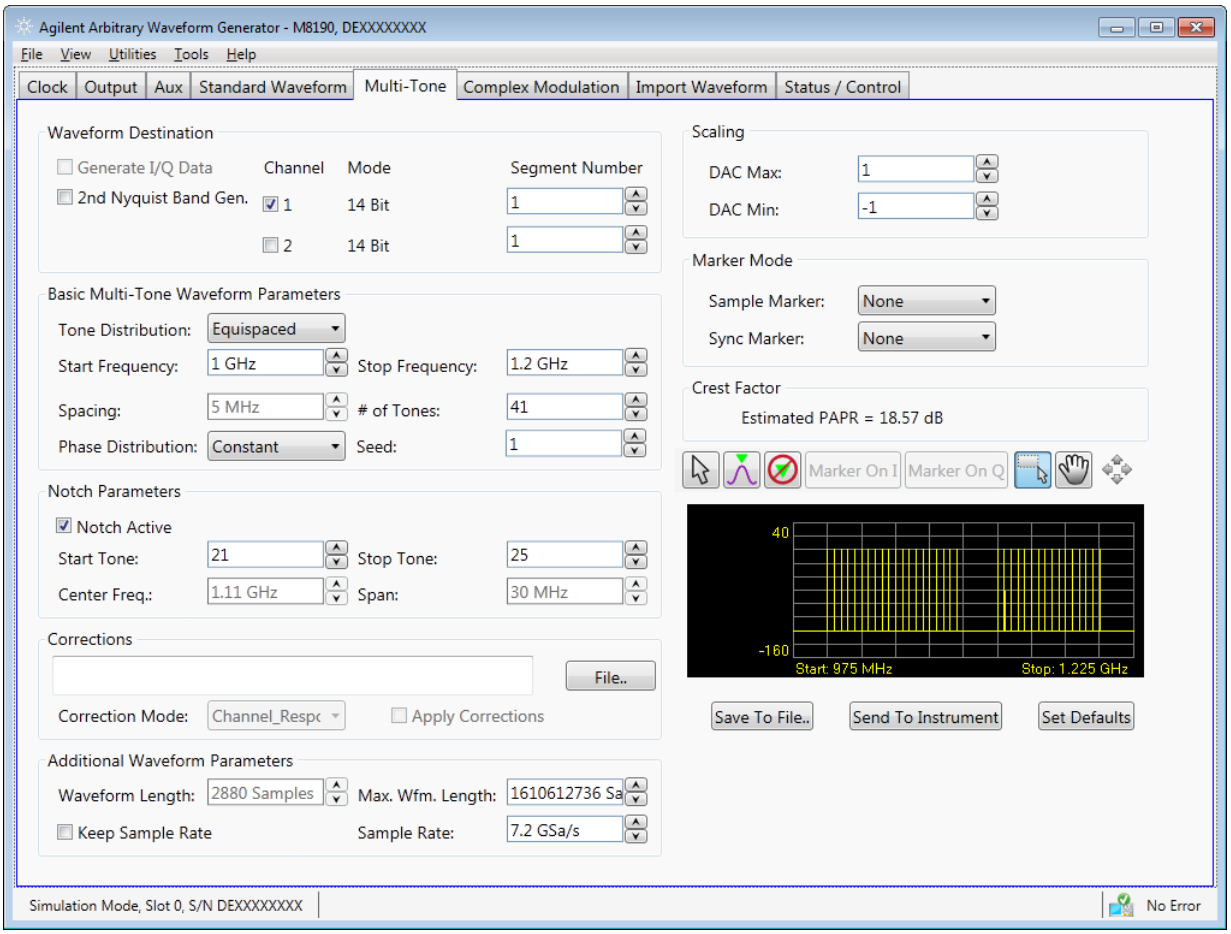


Figure 2-11: Multi-Tone Tab

There are two basic operation modes for the definition of equally spaced or arbitrarily distributed tones respectively. The selection between the two modes is made through the "Tone Distribution" drop-down list. This control affects the contents of the "Basic Multi-Tone Waveform Parameters" section of the user interface and the presence of the "Notch Parameter" section, which only makes sense in case of equally spaced tones. However, controls in the other control groups are valid and operative for both operating modes. Equally spaced tones are defined on the basis of their common parameters such as start and stop frequencies, and tone spacing or number of tones or both. Arbitrarily distributed tones are defined through a table. In order to simplify the creation of complex scenarios, the tones defined in the equally spaced mode are loaded into the tone table every time the user switches to the arbitrary mode and the tone table is empty. In this way, any number of tones may be easily defined in the equally spaced mode, and then the resulting table may be edited for frequency, amplitude, or phase for each individual tone. Tones may also be deleted or added.

This tab has the following controls:

Waveform Destination Section

- Generate I/Q Data

  It is always selected while in up-converter modes. If selected in direct mode, it will assign the I (In-phase) component to Channel 1 and the Q (Quadrature) component to Channel 2.

- 2nd Nyquist Band Gen.

  This checkbox must be selected for generation of signals in the second Nyquist band (FS/2 – FS). Its effect depends on the signal generation mode. For direct signal generation, it only has an effect if multi-tone correction is active. In this case, amplitude and phase corrections will be applied to each tone according to the location of the corresponding image in the second Nyquist band. For up-conversion modes, the location of each tone is reversed in the frequency domain as well as amplitude and phase corrections.

- Channel

  Independent checkboxes allow the definition of multi-tone waveforms for Channel 1, Channel 2, or both. One of the boxes will be always checked and checking both is only possible when both channels are coupled (see *Channel Coupling* control in Clock Tab).

- Segment Number

  Target segment for each channel can be defined independently.

Corrections Section

- File...

  Open a correction file selection dialog box. Default file extensions match the File Format selection. The name of the successfully loaded correction factors file is shown in the field located at the left of this button. The accepted format for correction files may be found in the *Correction File Format* section.

- Correction Mode

  Correction files may contain data relative to the correction (magnitude and phase) to be applied to the multi-tone signal (Correction_Factors) or it may represent the system response to be corrected for flatness and linear phase (Channel_Response).

- Apply Corrections

  This checkbox activates the application of the correction factors.

Additional Waveform Parameters Section

- Waveform Length

  It is indicator only. The length is in samples of the resulting segment.

- Max. Wfm. Length

  Maximum waveform length must be used to force the resulting waveform to be shorter or equal to the limit set by the user.

- Keep Sample Rate

  This check box preserves the sampling rate to a user-defined value irrespective of the manner in which other signal parameters may be defined. Keeping the sampling rate to a fixed value may be necessary when multiple waveforms are created for usage in a sequence or scenario.

- Sample Rate

  Final DAC conversion rate for the resulting signal. It may be set by the user or automatically calculated depending on other signal parameters.

Scaling Section

- DAC Max

  Multi-tone waveforms may occupy a limited range of the DAC's full scale. This parameter sets the maximum level. If set to a lower level than DAC Min, this will be automatically set to the same level. Acceptable range for this parameter is -1/+1, being the full dynamic range of the instrument's DAC.

- DAC Min

  Multi-tone waveforms may occupy a limited range of the DAC's full scale. This parameter sets the minimum level. If set to a higher level than DAC Max, this will be automatically set to the same level. Acceptable range for this parameter is -1/+1, being the full dynamic range of the instrument's DAC.

Marker Section

- Sample Marker

  Sample marker signaling the beginning of each segment may be activated (Segment selection) and deactivated (None selection).

- Sync Marker

  Sync marker signaling the beginning of each segment may be activated (Segment selection) and deactivated (None selection).

Crest Factor Section

- It is an indicator only.

  It shows the estimated PAPR for the current waveform in dB. Although the definition of the PAPR parameter is always the ratio between the peak and the average power for a signal, results change depending on the working mode. For up-converter modes, the result reflects the PAPR of the envelope of the resulting signal while for direct generation it reflects the overall signal. The difference between the former and the latter values is close to +3dBs in most cases.

Preview Section

- Multi-Tone Preview Toolbar

  The waveform preview toolbar includes the icons that provide different functionality to preview the waveform. For details, see *Preview Section Waveform* Preview Toolbar.

Compilation and Panel Control Section

- Save To File...

  Signals can be stored in files either in BIN (for non IQ modes) or IQBIN (for IQ modes) formats. These files may be reused within the Import Waveform tab. This button is not active if File Format is "SigStudioEncrypted".

- Send To Instrument

  Signal will be transferred to the selected segments of the selected channels. The previous running status for the target instrument will be

preserved but sampling rate may be modified depending on the waveform requirements.

- Set Default

  All the multi-tone waveform parameters are set automatically to their corresponding default values. Entries in the Arbitrary Tone table are not modified by this button.

Two control sections show-up for equally spaced tone definition ("Equispaced" selected in the Tone Distribution drop-down list): "Basic Multi-Tone Waveform Parameters" and "Notch Parameters".

Basic Multi-Tone Waveform Parameters Section

- Start Frequency

  It is the frequency of the first tone. If it is set to a value higher than the one in the Stop Frequency field, this is changed back to the previous Start Frequency.

- Stop Frequency

  It is the frequency of the last tone. If it is set to a value lower than the one in the Stop Frequency field, this is changed back to the previous Stop Frequency.

- Spacing

  It is an indicator only.

  Spacing = (Stop Frequency – Start Frequency)/(# of Tones – 1).

- # of Tones

  It is the total number of tones in the multi-tone signal including the ones in the notch, if any.

- Phase Distribution

  Phase for each tone can be set in the three different modes: constant, random, and parabolic. While constant phase multi-tone signals show a high crest factor, a random phase distribution results in a much lower value for this parameter while a parabolic distribution results in a close to optimal (or minimum) crest factor.

- Seed

  This parameter is associated to the random phase distribution and allows generating the same or different random sequences for the phases of each tone. It is also useful to look for a distribution resulting in a desired crest factor value.

Notch Parameters Section

- Notch Active

  This check box activates or deactivates the generation of a notch in the equally spaced multi-tone signal.

- Start Tone

  It is the index of the first tone to be removed in a notch. Acceptable indexes start with 1.

- Stop Tone

  It is the index of the last tone to be removed in a notch. Acceptable indexes start with 1.

- Center Frequency

  It is an indicator only. The central frequency for the notch is computed and shown in this field.

- Span

  It is an indicator only. The tone-free frequency span for the notch is computed and shown in this field.

Arbitrary Tones Section

Alternatively, an edition table shows-up for arbitrarily spaced tones definition ("Arbitrary" selected in the Tone Distribution drop-down list). When not previously edited (or empty), the table is automatically loaded with the parameters of the tones defined in the equally spaced tone sections. This allows for easy edition of individual tones or the creation of multiple notches, or both. Parameters for each tone include its frequency (in Hz), its relative amplitude (in dB), and phase (in degrees). Entries in the table may be added, edited, and deleted. Entries in the table may be also sorted in ascending or descending order of any parameter by clicking in the corresponding field name.

Addition of a new entry in the table must be done by editing the empty edition field located at the bottom of the table. Deletion of any number of entries can be performed by selecting the ones to be deleted and then hitting the <Del> key in the keyboard. Meaningful numeric values must be typed into the edition fields. Otherwise an error condition is triggered. While a valid frequency entry must be always entered, any of the amplitude and phase edition fields may kept empty so they take the default values (0.0 dB for Amplitude and 0.0 degrees for Phase).

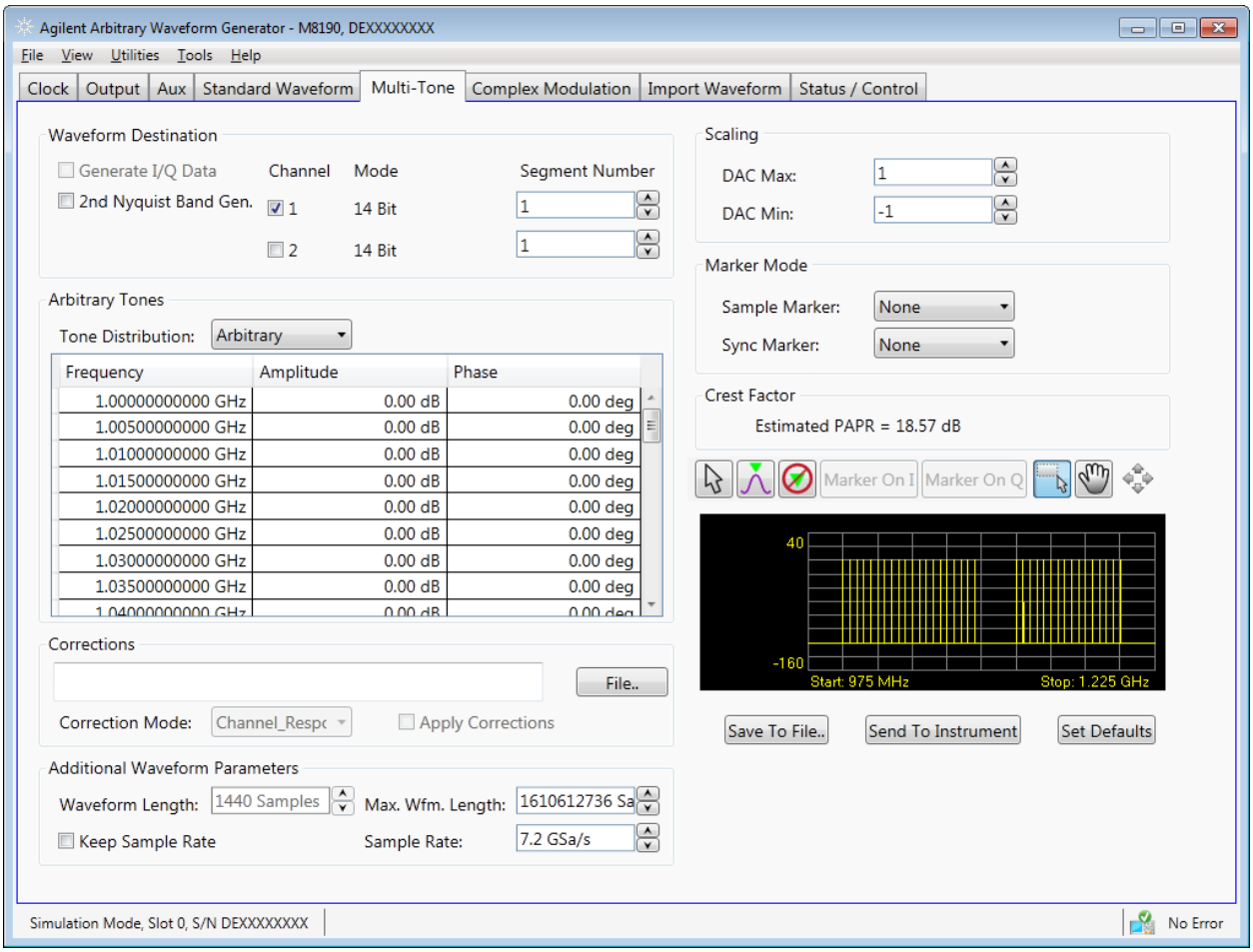


Figure 2-12: Multi-Tone Tab, Arbitrary Tone Distribution

2.11 Complex Modulation Tab

Use this tab to create baseband and IF/RF digitally modulated signals. User-defined corrections may be applied to signals to compensate for (or emulate) instrument, interconnections and channel linear distortions. The complex modulation tab allows you to generate signals for both direct conversion and external/internal up-converter (IQ) modes. It directly supports a large variety of single-carrier modulation schemes. This is a list of the currently supported standards, modulation orders and modulation parameters:

- ASK (Amplitude Shift Keying): Modulation Index (0%-100%).
- PSK (Phase Shift Keying): BPSK, QPSK, π/4-DQPSK, Offset-QPSK (OQPSK), 8PSK, and 3 π /8 8PSK (EDGE).
- QAM (Quadrature Amplitude Modulation): 8QAM, 16QAM, 32QAM, 64QAM, 128QAM, 256QAM, 512QAM, AND 1024QAM.
- MSK (Minimum Shift Keying)
- APSK (Amplitude-Phase Shift Keying): 16APSK and 32 APSK. R2/R1 and R3/R1 can be set by the user to any desired value.
- STAR: STAR16 and STAR32. The R2/R1 parameter may be set for the STAR16 modulation scheme.
- VSB (Vestigial Side Band): 8VSB and 16VSB.
- FSK (Frequency Shift Keying): 2FSK, 4FSK, 8FSK, and 16FSK. Peak deviation frequency may be set by the user to any desired value.
- Custom: Users may define arbitrary constellations through simple ASCII files that may be read by the SFP application. Modulations with offset (Q delayed by half a symbol time) and rotating constellations may be also defined. (Refer to the section *Custom Modulation File Format*)

Baseband filter type, characteristics, and different data options may be selected by the user. The panel provides a constellation preview functionality, which can be used to validate the selected modulation scheme and the corresponding modulation parameters. The application takes care of handling the requirements and limits of the target hardware with respect to maximum and minimum record lengths, sampling rate, and record length granularity. As a result, generation of the signals designed in this tab will always be feasible by the instrument and free of distortions such as wrap-around or timing artifacts at any signal domain (time, frequency, and modulation), even if the signal is generated in looped mode.

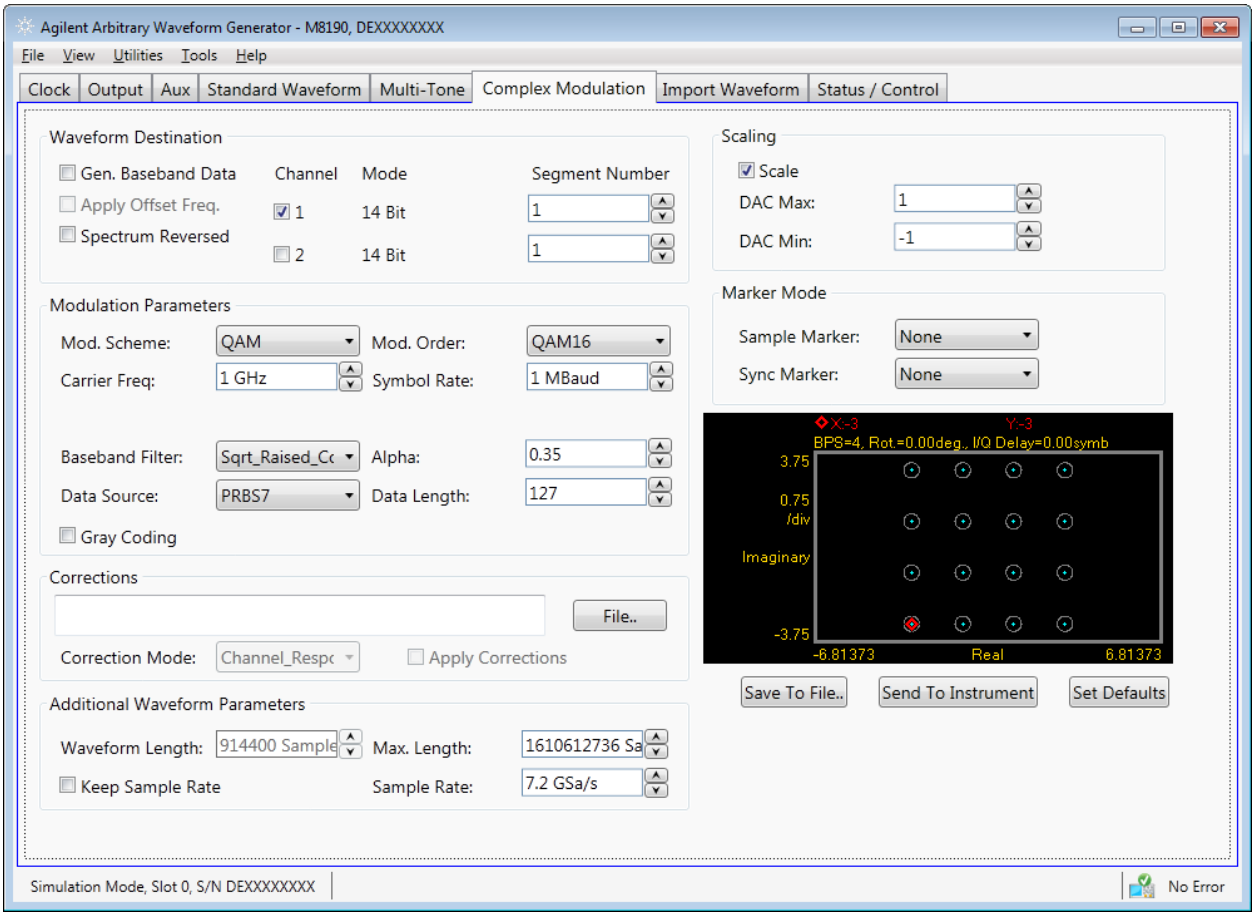


Figure 2-13: Complex Modulation

Only relevant parameters and edition fields are shown in the GUI at any time depending on the selected generation mode (Direct/Up-converter) and modulation scheme.

Waveform Destination Section

- Generate Baseband Data

  It is always seleced while in up-converter modes. If checked while in direct mode, it will assign the I (In-phase) component to Channel 1 and the Q (Quadrature) component to Channel 2.

- Apply Offset Freq.

  This checkbox is only active for up-converter modes and it applies a frequency shift to the signal according to the 'Offset Freq.' edition field. Frequency shift, unlike carrier frequency, may be positive or negative.

- Spectrum Reversed

  This checkbox must be selected for generation of signals in the second

Nyquist band (FS/2 – FS). Its effect is the reversion of the fundamental signal (in the 1st Nyquist Band) in the frequency domain. It also reverses the effect of any correction so correction factors obtained for the second Nyquist band will be applied appropriately.

- Channel

  Independent checkboxes allow the definition of waveforms for Channel 1, Channel 2, or both. One of the boxes will be always checked and checking both is only possible when both channels are coupled (see *Channel Coupling* control in Clock Tab).

- Segment Number

  Target segment for each channel can be defined independently.

Modulation Parameters Section

- Mod. Scheme

  This drop-down list selects the different modulation scheme categories that are supported (see list above).

- Mod. Type/Mod. Order

  This drop-down list selects the different modulation orders or modulation scheme sub-types for the selected modulation scheme category.

- Carrier Freq. / Offset Freq.

  The purpose and labeling of this edition field changes depending on the generation mode. For direct conversion modes, it handles the carrier frequency while for up-converter modes it deals with the offset frequency (see the *Apply Offset Freq.* control). Units in both cases are in Hz.

- Symbol Rate

  This edition field must be used to enter the signaling speed (or baud rate) for the modulated signal expressed in Bauds (1 Baud = 1 Symbol/s).

- Mod. Index(%)

  This edition field only shows up when the ASK modulation scheme is selected. It sets the modulation index as a percentage for the signal.

- R2/R1 Ratio

  This edition field only shows up when the 16APSK, 32APSK, and 16STAR modulation schemes are selected. It sets the ratio between the radius of the two inner symbol rings in the constellation.

- R3/R1 Ratio

  This edition field only shows up when the 32APSK modulation scheme is selected. It sets the ratio between the radius of the outer and the most internal symbol rings in the constellation.

- Freq. Dev.

  This edition field only shows up when the FSK modulation schemes are selected. It sets the peak frequency deviation in Hz.

- Mod. File..

  This button only shows up when 'Custom' modulation scheme is selected. It opens a file selection window where modulation definition files may be selected. If a valid file is selected, its name will show up in the text field located at the left of this button. Otherwise a "File Loading Error" message is shown.

- Baseband Filter

  This drop-down list can select different baseband filters to be applied to the baseband symbols. Choices are 'None', 'Raised_Cosine', 'Sqrt_Raised_Cosine', 'Gaussian_Dirac', 'Gaussian_Pulse', 'EDGE', and 'Half_Sine'.

- Alpha / BT

  The meaning and labeling of this edition field depends on the selected baseband filter. For "Nyquist" filters (Raised Cosine and Square Root of Raised Cosine) it is the 'Alpha' parameter (or roll-off factor) of the filter. For Gaussian filters it is the BT (Bandwidth/symbol period product) parameter. Some filter types do not require an additional filter parameter.

- Data Source

  This drop-down list allows the selection of different pseudo random binary sequences as data sources for modulation. Choices are PRBS7 (Polynomial D7+D6+1), PRBS10 (Polynomial x10+x7+1), PRBS11 (Polynomial x11+x9+1), PRBS15 (Polynomial x15+x14+1), PRBS23 (Polynomial x23+x18+1), PRBS23p (Polynomial x23+x21+x18+x15+x7+x2+1), and PRB31 (Polynomial x31+x28+1).

- Data Length

  This edition field may be used to set a given data length to be implemented by the modulated signal. This field defaults to the maximum non-repeating length of the selected PRBS. It also defaults to this value if the user types '0' (Zero). Otherwise the sequence will be truncated when the number of bits set by this control is reached. If this number is longer than the PRBS maximum length, the sequence will be re-started as many times as necessary.

- Gray Coding

  This checkbox enables gray coding for for the applicable modulation modes.

Corrections Section

- File...

  Opens a correction file selection dialog box. Default file extension is CSV (Comma-Separated Values). The name of the successfully loaded correction factors file is shown in the field located at the left of this button. The accepted format for correction files may be found in the *Correction File Format* section.

- Correction Mode

  Correction files may contain data relative to the correction (magnitude and phase) to be applied to the complex moduation signal (Correction_Factors) or it may represent the system response to be corrected for flatness and linear phase (Channel_Response).

- Apply Corrections

  This checkbox activates the application of the correction factors.

Additional Waveform Parameters Section

- Waveform Length

  It is an indictator only. The length is in samples of the resulting segment.

- Max. Length

  Maximum waveform length must be used to force the resulting waveform to be shorter or equal to a limit set by the user.

- Keep Sample Rate

  This check box preserves the sampling rate to a user-defined value irrespective of any other defined signal parameter. Keeping the sampling rate to a fixed value may be necessary when multiple waveforms are created for usage in a sequence or scenario.

- Sample Rate

  It is the final DAC conversion rate for the resulting signal. It may be set by the user or automatically calculated depending on other signal parameters.

Scaling Section

- DAC Max

  Waveforms may occupy a limited range of the DAC's full scale. This parameter sets the maximum level. If set to a lower level than DAC Min, this will be automatically set to the same level. Acceptable range for this parameter is -1/+1, being the full dynamic range of the instrument's DAC.

- DAC Min

  Waveforms may occupy a limited range of the DAC's full scale. This parameter sets the minimum level. If set to a higher level than DAC Max,

this will be automatically set to the same level. Acceptable range for this parameter is -1/+1, being the full dynamic range of the instrument's DAC.

Marker Section

- Sample Marker

  Sample marker signaling the beginning of each segment may be activated (Segment selection) and deactivated (None selection).

- Sync Marker

  Sync marker signaling the beginning of each segment may be activated (Segment selection) and deactivated (None selection).

Constellation Diagram Section

The constellation diagram section shows a graphic representation of the ideal constellation corresponding to the selected modulation scheme and modulation parameters. It also shows the location of symbols from valid modulation definition files for validation. The line above the constellation diagram shows the following modulation parameters:

- BPS (Bits Per Symbol)
- Per symbol rotation angle (in degrees)
- I/Q delay (in symbol times)

Compilation and Panel Control Section

- Save To File...

  Signals can be stored in files in whether BIN (for non IQ modes) or IQBIN (for IQ modes) formats. These files may be reused within the Import Waveform tab. This button is not active if File Format is the "SigStudioEncrypted".

- Send To Instrument

  Signal will be transferred to the selected segments of the selected channels. The previous running status for the target instrument will be preserved but sampling rate may be modified depending on the waveform requirements.

- Set Default

  All the Multi-Tone waveform parameters are set automatically to their corresponding default values. Entries in the Arbitrary Tone table are not modified by this button.

- Abort

  This button allows canceling signal calculation at any moment. It only shows up during signal compilation.

Custom Modulation File Format

A custom modulation file is an ASCII delimited file including all the information required to define a single carrier modulated signal based in quadrature (IQ) modulation. The file must be composed of a header including a series of lines with identifiers and parameters, and a list of numerical correction factors. For lines including more than one item (i.e. one identifier and one parameter), those must be separated using commas. Identifiers and parameters are not case sensitive. These are the significant fields for the header:

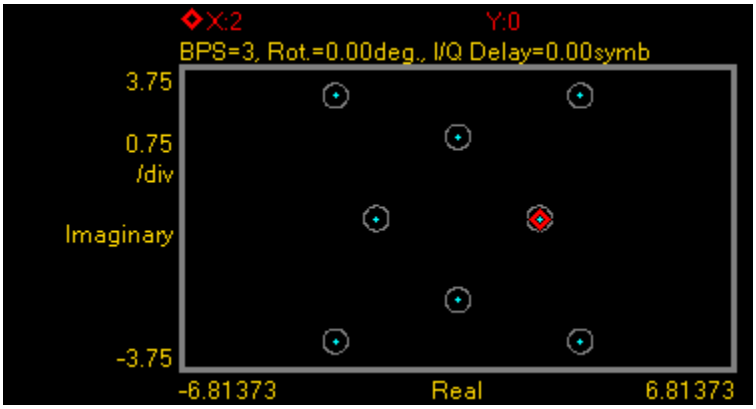
- #N: This is a mandatory field and it must be the first in the file. The N parameter is the bits per symbol parameter. 0<N<11.
- Offset: It indicates if the Q component must be delayed by half a symbol time respect to the I component. Accepted parameters are 'yes' or 'no'. This parameter is optional. It defaults to 'no' if not included in the file.
- Rotation: It sets the rotation of the constellation for each consecutive symbol in degrees. This parameter is optional. It defaults to 0.0 if not included in the file.
- RotMode: Rotation mode. Parameter may be 'cont' (continuous) or 'alt' (alternate). This parameter is optional. It defaults to 'cont' if not included in the file.
- Vsb: It indicates that vestigial side band baseband filtering must be applied. Accepted parameters are 'yes' or 'no'. This parameter is optional. It defaults to 'no' if not included in the file.

The order of the above entries is not relevant except for the '#N' field that must be placed first in the file. The symbol location section starts with a line including the 'IQ' characters (not case-sensitive). Entries in this section are made by IQ pairs separated by commas. The number of entries must be at least 2^N although additional entries will be ignored. Data to symbol mapping depends on the order of the symbols in the file so its position expressed in binary format corresponds to the binary code assigned to that symbol. Comments must start with the '//' character sequence and may use a complete line or be located at the end of any valid line (including the first line). Empty lines are also valid.

The following example illustrates a simple example of a 3 bit per symbol QAM8 modulation with a particular constellation.

#3 // MyModulationFile
Iq
// Inner symbols
2.0, 0.0
0.0, -2.0
-2.0, 0.0
0.0, 2.0
// Outer symbols
3.0, 3.0
-3.0, 3.0
-3.0, -3.0
3.0, -3.0 // Final symbol

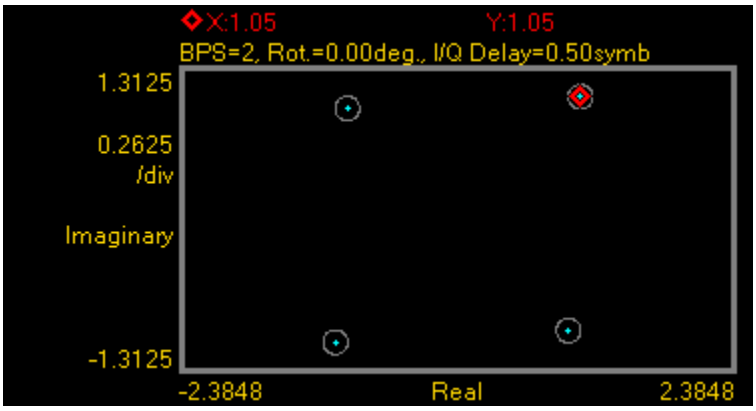
The above file does not include any unnecessary line in the header as it defines a non-rotating, non-offset modulation so default values for these fields are used instead. The resulting constellation after loading this file is shown as following:



The following example illustrates another possible use of custom modulation to define a distorted constellation. In this particular case, a O-QPSK modulation with a quadrature error (non-perpendicular I and Q axis) is defined:

#2
Offset, yes
iq
1.05, 1.05
-0.95, 0.95
-1.05, -1.05
0.95, -0.95

The above file includes a line to indicate that this is an offset modulation. The resulting constellation after loading this file is shown as following:



The following is a more complex example:

#3

Offset, no

Rotation, 10.0

RotMODE, cont

iq

1.0, 0.0

2.0, 0.0

0.0 ,1.0

0.0, 2.0

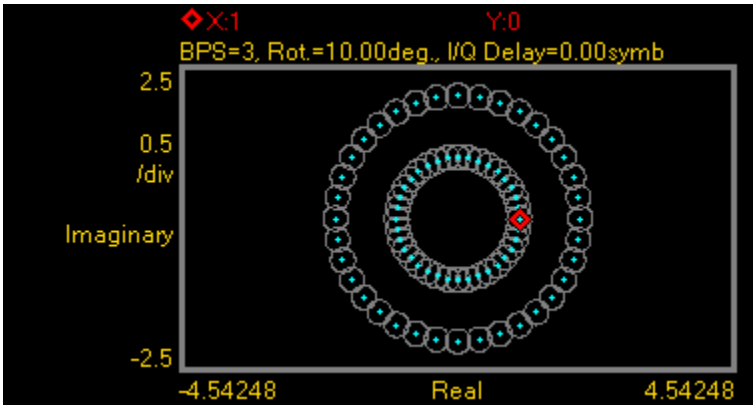
-1.0, 0.0

-2.0, 0.0

0.0, -1.0

0.0 ,-2.0

The above file is composed of a header with relevant information. In this particular case, the file contains 8 (2^3) IQ pairs. The 'IQ' characters indicate the starting point for the symbol location list composed by 8 lines with I/Q pairs separated by commas. I and Q will not be delayed ('Offset, no') and constellation will rotate by 10.0 degrees ('Rotation, 10.0') in a continuous fashion ('RotMODE, cont'). In fact, the 'Offset' and 'RotMode' fields could be removed without any effect on the final signal as these fields take the default values. The resulting constellation after loading this file is shown as following:



2.12 Import Waveform Tab

Use this tab to perform the functions such as importing, scaling, and resampling waveform files in a variety of formats for their generation by the M8190A arbitrary waveform generator. It provides the controls which allow the complete definition of signal processing parameters for the waveform file format (see *File Format*).

Depending on the file format and contents, marker information can be extracted and exported to the M8190A generator. Up to two markers can be supported and assigned to the AWG Sample and Sync markers. Also depending on the file format and contents, information regarding the original sampling rate and/or carrier frequency of the input waveforms can be extracted and re-used within the import tool. Resampling is performed so no images or aliases show up in the resampled waveform. The import tool supports the generation of signals in both direct and up-converter modes of the M8190A.

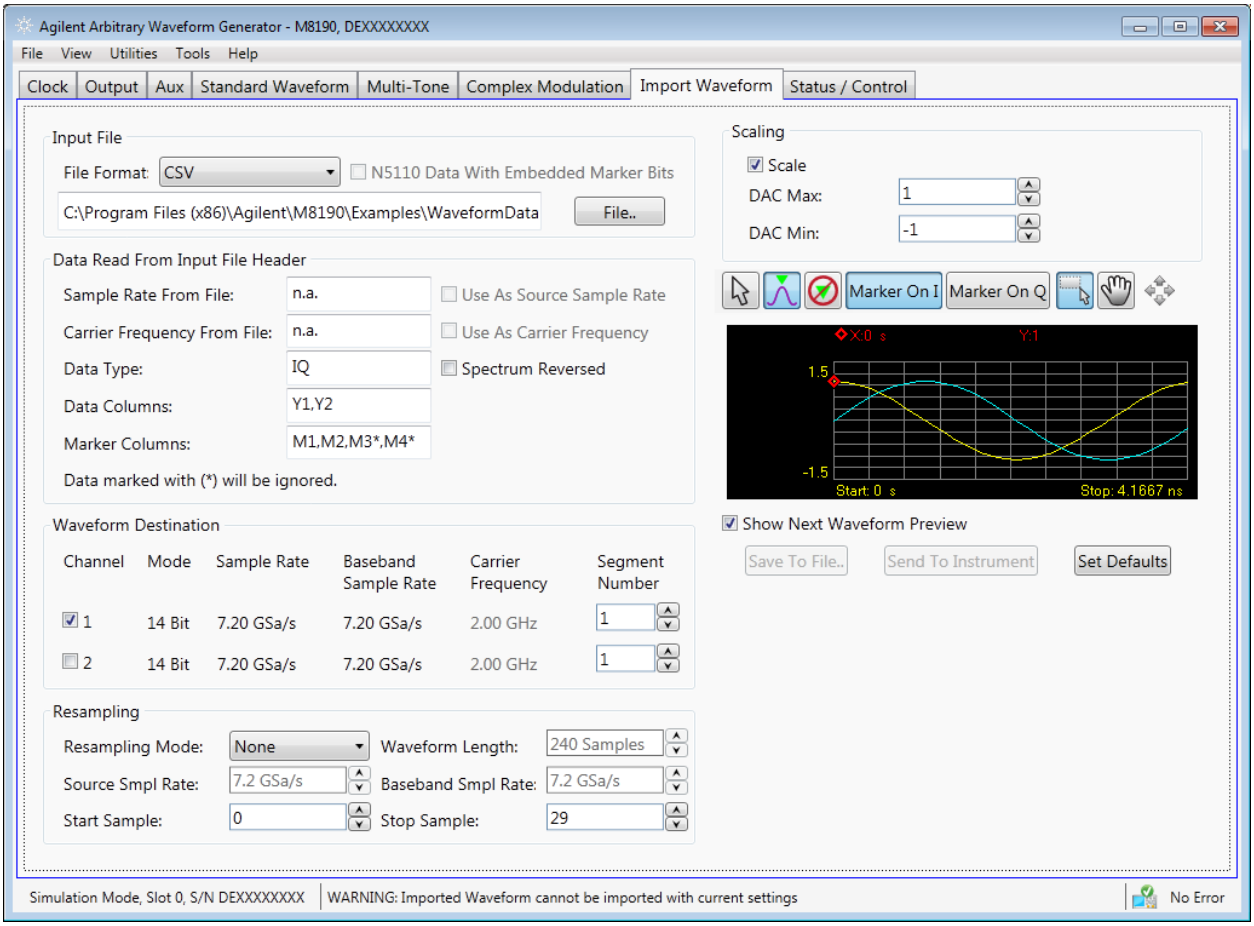


Figure 2-14: Import Waveform Tab

This tab has the following controls and indicators:

Input File Section

- File Format

  For details on the available file format, see File Format.

- N5110 Data With Embedded Marker Bits

  This checkbox is only enabled, if the File Format is BIN5110. If checked, the BIN5110 format with 14-bit data for I and Q and embedded marker bits is used. If unchecked, the BIN5110 format with 16-bit data for I and Q and no marker bits is used.

- File...

  Open a file selection dialog. Default file extensions match the File Format selection. Successful loading of a waveform updates multiple information fields through the panel reflecting the waveform settings and a graph of the waveform is shown in the preview display.

Data Read From Input File Header Section

- Sample Rate From File

  Indicator only. It shows the input waveform sample rate, if any, contained in the loaded file. If no sample rate is specified "n.a." (not available) is shown.

- Use As Source Sample Rate

  This checkbox assigns the sample rate specified in the file as the Source Sample Rate used for resampling.

- Carrier Frequency From File

  Indicator only. It shows the input waveform carrier frequency, if any, contained in the loaded file. If no carrier frequency is specified "n.a." (not available) is shown.

- Use As Carrier Frequency

  This checkbox assigns the carrier frequency specified in the file as the Carrier Frequency in the target AWG Channel. It only makes sense for up-converter (IQ) modes.

- Data Type

  This is the organization of samples within the file. It may be Single (real-only waveform) or IQ (complex waveforms).

- Spectrum Reversed

  This checkbox is only active for complex (IQ) waveforms. It results in an imported signal which is the complex conjugate of the input signal, thus

its spectrum will be reversed. It must be checked to preserve the spectrum shape and the IQ axis assignment when the M8190A' DAC is working in Doublet Mode and the desired signal sits in the second Nyquist band.

- Data Columns

  It shows the internal organization of the file regarding waveforms. It can show from one column (Y1) up to 4 (Y1, Y2, Y3*, Y4*). Although files with more than 2 columns are reported correctly, only the first two columns (Y1, Y2) are imported and processed.

- Marker Columns

  It shows the internal organization of the file regarding markers. It can show from one column (M1) up to 4 (M1, M2, M3*, M4*). Although files with more than 4 columns are reported correctly, only the first two columns (M1, M2) are imported and processed. M1 is assigned to the Sample Marker while M2 is assigned to the Sync Marker.

Waveform Destination Section

- Channel

  Independent checkboxes allow the definition of standard waveforms for Channel 1, Channel 2, or both. One of the boxes will be always checked and checking both is only possible when both channels are coupled (see *Channel Coupling* control in Clock Tab).

Resampling Section

- Resampling Mode

  It controls the way waveforms are imported and resampled. Please refer to the description of the resampling methodology in the *Appendix*. The following modes are available:

  – None: Baseband Sample Rate will be the same as the Source Sampling Rate. The output waveform will use the same number of samples as the selected portion of the input waveform. Granularity requirements will be met by repeating the basic waveform the minimum number of times so the combined length is a multiple of the granularity for the current DAC mode.

  – Timing: The time window of the input signal (Waveform Length / Sample Rate) will be used to calculate the best value for the output record length being a multiple of the granularity for the current DAC mode according to the output sampling rate defined by the user. Final output sampling rate will be slightly adjusted to accurately keep the timing of the original signal.

  - Output_SR: The user-defined output sampling rate will be used to calculate the best value for the output record length being a multiple of the granularity for the current DAC mode according to the time window of the input signal. Final time window will be slightly adjusted to keep the selected output sampling rate. This change is reflected in the Source Sampling Rate numeric entry field value.
  - Output_RL: The user-defined output Waveform Length will be used to calculate the best value for the output Sample Rate according to the time window of the input signal. Waveform Length will be adjusted to the nearest multiple of the granularity for the current DAC mode according to the time window of the input signal.
  - Zero_Padding: Output Waveform Length is calculated based on the input waveform time window and the user-defined output sampling rate. The resulting waveform length will not be, in general, a multiple of the granularity. To meet the granularity conditions, a number of zero samples are added until the combined number of samples is a multiple of the granularity. Output Sample Rate will be slightly adjusted to keep the input waveform time window.
  - Truncate: Output Waveform Length is calculated based on the input waveform time window and the user-defined output sampling rate. The resulting waveform length will not be, in general, a multiple of the granularity. To meet the granularity conditions, a number of samples is removed until the resulting number of samples is a multiple of the granularity. Output Sample Rate will be slightly adjusted to keep the input waveform time window.
  - Repeat: Output Waveform Length is calculated based on the input waveform time window and the user-defined output sampling rate. The resulting waveform length will not be, in general, a multiple of the granularity. To meet the granularity conditions, the base waveform is repeated the minimum number of times so the overall number of samples is a multiple of the granularity. Output Sample Rate will be slightly adjusted to keep the input waveform time window. The Waveform Length field will show the length of the combined waveform.

- Waveform Length

  It shows the number of samples of the resampled output waveform. It can be set when Resampling Mode is Output_RL. Otherwise, this field is an indicator. This field is not active if File Format is the "SigStudioEncrypted".

- Source Sample Rate

  The speed at which samples in the input waveform are sampled. It can be set by typing a valid value unless the "Use As Source Sample Rate" checkbox is checked. In this particular case, the sampling rate information contained in the input waveform file will be always used. This field is not active if File Format is the "SigStudioEncrypted".
- Baseband Sample Rate
  The speed at which samples in the output waveform will be converted. This value is equal to the DAC sampling rate in any of the DAC direct modes and a submultiple (x3, x12, x24, x48) for up-converter modes. It can be set in all Resampling Modes except for the Output_RL mode. This field is not active if File Format is the "SigStudioEncrypted".
- Start Sample
  This field can be used to select the starting sample of the section of the input waveform to be imported. It cannot be set to a value higher than the Stop Sample. This field is not active if File Format is the "SigStudioEncrypted".
- Stop Sample
  This field can be used to select the final sample of the section of the input waveform to be imported. It cannot be set to a value lower than the Start Sample. This field is not active if File Format is the "SigStudioEncrypted".
- Scale
  This checkbox controls the way the input output waveform will be scaled after resampling. If unchecked, the output waveform samples will not be re-scaled. Sample levels over +1.0 or under -1.0 will be clipped. This field has no effects if File Format is the "SigStudioEncrypted".

Scaling Section

- DAC Max
  Imported waveforms may occupy a limited range of the "DAC" full scale. This parameter sets the maximum level. If set to a lower level than DAC Min, this will be automatically set to the same level. Acceptable range for this parameter is -1/+1, being the full dynamic range of the "instrument" DAC. This field has no effects if File Format is the "SigStudioEncrypted".
- DAC Min
  DAC Max: Imported waveforms may occupy a limited range of the "DAC" full scale. This parameter sets the minimum level. If set to a higher level than DAC Max, this will be automatically set to the same level. Acceptable range for this parameter is -1/+1, being the full dynamic range of the "instrument" DAC. This field has no effects if File Format is the "SigStudioEncrypted".

Preview Section

- Waveform Preview Toolbar

  The waveform preview toolbar includes the icons which provides different functionality to preview the waveform. For details, see *Preview Section Waveform* Preview Toolbar.

- Show Next Waveform Preview
  This checkbox affects the behavior of the preview for the next waveform. If selected, a preview of the imported waveform is displayed. Leave this checkbox unselected to speed up the import of large waveforms.

- Save To File...
  Signals can be stored in files in whether BIN (for non IQ modes) or IQBIN (for IQ modes) formats. These files may be reused within the Import Waveform tab. This button is not active if File Format is the "SigStudioEncrypted".

- Send To Instrument
  Signal will be transferred to the selected segments of the selected channels. The previous running status for the target instrument will be preserved but sampling rate may be modified depending on the waveform requirements.

- Set Default
  All the imported waveform parameters are set automatically to their corresponding default values.

2.13 Status/Control Tab

This tab displays the most important status information of the M8190A module. Use this tab to start and stop signal generation and to send software triggers and events to the module.

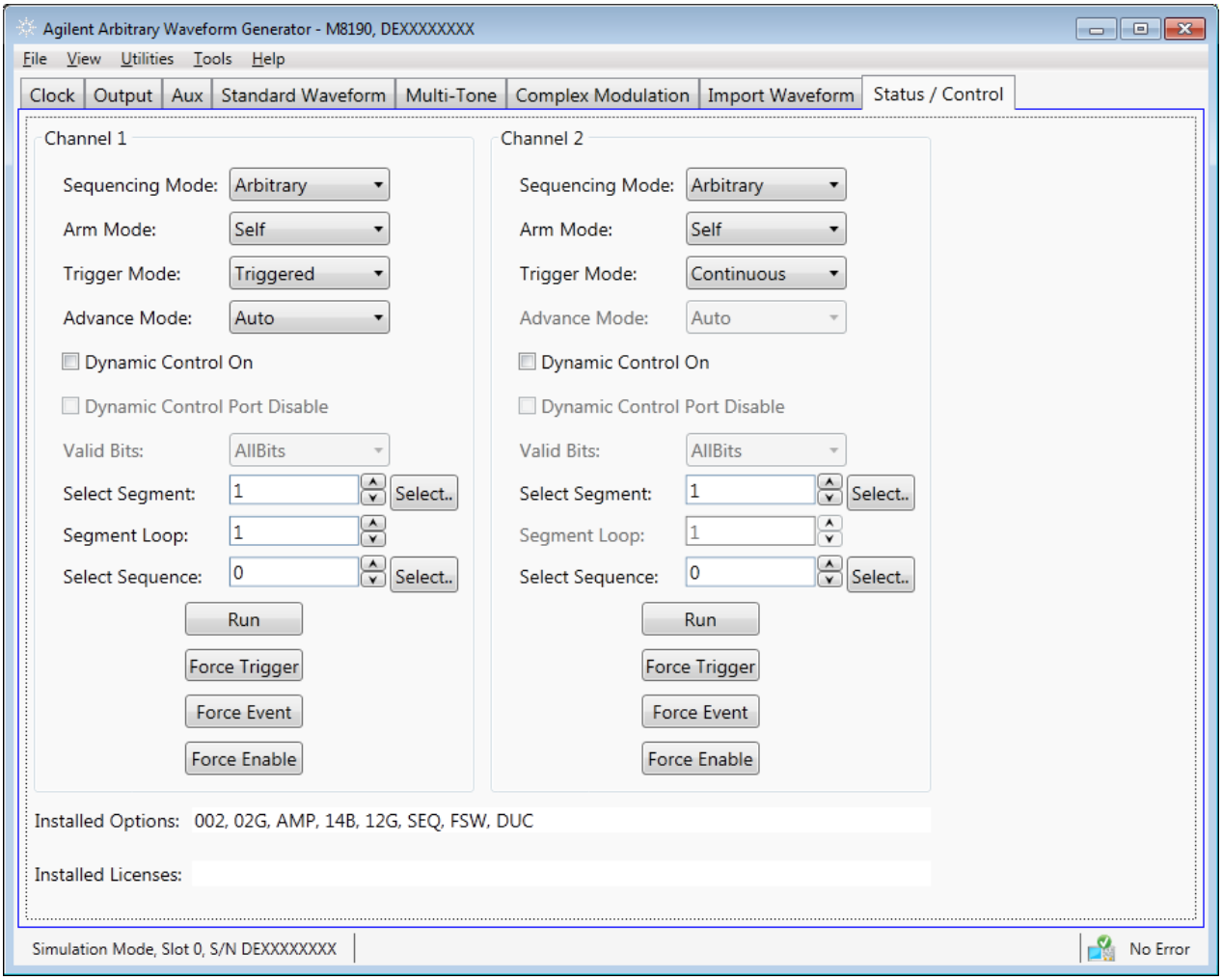


Figure 2-15: Status/Control Tab

This tab has the following controls.

- Sequencer Modes

  Arbitrary – Generate arbitrary waveform segments.

  STSequence – Generate sequences of segments.

  STSScenario – Generate scenarios (sequences of sequences).

- Arm mode

  Armed – Signal generation starts when an "enable" is received.

  Self – Signal generation starts as defined by the trigger mode.

- Trigger mode

  Continuous – Signal generation starts immediately after pressing the Run button. No trigger needed.

  Triggered – Signal generation starts after a trigger is received.

  Gated – Signal generation starts when a rising edge is received on the trigger input and pauses when a falling edge is received. Signal generation restarts after the next rising edge.

- Advance Mode

  Specifies how the sequencer proceeds from one segment iteration to the next in sequencing mode "Arbitrary" and trigger mode "Triggered". Signal generation starts after receiving the start trigger.
  Auto – The segment is repeated as specified by "Loop Count".
  Conditional – The segment is repeated infinitely. The value of "Loop Count" is not used.
  Repeat - The segment is repeated as specified by "Loop Count". After an advancement event is received, the segment is again played "Loop Count" times and so on.
  Single – The segment is played once. After an advancement event is received, the next segment iteration is played until all "Loop Count" iterations are played.

- Segment Loop
  Specifies the segment loop count (number of times the selected segment is repeated) for the selected segment in arbitrary mode. Value range is1 to 4,294,967,295 (4G-1).

- Dynamic Control On

  Use this check box to enable or disable dynamic segment or sequence selection mode. If enabled, segments or sequences can be switched using a command or by a signal on the dynamic control port.

- Dynamic Control Port Disable

  Use this check box to disable the dynamic control port hardware input. If disabled, segment or sequences can only be switched using a command.

- Valid Bits

  Selects the bit width of the dynamic control port.

  AllBits – The complete width of 19 bits is used for segment or sequence selection.

  LowerBits – The 13 lowest bits are used for segment or sequence selection. The upper 6 bits are set to 0.

- Select Segment

  Selects the segment to be played in sequencing mode "Arbitrary". In dynamic segment selection mode it selects the segment that is played before the first segment is dynamically selected.

  Pressing the "Select.." button opens a list box with the available segments and their lengths.

- Select Sequence

  Selects the sequence to be played in sequencing mode "STSequence". In dynamic sequence selection mode it selects the sequence that is played before the first sequence is dynamically selected.

  Pressing the "Select.." button opens a list box with the available sequences and their lengths. This list box only display the sequences defined using the commands in the Sequence subsystem and not the sequences defined using the commands in the STABle subsystem.

- Run/Stop

  Use this button to switch between Run and Program mode per channel.

- Force Trigger

  Use this button to send a software trigger to a channel.

- Force Event

  Use this button to send a software event to a channel.

- Force Enable

  Use this button to send a software "enable" to a channel.

- Installed Options

  This field displays the installed options of the module.

- Installed Licenses

  This field displays the installed licenses.

2.14 Correction File Format

A correction file is an ASCII delimited file carrying all the information required to compensate or embed a given frequency response in the multi-tone or complex modulation signal. The file must be composed of a header including a series of lines with identifiers and parameters, and a list of numerical correction factors. In lines including more than one item (i.e., one identifier and one parameter), the items must be separated using commas. Identifiers and parameters are not case sensitive.

These are the significant fields for the header:

- ChannelNum: It states the number of channels (1 or 2) supported by the correction file. It is a mandatory field.
- InputBlockSize: It states the number of valid correction factors in the file. It is a mandatory field.
- XStart: It is frequency in Hz corresponding to the first entry in the correction factor section of the file. It is a mandatory field for multi-tone generation in direct mode and optional for multitone in upconverter mode and complex modulation.
- XDelta: It is frequency distance in Hz between consecutive entries in the correction factor section of the file. It is a mandatory field.
- YUnit: Units for the amplitude values in the correction factor section of the file. Parameter associated to it may be 'dB' (for logarithmic relative amplitudes) or 'lin' (for dimensionless linear relative amplitude). This parameter is optional and its default value is 'lin'. Phase unit must be always stated in radians.

The order of the above entries is not relevant. The correction factor section starts with a line including a single 'y' or 'Y' character. For one-channel correction files, entries in this section are made by Amp1(Fi), Phase1(Fi) pairs while for two-channel files entries are made by Amp1(Fi), Phase1(Fi), Amp2(Fi), Phase2(Fi) sets. In particular, this format is compatible with adaptive equalizer files exported in comma-separated value (CSV) format from the Keysight 89600 VSA software package. These files reflect the channel response corrected by the equalizer so they should be applied through the selection of the 'Channel_Response' option in the corresponding 'CorrectionMode' drop-down list in the 'Corrections' section of the 'Multi-Tone' or 'Complex Modulation' panel, respectively. Comments must start with the '//' character sequence and may use a complete line or be located at the end of any valid line. Empty lines are also valid.

This is an example for a one-channel correction file:

```
// MyCorrectionFile
ChannelNum, 1
InputBlockSize, 1024
XStart, 1.0E+09 // 1.0GHz
XDelta, 1.0E+06
YUnit, lin
Y
0.987, -0.2343
0.995, 0.5674
...
...
1.269, -0.765
```

This is an example for a two-channel correction file:

```
ChannelNum, 2
InputBlockSize, 1024
XStart, 1.0E+09
XDelta, 1.0E+06
YUnit, lin
Y
0.987, -0.2343, 0.976, -0.1827
0.995, 0.5674, 0.975, -0.1645
...
...
1.269, -0.765, 1.190, -0.5632
```

The above files are composed of a header with relevant information. In these particular cases, the files contain 1024 linear correction factors spaced by 1 MHz and starting at 1GHz. The 'Y' character indicates the starting point for the correction factor list composed of 1024 lines with amplitude/phase pairs separated by commas. For one-channel files there is a amplitude/phase pair per line while for two channel files there are two pairs (Amp1, Phase1, Amp2, Phase2).

The way this information is applied by the Soft Front Panel software depends on the signal generation mode. For direct conversion modes ('Generate IQ'

unchecked), corrections are applied directly to the tones based on their absolute frequency. For up-converted modes ('Generate IQ' checked), corrections are applied to the complex baseband signals. So, the internal or external carrier frequency is represented by the central entry in the list (i.e., entry #512 in the 1024 entries example shown above) regardless of the 'XStart' parameter.

2.15 M8190 Soft Front Panel and M8190 Firmware

The M8190 Soft Front Panel can connect to an already running M8190 Firmware, which was started previously.

If no M8190 Firmware instance was started for the selected hardware yet, the Soft Front Panel will start it. In this case, the Firmware's window is minimized and only an icon in the taskbar's notification area is shown.

It is recommended to exit the Soft Front Panel before the Firmware; otherwise, the Soft Front Panel will continue to try to communicate with the Firmware. This will of course fail and the Soft Front Panel will display error messages.

---

3 Sequencing

3.1 Introduction

This chapter describes the sequencing capabilities of the instrument when having ordered the option sequencing (Option -SEQ).

3.1.1 Option Sequencing

Without the option sequencing (Option -SEQ) the capabilities of the instrument are limited to one segment which can be played in the non-dynamic arbitrary self armed mode.

3.1.2 Sequence Table

The sequencer is implemented in a table. Each table entry consists of a sequence vector, which contains all the necessary information that is required to play one single waveform segment like loop counter values, advancement parameters and references to the sample memory. Multiple adjacent sequence vectors can also be played together within one run. The first sequence table entry is marked with a start pointer. After having finished one segment, the next table entry of the list is selected. If an actually executed segment is the last segment of a loop a jump to the starting point of the loop might be initiated depending on the loop count.

The following drawing shows an example:

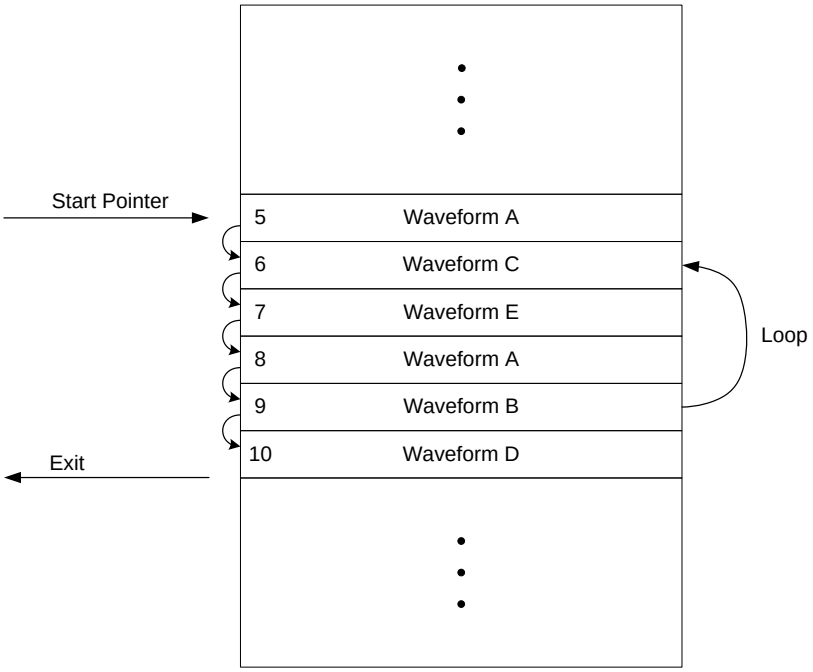


Figure 3-1: Sequence Table

The execution flow is started at address 5 and the waveform of every table entry is played. Within the sequence, a loop from address 9 to 6 is executed for a number of times, specified by loop counter values. It is possible to access the same sample data from different sequence vectors. In this example, waveform A is accessed from sequence vector 5 and 8.

3.1.3 Sequencer Granularity

The sequencer is running at a lower clock speed than the sample rate of the instrument. Therefore the sequencer has to play multiple samples within one sync clock cycle.

The number of samples played within one sync clock cycle is called sequencer granularity or segment granularity. For details, refer to the block diagram in section *1.5.1*.

3.2 Sequencing Hierarchy

3.2.1 Segment

A waveform segment consists of a defined number of samples, which are played, in a consecutive order. It is treated as a unit and can be repeated a specified number of times or can run continuously. The sample count of segments must be in multiples of the sequencer granularity. A minimum length is also required (see datasheet of the instrument).

A segment can be played standalone (see *Arbitrary Mode*) or can be part of a sequence.

3.2.2 Sequence

Multiple segments can be combined to a sequence. A sequence can be executed continuously or for a specified number of times.

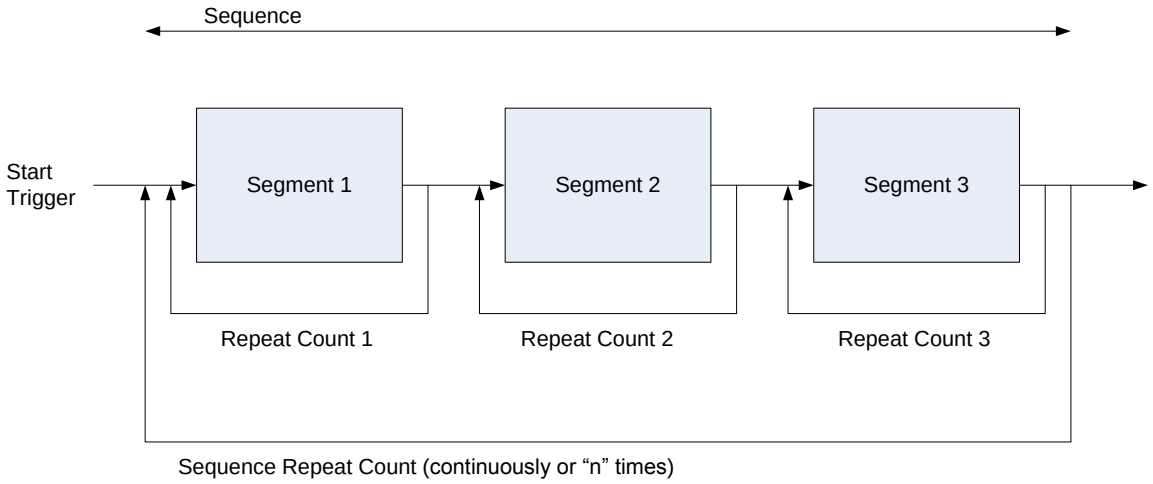


Figure 3-2: Sequence

A sequence can be played standalone (see *Sequence Mode*) or can be part of a scenario.

3.2.3 Scenario

Multiple sequences can be combined into a scenario (see *Scenario Mode*). A scenario can be executed continuously or a specified number of times.

3.3 Trigger Modes

The trigger mode defines the way, how segments, sequences and scenarios begin with playing waveform data. After having setup the instrument, it is started. Then the start of segments, sequences and scenarios depends on the different trigger modes.

3.3.1 Continuous

In the trigger mode **Continuous**, the sequencer is started immediately after the instrument. In this mode, the waveform execution is infinite.

3.3.2 Triggered

In the trigger mode **Triggered**, the sequencer needs a trigger to start. After having received the trigger, the waveform is played a defined number of times, and then the sequencer is stopped again and is prepared to accept the next trigger. Every trigger that occurs before the currently running segment/sequence/scenario has completed, is ignored. Alternatively, after having received a trigger, a waveform can also be played infinitely. See *Sequencer Modes* for more details.

3.3.3 Gated

In the trigger mode **Gated**, both edges of the gate signal are used to start and stop the execution of the sequencer. After being stopped, the sequencer is prepared to accept a new rising edge of the gate and can be restarted again. In Gated Mode, the advancement mode of the top level (e.g. sequence advancement mode for sequences) must be set to Continuous.

3.4 Arm Mode

Sometimes it is desired to play an idle waveform instead of a static idle value before having started to play the real waveform. With the arm mode it is possible to select the output signal of the instrument before having started the sequencer.

3.4.1 Self Armed

Whenever the arm mode is set to **Self Armed**, the instrument starts as defined by the selected trigger mode.

3.4.2 Armed

For all cases where the trigger mode is set to **Continuous** and the arm mode to **Armed**, the first segment/sequence is played infinitely after start. After having received a rising edge of Enable, the sequence/scenario advances to the next segment/sequence and continues to execute as described in trigger mode **Continuous**. This mode doesn't make any sense for the execution of standalone segments or for the trigger modes **Triggered** and **Gated**. Therefore in these cases the described behavior is not available and **Armed** is treated like **Self Armed** with an additional enable flag as start condition.

3.5 Advancement Modes

The advancement mode specifies the way of how one element like a segment, sequence or scenario advances to the next element or how it is repeated.

The advancement mode can be individually specified for each single element. The exact behavior depends on the sequencing, arm and trigger mode.

There could be different advancement modes on different hierarchy levels. Some of these modes require an advancement event to proceed. In cases where the advancement event has to be evaluated simultaneously in multiple hierarchy levels, the output behavior could be unexpected, especially when conditional advancement modes are used. For more details, refer to the examples given in the section *Sequencer Modes*.

3.5.1 Auto

After having executed all loops, the sequencer advances to the next element automatically. No external interaction is required for advancement.

3.5.2 Conditional

The sequencer repeats the current element until it receives the correct advancement event. After having received the advancement event, the current element is played to the end before switching to the next one.

3.5.3 Repeated

After having executed all loops the sequencer stops and plays the last value of the current element. After having received the advancement event, the sequencer starts playing the next element. When receiving the advancement event before having played all repetitions, all repetitions will be played before moving to the next element.

3.5.4 Single

After having executed an element once, the sequencer stops and plays the last value of the element. After having received the next advancement event the process is repeated until having executed all loops of the current element. Then the execution advances to the next element.

3.6 Sequencer Controls

Sequencer Controls are used to influence the sequencer. So they can control the waveform generation.

3.6.1 External Inputs

3.6.1.1 TRIGGER/EVENT

The M8190A accepts a wide range of external trigger signal levels to easily adapt to a measurement setup. The input threshold is user configurable along with the polarity or whether rising, falling or both edges are to be taken into account. The TRIGGER and EVENT input signals are clocked internally with the SYNC clock. [SYNC clock = Sample clock divided by 48 (64) in 14-bit (12-bit) mode or divided by 24*interpolation_factor in interpolated modes]. To reduce the TRIGGER to DATA out uncertainty the signal applied to the external input connector needs to meet a setup and hold window. The timing is specified with respect to the SYNC Clk Out port. See the data sheet for further details.

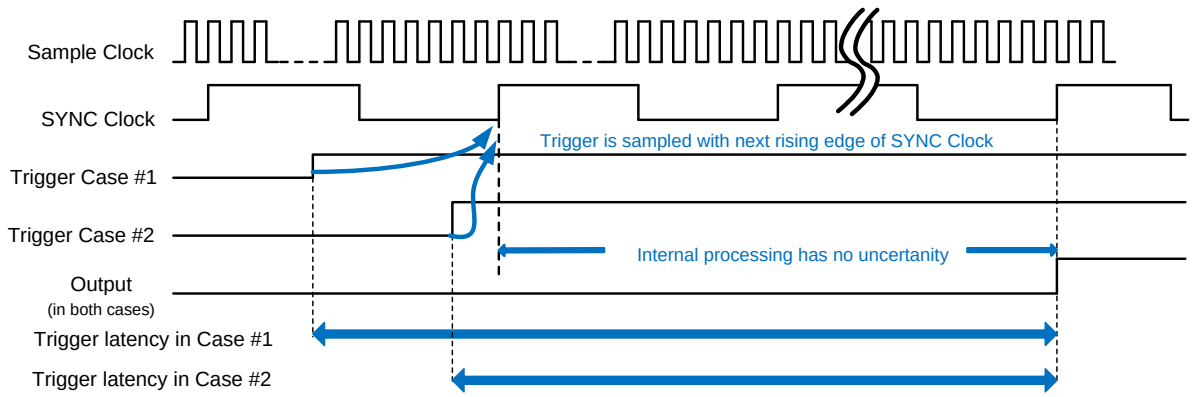


Figure 3-3: TRIGGER/EVENT

3.6.1.2 DYNAMIC CONTROL

The DYNAMIC CONTROL IN allows selecting a new segment or sequence at run time from an external source. The connector provides 13 data bits, a select bit and a load signal. As the internal sequencer table index is a 19 bit wide address the complete 19 bit have to be provided in two steps. The select signal is used to indicate which part of the index is present at the data pins. The data is latched with the rising edge of the Load signal.

DYNAMIC CONTROL IN Connector
Description: Receptacle, Mini D
Number of Contacts: 20
Manufacturer: 3M
Part Number: 10220-0210EC

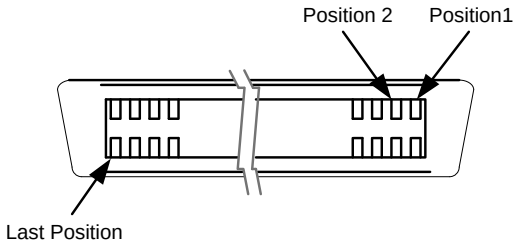


Figure 3-4: DYNAMIC CONTROL IN connector

Table 3-1: Pin Assignment

Pin No.	Signal Assignment
1	NC
2	Ground
3	Load
4	Data_Select
5	D0
6	D1
7	Ground
8	D2
9	D3
10	D4
11	D5
12	D6
13	D7
14	Ground
15	D8
16	D9
17	D10
18	D11
19	Ground
20	D12

Signal Levels

Low Level	0 V to +0.7 V
High Level	+1.6 V to +3.6 V
Impedance	Internal 10 kΩ pull-down resistor to GND

Signal Description

Data Input

The input data represents an index to the next sequence table entry to be played by the AWG module. 2^19 sequence table entries are available.

Data_Select

The data select signal controls which part of the 19 bit internal sequencer index is loaded from the Data_In pins. If Data_Select is 1 only the Data_In bits 5..0 are used to load bits 18..13 of the sequencer index. If Data_Select is 0 the Data_In bits 12..0 are loaded to internal sequencer index 12..0. To load the full 19 sequence index bits first the upper part needs to be loaded (Data_Select = 1) then the lower part.

Load

A rising edge on the Load signal latches the data present at the Data_In pins into the respective internal registers. If the Data_Select pin is low the loaded sequencer index is passed to the sequencer for execution.

NOTE In 13 bit mode only the lower part of the sequencer entry address is modified. Data_Select can be left unconnected (low) in this mode. Only 8 k entries can be addressed in 13 bit mode vs. 512 k entries in 19 bit mode.

Timing Diagrams 13 Bit Mode

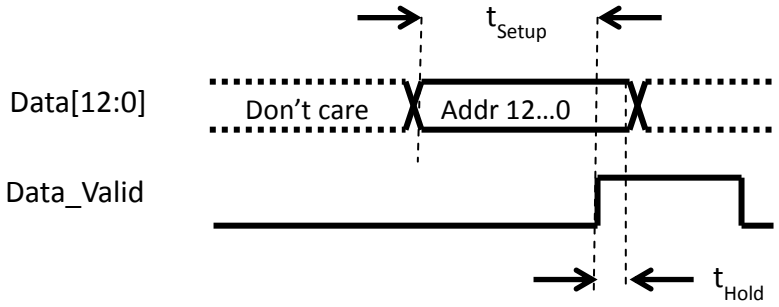


Figure 3-5: 13 Bit Mode

19 Bit Mode

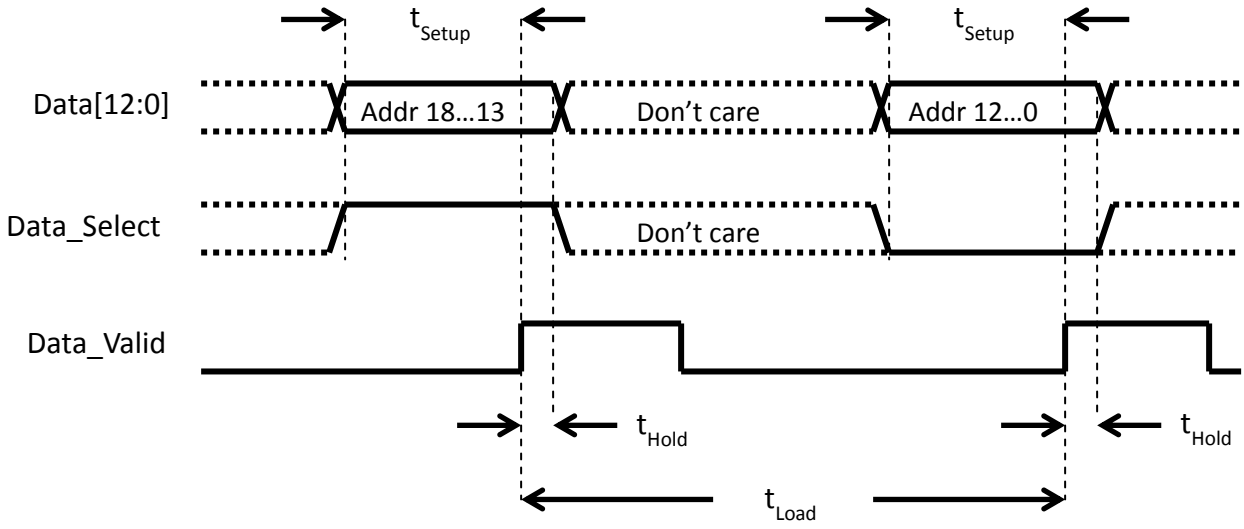


Figure 3-6: 19 Bit Mode

Timing in both cases:
t_{Setup} = 5 ns
t_{Hold} = 0 ns
t_{Load} = 7 SYNC clock cycles

Maximum update rate: 1MHz

3.6.2 Logical Functions

3.6.2.1 Trigger/Gate/Enable

The **trigger**, **gate** and **enable** signals are used to control the start behavior of the sequencer, depending on the selected mode. The **trigger** starts the sequencer in trigger mode **triggered**; the **gate** has the corresponding functionality (start and stop) in trigger mode **gated**. The **enable** is needed in the **armed/continuous** mode. In this mode the first element is hold in the conditional advancement mode until **enable** becomes active. During further loops of the sequence or scenario, the **enable** is ignored and the element is executed with the advancement mode specified in the sequence table. So the **enable** allows providing not only an initial offset value, before the real start of the sequencer, but also an initial segment or sequence.

3.6.2.2 Advancement Event

The advancement event is used to advance within a scenario or sequence. Responsible for the type of advancement is the selected advancement mode of the element. The advancement event is stored internally until the sequencer uses it.

Example:
When receiving an advancement event while executing a conditional segment, the advancement event is stored until reaching the end of the segment where the advancement is used. Then the stored advancement event is cleared and the instrument is able to receive the next one.

3.6.2.3 Dynamic Select

The instrument provides a dynamic sequencing mode, which allows changing the actually running segment or sequence without stopping and reprogramming the instrument. The selected sequencer index is modified either by the external DYNAMIC CONTROL input or via remote programming. Up to 512k sequencer table indices can be addressed.

3.6.2.4 Run

The run input is a software button or command, which switches the instrument from programming mode to run mode.

3.6.3 Internal Trigger Generator

The M8190A provides a configurable internal trigger generator that allows for generation of a periodic trigger signal that is frequency locked to the clock of the sequencer engine.

3.6.4 Mapping External Inputs to Logical Functions

The logical functions controlling the sequencer can be connected to multiple sources. The following table shows all possible mappings.

Table 3-2

Functions	Inputs				
	Trigger/Gate Input (SMA Connector)	Trigger Generator	Event Input (SMA Connector)	Dynamic Port (Dynamic Control In Connector)	Software
Trigger/Gate	Default				
Enable (= Start in armed mode)	Default (Armed)				
Advancement Event			Default		
Dynamic Select				Default	
RUN					

NOTE

The software controls are logically ored with the external input or in case of the dynamic control, the software controls have precedence unless the hardware inputs are explicitly disabled using the commands.

The figure below shows the logical input connectivity options provided by the M8190A for the TRIGGER and EVENT inputs.

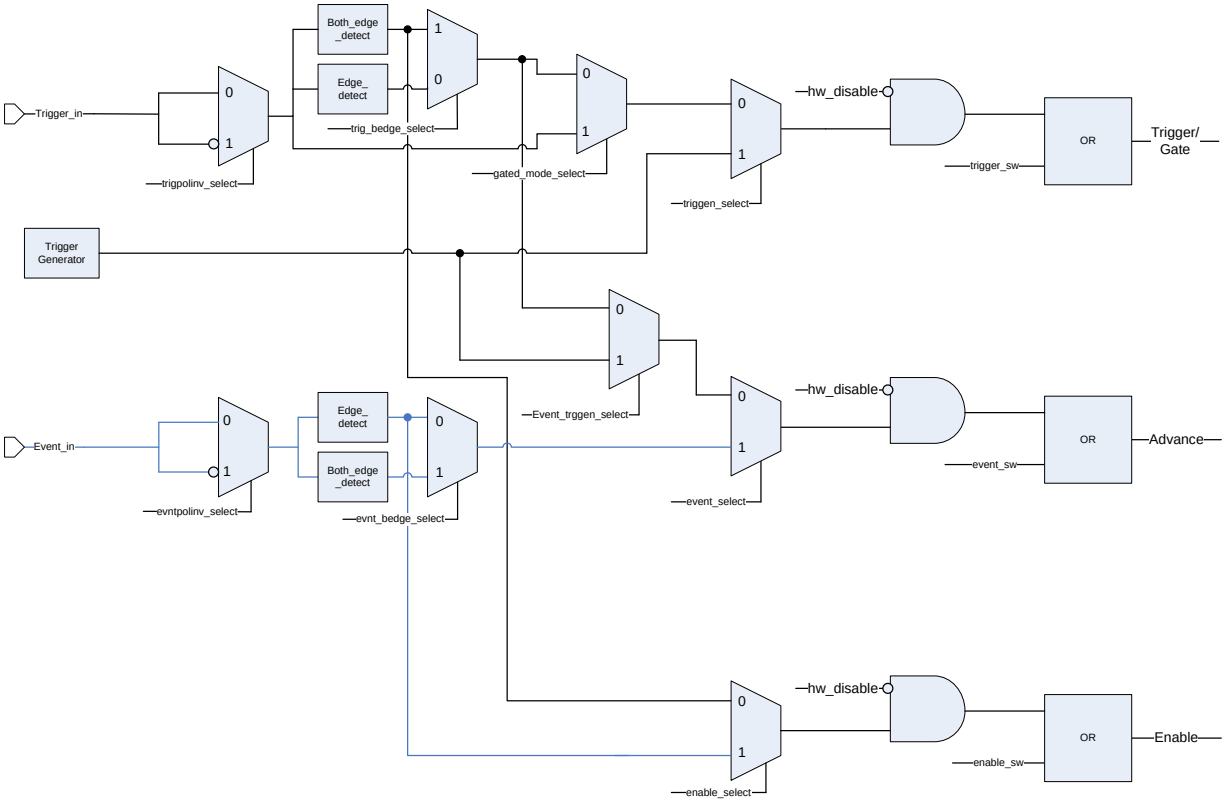


Figure 3-7: Logical input connectivity options provided by M8190A

3.7 Sequencer Execution Flow

The given drawing shows an overview of the different trigger modes and the interaction with some of the conditional inputs.

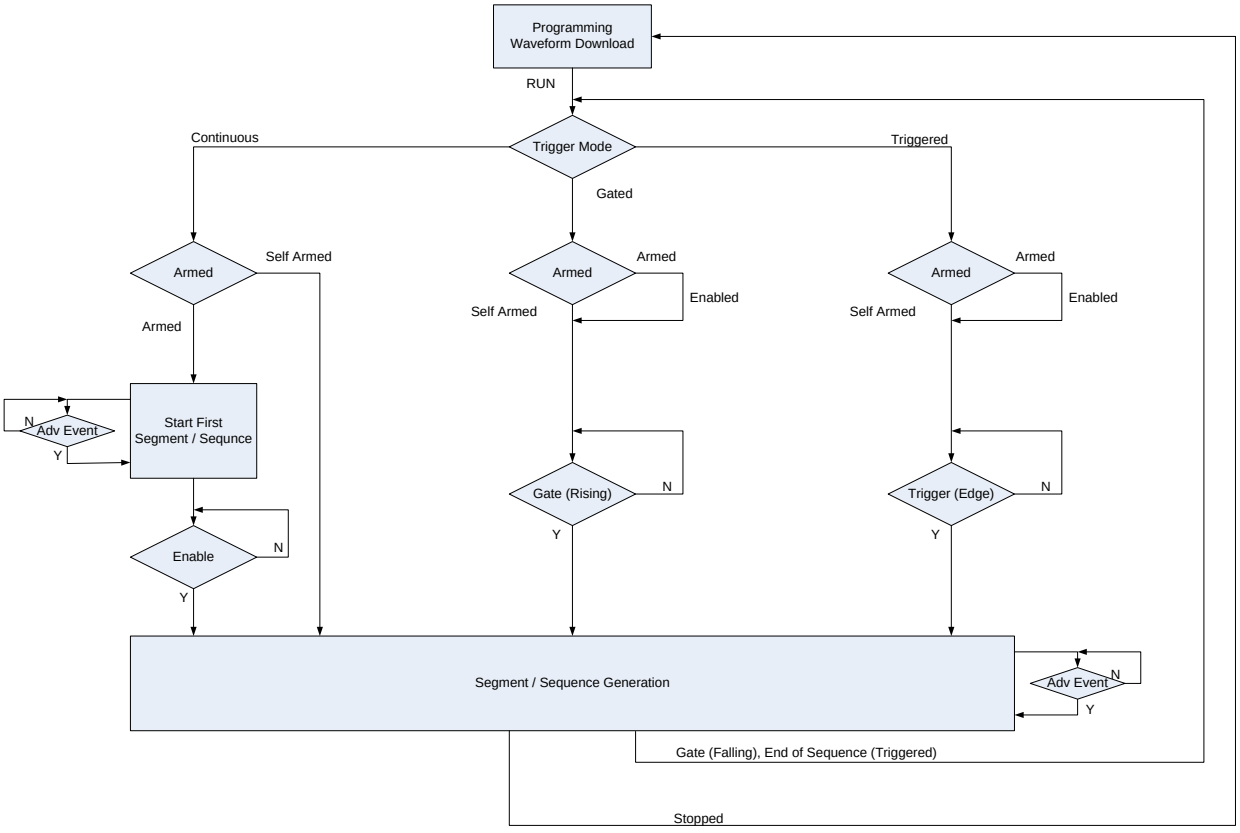


Figure 3-8: Sequencer execution flow

RUN is moving the instrument from the programming mode to execution mode. Dependent from the selected trigger mode, the behavior in **Armed** mode is different. In trigger mode **Continuous**, the enable signal is used to control the execution of the first segment or sequence. In trigger mode **Gated** or **Triggered**, the **enable** is used as an additional start input.

3.8 Sequencer Modes

This section describes the various sequence modes and their behavior depending on trigger mode and arm mode. Some of them are illustrated with examples. Every run of the sequencer starts with a static offset value, which represents the DAC value zero in the signed interpretation.

So this value is:

$$Offset = \frac{Max.Dac - Min.Dac}{2}$$

A stop (See SCPI command *:ABORt[1|2]*) of the instrument is an abort initiated by software which is unrelated to the currently running sequencer. So the currently running segment/sequence or scenario is not completed before stopping.

3.8.1 Arbitrary Mode

In Arbitrary Mode, a single segment is played.

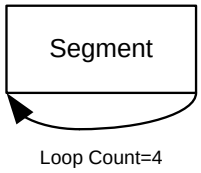


Figure 3-9: Segment

3.8.1.1 Self Armed

Trigger Mode Continuous

After programming, the segment is started automatically and is repeated infinitely.

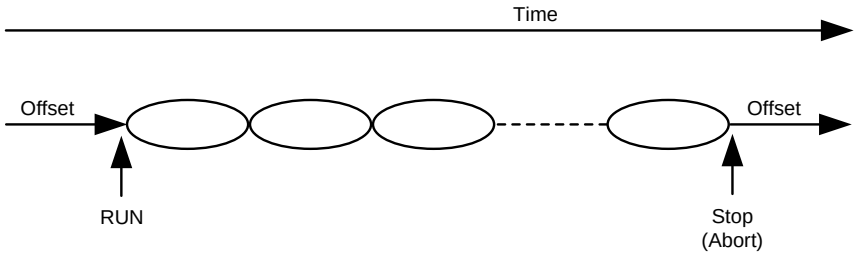


Figure 3-10: Trigger Mode Continuous

Trigger Mode Triggered

An Offset value is provided after programming. A trigger starts the segment.

The following segment advancement modes are available:

- Auto: The segment is executed the number of times specified by its loop count. Then the last sample is played at the end.
- Repeat: This advancement mode is quite the same like “Auto” with the difference that an advancement event is required at the end.
- Single: An advancement event is required for each segment repetition.
- Conditional: The segment is played infinitely after receiving a trigger. After being stopped (See SCPI command *:ABORt[1|2]*) the offset value is played.

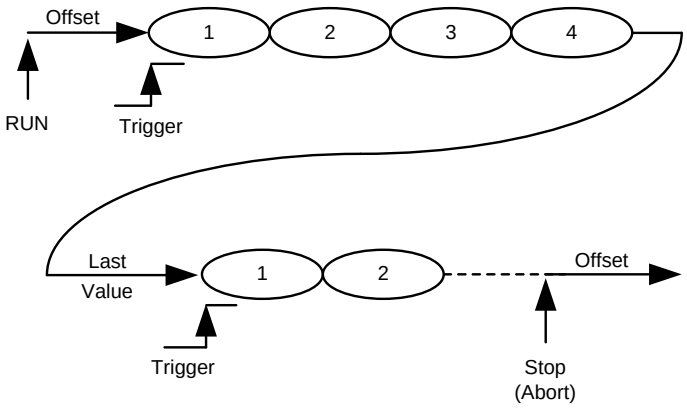


Figure 3-11: Segment Advance = Auto

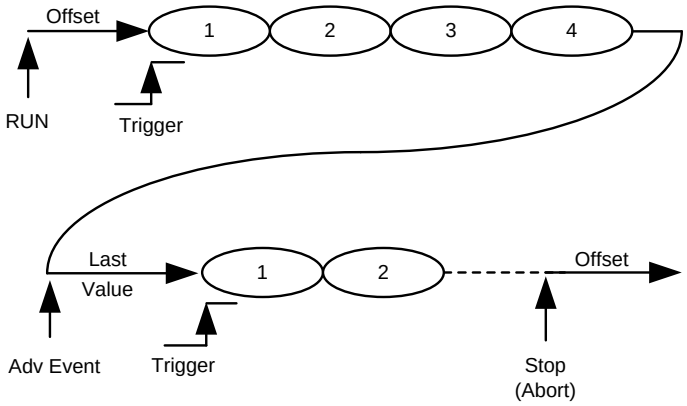


Figure 3-12: Segment Advance = Repeat

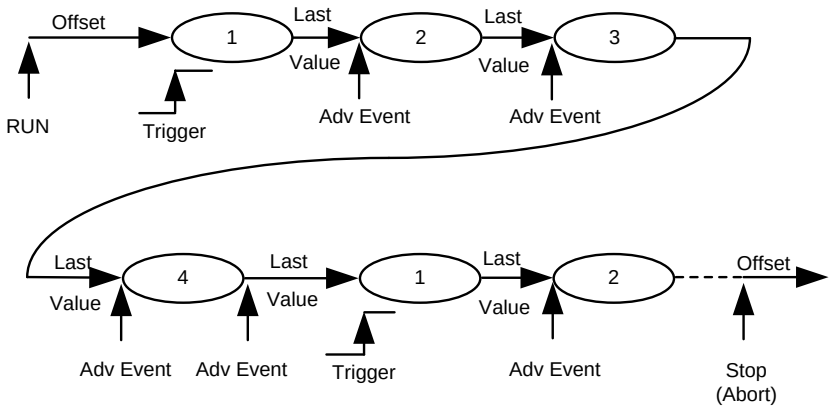


Figure 3-13: Segment Advance = Single

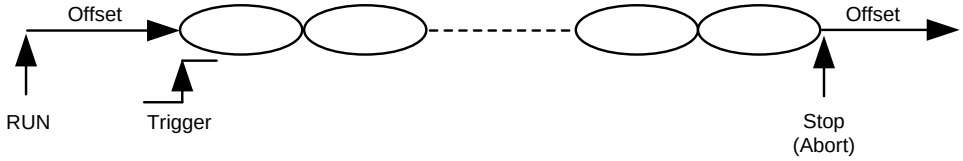


Figure 3-14: Segment Advance = Conditional

Trigger Mode Gated An Offset value is provided after programming. The rising edge of the gate starts the sequence and plays the segment infinitely until receiving the falling edge of the gate. After having received the falling edge of the gate, the segment is played for a number of times specified by the segment loop count. Then the segment is stopped at its end. Then the last sample value is provided.

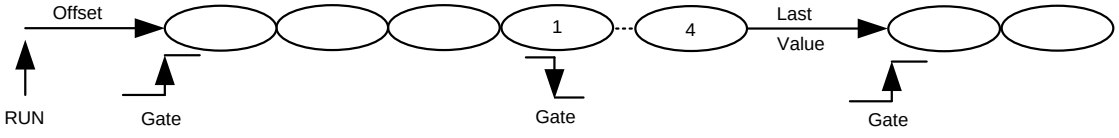


Figure 3-15: Trigger Mode Gated

3.8.1.2 Armed

Behavior is like self armed with an additional ENABLE. The enable is evaluated only once at the beginning. Later, changes of this signal are ignored.

Trigger Mode Continuous

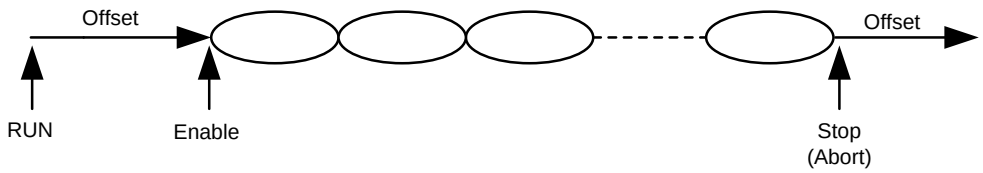


Figure 3-16: Trigger Mode Continuous

Trigger Mode Triggered

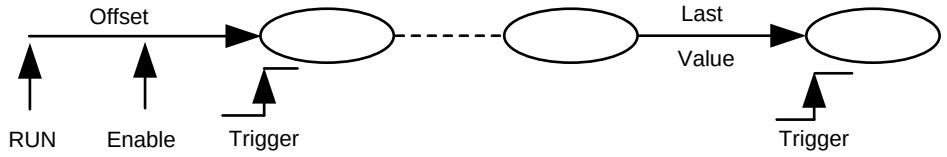


Figure 3-17: Trigger Mode Triggered

Trigger Mode Gated

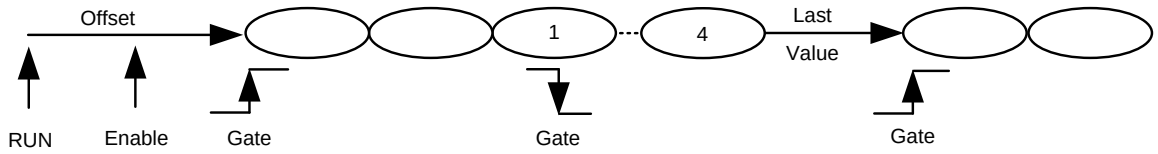


Figure 3-18: Trigger Mode Gated

3.8.2 Sequence Mode

In Sequence Mode, one or multiple segments are played.

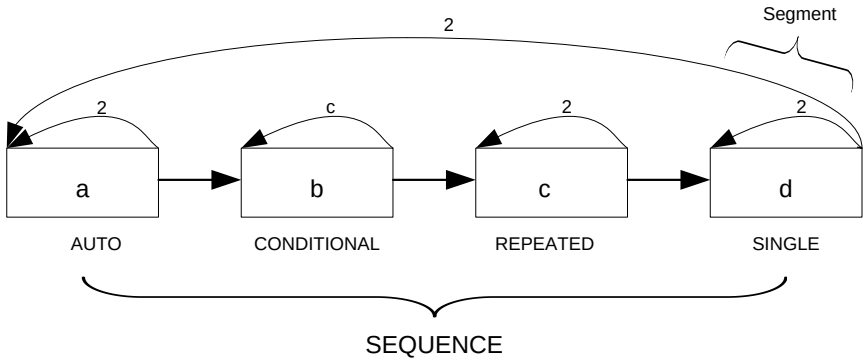


Figure 3-19: Sequence Mode

3.8.2.1 Self Armed

Trigger Mode
Continuous

After programming, the sequence is started automatically and played infinitely. The following segment advancement modes are available:

- Auto
- Conditional (Advancement Event)
- Repeated (Advancement Event)
- Single (Advancement Event)

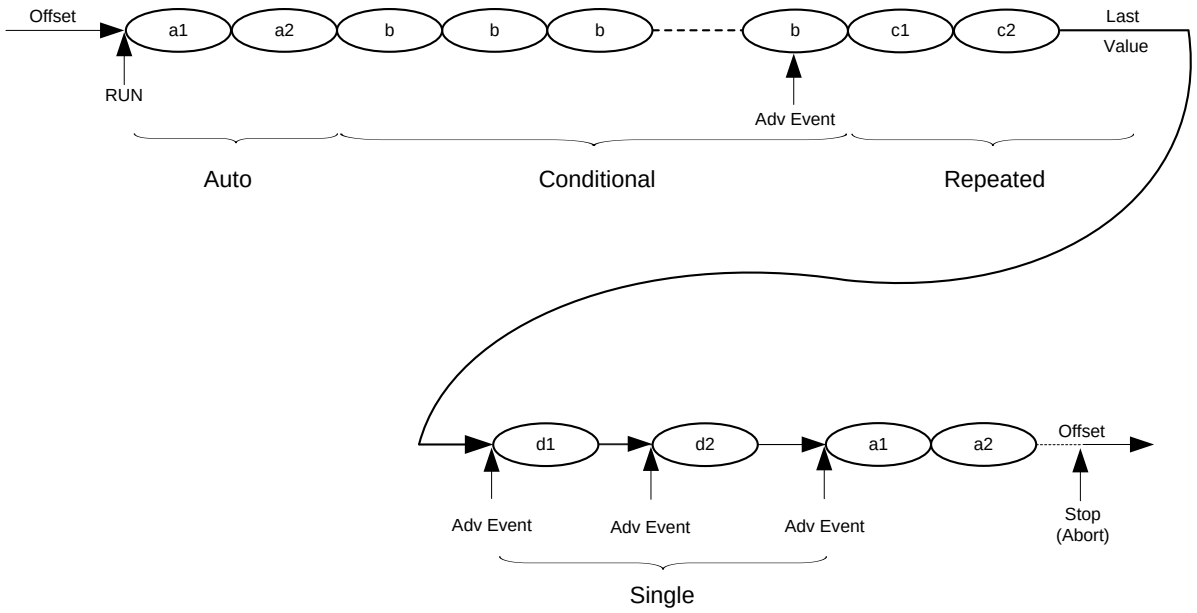


Figure 3-20: Trigger Mode Continuous

Trigger Mode Triggered

An Offset value is provided after programming. A trigger starts the sequence.

The following sequence advancement modes are available:

- Auto: The sequence is executed the number of times specified by its loop count. Then the last sample is played at the end.
- Repeat: This advancement mode is quite the same like “Auto” with the difference that an advancement event is required at the end.
- Single: An advancement event is required for each sequence repetition.
- Conditional: The sequence is played infinitely after receiving a trigger. After being stopped (See SCPI command *:ABORt[1|2]*) the offset value is played.

The following segment advancement modes are available:

- Auto
- Conditional (Advancement Event)
- Repeat (Advancement Event)
- Single (Advancement Event)

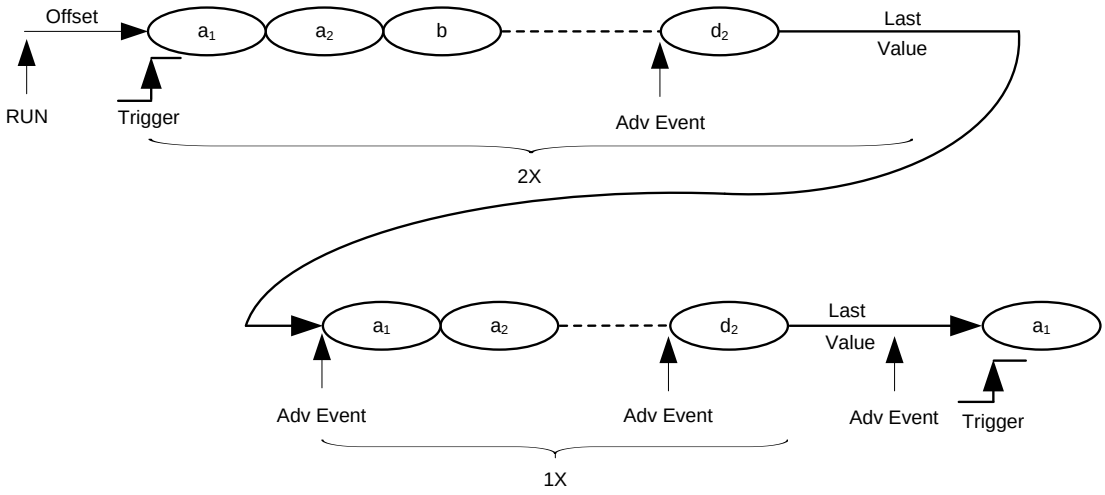


Figure 3-21: Sequence Advance = Auto

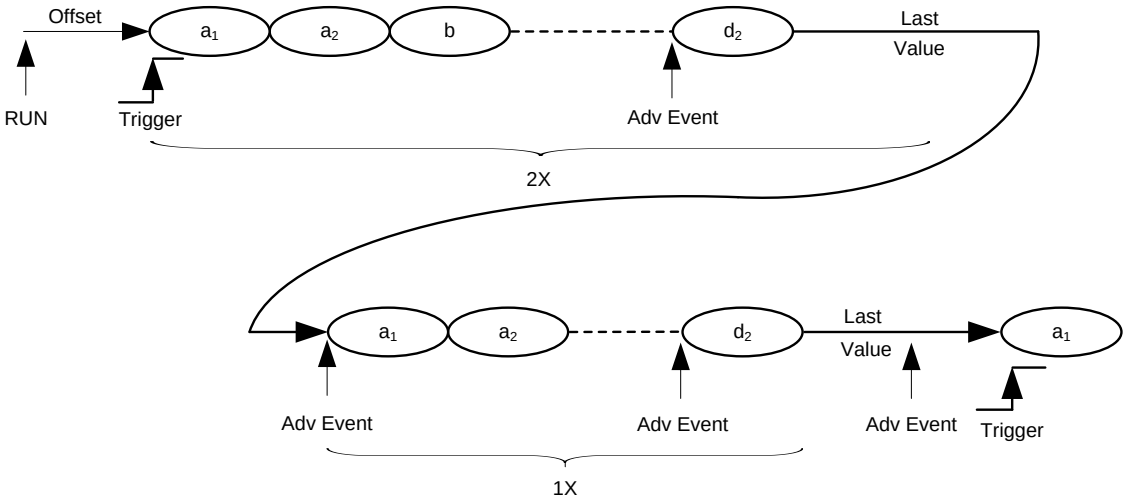


Figure 3-22: Sequence Advance = Repeated

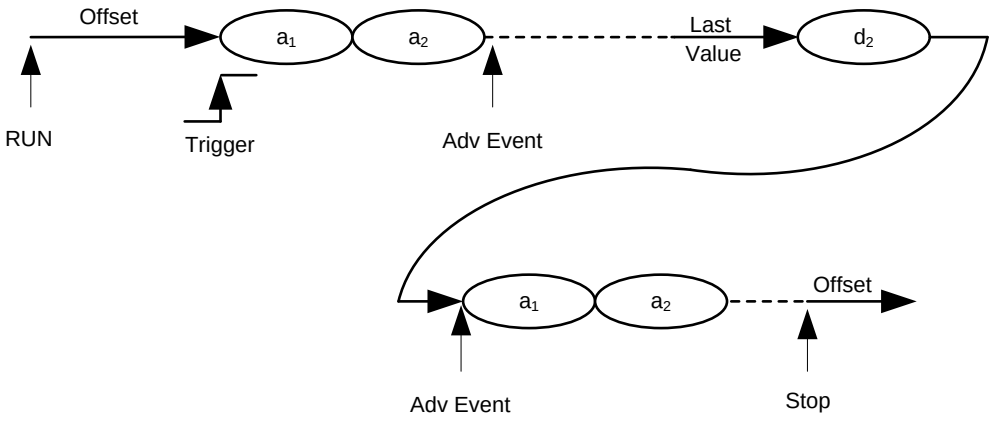


Figure 3-23: Sequence Advance = Conditional

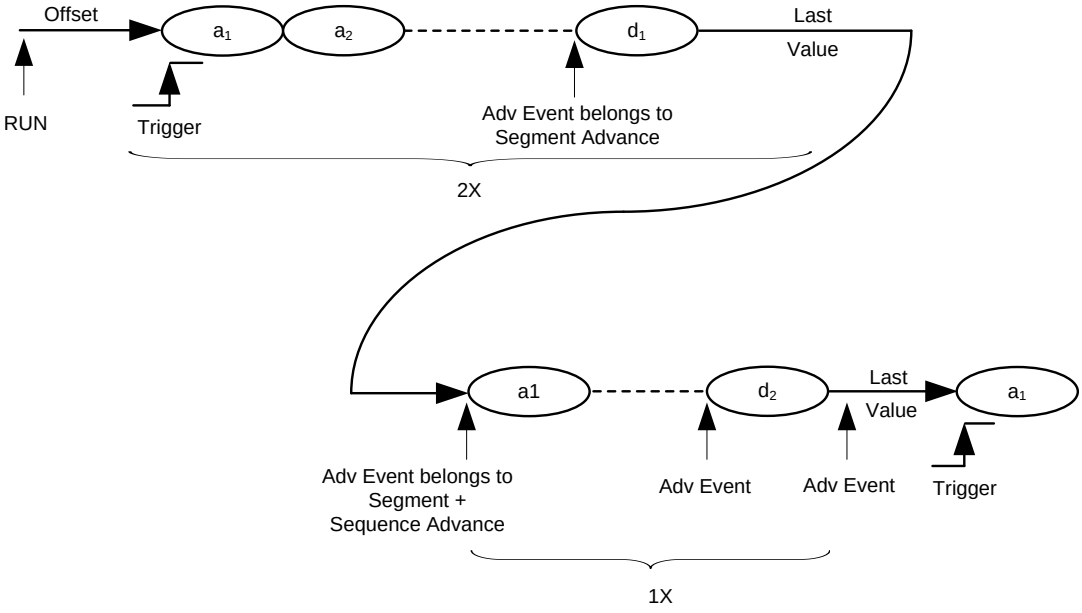


Figure 3-24: Sequence Advancement = Single

Trigger Mode Gated An Offset value is provided after programming. The rising edge of the gate starts the sequence and plays the sequence infinitely until receiving the falling edge of the gate. After having received the falling edge of the gate, the sequence is played for a number of times specified by the sequence loop count. Then the sequence is stopped at its end. Then the last sample value is provided.

The following segment advancement modes are available:

- Auto
- Conditional (Advancement Event)
- Repeat (Advancement Event)
- Single (Advancement Event)

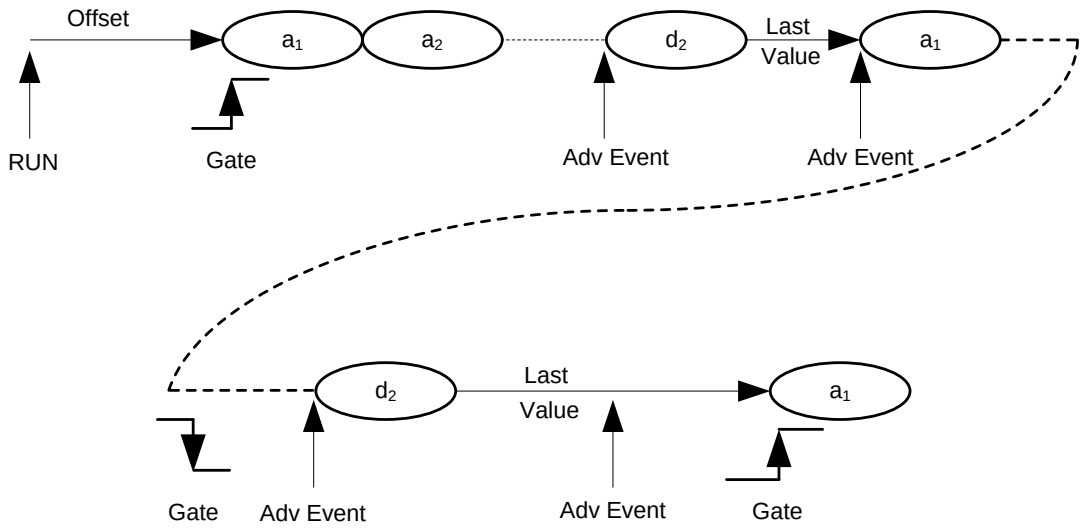


Figure 3-25: Trigger Mode Gated

3.8.2.2 Armed

Trigger Mode Continuous

After programming, the sequence is started automatically and the first segment is played repetitively until receiving an Enable. Then the first segment is played until the end and the sequence is continued.

The following segment advancement modes are available:

- Auto
- Conditional (Advancement Event)
- Repeat (Advancement Event)
- Single (Advancement Event)

The following sequence advancement mode is available:

- The sequence is played infinitely until being stopped. After being restarted, the first segment is played until it receives an enable.

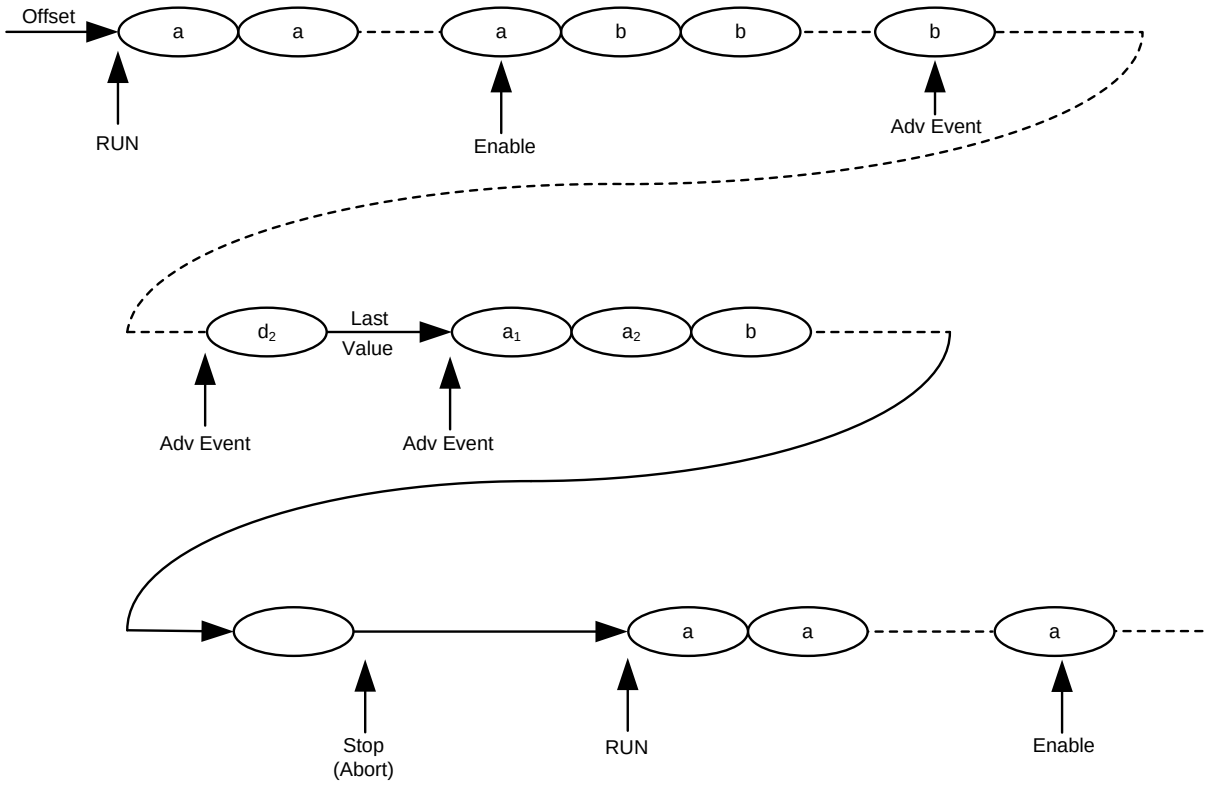


Figure 3-26: Trigger Mode Continuous

Trigger Mode Triggered Behavior is like self armed with an additional ENABLE. The enable is evaluated only once at the beginning. Later changes of this signal are ignored.

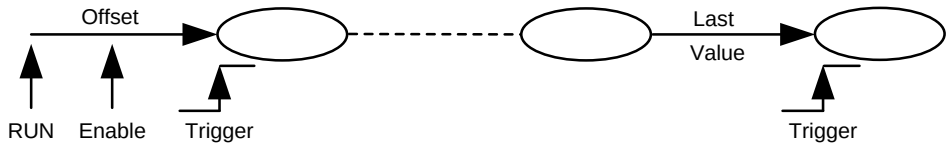


Figure 3-27: Trigger Mode Triggered

Trigger Mode Gated Behavior is like self armed with an additional ENABLE. The enable is evaluated only once at the beginning. Later changes of this signal are ignored.

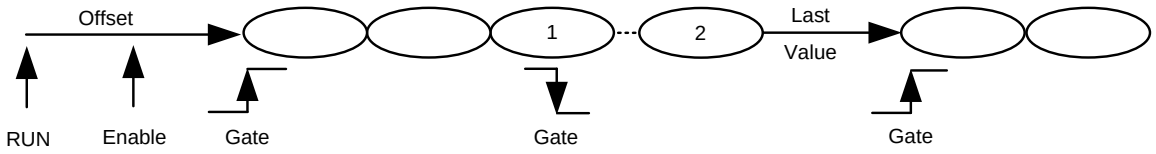


Figure 3-28: Trigger Mode Gated

3.8.3 Scenario Mode

In Scenario Mode, one or multiple sequences are played.

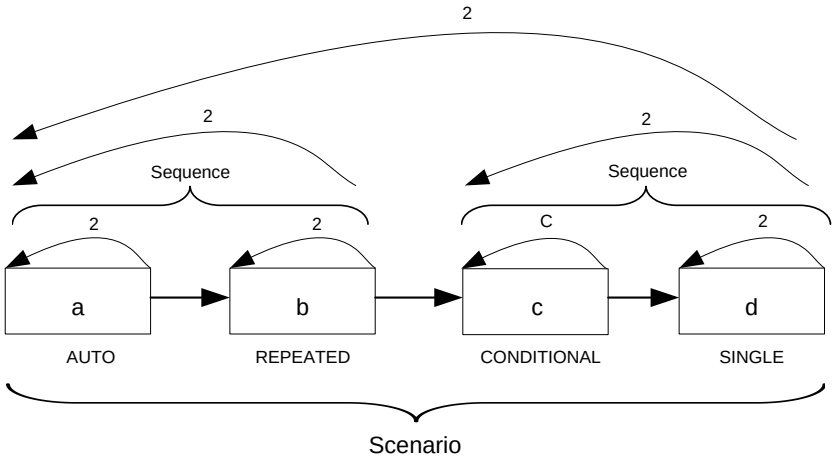


Figure 3-29: Scenario Mode

3.8.3.1 Self Armed

Trigger Mode
Continuous

After programming, the scenario is started automatically and played infinitely. The following segment/sequence advancement modes are available:

- Auto
- Conditional (Advancement Event)
- Repeat (Advancement Event)
- Single (Advancement Event)

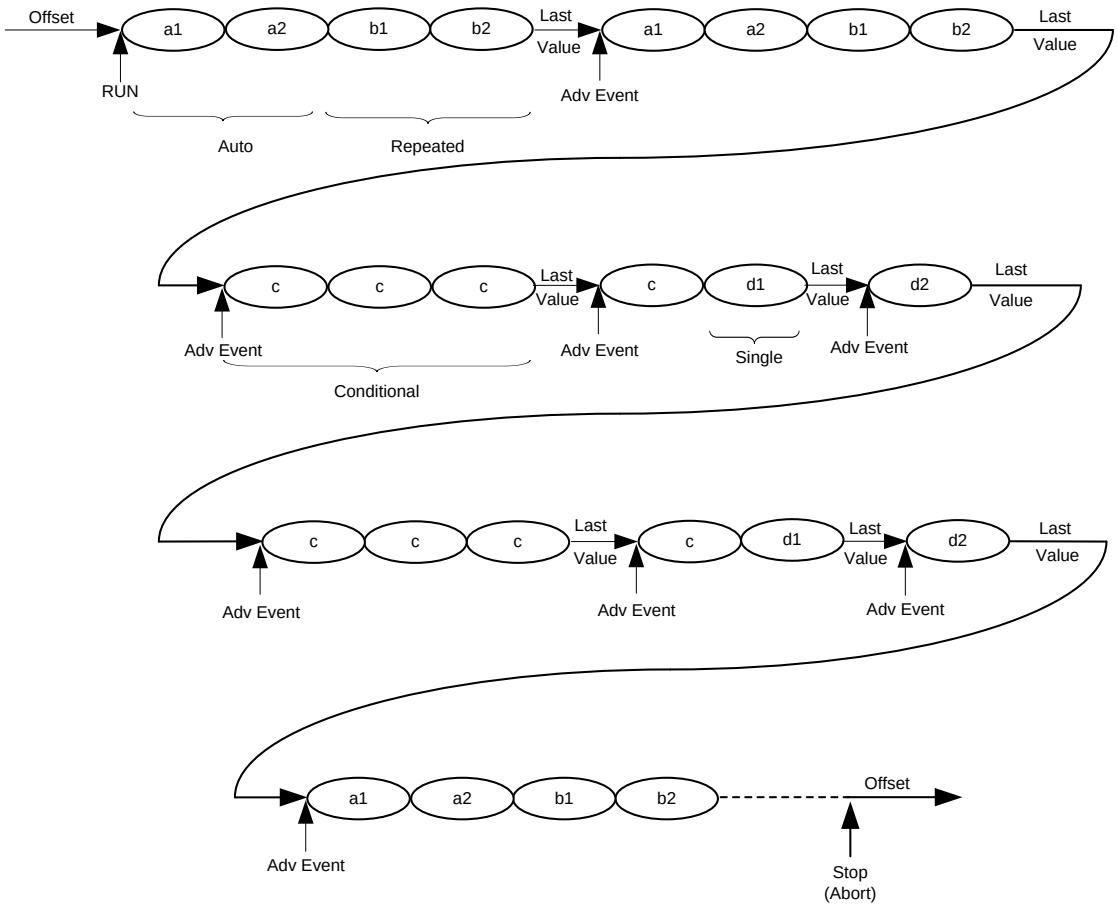


Figure 3-30: Trigger Mode Continuous

Trigger Mode Triggered

An Offset value is provided after programming. A trigger starts the scenario.

The following scenario advancement modes are available:

- Auto: The scenario is executed the number of times specified by its loop count. Then the last sample is played at the end.
- Repeat: This advancement mode is quite the same like "Auto" with the difference that an advancement event is required at the end.
- Single: An advancement event is required for each scenario repetition.
- Conditional: The scenario is played infinitely after receiving a trigger. After being stopped (See SCPI command *:ABORt[1|2]*) the offset value is played.

The following segment/sequence advancement modes are available:

- Auto
- Conditional (Advancement Event)
- Repeat (Advancement Event)
- Single (Advancement Event)

Trigger Mode Gated

An Offset value is provided after programming. The rising edge of the gate starts the scenario and plays the scenario infinitely until receiving the falling edge of the gate. After having received the falling edge of the gate, the scenario is played for a number of times specified by the scenario loop count. Then the scenario is stopped at its end. Then the last sample value is provided.

The following segment/sequence advancement modes are available:

- Auto
- Conditional (Advancement Event)
- Repeat (Advancement Event)
- Single (Advancement Event)

3.8.3.2 Armed

Trigger Mode Continuous

After programming, the scenario is started automatically and the first sequence is played repetitively until receiving an Enable. Then the first sequence is played until the end and the scenario is continued

The following segment/sequence advancement modes are available:

- Auto
- Conditional (Advancement Event)
- Repeat (Advancement Event)
- Single (Advancement Event)

The following scenario advancement mode is available:

- The scenario is played infinitely until being stopped. After being restarted, the first sequence is played until it receives an enable.

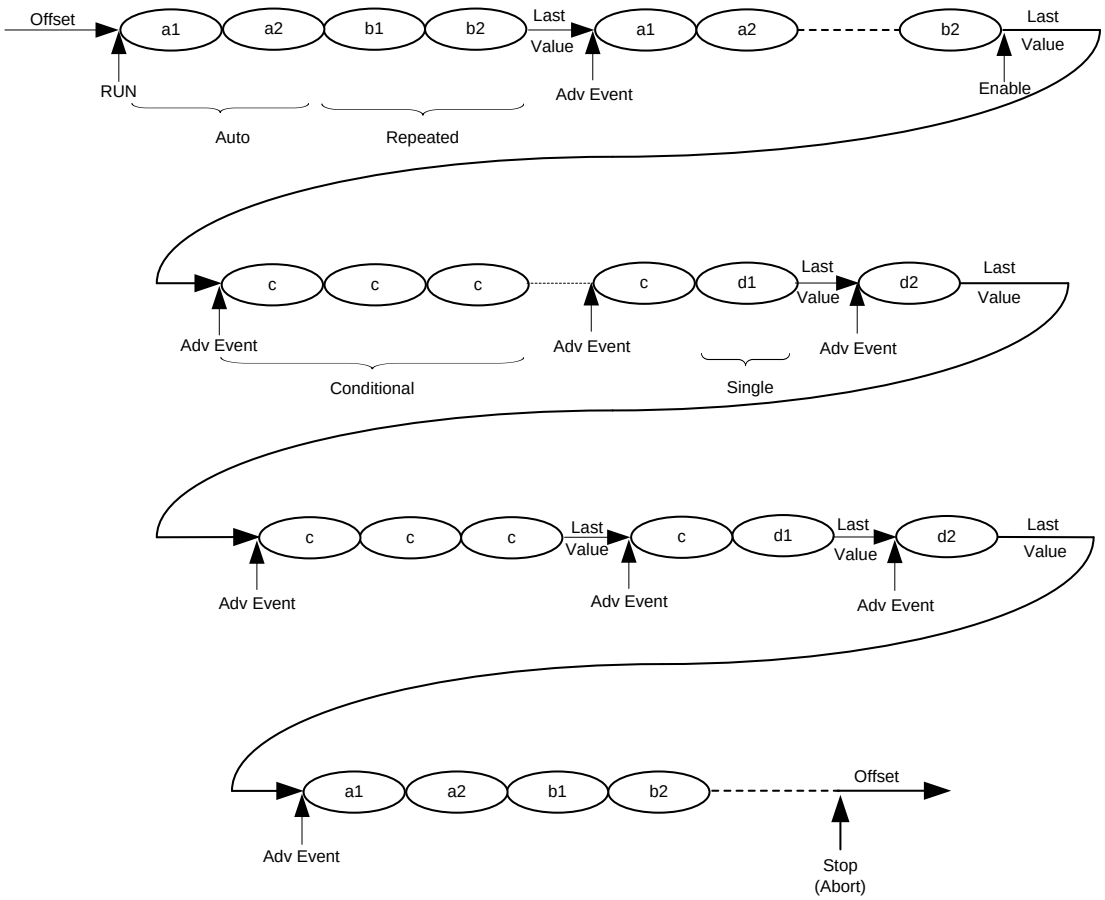


Figure 3-31: Trigger Mode Continuous

Trigger Mode Triggered Behavior is like self armed with an additional ENABLE. The enable is evaluated only once at the beginning. Later changes of this signal are ignored.

Trigger Mode Gated Behavior is like self armed with an additional ENABLE. The enable is evaluated only once at the beginning. Later changes of this signal are ignored.

3.9 Dynamic Sequencing

Dynamic Sequencing is a way to dynamically select segments/sequences to be played. The selection can be done by software or by the external dynamic input port. The time from selecting a new segment/sequence to the time the change is visible at the output is not specified and is dependent from the actually played segment's/sequence's end relative to arrival of the change event.

When using dynamic sequencing, the arm mode must be set to **self-armed** and all advancement modes must be set to **Auto**. Additionally, the trigger mode **Gated** is not allowed.

3.9.1 Dynamic Continuous

The selected segment or sequence is infinitely played until a new segment/sequence is selected which then is played instead. After a change request the actually selected segment/sequence is played until the end (including loop counts). Then the change towards the new segment or sequence is performed without any gap.

Limitations:

- The time between two change requests of waveforms must be bigger than the waveform length of the biggest waveform including the loop counts.
- The change delay from applying changes at the dynamic port to seeing them at the output is the trigger to output delay (see datasheet) plus 256 sync clock cycles minimum. Due to instrument internal functionality, this delay cannot be specified exactly and it is always possible that one more segment/sequence A is played before switching to B.

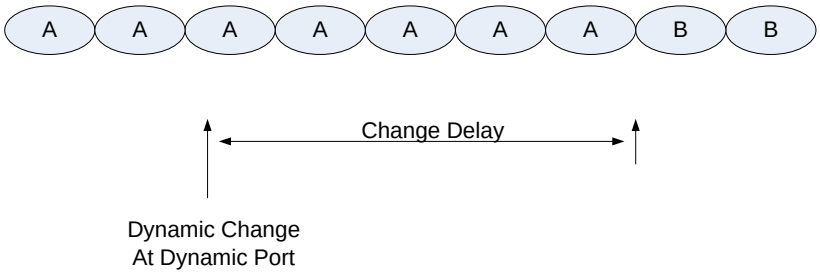


Figure 3-32: Dynamic Continuous

3.9.2 Dynamic Triggered

After having received a trigger, the selected segment or sequence is played (including loop counts). After having selected a new sequence/segment, this sequence/segment is played instead. Based on the timing relationship of the change request and the next trigger, it is possible that the actually selected (old) waveform is played one more time, before switching to the new one.

Limitations:

- The trigger period must be bigger than the waveform length of the biggest waveform including the loop counts.
- The change delay from applying changes at the dynamic port to seeing them at the output is the trigger to output delay (see datasheet) plus 256 sync clock cycles minimum. Due to instrument internal functionality, this delay cannot be specified exactly and it is always possible that one more segment/sequence A is played before switching to B.

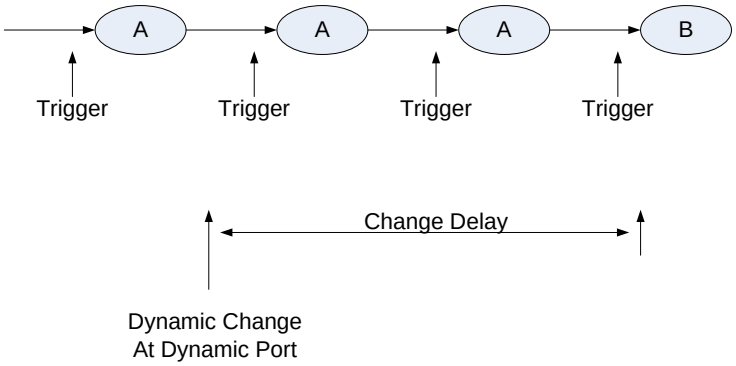


Figure 3-33: Dynamic Triggered

3.10 Idle Command Segments

For some waveform types, like e.g. radar pulses, huge pause segments with a static output are required between the real waveform segments. The gap between the real segments should be adjustable in a fine granularity

The idle command segment allows setting a pause between segments in a granularity that is smaller than the sync clock granularity. A minimum length of this pause is required (see section *7.21.6*). The idle command segment is treated as a segment within sequences or scenarios. There is no segment loop count but a sequence loop counter value is required for cases where the idle command segment is the first segment of a sequence.

The following table shows the granularity of the idle delay:

Table 3-3: Idle delay granularity

Mode	Idle Delay Granularity
High Speed (Option -12G)	1 DAC Output Sample
High Precision (Option -14B)	1 DAC Output Sample
INT_x3 (Option -DUC)	8 I/Q Sample Pairs
INT_x12 (Option -DUC)	2 I/Q Sample Pairs
INT_x24 (Option -DUC)	1 I/Q Sample Pair
INT_x48 (Option -DUC)	1 I/Q Sample Pair

Limitations:

- The logic that executes idle command segments uses some elements, which are not in sync clock granularity. To guarantee the trigger to sample output delay or the advancement event to sample output delay, these elements need to be reset before accepting new trigger or advancement events. This requires the waveform generation to be stopped for at least 3 sync clock cycles before being restarted by a trigger or an advancement event. A violation of this requirement leads to an unexpected output behavior for some sync clock cycles.
- The sync marker output of the corresponding channel is directly generated by the sequencer in the sync clock domain and is not shifted in sample granularity. So some jitter between sample output and sync marker output might be observed, when using idle command segments within a sequence or scenario.
- Multiple adjacent idle command segments are not allowed. If the playtime of one idle command segment is not sufficient, the overall required idle length can be separated into multiple idle command segments where a normal data segment providing the static idle value is put in between. Even this wouldn't be really necessary. One idle command segment (delay of up to 2^25 sync clock cycles) and one additional small segment (e.g. length: 10 * segment vectors, loop count: up to 2^32) would provide an idle delay of more than 200 seconds in high speed mode at 12 GSa/s and should be sufficient for most applications.

3.11 Limitations

3.11.1 Segment Length and Linear Playtime

Due to the type of memory technology and the implementation of the memory interface, every physical address jump within the sample memory will reduce the bandwidth at the memory interface. The drawing below shows such an address jump.

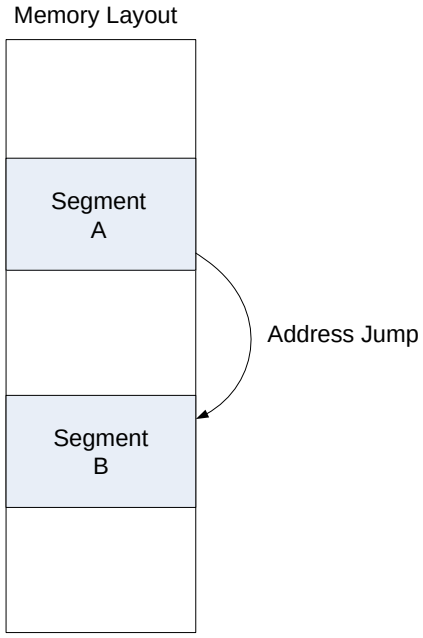


Figure 3-34: Address jump within segments

To put the density of address jumps below a limit, a minimum segment length of 257 sample vectors (big segment) is required. Small segments (256 vectors down to 5 vectors) are also possible but then the linear playtime requirement must be met.

Linear Playtime Requirement:

- The playtime of at least 257 sample vectors (257 sync clock cycles) must be placed in the sample memory in an ascending address order. When writing samples to a totally cleared memory, the order of segments is the order of how these segments are written to the memory.

Examples:

- One segment with 257 or more sample vectors (big segment)
- One segment with 129 vectors and a loop count of 2 (Loop count multiplies the segment length)
- Two segments with 126 and 131 vectors. (Multiple small segments are combined to meet the requirement)
- One segment of 5 vectors and one big segment. (One or multiple small segments which don't meet the linear playtime requirement by themselves, must be located in the memory in front of the next big segment)
- Any small segment with a conditional advancement causes the linear playtime requirement to be met automatically. The advancement event to exit the segment is delayed internally until the linear playtime condition is met. A status register signals any linear playtime violation.
- Any small segment with an advancement mode set to repeated or single causes the linear playtime requirement to be met automatically. The advancement event is delayed internally until the linear playtime condition is met. A status register signals any linear playtime violation.

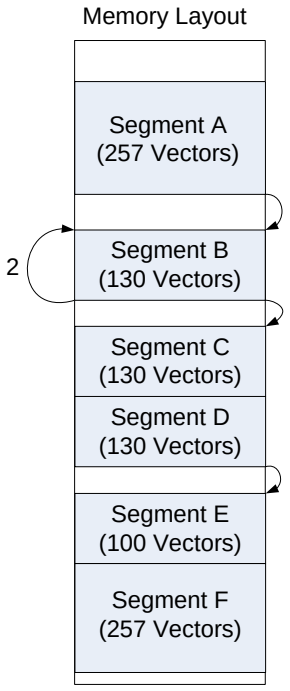


Figure 3-35: Linear playtime requirement

For the given example sequence the linear play time requirement is met. Segment A is a segment with a play time that is bigger than 256 vectors. Due to its loop count, segment B is also bigger than 256 vectors. Segment C and D are placed next to each other and the resulting length is 260 vectors. The small segment E is placed in front of a big segment.

4 Digital Up-Conversion

Required Option

This chapter describes the digital up-conversion capabilities of the instrument when having ordered the Option -DUC.

Parts of the functionality described in this chapter uses sophisticated sequencing capability of the M8190A. As a result, the functionality described in the sections *Configuration at Run-Time* and *Amplitude Scaling* require Option -SEQ.

4.1 Introduction

The following diagram shows the hardware structure of features for Digital Up-Conversion.

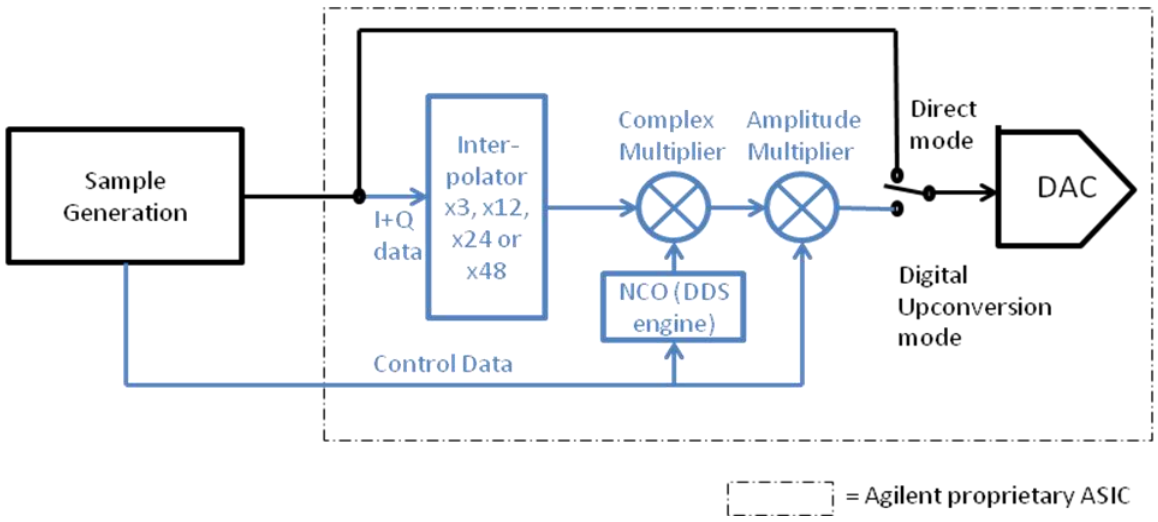


Figure 4-1: Hardware Structure of Features for Digital Up-Conversion

The M8190A provides a direct mode, which allows sending sample data from the sample memory directly to the output of the instrument. Additionally, it provides mechanisms to do IQ modulation internally (highlighted in blue in *Figure 4-1*). This includes the interpolation of IQ sample pairs from a low input data rate to the sampling rate of the instrument, the generation of a carrier frequency by a Numerically Controlled Oscillator (NCO) and the IQ modulation itself. Control data that is in time aligned to the sample data allows changing settings, like the NCO configuration, at run-time. Therefore, it is possible to change the carrier frequency, the phase and an amplitude multiplier in real time and sequence control. Complex operations, like a frequency sweep are also possible.

4.1.1 Direct Mode

In direct mode, (see *Option -14B, -12G*) the samples are sent directly to the DAC. IQ modulation can be implemented by:

- Using two channels providing I and Q data separately to an external analog IQ modulator. This setup is expensive and causes analog distortions.
- IQ modulation can be done by software.

4.1.2 Digital Up-Conversion Modes

In these modes (see *Option -DUC*), IQ sample pairs provided by the sample data generation are interpolated to the actually selected sample rate and are fed to the IQ modulator. The result is sent to the DAC. The benefits are:

- Efficient utilization of the sample memory, which is part of the sample generation unit.
- No distortions caused by an external analog IQ modulator.

4.1.3 Configuration at Run-Time

This feature is part of the option -DUC. The sequencer controls the configuration at run-time. This feature provides a possibility to change settings fast and with a defined timing relationship relative to the corresponding IQ data.

4.2 IQ Modulation

The instrument is able to provide an IQ modulated signal at its output, which is based on the IQ data provided by the customer, the selected interpolation mode, the carrier frequency and amplitude multiplication factor.

4.2.1 Interpolated Modes

The instrument provides interpolation filters that allow an interpolation by x3, x12, x24 and x48.

The available signal bandwidth of each interpolation filter is 0.8 x F_s where F_s is the input I/Q sample rate. F_s multiplied with the interpolation factor leads to the DAC sampling rate.

The following table shows examples for the maximum DAC sample rate of 7200 MSa/s:

Table 4-1: Mode dependent modulation bandwidth

Interpolation Mode	Max. Input Sample Rate	Max. Modulation Bandwidth
INT_x3	2400 MSa/s	1920 MHz
INT_x12	600 MSa/s	480 MHz
INT_x24	300 MSa/s	240 MHz
INT_x48	150 MSa/s	120 MHz

4.2.2 Markers in Interpolated Modes

Markers can be provided for each incoming IQ sample pair. It is not possible to perform marking of interpolated sample values.

For more details about markers, refer to chapter *Markers*.

NOTE The marker to sample output delay is different in the digital up-conversion mode compared to the corresponding value in direct modes (see datasheet).

4.3 Configuration at Run-Time

Required Option This section describes functionality which requires Option -SEQ.

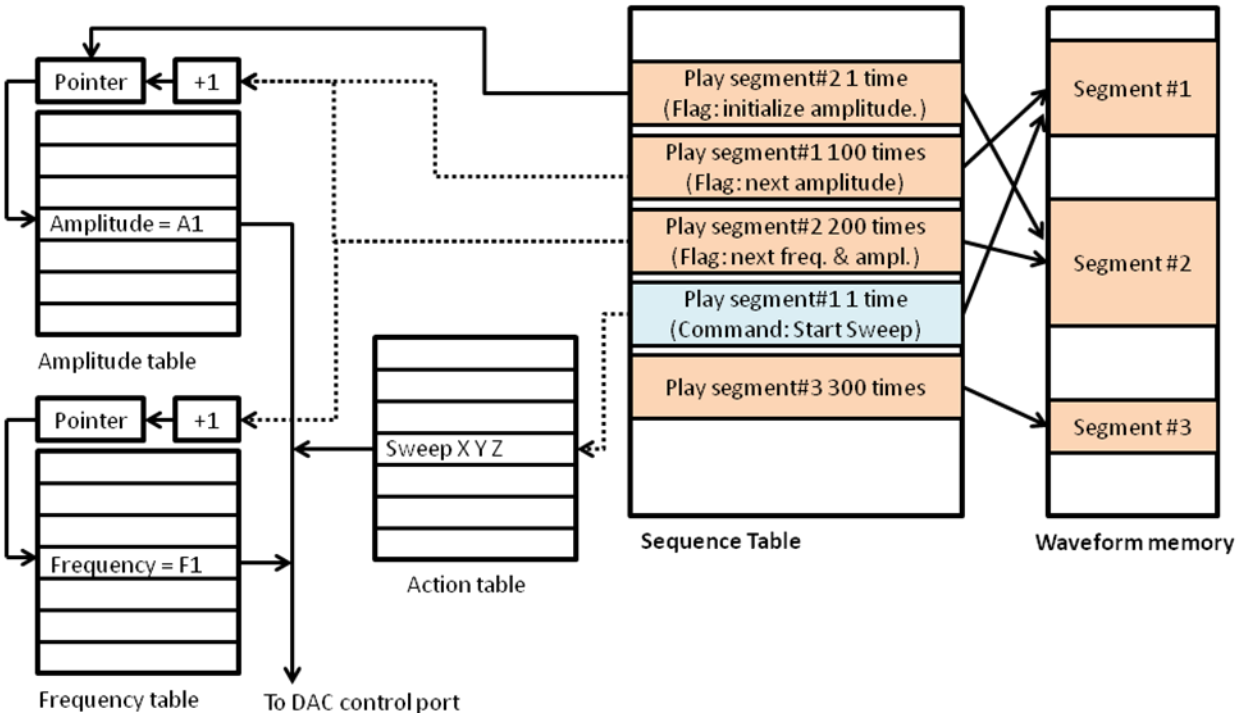


Figure 4-2: Mechanisms to Change Configuration at Run-Time

The drawing above illustrates the mechanisms of how the configuration of the instrument can be changed at run-time. The right side shows the sequence table and the waveform memory (see *Sequencing*). The left side shows tables containing configuration data provided by the user.

The following tables are available:

- Amplitude Table: stores multiplication factors for the amplitude multiplier (see *Figure 4-1*). For more details how to program an amplitude table, see *:ATABle Subsystem*.
- Frequency Table: stores frequency values for the NCO. For more details how to program a frequency table, see *:FTABle Subsystem*.
- Action Table: stores sets of complex commands. For more details how to program an action table, see *:ACTion Subsystem*.

The access to these tables is controlled by the sequencer of the instrument. For more details how to access the tables, see *:STABle Subsystem.*

4.3.1 Configuration using Standard Data Segments

The access to the amplitude and frequency table is possible via standard data segments.

Every standard segment can be armed with 4 flags (see :STABle Subsystem):

Table 4-2: Flags

Flag	Purpose
Initialize Amplitude Table Address	Initializes the address pointer of the amplitude table to zero and causes the amplitude table value of index zero to be transferred to the DAC. The new amplitude multiplication factor becomes active immediately.
Increment Amplitude Table Address	Increments the address pointer of the amplitude table by one and causes the amplitude table value of the new index to be transferred to the DAC. The new amplitude multiplication factor becomes active immediately.
Initialize Frequency Table Address	Initializes the address pointer of the frequency table to zero and causes the frequency table value of index zero to be transferred to the DAC. The new NCO frequency value becomes active immediately.
Increment Frequency Table Address	Increments the address pointer of the frequency table by one and causes the frequency table value of the new index to be transferred to the DAC. The new NCO frequency value becomes

active immediately.

Frequency and amplitude table are altogether independent. Therefore, it is allowed to change the amplitude multiplication factor and the NCO frequency together.

Initializing and incrementing a table offset at the same time does not make any sense and is therefore not allowed.

Each table has a storage capability of 32k entries.

When looping over a segment multiple times, increment and initialization flags are only evaluated once at the beginning of the first loop.

Frequency and amplitude changes are not aligned to the corresponding segment boundaries. The following table shows a rough timing relationship between the beginning of a segment containing increment or init flags and the corresponding change of the frequency or amplitude.

Table 4-3: Timing relationship for table access

Mode	Delay from Segment Start to Amplitude Change	Delay from Segment Start to Frequency Change
INT_x3	Approx. 0 – 24 Samples (varying from start to start)	Approx. 288 – 312 Samples (varying from start to start)
INT_x12	Approx. -24 Samples	Approx. 264 Samples
INT_x24	Approx. -24 Samples	Approx. 264 Samples
INT_x48	Approx. 0 Samples	Approx. 288 Samples

4.3.2 Configuration using Configuration Segments

A configuration segment is, like normal data segments, connected to sample data from the waveform memory but no segment loop count is available. Each configuration segment has a link to an entry of the action table and can initiate the execution of the corresponding action while playing the associated sample data. One action consists of one basic command or a so-called composite command. A composite command contains multiple combined basic commands and allows building complex operations.

A configuration change based on action table entries becomes active with the next sample marker. The minimum length of a configuration segment is 240 IQ sample pairs. Sample markers are not allowed within these first 240 IQ sample pairs.

Each command has an individual length. This length represents the amount of space required to store such a command in the action table, which has an overall size of 32k.

Table 4-4: Commands with their individual length

Commands	Length
Carrier Frequency	7
Amplitude	5
Phase Reset	5
Phase Offset	4
Phase Bump	5
Sweep Rate	7
Sweep Run	4
Sweep Hold	4
Sweep Restart	4
Set Carrier Frequency With Amplitude	9
Set Carrier Frequency with Known Phase	9 - 11
Start Frequency Sweep	8 - 14
Start Frequency Sweep with Known Phase	10 - 16
Stop Frequency Sweep and Switch To New Frequency	8 - 10

4.3.2.1 Basic Commands

The following basic commands are available.

4.3.2.1.1 Carrier Frequency

This command allows changing the carrier frequency at run-time.

4.3.2.1.2 Amplitude

This command allows setting the amplitude at run-time.

4.3.2.1.3 Phase Reset

This command allows setting the phase to a specified absolute value. The following picture shows an example. The phase value of the green channel is set to zero, the phase of the purple channel is set to +0.25.

Figure 4-3: Phase Reset

4.3.2.1.4 Phase Offset

This command allows setting an offset to the current phase. The purple channel in the following picture shows a phase offset of zero. The green channel shows a phase offset of 0.25. The phase offset is a constant value added to the actual phase. It does not affect the NCO counter itself. Therefore, the phase offset value could be set back to zero again.

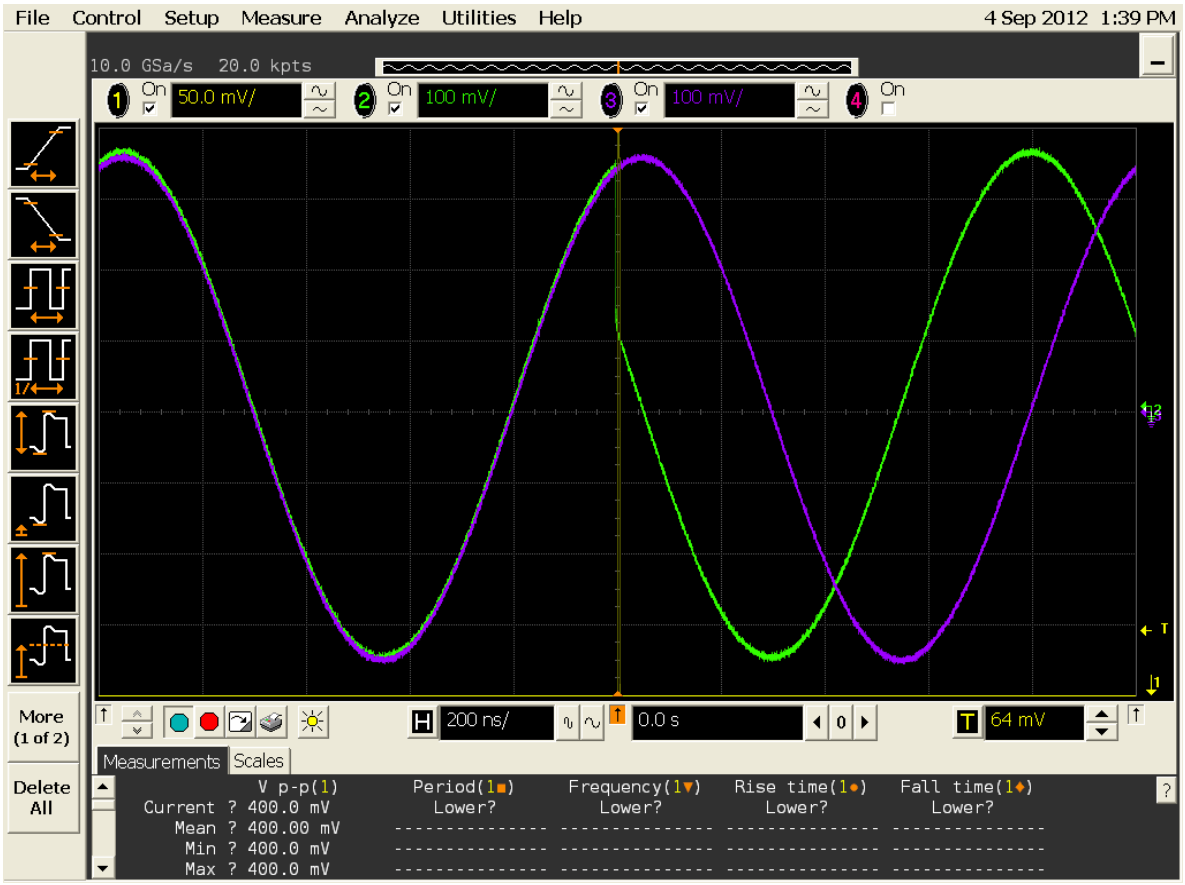


Figure 4-4: Phase Offset

4.3.2.1.5 Phase Bump

The phase bump affects the NCO counters itself. The following picture shows a phase bump of 10% (0.10). A following phase bump of again 10% will generate a phase offset of 20%. After this, a following phase bump of -20% (-0.2) would install the original phase again.

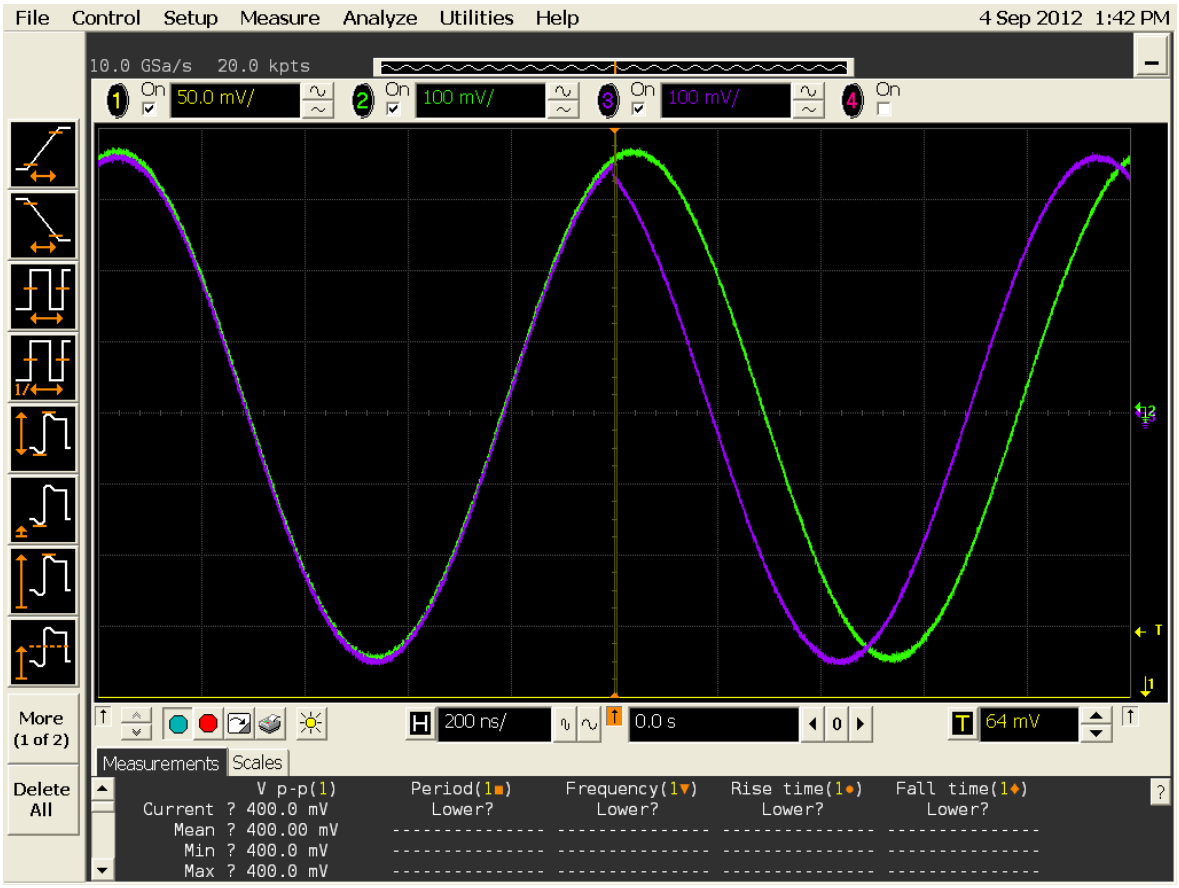


Figure 4-5: Phase Bump

4.3.2.1.6 Frequency Sweeps

Sweep Rate: Sets the sweep rate to a specific value.

Sweep Run: Starts a frequency sweep.

Sweep Hold: Stops a frequency sweep.

Sweep Restart: Stops a currently running frequency sweep and starts it again with the start conditions of the previous sweep.

The following picture shows a frequency sweep (sweep rate: 500 kHz/us). After a few clock cycles, the sweep is stopped again, the frequency is not increasing anymore.

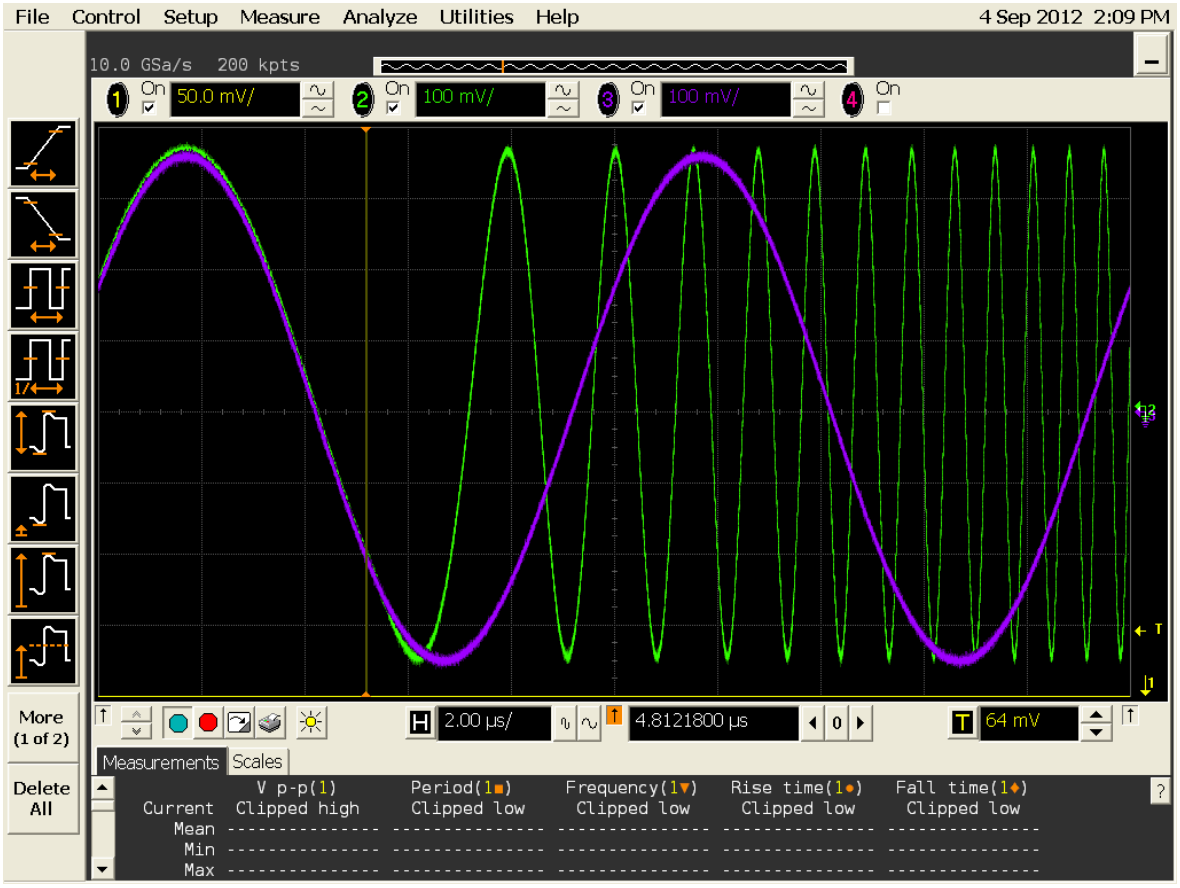


Figure 4-6: Frequency Sweeps

4.3.2.2 Composite Commands

Basic commands can be grouped together to form more complex “composite commands”. Some of these commands have optional parameters shown in enclosing [ ].

The composite commands consist of a list of basic commands. The following examples are the supported ones. Other combinations may also be possible, but only the given examples are supported and also tested.

4.3.2.2.1 Set Carrier Frequency With Amplitude

Used basic commands:

Carrier Frequency

Amplitude

In the following picture, the frequency and amplitude is changed twice.

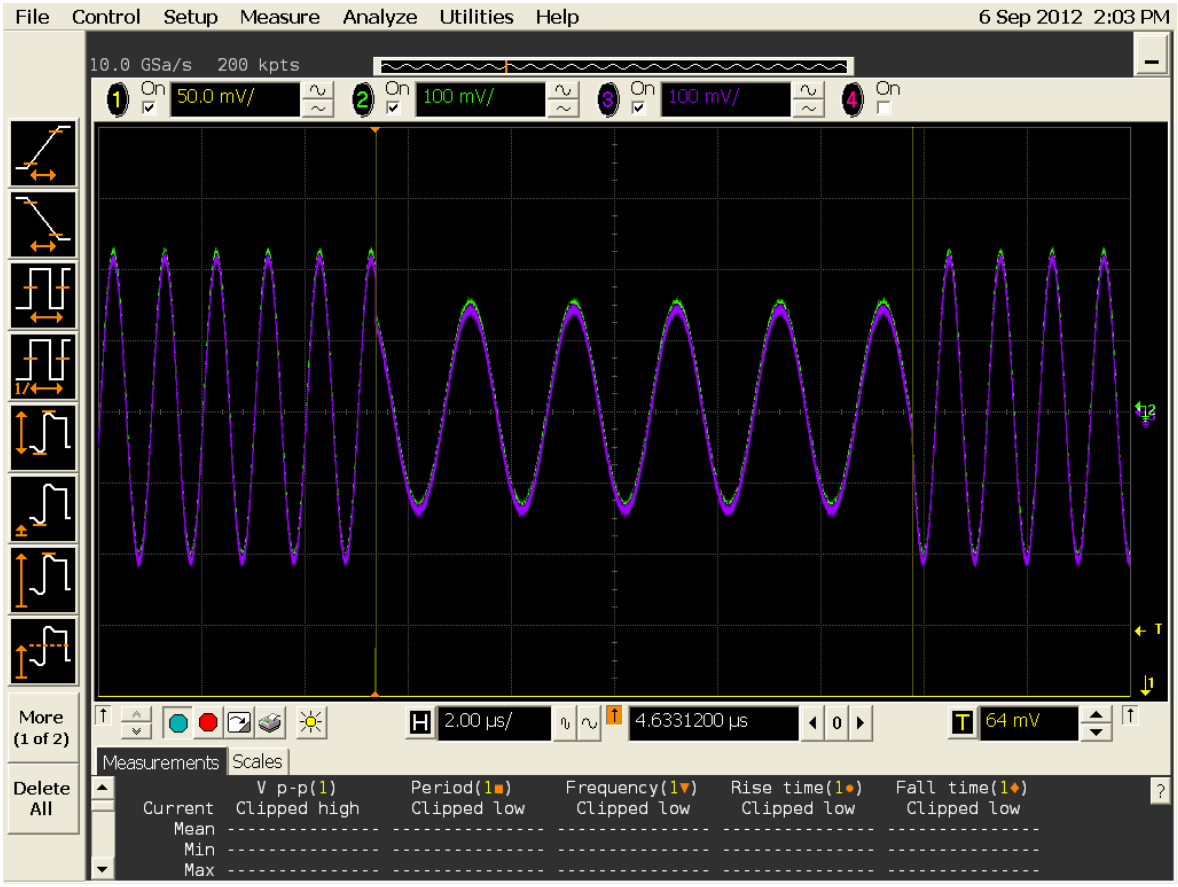


Figure 4-7: Frequency and amplitude changes

4.3.2.2.2 Set Carrier Frequency With Known Phase

Used basic commands:

Carrier Frequency

Reset Phase

optional: [Amplitude]

The following picture shows again a frequency and amplitude change. Additionally to the previous one, the phase is also modified. The first change sets the phase to 0, the second to 0.5 cycles.

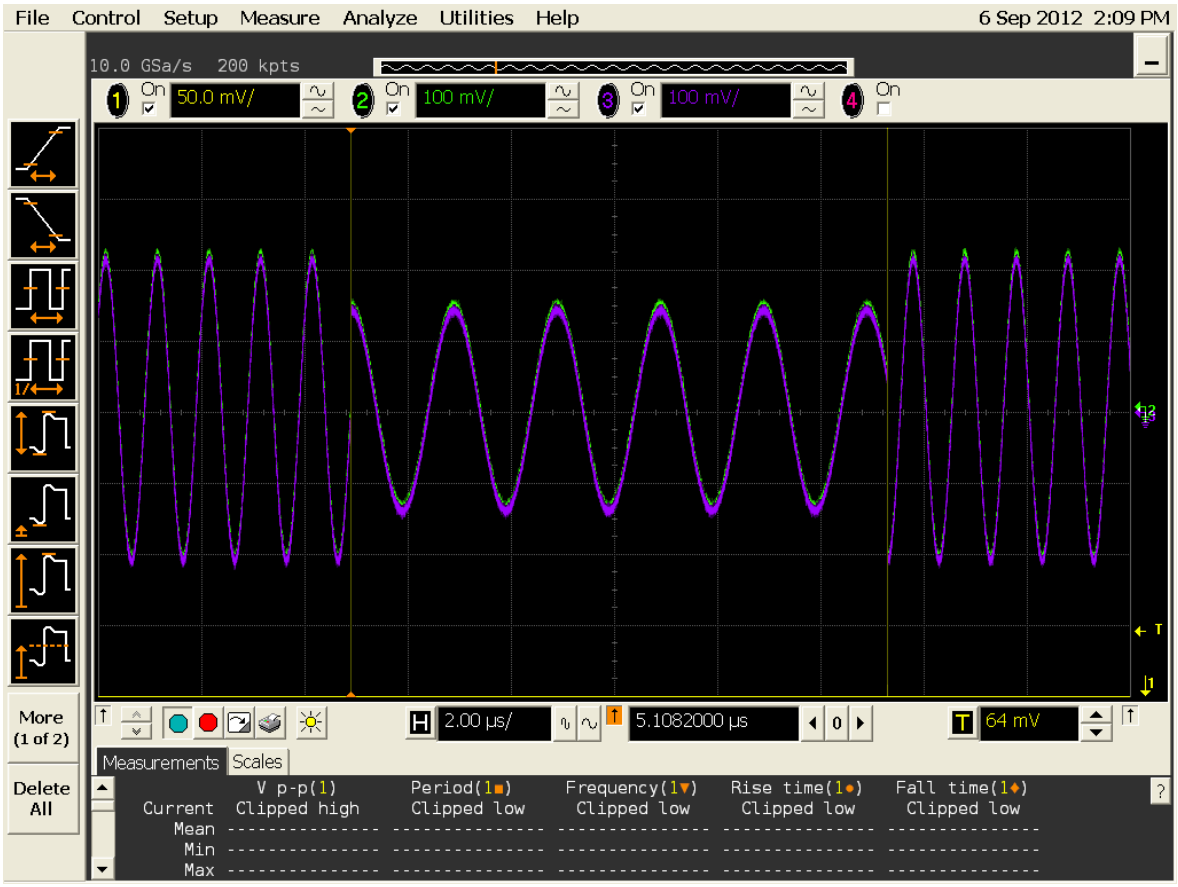


Figure 4-8: Frequency and amplitude changes

4.3.2.2.3 Start Frequency Sweep

Used basic commands:

Carrier Frequency

optional: [Sweep Rate]

optional: [Amplitude]

Sweep Run

The following picture shows a start and stop of frequency sweep. Additionally, the amplitude is changed, as well. Together with the stop, the frequency is set back to the start value.

Figure 4-9: Start and stop of frequency sweep

4.3.2.2.4 Start Frequency Sweep Known Phase

Used basic commands:

Carrier Frequency

optional: [Sweep Rate]

optional: [Amplitude]

Phase Reset

Sweep Run

In the following example, the sweep is started with a well-defined phase value of zero.

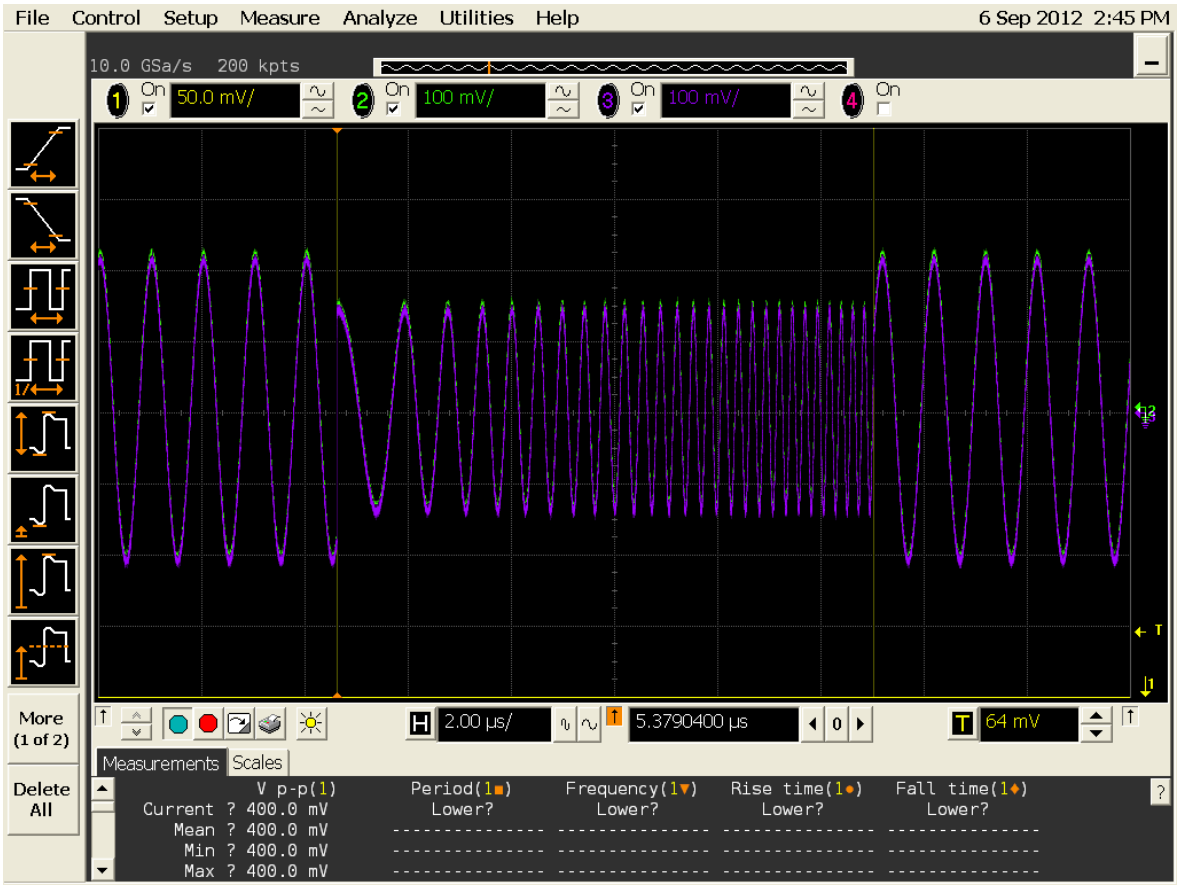


Figure 4-10: Frequency sweep

4.3.2.2.5 Stop Frequency Sweep And Switch To New Frequency

Used basic commands:
Sweep Hold
Carrier Frequency
optional: [Amplitude]

4.3.3 Sample-Aligned Configuration Using Markers

Configuration changes based on the frequency or amplitude table are not exactly aligned to the sample data and may vary for each change by some samples.

For more accurate and exactly reproducible results, the action table needs to be used instead. All configuration changes, based on this mechanism become active with the next sample marker. So a sample marker can be used to select a well-defined position relative to the sample data at which a configuration change becomes active.

The following table shows the accuracy of configuration changes depending on the interpolation mode when using the action table:

Table 4-5: Configuration Accuracy

Interpolation Modes	Configuration Accuracy (In IQ Sample Pairs)
INT_x3	8
INT_x12	2
INT_x24	1
INT_x48	1

4.4 Amplitude Scaling

Required Option This section describes functionality which requires Option -SEQ.

The amplitude multiplier allows to scale the amplitude of the DAC output signal (see *Figure 4-1*).

The chain of interpolation filters and the IQ modulator could increase the signal value, which then exceeds the valid DAC range. This leads to a clipping of samples. This effect depends on the input data and therefore cannot be handled automatically.

The instrument is able to detect clipping and signals it to the user by the status register or a dialog box in the soft front panel. Clipping could be prevented by using scale factor values of 1.0 or smaller. This scale factor value reduces the output amplitude relative to the direct mode to approx. 37.6%.

It is possible to increase the scale factor up to 2.828, but depending on the input IQ sample sequence this can result in signal clipping. The value can be separated into two multiplication factors:

- Factor 2.0 is related to the interpolation filters and is dependent from the input sample sequence.
- Factor 1.414 is related to the IQ modulators. The maximum I value (1.0) and the maximum Q value (1.0) causes a resulting vector with a length of sqrt (2)≈1.414.

So whenever the user can guarantee that the sum of I and Q doesn't exceed the unit circle, it is safe to set the scale value to 1.414 and increase the overall amplitude (see section *7.24.4*).

There are 3 possibilities to scale the DAC output signal:

4.4.1 Scaling by using the amplitude table

This mode allows using different values within one sequence. The range of the scale factor is from 0.0 to 1.0.

4.4.2 Scaling by using the action table

This mode allows using different values within one sequence. The range of the scale factor is from 0.0 to 1.0.

4.4.3 Scaling by using the SCPI command or the Soft Front Panel

A scale factor of 1.0, which is a "safe" value in terms of clipping, reduces the overall possible output amplitude to approx. 37.6%. Clipping is highly depending on IQ input data provided by the user. Therefore, this 3rd mechanism allows scale factor values in the range from 0.0 to 2.828 and allows using as much as possible from the available DAC range. The user is responsible to select a proper scale value depending on his input data.

Additionally, this scale factor input is automatically multiplied with the values provided by the action table or amplitude table. So, with e.g. a scale factor set by the soft front panel to the value of 1.5 and a value of 0.5 provided by the amplitude table, the resulting output scale factor that is written to the amplitude multiplier is 0.75.

So scaling by the tables is used to scale segments relative to each other e.g. play one segment multiple times with individual output amplitudes. An overall adjustment of the output amplitudes (valid for all segments together) can be done using the SCPI (see section *7.24.4*) command or the soft front panel.

4.5 Coarse Delay and Digital Up-Conversion

In direct modes, there is a coarse delay available that allows delaying the whole channel by multiple clock cycles (see section *7.8.3*). This can be done at run-time without stopping and restarting the instrument. In the digital up-conversion modes, this coarse delay is still available, but it cannot be changed at run-time. A stop and restart is required.

4.6 Doublet Mode and Digital Up-Conversion

The doublet mode (see section *1.5.2*) is also available for the digital up-conversion modes.

The frequency band is mirrored at the half of the sample frequency (7.2 GHz max). The content of the first Nyquist band (0 – 3.6 GHz max.) is transferred to the second Nyquist band (3.6 GHz to 7.2 GHz max.).

---

5 Streaming

This chapter describes the streaming capabilities of the M8190A.

5.1 Introduction

The streaming feature of the M8190A allows re-loading the sample memory while being in the run mode. This capability provides a method to generate waveforms with an infinite playtime. Streaming is possible using the high speed, high precision or DUC mode. The most efficient way is to use the DUC mode (IQBIN data format) The streaming feature requires the M8190A software version 3.0 or later.

Streaming is supported by the following two modes:

- *Dynamic Mode*
- *Ring Buffer Mechanism*

5.2 Streaming Implementation Using Dynamic Modes

The dynamic modes (refer to the section *3.9*) allow switching between segments (Arbitrary Mode) or sequences (Sequence Mode) using the external dynamic input port or by the software. A continuous or triggered execution is possible.

Since the M8190A software version 3.0, it is possible to modify the content of the sample memory when having selected one of the dynamic modes. Therefore, all segments or sequences that are currently not in use can be changed in run mode.

The following rules apply for implementing streaming using dynamic modes:

- The sample data can be changed in run mode. Sequencing controls the information such as loop counters, which cannot be modified in run mode.
- Changing the content of segments or sequences, which are currently executed or which are already selected by the dynamic port or by software to be executed next, is not allowed.

NOTE The hardware or software is not able to check this limitation. Obeying this rule is the responsibility of the user. In order to meet this rule, the user can query the segment number that is currently played by the M8190A.

The dynamic modes have some limitations. The main problem for streaming applications is the fact that a pre-defined timing relationship is not always guaranteed when switching from one sequence to another sequence. Therefore, especially in the continuous modes, it might happen that the current sequence is played one or more times before switching to the next sequence. This means the exact number of repetitions of a certain sequence cannot be determined. I.e. streaming implementation using dynamic modes is not entirely deterministic. A streaming application with an entirely deterministic output behavior is described in the chapter that follows.

5.3 Streaming Implementation Using the Ring Buffer Mechanism

The dynamic sequencing allows switching between the segments or sequences dynamically without any interruptions. The change from one sequence to another, needs to be initiated by the software or by an externally applied change request using the dynamic control input.

The ring buffer mechanism is using one single sequence where the change request is applied automatically by hardware that restarts the sequence itself again. This hardware controlled mechanism allows changing sample data of this sequence while being executed.

5.3.1 Requirements

5.3.1.1 Instrument Settings

The special instrument settings that are required for implementing a ring buffer mechanism, are listed below:

- Dynamic sequencing is required in either continuous or triggered mode (refer to the sections *7.8.6*, *7.8.7* and *7.21.10*)
- A special streaming mode needs to be selected (refer to the section *7.21.15*).

5.3.1.2 Sequence Parameters

The special sequence settings that are required for implementing a ring buffer mechanism, are listed below:

- The overall minimum length of the sequence needs to be 1024 sequence vectors.
- A minimum of five data segments is required. The overall length of the second data segment down to the last data segment must be more than 512 vectors.
- The first sequence table entry must be at address zero and must be a data segment. Idle delay segments (refer to the section *7.20*) are not allowed.
- Data segments should be located linearly in the sample memory (refer to the section *7.22.12)*.
- The last segment of the sequence must be a data segment. Idle delay segments (refer to the section *7.20*) are not allowed.
- For each data segment, command segments for action table, frequency table or amplitude table access are also allowed.

5.3.1.3 Streaming Modes

The streaming modes are of following two types:

- Continuous streaming
- Triggered streaming

5.3.1.3.1 Continuous Streaming

Continuous streaming requires the continuous dynamic sequence mode. All segments, as well as the used sequence need to be setup with the advancement mode set to AUTO.

The following diagram illustrates the continuous streaming mode.

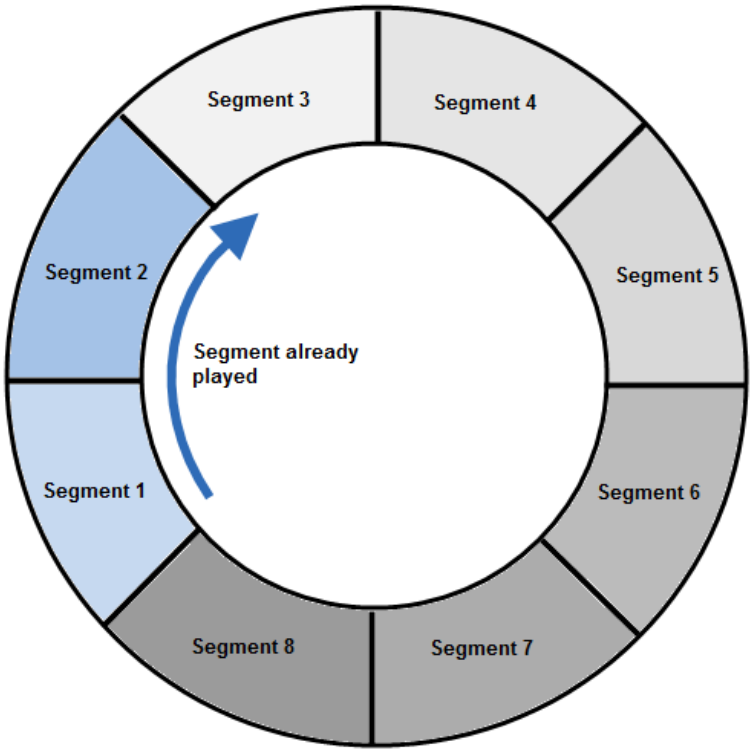


Figure 5-1: Continuous Streaming

5.3.1.3.2 Triggered Streaming

Triggered streaming requires the triggered dynamic sequence mode. In this mode, a trigger is required for each time the sequencer is passing the start of the sequence. Additional interruptions can be achieved by using the SINGLE or REPEATED mode for other segments of the sequence. The advancement mode of the sequence itself needs to be set to AUTO. The advancement mode of the last segment needs to be AUTO. In triggered streaming, the instrument must be setup correctly to generate events with triggers (refer to the section *7.9.1*).

The following diagram illustrates the triggered streaming mode.

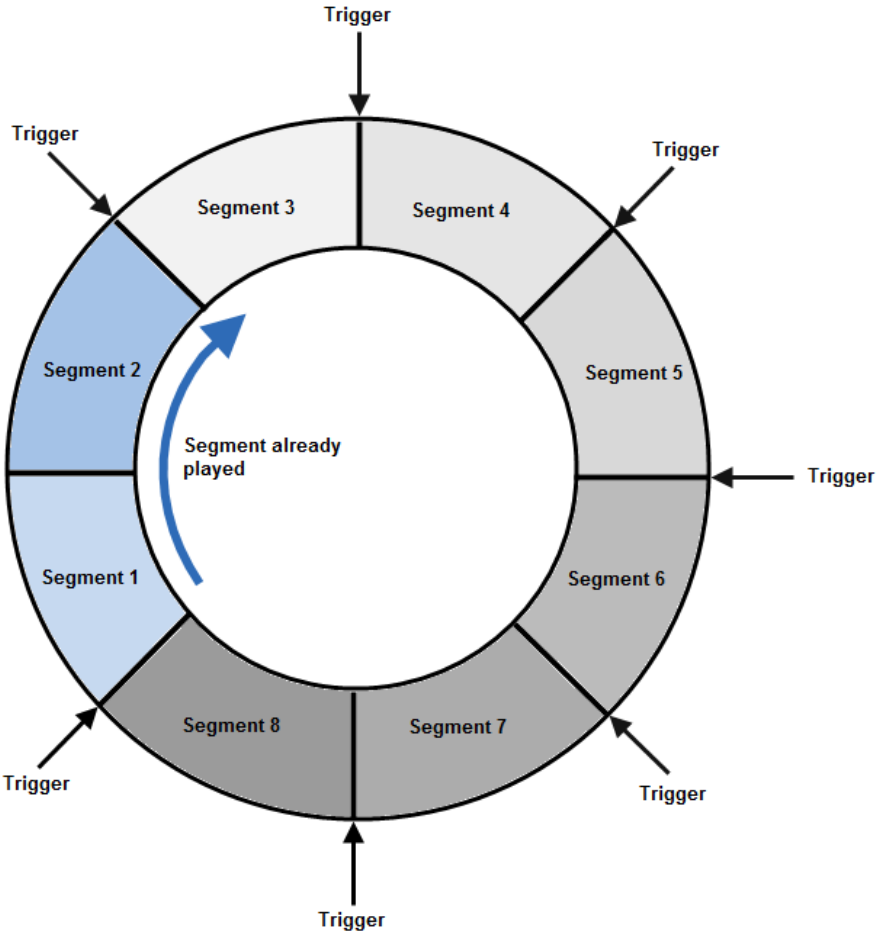


Figure 5-2: Triggered Streaming

5.3.2 Theory of Operation

The following drawing shows a sequence used as ring buffer.

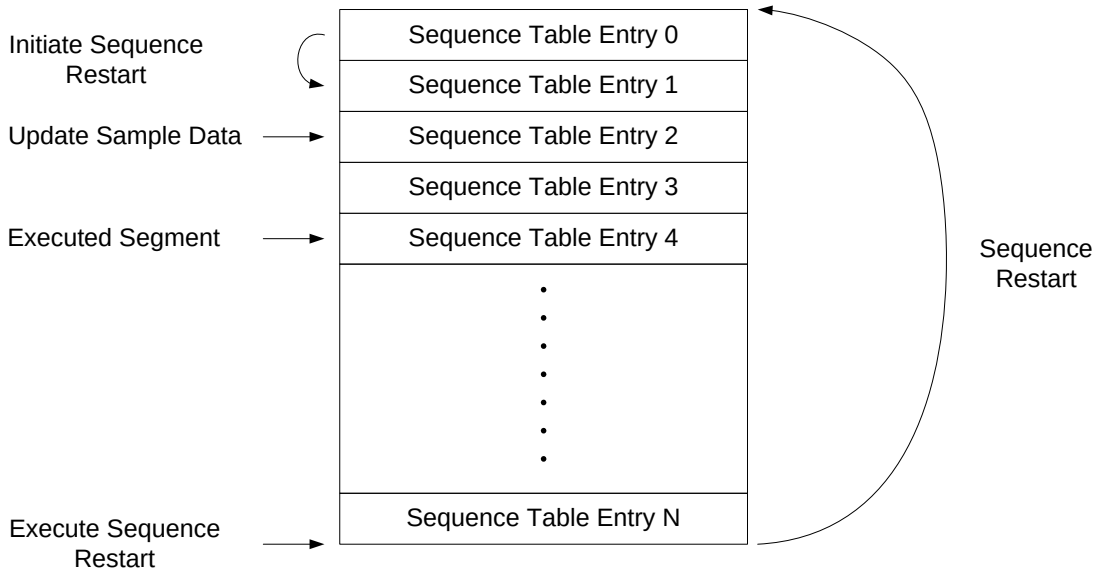


Figure 5-3: Sequence used as Ring Buffer

The sequence starts at sequence table entry 0 and consists of more than 5 sequence table entries. The automated restart is initiated when starting to play the second segment. The restart itself is done at the end of the sequence. In this example, the sample memory of the segment related to sequence table 2 is written, while sequence table entry 4 is executed.

Streaming requires reloading the sample memory while executing to other parts of the memory. It must be guaranteed that only those parts are overwritten, which have already been played.

In order to handle this issue, the following two things are required:

1. The controlling software must know exactly the segment that is currently in use. The state of the sequencer, including the currently executed segment can be read using an API call (refer to the sections *7.7.7* and *7.21.9*).
2. A minimum distance of 512 sequence vectors between the sample data of the currently executed segment and the currently modified sample data is required.

5.3.3 Examples

One problem for streaming is to provide sample data at a sufficient speed.

An example program is part of the software installation which can be found at the following installation path (standard installation):

C:\ Program Files (x86)\Keysight\M8190\Examples\Cpp_Streaming

More details on how to use this example is available in the source code and the accompanying ReadMe.txt.

The following two modes of operation are explained:

5.3.3.1 Algorithmic Sample Data Generation

The example program itself is modifying sample data on the fly. Download data rates of about 670 MByte/s are possible, providing a modulation bandwidth of about 140 MHz. For this test, the example program has been executed on a PC from Dell (Precision T7600).

5.3.3.2 Sample Data Storage Using a RAID System

This example shows the use of streaming with data stored on hard disks. A RAID system is used to generate an adequate read performance.

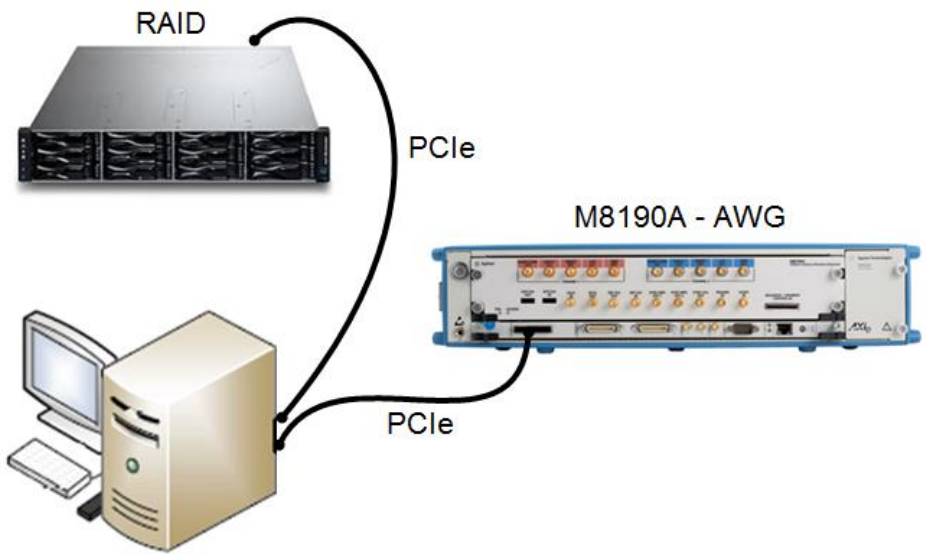


Figure 5-4: Sample Data Storage Using a Raid System

The use of RAID systems allows a playtime that is only limited by the size of the hard disks. The measured download rate with a 16 TByte RAID system from JMR Electronics (Model AGIL-G4-16T) connected to a Dell Precision T7600 PC is around 620 MByte/s providing a modulation bandwidth of around 130 MHz.

6 Markers

6.1 Introduction

The instrument provides output signals with a defined timing relationship to the output sample stream. These signals are called markers.

There are two different types of markers:

- Sample Markers
- Sync Markers

The details of these markers are explained in the sections that follow.

6.2 Sample Markers

Sample markers can be used to mark individual samples.

The following table shows the marker granularity:

Table 6-1: Marker granularity

Mode	Granularity in DAC Samples
High Speed (Option -12G)	48
High Precision (Option -14B)	40
INT_x3 (Option -DUC)	24
INT_x12 (Option -DUC)	24
INT_x24 (Option -DUC)	24
INT_x48 (Option -DUC)	48

The marker width of one marked sample is fixed to the marker granularity. Multiple consecutive marked samples generate one output marker, which is expanded to cover all marked bits. The length of this marker is quantized to the marker granularity.

The sample marker is always aligned to the sample data (see *Figure 1-3*). This alignment varies depending on the mode (see datasheet). Delay adjustments also change the delay of the sample marker.

The following drawing shows an example. The dots represent the marked samples.

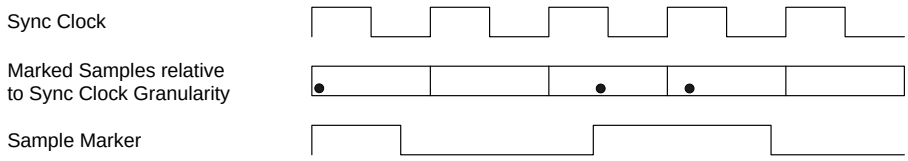


Figure 6-1: Samples marked by sample marker

6.2.1 Pause Between Multiple Marked Samples

Due to the quantization of markers, a minimum pause between blocks of marked samples is required to see the gap at the sample marker output.

Table 6-2: Pause between multiple marked samples

Mode	Minimum Gap Size
High Speed (Option -12G)	128 Samples
High Precision (Option -14B)	96 Samples
INT_x3 (Option -DUC)	8 IQ Sample Pairs
INT_x12 (Option -DUC)	2 IQ Sample Pairs
INT_x24 (Option -DUC)	1 IQ Sample Pairs
INT_x48 (Option -DUC)	1 IQ Sample Pairs

The table above shows the minimum gap size between blocks of multiple marked samples/IQ sample pairs. If this condition is violated, it might be possible to get one “combined marker” instead two individual markers.

6.2.2 Sample Marker in Looped Segments

Sample markers start always at the correct position. However, whenever loops are involved in a block of continuously marked samples, the end position of the marker is not well defined and may vary for some samples according to the following table.

Table 6-3: Sample Marker in Looped Segments

Sample Rate	High Precision Mode	High Speed Mode	Interpolated Modes
Down to 1Gs	40 Samples	48 Samples	Up to 8 IQ Sample Pairs
0.500Gs - 0.999Gs	20 Samples	24 Samples	-
0.250Gs - 0.499Gs	10 Samples	12 Samples	-
0.125Gs - 0.249Gs	5 Samples	6 Samples	-

6.2.3 Sample Marker in Segments which are Addressed Offset Based

The instrument provides a mode where sequence table entries address the content of segments by offset. Whenever a sequence table entry accesses a segment with an offset not equal to zero (not starting from the beginning of the segment), this may result in unexpected sample marker behavior like gaps when having marked the samples of the data vector referenced by the offset value.

6.3 Sync Markers

Additionally to the high-speed marker, each channel provides also a second marker called sync marker. This sync marker can only be set in sync clock granularity which is equal to the sample vector granularity.

The sync marker is an output of the pattern generation. For details, refer to section *1.5.1*. Delay adjustments don not change the delay of the sync marker.

The following drawing shows an example. The dots represent the marked sync clock vectors.

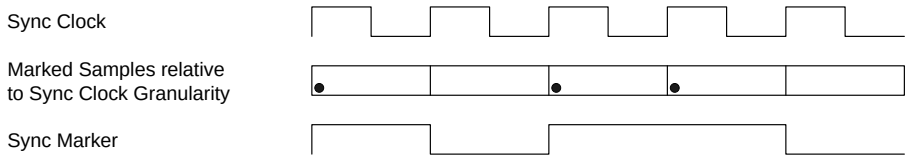


Figure 6-2: Marked sync clock vectors

7 General Programming

7.1 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:

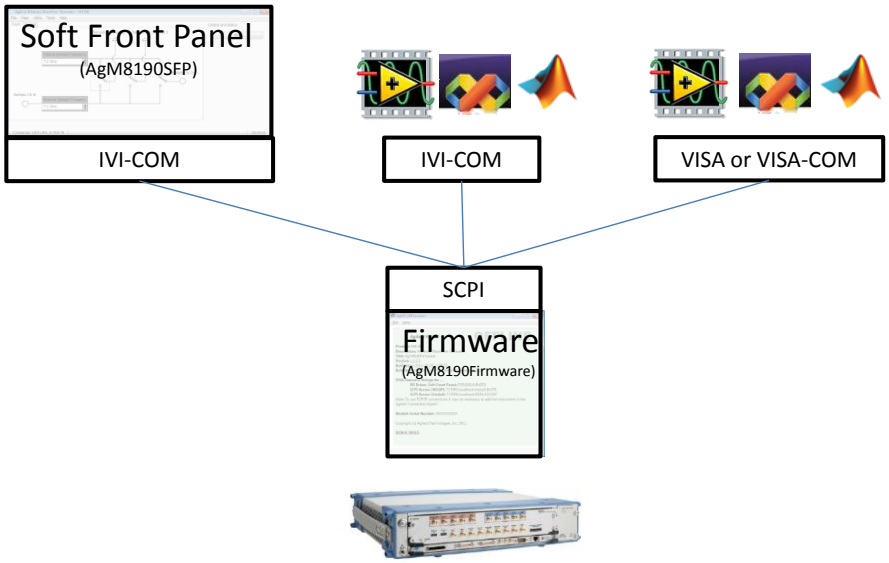


Figure 7-1: M8190A Programming

Firmware talks to the actual M8190A module using a PCI express connection. I/O to the module is done using VISA library of Keysight I/O library. Addressing is done with PXI resource strings, e.g. “PXI36::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 Keysight 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...*. In addition, you can verify changes done from a remote program with *View → Refresh*.

7.2 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 "PXI36::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.

7.3 SCPI Programming

Introduction

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

VXI-11**:** The Visa Resource String is e.g. "TCPIP0::localhost::inst0::INSTR".

- **HiSLIP:** this protocol is recommended. It offers the functionality of VXI-11 protocol with better performance that is near socket performance. Visa Resource Strings look like "TCPIP0::localhost::hislip0::INSTR". The correct resource string is shown in the Ag8190Firmware main window. To use the HiSlip protocol an I/O library such as the Keysight I/O Libraries Suite must be installed. Since the protocol is new it might not be supported by the installed I/O library. The Keysight I/O Libraries Suite 16.1 and above supports it. However, the Keysight I/O Libraries Suite 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.

NOTE AgM8190Firmare.exe must be started prior to sending SCPI to the instrument. (See *AgM8190Firmware.exe*)

7.3.1 AgM8190Firmware.exe

Before sending SCPI commands to the instrument, the firmware (AgM8190Firmware.exe) must be started. This can be done in the Windows Start menu (*Keysight → M8190 → M8190*) or by starting the Soft Front Panel.

If AgM8190Firmware.exe is started implicitly by using the Soft Front Panel or IVI Driver, it is minimized to the notification area on the Windows Task Bar. You can open it to see the VISA Resource String for the different connection types.

7.3.1.1 Command Line Arguments

(See *Communication* for details about /s, /t, /i, /AutoID, /NoAutoID, /FallBack).

Table 7-1: Command Line Arguments

Option	Description
/Minimized	Start minimized.
/Hide	Start without application window; only show notify icon.
/s socketPort	Set the socket port at which the firmware waits for SCPI commands
/t telnetPort	Set the telnet port at which the firmware waits for SCPI commands
/i instrumentNumber	Set the instrument number (instN, hislipN) at which the firmware waits for SCPI commands
/AutoID	Automatically select ports and number for the connections (default behavior).
/NoAutoID	Disable the default behavior; i.e. do not automatically select ports and number for the connections.
/FallBack	Try to find unused ports and number if starting a server fails.
/r resourceName	Visa PXI resource string of the module to connect to, e.g. PXI12::0::0::INSTR

7.3.1.2 Communication

Depending on the command line arguments /s, /t, /i, /AutoID, /NoAutoID, /FallBack, the firmware starts several servers to handle SCPI commands. (Refer the table above.)

/s, /t, /i: If -1, don't start the respective servers

- Defaults:
  - Socket port: 5025 (e.g. TCPIP0::localhost::5025::SOCKET)
  - Telnet port: 5024
  - HiSLIP, VXI-11.3: 0 (e.g. TCPIP0::localhost::hislip0::INSTR, TCPIP0::localhost::inst0::INSTR)

/FallBack : If starting a server fails because of a conflict, try using another port or number

- HiSLIP, VXI-11.3: increase the index until a server can be started successfully
- Socket, Telnet: start with port 60000, then increase it until the servers can be started successfully. If neither socket nor telnet is disabled the firmware tries to start the servers on two consecutive ports

  (socket port = telnet port + 1)

/AutoID : Automatically select ports and number for the connections, which are unique per instrument.

- This is the default behavior; it is not necessary to specify this argument on the command line.
- If only one AXIe module is connected to this PC and it is an M8190 module, first try to use the command line arguments /s, /t, /i or their respective default values if they are not specified. If starting the servers fails, proceed with the steps below.
- /s, /t, /i are ignored (unless they are -1 and a server is disabled)
- If the firmware detects more than one AXIe module, use a special mechanism to obtain a number for the HiSLIP and VXI-11.3 servers, which makes sure that the firmware uses always the same VISA resource string per module
- The socket and telnet port are then calculated from the HiSLIP index:
  - telnet port = 60000 + 2 * <HiSLIP index>
  - socket port = 60000 + 2 * <HiSLIP index> + 1

  Note: Ports may already be in use by Windows or other applications, so they are not available for M8190A.

/NoAutoID : Do not automatically select ports and number for the connections, use the values specified with /s, /t, /i or their respective default values instead.

If both /NoAutoID and /AutoID are specified, /AutoID overrides /NoAutoID.

---

[1] The first port not assigned by IANA is 49152 (IANA, Internet Assigned Numbers Authority, http://www.iana.org)

7.4 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.
```

7.5 System Related Commands (SYSTem Subsystem)

7.5.1 :SYSTem:ERRor[:NEXT]?

Command `:SYST:ERR?`

Long `:SYSTem:ERRor?`

Parameters None

Parameter Suffix None

Description 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, when the power is cycled, or when the firmware is re-started.

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):

error number,"Description", e.g.

-113,"Undefined header".

Example Query

```
:SYST:ERR?
```

7.5.2 :SYSTem:HELP:HEADers?

Command :SYST:HELP:HEAD?

Long :SYSTem:HELP:HEADers?

Parameters None

Parameter Suffix None

Description The HEADers? query returns all SCPI commands and queries and IEEE 488.2 common commands and common queries implemented by the instrument. The response is a <DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA> element. The full path for every command and query is 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 contains all the nodes from the root. The <SCPI program mnemonic> contains the node in standard SCPI format. The short form uses uppercase characters while the additional characters for the long form are in lowercase characters. Default nodes are surrounded by square brackets ([]).

Example Query
:SYST:HELP:HEAD?

7.5.3 :SYSTem:LICense:EXTended:LIST?

Command :SYST:LIC:EXT:LIST?

Long :SYSTem:LICense:EXTended:LIST?

Parameters None

Parameter Suffix None

Description This query lists the licenses installed.

Example

Query
```
:SYST:LIC:EXT:LIST?
```

7.5.4 :SYSTem:SET[?]

Command `:SYST:SET[?]`

Long `:SYSTem:SET[?]`

Parameters <binary block data>

Parameter Suffix None

Description In query form, the command reads a block of data containing the instrument's complete set-up. The set-up information includes all parameter and mode settings, but does not include the contents of the instrument setting memories or the status group registers. The data is in a binary format, not ASCII, and cannot be edited.

In set form, the block data must be a complete instrument set-up read using the query form of the command.

This command has the same functionality as the `*LRN` command.

Example

Command
```
:SYST:SET <binary block data>
```

Query
```
:SYST:SET?
```

7.5.5 :SYSTem:VERSion?

Command :SYST:VERS?

Long :SYSTem:VERSion?

Parameters None

Parameter Suffix None

Description This query returns a formatted numeric value corresponding to the SCPI version number for which the instrument complies.

Example Query
:SYST:VERS?

7.5.6 :SYSTem:COMMunicate:*?

Command :SYST:COMM:*?

Long :SYSTem:COMMunicate:*?

Parameters None

Parameter Suffix None

Description These queries return information about the instrument firmware's available connections. If a connection is not available, the returned value is -1.

This is only useful if there is more than one Keysight module connected to a PC, otherwise one would normally use the default connections (HiSLIP and VXI-11 instrument number 0, socket port 5025, telnet port 5024)

One can never be sure if a socket port is already in use, so one could e.g. specify a HiSLIP number on the command line (`AgM8190Firmware.exe /AutoID /i 5 /FallBack /r …`) and let the firmware find an unused socket port. Then this socket port can be queried using the HiSLIP connection.

Example

Query

```
:SYST:COMM:*?
```

7.5.6.1 :SYSTem:COMMunicate:INSTr[:NUMBer]?

Command

```
:SYST:COMM:INST?
```

Long

```
:SYSTem:COMMunicate:INSTr?
```

Parameters None

Parameter Suffix None

Description This query returns the VXI-11 instrument number used by the firmware.

Example

Query

```
:SYST:COMM:INST?
```

7.5.6.2 :SYSTem:COMMunicate:HISLip[:NUMBer]?

Command `:SYST:COMM:HISL?`

Long `:SYSTem:COMMunicate:HISLip?`

Parameters None

Parameter Suffix None

Description This query returns the HiSLIP number used by the firmware.

Example Query
`:SYST:COMM:HISL?`

7.5.6.3 :SYSTem:COMMunicate:SOCKet[:PORT]?

Command `:SYST:COMM:SOCK?`

Long `:SYSTem:COMMunicate:SOCKet?`

Parameters None

Parameter Suffix None

Description This query returns the socket port used by the firmware.

Example Query
```
:SYST:COMM:SOCK?
```

7.5.6.4 :SYSTem:COMMunicate:TELNet[:PORT]?

Command `:SYST:COMM:TELN?`

Long `:SYSTem:COMMunicate:TELNet?`

Parameters None

Parameter Suffix None

Description This query returns the telnet port used by the firmware.

Example Query
```
:SYST:COMM:TELN?
```

7.5.6.5 :SYSTem:COMMunicate:TCPip:CONTrol?

Command `:SYST:COMM:TCP:CONT?`

Long `:SYSTem:COMMunicate:TCPip:CONTrol?`

Parameters None

Parameter Suffix None

Description

This query returns the port number of the control connection. You can use the control port to send control commands (for example “Device Clear”) to the instrument.

Example

Query

```
:SYST:COMM:TCP:CONT?
```

7.6 Common Command List

7.6.1 *IDN?

Read the instrument's identification string which contains four fields separated by commas. The first field is the manufacturer's name, the second field is the model number, the third field is the serial number, and the fourth field is a revision code which contains four numbers separated dots and a fifth number separated by a dash:

Keysight Technologies, M8190A,<serial number>, x.x.x.x-h

x.x.x.x= Firmware revision number, e.g. 2.0.0.0

h= Hardware revision number

7.6.2 *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.

7.6.3 *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.

7.6.4 *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.

7.6.5 *OPC

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

7.6.6 *OPC?

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

7.6.7 *OPT?

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

7.6.8 *RST

Reset instrument to its factory default state.

7.6.9 *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.

7.6.10 *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.

7.6.11 *TST?

Execute Self Tests. If self-tests pass, a 0 is returned. A number lager than 0 indicates the number of failed tests.
To get actual messages, use `:TEST:TST?`

7.6.12 *LRN?

Query the instrument and return a binary block of data containing the current settings (learn string). You can then send the string back to the instrument to restore this state at a later time. For proper operation, do not modify the returned string before sending it to the instrument. Use `:SYST:SET` to send the learn string. See `:SYSTem:SET[?]`.

7.6.13 *WAI?

Prevents the instrument from executing any further commands until the current command has finished executing.

7.7 Status Model

Introduction

This section describes the structure of the SCPI status system used by the M8190A. 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`.

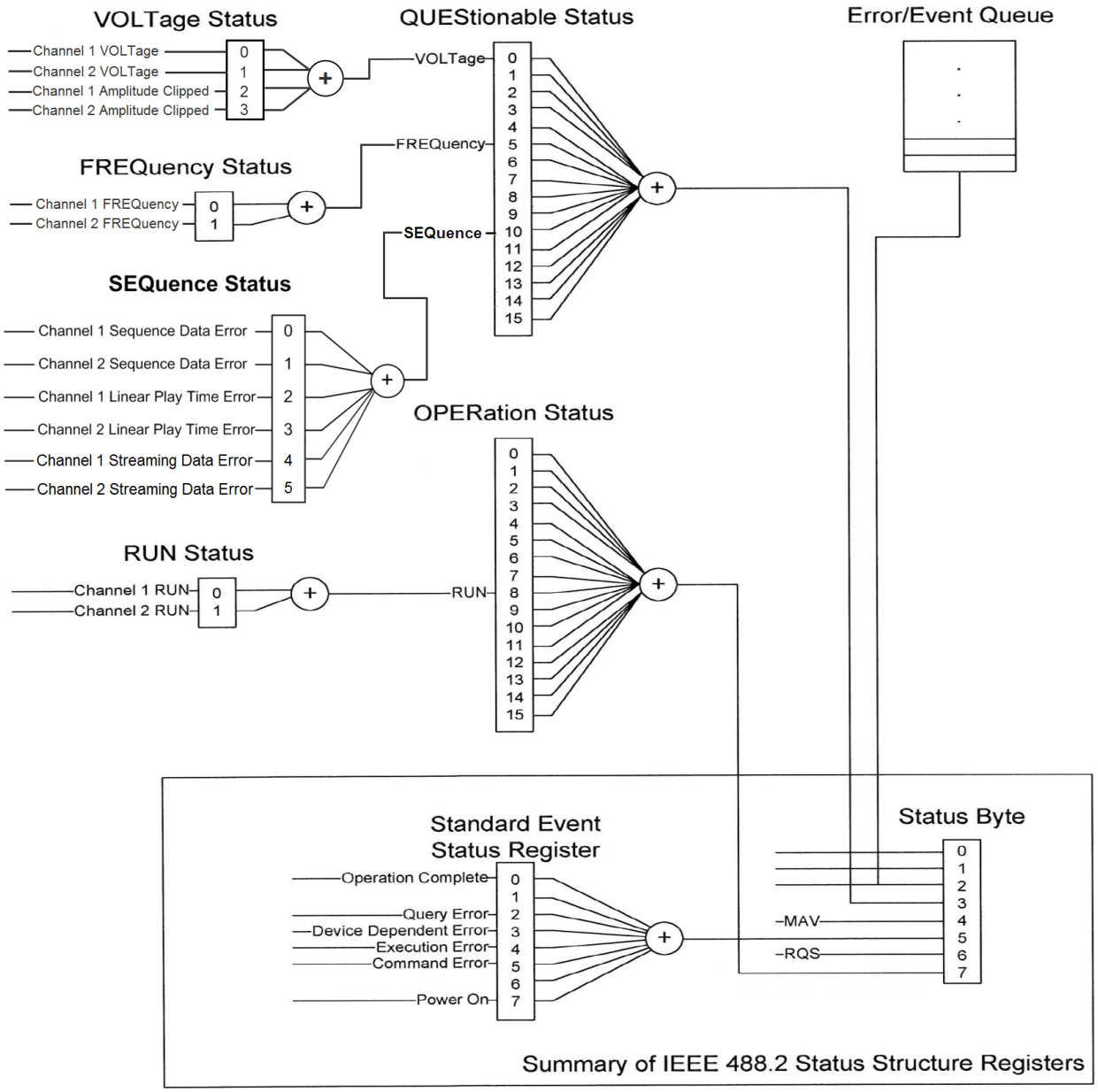


Figure 7-2: Status Register Structure

7.7.1 :STATus:PRESet

Clears all status group event registers. Presets the status group enables PTR and NTR registers as follows:

ENABle = 0x0000, PTR = 0xffff, NTR = 0x0000

7.7.2 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.

Table 7-2: Status Byte Register

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)

7.7.3 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.

Table 7-3: Questionable Data Register

Bit Number		Decimal Value	Definition
0	Voltage warning	1	Output has been 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	Sequence Status	1024	Sequence Generation Errors happened
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.

7.7.3.1 :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 the `*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.

7.7.3.2 :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.

7.7.3.3 :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.

7.7.3.4 :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`.

7.7.3.5 :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`.

7.7.4 Operation Status Subsystem

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

Table 7-4: Operation Status Register

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 was executed 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.

7.7.4.1 :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.

7.7.4.2 :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.

7.7.4.3 :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.

7.7.4.4 :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`.

7.7.4.5 :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`.

7.7.5 Voltage Status Subsystem

The Voltage Status register contains the voltage conditions of the individual channels.

The following SCPI commands and queries are supported:

:STATus:QUEStionable:VOLTage[:EVENt]?

:STATus:QUEStionable:VOLTage:CONDition?

:STATus:QUEStionable:VOLTage:ENABle[?]

:STATus:QUEStionable:VOLTage:NTRansition[?]

:STATus:QUEStionable:VOLTage:PTRansition[?]

Table 7-5: Voltage Status Register

Bit Number		Decimal Value	Definition
0	Voltage warning	1	Output 1 has been switched off (to protect itself)
1	Voltage warning	2	Output 2 has been switched off (to protect itself)
2	Amplitude Clipped	4	The amplitude for output 1 has been clipped to the highest/lowest possible DAC value.
3	Amplitude Clipped	8	The amplitude for output 2 has been clipped to the highest/lowest possible DAC value.

7.7.6 Frequency Status Subsystem

The Frequency Status register contains the frequency conditions of the individual channels.

The following SCPI commands and queries are supported:

:STATus:QUEStionable:FREQuency[:EVENt]?

:STATus:QUEStionable:FREQuency:CONDition?

:STATus:QUEStionable:FREQuency:ENABle[?]

:STATus:QUEStionable:FREQuency:NTRansition[?]

:STATus:QUEStionable:FREQuency:PTRansition[?]

Table 7-6: Frequency Status Register

Bit Number		Decimal Value	Definition
0	Frequency warning	1	Output 1 signal is invalid
1	Frequency warning	2	Output 2 signal is invalid

7.7.7 Sequence Status Subsystem

The Sequence Status register is used to indicate errors in the sequence table data provided by the user. It contains the conditions of the individual channels.

The following SCPI commands and queries are supported:

:STATus:QUEStionable:SEQuence[:EVENt]?

:STATus:QUEStionable:SEQuence:CONDition?

:STATus:QUEStionable:SEQuence:ENABle[?]

:STATus:QUEStionable:SEQuence:NTRansition[?]

:STATus:QUEStionable:SEQuence:PTRansition[?]

Table 7-7: Sequence Status Register

Bit Number	Decimal Value	Definition
0 Sequence data error	1	Output 1 sequence has errors
1 Sequence data error	2	Output 2 sequence has errors
2 Linear-playtime error	4	Output 1 has a linear-playtime error
3 Linear-playtime error	8	Output 2 has a linear-playtime error
4 Streaming data error	16	Output 1 ran out of streaming data
5 Streaming data error	32	Output 2 ran out of streaming data

7.7.8 Run Status Subsystem

The Run Status register contains the run status conditions of the individual channels.

The following SCPI commands and queries are supported:

:STATus:OPERation:RUN[:EVENt]?

:STATus:OPERation:RUN:CONDition?

:STATus:OPERation:RUN:ENABle[?]

:STATus:OPERation:RUN:NTRansition[?]

:STATus:OPERation:RUN:PTRansition[?]

Table 7-8: Run Status Register

Bit Number		Decimal Value	Definition
0	Run Status	1	Indicates if channel 1 is running (:INIT:IMM1 was executed and signals are generated).
1	Run Status	2	Indicates if channel 2 is running (:INIT:IMM2 was executed and signals are generated).

7.8 :ARM/TRIGger Subsystem

7.8.1 :ABORt[1|2]

Command :ABOR[1|2]

Long :ABORt[1|2][?]

Parameters None

Parameter Suffix None

Description Stops signal generation on channel. If channels are coupled, both channels are stopped.

Example

Command
```
:ABOR1
```

7.8.2 :ARM[:SEQuence][:STARt][:LAYer]:DELay[1|2][?] <delay>|MINimum|MAXimum

Command :ARM:DEL[?]

Long :ARM[:SEQuence]|[STARt][:LAYer]:DELay[1|2][?]

Parameters {<delay> | MINimum | MAXimum}

Parameter Suffix HZ

Description Specifies the frequency of the internal trigger period generator of the selected channel. To select the internal trigger generator use `:ARM:SOUR[1|2] INT2`

Example

Command
```
:ARM:DEL 1E-13
```

Query
```
:ARM:DEL?
```

Set or query the fine delay settings. The unit is in seconds.

Table 7-9: Fine Delay Settings

Sample Frequency	Fine Delay Range
$f_{Sa} \geq 6.25$ GSa/s	0 .. 30e-12
2.5 GSa/s $\leq f_{Sa} <$ 6.25 GSa/s	0 .. 60e-12
$f_{Sa} <$ 2.5 GSa/s	0 .. 150e-12

7.8.3 :ARM[:SEQuence][:STARt][:LAYer]:CDELay[1|2][?] <coarse_delay>|MINimum|MAXimum

Command :ARM:CDEL[?]

Long :ARM[:SEQuence][:STARt][:LAYer]:CDELay[1|2][?]

Parameters <coarse_delay> | MINimum | MAXimum

Parameter Suffix None

Description Set or query the coarse delay settings. The unit is in seconds.

Example

Command
```
:ARM:CDEL 3e-9
```

Query
```
:ARM:CDEL?
```

Table 7-10: Coarse Delay Settings

Sample Frequency	Coarse Delay Range	Coarse Delay Resolution
$f_{Sa} \geq 6.25$ GSa/s	0 .. 10e-9	10e-12
2.5 GSa/s $\leq f_{Sa} <$ 6.25 GSa/s	0 .. 10e-9	20e-12
$f_{Sa} <$ 2.5 GSa/s	0 .. 10e-9	50e-12

7.8.4 :ARM[:SEQuence][:STARt][:LAYer]:RNOisefloor[1|2][?] OFF|ON|0|1

Command :ARM:RNO[?]

Long :ARM:RNOisefloor[?]

Parameters OFF|ON|0|1

Parameter Suffix None

Description Set or query the state of the "Reduced Noise Floor" feature.
0/OFF – Noise reduction disabled.
1/ON – Noise reduction enabled.

NOTE If the noise reduction is enabled, coarse and fine delay are constant and cannot be adjusted.

Example

Command
```
:ARM:RNO ON
```

Query
```
:ARM:RNO?
```

7.8.5 :INITiate:CONTinuous[1|2]:ENABle[?] SELF|ARMed

Command :INIT:CONT[1|2]:ENAB[?]

Long :INITiate:CONTinuous[1|2]:ENABle[?]

Parameters SELF|ARMed

Parameter Suffix None

Description Set or query the arming mode.

Example

Command
```
:INIT:CONT:ENAB SELF
```

Query
```
:INIT:CONT:ENAB?
```

7.8.6 :INITiate:CONTinous[1|2][:STATe][?] OFF|ON|0|1

Command `:INIT:CONT[1|2]:STAT[?]`

Long `:INITiate:CONTinuous[1|2]:STATe[?]`

Parameters OFF | ON | 0 | 1

Parameter Suffix None

Description Set or query the continuous mode. This command must be used together with `INIT:GATE[1|2]` to set the trigger mode.

0/OFF – Continuous mode is off. If gate mode is off, the trigger mode is "triggered", else it is "gated".

1/ON – Continuous mode is on. Trigger mode is "automatic". The value of gate mode is not relevant.

Example

Command
```
:INIT:CONT:STAT ON
```

Query
```
:INIT:CONT:STAT?
```

7.8.7 :INITiate:GATE[1|2][:STATe][?] OFF|ON|0|1

Command

```
:INIT:GATE[1|2]:STAT[?]
```

Long

```
:INITiate:GATE[1|2]:STATe[?]
```

Parameters

OFF | ON | 0 | 1

Parameter Suffix

None

Description

Set or query the gate mode. This command must be used together with `INIT:CONT[1|2]` to set the trigger mode.

0/OFF – Gate mode is off.

1/ON – Gate mode is on. If continuous mode is off, the trigger mode is "gated".

Example

Command
```
:INIT:GATE:STAT ON
```

Query
```
:INIT:GATE:STAT?
```

Table 7-11: Trigger Mode Settings

INIT:CONT	INIT:GATE	Trigger Mode
0	0	Triggered
0	1	Gated
1	0	Continuous
1	1	Continuous

7.8.8 :INITiate:IMMediate[1|2]

Command `:INIT:IMM[1|2]`

Long `:INITiate:IMMediate[1|2]`

Parameters None

Parameter Suffix None

Description Start signal generation on a channel. If channels are coupled, both channels are started.

Example Command
`:INIT:IMM`

7.8.9 :ARM[:SEQuence][:STARt][:LAYer]:TRIGger:LEVel[?] <level>|MINimum|MAXimum

Command `:ARM:TRIG:LEV[?]`

Long :ARM:TRIGger:LEVel[?]

Parameters <level>|MINimum|MAXimum

Parameter Suffix None

Description Set or query the trigger input threshold level.
<level> – Threshold level voltage.

Example Command
```
:ARM:TRIG:LEV 3e-9
```

Query
```
:ARM:TRIG:LEV?
```

7.8.10 :ARM[:SEQuence][:STARt][:LAYer]:TRIGger:IMPedance[?] LOW|HIGH

Command :ARM:TRIG:IMP[?]

Long :ARM:TRIGger:IMPedance[?]

Parameters LOW|HIGH

Parameter Suffix None

Description Set or query the trigger input impedance.
- LOW – low impedance
- HIGH – high impedance

Example

Command
```
:ARM:TRIG:IMP HIGH
```

Query
```
:ARM:TRIG:IMP?
```

7.8.11 :ARM[:SEQuence][:STARt][:LAYer]:TRIGger:SLOPe[?] POSitive | NEGative | EITHer

Command

```
:ARM:TRIG:SLOP[?]
```

Long

```
:ARM:TRIGger:SLOPe[?]
```

Parameters

POSitive | NEGative | EITHer

Parameter Suffix

None

Description

Set or query the trigger input slope.

- POSitive – rising edge
- NEGative – falling edge
- EITHer – both

Example

Command
```
:ARM:TRIG:SLOP POS
```

Query
```
:ARM:TRIG:SLOP?
```

7.8.12 :ARM[:SEQuence][:STARt][:LAYer]:TRIGger:SOURce[?] EXTernal|INTernal

Command :ARM:TRIG:SOUR[?]

Long :ARM:TRIGger:SOURce[?]

Parameters EXTernal|INTernal

Parameter Suffix None

Description Set or query the source for the trigger function.

- EXTernal – external trigger input
- INTernal – internal trigger generator

Example

Command
```
:ARM:TRIG:SOUR EXT
```

Query
```
:ARM:TRIG:SOUR?
```

7.8.13 :ARM[:SEQuence][:STARt][:LAYer]:TRIGger:FREQuency[?] <frequency>|MINimum|MAXimum

Command :ARM:TRIG:FREQ[?]

Long :ARM:TRIGger:FREQuency[?]

Parameters <frequency>|MINimum|MAXimum

Parameter Suffix None

Description Set or query the frequency of the internal trigger generator.

- <frequency> – internal trigger frequency

Example

Command
```
:ARM:TRIG:FREQ 1
```

Query
```
:ARM:TRIG:FREQ?
```

7.8.14 :ARM[:SEQuence][:STARt][:LAYer]:EVENt:LEVel[?] <level> | MINimum | MAXimum

Command `:ARM:EVEN:LEV[?]`

Long `:ARM:EVENt:LEVel[?]`

Parameters <level> | MINimum | MAXimum

Parameter Suffix None

Description Set or query the input threshold level.

- <level> – Threshold level voltage.

Example

Command
```
:ARM:EVEN:LEV 2e-9
```

Query
```
:ARM:EVEN:LEV?
```

7.8.15 :ARM[:SEQuence][:STARt][:LAYer]:EVENt:IMPedance[?] LOW | HIGH

Command :ARM:EVEN:IMP[?]

Long :ARM:EVENt:IMPedance[?]

Parameters LOW | HIGH

Parameter Suffix None

Description Set or query the event input impedance.

- LOW – low impedance
- HIGH – high impedance

Example

Command
```
:ARM:EVEN:IMP LOW
```

Query
```
:ARM:EVEN:IMP?
```

7.8.16 :ARM[:SEQuence][:STARt][:LAYer]:EVENt:SLOPe[?] POSitive | NEGative | EITHer

Command :ARM:EVEN:SLOP[?]

Long :ARM:EVENt:SLOPe[?]

Parameters POSitive | NEGative | EITHer

Parameter Suffix None

Description Set or query the event input slope.
POSitive – rising edge
NEGative – falling edge
EITHer – both

Example

Command
```
:ARM:EVEN:SLOP POS
```

Query
```
:ARM:EVEN:SLOP?
```

7.8.17 :ARM[:SEQuence][:STARt][:LAYer]:DYNPort:WIDTh[?] LOWerbits|ALLBits

Command `:ARM:DYNP:WIDT[?]`

Long `:ARM:DYNPort:WIDTh[?]`

Parameters LOWerbits|ALLBits

Parameter Suffix None

Description

Use this command to set or query the number of valid bits of the dynamic control input. The input connector has 13 data pins Data_In[0..12], a Data_Select and a Load pin. Internally, 19 bits of data are used, which are multiplexed using Data_Select. Data_In[0..12] and Data_Select will be stored on rising edge of Load signal.

LOWerbits – 13 Bits are used to select a segment dynamically. Data_Select = Low: Data[0..12] = Data_In[0..12]. Data[13..18] = 0.

ALLBits – 19 Bits are used to select a segment dynamically. Data_Select = Low: Data[0..12] = Data_In[0..12]. Data_Select = High: Data[13..18] = Data_In[0..5].

Example

Command

```
:ARM:DYNP:WIDT ALLB
```

Query

```
:ARM:DYNP:WIDT?
```

7.8.18 :TRIGger[:SEQuence][:STARt]:SOURce:ENABle[?] TRIGger|EVENt

Command

```
:TRIG:SOUR:ENAB[?]
```

Long

```
:TRIGger:SOURce:ENABle[?]
```

Parameters

TRIGger|EVENt

Parameter Suffix

None

Description

Set or query the source for the enable event.

TRIGger - trigger input

EVENt - event input

Example

Command
```
:TRIG:SOUR:ENAB TRIG
```

Query
```
:TRIG:SOUR:ENAB?
```

7.8.19 :TRIGger[:SEQuence][:STARt]:ENABle[1|2]:HWDisable[:STATe][?] 0|1|OFF|ON

Command

```
:TRIG:ENAB:HWD[?]
```

Long

```
:TRIGger:ENABle:HWDisable[?]
```

Parameters

0|1|OFF|ON

Parameter Suffix

None

Description

Set or query the hardware input disable state for the enable function. When the hardware input is disabled, an enable event can only be generated using the :TRIGger[:SEQuence][:STARt]:ENABle[1|2][:IMMediate] command. When the hardware input is enabled, an enable event can be generated by command or by a signal present at the trigger or event input.

Example

Command
```
:TRIG:ENAB:HWD ON
```

Query
```
:TRIG:ENAB:HWD?
```

7.8.20 :TRIGger[:SEQuence][:STARt]:BEGin[1|2]:HWDisable[:STATe][?] 0|1|OFF|ON

Command `:TRIG:BEG:HWD[?]`

Long `:TRIGger:BEGin:HWDisable[?]`

Parameters 0|1|OFF|ON

Parameter Suffix None

Description Set or query the hardware input disable state for the trigger function. When the hardware input is disabled, a trigger can only be generated using the :TRIGger[:SEQuence][:STARt]:BEGin[1|2][:IMMediate] command. When the hardware input is enabled, a trigger can be generated by command, by a signal present at the trigger input or the internal trigger generator.

Example

Command
```
:TRIG:BEG:HWD ON
```

Query
```
:TRIG:BEG:HWD?
```

7.8.21 :TRIGger[:SEQuence][:STARt]:ADVance[1|2]:HWDisable[:STATe][?] 0|1|OFF|ON

Command `:TRIG:ADV:HWD[?]`

Long `:TRIGger:ADVance:HWDisable[?]`

Parameters 0|1|OFF|ON

Parameter Suffix None

Description Set or query the hardware input disable state for the advancement function. When the hardware input is disabled, an advancement event can only be generated using the :TRIGger[:SEQuence][:STARt]:ADVance[1|2][:IMMediate] command. When the hardware input is enabled, an advancement event can be generated by command or by a signal present at the trigger or event input.

Example

Command
```
:TRIG:ADV:HWD 0
```

Query
```
:TRIG:ADV:HWD?
```

7.9 TRIGger - Trigger Input

7.9.1 :TRIGger[:SEQuence][:STARt]:SOURce:ADVance[?] TRIGger | EVENt | INTernal

Command `:TRIG:SOUR:ADV[?]`

Long `:TRIGger:SOURce:ADVance[?]`

Parameters TRIGger | EVENt | INTernal

Parameter Suffix None

Description Set or query the source for the advancement event.
TRIGger - trigger input
EVENt - event input
INTernal – internal trigger generator

Example

Command
```
:TRIG:SOUR:ADV TRIG
```

Query
```
:TRIG:SOUR:ADV?
```

7.9.2 :TRIGger[:SEQuence][:STARt]:ENABle[1 | 2][:IMMediate]

Command `:TRIG:ENAB`

Long `:TRIGger:ENABle`

Parameters None

Parameter Suffix None

Description Send the enable event to a channel.

Example

Command
```
:TRIG:ENAB
```

7.9.3 :TRIGger[:SEQuence][:STARt]:BEGin[1|2][:IMMediate]

Command
```
:TRIG:BEG
```

Long
```
:TRIGger:BEGin
```

Parameters None

Parameter Suffix None

Description In triggered mode send the start/begin event to a channel.

Example

Command
```
:TRIG:BEG
```

7.9.4 :TRIGger[:SEQuence][:STARt]:BEGin[1|2]:GATE[:STATe][?] OFF|ON|0|1

Command
```
:TRIG:BEG:GATE[?]
```

Long
```
:TRIGger:BEGin:GATE[?]
```

Parameters OFF|ON|0|1

Parameter Suffix None

Description In gated mode send a "gate open" (ON|1) or "gate close" (OFF|0) to a channel.

Example

Command
```
:TRIG:BEG:GATE ON
```

Query
```
:TRIG:BEG:GATE?
```

7.9.5 :TRIGger[:SEQuence][:STARt]:ADVance[1|2][:IMMediate]

Command :TRIG:ADV

Long :TRIGger:ADVance

Parameters None

Parameter Suffix None

Description Send the advancement event to a channel.

Example

Command
```
:TRIG:ADV
```

7.10 :FORMat Subsystem

7.10.1 :FORMat:BORDer NORMal|SWAPped

Command `:FORM:BORD[?]`

Long `:FORMat:BORDer[?]`

Parameters NORMal|SWAPped

Parameter Suffix None

Description Byte ORDer. Controls whether binary data is transferred in normal ("big endian") or swapped ("little endian") byte order. Affects `[:SOURce]:SEQuence:DATA`, `[:SOURce]:STABle:DATA` and `TRACe:DATA`.

Example

Command
```
:FORM:BORD NORM
```

Query
```
:FORM:BORD?
```

7.11 :INSTrument Subsystem

7.11.1 :INSTrument:COUPle:STATe[1|2][?] OFF|ON|0|1

NOTE Not available with B02 option.

Command `:INST:COUP:STAT[?]`

Long `:INSTrument:COUPle:STATe[?]`

Parameters OFF|ON|0|1

Parameter Suffix None

Description Switch coupling on/off. If coupling is switched on, the values of the channel where coupling is switched on are taken. Coupled values are:

- FREQ:RAST:SOUR[1|2]
- TRAC[1|2]:DWID
- OUTP:SCLK:SOUR

Example

Command
```
:INST:COUP:STAT ON
```

Query
```
:INST:COUP:STAT?
```

7.11.2 :INSTrument:SLOT[:NUMBer]?

Command `:INST:SLOT?`

Long `:INSTrument:SLOT?`

Parameters None

Parameter Suffix None

Description Query the instrument's slot number in its AXIe frame.

Example Query
`:INST:SLOT?`

7.11.3 :INSTrument:IDENtify [<seconds>]

Command `:INST:IDEN`

Long `:INSTrument:IDENtify`

Parameters <seconds>

Parameter Suffix None

Description Identify the instrument by flashing the green "Access" LED on the front panel for a certain time.
<seconds> optional length of the flashing interval, default is 10 seconds.

Example Comand
```
:INST:IDEN 5
```

7.11.4 :INSTrument:IDENtify:STOP

Command
```
:INST:IDEN:STOP
```

Long
```
:INSTrument:IDENtify:STOP
```

Parameters None

Parameter Suffix None

Description Stop the flashing of the green "Access" LED before the flashing interval has elapsed.

Example Command
```
:INST:IDEN:STOP
```

7.11.5 :INSTrument:MMODule:CONFig?

Command
```
:INST:MMOD:CONF?
```

Long
```
:INSTrument:MMODule:CONFig?
```

Parameters None

Parameter Suffix None

Description

This query returns the state of the multi-module configuration mode (0: disabled, 1: enabled). Some parameters are only available on the master of a multi-module group and not on the slaves. They can only be changed on the master module when multi-module configuration mode is enabled. The corresponding commands are:

- :TRACe[1|2]:DWIDth WSPeed|WPRecision|INTX3|INTX12|INTX24|INTX48
- [:SOURce]:FREQuency:RASTer <frequency>
- [:SOURce]:FREQuency:RASTer:EXTernal <frequency>
- [:SOURce]:FREQuency:RASTer:SOURce[1|2] INTernal|EXTernal
- [:SOURce]:ROSCillator:SOURce EXTernal|AXI|INTernal
- [:SOURce]:ROSCillator:FREQuency <frequency>|MINimum|MAXimum

Example

Query

```
:INST:MMOD:CONF?
```

7.11.6 :INSTrument:MMODule:MODE?

Command

```
:INST:MMOD:MODE?
```

Long

```
:INSTrument:MMODule:MODE?
```

Parameters

None

Parameter Suffix

None

Description

This query returns the multi-module mode.

NORMal – Module does not belong to a multi-module group.

MASTer – Module is the master of a multi-module group.

SLAVe – Module is a slave in a multi-module group.

Example

Query

```
:INST:SLOT?
```

7.12 :MMEMory Subsystem

NOTE MMEM commands requiring <directory_name> assume the current directory if a relative path or no path is provided. If an absolute path is provided, then it is ignored.

7.12.1 :MMEMory:CATalog? [<directory_name>]

Command `:MMEM:CAT?`

Long `:MMEMory:CATalog?`

Parameters None

Parameter Suffix None

Description Query disk usage information (drive capacity, free space available) and obtain a list of files and directories in a specified directory in the following format:

<numeric_value>,<numeric_value>,{<file_entry>}

This command returns two numeric parameters and as many strings as there are files and directories. The first parameter indicates the total amount of storage currently used in bytes. The second parameter indicates the total amount of storage available, also in bytes. The <file_entry> is a string. Each <file_entry> indicates the name, type, and size of one file in the directory list:

<file_name>,<file_type>,<file_size>

As the Windows file system has an extension that indicates file type, <file_type> is always empty. <file_size> provides the size of the file in bytes. In case of directories, <file_entry> is surrounded by square brackets and both <file_type> and <file_size> are empty.

Example

Query
`:MMEM:CAT?`

7.12.2 :MMEMory:CDIRectory [<directory_name>]

Command

`:MMEM:CDIR`

Long

`:MMEMory:CDIRectory`

Parameters

None

Parameter Suffix

None

Description

Changes the default directory for a mass memory file system. The <directory_name> parameter is a string. If no parameter is specified, the directory is set to the `*RST` value. At `*RST`, this value is set to the default user data storage area, that is defined as System.Environment.SpecialFolder.Personal
e.g. C:\Users\Name\Documents
`MMEMory:CDIRectory?` — Query returns full path of the default directory.

Example

Command
`:MMEM:CDIR "C:\Users\Name\Documents"`

Query
`:MMEM:CDIR?`

7.12.3 :MMEMory:COPY <string>,<string>[,<string>,<string>]

Command :MMEM:COPY

Long :MMEMory:COPY

Parameters <string>,<string>

Parameter Suffix None

Description Copies an existing file to a new file or an existing directory to a new directory. Two forms of parameters are allowed. The first form has two parameters. In this form, the first parameter specifies the source, and the second parameter specifies the destination.

The second form has four parameters. In this form, the first and third parameters specify the file names. The second and fourth parameters specify the directories. The first pair of parameters specifies the source. The second pair specifies the destination. An error is generated if the source doesn't exist or the destination file already exists.

Example Command

```
:MMEM:COPY "C:\data.txt", "C:\data_new.txt"
```

7.12.4 :MMEMory:DELete <file_name>[,<directory_name>]

Command :MMEM:DEL

Long :MMEMory:DELete

Parameters <file_name>

Parameter Suffix None

Description Removes a file from the specified directory. The <file_name> parameter specifies the file to be removed.

Example Command
`:MMEM:DEL "C:\data.txt"`

7.12.5 :MMEMory:DATA <file_name>, <data>

Command `:MMEM:DATA`

Long `:MMEMory:DATA`

Parameters <file_name>, <data>

Parameter Suffix None

Description The command form is `MMEMory:DATA <file_name>,<data>`. It loads <data> into the file <file_name>. <data> is in 488.2 block format. <file_name> is string data.

Example Command
`:MMEM:DATA "C:\data.txt", #14test`

7.12.6 :MMEMory:DATA? <file_name>

Command :MMEM:DATA?

Long :MMEMory:DATA?

Parameters <file_name>

Parameter Suffix None

Description The query form is MMEMory:DATA? <file_name> with the response being the associated <data> in block format.

Example Query
:MMEM:DATA? "C:\data.txt"

7.12.7 :MMEMory:MDIRectory <directory_name>

Command :MMEM:MDIR

Long :MMEMory:MDIRectory

Parameters <directory_name>

Parameter Suffix None

Description Creates a new directory. The <directory_name> parameter specifies the name to be created.

Example

Command
```
:MMEM:MDIR "C:\data_dir"
```

7.12.8 :MMEMory:MOVE <string>,<string>[,<string>,<string>]

Command

```
:MMEM:MOVE
```

Long

```
:MMEMory:MOVE
```

Parameters

<string>,<string>

Parameter Suffix

None

Description

Moves an existing file to a new file or an existing directory to a new directory. Two forms of parameters are allowed. The first form has two parameters. In this form, the first parameter specifies the source, and the second parameter specifies the destination.

The second form has four parameters. In this form, the first and third parameters specify the file names. The second and fourth parameters specify the directories. The first pair of parameters specifies the source. The second pair specifies the destination. An error is generated if the source doesn't exist or the destination file already exists.

Example

Command
```
:MMEM:MOVE "C:\data_dir","C:\newdata_dir"
```

7.12.9 :MMEMory:RDIRectory <directory_name>

Command

```
:MMEM:RDIR
```

Long	`:MMEMory:RDIRectory`
Parameters	<directory_name >
Parameter Suffix	None
Description	Removes a directory. The <directory_name> parameter specifies the directory name to be removed. All files and directories under the specified directory are also removed.
Example	Command `:MMEM:RDIR "C:\newdata_dir"`

7.12.10 :MMEMory:LOAD:CSTate <file_name>

Command	`:MMEM:LOAD:CST`
Long	`:MMEMory:LOAD:CSTate`
Parameters	<file_name >
Parameter Suffix	None
Description	Current state of instrument is loaded from a file.
Example	Command `:MMEM:LOAD:CST "C:\data.txt"`

7.12.11 :MMEMory:STORe:CSTate <file_name>

Command :MMEM:STOR:CST

Long :MMEMory:STORe:CSTate

Parameters <file_name >

Parameter Suffix None

Description Current state of instrument is stored to a file.

Example Command
```
:MMEM:STOR:CST "C:\data.txt"
```

7.13 :OUTPut Subsystem

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

Command :OUTP:NORM[?]

Long :OUTPut:NORMal[?]

Parameters OFF|ON|0|1

Parameter Suffix None

Description Switch (normal) output on or off.

Example

Command
```
:OUTP:NORM ON
```

Query
```
:OUTP:NORM?
```

7.13.2 :OUTPut[1|2]:COMPlement[:STATe][?] OFF|ON|0|1

Command
```
:OUTP:COMP[?]
```

Long
```
:OUTPut:COMPlement[?]
```

Parameters OFF|ON|0|1

Parameter Suffix None

Description Switch complement output on or off.

Example

Command
```
:OUTP:COMP ON
```

Query
```
:OUTP:COMP?
```

NOTE Only affects route “DC”.

7.13.3 :OUTPut:SCLK:SOURce[?] INTernal | EXTernal

Command `:OUTP:SCLK:SOUR[?]`

Long `:OUTPut:SCLK:SOURce[?]`

Parameters INTernal | EXTernal

Parameter Suffix None

Description Select which clock source is routed to the SCLK output:
INTernal: the internally generated clock
EXTernal: the clock that is received at SCLK input

Example

Command
```
:OUTP:SCLK:SOUR INT
```

Query
```
:OUTP:SCLK:SOUR?
```

NOTE Not supported for Rev. 1 Modules (-B02)

7.13.4 :OUTPut[1|2]:ROUTe[:SELect][?] DAC|DC|AC

Command :OUTP:ROUT[?]

Long :OUTPut:ROUTe[?]

Parameters DAC|DC|AC

Parameter Suffix None

Description Select the output path:
DAC: Direct DAC output
DC: Amplified differential output
AC: Single ended AC coupled output with up to 10 dBm output level

Example

Command
```
:OUTP:ROUT DAC
```

Query
```
:OUTP:ROUT?
```

7.13.5 :OUTPut[1|2]:DIOFfset[?] <value>|MINimum|MAXimum

Command :OUTP:DIOF[?]

Long :OUTPut:DIOFfset[?]

Parameters <value>|MINimum|MAXimum

Parameter Suffix None

Description Differential Offset: The hardware can compensate for little offset differences between the normal and complement output. "<value>" is the offset to the calibrated optimum DAC value, so the minimum and maximum depend on the result of the calibration.

Example

Command
```
:OUTP:DIOF MAX
```

Query
```
:OUTP:DIOF?
```

Table 7-12: Differential Offset

Value	Normal Output	Complement Output
< 0	Offset decreased	Offset increased
0	No offset	
> 0	Offset increased	Offset decreased

Due to the use of DAC values, the granularity is 1. At the direct output (DAC) at 50Ω: 1 ≈ 1.526µV. At the DC output the resulting offset depends on the voltage settings.

7.14 Sampling Frequency Commands

7.14.1 [:SOURce]:FREQuency:RASTer[?] <frequency> | MINimum | MAXimum

Command :FREQ:RAST[?]

Long :FREQuency:RASTer[?]

Parameters <frequency> | MINimum | MAXimum

Parameter Suffix None

Description None

Example

Command
```
:FREQ:RAST MIN
```

Query
```
:FREQ:RAST?
```

7.14.2 [:SOURce]:FREQuency:RASTer:EXTernal[?] <frequency> | MINimum | MAXimum

Command :FREQ:RAST:EXT[?]

Long :FREQuency:RASTer:EXTernal[?]

Parameters <frequency> | MINimum | MAXimum

Parameter Suffix None

Description Set or query the external sample frequency. The internal sample clock $f_{Sa,i}$ is derived from the sample clock provided at SCLK IN. For $f_{Sa,i}$ = 500 MSa/s...1 GSa/s the sample clock input must be twice of $f_{Sa,i}$. For $f_{Sa,i}$ = 250 MSa/s...500 MSa/s the sample clock input must be four times $f_{Sa,i}$. For $f_{Sa,i}$ = 125 MSa/s...250 MSa/s the sample clock input must be eight times $f_{Sa,i}$.

Example

Command
```
:FREQ:RAST:EXT MIN
```

Query
```
:FREQ:RAST:EXT?
```

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

Command
```
:FREQ:RAST:SOUR[?]
```

Long
```
:FREQuency:RASTer:SOUR[?]
```

Parameters INTernal|EXTernal

Parameter Suffix None

Description 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.

Example

Command
```
:FREQ:RAST:SOUR INT
```

Query
```
:FREQ:RAST:SOUR?
```

7.15 Reference Oscillator Commands

7.15.1 [:SOURce]:ROSCillator:SOURce[?] EXTernal | AXI | INTernal

Command

```
:ROSC:SOUR[?]
```

Long

```
:ROSCillator:SOURce[?]
```

Parameters

EXTernal | AXI | INTernal

Parameter Suffix

None

Description

Set or query the reference clock source.

EXTernal: reference is taken from REF CLK IN.

AXI: reference is taken from AXI backplane.

INTernal: module internal reference. May not be available with every hardware; see 7.15.2.

Example

Command
```
:ROSC:SOUR AXI
```

Query
```
:ROSC:SOUR?
```

7.15.2 [:SOURce]:ROSCillator:SOURce:CHECk? EXTernal | AXI | INTernal

Command :ROSC:SOUR:CHEC?

Long :ROSCillator:SOURce:CHECk?

Parameters EXTernal | AXI | INTernal

Parameter Suffix None

Description Check if a reference clock source is available. Returns 1 if it is available and 0 if not.

Example Query
:ROSC:SOUR:CHEC? AXI

7.15.3 [:SOURce]:ROSCillator:FREQuency[?] <frequency> | MINimum | MAXimum

Command :ROSC:FREQ[?]

Long :ROSCillator:FREQuency[?]

Parameters <frequency> | MINimum | MAXimum

Parameter Suffix None

Description Set or query the expected reference clock frequency, if the external reference clock source is selected.

Example

Command
```
:ROSC:FREQ MIN
```

Query
```
:ROSC:FREQ?
```

7.16 :DAC|DC|AC Subsystem

Set or query voltages or format for the different output paths (DAC, DC, AC selected with the `OUTP:ROUT:SEL` command).

7.16.1 [:SOURce]:DAC|DC|AC[1|2]:FORMat[?] RZ|DNRZ|NRZ|DOUBlet

Command `:DAC|DC|AC:FORM[?]`

Long `:DAC|DC|AC:FORMat[?]`

Parameters RZ|DNRZ|NRZ|DOUBlet

Parameter Suffix None

Description Set or query the DAC format mode.

Example

Command

```
:DAC:FORM DNRZ
:DC:FORM NRZ
:AC:FORM RZ
```

Query

```
:DAC:FORM?
:DC:FORM?
:AC:FORM?
```

DAC Format Modes For further details, refer to the section NRZ / DNRZ / Doublet.

7.16.2 Voltage Ranges

Formulae

High Level = Offset + Amplitude / 2
Low Level = Offset – Amplitude / 2

Ranges The table below shows absolute voltage limits. Effective values may depend on the route's current other levels.

All values are given in V.

Table 7-13: Output Level Ranges

	DAC	DC	AC
AMPlitude	+0.35 .. +0.7	0.15 .. +1.0	+0.1 ..+2.0
OFFset	-0.02 .. +0.02	-0.925 .. +3.225	N/A
HIGH	+0.155 .. +0.37	-0.85 .. +3.3	N/A
LOW	-0.37 .. +0.155	-1.0 .. +3.15	N/A
TERMination	N/A	-1.5 .. +3.5	N/A

7.16.3 [:SOURce]:DAC|DC|AC[1|2]:VOLTage[:LEVel][:IMMediate]:AMPLitude[?] <level>

Command `:DAC|DC|AC:VOLT:AMPL[?]`

Long `:DAC|DC|AC:VOLTage:AMPLitude[?]`

Parameters <level>

Parameter Suffix None

Description Set or query the output amplitude.

Example

Command
```
:DAC:VOLT:AMPL 0.685
```

Query
```
:DAC:VOLT:AMPL?
```

7.16.4 [:SOURce]:DAC|DC[1|2]:VOLTage[:LEVel][:IMMediate]:OFFSet[?] <level>

Command `:DAC|DC:VOLT:OFFS[?]`

Long `:DAC|DC:VOLTage:OFFSet[?]`

Parameters <level>

Parameter Suffix None

Description Set or query the output offset.

Example

Command
```
:DAC:VOLT:OFFS 0.02
```

Query
```
:DAC:VOLT:OFFS?
```

7.16.5 [:SOURce]:DAC|DC[1|2]:VOLTage[:LEVel][:IMMediate]:HIGH[?] <level>

Command
```
:DAC|DC:VOLT:HIGH[?]
```

Long
```
:DAC|DC:VOLTage:HIGH[?]
```

Parameters <level>

Parameter Suffix None

Description Set or query the output high level.

Example

Command
```
:DAC:VOLT:HIGH 3e-1
```

Query
```
:DAC:VOLT:HIGH?
```

7.16.6 [:SOURce]:DAC|DC[1|2]:VOLTage[:LEVel][:IMMediate]:LOW[?] <level>

Command :DAC|DC:VOLT:LOW[?]

Long :DAC|DC:VOLTage:LOW[?]

Parameters <level>

Parameter Suffix None

Description Set or query the output low level.

Example

Command
```
:DAC:VOLT:LOW -0.3
```

Query
```
:DAC:VOLT:LOW?
```

7.16.7 [:SOURce]:DC[1|2]:VOLTage[:LEVel][:IMMediate]:TERMination[?] <level>

Command :DC:VOLT:TERM[?]

Long :DC:VOLTage:TERMination[?]

Parameters <level>

Parameter Suffix None

Description Set or query the termination voltage level.

Example

Command
```
:DC:VOLT:TERM 3e-1
```

Query
```
:DC:VOLT:TERM?
```

7.17 :VOLTage Subsystem

Set the voltages for the currently selected output path (DAC, DC, or AC; selected with the `OUTP:ROUT:SEL` command).

7.17.1 [:SOURce]:VOLTage[1|2][:LEVel][:IMMediate][:AMPLitude][?] <level>

Command `:VOLT[?]`

Long `:VOLTage [?]`

Parameters <level>

Parameter Suffix None

Description Set or query the output amplitude.

Example

Command
```
:VOLT 0.685
```

Query
```
:VOLT?
```

7.17.2 [:SOURce]:VOLTage[1|2][:LEVel][:IMMediate]:OFFSet[?] <level>

Command

```
:VOLT:OFFS[?]
```

Long

```
:VOLTage:OFFSet[?]
```

Parameters

<level>

Parameter Suffix

None

Description

Set or query the output offset.

Example

Command
```
:VOLT:OFFS 0.02
```

Query
```
:VOLT:OFFS?
```

7.17.3 [:SOURce]:VOLTage[1|2][:LEVel][:IMMediate]:HIGH[?] <level>

Command

```
:VOLT:HIGH[?]
```

Long :VOLTage:HIGH[?]

Parameters <level>

Parameter Suffix None

Description Set or query the output high level.

Example

Command
```
:VOLT:HIGH 3e-1
```

Query
```
:VOLT:HIGH?
```

7.17.4 [:SOURce]:VOLTage[1|2][:LEVel][:IMMediate]:LOW[?] <level>

Command :VOLT:LOW[?]

Long :VOLTage:LOW[?]

Parameters <level>

Parameter Suffix None

Description Set or query the output low level.

Example

Command
```
:VOLT:LOW -0.3
```

Query
```
:VOLT:LOW?
```

7.18 :MARKer Subsystem

7.18.1 Ranges

Table 7-14: Sync, Sample Marker levels: Ranges

	Level
AMPlitude	0.0 .. +2.25
OFFset	-0.5 .. +1.75
HIGH	+0.5 .. +1.75
LOW	-0.5 .. +1.75

7.18.2 [:SOURce]:MARKer[1|2]:SAMPle:VOLTage[:LEVel][:IMMediate]:AMPLitude[?] <level>

Command
```
:MARK:SAMP:VOLT:AMPL[?]
```

Long
```
:MARKer:SAMPle:VOLTage:AMPLitude[?]
```

Parameters <level>

Parameter Suffix None

Description Set or query the output amplitude for sample marker.

Example

Command
```
:MARK:SAMP:VOLT:AMPL 3.0e-9
```

Query
```
:MARK:SAMP:VOLT:AMPL?
```

7.18.3 [:SOURce]:MARKer[1|2]:SAMPle:VOLTage[:LEVel][:IMMediate]:OFFSet[?] <level>

Command

```
:MARK:SAMP:VOLT:OFFS[?]
```

Long

```
:MARKer:SAMPle:VOLTage:OFFSet[?]
```

Parameters

<level>

Parameter Suffix

None

Description

Set or query the output offset for sample marker.

Example

Command
```
:MARK:SAMP:VOLT:OFFS 2.5e-9
```

Query
```
:MARK:SAMP:VOLT:OFFS?
```

7.18.4 [:SOURce]:MARKer[1|2]:SAMPle:VOLTage[:LEVel][:IMMediate]:HIGH[?] <level>

Command

```
:MARK:SAMP:VOLT:HIGH[?]
```

Long :MARKer:SAMPle:VOLTage:HIGH[?]

Parameters <level>

Parameter Suffix None

Description Set or query the output high level for sample marker.

Example

Command
```
:MARK:SAMP:VOLT:HIGH 1.5
```

Query
```
:MARK:SAMP:VOLT:HIGH?
```

7.18.5 [:SOURce]:MARKer[1|2]:SAMPle:VOLTage[:LEVel][:IMMediate]:LOW[?] <level>

Command :MARK:SAMP:VOLT:LOW[?]

Long :MARKer:SAMPle:VOLTage:LOW[?]

Parameters <level>

Parameter Suffix None

Description Set or query the output low level for sample marker.

Example

Command
```
:MARK:SAMP:VOLT:LOW -0.5
```

Query
```
:MARK:SAMP:VOLT:LOW?
```

7.18.6 [:SOURce]:MARKer[1|2]:SYNC:VOLTage[:LEVel][:IMMediate]:AMPLitude[?] <level>

Command `:MARK:SYNC:VOLT:AMPL[?]`

Long `:MARKer:SYNC:VOLTage:AMPLitude[?]`

Parameters <level>

Parameter Suffix None

Description Set or query the output amplitude for sync marker.

Example

Command
```
:MARK:SYNC:VOLT:AMPL 2.5e-9
```

Query
```
:MARK:SYNC:VOLT:AMPL?
```

7.18.7 [:SOURce]:MARKer[1|2]:SYNC:VOLTage[:LEVel][:IMMediate]:OFFSet[?] <level>

Command `:MARK:SYNC:VOLT:OFFS[?]`

Long :MARKer:SYNC:VOLTage:OFFSet[?]

Parameters <level>

Parameter Suffix None

Description Set or query the output offset for sync marker.

Example

Command
```
:MARK:SYNC:VOLT:OFFS 2.5e-9
```

Query
```
:MARK:SYNC:VOLT:OFFS?
```

7.18.8 [:SOURce]:MARKer[1|2]:SYNC:VOLTage[:LEVel][:IMMediate]:HIGH[?] <level>

Command :MARK:SYNC:VOLT:HIGH[?]

Long :MARKer:SYNC:VOLTage:HIGH[?]

Parameters <level>

Parameter Suffix None

Description Set or query the output high level for sync marker.

Example

Command
```
:MARK:SYNC:VOLT:HIGH 1.5
```

Query
```
:MARK:SYNC:VOLT:HIGH?
```

7.18.9 [:SOURce]:MARKer[1|2]:SYNC:VOLTage[:LEVel][:IMMediate]:LOW[?] <level>

Command

```
:MARK:SYNC:VOLT:LOW[?]
```

Long

```
:MARKer:SYNC:VOLTage:LOW[?]
```

Parameters

<level>

Parameter Suffix

None

Description

Set or query the output low level for sync marker.

Example

Command
```
:MARK:SYNC:VOLT:LOW -0.5
```

Query
```
:MARK:SYNC:VOLT:LOW?
```

7.19 [:SOURce]:FUNCtion[1|2]:MODE ARBitrary|STSequence|STSCenario

Command `[:SOUR]:FUNC:MODE[?]`

Long `[:SOURce]:FUNCtion:MODE[?]`

Parameters ARBitrary|STSequence|STSCenario

Parameter Suffix None

Description Use this command to set or query the type of waveform that will be generated.

ARBitrary – arbitrary waveform segment

STSequence – sequence

STSCenario – scenario

Example

Command

```
:FUNC:MODE ARB
```

Query

```
:FUNC:MODE?
```

7.20 :SEQuence Subsystem

Sequences are made of a number of arbitrary waveforms, which can be linked and looped in user-programmable order. Sequences are generated from waveforms stored in the waveform memory as memory segments.

Steps to generate a sequence:

- Generate the waveform segments using the Arbitrary Waveform Commands (TRACe Subsystem).
- Define a sequence (:SEQ[1|2]:DEF:NEW).
- Fill the sequence with segments (:SEQ[1|2]:DATA).
- Select sequence mode (:SOUR:FUNC[1|2]:MODE STSequence).
- Select a sequence (:STAB[1|2]:SEQ:SEL).
- Start signal generation with :INIT:IMM.

This subsystem does not support the following functionality offered by the STABle subsystem.

- Scenarios
- Idle delay commands

7.20.1 [:SOURce]:SEQuence[1|2]:DEFine:NEW <length>

Command `:SEQ:DEF:NEW`

Long `:SEQuence:DEFine:NEW`

Parameters <length>

Parameter Suffix None

Description

<length> number of sequence table entries.

This command defines a new sequence. The <length> parameter specifies the number of sequence table entries in the sequence.

The new <sequence-id> in the range 0...512K–2 is returned. This value is the index of the first sequence table entry of this sequence.

Example

Query

```
:SEQ:DEF:NEW? 480
```

7.20.2 [:SOURce]:SEQuence[1|2]:DATA[?] <sequence_id>,<step>[,<length>|,<value>,<value>...|,<data block>]

Command

```
:SEQ:DATA[?]
```

Long

```
:SEQuence:DATA[?]
```

Parameters

<length>

Parameter Suffix

None

Description

<length> number of sequence table entries.

This command defines a new sequence. The <length> parameter specifies the number of sequence table entries in the sequence.

The new <sequence-id> in the range 0...512K–2 is returned. This value is the index of the first sequence table entry of this sequence.

This SCPI has the following syntax for command/query:

Command

[:SOURce]:SEQuence[1|2]:DATA <sequence_id>,<step>,[<value>,<value>,...|<data block>]

Query

[:SOURce]:SEQuence[1|2]:DATA? <sequence_id>,<step>,<length>

<sequence_id> 0...512K-2

<step> sequence table entry in the sequence. Valid range is 0 to <length>-1 defined with seq:def:new

<length> number of sequence steps to be read

The command form writes one or multiple sequence steps beginning at <step>. Each step consists of 6 32-bit words per entry with the following meanings.

<segment_id>id of the segment.

<loop_count> number of segment loop iterations 1...4G-1

<advance_mode> 0: AUTO, 1: CONDitional, 2: REPeat, 3: SINGle. Specifies how the generator advances from one sequence table entry to the next one.

<marker_enable> 0: Marker disabled, 1: Marker enabled

<start_addr> Address of the first sample within the segment to be inserted into the sequence.

<end_addr> Address of the last sample in the segment to be inserted into the sequence. You can use 0xffffffff to specify the last sample of the segment.

Individual entries, multiple entries or even the complete sequence at once can be written. The data can be given in IEEE binary block format or as a comma-separated list of 32-bit values.

The query form returns for a given sequence-id and step the segment data items (segment-id, loop-count, advance-mode, marker-enable, start-addr, end-addr).

If the sequence does not exist or the step is greater than the sequence length – 1, an error is generated.

Example

Command

```
:SEQ1:DATA 0,0,1,1,1,0,0,239
```

Command using data block

[:SOURce]:SEQuence[1|2]:DATA <sequence_id>,<step>, <data_block>

```
:SEQ1:DATA 0,0,<data_block>
```

Query

```
:SEQ:DATA? 0,1,1
```

7.20.3 [:SOURce]:SEQuence[1|2]:DELete <sequence_id>

Command :SEQ:DEL

Long :SEQuence:DELete

Parameters < sequence_id >

Parameter Suffix None

Description < sequence_id > 0...512K-2
Delete a sequence.

Example Command
:SEQ:DEL 0

7.20.4 [:SOURce]:SEQuence[1|2]:DELete:ALL

Command :SEQ:DEL:ALL

Long :SEQuence:DELete:ALL

Parameters None

Parameter Suffix None

Description Delete all sequences.

Example Command
`:SEQ:DEL:ALL`

7.20.5 [:SOURce]:SEQuence[1|2]:CATalog?

Command `:SEQ:CAT?`

Long `:SEQuence:CATalog?`

Parameters None

Parameter Suffix None

Description The query returns a comma-separated list of sequence-ids that are defined and the length in segments of each sequence. So first number is a sequence id, next length ...
If no sequence is defined, “0, 0” is returned.

Example Query
`:SEQ:CAT?`

7.20.6 [:SOURce]:SEQuence[1|2]:FREE?

Command `:SEQ:FREE?`

Long `:SEQuence:FREE?`

Parameters None

Parameter Suffix None

Description The query returns the number of free sequence entries in the following form: <sequence entries available>, < sequence entries in use>, < contiguous sequence entries available>.

Example

Query
```
:SEQ:FREE?
```

7.20.7 [:SOURce]:SEQuence[1|2]:ADVance[?] <sequence_id>,AUTO|CONDitional|REPeat|SINGle

Command `:SEQ:ADV[?]`

Long `:SEQuence:ADVance[?]`

Parameters <sequence_id>, AUTO | COND | REP | SING

Parameter Suffix None

Description < sequence_id > 0...512K-2

Set or query the advancement mode between iterations of a sequence.

Example

Command
```
:SEQ:ADV 0,AUTO
```

Query
```
:SEQ:ADV? 0
```

7.20.8 [:SOURce]:SEQuence[1|2]:COUNt <sequence_id>,<count>

Command :SEQ:COUN

Long :SEQuence:COUNt

Parameters <sequence_id>, <count>

Parameter Suffix None

Description < sequence_id > 0...512K-2
<count> 1...4G-1 number of times the sequence is executed.
Set or query the number of iterations of a sequence.

Example Command
```
:SEQ:COUN 0,1
```

7.20.9 [:SOURce]:SEQuence[1|2]:NAME[?] <sequence_id>,<name>

Command :SEQ:NAME[?]

Long :SEQuence:NAME[?]

Parameters <sequence_id>, <name>

Parameter Suffix None

Description Associate a <name> to <sequence_id>.

Example

Command
```
:SEQ:NAME 0,"JIM"
```

Query
```
:SEQ:NAME? 0
```

7.20.10 [:SOURce]:SEQuence[1|2]:COMMent[?] <sequence_id>,<comment>

Command

```
:SEQ:COMM[?]
```

Long

```
:SEQuence:COMMent[?]
```

Parameters

<sequence_id>, <comment>

Parameter Suffix

None

Description

Associate a <comment> to <sequence_id>.

Example:

seq2:comm 5,"any information that should be associated with this segment"

Example

Command
```
:SEQ:COMM 0,"Information associated with this segment"
```

Query
```
:SEQ:COMM? 0
```

7.21 :STABle Subsystem

Use this subsystem to directly manipulate the sequencer table. This is necessary when full control of the sequence capabilities is desired. Unlike the SEQuence subsystem, the STABle subsystem also supports scenarios, idle delay commands and configuration segment entries.

Follow these steps for all function modes:

First create waveform data segments in the module memory like described in the “Arbitrary Waveform Generation” paragraph of the “TRACe subsystem”.

Create sequence table entries that refer to the waveform segments using the `STAB[1|2]:DATA` command.

The next steps depend on the selected mode:

7.21.1 Generating arbitrary waveforms in dynamic mode

Select the sequence table entry that describes the segment that will be executed first, before any dynamic segment selection, using the `TRAC[1|2]:SEL` command.

Select dynamic mode using the command `STAB[1|2]:DYN ON`.

Before starting signal generation with `INIT:IMM[1|2]` select the arbitrary waveform mode using the `FUNC:MODE ARB` command.

7.21.2 Generating sequences in non-dynamic mode

Select the sequence table entry that describes the first segment in your sequence using the `STAB[1|2]:SEQ:SEL` command.

Select non-dynamic mode using the command `STAB[1|2]:DYN OFF`.

Before starting signal generation with `INIT:IMM[1|2]` select the sequence mode using the `FUNC:MODE STS` command.

7.21.3 Generating sequences in dynamic mode

Select the sequence table entry that describes the first segment in your sequence that will be executed first, before any dynamic sequence selection, using the `STAB[1|2]:SEQ:SEL` command.

Select dynamic mode using the command `STAB[1|2]:DYN ON`.

Before starting signal generation with `INIT:IMM[1|2]` select the sequence mode using the `FUNC:MODE STS` command.

7.21.4 Generating scenarios

Set the scenario advancement mode using the `STAB[1|2]:SCEN:ADV` command.

Set the scenario loop count using the `STAB[1|2]:SCEN:COUN` command.

Select the sequence table entry that describes the first segment in your scenario using the `STAB[1|2]:SCEN:SEL` command.

Select non-dynamic mode using the command `STAB[1|2]:DYN OFF`.

Before starting signal generation with `INIT:IMM[1|2]` select the scenario mode using the `FUNC:MODE STSC` command.

7.21.5 [:SOURce]:STABle[1|2]:RESet

Command `:STAB:RES`

Long `:STABle:RESet`

Parameters None

Parameter Suffix None

Description Reset all sequence table entries to default values.

Example

Command
```
:STAB:RES
```

7.21.6 [:SOURce]:STABle[1|2]:DATA[?] <sequence_table_index>,(<length>|<block>|<value>,<value>...)

Command
```
:STAB:DATA[?]
```

Long
```
:STABle:DATA[?]
```

Parameters <sequence_table_index>,(<length>|<block>|<value>,<value>...)

Parameter Suffix None

Description

The command form writes directly into the sequencer memory. The query form reads the data from the sequencer memory, if all segments are read-write. The query returns an error, if at least one write-only segment in the waveform memory exists.

The sequencer memory has 524287 (512k – 1) entries. With this command entries can be directly manipulated using 6 32-bit words per entry. Individual entries, multiple entries or even the complete sequencer memory at once can be manipulated. The data can be given in IEEE binary block format or in comma-separated list of 32-bit values.

<sequence_table_index> – index of the sequence table entry to be accessed (0..512k-2)

<length> - number of entries to be read

<block>– up to 512k – 1 sequence vectors, each consisting of 6 32-bit parameter values

<value>– a 32-bit parameter value; the meaning depends on the type of sequence entry to be created and on the index position for an entry, see following tables .

Example

Command using comma separated parameter:

// Data Entry:
[:SOURce]:STABle[1|2]:DATA
<sequence_id>,<control_parameter>,<sequence_loop><segment_loop><segment_id>,<start_address>,<end_address>

Example:
// Create a data entry at index 0 of the sequence table.
// Mark as start of sequence, sequence loop count = 1, segment loop count = 2, segment id = 1,
// segment start offset is 0, segment end offset is equal to the end of the segment.

```
:STAB1:DATA 0,#0x10000000,1,2,1,0, #0xFFFFFFFF
```

// Idle entry
[:SOURce]:STABle[1|2]:DATA
<sequence_id>,<control_parameter>,<sequence_loop><command-code><idel_sample>,<idle_delay>,<0>

Example:
// Create an idle delay entry at index 0 of the sequence table.
// Mark as command (control_parameter = 0x80000000) , sequence loop count = // 1, command code = 0 , idle sample = 0, idle delay = 960

```
:STAB1:DATA 0, #0x80000000,1,0,0,960,0
```

// Configuration entry
[:SOURce]:STABle[1|2]:DATA
<sequence_id>,<control_parameter>,<sequence_loop><command_code><segment_id>,<start_address>,<end_address>

Example:
// Create a configuration entry at index 0 of the sequence table.
// Mark as command(control_parameter = 0x80000000), sequence loop count = 1, action id + command code = 0x00020001, segment id = 1,
// segment start offset is 0, segment end offset is 239

```
:STAB1:DATA 0, #0x80000000, 1, #0x00020001, 1, 0, 239
```

Using Data block:

[:SOURce]:STABle[1|2]:DATA <sequence_id>,<data_block>

```
:STAB1:DATA 0, <data_block>
```

Query

[:SOURce]:STABle[1|2]:DATA? <sequence_id>,<length>

```
:STAB1:DATA? 0, 6
```

The sequence table can contain data, idle delay and configuration entries. The following table shows, which parameters are needed to create the different entry types.

Table 7-15: Sequencer Table Entries

Parameter Index	Data Entry	Idle Delay Entry	Configuration Entry
0	Control	Control	Control
1	Sequence Loop Count	Sequence Loop Count	Sequence Loop Count
2	Segment Loop Count	Command Code	Command Code
3	Segment Id	Idle Sample	Segment Id
4	Segment Start Offset	Idle Delay	Segment Start Offset
5	Segment End Offset	0	Segment End Offset

The following tables show the meaning of the parameters and the applicability per sequence entry type. Bits marked as “Reserved” or “N/A” must be set to 0.

Table 7-16: Control

Bit	Width	Meaning	Data	Idle	Config
31	1	Data/command selection 0: Data 1: Command (type of command is selected by command code)	X	X	X
30	1	End Marker Sequence	X	X	X
29	1	End Marker Scenario	X	X	X
28	1	Init Marker Sequence	X	X	X
27:25	3	Reserved	X	X	X
24	1	Marker Enable	X	N/A	X
23:20	4	Advancement Mode Sequence 0: Auto 1: Conditional 2: Repeat 3: Single 4 – 15: Reserved	X	X	X
19:16	4	Advancement Mode Segment 0: Auto 1: Conditional 2: Repeat 3: Single 4 – 15: Reserved	X	N/A	X
15	1	Amplitude Table Init	X	X	N/A
14	1	Amplitude Table Increment	X	X	N/A
13	1	Frequency Table Init	X	X	N/A
12	1	Frequency Table Increment	X	X	N/A
11:0	12	Reserved	X	X	X

Table 7-17: Sequence Loop Count

Bit	Width	Meaning	Data	Idle	Config
31:0	32	Number of sequence iterations (1..4G-1), only applicable in the first entry of a sequence	X	X	X

Table 7-18: Segment Loop Count

Bit	Width	Meaning	Data	Idle	Config
31:0	32	Number of segment iterations (1..4G-1)	X	N/A	N/A

Table 7-19: Segment Id

Bit	Width	Meaning	Data	Idle	Config
31:19	13	Reserved	X	N/A	X
18:0	19	Segment id (1 .. 512K)	X	N/A	X

Table 7-20: Segment Start Offset

Bit	Width	Meaning	Data	Idle	Config
31:0	32	Allows specifying a segment start address in samples, if only part of a segment loaded into waveform data memory is to be used. The value must obey the granularity of the selected waveform output mode.	X	N/A	X

Table 7-21: Segment End Offset

Segment End Offset					
Bit	**Width**	**Meaning**	**Data**	**Idle**	**Config**
31:0	32	Allows specifying a segment end address in samples, if only part of a segment loaded into waveform data memory is to be used. The value must obey the granularity of the selected waveform output mode. You can use the value 0xffffffff, if the segment end address equals the last sample in the segment.	X	N/A	X

Table 7-22: Command Code

Command Code					
Bit	**Width**	**Meaning**	**Data**	**Idle**	**Config**
31:16	16	Action table id	N/A	N/A	X
15:0	16	Command code 0: Idle Delay 1: Configuration	N/A	X	X

Table 7-23: Idle Sample

Idle Sample					
Bit	**Width**	**Meaning**	**Data**	**Idle**	**Config**
31:0	32	Sample to be played during pause. In interpolated mode the I value is in bits 14:0, the Q value in bits 30:16. Bits 13:0 in precision mode and bits 11:0 in speed mode contain the DAC value.	N/A	X	N/A

Table 7-24: Idle Delay

Idle Delay							
Bit	**Width**	**Meaning**				**Data**	**Idle**	**Config**
31:0	32	Idle delay in unit depending on the DAC output mode.				N/A	X	N/A
		DAC Output Mode	Unit	Min	Max			
		WSPeed	Sample clock	10*64	2^{25}*64+63			
		WPRecision	Sample clock	10*48	2^{25}*48+47			
		INTX3	Sample clock*3	10*24	2^{25}*24+23			
		INTX12	Sample clock*12	10*24	2^{25}*24+23			
		INTX24	Sample clock*24	10*24	2^{25}*24+23			
		INTX48	Sample clock*48	10*24	2^{25}*24+23			

Example:

// Create a data entry at index 0 of the sequence table.

// Mark as start of sequence, sequence loop count = 1, segment loop count = 2, segment id = 3,

// segment start offset is 240, segment end offset is equal to the end of the segment.

```
STAB1:DATA 0,#h10000000,1,2,3,240,#hffffffff
```

// Create an idle delay entry at index 0 of the sequence table.

// Mark as command, sequence loop count = 1, command code = 0 , idle sample = 0,

// idle delay = 960, last parameter word is unused.

```
STAB1:DATA 0,#h80000000,1,0,0,960,0
```

// Create a configuration entry at index 0 of the sequence table.

// Mark as command, sequence loop count = 1, action id + command code = 0x00020001, segment id = 3,

// segment start offset is 0, segment end offset is equal to the end of the segment

```
STAB1:DATA 0,#h80000000,1,#h00020001,3,0,#hffffffff
```

7.21.7 [:SOURce]:STABle[1|2]:DYNamic:HWDisable[:STATe][?] OFF|ON|0|1

Command `:STAB:DYN:HWD[?]`

Long `:STABle:DYNamic:HWDisable [?]`

Parameters OFF | ON | 0 | 1

Parameter Suffix None

Description Set or query the dynamic port hardware input disable state. The command has only an effect when the dynamic mode for segments or sequences is active. When the hardware input is disabled, segments or sequences can only be selected dynamically using the [SOURce]:STABle:DYNamic:SELect < sequence-table-index> command. When the hardware input is enabled, segments or sequences can be selected dynamically by command or by the signal present at the dynamic control port.

Example

Command
```
:STAB:DYN:HWD 1
```

Query
```
:STAB:DYN:HWD?
```

7.21.8 [:SOURce]:STABle[1|2]:SEQuence:SELect[?] <sequence_table_index>

Command
```
:STAB:SEQ:SEL[?]
```

Long
```
:STABle:SEQuence:SELect[?]
```

Parameters <sequence_table_index>

Parameter Suffix None

Description Select where in the sequence table the sequence starts in STSequence mode. In dynamic sequence selection mode select the sequence that is played before the first sequence is dynamically selected.

Example

Command
```
:STAB:SEQ:SEL 0
```

Query
```
:STAB:SEQ:SEL?
```

7.21.9 [:SOURce]:STABle[1|2]:SEQuence:STATe?

Command

```
:STAB:SEQ:STAT?
```

Long

```
:STABle:SEQuence:STATe?
```

Parameters

None

Parameter Suffix

None

Description

This query returns an integer value containing the sequence execution state and the currently executed sequence table entry. The query can be used in non-dynamic sequence and scenario mode and in dynamic arbitrary and dynamic sequence mode.

Table 7-25: Returned Sequence State

Bit	Width	Meaning
31:21	11	Reserved
20:19	2	Sequence execution state 0: Idle 1: Waiting for Trigger 2: Running 3: Waiting for Advancement Event
18:0	19	Index of currently executed sequence table entry. In Idle state the value is undefined.

Example

Query
```
:STAB:SEQ:STAT?
```

7.21.10 [:SOURce]:STABle[1|2]:DYNamic:[STATe][?] OFF|ON|0|1

Command

```
:STAB:DYN[?]
```

Long

```
:STABle:DYNamic[?]
```

Parameters

OFF | ON | 0 | 1

Parameter Suffix

None

Description

Use this command to enable or disable dynamic mode.

If dynamic mode is switched off, segments or sequences can only be switched in program mode, that is signal generation must be stopped. In arbitrary mode use TRACe[1|2]:SELect to switch to a new segment. In sequence mode use [:SOURce]:STABle[1|2]:SEQuence:SELect to switch to a new sequence.

If dynamic mode is switched on, segments or sequences can be switched dynamically when signal generation is active. The next segment or sequence is either selected by the command [:SOURce]:STABle[1|2]:DYNamic:SELect or by a signal fed into the dynamic port on the module front panel. The external input values select sequence table entries with corresponding indices.

The number of valid bits of the dynamic port is specified by the :ARM[:SEQuence][:STARt][:LAYer]: DYNPort:WIDTh[?][LOWerbits|ALLBits] command.

Example

Command
```
:STAB:DYN 0
```

Query
```
:STAB:DYN?
```

7.21.11 [:SOURce]:STABle[1|2]:DYNamic:SELect[?] <sequence_table_index>

Command

```
:STAB:DYN:SEL[?]
```

Long

```
:STABle:DYNamic:SELect[?]
```

Parameters

<sequence_table_index>

Parameter Suffix

None

Description

When the dynamic mode for segments or sequences is active, set or query the selected sequence table entry.

Example

Command
```
:STAB:DYN:SEL 0
```

Query
```
:STAB:DYN:SEL?
```

7.21.12 [:SOURce]:STABle[1|2]:SCENario:SELect[?] <sequence_table_index>

Command

```
:STAB:SCEN:SEL[?]
```

Long :STABle:SCENario:SELect[?]

Parameters <sequence_table_index>

Parameter Suffix None

Description Select where in the sequence table the scenario starts in STSCenario mode.

Example

Command
```
:STAB:SCEN:SEL 0
```

Query
```
:STAB:SCEN:SEL?
```

7.21.13 [:SOURce]:STABle[1|2]:SCENario:ADVance[?] AUTO|CONDitional|REPeat|SINGle

Command :STAB:SCEN:ADV[?]

Long :STABle:SCENario:ADVance[?]

Parameters AUTO | COND | REP | SING

Parameter Suffix None

Description Set or query the advancement mode for scenarios.

Example

Command
```
:STAB:SCEN:ADV AUTO
```

Query
```
:STAB:SCEN:ADV?
```

7.21.14 [:SOURce]:STABle[1|2]:SCENario:COUNt[?] <count>

Command

```
:STAB:SCEN:COUN[?]
```

Long

```
:STABle:SCENario:COUNt[?]
```

Parameters

<count>

Parameter Suffix

None

Description

Set or query the loop count for scenarios.
<count> – 1..4G-1: number of times the scenario is repeated.

Example

Command
```
:STAB:SCEN:COUN 2
```

Query
```
:STAB:SCEN:COUN?
```

7.21.15 [:SOURce]:STABle[1|2]:STReaming[:STATe][?] 0|1|OFF|ON

Command

```
:STAB:STR[?]
```

Long `:STABle:STReaming[?]`

Parameters 0|1|OFF|ON

Parameter Suffix None

Description Use this command to enable or disable streaming mode. Streaming mode can only be enabled in dynamic sequencing mode.

For more details on streaming mode, refer to the chapter *Streaming* .

Example

Command
```
:STAB:STR ON
```

Query
```
:STAB:STR?
```

7.22 :TRACe Subsystem

Use the :TRACe subsystem to control the arbitrary waveforms and their respective parameters:

- Select the DAC width in direct mode and the interpolation factor in interpolated mode.
- Create waveform segments of arbitrary size with optional initialization.
- Download waveform data into the segments.
- Delete one or all waveform segments from the waveform memory.

7.22.1 Direct and Interpolated Mode

In direct mode (speed and precision mode) arbitrary waveforms are generated directly from the DAC values in the waveform data. Each DAC value has a resolution of 12 bits in speed mode and 14 bits in precision mode.

In interpolated mode the waveform data consists of I/Q sample pairs. They are first interpolated and then sent to the internal IQ modulator. The result is sent to the DAC. I and Q values have a size of 15 bits each.

7.22.2 Waveform Memory Capacity

DAC values and I/Q samples are stored in a dedicated waveform memory. The capacity of this memory in samples depends on the waveform output mode and the installed memory option.

Table 7-26: Waveform Memory Capacity

Waveform output mode	Standard memory configuration	Memory option 02G
Speed	128 MSa	2 GSa
Precision	128 MSa	1.5 GSa
x3 interpolation	64 MSa	768 MSa
x12 interpolation	64 MSa	768 MSa
x24 interpolation	64 MSa	768 MSa
x48 interpolation	64 MSa	768 MSa

7.22.3 Waveform Granularity and Size

The waveform memory is internally organized in fixed size memory vectors. Their size depends on the selected waveform output mode. The segment size, the data size and the offset values passed in the commands of this subsystem must be multiples of the memory vector size. The minimum segment size also depends on the waveform output mode. The maximum segment size is only limited by the size of the available memory.

Table 7-27: Waveform Granularity and Size

Waveform output mode	Memory vector size in samples	Minimum segment size in samples
Speed	64	320
Precision	48	240
x3 interpolation	24	120
x12 interpolation	24	120
x24 interpolation	24	120
x48 interpolation	24	120

7.22.4 Waveform Data Format in Direct Mode

DAC values have 14 bit resolution in precision mode and 12 bit in speed mode. DAC values are signed. Valid range is -8192 to +8191 in precision mode and -2048 to 2047 in speed mode.

Marker position data consists of two bits, one bit for sample markers (SMPM) and one bit for sync markers (SYNM). The marker position data is stored together with the DAC values in a 16-bit signed integer. The DAC values occupy the most significant bits and the marker data the least significant bits. In speed mode bits 2 and 3 are don't-care. The same data format can be used for both modes. In speed mode the two least significant bits are simply ignored.

Only the sync marker bit in the first sample of each memory vector is used. Sync marker bits in the other samples are ignored.

Dac Value binary format in precision mode:

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	SYNM	SMPM

Dac Value binary format in speed mode:

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

7.22.5 Waveform Data Format in Interpolated Mode

I and Q values have 15 bit resolution in interpolated mode. I and Q values are signed. Valid range is -16384 to +16383.

Marker position data consists of one bit for a sample marker (SMPM) and one bit for a sync marker (SYNM) for each I/Q sample pair. The 24 I and Q samples are stored interlaced. The marker position data is stored together with I and Q values in two 16-bit signed integers.

Only the sync marker bit in the first sample of each memory vector is used. Sync marker bits in the other samples are ignored.

The same data format can be used for all four interpolated modes (x3, x12, x24 and x48).

I/Q value binary format in interpolated mode:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0	SMPM
Q14	Q13	Q12	Q11	Q10	Q9	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0	SYNM

7.22.6 Arbitrary Waveform Generation

To prepare your module for arbitrary waveform generation follow these steps:

Select one of the direct modes (precision or speed mode) or one of the interpolated modes ((x3, x12, x24 and x48) using

:TRACe[1|2]:DWIDth

Define one or multiple segments using the various forms of the `:TRAC[1|2]:DEF` command

Fill the segments with values and marker data using `:TRAC[1|2]:DATA`

Select the segment to be output in arbitrary waveform mode using `:TRAC[1|2]:SEL`

Signal generation starts after calling `INIT:IMM`

7.22.7 :TRACe[1|2]:DEFine <segment_id>,<length>[,<init_value1>[,<init_value2>]]

Command `:TRAC:DEF`

Long `:TRACe:DEFine`

Parameters

<segment_id>,<length>[,<init_value1>[,<init_value2>]]

- <segment_id > – id of the segment, 1..512k for option SEQ, 1 if not installed
- <length> – length of the segment in samples for direct modes or in I/Q sample pairs for interpolated modes
- <init_value1> – optional initialization value. For direct modes this is a DAC value. For interpolated modes this is the I-part of an I/Q sample pair.
- <init_value2> – optional initialization value, only applicable for interpolated modes. This is the Q-part of an I/Q sample pair.

Parameter Suffix

None

Description

Use this command to define the size of a waveform memory segment. If <init_value1> is specified (direct modes) or <init_value1> and <init_value2> (interpolated modes) are specified, all sample values in the segment are initialized. If not specified, memory is only allocated but not initialized.

Example

Commands

To set precision mode

```
TRAC1:DWID WPR
```

To define a segment with id 1 and length 480 samples. Initialize to sample value 0.

```
TRAC1:DEF 1,480,0
```

To set interpolated mode, interpolation factor 3.

```
TRAC1:DWID INTX3
```

To define a segment with id 1 and length 480 samples. Initialize with I/Q value pair (0,1).

```
TRAC:DEF 1,480,0,1
```

7.22.8 :TRACe[1|2]:DEFine:NEW? <length> [,<init_value1>[,<init_value2>]]

Command :TRAC:DEF:NEW?

Long :TRACe:DEFine:NEW?

Parameters <length> [,<init_value1>[,<init_value2>]]

- <length> – length of the segment in samples for direct modes or in I/Q sample pairs for interpolated modes
- <init_value1> – optional initialization value. For direct modes this is a DAC value. For interpolated modes this is the I-part of an I/Q sample pair.
- <init_value2> – optional initialization value, only applicable for interpolated modes. This is the Q-part of an I/Q sample pair.

Parameter Suffix None

Description Use this query to define the size of a waveform memory segment. If <init_value1> is specified (direct modes) or <init_value1> and <init_value2> (interpolated modes) are specified, all sample values in the segment are initialized. If not specified, memory is only allocated but not initialized. If the query was successful, a new <segment_id> will be returned.

Example Query

Define a segment of length 480 samples. Returns the segment id.

```
TRAC1:DEF:NEW? 480
```

7.22.9 :TRACe[1|2]:DEFine:WONLy <segment_id>,<length>[,<init_value1>[,<init_value2>]]

Command :TRAC:DEF:WONL

Long :TRACe:DEFine:WONLy

Parameters <segment_id>,<length>[,<init_value1>[,<init_value2>]]

- < segment_id > – id of the segment, 1..512k for option SEQ, 1 if not installed
- <length> – length of the segment in samples for direct modes or in I/Q sample pairs for interpolated modes
- <init_value1> – optional initialization value. For direct modes this is a DAC value. For interpolated modes this is the I-part of an I/Q sample pair.
- <init_value2> – optional initialization value, only applicable for interpolated modes. This is the Q-part of an I/Q sample pair.

Parameter Suffix None

Description Use this command to define the size of a waveform memory segment. If <init_value1> is specified (direct modes) or <init_value1> and <init_value2> (interpolated modes) are specified, all sample values in the segment are initialized. If not specified, memory is only allocated but not initialized. The segment will be flagged write-only, so it cannot be read back or stored.

Example Command

Define a write-only segment with id 1 and length 480 samples.

```
:TRAC1:DEF:WONL 1,480
```

7.22.10 :TRACe[1|2]:DEFine:WONLy:NEW? <length>[,<init_value1>[,<init_value2>]]

Command :TRAC:DEF:WONL:NEW?

Long :TRACe:DEFine:WONLy:NEW?

Parameters <length>[,<init_value1>[,<init_value2>]]

- <length> – length of the segment in samples for direct modes or in I/Q sample pairs for interpolated modes
- <init_value1> – optional initialization value. For direct modes this is a DAC value. For interpolated modes this is the I-part of an I/Q sample pair.
- <init_value2> – optional initialization value, only applicable for interpolated modes. This is the Q-part of an I/Q sample pair.

Parameter Suffix None

Description Use this query to define the size of a waveform memory segment. If <init_value1> is specified (direct modes) or <init_value1> and <init_value2> (interpolated modes) are specified, all sample values in the segment are initialized. If not specified, memory is only allocated but not initialized. If the query was successful, a new <segment_id> will be returned. The segment will be flagged write-only, so it cannot be read back or stored.

Example Query

Define a write-only segment with id 1 and length 480 samples.

```
:TRAC1:DEF:WONL:NEW? 2400
```

7.22.11 :TRACe[1|2]:DWIDth[?] WSPeed|WPRecision|INTX3|INTX12|INTX24|INTX48

Command `:TRAC:DWID[?]`

Long `:TRACe:DWIDth[?]`

Parameters WSPeed|WPRecision|INTX3|INTX12|INTX24|INTX48

- WSPeed – speed mode, 12 bit DAC resolution
- WPRecision – precision mode, 14 bit DAC resolution
- INTX3 – interpolation x3 mode
- INTX12 – interpolation x12 mode
- INTX24 – interpolation x24 mode
- INTX48 – interpolation x48 mode

Parameter Suffix None

Description Use this command or query to set or get the waveform output mode. If the DAC resolution is changed in direct mode or if there is a switch from direct mode to interpolated mode or vice versa, all waveform segments for this channel are invalidated.

The interpolated modes INTX3, INTX12, INTX24 and INTX48 are only available when the DUC option is installed.

Example

Command
```
:TRAC:DWID INTX3
```

Query
```
:TRAC:DWID?
```

7.22.12 :TRACe[1|2][:DATA][?] <segment_id>,<offset>,(<length>|<block>|<numeric_values>)

Command :TRAC:DATA[?]

Long :TRACe:DATA[?]

Parameters <segment_id>,<offset>,(<length>|<block>|<numeric_values>)

- < segment_id > – id of the segment, 1..512k for option SEQ, 1 if not installed
- <offset> offset in samples for direct modes and I/Q sample pairs for interpolated modes to allow splitting the transfer in smaller portions. The offset parameter is necessary to overcome the SCPI restriction that only allows transferring up to 999999999 bytes at once.
- <block> waveform data samples and marker values in the data format described above in IEEE binary block format
- <numeric_values> waveform data samples and marker values in the data format described above in comma separated list format; each element is a 16-bit integer values representing DAC samples and marker data Di in direct modes or I/Q samples and marker data Ii, Qi in interpolated modes.
- Direct mode: D0, D1, D2,...
- Interpolated mode: I0, Q0, I1, Q1, I2, Q2...

Parameter Suffix None

Description Use this command to load waveform and marker data into the module memory. If <segment_id> is already filled with data, the new values overwrite the current values. If length is exceeded error -223 (too much data) is reported.

NOTE

If the segment is split in smaller sections, the sections have to be written in order of ascending <offset> values. If modification of the segment contents is necessary, the whole segment with all sections must be rewritten.

If segments are created and deleted in arbitrary order, their position and order in memory cannot be controlled by the user, because the M8190 reuses the memory space of deleted segments for newly created segments. To fulfill the streaming and minimum linear playtime requirements the only way to control the position of the first downloaded segment and the order of the following segments is to delete all segments from memory (:TRACe[1|2]:DELete:ALL) and then creating the segments in the order in which they should be placed in memory.

This SCPI has the following syntax for command/query:

Command

:TRACe[1|2][:DATA] <segment_id>,<offset>,(<block>|<numeric_values>)

Query

:TRACe[1|2][:DATA][?] <segment_id>,<offset>,<length>

Example

Command

Load data consisting of 480 samples as comma-separated list into previously defined segment 1 starting at sample offset 0.

```
:TRAC1:DATA 1,0,0,1,2,…,479
```

Query

```
:TRAC:DATA? 1,0,48
```

7.22.13 :TRACe[1|2]:IQIMport <segment_id>,<file_name>,TXT|TXT14|BIN|IQBIN|BIN6030|BIN5110|LICensed|MAT89600|DSA90000|CSV,IONLy|QONLy|BOTH,ON|OFF|1|0,[,ALENgth|FILL|[,<init_valueI>[,<init_valueQ>[,<ignore_header_parameters>]]]

Command :TRAC:IQIM

Long :TRACe:IQIMport

Parameters

<segment_id>,<file_name>,TXT|TXT14|BIN|IQBIN|BIN6030|BIN5110|LICensed|MAT89600|DSA90000|CSV,IONLy|QONLy|BOTH,ON|OFF|1|0,[,ALENgth|FILL|[,<init_valueI>[,<init_valueQ>[,<ignore_header_parameters>]]

File Format

Import segment data from a file. Different file formats are supported. An already existing segment can be filled, or a new segment can be created. This can be used to import real waveform data as well complex I/Q data.

<segment_id> 1...512k for option SEQ, 1 if not installed.

<file_name> file name.

TXT|TXT14|BIN|IQBIN|BIN6030|BIN5110|LICensed|MAT89600|DSA90000|CSV. (See *File Type*)

<data_type> IONLy|QONLy|BOTH

<marker_flag> ON|OFF|1|0 (See Marker Flag)

<padding> ALENgth|FILL (See Padding)

<init_valueI> – optional initialization value. For non-I/Q format this is a DAC value. For I/Q file format this is the I-part of an I/Q sample pair in binary format (int16). Defaults to 0 if not specified.

<init_valueQ> – optional initialization value, only applicable for I/Q file formats. This is the Q-part of an I/Q sample pair in binary format (int16). Defaults to 0 if not specified.

<ignore_header_parameters> ON|OFF|1|0 (See *Ignore Header Parameters*)

File Type

TXT

Compatibility: Keysight M8190A Tek AWG 7000

Normalized values (-1.0 .. +1.0) and (optionally marker values) separated by ',' or ';' or '\t'. Not given markers values are just set to 0. Space ' ' and '\t' are ignored. Line end can be \r or \r\n.

Example (US locale)

0.7,0,1

0.9,1

Example (German locale):

0,7;0;1

0,9;1

NOTE

In German locale it is recommended (but not required) to use ';' or '\t' as separator. But it must then be ensured that the double really has a decimal point (',') or there is some space inserted to ensure correct parsing:

0,7,0,1

0 ,0,1

TXT14

Compatibility: Keysight M8190A Tek AWG 5000

14 digit DAC value + 2 Marker values

<D13>,<D12>,<D11>,<D10>,<D9>,<D8>,<D7>,<D6>,<D5>,<D4>,<D3>,<D2>,<D1>,<D0>,<M1>,<M2>

BIN

Compatibility: Keysight M8190A. Recommended as standard binary file import format.

Binary int16 values (in little endian byte order) directly in M8190A 14 bit precision mode format:

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	SYNM	SMPM

In 12 bit mode DB1 and DB0 are simply ignored.

IQBIN

Compatibility: Keysight M8190A. Recommended as standard binary file import format.

Binary I/Q values (in little endian byte order) directly in M8190A x12, x24 and x48 interpolated mode format, see description in *Waveform Data Format in Interpolated Mode*. In x3 interpolated mode I0 and Q0 are simply ignored.

BIN6030

Compatibility: Keysight N6030

Binary int16 values (in little endian byte order). The Keysight N6030 has 15 bits and uses the most significant digits, ignoring the LSB. While importing LSB bits are ignored (truncation to 14bit or 12bit depending on the DAC mode).

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	X	X

BIN5110

Compatibility: 5110A

Binary int16 I/Q sample pairs (in little endian byte order). May contain full 16 bit DAC values without the marker bits or 14 bit value plus two markers. While importing 16 bit values (without markers) the marker flag should be set to 'OFF' so that 2 LSB's are ignored.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0	X	SMPM
Q13	Q12	Q11	Q10	Q9	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0	X	SYNM

LICensed (SigStudioEncrypted in the SFP)

Compatibility: Keysight Signal Studio generated encrypted file

An encrypted file created with Keysight Signal Studio. This file contains I/Q sample pairs, markers, and some other waveform information, from which all but the sample rate is ignored.

This file type supports 8 markers per sample but we do not use all of them:

- Interpolated Mode or I Data in Direct Mode:
  bit 0 as the sample marker and bit 1 as the sync marker.
- Q Data in Direct Mode:
  bit 2 as the sample marker and bit 3 as the sync marker.

The sample rate is used to set the currently selected sample frequency (external or internal). If one of the interpolated modes is used, we set the sample frequency to a multiple of the sample rate accordingly, e.g. INTX3 à <M8190A sample frequency> = 3 x <sample rate from imported file>

MAT89600

Compatibility: 89600 VSA

89600 VSA recording file in MATLAB binary format (.mat) containing floating point values (without markers). Both real and complex I/Q data formats are supported. For I/Q format the values are stored as array of complex numbers with the real part corresponding to I value and the imaginary part corresponding to Q value. The header variable 'XDelta' (1/XDelta) is used to set the currently selected sample frequency (external or internal).. If one of the interpolated modes is used, we set the sample frequency to a multiple of the sample rate accordingly, e.g. INTX3 →

<M8190A sample frequency> = 3 x <1 / XDelta>. Only MATLAB level 4.0 and 5.0 files are supported.

DSA90000

Compatibility: DSA90000 Oscilloscope

DSA90000 waveform file in binary format (.bin) containing header and floating point data (without markers). Only waveform type 'Normal' is supported. If the file contains more than one waveform only the first waveform will be imported.

CSV

Compatibility: M8190A

Normalized values (-1.0 .. +1.0) and markers in comma delimited format. The file can contain either 1 (waveform data for channel 1), 2 (waveform data for channel 1 & 2), 3 (waveform data, sample marker, sync marker for channel 1) or 6 (waveform data, sample marker, sync marker for channel 1 and 2) columns. If the file contains data for two channels, it will be treated as IQ file.

Examples:

1 channel (without markers):

0.7

0.9

2 channel (without markers):

0.7,0.7

0.9,1.0

1 channel (with markers):

0.7,1,1

0.9,1,0

2 channel (with markers):

0.7,1,1,0.7,1,1

0.9,1,0,1.0,0,0

The CSV format may contain optional header information as follows:

Parameter Header

The parameter header contains optional header parameters as name and value pairs separated by '='. Each parameter should be specified in a single line. This header is optional. There are following header parameters:

SampleRate

The sample rate.

CarrierFrequencyIntegral
The integral part of the carrier frequency.

CarrierFrequencyFractional
The fractional part of the carrier frequency.

SetConfig
Flag to indicate if the header parameters need to be set. This can be set to either ‘true’ or ‘false’. If this flag is ‘false’ header parameters will not be set. If this flag is omitted header parameters are set.

Data Header
The data header contains the names of the data columns separated by ','. The waveform data are specified after the data header. This header is optional. If this header is not specified the data need to be defined similar to CSV files without the header (see above). The data columns are as follows:
Y1
Waveform data for channel 1.

SyncMarker1
Sync marker for channel 1.

SampleMarker1
Sample Marker for channel 1.

Y2
Waveform data for channel 2.

SyncMarker2
Sync marker for channel 2.

SampleMarker2
Sample Marker for channel 2.

If the file contains data for both channel 1 and 2 it will be treated as I/Q file. If any of the marker columns (SyncMarker or SampleMarker) is present for a channel the data header must contain the waveform data column (Y1 or Y2) for that channel. It is possible to have only the data columns (Y1 , Y2 or both) without the marker columns though.

Example:
SampleRate = 7.2 GHz
CarrierFrequencyIntegral = 2.0 GHz
CarrierFrequencyFractional = 1 Hz
Y1, SyncMarker1, SampleMarker1, Y2, SyncMarker2, SampleMarker2
0.7, 1, 1, 0.7, 1,1
0.9,1,0,1.0,0,0

Marker Flag

This flag is used to specify if the marker data need to be downloaded or not. If this flag is 'OFF' marker data will not be downloaded. Default value is 'ON'. This flag is applicable to BIN5110 format only . If BIN5110 format consists of full 16 bit DAC values (without markers) this flag should be set to 'OFF' so that 2 LSB's are ignored.

Padding

ALENgth: Automatic LENgth : <segment_id> may or may not exist. After execution <segment_id> has exactly the length of the pattern in file or a multiple of this length to fulfill granularity and minimum segment length requirements. Formula: length = granularity * max(5 , # samples in file). This behavior is default if <padding> is omitted.

FILL: <segment_id> must exist. If pattern in file is larger than the defined segment length, just ignore excessive samples. If pattern in file is smaller than defined segment length, fill remaining samples with <init_value>. <init_value> defaults to 0 if it is not specified.

Ignore Header Parameters Flag

This flag is used to specify if the header parameters need to be set. If this flag is 'ON' header parameters will not be set. This flag is optional and the default value is 'OFF' i.e. by default the header parameters are set . This flag is applicable to formats (CSV, LICensed and MAT89600) that contain header parameters.

Example

Command

```
:TRAC1:IQIMP 1, "C:\Program Files
(x86)\Keysight\M8190\Examples\WaveformDataFiles\Si
n1MHzAt7p68GHz.bin", BIN, IONLY, ON, ALEN
```

7.22.14 :TRACe[1|2]:IMPort <segment_id>,<file_name>,TXT|TXT14|TXTD|BIN|BIN6030[,ALENgth|FILL|[,<dac_value>]]

Command :TRAC:IMP

Long :TRACe:IMPort

Parameters <segment_id>,<file_name>,TXT|TXT14|TXTD|BIN|BIN6030[,ALENgth|FILL|[,<dac_value>]]

NOTE This command is deprecated and has been replaced by *:TRACe:IQIMport* to support additional file formats including I/Q data.

Import segment data from a file. Different file formats are supported. An already existing segment can be filled, or a new segment can be created.

<segment_id> 1...512k for option SEQ, 1 if not installed.

<file_name> file name.

<type> TXT|TXT14|TXTD|BIN|BIN6030. File format. (See *File Type*)

<padding> ALENgth|FILL. (See Padding)

<dac_value> a DAC value in binary format

Padding

ALENgth: Automatic LENgth : <segment_id> may or may not exist. After execution <segment_id> has exactly the length of the pattern in file or a multiple of this length to fulfill granularity and minimum segment length requirements. Formula: length = granularity * max(5 , # samples in file). This behavior is default if <padding> is omitted.

FILL: <segment_id> must exist. If pattern in file is larger than the defined segment length, just ignore excessive samples. If pattern in file is smaller than defined segment length, fill remaining samples with <dac_value>. <dac_value> defaults to 0 if it is not specified.

File Type

TXT

Compatibility: Keysight M8190A Tek AWG 7000

Normalized values (-1.0 .. +1.0) and (optionally marker values) separated by ',' or ';' or '\t'. Not given markers values are just set to 0. Space ' ' and '\t' are ignored. Line end can be \r or \r\n.

Example (US locale)

0.7,0,1

0.9,1

Example (German locale):

0,7;0;1

0,9;1

NOTE

In German locale it is recommended (but not required) to use ';' or '\t' as separator. But it must then be ensured that the double really has a decimal point (',') or there is some space inserted to ensure correct parsing:

0,7,0,1

0 ,0,1

TXT14

Compatibility: Keysight M8190A Tek AWG 5000

14 digit DAC value + 2 Marker values

<D13>,<D12>,<D11>,<D10>,<D9>,<D8>,<D7>,<D6>,<D5>,<D4>,<D3>,<D2>,<D1>,<D0>,<M1>,<M2>

TXTD

Compatibility: Keysight M8190A. Recommended as standard ASCII file import format.

Decimal DAC Value -8191 .. +8191 and (optionally up to two marker values) separated by ',' or ';' or '\t'. Not given markers values are just set to 0. Space ' ' and 't' are ignored. Line end can be \r or \r\n.

Example:

0,1,0

5

10,1,0

BIN

Compatibility: Keysight M8190A. Recommended as standard binary file import format.

Binary int16 values (in little endian byte order) directly in M8190A 14 bit precision mode format:

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	SYNM	SMPM

In 12 bit mode DB1 and DB0 are simply ignored.

BIN6030

Compatibility: Keysight N6030

Binary int16 values (in little endian byte order). The Keysight N6030 has 15 bits and uses the most significant digits, ignoring the LSB. While importing LSB bits are ignored (truncation to 14bit or 12bit depending on the DAC mode).

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	X	X

7.22.15 :TRACe[1|2]:DELete <segment_id>

Command `:TRAC:DEL`

Long `:TRACe:DELete`

Parameters <segment_id>

Parameter Suffix None

Description <segment_id> 1...512k for option SEQ, 1 if not installed

Delete a segment. The command can only be used in program mode.

Example Command

```
:TRAC:DEL 5
```

7.22.16 :TRACe[1|2]:DELete:ALL

Command :TRAC:DEL:ALL

Long :TRACe:DELete:ALL

Parameters None

Parameter Suffix None

Description Delete all segments. The command can only be used in program mode.

Example Command
:TRAC:DEL:ALL

7.22.17 :TRACe[1|2]:CATalog?

Command :TRAC:CAT?

Long :TRACe:CATalog?

Parameters None

Parameter Suffix None

Description The query returns a comma-separated list of segment-ids that are defined and the length of each segment. So first number is a segment id, next length ...
If no segment is defined, "0, 0" is returned.

Example
Query
```
:TRAC1:CAT?
```

7.22.18 :TRACe[1|2]:FREE?

Command
```
:TRAC:FREE?
```

Long
```
:TRACe:FREE?
```

Parameters None

Parameter Suffix None

Description The query returns the amount of memory space available for waveform data in the following form: <bytes available>, <bytes in use>, < contiguous bytes available>.

Example
Query
```
:TRAC:FREE?
```

7.22.19 :TRACe[1|2]:SELect[?] <segment_id>

Command :TRAC:SEL[?]

Long :TRACe:SELect[?]

Parameters <segment_id>

Parameter Suffix None

Description Selects the segment, which is output by the instrument in arbitrary function mode. In dynamic segment selection mode select the segment that is played before the first segment is dynamically selected.

- <segment_id> – 1...512k for option SEQ, 1 if not installed.

Example

Command
```
:TRAC:SEL 5
```

Query
```
:TRAC:SEL?
```

7.22.20 :TRACe[1|2]:NAME[?] <segment_id>,<name>

Command :TRAC:NAME[?]

Long :TRACe:NAME[?]

Parameters <segment_id>,<name>

Parameter Suffix None

Description This command associates a name to a segment. The query gets the name for a segment.
<segment_id> – 1...512k for option SEQ, 1 if not installed
<name> – string of at most 32 characters

Example

Command
```
:TRAC:NAME 1,"ADY"
```

Query
```
:TRAC:NAME? 1
```

7.22.21 :TRACe[1|2]:COMMent[?] <segment_id>,<comment>

Command `:TRAC:COMM[?]`

Long `:TRACe:COMMent[?]`

Parameters <segment_id>,<comment>

Parameter Suffix None

Description This command associates a comment to a segment. The query gets the comment for a segment.
<segment_id> – 1...512k for option SEQ, 1 if not installed
<comment> – string of at most 256 characters

Example

Command
```
:TRAC:COMM 1, "Comment"
```

Query
```
:TRAC:COMM? 1
```

7.22.22 :TRACe[1|2]:ADVance[?] AUTO|CONDitional|REPeat|SINGle

Command

```
:TRAC:ADV[?]
```

Long

```
:TRACe:ADVance[?]
```

Parameters

AUTO | COND | REP | SING

Parameter Suffix

None

Description

Use this command or query to set or get the advancement mode for the selected segment. The advancement mode is used, if the segment is played in arbitrary mode.

Example

Command
```
:TRAC:ADV AUTO
```

Query
```
:TRAC:ADV?
```

7.22.23 :TRACe[1|2]:COUNt[?] <count>

Command :TRAC:COUN[?]

Long :TRACe:COUNt[?]

Parameters <count>

Parameter Suffix None

Description Use this command or query to set or get the segment loop count for the selected segment. The segment loop count is used, if the segment is played in arbitrary mode.

<count> – 1..4G-1: number of times the selected segment is repeated.

Example

Command
```
:TRAC:COUN 1
```

Query
```
:TRAC:COUN?
```

7.22.24 :TRACe[1|2]:MARKer[:STATe][?] OFF|ON|0|1

Command :TRAC:MARK[?]

Long :TRACe:MARKer[?]

Parameters OFF | ON | 0 | 1

Parameter Suffix None

Description Use this command to enable or disable markers for the selected segment. The query form gets the current marker state.

Example

Command
```
:TRAC:MARK 1
```

Query
```
:TRAC:MARK?
```

7.23 :TEST Subsystem

7.23.1 :TEST:PON?

Command :TEST:PON?

Long :TEST:PON?

Parameters None

Parameter Suffix None

Description Return the results of the power on self-tests.

Example Query
:TEST:PON?

7.23.2 :TEST:TST?

Command :TEST:TST?

Long :TEST:TST?

Parameters None

Parameter Suffix None

Description Same as `*TST?`, but the actual test messages are returned.

Example

Query

```
:TEST:TST?
```

NOTE Currently same as `:TEST:PON?`

7.24 CARRier Subsystem

7.24.1 [:SOURce]:CARRier[1|2]:FREQuency[?] <frequency_integral>[,<frequency_fractional>]|DEFault

Command

```
CARR:FREQ[?]
```

Long

```
:CARRier:FREQuency[?]
```

Parameters

<frequency_integral>

Parameter Suffix

Hz

Description

Set or query the carrier frequency used for interpolated modes. The command form expects an integral value in the range [0 .. 12] GHz and a fractional value in the range [0,1) Hz. To query the minimum and maximum values use the following two commands.
`[:SOURce]:CARRier[1|2]:FREQuency:INTegral[?] [<frequency_integral>|MIN|MAX|DEFault]` and
`[:SOURce]:CARRier[1|2]:FREQuency:FRACtional[?] [<frequency_fractional>|MIN|MAX|DEFault`.

- <frequency_integral> – Integral part of the carrier frequency (in Hz)
- <frequency_fractional> – Fractional part of the carrier frequency (optional, defaults to 0.0)

Example

Command
`:CARR1:FREQ 1e9,0.5`
Sets the carrier frequency to 1.0000000005GHz (1GHz + 0.5Hz).

Query
`:CARR1:FREQ?`

7.24.2 [:SOURce]:CARRier[1|2]:FREQuency:INTegral[?] <frequency_integral>|MIN|MAX|DEFault

Command `CARR:FREQ:INT[?]`

Long `:CARRier:FREQuency:INTegral[?]`

Parameters <frequency_integral> | MIN | MAX | DEF

Parameter Suffix Hz

Description Set or query the integral part of the carrier frequency used for interpolated modes.

- <frequency_integral> – Integral part of the carrier frequency (in Hz)

Example

Command
```
:CARR1:FREQ:INT 1e9
```

Query
```
:CARR1:FREQ:INT?
```

7.24.3 [:SOURce]:CARRier[1|2]:FREQuency:FRACtional[?] <frequency_fractional>|MIN|MAX|DEFault

Command CARR:FREQ:FRAC[?]

Long :CARRier:FREQuency:FRACtional[?]

Parameters <frequency_fractional> | MIN | MAX | DEF

Parameter Suffix Hz

Description Set or query the fractional part of the carrier frequency used for interpolated modes.

- <frequency_fractional> – Fractional part of the carrier frequency.

Example

Command
```
:CARR1:FREQ:FRAC 1e-9
```

Query
```
:CARR1:FREQ:FRAC?
```

7.24.4 [:SOURce]:CARRier[1|2]:SCALe[?] <scale>|MINimum|MAXimum|DEFault

Command CARR:SCAL[?]

Long :CARRier:SCALe[?]

Parameters <scale> | MIN | MAX | DEF

Parameter Suffix None

Description Set or query the amplitude scale used for interpolated modes.

Example

Command
```
:CARR1:SCAL 0.9
```
Sets the carrier amplitude scale to 0.9.

Query
```
:CARR1:SCAL?
```

7.24.5 [:SOURce]:CARRier[1|2]: POFFset [?] <phase- offset> |MINimum|MAXimum|DEFault

Command `CARR:POFF[?]`

Long `:CARRier:POFFset[?]`

Parameters <phase-offset> | MIN | MAX | DEF

Parameter Suffix None

Description Set or query the carrier phase offset used for interpolated modes.

Example

Command
```
:CARR1:POFF 0.1
```
Set the carrier phase offset to 0.1 cycles.

Query
```
:CARR1:POFF?
```

7.25 :FTABle Subsystem

The FTABle (Frequency Table) subsystem is available in interpolated modes. It offers the following functionality:

- Definition of up to 32K frequencies that can afterwards be used as carrier frequencies for the I/Q data segments referenced from sequence table entries.

NOTE Changes to the frequency table are only possible while the corresponding channel is idle, i.e. not armed and not generating a waveform.

7.25.1 [:SOURce]:FTABle[1|2]:DATA[?] <frequency_table_index>, (<length>|<block>|<frequency_integral_part>,<frequency_fractional_part>...)

Command `:FTAB:DATA[?]`

Long `:FTABle:DATA[?]`

Parameters <frequency_table_index>

Parameter Suffix Hz

Description The command form writes one or multiple frequency table entries starting at <frequency_table_index>. The query form reads <length> entries starting at <frequency_table_index>. The data for each entry consists of a double value for the integral and the fractional part of the frequency each. The data can be given in IEEE binary block format or as a comma-separated list of doubles (in Hz).

Example

Command
```
:FTAB:DATA 0,1
```

Query
```
:FTAB:DATA? 0,1
```

7.25.2 [:SOURce]:FTABle[1|2]:RESet

Command

```
:FTAB:RES
```

Long

```
:FTABle:RESet
```

Parameters

None

Parameter Suffix

None

Description

Reset all frequency table entries to default values.

Example

Command
```
:FTAB:RES
```

7.26 :ATABle Subsystem

The ATABle (Amplitude Table) subsystem is available in interpolated modes. It offers the following functionality:

- Definition of up to 32K amplitude scale values that can afterwards be used to scale the amplitudes for the I/Q data segments referenced from sequence table entries.
- Range: [0.0, 1.0]

NOTE Changes to the amplitude table are only possible while the corresponding channel is idle, i.e. not armed and not generating a waveform

7.26.1 [:SOURce]:ATABle[1|2]:DATA[?] <amplitude_table_index>, (<length>|<block>|<amplitude_scale>,<amplitude_scale>...)

Command `:ATAB:DATA[?]`

Long `:ATABle:DATA[?]`

Parameters None

Parameter Suffix None

Description Reset all amplitude table entries to default values.

Example

Command
```
:ATAB:DATA 0,1
```

Query
```
:ATAB:DATA? 0,1
```

The command form writes one or multiple amplitude table entries starting at <amplitude_table_index>. The query form reads <length> entries starting at <amplitude_table_index>. The data for each entry consists of a double value for the amplitude scale factor in the range [0.0, 1.0]. The data can be given in IEEE binary block format or as a comma-separated list of values.

7.26.2 [:SOURce]:ATABle[1|2]:RESet

Command `:ATAB:RES`

Long `:ATABle:RESet`

Parameters None

Parameter Suffix None

Description Reset all amplitude table entries to default values.

Example

Command
```
:ATAB:RES
```

7.27 :ACTion Subsystem

The ACTion subsystem is available in interpolated modes. It offers the following functionality:

- Definition of actions (commands) that can afterwards be referenced from sequence table entries.
- Basic commands can be grouped together to form more complex "composite commands" or "action sequence".
- These “composite commands” form the “action table”.

NOTE

There are no restrictions regarding changes to the action table while the corresponding channel is idle, i.e. not armed and not generating a waveform. As soon as the channel leaves the idle state, some restrictions apply:

- Only changes are allowed, which do not change the size of an action sequence or the action table. That is, only the [:SOURce]:ACTion[1|2]:APPend command can be used, but only to replace a previously appended value (like carrier frequency) or to replace e.g. a previously appended “sweep run” with “sweep hold”.
- The action sequence must not be in use while it is being changed.

After stopping waveform generation, the restrictions do not apply any more.

7.27.1 [:SOURce]:ACTion[1|2]:DEFine[:NEW]?

Command `:ACT:DEF?`

Long `:ACTion:DEFine?`

Parameters None

Parameter Suffix None

Description This query defines a new empty action sequence and returns an <action-sequence-index>. Steps can then be added to the sequence using the [:SOURce]:ACTion[1|2]:APPend command. If the action table is full, an error is returned.

Example

Query
```
:ACT:DEF?
```

7.27.2 [:SOURce]:ACTion[1|2]:APPend <action_sequence_index>,<action>[,<value> [,<value>]]

Command `:ACT:APP`

Long `:ACTion:APPend`

Parameters <action_sequence_index>,<action>

Parameter Suffix None

Description

<action_sequence_index>: the number returned by [:SOURce]:ACTion[1|2]:DEFine[:NEW]?

- <action>: Basic command enum
- <value>: Value(s) to be set

This command defines the next step of an action sequence. Each step consists of an action identifier enum and 0 or more double values.

If the action table is full, an error is returned.

If the same <action> is appended more than once, only its last value is used.

If more than on <action> in the set {Sweep Run, Sweep Hold, Sweep Restart} is issued, all but the last are discarded.

The default value for the optional fractional parts is 0.0.

Example

Command
```
:ACT:APP 1,CFR,0,0.1
```

Table 7-28: [:SOURce]:ACTion:APPend <action> enums

	Mnemonic	Value 1	Value 2
Carrier Frequency	CFRequency	Frequency, integral part (in Hz)	Frequency, fractional part (in Hz)
Phase Offset	POFFset	Phase (-0.5...+0.5 cycles)	N/A
Phase Reset	PRESet	Phase (0.0...1.0 cycles)	N/A
Phase Bump	PBUMp	Phase (-0.5...+0.5 cycles)	N/A
Sweep Rate	SRATe	Sweep Rate, integral part (Hz/µs)	Sweep Rate, fractional part (Hz/µs)
Sweep Run	SRUN	N/A	N/A
Sweep Hold	SHOLd	N/A	N/A
Sweep Restart	SREStart	N/A	N/A
Amplitude	AMPLitude	Amplitude (0.0...1.0 relative factor)	N/A

7.27.3 [:SOURce]:ACTion[1|2]:DELete <action_sequence_id>

Command

```
:ACT:DEL
```

Long

```
:ACTion:DELete
```

Parameters

<action_sequence_id>

Parameter Suffix None

Description <action-sequence-id> Id of the action sequence to be deleted
This command deletes an action sequence from the action table.

Example Command
```
:ACT:DEL 1
```

7.27.4 [:SOURce]:ACTion[1|2]:DELete:ALL

Command `:ACT:DEL:ALL`

Long `:ACTion:DELete:ALL`

Parameters None

Parameter Suffix None

Description This command deletes the complete action table.

Example Command
```
:ACT:DEL:ALL
```

7.28 Example Programs

Example programs can be found in your installation of the M8190 software.

8 Characteristics

8.1 Performance Specification

The performance specification can be found at:
http://www.keysight.com/find/M8190A

8.2 General

Power consumption	210 W (nom), 12 GSa/s operation
Operating temperature	0 °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.

Appendix

A.1 Resampling Algorithms for Waveform Import

A.1.1 Resampling Requirements

Resampling is typically associated to a series of processes applied to a waveform sampled at a given sampling frequency to generate a new waveform with a different sampling rate while preserving all the original information contained in the signal within the Nyquist bandwidth corresponding to the output sampling rate. Processes involved in resampling may vary depending on the output to input sampling rate ratio (or resampling factor) and the integer nature of the ratio itself. Resampling calculations, when applied to arbitrary waveform generation, must meet additional constraints such as available record length boundaries, record length granularity requirements, or acceptable sampling rate range.

Typically the characteristics of the input waveform (sampling rate, record length) are externally defined (i.e. by the horizontal settings of an oscilloscope used to capture the waveform). Users may be interested in resampling the signal to adapt the input waveform to the AWG requirements or the user desires. In some cases it may be necessary to reduce the sampling rate if it has been captured at a higher sampling rate than the one allowed by the AWG or to reduce the record length required to generate it. The opposite is also true as oversampling may help to “smooth” the signal as increasing sampling rate will shift the images created by the DAC to a higher frequency. Finally, resampling may be also necessary to adapt the record length of the input waveform to a legal record length that can be applied to a real AWG (i.e. to meet the record length granularities) without applying truncation or “zero padding” to the input waveform.

A.1.2 Resampling Methodology

Generally speaking, resampling factors do not have to be an integer or a simple fractional ratio. Because of that, traditional methods based in upsampling/filtering/decimation techniques may not be suitable given the amount of calculations resulting from the typical input waveform sizes involved. Instead of this, a more straight forward approach has been chosen. This approach is based in the following principles:

- Only output samples will be calculated so there is not any up-sampling and/or down-sampling operations involved.
- Filtering calculations will be kept to a minimum by using a filter with a fast enough roll-off and sufficient stop band attenuation according to the target AWG dynamic range.
- Interpolation filter and anti-alias filters are exactly the same although the filter parameters will depend on the resampling parameters.
- The implemented algorithm does perform filtering and interpolation simultaneously so the number of calculations is greatly reduced. Additionally, filters are implemented as look-up tables so those are calculated only once during the process.
- Timing parameters are based in double precision floating-point numbers while amplitude related parameters are single precision numbers. Most calculations consist in multiplication/addition operations required by convolution processes and only involve amplitude related variables (input samples and filter coefficients). Single precision numbers will minimize calculation time while offering more than enough dynamic range.

Interpolators and anti-aliasing filters share most characteristics as they are required to be low-pass with good flatness, linear phase, fast roll-off, and high stop-band rejections ratio. Ideal interpolator filters show a “brick-wall” response. However, such filters require a very long “sinc-like” impulse response to obtain good-enough performance. Impulse response length has a direct effect on calculation times resulting of applying the filter. Roll-off characteristics are especially important when applying the filter as the anti-alias filter required for down-sampling. The filter implemented in these algorithms has been designed with the following objectives:

- Pass band flatness better than 0.01 dB
- Stop band attenuation better than 80 dB
- F80dB/F3dB ratio better than 1.15

The final filter consists in a sinc signal with a 41 sample periods length after applying a Blackman-Harris time-domain window.

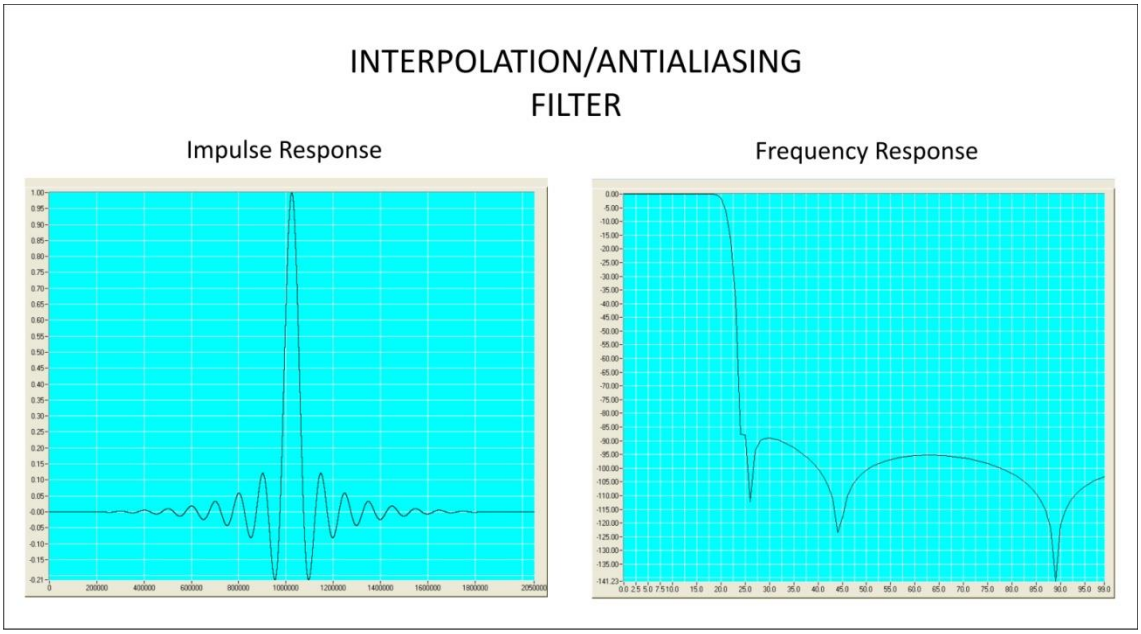


Figure 8-1: Interpolation/Antialiasing Filter

The filter shape remains the same no matter the resampling characteristics. For resampling ratios greater than 1.0, filter will implement an interpolator so nulls in the impulse response must be located at multiples of the sampling period of the input signal. For ratios lower than 1.0 the filter will implement an antialiasing filter. In this case, distance between nulls will have to be longer than the output waveform sampling period so the filter reach the required attenuation (>80dB) at the output signal Nyquist frequency. For the implemented filter this is accomplished by choosing 0.89 ratio between the output sampling period and the distance between consecutive nulls in the filter's impulse response.

Edition 11, August 2014
M8190-91030
www.keysight.com