

Keysight M8085A MIPI[®] C-PHY Editor

User Guide

Notices

© Keysight Technologies 2018-2019

No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Keysight Technologies as governed by United States and international copyright laws.

Trademarks

MIPI® C-PHY™ and MIPI® D-PHY™ are registered trademarks of the MIPI Alliance.

Manual Part Number

M8085-91010

Edition

Edition 6.3, July 2019

Keysight Technologies Deutschland GmbH
Herrenberger Strasse 130,
71034 Böblingen, Germany

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.

U.S. Government Rights

The Software is “commercial computer software,” as defined by Federal Acquisition Regulation (“FAR”) 2.101. Pursuant to FAR 12.212 and 27.405-3 and Department of Defense FAR Supplement

(“DFARS”) 227.7202, the U.S. government acquires commercial computer software under the same terms by which the software is customarily provided to the public. Accordingly, Keysight provides the Software to U.S. government customers under its standard commercial license, which is embodied in its End User License Agree-

ment (EULA), a copy of which can be found at <http://www.keysight.com/find/sweula>. The license set forth in the EULA represents the exclusive authority by which the U.S. government may use, modify, distribute, or disclose the Software. The EULA and the license set forth therein, does not require or permit, among other things, that Keysight: (1) Furnish technical information related to commercial computer software or commercial computer software documentation that is not customarily provided to the public; or (2) Relinquish to, or otherwise provide, the government rights in excess of these rights customarily provided to the public to use, modify, reproduce, release, perform, display, or disclose commercial computer software or commercial computer software documentation. No additional government requirements beyond those set forth in the EULA shall apply, except to the extent that those terms, rights, or licenses are explicitly required from all providers of commercial computer software pursuant to the FAR and the DFARS and are set forth specifically in writing elsewhere in the EULA. Keysight shall be under no obligation to update, revise or otherwise modify the Software. With respect to any technical data as defined by FAR 2.101, pursuant to FAR 12.211 and 27.404.2 and DFARS 227.7102, the U.S. government acquires no greater than Limited Rights as defined in FAR 27.401 or DFAR 227.7103-5 (c), as applicable in any technical data.

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 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 WILL CONTROL.

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.

Contents

1 Introduction

Overview	8
Overview of MIPI C-PHY Functionality	9
Overview of Lane Signaling States	9
Representation of Symbols in High-Speed Mode	10
High-Speed Data Transmission Burst	11
16-Bit to-7-Symbol Mapping	14
Transmit Lane PRBS Register Operation	15
ISI Generation - S6P Support	16
Start Pattern and Triggered Start	17
De-skew without Re-cabling	17
Duty Cycle Distortion	17
Skew Calibration Values	21
Hysteresis	21
Multi-lane Structure	26
Multi-lane Support:	26

2 Sequence and Data Files

Patterns, Data Format and Sequences	28
*.ptrn File Format (P Macro)	29
*.dat or *.txt File Format (B/LPB Macro)	32
*.seq File Format	33
CSI and DSI Sequences	38

Sequence File Definition for CSI	41
Overview	41
Long and Short Packet Formats	41
Frame and Line Synchronization Packets	43
Understanding a CSI sequence file	46
Calculating HS Data Rate for CSI sequence	48
Sequence File Definition for DSI	50
Overview	50
Long and Short Packet Formats	51
DSI Sequence Format Description	52
Transmission Packet Sequences	54
Replacing B-Macros with C-Macros in a sequence file	58
Understand a DSI sequence file	62
Calculating HS Data Rate for the DSI sequence	65

3 Using the MIPI C-PHY Editor User Interface

Basic Requirements	68
Hardware Setup using M8195A AWG Module	68
Software Requirements	71
License Requirements	71
Installing Plug-in	73
Related Documents	74
Starting the MIPI C-PHY Editor Plug-in	75
Contacting Keysight Technologies	77
MIPI C-PHY Editor User Interface	78
Connection Setup	80
Main User Interface	83

Setting up MIPI C-PHY Editor Parameters 97

Signal Levels Group	98
Protocol Group	112
Jitter Group	114
Intersymbol Interference Group	116
Disturbances Group	118
Signal Interference Group	123
Skew Group	125
Delay Group	127

Performing In-System AWG Calibration with Keysight IQ Tools 128**4 SCPI Programming****SCPI Command Language 144**

Data Types	144
Important Points about SCPI	145
SCPI Editor	148

SCPI Command Reference 150

SCPI Commands for connection to instruments	150
SCPI Commands for Data Pattern Group	156
SCPI Commands for Data Rate and Transition Time Group	165
SCPI Commands for Delay Group	168
SCPI Commands for Disturbances Group	172
SCPI Commands for InterSymbol Interference Group	178
SCPI Commands for Jitter Group	182
SCPI Commands for Protocol Group	187
SCPI Commands for Signal Interference Group	194
SCPI Commands for Signal Levels Group	197
SCPI Commands for Skew Group	201
Other SCPI Commands	203
Remote Queries to find Parameter ranges (maximum and minimum limits)	210

1 Introduction

[Overview](#) / 8

[Multi-lane Structure](#) / 26

This chapter describes a high-speed serial interface called MIPI[®] C-PHY, which provides high throughput performance over bandwidth limited channels for connecting to peripherals.

Overview

The M8070B software has an add-on **MIPI C-PHY Editor** that generates MIPI C-PHY signals so as to test the DUTs that are compatible to this standard.

The MIPI C-PHY describes a high-speed, rate-efficient PHY where channel rate limitations are a factor. The requirements for rate limited channels are accomplished through the use of 3-Phase symbol encoding technology delivering approximately 2.28 bits per symbol over a three-wire group of conductors. This MIPI C-PHY specification (<http://mipi.org/>) has been written primarily for the connection of cameras and displays to a host processor. Nevertheless, it can be applied to many other applications.

Key characteristics of MIPI C-PHY (High-Speed Mode) are:

- Uses a group of three conductors rather than differential pairs. The group of three wires is called a lane, and the individual lines of the lane are called: A, B and C. MIPI C-PHY does not have a separate clock lane.
- Within a three-wire lane, two of the three wires are driven to opposite levels. All wires are terminated at the end to the same star termination point. At one single time one line needs to have high, one line mid and the third line low level. No two lines may have the same level at one point in time. The voltages at which the wires are driven changes at every symbol.
- Multiple bits are encoded into each symbol epoch, the data rate is ~2.28x the symbol rate. There is a transition coding.
- Clock timing is encoded into each symbol. This is accomplished by requiring that the combination of voltages driven onto the wires must change at every symbol boundary on at least one wire. This simplifies clock recovery.

The **MIPI C-PHY Editor** is a licensed feature. To enable the **MIPI C-PHY Editor**, the required licenses/options are mentioned in chapter 3 “**License Requirements**” on page 71:

- M8195A requires the following license options:
 - Option -001, -002, or -004: With these options the number of channels is selected. The M8195A is available in a one channel (-001), two channel (-002) or 4 channel (-004) version. A software upgrade from one to two channels is possible by installing option U02. A software upgrade from two to four channels is possible by installing option U04. In order to upgrade from one to four channels, first option -U02 and next -U04 must be installed.
 - Option -16G: This option offers 16384 MSa (=16 GSa) waveform memory for the M8195A. Option -16G is software upgradeable.

- Option -SEQ: This option offers extensive sequencing capabilities.
- Option -FSW: This option enables the M8195A to externally select or step through segments or sequences faster than every 500 μ s. Option -FSW is export controlled and is software upgradeable.
- Option -1A7, -Z54: Calibration options.

For more details on how to install these licenses, refer to *Keysight M8000 Series of BER Test Solutions User Guide*.

Overview of MIPI C-PHY Functionality

MIPI C-PHY provides a synchronous connection between master and slave. A practical MIPI C-PHY configuration consists of one or more three-wire lanes. The link includes a high-speed signaling mode for fast-data traffic and a low-power signaling mode for control purposes. Optionally, a low-power escape mode can be used for low speed asynchronous data communication. High-speed data communication appears in bursts with an arbitrary number of payload data bytes.

The MIPI C-PHY uses three wires per lane, so three wires are required for the minimum MIPI C-PHY configuration. In high-speed mode each lane is typically terminated into a star point at mid level in case of the same high low levels on all three lines with a symmetric offset to the mid level driven by a low-swing, 3-Phase signal. In low-power mode all wires are operated single-ended and non-terminated. To minimize EMI, the drivers for this mode shall be slew-rate controlled and current limited.

The maximum achievable bit rate in high-speed mode is determined by the performance of transmitter, receiver and interconnect implementations. This specification is primarily intended to define a solution for a symbol rate range of 80 Msps to 4.5 Gsps per lane, which is equivalent to 182.8 Mbps to 10.29 Gbps per lane. Although MIPI C-PHY configurations are not limited to this range, practical constraints make it the most suitable range for the intended applications. For a fixed clock frequency, the available data capacity of a MIPI C-PHY configuration can be increased by using more lanes. Effective data throughput can be reduced by employing burst mode communication. The maximum data rate in low-power mode is 10 Mbps.

Overview of Lane Signaling States

A MIPI C-PHY configuration consists of one or more lanes. All lanes supports high-speed transmission and escape mode in the forward direction. The current flow through the lane for all six wire states. The positive-polarity wire states on the left and negative-polarity wire states

on the right. The three rotation states (x, y and z) are shown from top to bottom. The six driven states (called wire states) on a MIPI C-PHY lane are called: +x, -x, +y, -y, +z, and -z. The positive polarity wire states have the same wires driven as the corresponding negative polarity states, but the polarity is opposite on the driven pair of wires. For example: the +x wire state is defined as A being driven high and B driven low, while the -x wire state is B driven high and A driven low. The “undriven” conductor can be undriven when operating at lower symbol rates, or is actually driven by a termination at a voltage half way between the highest and lowest driven levels if operating at higher symbol rates.

Representation of Symbols in High-Speed Mode

One of the symbol to wire state encoding rules is that a state-transition exists at every symbol boundary. The reason for this rule is that it encodes the clock timing within the symbol, which has a number of advantages.

With six possible wire states there are always 5 possible transitions to the next wire state from any present wire state. The possible state transitions are illustrated in the state diagram in [Figure 1](#). The symbol value is defined by the change in wire state values from one unit interval to the next. Note that more than two bits of information (actually $\log_2(5) = 2.3219$ bits) can be encoded into each symbol. Seven consecutive symbols are used to transmit 16 bits of information. (Note that $5^7 = 78,125$ permutations in seven consecutive symbols, with five possible wire state transitions that define each symbol. The information encoded in seven symbols is more than sufficient to represent a 16-bit binary value, $2^{16} = 65,536$).

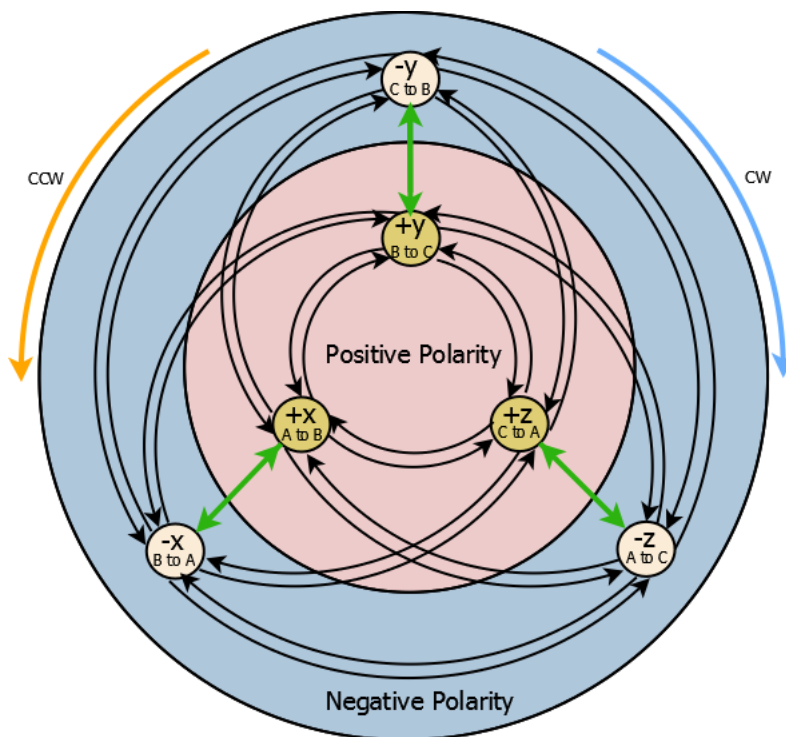


Figure 1 State Diagram showing all six wire states, and all possible transitions

High-Speed Data Transmission Burst

The sequence of events during the transmission of a Data Burst is shown in [Figure 2](#). For any lane, transmission can be started and ended independently by the protocol layer. However, for most applications the lanes will start synchronously but may end at different times due to an unequal amount of transmitted bytes per lane.

Beginning from the TS-HS-Exit state, LP-111, the signals transition to LP-001 and then LP-000 to signal that high-speed data transmission will begin soon. At the end of TX-HS-Prepare Duration, the low-power drivers are disabled and the high-speed drivers are enabled simultaneously. The first high-speed wire state transmitted at the beginning of TX-HS-Preamble Pattern shall be the “+x” state. This does not correspond to any particular symbol value because there is no previous HS wire state before it. It is recommended that the receive circuitry be initialized so the

non-existent prior state value corresponds to the “-z” state so the decoded symbol resulting from this first wire state is “3” (Flip = 0, Rotation = CW, Polarity = opposite). Although this first wire state shall be “+x”, it is likely that the first few wire states of the TX-HS-Preamble Pattern interval will not be seen at the high-speed receiver. This is because there will be some delay for the high-speed drivers to reach their required signal levels at the beginning of TX-HS-Preamble Pattern, and also the high-speed receivers will be enabled and at some point start producing outputs toward the end of TX-HS-Preamble. The receive circuitry shall be enabled toward the end of TX-HS-Preamble when it is safe for it to reliably decode the “3” symbols during TX-HS-Preamble. It is not guaranteed at exactly which symbol clock generation and symbol decoding will begin at the end of TX-HS-Preamble. The TX-HS-Preamble field may often consist of multiple groups of seven “3” symbols to provide a sufficient number of clocks to the upper layer protocol to initialize any pipeline stages prior to receiving data. The length of TX-HS-Preamble is a programmable value set in the master.

The master may output a programmable sequence during TX-HS-Prog-Seq Pattern of the preamble, if it is enabled using a programmable sequence enable bit such as the MSB of the control register. The symbol values transmitted in the programmable sequence, or whether the programmable sequence is used at all, is a choice of the system designer.

Figure 2 shows example of the preamble with the programmable sequence. Seven symbols of value “3” are sent during TX-HS-Pre-End Pattern just prior to sending the Sync Word.

The Sync Word precisely identifies the beginning of the Packet Data and also identifies the timing alignment of word boundaries in the Packet Data. The Sync Word contains a sequence of five “4” symbols which does not occur in any sequence of symbols generated by the Mapper. The Sync Word may also be transmitted later in the burst to mark the beginning of redundant Packet Headers transmitted by the upper layer protocol.

The end of Packet Data is identified by a unique sequence of “4” symbols in TX-HS-Post Pattern. The receiver identifies the end of Packet Data when it detects a sequence of seven consecutive “4” symbols. The Post field may often consist of multiple groups of seven “4” symbols to provide a sufficient number of clocks to the upper layer protocol to clear out any pipeline stages that may contain received data.

The high-speed drivers are disabled and the low-power drivers are enabled simultaneously, at the end of TX-HS-Post Pattern. All three signals of the lane are driven high together to LP-111, the TX-HS-Exit state.

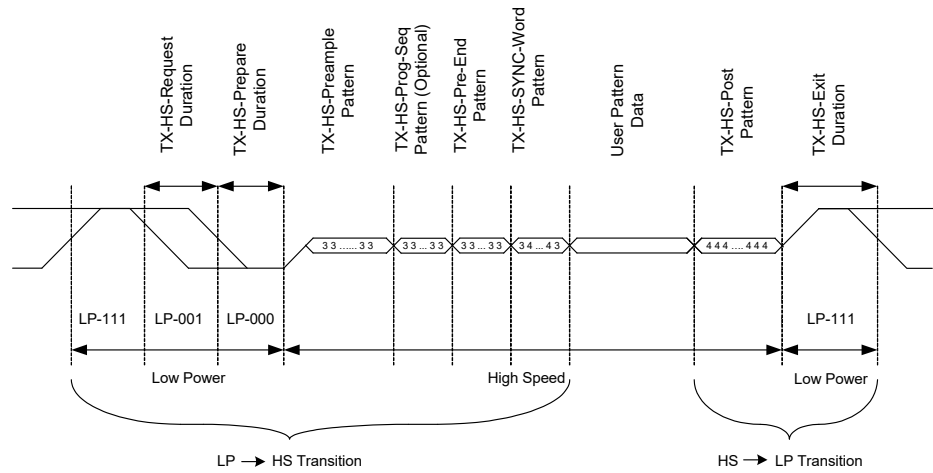


Figure 2 High-Speed Data Transmission in Burst

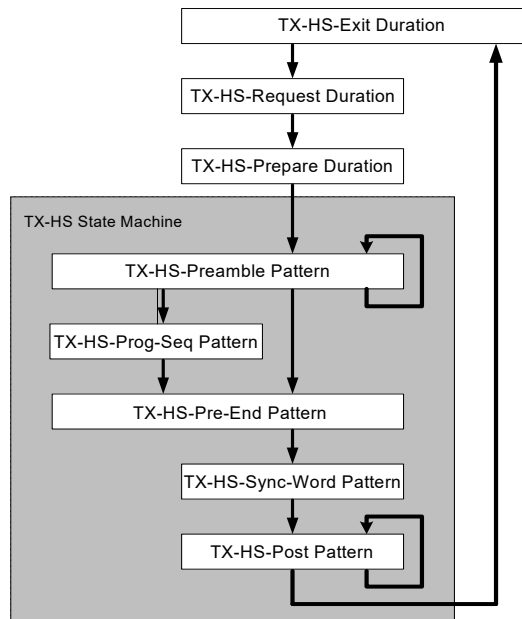


Figure 3 TX State for High-Speed Data Transmission

The following table describes the high-speed data transmission:

Table 1 High-Speed Data Transmission Description

State	Line State	Exit State	Exit Condition
TX-HS-Exit Duration	Transmit LP-111	TX-HS-Request Duration	On request of Protocol for High-Speed Transmission
TX-HS-Request Duration	Transmit LP-001	TX-HS-Prepare Duration	End of timed interval t_{LPX}
TX-HS-Prepare Duration	Transmit LP-000	TX-HS-Preamble Pattern	End of interval $t_{3-PREPARE}$
TX-HS-Preamble Pattern	Preamble 3,3,3,3...	TX-HS-Prog-Seq Pattern	End of Preamble & Prog-Seq selected
		TX-HS-Pre-End Pattern	End of Preamble & Prog-Seq not selected
		TX-HS-Preamble Pattern	Preamble words remaining count > 0
TX-HS-Prog-Seq Pattern	Prog-Seq	TX-HS-Pre-End Pattern	End of Prog-Seq
TX-HS-Pre-End Pattern	Pre-End	TX-HS-Sync-Word Pattern	End of Pre-End
TX-HS-Sync-Word Pattern	Sync Word	TX-HS-Post Pattern	End of Sync-Word
TX-HS-Post Pattern	Post 4,4,4,4...	TX-HS-Exit Duration	Last word of Post sent

16-Bit to-7-Symbol Mapping

The Mapper is the outer-most function in the MIPI C-PHY digital coding system that occurs on the transmit side. It converts a 16-bit word into a group of seven symbols at the transmitting end. The 16-bit to-7-symbol Mapper perform a mapping function between 16-bit input/output values and a group of 7 symbols which is comprised of seven 3-bit symbol values. Each symbol is comprised of a flip, rotate and polarity bit, so for any particular symbol, n , $sn = [Flip[n], Rotation[n], Polarity[n]]$. A seven-symbol Mapper output value is defined for every possible 16-bit Mapper input value. Since the mapping is completely feed-forward function, pipeline registers can be inserted between intermediate stages if necessary to lessen timing constraints in systems that operate at a high symbol rate.

The following table shows 16-bit to 7-symbol mapper:

Table 2 16-Bit to 7-Symbol Mapper

Pattern File Entry	Description
% (Percentage)	Starts or stops binary to symbol mapping. Number of enclosed bits must a multiple of 16. Once the first % is found the subsequent symbols may only contain the binary states 0 and 1 interpreted MSB (Most Significant Bit) first. The % may only occur in pairs. It can be placed freely inside the pattern definition and its output will always be regarded as high speed mode symbols.
0	Binary 0 (Low)
1	Binary 1 (High)

Transmit Lane PRBS Register Operation

The PRBS generator generates bits. These bits are then put into the 16-Bit to 7-Symbol Mapper resulting in transitions which will be played back indefinitely after having been converted to wire states.

The Transmit Lane PRBS Register Q[16:1] is the source of data input to TxD[15:0] of the Mapper when the lane master is transmitting one of the three PRBS patterns as defined by the TLRn_Test_Patterns_Select register. The Transmit Lane PRBS register is initialized using the seed values TLRn_PRBS_Seed_0, TLRn_PRBS_Seed_1 and TLRn_PRBS_Seed_2. The first word transmitted from the PRBS generator is equal to the seed value: [TLRn_PRBS_Seed_1[7:0], TLRn_PRBS_Seed_0[7:0]]. This initial 16-bit value from the PRBS register is transmitted immediately following transmission of the first Sync Word after the low-power to high-speed mode transition. The Transmit Lane PRBS Register is shifted 16 bit positions after each 16-bit word is output to minimize correlation from one data value to the next. This way no bits are re-used in successive samples.

The important part for PRBS generation is that for degree 9 and 11 the seed value register is 16 bits wide. The seed defines the whole 16 bits and the first value shifted out of the PRBS is the value at bit 16. This means that the first 7 bits for degree 9 will only occur once and will not be looped. For PRBS 11 this will be true for the first 5 bits. PRBS for degree 18 is different. The Seed register is 18 bits wide and all bits will be looped. But Bit position 16 is still where the values from the PRBS will be captured.

For example: Seed values and data sequences for the chosen PRBS mode are as follows:

- PRBS9 – Seed = 0x789a; TLRn_PRBS_Seed_0[7:0] = 0x9a; TLRn_PRBS_Seed_1[7:0] = 0x78; TLRn_PRBS_Seed_2[7:0] value does not matter; Transmit data sequence: 0x789a, 0x9980, 0xc651, 0xa5fd, 0x163a, 0xcb3c, 0x7dd0...
- PRBS11 – Seed = 0x789a; TLRn_PRBS_Seed_0[7:0] = 0x9a; TLRn_PRBS_Seed_1[7:0] = 0x78; TLRn_PRBS_Seed_2[7:0] value does not matter; Transmit data sequence: 0x789a, 0x5e64, 0xfe0, 0xac43, 0xa9a1, 0xe4ce, 0xfea0...
- PRBS18 – Seed = 0x2789a; TLRn_PRBS_Seed_0[7:0] = 0x9a; TLRn_PRBS_Seed_1[7:0] = 0x78; TLRn_PRBS_Seed_2[7:0] = 0x02; Transmit data sequence: 0x789a, 0x8d77, 0x0dbc, 0x74e1, 0x8108, 0x414a, 0x3915...

For details, visit <http://mipi.org/>.

ISI Generation – S6P Support

Scattering Parameter are used to describe the electrical characteristics of a linear system.

So when a signal is transmitted at some given frequency through a port (in circuit), some part of it is reflected and rest is transmitted. Transmitted signal reaches the receiver at different time, with different magnitudes and phase and mingles with existing symbols, which means part or all of a given signals will be spread into smaller signals, thereby interfering with the correct detection of the signal, which is one of the causes of ISI.

In order to define characteristics of such system Scattering matrix or S-Parameter is used in the form of SnP files. Where n is the number of ports. 2 ports (S2P) file contains four pairs of magnitude and amplitude values at given frequency.

By applying these set of magnitude and phase values at given frequency, ISI is emulated on respective lines.

In addition to S2P support, now S6P files are also supported to emulate ISI on all 3 lines.

S6P file are transformed to 2 port parameters (magnitude and phase) which are applied to line A, B and C.

Start Pattern and Triggered Start

The purpose of the triggered startup is to enable the MIPI C-PHY plug-in to provide a static LP STOP signal.

MIPI C-PHY is represented as an LP-111 state on the data lane(s). Triggered start mode can be also selected as LP-000 state from Trigger Start property.

This triggered startup is only active when setting the parameter “AWG Setup. Startup Mode” to “Triggered”.

The parameter can only be changed while the plug-in is stopped. When this feature is selected and signal generation is started the plug-in will calculate the complete waveform and downloads it into the AWG as it would occur for the “Immediate” startup mode. The difference is then that once calculation and download is finished the AWG outputs will drive a static voltage at the LP level with is selected for the respective wire/lane.

To advance from this idle state either the GUI trigger button can be pressed, or the respective SCPI command can be called, or the TRIG IN of the AWG module can be called. After doing this the run mode is active and parameters can be reprogrammed on the fly.

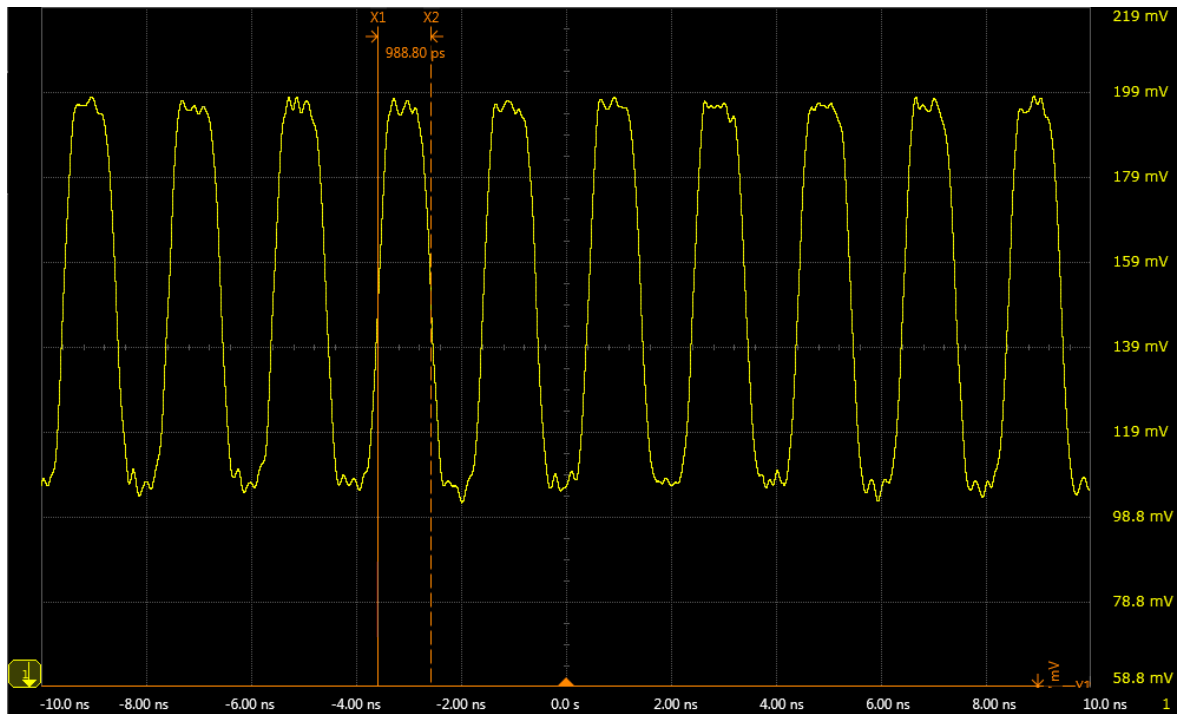
De-skew without Re-cabling

For the M8195A, the clock sync module is used to align the AWG modules but the de-skew of the cables is not done automatically.

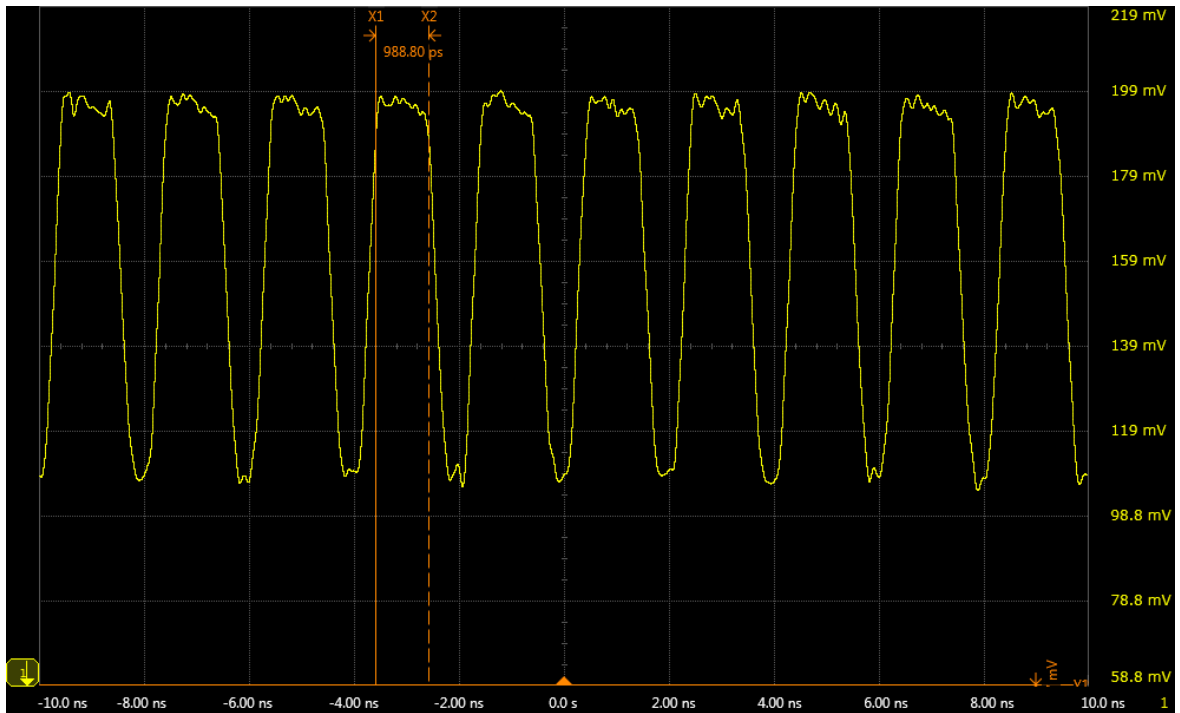
Duty Cycle Distortion

Duty cycle distortion is a timing impairment that falls under deterministic jitter category. It is used to achieve eye closure by varying the transition time of rising and falling edge. The time domain behavior of DCD is that all rising edges are delayed or advanced by the same amount. The same behavior holds good for falling edges, which are delayed or advanced by the same amount. There are two scenarios, one in which rising edge is advanced and falling edge is delayed and second in which rising edge is delayed and falling edge is advanced. By delaying or advancing transition time of edges in this manner, duration of levels are either increased or decreased, which results in eye closure. Both edges cannot be advanced or delayed at the same time, otherwise DCD will not have desired behavior.

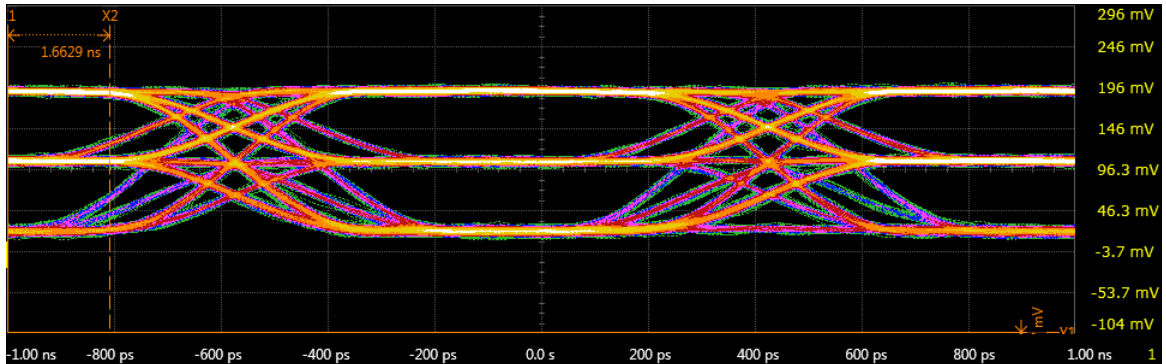
The following figure shows an example when no DCD is applied on high speed symbols:



The following figure shows an example when DCD of 300 mUI applied on high speed symbols:

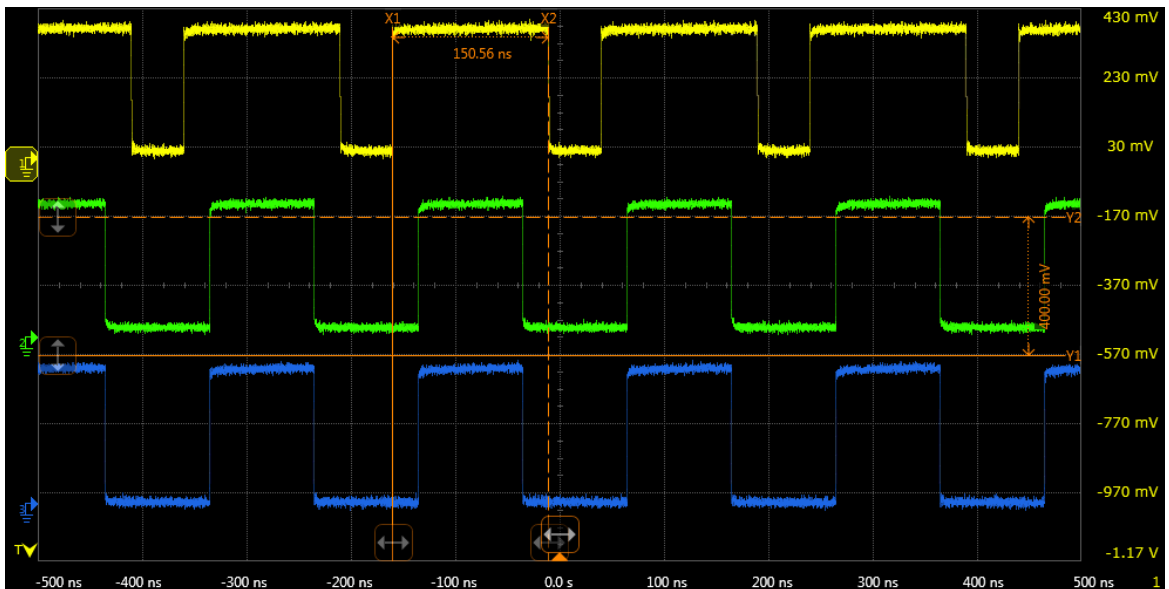


The following figure shows an example of eye when DCD of 300 mUI applied on high speed symbols:



Same holds good for LP Pulse width, except it works for Low Power symbols only. But in case of LP pulse width instead of saying by how much amount edges are shifted or delayed, we can control the width of pulse with respect to high level.

The following figure shows an example of LP pulse of 150 ns width on line A.



Different values of DCD and Pulse width applied to channels, results in skew among channels due to shift in transition times. At the boundary of LP and HS symbols DCD or pulse width doesn't apply. As both DCD and pulse width requires two consecutive high speed and low power rising and falling edges.

Skew Calibration Values

To access skew calibration values, read-only fields in GUI under calibration functional block is provided. This read-only value is the sum of measured delay (skew) between AWG channels and delay at which marker bit is set in delay segment to trigger second AWG. Skew calibrated values are provided per channel and can be accessed via SCPI too.

These properties will indicate how much skew between the 3 different AWG channels was measured during calibration. It shifts the pattern in way that all three outputs will be synchronous at the receiver.

A pre-requisite to use calibration on the complement and normal pins is to have matched cable pairs, which connects scope and DUT.

Hysteresis

The Hysteresis is incorporated into Low Power receivers to reduce sensitivity to noise, and prevents Logic state changes due to short-term, low amplitude excursions below the VIH Logic-1 threshold (or above VIL Logic-0 threshold), after the instantaneous voltage has initially crossed the threshold for any given bit interval.

In order to test, if DUT can successfully receive Low Power test sequence after applying a sin wave signal of desired peak voltage (Hysteresis voltage which is half of peak to peak amplitude) on signal, which results in additive noise signal on line A, B or C. Period of applied sin wave signal is 2.3 times the TLPX value of the applied test sequence. Applied signal will modulate the phase of the additive noise with respect to the LP data test sequence, for added impact. The amplitude of the additive noise should be calibrated to produce an approximately 25 mVpk (50 mVpp) deviation from the nominal LP-0/1 levels of the Test System (which will be set to the measured VIH and VIL values for the DUT). When the additive noise is enabled, the DUT should still be able to successfully receive the LP test sequence without error, if it employs sufficient hysteresis on its LP-RX.

Sinusoidal Noise is applicable to Low Power symbols only. High Speed symbols are unaffected by this noise.

Examples of hysteresis

NOTE

All the images (Figure 4 to Figure 7) of hysteresis show the single ended signal of wire A (channel 1 shown by yellow), wire B (channel 2 shown by green), and wire C (channel 3 shown by blue).

- 1 Figure 4 shows the common mode interference with high speed switching rate of 1 Gbps and the frequency 4 GHz.

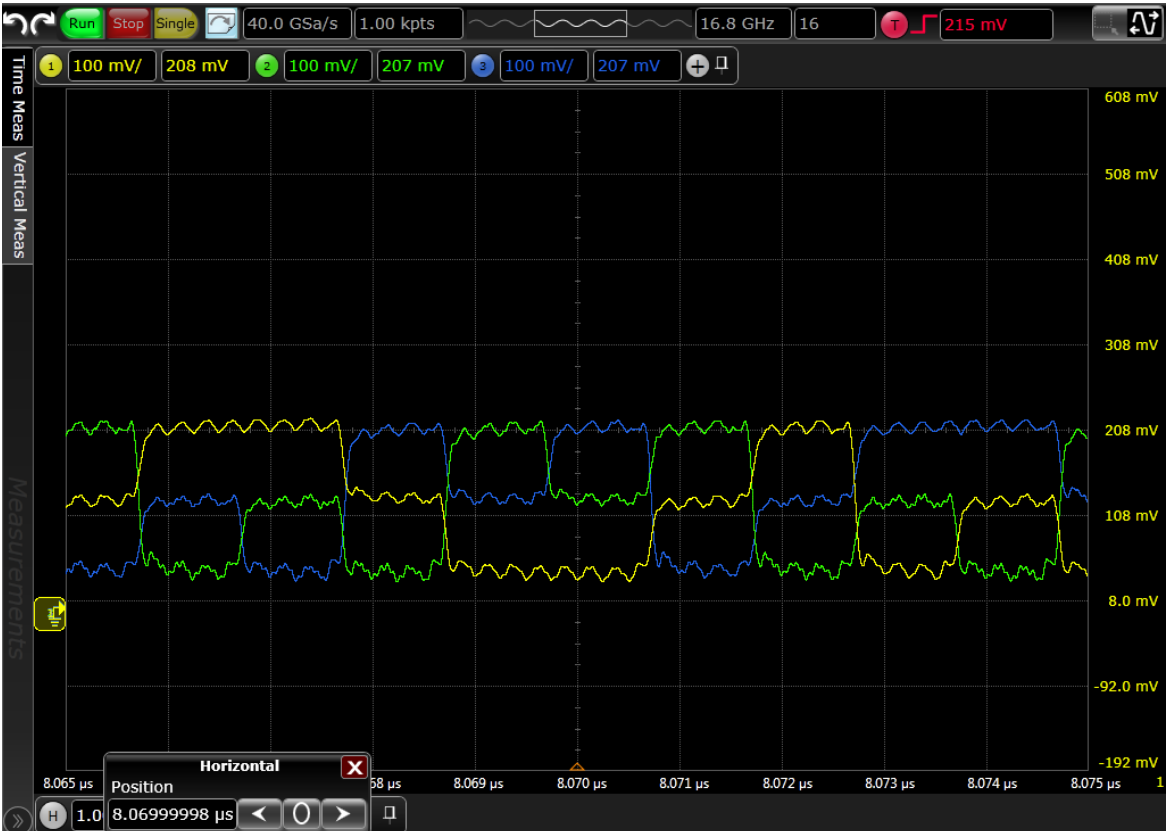


Figure 4 Common mode interference on high speed

- 2 Figure 5 shows the common mode interference with same high speed switching rate and interference frequency as shown in above example but with double interference amplitude.

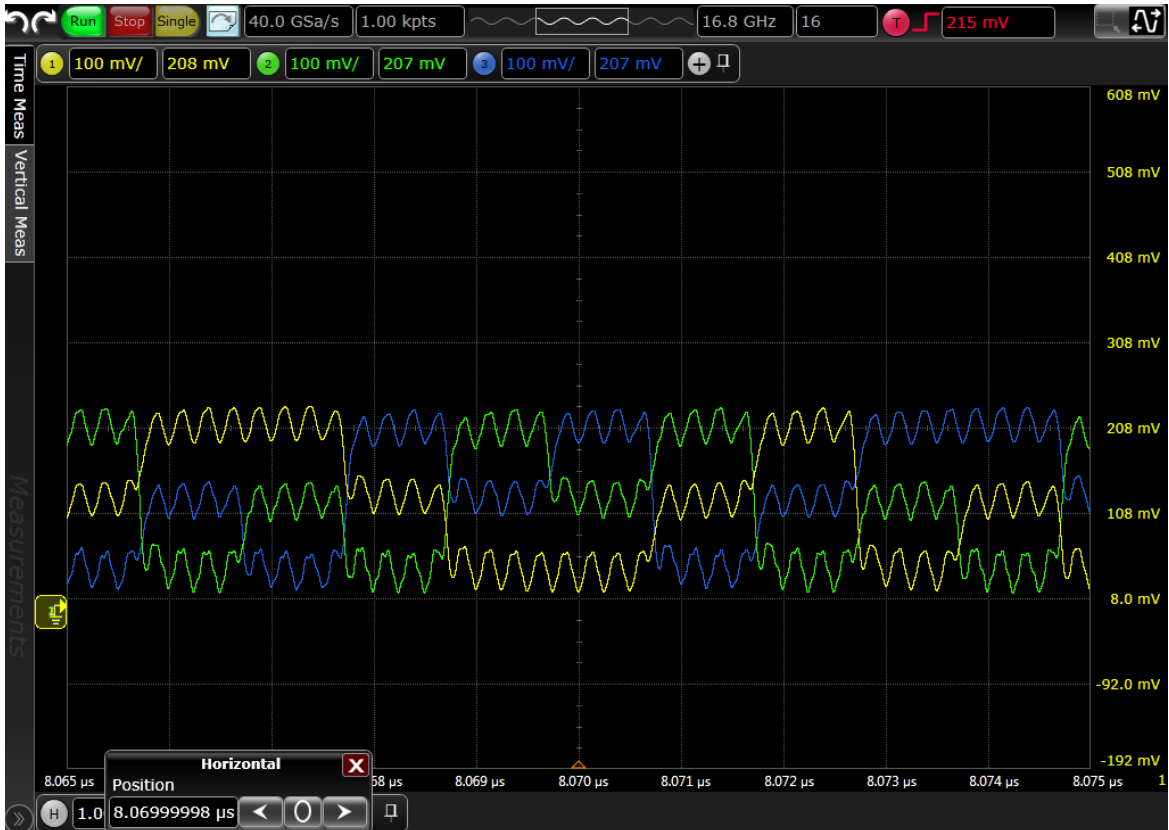


Figure 5 Common mode interference on high speed

- 3 Figure 6 shows the Low Power interference with a frequency of 50 MHz.

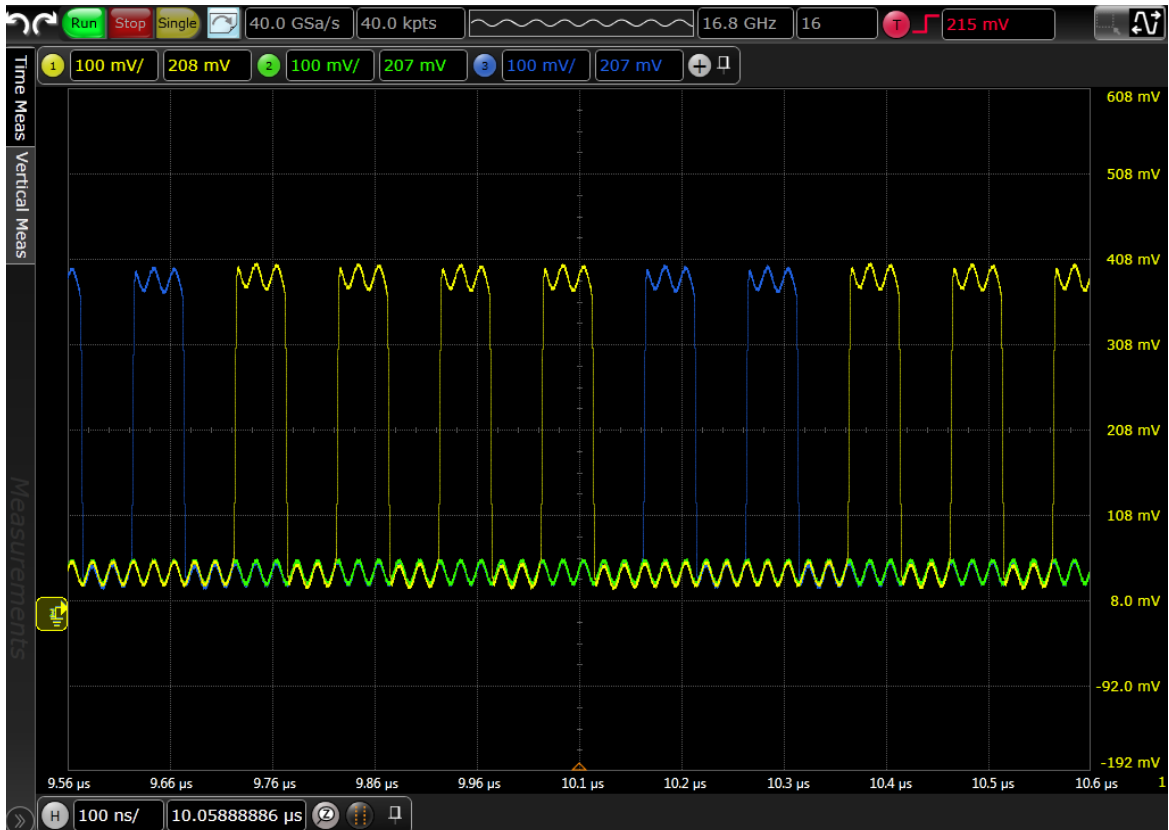


Figure 6 Low Power interference

- 4 Figure 7 shows the (advanced) emphasis of MIPI C-PHY. The MIPI C-PHY always requires at least two wires to switch the level and sometimes, the third wire may stay at the same level from one state to the other. So MIPI C-PHY provides a view of non transition states only. Following are the observations of figure:
- the markers are set to the non transition high (shown by straight line in orange) and mid levels (shown by broken line in orange).
 - the transition states are showing an higher amplitude then the non-transition states.

- the green wire stays at the beginning at high and the second state is lower then the first state, and then it goes to mid level followed by two states at low level.
- The deviation for state switches from low to high and high to low are deviating by a factor of two relatively to switching states from mid to high, high to mid, mid to low and low to mid. Currently, the pre-emphasis scheme pertaining to CTS v1.1 specification is not supported.



Figure 7 Pre-emphasis of MIPI C-PHY Editor

Multi-lane Structure

Multi-lane structure means working with more than one module.

Multi-lane Support:

M8085A plug-in supports up to 3 lanes. For every increased lane, you require an additional M8195A AWG. This means that for two and three lane structure, you require two and three M8195A AWG modules, respectively. These can be called as M8195A Lane 1, M8195A Lane 2 and M8195A Lane 3.

A multi-channel synchronization module (M8197A) is required to make a proper synchronization and to establish master slave relation among the AWG(s).

2 Sequence and Data Files

Patterns, Data Format and Sequences / 28

CSI and DSI Sequences / 38

Sequence File Definition for CSI / 41

Sequence File Definition for DSI / 50

Patterns, Data Format and Sequences

The MIPI C-PHY Editor supports four different mode groups “Pattern”, “Burst”, “Pure HS” and “Frames”, which are as following (see [Figure 8](#)):

The screenshot shows the 'Data Pattern' configuration window. At the top, 'Run Mode' is set to 'Interrupted' and 'Triggers Start' is set to 'LP000'. 'Idle Voltage' is '300 mV' and 'LSB First' is checked. Under 'Mode', the 'Burst' radio button is selected. The 'HS Data' and 'LP Data' fields both contain the hexadecimal sequence '0xE7, 0xE7, 0xE7, 0xE7', each with a 'Browse...' button to its right.

Figure 8 Different Modes of MIPI C-PHY Editor

- **Pattern:** A *.ptrn file contains the definition of LP and/or HS C-PHY states. The pattern files are loaded with the extension of '*.ptrn' and write in terms of line states.
- **Burst:** It is a block of binary data, which are converted to MIPI C-PHY states, and is repeated infinitely. The burst block may contain either LP Data and HS Data both, pure LP data, or pure HS data, depending on the content of the data given for HS and LP. If either field of HS or LP data is blank (empty text box for that particular data type, see [Figure 8](#)), the software generates pure LP or HS data respectively. However, even if LP data entry is empty, the HS data are organized in a burst, surrounded by LP111 states, and a LP to HS transition, and a HS to LP transition. If Pure HS is selected, no LP transitions are included and all LP data are neglected. The files are loaded with the extension of '*.dat'
- **Pure HS:** The Pure HS mode contains only HS data and no LP111 transitions are included. For pure HS mode all the LP data is neglected.
- **Frames:** It is the most flexible mode. The frame mode allows to organize the data in blocks, and a sequence of blocks containing loops. The definition of a sequence is done via a sequence file and it can be loaded with the extension '*.seq'. Each individual block can be repeated ("looped") N-times and the number of repetitions N can be selected for each block separately. In addition to the sequence file, the frame mode may require one or more data files.

*.ptrn File Format (P Macro)

The following table shows the line states available in **High-Speed Mode**:

Table 3 Line States

Pattern File Entry	Line State	Line State {Line A, Line B, Line C}
X	+x	{1, 0, ½ }
x	-x	{0, 1, ½ }
Y	+y	{½ , 1, 0}
y	-y	{½, 0, 1}
Z	+z	{0, ½, 1}
z	-z	{1, ½, 0}

The following table shows the symbols (transitions) available in **High-Speed Mode**:

Table 4 Symbol (Transitions))

Pattern File Entry	Symbol Input Value	Activity
0	000	Rotate CCW, polarity stays same
1	001	Rotate CCW, polarity is inverted
2	010	Rotate CW, polarity stays same
3	011	Rotate CW, polarity is inverted
4	1xx	Same phase, polarity is inverted

The following table shows the line states provided by **Low Power Mode**:

Table 5 Lines states provided by Low Power Mode

Pattern File Entry	Line A	Line B	Line C	Activity
L	0	0	0	One "L" sets all three wires to low state

Pattern File Entry	Line A	Line B	Line C	Activity
C	0	0	1	One pattern file entry defines the state of all 3 wires (1 Symbol). An upper case letter means only the selected wire is high, the 2 other wires are low. A lower case letter means only the selected wire is low, the 2 other wires are high.
B	0	1	0	
a	0	1	1	
A	1	0	0	
b	1	0	1	
c	1	1	0	
H	1	1	1	One "H" sets all three wires to high state

Pattern Coding Examples

The following table shows the example of different types of patten coding used:

Table 6 Pattern Coding Examples

Pattern Coding	Line States	Description
X0123 (XZyzX)	210432113 Resulting Line States: X -> YxzZxyXzX	High speed init pattern and high speed loop pattern. The high speed signal can be coded with transition symbols too.
X0123 (XZyzX)	%0101000011101101% Resulting Transitions: X -> 1432300 Resulting Line States: zZxyZYX	High speed init pattern and coded binary loop pattern. Between the percentage symbols (%) binary coding can be used. The last wire state of the init pattern needs to be equal to the last wire state of the loop pattern (X). If this is not the case coding of the looped pattern will be erroneous.
xXyx	X0241%1011011101001010%x Resulting Line States: X ZXxZ yYzxYZY x	High speed init pattern and high speed loop pattern mixed with binary coding. You can mix the use of wire states, transition symbols and binary coding. This is mainly useful for introducing coding errors.
HLB	xYzxZyyZy	Low power init pattern and high speed loop pattern. There will be a low power to high speed transition at the end of the init pattern.
HLBZ	xYzxZyyZ	Mixed low power/high speed init pattern and high speed loop pattern. The transition from low power to high speed will happen within the init pattern.

Pattern Coding		Line States		Description	
acL		y -> 0120412CLAbc		Low power init pattern and high speed and low power loop pattern. There will be a high speed low power transition within the loop pattern and a low power high speed transition at the end of the loop pattern. Limitation: The "LP->HS Start Wire State" cannot be modified.	
	Expanded Pattern	Wire States		Expanded Pattern	Wire States
Init Pattern	acL	acL	Loop Pattern 1	0120412	xZXzYZ
LP->HS Start	H	H	HS->LP Post	4444444	zZzZzZ
LP->HS Request	C	C	HS->LP Exit	C	C
LP->HS Prepare	L	L	Loop Pattern 2	CLAbc	CLAbc
LP->HS Start Wire State	X	X	LP->HS Start	H	H
LP->HS Preamble	3333333	yZxYzXy	LP->HS Request	C	C
LP->HS Pre-End	3333333	ZxYzXyZ	LP->HS Prepare	L	L
LP->HS Sync	3444443	xXxXxXy	LP->HS Start Wire State	X	X
			LP->HS Preamble	3333333	yZxYzXy
			LP->HS Pre-End	3333333	ZxYzXyZ
			LP->HS Sync	3444443	xXxXxXy

***.dat or *.txt File Format (B/LPB Macro)**

For data file the hexadecimal (HEX) format is required. Bytes are represented in two digits, ranging from 0 to 9 and A to F. The leading string "0x" is optional. Supported separators between data bytes are

, (comma)

; (semicolon)

space (blank)

tab

line feed

nothing

Some examples:

- 0x01, 0xF3, 0x23
- 0134E734FF
- 32 FF E5 44

In addition to the pure HEX data, special commands are abbreviations of lists of hex bytes:

- 0x<HEX code> N <count>: repeat the byte <HEX code> N times

Example: 0xABN5 is equal to AB AB AB AB AB

- 0x<HEX code 1>x<HEX code 2>: count up/down from <HEX code 1> to <HEX code 2>

Example: 0x05x0A is equal to 05 06 07 08 09 0A

- 0x<HEX code 1>c<HEX code 2>: count up/down from <HEX code 1> to <HEX code 2> for each data lane separately. If there is only one data lane, this command is equal to the one before. However, in the case of multiple data lanes, the values are counted with a step size of one for each data lane separately.

Example for 2 data lanes:

0x02c05 is equal to D0: 02 03 04 05 and D1: 02 03 04 05.

For the counter with the "x" and two data lanes, 0x02x05 would lead to D0: 02 04 and D1: 03 05.

NOTE

The special commands require the leading "0x", otherwise they will not be recognized.

***.seq File Format**

In the sequence file the data rate, data blocks and sequence are defined. Note that all parameters are even integers. The structure of a sequence file is shown in [Figure 9](#).

HSFreq: <frequency in bits/s>

Blocks:

<BlockName 1>: <Block Definition 1>, ..., <Block Definition n1>;

...

<BlockName M>: <Block Definition 1>, ..., <Block Definition nM>;

Sequence:

1. <BlockName J>, <Loop Count R>; - First block

...

<N>. <BlockName K>, <Loop Count S>; - Nth block

...

<P>. <BlockName L>, <Loop Count T>; - Pth block

[LoopTo N]

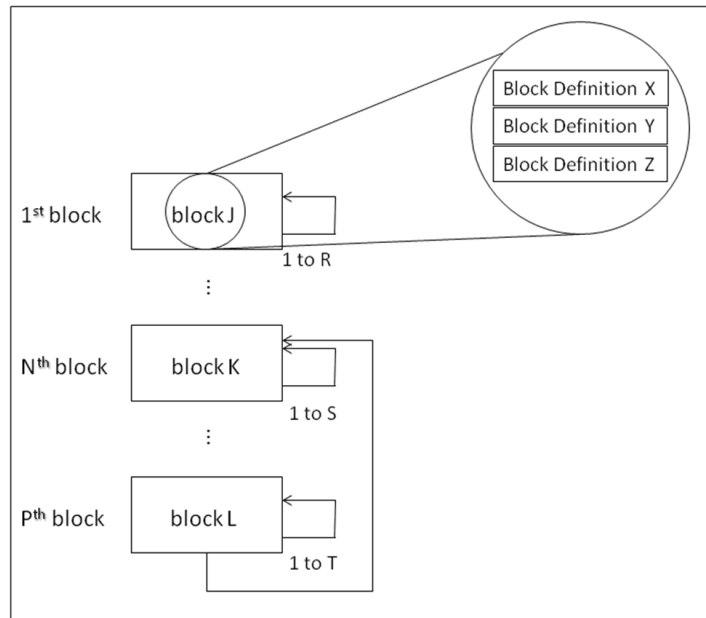


Figure 9 Block Diagram of the structure of sequence file

Each block may comprise multiple sub-blocks (1 to n). Sub-blocks can be used in multiple blocks. In the sequence, blocks can be used as often as needed. Within the sequence, the “LoopTo” expression starts an infinite loop from block <N> to the last block <P>. If no “LoopTo” expression is specified, an infinite loop is created from block 1 to block P (last block). The valid block definitions (macros) may contain the following data:

- LP00, LP01, LP10, LP11, LP000,...,LP111: for a single LP state.
- LPB “<filename>”: for generating LP data specified in the file with the name <filename>. The file should be in the same folder as the sequence file. At the beginning of the data an escape trigger for LP or ULP data mode is sent before the data and a Mark-0/1 sequence is sent after the end of the data. For the data, the data file format given above must be used.
- LP<00, 01, 10, or 11, 000, ..., 111> N <number of bits>: The LP state is sent <number of [HS] bits> times.
- LP<00, 01, 10, or 11 000, ..., 111> E <number of bits>: The LP state is sent until the block size reaches the number of HS bits given in <number of bits>.

- LPHSE <number of bits>: The block is filled with LP111 states and a LP-HS transition until the number of HS bits <number of bits> is reached.
- LPHS: The macro adds a LP-HS transition. It is mainly used for influencing the block ending. For example, if the following block starts with HS data, then the LP-HS transition will be done at the end of the actual block. Without this macro the LP-HS transition would be added to the beginning of the following block. You do not need the macro for the following B macros, since they trigger a LP-HS transition automatically if the previous block description contains the LP states.
- B"<filename>": for generating HS data given in the file <filename>. The file should be in the same folder as the sequence file. If necessary, a LP-to-HS transition is generated before the data. A HS-to-LP transition is added if the following block contains LP states.
- BL <number of blanking bytes>: for generating HS blanking packets with a number of blanking bytes given in <number of blanking bytes>. In this case the header for DSI is different to CSI, and also the checksum. Please refer to CSI and DSI specification for long packets.
- P"<filename>": for backward compatibility a PTRN file can be used (see pattern definition for PTRN files).
- C<3 hex bytes>: For generating short packets like they are described in the CSI or DSI specification for MIPI C-PHY.
- C<1 hex byte>"<filename>": for generating a long packet. The content in the filename will be taken as payload. The header will have the given hex byte as ID followed by a two byte word counter, followed by a ECC for that header data. At the end of the payload a two byte CRC is added. If necessary, a LP-to-HS transition is generated before the data. A HS-to-LP transition is added if the following block contains LP states.
- PRBS<no.>(<seed1>|<seed2>| ... | <seedN>): for generating a PRBS of the polynomial <no.> with a seed of <seed1-N> for each lane. The <no.> is just a decimal number (only 9, 11, and 18 are allowed), and the seeds are given in a hex number (example 0x789A). The number of input seeds should be the same as active lanes (example for 3 lanes <seed1> goes to D0, <seed2> goes to D1 and <seed3> goes to D2). If #seeds > #lanes the latest seeds will be ignored. If #seeds < #lanes an exception will be thrown. As a special case, if only one seed is provided but more than one lane is active, the pattern is distributed among all lanes.

- **ULPEntry**: Adds the ULP Entry escape sequence to the block. After the ULPEntry LP000 states plus finally a ULPExit should follow to create a specification conform ULP sequence.
- **ULPExit <number of LP100 states>**: Creates a ULP exit sequence. It is not allowed to combine this block definitions with other definitions, which means in this case the block must only contain this macro and no other.

NOTE

- If blocks are looped, then the beginning of the block should have the same kind of data mode (LP or HS) as the block following it, otherwise the block loop will result in invalid LP-to-HS transitions.
- Video Frames that contain LP111 blanking periods should be rotated so that the block definition always ends with a LP111E command.
- If only the header contains LP111 states, the header block should end with LPHSE to start the HS transmission at the end of the header block.
- In case of PureHS mode, an initial LP to HS transition is added in the form of a hidden intro block in the waveform generation, before an infinite loop of pure HS data stream is generated.

In any case, an LP111 block is added to each sequence. A LP–HS transition is added if needed to switch the device into HS mode. These blocks need not to be added explicitly to the sequence. They are added automatically for all sequences, i.e. even if a sequence with pure HS blocks is given.

Example of a Sequence File:

HSFreq: 200MBit/s;

Blocks:

LPInit1: LPB"Esc0ms.txt",LP111E13728;

LPPause: LP11N1024;

LPInit2: LPB"Esc100ms.txt",LP111E13728;

LPInit3: LPB"Esc200ms.txt",LP111E13728;

Header: B"FirstHsLine.txt",LP111E6016;

Video: B"VideoLine.txt",LP111E6016;

Sequence:

```

1  LPIInit1,1;
2  LPPause,20000;
3  LPIInit2,1;
4  LPPause,20000;
5  LPIInit3,1;
6  LPPause,20000;
7  Header,1;
8  Video,319;

LoopTo 6;

```

NOTE

Some Examples of DSI sequence, CSI sequences and data blocks are provided with MIPI C-PHY Editor. To access them use the following directory path:

C:\ProgramData\BitifEye\C-PHY Editor\Pattern

The following sections describe the various elements of a sequence file for the CSI and DSI protocols, such that the sequence file definition generates a waveform that conforms to the MIPI Specification for Camera Serial Interface (CSI) and MIPI Alliance Specification for Display Serial Interface (DSI), respectively.

CSI and DSI Sequences

Generally, a sequence file consists of three elements that form together a sequence:

- HS Data Rate
- Blocks
- Sequence

Following is a real time example of a sequence file definition:

Example of a Sequence File Definition:

```
HSFreq: 200MBit/s;
Blocks:
LPInit1: LPB"Esc0ms.txt",LP11E13728;
LPPause: LP11N1024;
LPInit2: LPB"Esc100ms.txt",LP11E13728;
LPInit3: LPB"Esc200ms.txt",LP11E13728;
Header: B"FirstHsLine.txt",LP11E6016;
Video: B"VideoLine.txt",LP11E6016;
Sequence:
1  LPInit1,1;
2  LPPause,20000;
3  LPInit2,1;
4  LPPause,20000;
5  LPInit3,1;
6  LPPause,20000;
7  Header,1;
8  Video,319;
LoopTo 6;
```

Following are real time examples of DSI and CSI sequence file definitions, respectively:

Example of a DSI Sequence File Definition:

```
HSFreq: 432.432MBit/s;
Blocks:
LPInit: LPB"HSyncEnd.txt",LP11E13728;
HSync: B"HSyncEnd.txt",LP11E12736,B"HSyncStart.txt",LP11E13728;
VSyncStart: B"HSyncEnd.txt",LP11E12736,B"VSyncStart.txt",LP11E13728;
VSyncEnd: B"HSyncEnd.txt",LP11E12736,B"VSyncEnd.txt",LP11E13728;
Video: B"HSyncEnd.txt",LP11E960,B"Video480pHSyncStart.txt",LP11E13728;
```

Sequence:

```
1 LPInit,1;
2 HSync,5;
3 VSyncEnd,1;
4 HSync,29;
5 Video,480;
6 HSync,9;
7 VSyncStart,1;
LoopTo 2;
```

Example of a CSI Sequence File Definition:

```
HSFreq: 158 MBit/s;

Blocks:
FrameStart: B"FrameStart.txt",LP11E10880;
Blanking: LP11E10880;
Video: C1E"compliance640_480.txt",LP11E10880;
FrameEnd: B"FrameEnd.txt",LP11E2048;
```

Sequence:

```
1  FrameStart,1;  
2  Blanking,1;  
3  Video,480;  
4  Blanking,2;  
5  FrameEnd,1;
```

Sequence File Definition for CSI

Overview

CSI is a MIPI Alliance standard for serial interface between a camera module and host processor. CSI adheres to the Low-Level Protocol (LLP), which is a byte orientated, packet based protocol that supports the transport of arbitrary data using Short and Long packet formats. Two packet structures are defined for low-level protocol communication: Long packets and Short packets. The format and length of Short and Long Packets depends on the choice of physical layer (MIPI C-PHY or MIPI D-PHY). For each packet structure, exit from the low power state followed by the Start of Transmission (SoT) sequence indicates the start of the packet. The End of Transmission (EoT) sequence followed by the low power state indicates the end of the packet. However, in CSI implementation, one burst consists of only one packet and LP11 state must be inserted before the start of a burst. Since it requires to go to LP state always, an explicit EoT packet is not required.

NOTE

A sequence file used for D-PHY or C-PHY conformance testing cannot be used for CSI/DSI conformance testing unless the header, payload and checksum data is included in the CSI/DSI block definitions in the sequences else the device rejects the packet.

Long and Short Packet Formats

Long Packet

For D-PHY, a Long Packet shall be identified by Data Types 0x10 to 0x37. A Long Packet for the D-PHY physical layer option shall consist of three elements: a 32-bit Packet Header (PH), an application specific Data Payload with a variable number of 8-bit data words, and a 16-bit Packet Footer (PF). The Packet Header is further composed of three elements: an 8-bit Data Identifier, a 16-bit Word Count field and an 8-bit ECC. The Packet footer has one element, a 16-bit checksum (CRC).

For MIPI C-PHY, the Long Packet structure for the C-PHY physical layer option shall consist of four elements: a Packet Header (PH), an application specific Data Payload with a variable number of 8-bit data words, a 16-bit Packet Footer (PF), and zero or more Filler bytes (FILLER). The Packet Header is 6N x 16-bits long, where N is the number of C-PHY physical layer Lanes. The Packet Header consists of two identical 6N-byte halves, where each half consists of N sequential copies of each of the following

fields: a 16-bit field containing eight Reserved bits plus the 8-bit Data Identifier (DI); the 16-bit Packet Data Word Count (WC); and a 16-bit Packet Header checksum (PH-CRC) which is computed over the previous four bytes. The value of each Reserved bit shall be zero. The Packet Footer consists of a 16-bit checksum (CRC) computed over the Packet Data using the same CRC polynomial as the Packet Header CRC and the Packet Footer used in the D-PHY physical layer option. Packet Filler bytes are inserted after the Packet Footer, if needed, to ensure that the Packet Footer ends on a 16-bit word boundary and that each C-PHY physical layer Lane transports the same number of 16-bit words (i.e. byte pairs).

For both physical layer options, the 8-bit Data Identifier field and the 16-bit Word Count (WC) field contain identical data. The CSI receiver reads the next WC 8-bit data words of the Data Payload following the Packet Header. The length of the Data Payload shall always be a multiple of 8-bit data words. For both physical layer options, once the CSI receiver has read the Data Payload, it then reads the 16-bit checksum (CRC) in the Packet Footer and compares it against its own calculated checksum to determine if any Data Payload errors have occurred.

In either case, Packet Data length = Word Count (WC) * Data Word Width (8-bits).

Short Packet

For each option (MIPI C-PHY and MIPI D-PHY), the Short Packet structure matches the Packet Header of the corresponding Low Level Protocol Long Packet structure with the exception that the Packet Header Word Count (WC) field shall be replaced by the Short Packet Data Field. A Short Packet shall be identified by Data Types 0x00 to 0x0F. A Short Packet shall contain only a Packet Header; neither Packet Footer nor Packet Filler bytes shall be present. For Frame Synchronization Data Types, the Short Packet Data Field shall be the frame number. For Line Synchronization Data Types, the Short Packet Data Field shall be the line number.

For the D-PHY physical layer option, the Error Correction Code (ECC) byte allows single-bit errors to be corrected and 2-bit errors to be detected in the Short Packet.

For the MIPI C-PHY physical layer option, the 16-bit Checksum (CRC) allows one or more bit errors to be detected in the Short Packet but does not support error correction.

Short Packet Data Types shall be transmitted using only the Short Packet format. Refer to Table 6 Synchronization Short Packet Data Type Codes of the MIPI Alliance Specification for Camera Serial Interface (CSI), which indicates that Data Type for Frame Start Code is 0x00 and Data Type for Frame End Code is 0x01.

NOTE

Between Low Level Protocol packets, there must always be an HS-LP or an LP-HS transition.

Frame and Line Synchronization Packets

Frame Synchronization Packets

Each image frame shall begin with a Frame Start (FS) Packet containing the Frame Start Code. The FS Packet shall be followed by one or more long packets containing image data and zero or more short packets containing synchronization codes. Each image frame shall end with a Frame End (FE) Packet containing the Frame End Code. For FS and FE synchronization packets, the Short Packet Data Field shall contain a 16-bit frame number. This frame number shall be the same for the FS and FE synchronization packets corresponding to a given frame.

Line Synchronization Packets

Line synchronization packets are optional. For Line Start (LS) and Line End (LE) synchronization packets, the Short Packet Data Field shall contain a 16-bit line number. This line number shall be the same for the LS and LE packets corresponding to a given line.

Frame Blanking and Line Blanking

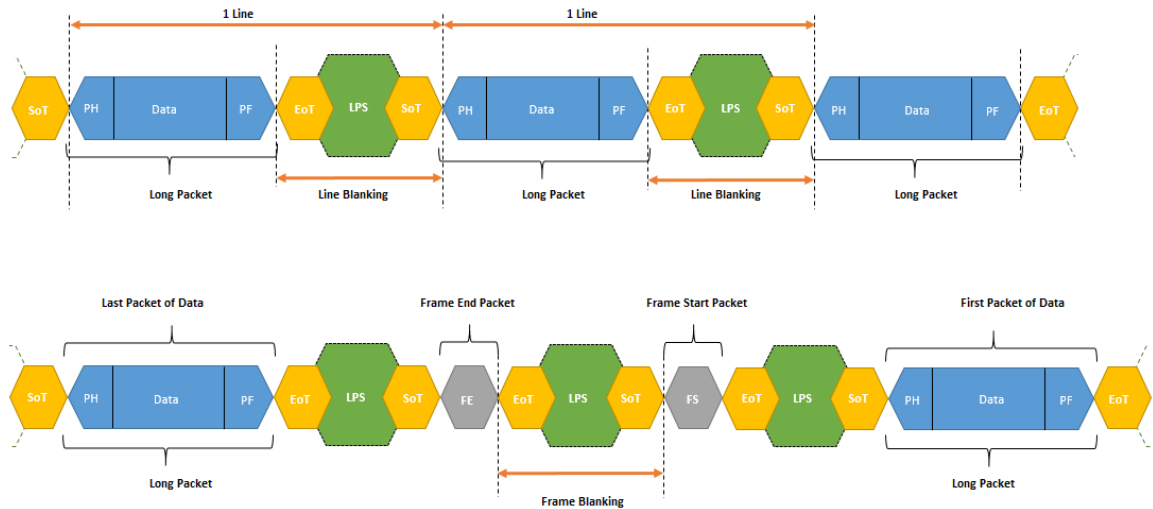


Figure 10 Block Diagram depicting packet structure for CSI sequence

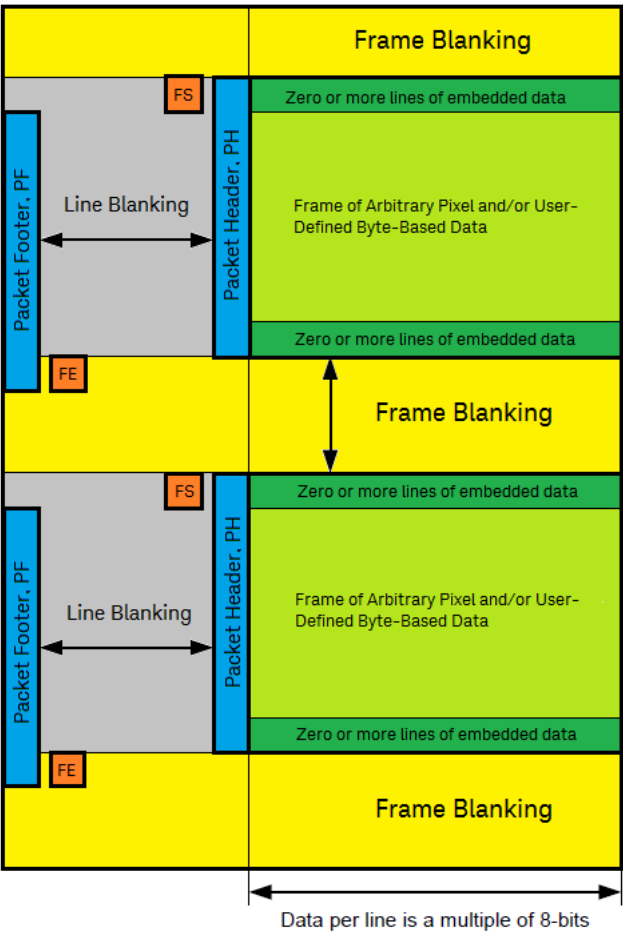
Frame Blanking – The period between the Frame End packet in frame N and the Frame Start packet in frame N+1 is called the Frame Blanking Period.

Line Blanking – The period between the end of the Packet Footer (or the Packet Filler, if present) of one long packet and the Packet Header of the next long packet is called the Line Blanking Period.

Packet Data Payload Size Rules

For YUV, RGB or RAW data types, one long packet shall contain one line of image data. The total size of payload data within a long packet for all data types shall be a multiple of eight bits. The packet payload data format shall agree with the Data Type value in the Packet Header. Refer to *Section 11 Data Formats, Table 3 - Data Type Classes for eight different data type classes and Table 8 - Primary and Secondary Data Formats Definitions of the MIPI Alliance Specification for Camera Serial Interface (CSI)*.

To understand the concept of sequences in CSI implementation, consider the block diagram of a video packet shown below.



KEY:
PH – Packet Header
PF – Packet Footer + Filler (if applicable)
FS – Frame Start
FE – Frame End
LS – Line Start
LE – Line End

Figure 11 Block Diagram of a CSI Video Frame

For each video line transmission, the short packet or the FrameStart (FS) indicates start of transmission of the video packet. The payload data, which is contained in a long packet, consists of the Packet Header (PH), followed by the actual arbitrary data and ending with the Packet Footer (PF). Another short packet or the FrameEnd (FE) indicates the end of transmission of the video packet. This data burst is in an HS state.

A Line Blanking (LP state) is transmitted between each video line, that is, after the end of a Packet Footer (PF) till the beginning of the next Packet Header (PH).

A Frame Blanking (LP state) is transmitted between each video frame, that is, after the end of a FrameEnd (FE) till the beginning of the next FrameStart (FS) short packet.

Understanding a CSI sequence file

The description of the block diagram corroborates the structure of the CSI sequence file shown below, for CSI implementation. Note that all the text files defined in the sequence must be placed in the same folder directory where the sequence file is located.

HSFreq: 158 MBit/s;

Blocks:

FrameStart: B"FrameStart.txt",LP11E10880;

Blanking: LP11E10880;

Video: C1E"compliance640_480.txt",LP11E10880;

FrameEnd: B"FrameEnd.txt",LP11E2048;

Sequence:

1. FrameStart,1;

2. Blanking,1;

3. Video,480;

4. Blanking,2;

5. FrameEnd,1;

The sequence definition in the given example contains an intrinsic looping. In the sequence, the blanking lines generate the frame blanking, and the video lines contain the line blanking, which is generated by the LP states. The number of lines for the video data and the associated blanking is defined by the device manufacturer. The sequence begins with the FrameStart block running once, followed by a Blanking line. Then, the Video block runs for 480 lines ending with another Blanking line, followed by the FrameEnd block running once. This sequence loops over until manually aborted.

If considered closely, the FrameStart block generates the Frame Start (FS) short packet, with the Frame Start Code 0x00 as its header. This follows an HS-LP transition using LP11, where a line blanking is performed with an LPE marker, which fills the LP states until 10880 HS states are attained. The Blanking block generates a frame blanking packet of LP11, which fills the LP states until 10880 HS states are attained and is generated to provide for the LP-HS transition before the actual video payload begins transmitting. In the Video block, the C-Macro with a 1-byte data type of 0x1E generates the Packet Header of the long packet, followed by the actual payload video data (in Hex format) in the compliance640_480.txt file.

Since the video data is High-Speed, the end of the video packet follows an HS to LP transition with a line blanking packet of LP11, which fills the LP states until 10880 HS states are attained. Two blanking lines are sent to indicate the end of the video payload data and to save energy. The FrameEnd block generates the Frame End (FE) short packet, with the Frame End Code 0x01 as its header. This follows an HS-LP transition with a line blanking packet of LP11, which fills the LP states until 2046 HS states are attained.

Some other points to note are:

- Since you cannot send more than one packet per burst, Blanking (LP state) is inserted at the end of each burst to avoid HS data from concatenating.
- The length of the line and frame blanking is device dependent.
- The data type 0x1E corresponds to the pixel color code YUV422 8-bit used in the video. Refer to Section 11 Data Formats of the MIPI Alliance Specification for Camera Serial Interface (CSI) for more information about the other Data Types for various color codes.
- The line blanking length and bits per pixel of a specific color code helps you in determining the total line length in HS states. The E-marker is used for the LP states instead of N, such that it fills up the blocks until the total defined length of HS States is attained.

NOTE

To retain the same line rate for CSI implementation between D-PHY and C-PHY, it is recommended that you keep the number of HS states equal in the LP definition.

If all lines in a sequence file definition have an LPE statement in the end, you may calculate the Frame Rate:

- 1 Multiply the number of lines with the number defined in the LPE statement of the sequence file definition.
- 2 Repeat step 1 for all lanes in the frame and add the resulting values for each lane.
- 3 Multiply the sum of all lanes with the HS period length, which derives the Frame Rate.

Calculating HS Data Rate for CSI sequence

To calculate the minimum HS data rate required to run the sequence, you must be aware of at least the frame rate and the device's display resolution, which is provided by the device manufacturer. The HS Frequency is the first line in the definition of a sequence file.

For example, let us consider that the device under test has a Frame Rate of 30 Hz and a display resolution of 640 x 480 pixels, where 640 is the horizontal resolution (or the length of each line) and 480 is the vertical resolution (or the number of video lines).

- 1 Calculate the line rate using the equation:

$$\text{Line Rate} = \text{Frame Rate} * \text{Vertical Resolution}$$

However, the number of video lines has certain number of blanking lines preceding and following the video data, which must be considered as well for data rate calculation. The equation for line rate is, therefore, modified to:

$$\text{Line Rate} = \text{Frame Rate} * (\text{Vertical Resolution} + \text{no. of blanking lines})$$

Let us assume that there are 10 blanking lines in a frame.

In this case, $\text{Line Rate} = 30 \text{ Hz} * (480 + 10) = 14700 \text{ Hz}$

- 2 Determine the total length of lines in HS states.
 - i Determine the number of bits required for transmission of the video data. To do so, check the pixel color coding for the Data Type in the video. In this case, the pixel color code is YUV422, which uses 8-bit per pixel.

$\text{Total no. of bits per line} = \text{Bit-size per pixel} * \text{Horizontal resolution}$

In this case, $\text{Total no. of bits} = 8 * 640 = 5120 \text{ bits per line}$.

- ii The number of bits for video transmission is not sufficient for determining the total length of lines in HS states, since extra time is required for the LP states in the line blanking. Therefore,

you must consider the LP states and accordingly extend the bits per line. Considering these factors, a total length of 10880 lines in HS states can be safely used for calculation of the HS data rate.

- 3 Calculate the HS Data Rate using the equation:

Data Rate = Line Rate * Total length of lines in HS state

In this case, Data Rate = 14700 Hz x 10880 = 159.936 Mbps

Therefore, you can define the HS Data Rate (HSFreq) in the beginning of the sequence file, as shown in the example above.

For information on the CSI implementation in MIPI D-PHY and MIPI C-PHY physical layer and detailed understanding of the protocol layer, refer to the *MIPI Alliance Specification for Camera Serial Interface (CSI)*.

Sequence File Definition for DSI

Overview

DSI specifies the interface between a host processor and a peripheral such as a display module. It builds on existing MIPI Alliance specifications by adopting pixel formats and command set specified in DPI-2, DBI-489 2 and DCS standards. Some significant differences between DSI and CSI are:

- CSI uses unidirectional high-speed Link, whereas DSI is half-duplex bidirectional Link
- CSI makes use of a secondary channel, based on I2C, for control and status functions
- CSI data direction is from peripheral (Camera Module) to host processor, while DSI's primary data direction is from host processor to peripheral (Display Module)
- CSI sequence file structure is different from that of the DSI sequence file structure. The former consists of only of the FrameStart and FrameEnd packet along with some blanking lines, whereas the latter consists of HSync, VSync and Blanking packages.

At the lowest level, DSI protocol specifies the sequence and value of bits and bytes traversing the interface. It specifies how bytes are organized into defined groups called packets. The protocol defines required headers for each packet, and how header information is generated and interpreted.

On the transmitter side of a DSI Link, parallel data, signal events, and commands are converted in the Protocol layer to packets, following the packet organization. The Protocol layer appends packet-protocol information and headers, and then sends complete bytes through the Lane Management layer to the PHY. Packets are serialized by the PHY and sent across the serial Link. The receiver side of a DSI Link performs the converse of the transmitter side, decomposing the packet into parallel data, signal events and commands.

If there are multiple Lanes, the Lane Management layer distributes bytes to separate PHYs, one PHY per Lane, as described in Section 6 of the MIPI Alliance Specification for Display Serial Interface (DSI). Packet protocol and formats are independent of the number of Lanes used. The DSI protocol permits multiple packets to be concatenated, which substantially boosts effective bandwidth. This is useful for events such as peripheral initialization, where many registers may be loaded with separate write commands at system startup. There are two modes of data transmission, HS and LP transmission modes, at the PHY layer. Before an HS transmission can be started, the transmitter PHY issues a SoT sequence to

the receiver. After that, data or command packets can be transmitted in HS mode. Multiple packets may exist within a single HS transmission and the end of transmission is always signaled at the PHY layer using a dedicated EoT sequence. To enhance the overall robustness of the system, DSI defines a dedicated EoT packet (EoTp) at the protocol layer for signaling the end of HS transmission. In HS mode, time gaps between packets shall result in separate HS transmissions for each packet, with a SoT, LPS, and EoT issued by the PHY layer between packets. This constraint does not apply to LP transmissions.

Long and Short Packet Formats

Two packet structures are defined for low-level protocol communication: Long packets and Short packets. For both packet structures, the Data Identifier (DI) is always the first byte of the packet, which includes information specifying the type of the packet.

Long Packet

A Long packet shall consist of three elements: a 32-bit Packet Header (PH), an application-specific Data Payload with a variable number of bytes, and a 16-bit Packet Footer (PF). The Packet Header is further composed of three elements: an 8-bit Data Identifier, a 16-bit Word Count, and 8-bit ECC. The Packet Footer has one element, a 16-bit checksum. Long packets can be from 6 to 65,541 bytes in length.

After the end of the Packet Header, the receiver reads the next Word Count multiplied by the bytes of the Data Payload.

Once the receiver has read the Data Payload it reads the Checksum in the Packet Footer. The host processor shall always calculate and transmit a Checksum in the Packet Footer. Peripherals are not required to calculate a Checksum. Also, note the special case of zero-byte Data Payload: if the payload has length 0, the Checksum calculation results in (0xFFFF). If the Checksum is not calculated, the Packet Footer shall consist of two bytes of all zeros (0x0000). In the generic case, the length of the Data Payload shall be a multiple of bytes. In addition, each data format may impose additional restrictions on the length of the payload data, e.g. multiple of four bytes.

Short Packet

A Short packet shall contain an 8-bit Data ID followed by two command or data bytes and an 8-bit ECC; a Packet Footer shall not be present. Short packets shall be four bytes in length. The Error Correction Code (ECC) byte allows single-bit errors to be corrected and 2-bit errors to be detected in the Short packet. Some short packets may also contain some data in the payload.

Long and Short packets have several common elements. The first byte of any packet is the DI (Data Identifier) byte. The Error Correction Code allows single-bit errors to be corrected and 2-bit errors to be detected in the Packet Header. The host processor shall always calculate and transmit an ECC byte. Peripherals shall support ECC in both forward- and reverse-direction communications.

DSI Sequence Format Description

Sync Event (H Start, H End, V Start, V End), Data Type = XX 0001 (0xX1)

Sync Events are Short packets and, therefore, can time-accurately represent events like the start and end of sync pulses. As “start” and “end” are separate and distinct events, the length of sync pulses, as well as position relative to active pixel data, e.g. front and back porch display timing, may be accurately conveyed to the peripheral. The Sync Events are defined as follows:

- Data Type = 00 0001 (0x01) V Sync Start
- Data Type = 01 0001 (0x11) V Sync End
- Data Type = 10 0001 (0x21) H Sync Start
- Data Type = 11 0001 (0x31) H Sync End

To represent timing information as accurately as possible a V Sync Start event represents the start of the VSA. It also implies an H Sync Start event for the first line of the VSA. Similarly, a V Sync End event implies an H Sync Start event for the last line of the VSA.

Sync events should occur in pairs, Sync Start and Sync End, if accurate pulse-length information must be conveyed. Alternatively, if only a single point (event) in time is required, a single sync event (normally, Sync Start) may be transmitted to the peripheral. Sync events may be concatenated with blanking packets to convey inter-line timing accurately and avoid the overhead of switching between LPS and HS for every event. Display modules that do not need traditional sync/blanking/pixel timing should transmit pixel data in a high-speed burst then put the bus in Low Power Mode, for reduced power consumption.

EoTp, Data Type = 00 1000 (0x08)

This short packet is used for indicating the end of a HS transmission to the data link layer. Therefore, detection of the end of HS transmission may be decoupled from physical layer characteristics. The main objective of the EoTp is to enhance overall robustness of the system during HS transmission mode. Therefore, DSI transmitters should not generate an EoTp when transmitting in LP mode. The Data Link layer of DSI receivers

shall detect and interpret arriving EoTps regardless of transmission mode (HS or LP modes) to decouple itself from the physical layer. Unlike other DSI packets, an EoTp has a fixed format as follows:

- Data Type = DI [5:0] = 0b001000
- Virtual Channel = DI [7:6] = 0b00
- Payload Data [15:0] = 0x0F0F
- ECC [7:0] = 0x01

Blanking Packet (Long), Data Type = 01 1001 (0x19)

A Blanking packet is used to convey blanking timing information in a Long packet. Normally, the packet represents a period between active scan lines of a Video Mode display, where traditional display timing is provided from the host processor to the display module. The blanking period may have Sync Event packets interspersed between blanking segments. Like all packets, the Blanking packet contents shall be an integer number of bytes. Blanking packets may contain arbitrary data as payload.

The Blanking packet consists of the DI byte, a two-byte WC, an ECC byte, a payload of length WC bytes, and a two-byte checksum.

Packed Pixel Stream, 16-bit Format, Long Packet, Data Type 00 1110 (0x0E)

This long packet (shown in the sequence example) is used to transmit image data formatted as 16-bit pixels to a Video Mode display module. The packet consists of the DI byte, a two-byte WC, an ECC byte, a payload of length WC bytes and a two-byte checksum.

Packet Header Error Detection/Correction

The host processor in a DSI-based system shall generate an error-correction code (ECC) and append it to the header of every packet sent to the peripheral. The ECC takes the form of a single byte following the header bytes. The ECC byte shall provide single-bit error correction and 2-bit error detection for the entire Packet Header.

Checksum Generation for Long Packet Payloads

Long packets are comprised of a Packet Header protected by an ECC byte and a payload of 0 to 216-1 bytes. To detect errors in transmission of Long packets, a checksum is calculated over the payload portion of the data packet. Note that, for the special case of a zero-length payload, the 2-byte checksum is set to 0xFFFF.

Checksum generation and transmission is mandatory for host processors sending Long packets to peripherals. It is optional for peripherals transmitting Long packets to the host processor. However, the format of Long packets is fixed; peripherals that do not support checksum generation shall transmit two bytes having value 0x0000 in place of the checksum bytes when sending Long packets to the host processor. The host processor shall disable checksum checking for received Long packets from peripherals that do not support checksum generation.

NOTE

An ECC byte can be applied to both Short and Long packets. Checksum bytes shall only be applied to Long packets.

Transmission Packet Sequences

DSI supports several formats, or packet sequences, for Video Mode data transmission. The peripheral timing requirements dictate which format is appropriate. In the following sections, Burst Mode refers to time-compression of the RGB pixel (active video) portion of the transmission.

Non-Burst Mode with Sync Pulses

This mode enables the peripheral to accurately reconstruct original video timing, including sync pulse widths. Normally, periods shown as HSA (Horizontal Sync Active), HBP (Horizontal Back Porch) and HFP (Horizontal Front Porch) are filled by Blanking Packets, with lengths (including packet overhead) calculated to match the period specified by the peripheral data sheet. Alternatively, if there is sufficient time to transition from HS to LP mode and back again, a timed interval in LP mode may substitute for a Blanking Packet, thus saving power. During HSA, HBP and HFP periods, the bus should stay in the LP-11 state.

Values in Red text are for a vertical resolution of 480p. Active image size is 720CLK (Horizontal) x 480 Lines (Vertical).

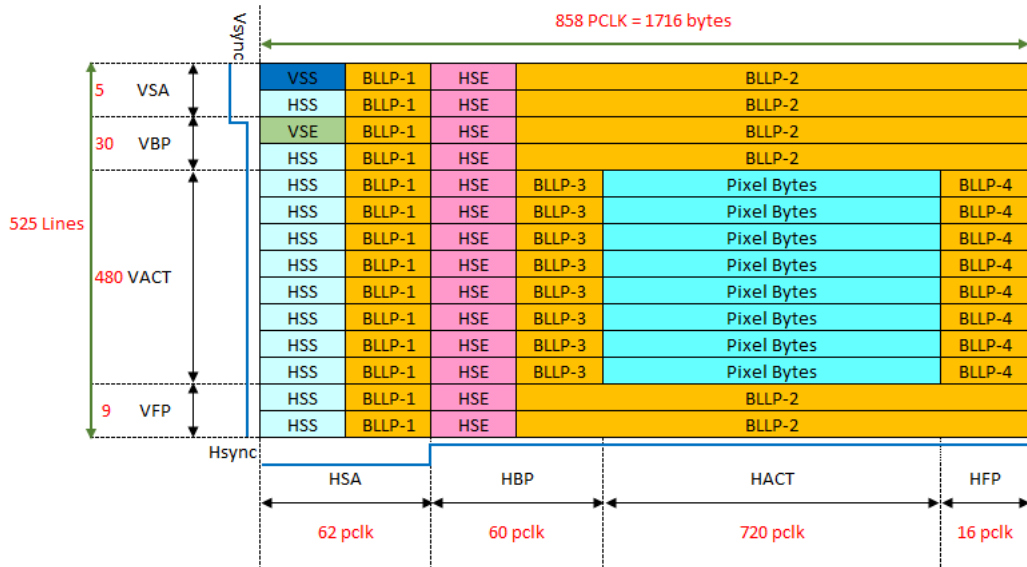


Figure 12 Block Diagram for Video transmission in Non-Burst mode with Sync Pulses

Non-Burst Mode with Sync Events

This mode functions in the same manner as the previous mechanism, but accurate reconstruction of sync pulse widths is not required, so a single Sync Event is substituted. Here, only the start of each synchronization pulse (VSyncSHSyncStart and HSyncStart) is transmitted. The peripheral may regenerate sync pulses as needed from each Sync Event packet received. Pixels are transmitted at the same rate as they would in a corresponding parallel display interface such as DPI-2. As with the previous Non-Burst Mode, if there is sufficient time to transition from HS to LP mode and back again, a timed interval in LP mode may substitute for a Blanking Packet, thus saving power.

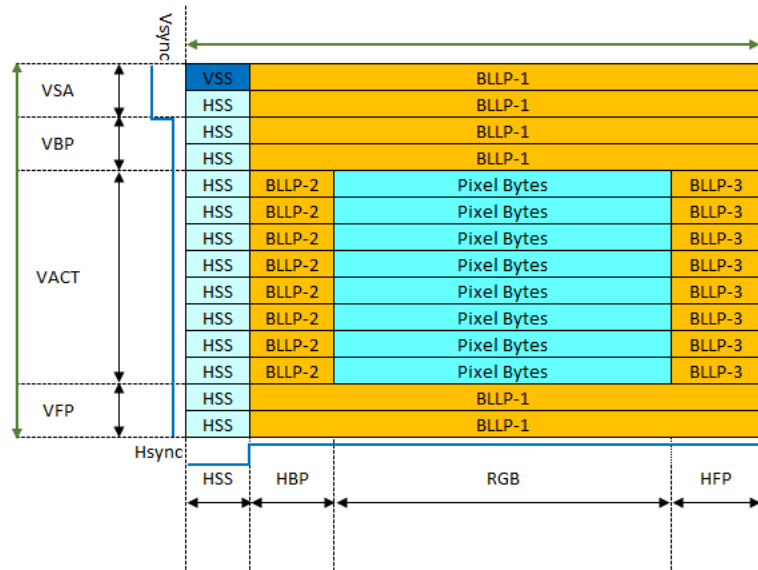


Figure 13 Block Diagram for Video transmission in Non-Burst mode with Sync Events

Burst Mode

RGB pixel packets are time-compressed, leaving more time during a scan line for LP mode (saving power) or for multiplexing other transmissions onto the DSI link. In this mode, blocks of pixel data can be transferred in a shorter time using a time-compressed burst format. This is a good strategy to reduce overall DSI power consumption, as well as enabling larger blocks of time for other data transmissions over the Link in either direction. In the same manner as the Non-Burst Mode scenario, if there is sufficient time to transition from HS to LP mode and back again, a timed interval in LP mode may substitute for a Blanking Packet, thus saving power.

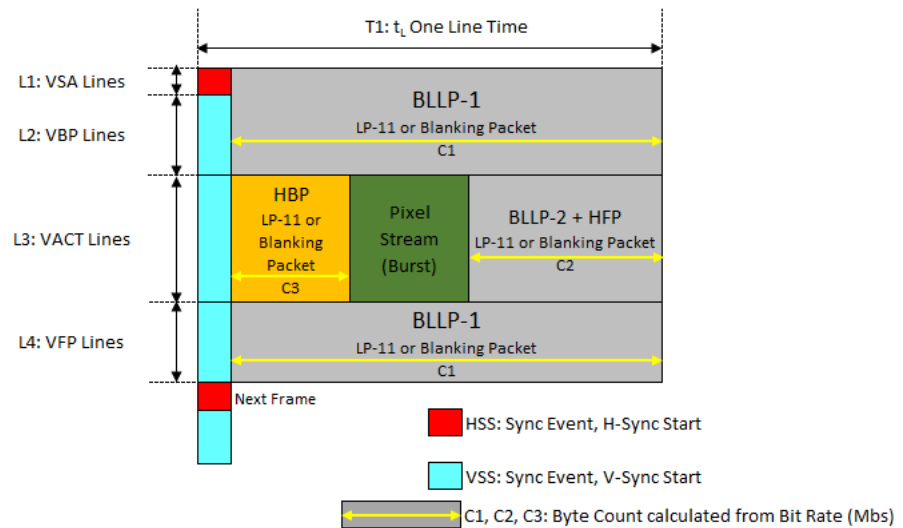


Figure 14 Block Diagram for Video transmission in Burst mode

Note that for accurate reconstruction of timing, packet overhead including Data ID, ECC, and Checksum bytes should be taken into consideration.

To enable PHY synchronization the host processor should periodically end HS transmission and drive the Data Lanes to the LP state. This transition should take place at least once per frame; The host processor should return to LP state once per scan-line during the horizontal blanking time. Regardless of the frequency of BLLP periods, the host processor is responsible for meeting all documented peripheral timing requirements. Note, at lower frequencies BLLP periods will approach, or become, zero, and burst mode will be indistinguishable from non-burst mode.

The sequence of packets within the BLLP or RGB portion of a HS transmission is arbitrary. The host processor may compose any sequence of packets, including iterations, within the limits of the packet format definitions. For all timing cases, the first line of a frame shall start with VSS; all other lines shall start with VSE or HSS. Note that the position of synchronization packets, such as VSS and HSS, in time is of utmost importance since this has a direct impact on the visual performance of the display panel.

Replacing B-Macros with C-Macros in a sequence file

Table 7 shows how a sequence file written originally using the B-Macros can be alternatively written using C-Macros.

Table 7 Sequence file definition using B-Macros and C-Macros

Sequence File Definition using B-Macro	Sequence File Definition using C-Macro
HSFreq: 432.432MBit/s; Blocks: LPInit: LPB"HSyncEnd.txt",LP11E13728; HSync: B"HSyncEnd.txt",LP11E12736,B"HSyncStart.txt",LP11E13728; VSyncStart: B"HSyncEnd.txt",LP11E12736,B"VSyncStart.txt",LP11E13728; VSyncEnd: B"HSyncEnd.txt",LP11E12736,B"VSyncEnd.txt",LP11E13728; Video: B"HSyncEnd.txt",LP11E960,B"Video480pHSyncStart.txt",LP11E13728; Sequence: 1. LPInit,1; 2. HSync,9; 3. VSyncStart,1; 4. HSync,5; 5. VSyncEnd,1; 6. HSync,29; 7. Video,480; LoopTo 2;	HSFreq: 432.432MBit/s; Blocks: LPInit: LPB"HSyncEnd.txt",LP11E13728; HSync: C310000,C080F0F,LP11E12736,C210000,C080F0F,LP11E13728; VSyncStart: C310000,C080F0F,LP11E12736,C010000,C080F0F,LP11E13728; VSyncEnd: C310000,C080F0F,LP11E12736,C110000,C080F0F,LP11E13728; Video: C310000,C080F0F,LP11E960,C0E"Video480p.txt",BL20,C210000,C080F0F,LP11E13728; Sequence: 1. LPInit,1; 2. HSync,9; 3. VSyncStart,1; 4. HSync,5; 5. VSyncEnd,1; 6. HSync,29; 7. Video,480; LoopTo 2;

Upon considering each block closely, we notice the following differences, otherwise the rest of the sequence definition remains the same.

- In the HSync block, the B"HSyncEnd.txt" is replaced by C310000,C080F0F and B"HSyncStart.txt" is replaced by C210000,C080F0F.
- In the VSyncStart block, the B"HSyncEnd.txt" is replaced by C310000,C080F0F and B"VSyncStart.txt" is replaced by C010000,C080F0F.
- In the VSyncEnd block, the B"HSyncEnd.txt" is replaced by C310000,C080F0F and B"VSyncEnd.txt" is replaced by C110000,C080F0F.
- In the Video block, the B"HSyncEnd.txt" is replaced by C310000,C080F0F and B"Video480pHSyncStart.txt" is replaced by C0E"Video480p.txt",BL20,C210000,C080F0F.

To understand how the replacements were done, you must read the description given in the earlier sections about the B“<filename>” macro, C<3 hex bytes> and C<1 hex byte>” <filename>” macros.

Let us consider the contents of each text file closely.

Contents of HSyncEnd.txt

3100 0001

080F 0F01

As mentioned in the previous sections, the Data Type for the HSyncEnd signal is 0x31, which means this short packet has its Data ID as 31 and the following two bytes of Packet DATA as 00 00 followed by the ECC of 01.

Therefore, in a sequence, B“HSyncEnd.txt” can be written using the C<3 hex bytes> macro as C310000.

Contents of HSyncStart.txt

2100 0012

080F 0F01

As mentioned in the previous sections, the Data Type for the HSyncStart signal is 0x21, which means this short packet has its Data ID as 21 and the following two bytes of Packet DATA as 00 00 followed by the ECC of 12.

Therefore, in a sequence, B“HSyncStart.txt” can be written using the C<3 hex bytes> macro as C210000.

Contents of VSyncEnd.txt

1100 0014

080F 0F01

As mentioned in the previous sections, the Data Type for the VSyncEnd signal is 0x11, which means this short packet has its Data ID as 11 and the following two bytes of Packet DATA as 00 00 followed by the ECC of 14.

Therefore, in a sequence, B“VSyncEnd.txt” can be written using the C<3 hex bytes> macro as C110000.

Contents of VSyncStart.txt

0100 0007

080F 0F01

As mentioned in the previous sections, the Data Type for the VSyncStart signal is 0x01, which means this short packet has its Data ID as 01 and the following two bytes of Packet DATA as 00 00 followed by the ECC of 07.

Therefore, in a sequence, B"VSyncStart.txt" can be written using the C<3 hex bytes> macro as C010000.

Notice that within each text file considered so far, a hexadecimal value 080F 0F01 is mentioned. This is the EoT package, which is used at end of each HS transmission in a short packet. The EoT package has a fixed format with Data ID as 08, Payload Data as 0F0F and ECC of 01.

Therefore, at the end of each C<3 hex bytes> macro defined for the four files, you must add the C<3 hex bytes> macro for the EoT package as C080F0F, as shown in the sequence definition.

Contents of Video480pHSyncStart.txt

```
0EA0 0508
1084 1084
1084 1084
1084 1084
.
.
.
1084 1084
FA44 1914
001F 0000
0000 0000
0000 0000
0000 0000
0000 0000
0000 6F1D
2100 0012
080F 0F01
```

The elements of the actual video data in the file 'Video480pHSyncStart.txt' can be divided as follows:

- The header consists of Data ID: 0E followed by a two-byte word counter A005, followed by the ECC for that header data as 08.
- The actual video payload data starts from a hex value of 1084 and ends at a hex value of 1084. Note that the payload data is too long and has been truncated for documentation purpose.
- At the end of the payload, a two byte CRC is added, which is FA44.
- After the checksum, a Blanking packet is added to convey the blanking timing information in this video packet. This Blanking packet consists of the DI byte of 0x19, a two-byte Word Count of 1400, an ECC byte of 1F, a payload of 20 bytes, and a two-byte checksum of 6F1D. This blanking packet enables the HS-LP transition and corresponds to the HFP (Horizontal Front Porch) before the HSyncStart.
- In the end, a short packet in HS mode is generated, which has the HSyncStart data, which allows the HS-LP transition, before the device goes into a low power state.

Looking back at the Video block, the B"Video480pHSyncStart.txt" is replaced by C0E"Video480p.txt",BL20,C210000,C080F0F, which indicates that only the actual payload data has been extracted into another text file named Video480p.txt and the rest of the data is appended using the C-Macro to the beginning and to the end of the payload data file. Notice that the Blanking packet with Data ID 0x19, which was part of the initial payload data, has been rewritten using the BL<number of blanking bytes> macros as BL20.

Irrespective of whether you use the B-Macros or the C-Macros, you must ensure that the data is defined in the correct order and you can use the sequence file in either format for waveform generation in the plug-in.

Notice that there has been no change made to the LPInit block of the sequence file, even though the HSyncEnd.txt file is used. This is because even though the HSyncEnd.txt contains hexadecimal data for HS mode, the LPB"filename" macro is used for generating Low Power data specified in the file HSyncEnd.txt, such that the display device is powered on.

To know more about the Data Types required to construct the short or long packets or to define C-Macros in a sequence file, refer to the *MIPI Alliance Specification for Display Serial Interface*.

Understand a DSI sequence file

The DSI sequence file corresponds to the package structure shown in the image below, which is based on the Non-Burst mode with Sync Pulses:

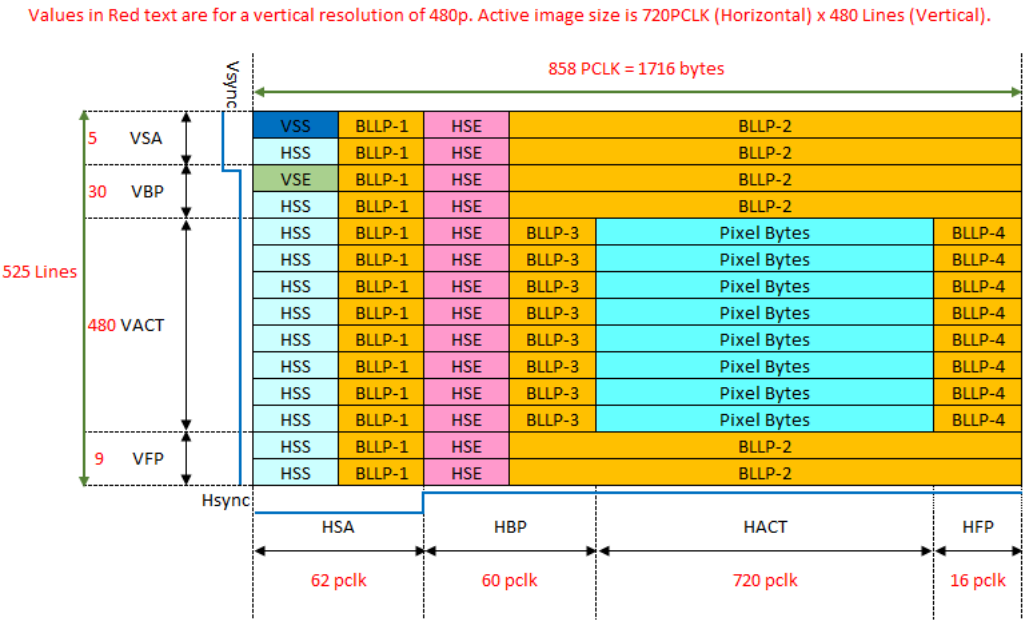


Figure 15 Block Diagram for Video transmission in Non-Burst mode with Sync Pulses

In general, the way the DVI functions is like the CRT mode. In a video frame of the CRT mode, an electron beam performed several horizontal traces and a vertical trace to begin the next frame. In the current technology, the HSS corresponds to the several horizontal traces whereas VSS corresponds to the vertical trace.

The VSyncStart (VSS) and HSyncStart (HSS) pulse are generated to mark the start of the video frame and define the timings of the device. For a video with resolution 720x480p, there are 720 horizontal active pixels, represented by HACT and 480 vertical active lines, represented by VACT. The device manufacturer provides the information about the horizontal blanking pixels and the vertical blanking lines that must be included to meet the timing and power saving requirements of the device, apart from the video transmission. In this case, a total of 138 horizontal blanking pixels are added to the 720 horizontal active pixels. Also, a total of 45

vertical blanking lines are added to the 480 vertical active lines. These HSync and VSync data is required for proper synchronization of the video data and to achieve proper timing in displaying the video data and is represented by HSyncActive (HSA) and VSyncActive (VSA), respectively. As described earlier, HSS, HSE, VSS and VSE are short packets containing High Speed data and blanking header information. They are part of the HSA and VSA.

Sync events are introduced such that only HSS have to be added. In this case, where Sync pulses are used, Blanking Lines can be added, which may be HS or LP, depending on the timing set between HSS and HSE.

The Horizontal Back Porch (HBP), Horizontal Front Porch (HFP), Vertical Back Porch (VBP), Vertical Front Porch (VFP) are indicated by BLLPs. The Blanking or Low-Power Interval (BLLP) is defined as a period during which video packets such as pixel-stream and sync event packets are not actively transmitted to the peripheral. The BLLP provides for the HS-LP and LP-HS transition when the device is powered on, during the switch to video mode and to save the device's power either in the idle state or just before the start of the video data. In this image, the HBP and HFP are indicated by BLLP-3 and BLLP-4, respectively whereas VBP and VFP are indicated by BLLP-1 and BLLP-2, respectively.

Now, let us consider the example given below of a sequence file for DSI implementation, derived from the video package displayed in the image above.

HSFreq: 432.432MBit/s;

Blocks:

LPInit: LPB"HSyncEnd.txt",LP11E13728;

HSync: C310000,C080F0F,LP11E12736,C210000,C080F0F,LP11E13728;

VSynStart: C310000,C080F0F,LP11E12736,C010000,C080F0F,LP11E13728;

VSynEnd: C310000,C080F0F,LP11E12736,C110000,C080F0F,LP11E13728;

Video:

C310000,C080F0F,LP11E960,C0E"Video480p.txt",BL20,C210000,C080F0F,LP11E13728;

Sequence:

- 1. LPInit,1;**
- 2. HSync,9;**
- 3. VSynStart,1;**
- 4. HSync,5;**

5. VSyncEnd,1;

6. HSync,29;

7. Video,480;

LoopTo 2;

The LPInit block generates the HsyncEnd signal is a low power mode to power on the display device.

To achieve proper synchronization and timing for the video data, the HSync and VSync signals are used. The number of vertical blanking lines and horizontal blanking pixels are provided by the device manufacturer.

The HSync block, which loops over for 9 lines, generates the HSE signal, indicated by C310000,C080F0F before the device switches to low power mode until 12736 HS states are attained. The LP11E12736 corresponds to BLLP-2 on the image. Then, HSS signal is generated, indicated by C210000,C080F0F, after which the device enters into low power mode again until 13728 HS states are attained. The LP11E13728 corresponds to BLLP-1 on the image.

The VSyncStart block, which loops over once, generates the HSE signal followed by LP11E12736 state. Then, VSS signal is generated, indicated by C010000,C080F0F, after which the device enters into low power mode again until 13728 HS states are attained.

The HSync block loops over for 5 lines again before the VSyncEnd block loops over once. The VSyncEnd block generates the HSE signal followed by LP11E12736 state. Then, VSE signal is generated, indicated by C110000,C080F0F, after which the device enters into low power mode again until 13728 HS states are attained.

The HSync block loops over for 29 lines again before the Video block loops over for 480 lines. In the Video block, the HSE signal is generated followed by LP11E960 state. Here, in the duration between the Video line and HSyncEnd, there is a short LP state to allow the device to save power before the video starts. LP11E960, which forms the HBP, allows device to save power before switching to video mode. COE"Video480p.txt" indicates the header information followed by the active payload data contained in the "Video480p.txt" file. The Video block displays BL20, which forms the HFP, before the HSS is generated again followed by switching into low power mode until 13728 HS states are attained.

Note that BLLP-1 and BLLP-2, that is, LP11E12736 and LP11E13728 are blanking lines. Instead of sending video data, long LP states are generated. The total number of HS states is driven by the E marker, which

means each line has a fixed duration, depending on the data rate. It includes the HS states before the LPE marker is defined and the time taken to switch to the HS mode.

You must always ensure that to define a proper sequence, each block must end with the same HS or LP state to avoid any unexpected loops.

Calculating HS Data Rate for the DSI sequence

To calculate the minimum HS data rate required to run the sequence, you must be aware of at least the frame rate, the Blanking Lines (HBlank and VBlank) and the device's display resolution, in other words, the pixels per line and lines per frame, which are provided by the device manufacturer. The HS Frequency is the first line in the definition of a sequence file.

For example, let us consider that the device under test has a Frame Rate of 60 Hz and a display resolution of 640 x 480 pixels, where 640 is the horizontal resolution (or the length of each line) and 480 is the vertical resolution (or the number of video lines). The number of header (blanking lines) is given as 45.

- 1 Calculate the total no. of lines using the equation:

Total Lines = Video Lines + Header (Blanking) lines = 480 + 45 = 525 lines

- 2 Calculate the Line Rate using the equation:

Line Rate = Frame Rate * Total Lines = 60 Hz x 525 lines = 31.5 kHz

- 3 Calculate the No. of pixels per line using the equation:

No. of pixels per line = HBlanking + Horizontal Resolution

where, the HBlanking data is given by the device manufacturer. Let us assume 160 HBlanking lines.

Therefore, No. of pixels per line = 160 + 640 = 800 pixels per line.

- 4 Determine the number of bits required for transmission of the video data. To do so, check the pixel color coding for the Data Type in the video from the device's specification. In this case, the pixel color code is RGB, which uses 24-bits per pixel.

Calculate the total no. of bits per line using the equation:

Total no. of bits per line = Bit-size per pixel * No. of pixels per line

In this case, Total no. of bits per line = 24 * 800 = 19.2 kbits per line

- 5 Calculate the HS Data Rate using the equation:

Data Rate = Line Rate * Total no. of bits per line

In this case, Data Rate = 31.5 kHz * 19.2 kbits per line = 604.8 Mbps

Therefore, for a VGA mode video with a frame rate of 60 Hz, you must set a data rate of 604.8 Mbps, which you can define as HSFreq in the beginning of the sequence file.

For information on the DSI implementation in MIPI D-PHY and MIPI C-PHY physical layer and detailed understanding of the protocol layer, refer to the *MIPI Alliance Specification for Display Serial Interface*.

3 Using the MIPI C-PHY Editor User Interface

Basic Requirements	/ 68
Installing Plug-in	/ 73
Related Documents	/ 74
Starting the MIPI C-PHY Editor Plug-in	/ 75
Contacting Keysight Technologies	/ 77
MIPI C-PHY Editor User Interface	/ 78
Setting up MIPI C-PHY Editor Parameters	/ 97
Performing In-System AWG Calibration with Keysight IQ Tools	/ 128

Basic Requirements

Hardware Setup using M8195A AWG Module

Single-lane Setup

The required hardware setup for single-lane using M8195A module are following:

- One M8195A module
- An Embedded Controller (such as M9537A) or an external controller
- M9502A AXIe chassis
- An Infiniium or SCPI compatible oscilloscope
- LAN or GPIB adapter

For hardware setup, follow the given steps:

- 1 Put M8195A and an Embedded Controller in the 2 slot frame AXIe chassis as shown in [Figure 16](#).

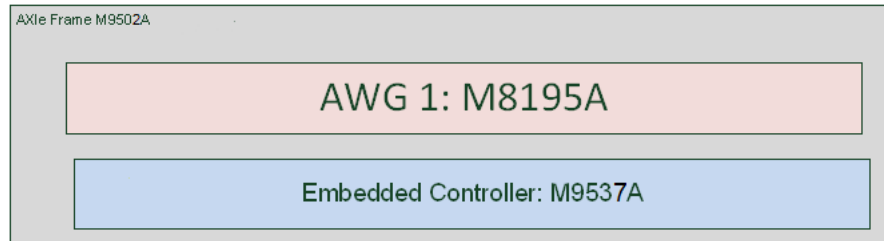


Figure 16 Hardware setup for Single-lane with module M8195A

- 2 M9502A contains only two slots. The Embedded Controller must be installed in the slot 1 of the AXIe chassis otherwise, it will not be able to connect to the internal PCIe interface of the frame.
- 3 Then slot 2 contains the M8195A AWG module.

NOTE

The module itself is already de-skewed. It means that all the signal outs are synchronized with respect to each other, i.e. Data Out 1 is synchronous to rest of the data out locations.

The only prerequisite is to have 'matched triplet' length for the 3-cable MIPI C-PHY connections. This can be done by doing a de-skew calibration.

For more detail, refer to [De-skew without Re-cabling](#) on page 17.

Multi-lane Setup

Following hardware setup is required for multi-lane:

- Two or three M8195A AWG modules to enable the multi-lane support
- One M8197A module to synchronize multi M8195A modules
- AXIe chassis
 - 5 slot AXIe chassis for two lane structure i.e. 2 slots for two M8195A modules and 1 slot for one M8197A module
 - 5 slot AXIe chassis for three lane structure i.e. 3 slots for three M8195A modules and 1 slot for one M8197A module

NOTE

M8197A module is used to synchronize M8195A modules (Lane 1, Lane 2 and Lane 3). However, if you are using only single M8195A module (Lane 1), then M8197A is not required.

For hardware setup (e.g. three Lane structure i.e. three M8195A modules), follow the given steps:

- 1 Put three M8195A and one M8197A in the 5 slot frame AXIe chassis as shown in the [Figure 17](#).

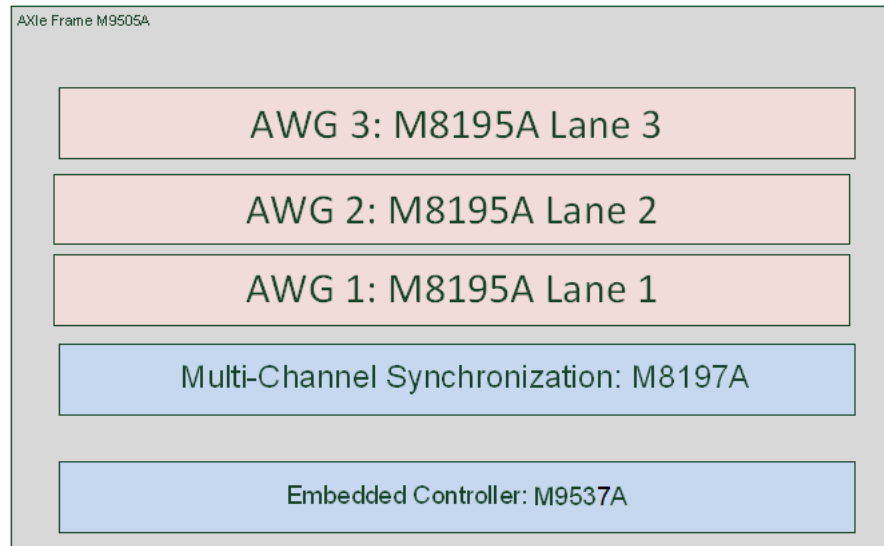


Figure 17 Hardware setup with Multi-lane M8195A

- 2 The modules should be arranged inside a 5 slot chassis in the following order:
 - Slot 1: M8197A (Used for synchronization)
 - Slot 2: M8195A (Used for lane 1)
 - Slot 3: M8195A (Used for lane 2)
 - Slot 4: M8195A (Used for lane 3)

NOTE

You can also use an embedded controller for M8070B and/or the AWG SFPs. It should be installed in slot 1 of AXI chassis, thereby shifting the other modules one slot up.

NOTE

The module for multi-lane M8195A is also de-skewed itself. For more detail, refer to [De-skew without Re-cabling](#) on page 17.

Software Requirements

To install the MIPI C-PHY Editor plug-in, the M8070B software (version S6.0.100.2 or later) is required. You can download the software from the following link:

<http://www.keysight.com/find/M8070B>

License Requirements

The MIPI C-PHY Editor plug-in is a licensed feature. To enable it, following are the required licenses:

Table 8 License required for MIPI C-PHY Editor plug-in

P/N	License	Description
M8085CE1A	-1TP	MIPI C-PHY 1.1 Editor for M819xA AWG, Transportable, Perpetual License
	-1NP	MIPI C-PHY 1.1 Editor for M819xA AWG, Network/Floating, Perpetual License
	-TRL	MIPI C-PHY 1.1 Editor for M819xA AWG, 30 Day Trial License
M8085CC1A	-1TP	MIPI C-PHY 1.1 Calibration, Conformance and Characterization Procedures for M819xA AWG, Transportable, Perpetual License
	-1NP	MIPI C-PHY 1.1 Calibration, Conformance and Characterization Procedures for M819xA AWG, Network/Floating, Perpetual License
	-TRL	MIPI C-PHY 1.1 Calibration, Conformance and Characterization Procedures for M819xA AWG, 30 Day Trial License
M8085CUEA	-1TP	Upgrade C-PHY Editor from M8085A-CT1 to C-PHY 1.1, Transportable, Perpetual License
	-1NP	Upgrade C-PHY Editor from M8085A-CN1 to C-PHY 1.1, Network/Floating, Perpetual License

P/N	License	Description
M8085CUCA	-1TP	Upgrade C-PHY Editor plus Calibration, Conformance and Characterization Procedures from M8085A-CT1 and M8085A-CTA to C-PHY 1.1, Transportable, Perpetual License
	-1NP	Upgrade C-PHY Editor plus Calibration, Conformance and Characterization Procedures from M8085A-CN1 and M8085A-CNA to C-PHY 1.1, Network/Floating, Perpetual License
N5990A	-010	Test Sequencer

Installing Plug-in

The MIPI C-PHY Editor plug-in must be installed separately by the M8070B system software.

Please make sure that the system should already have M8070B (version S6.0.100.2 or later) software installed on it.

M8195A SFP should be installed from v3.6.0.0 and M8197A SFP should be installed from v3.6.0.0.

The installer for the MIPI C-PHY Editor plug-in is available either on CD or you may download it from the from the following Keysight web-page: www.keysight.com/find/m8085a.

For details on how to install the MIPI C-PHY Editor plug-in, refer to the *Keysight M8085A Plugins for MIPI Receiver Test Solutions Installation Guide*.

Related Documents

The plug-ins are installed separately from the plug-in manager.

For details on how to use the **Plug-in Manager** to install, uninstall and update the **M8085A** plug-in, refer to the *Installing M8085A Plugins for MIPI Receiver Test Solutions* section in *Keysight M8000 Series of BER Test Solutions Plugins for M8070B System Software Getting Started Guide*.

For M8070B plug-ins related documents, click **Start > Keysight M8070B > Keysight M8070B Documentation**.

Starting the MIPI C-PHY Editor Plug-in

To access the installed MIPI C-PHY Editor plug-in through the M8070B system software:

- 1 Click **Start** > **Keysight M8070B**. The user interface for the M8070B system software appears.
- 2 From the M8070B user interface menu, click **Application** to view the list of all installed plug-ins.
- 3 Select the **MIPI C-PHY Editor** plug-in.
- 4 The **MIPI C-PHY Editor** user interface appears as shown in **Figure 18**.

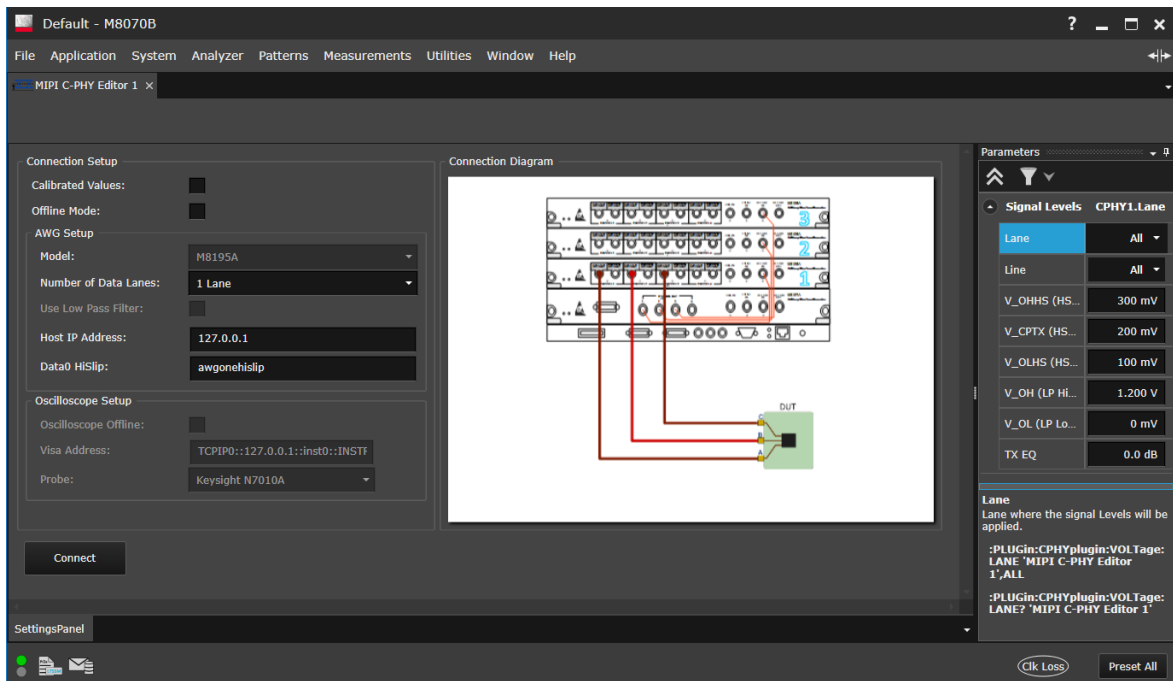


Figure 18 MIPI C-PHY Editor Plug-in User Interface

NOTE

If you are unable to see the MIPI C-PHY Editor option in the **Applications** menu, you must install it separately. Ensure that you have a valid license to install the software.

Currently, the shortcut for Keysight M8085A plug-ins is not auto-generated during installation.

To manually create a shortcut to the Keysight M8070B software on your desktop to access the M8085A plug-ins, perform the following steps:

- 1 Navigate to *C:\Program Files\Keysight\M8070B\bin* folder on your machine.
- 2 Right-click *Keysight.M8070B.exe* and click **Create Shortcut**.
- 3 Click Yes on the 'Shortcut' prompt to place the shortcut on the desktop.
- 4 On the desktop, right-click *Keysight.M8070B – Shortcut* icon and click **Properties**.
- 5 In the **Properties** window, modify the 'Target' location to "*C:\Program Files\Keysight\M8070B\bin\Keysight.M8070B.exe*" */IgnoreAwg*.
- 6 Click **Apply** and exit the **Properties** window.

You may launch the *Keysight.M8070B – Shortcut* to access the M8085A plug-ins.

Contacting Keysight Technologies

For more information on products, applications or services associated with Keysight Technologies, contact your local Keysight office. The complete list is available at: www.keysight.com/find/contactus.

MIPI C-PHY Editor User Interface

Beginning with version 2.7 of the M8085A plug-ins, the MIPI C-PHY Editor supports the M8195A AWG configuration only.

M8195A Configuration

With the M8195A configuration, you can generate up to three MIPI C-PHY lanes. For single lane configuration, you need only one M8195A module. However, for multi-lane configuration, that is, two lane or three lane structure, you require two or three M8195A modules respectively. Additionally, for multi-lane configuration, an M8197A synchronization module is also required for proper alignment between the modules.

Connection Diagrams

The M8085A MIPI C-PHY Editor is a stand-alone software utility that allows you to set the DUT (Device Under Test) and test configuration. The MIPI C-PHY Editor is a flexible tool for trouble-shooting and debugging. It complements the full Test Automation Software, which provides automated physical layer compliance tests and characterization. The software runs on a standard Windows PC and controls the hardware test resources through appropriate interfaces, for example; LAN.

The following figure represents a typical MIPI C-PHY Editor setup for M8195A (with one data lane):

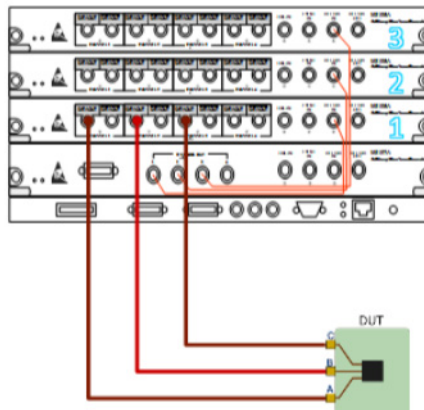


Figure 19 M8085A MIPI C-PHY Editor for M8195A (with one data lane)

The following figure represents a typical MIPI C-PHY Editor setup for M8195A (with two data lanes):

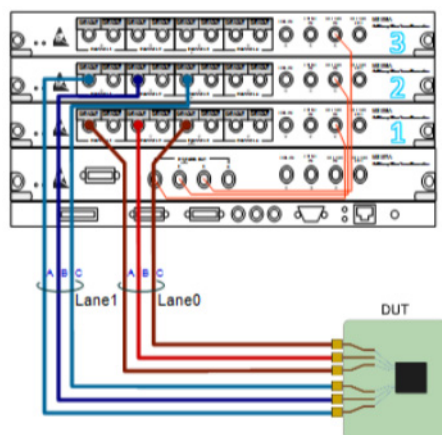


Figure 20 M8085A MIPI C-PHY Editor for M8195A (with two data lanes)

The following figure represents a typical MIPI C-PHY Editor setup for M8195A (with three data lanes):

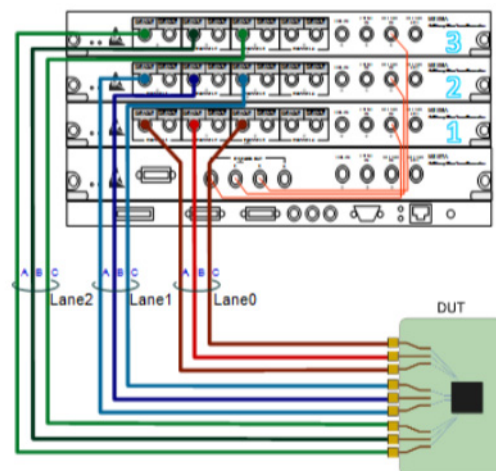


Figure 21 M8085A MIPI C-PHY Editor for M8195A (with three data lanes)

The **MIPI C-PHY Editor** user interface includes the following GUI elements:

Connection Setup

The connection setup is the initial step before starting to generate MIPI C-PHY signals. The connection setup shows the connection diagram to instruct the user on how to connect the system. All the configuration options, to properly set up the system, are available in this panel.

Regardless of the selected configuration of the M8195A AWG, the connection panel provides two special options as following:

- **Calibrated Values:** Select this option to run the MIPI C-PHY Editor with calibrated signal levels. The calibrated values are available after running some of the calibration procedures that are part of the MIPI C-PHY CTS.
When this option is not selected or when no calibration data is available the uncalibrated values are used.
- **Offline Mode:** Select this option to run the MIPI C-PHY Editor without connecting to the real instruments. This is intended to familiarize the user with the MIPI C-PHY Editor User Interface and for preliminary debugging without connecting to real instruments.

M8195A Connection Setup

The M8195A AWG supports up to three C-PHY lanes. For every increased lane you require an additional M8195A AWG. This means that for two and three lane structure you require two and three M8195A AWG modules, respectively. These can be called as M8195A Lane 1, M8195A Lane 2, and M8195A Lane 3. A multi channel synchronization module (M8197A) is required to make a proper synchronization and therefore there is no need to connect to an oscilloscope for synchronization.

Connection Setup

Calibrated Values: ☐

Offline Mode: ☐

AWG Setup

Model: M8195A

Number of Data Lanes: 1 Lane

Use Low Pass Filter: ☐

Host IP Address: 127.0.0.1

Data0 HiSlip: awgonehislip

Oscilloscope Setup

Oscilloscope Offline: ☐

Visa Address: TCPIP0::127.0.0.1::inst0::INST1

Probe: Keysight N7010A

Connect

Figure 22 M8195A Connection Setup (1 Lane)

- **AWG Setup**
 - **Model:** Select M8195A module for M8195A configuration.
 - **Number of Data Lanes:** Select the number of lanes to generate the MIPI C-PHY signal. Currently, the software supports up to 3 lanes.
 - **Host IP Address:** This is the network address of the controller computer where the M8195A system is connected. The default Host IP Address is defined as “127.0.0.1”.
 - **Clock Sync HiSlip:** When more than one lane is selected, the Clock Sync HiSlip identifier needs to be filled. The HiSlip should be in the format “hislip<no.>”, for example, hislip2.
 - **Data0 HiSlip:** This is the HiSlip identifier of the AWG module that generates the MIPI C-PHY lane signal. The HiSlip for Data0 is defined as “hislip<no.>”, for example, hislip0. Similarly, the HiSlip address for Data1 and Data2 are defined in the same format.

NOTE

The values for Host IP Address, Data0 HiSlip, Data1 HiSlip, Data2 HiSlip, Clock Sync HiSlip given above for M8195A are just examples of the correct format. It is recommended to check and type the correct values for each of these setup characteristics on the respective device at your end.

Once all above mentioned connection settings are done, click the “Connect” button or execute the SCPI command to connect to the instruments. The main user interface appears. For details, refer to the section [Main User Interface](#).

Main User Interface

The main user interface is shown in [Figure 23](#) for the M8195A AWG configuration and is available after connecting the MIPI C-PHY even in “Offline Mode”. The main user interface contains all the parameters of MIPI C-PHY.

At any moment, it is possible to disconnect from the instruments by clicking the “Disconnect” button.

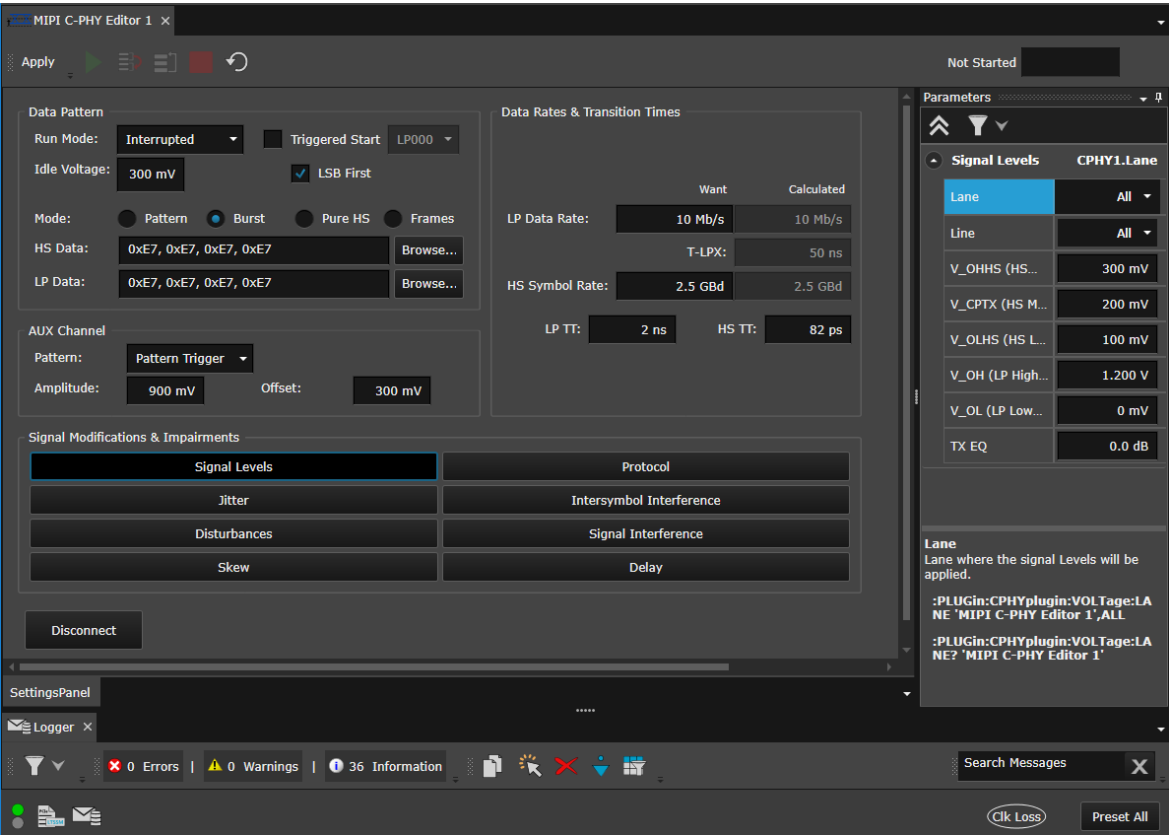


Figure 23 MIPI C-PHY Editor Main User Interface for M8195A

Main user interface consists of the following elements:

- 1 [Toolbar](#)
- 2 [Data Pattern Selection](#)

- 3 Data Rates and Transition Times
- 4 AUX Channel
- 5 Signal Modifications & Impairments
- 6 Parameters
- 7 Status Indicator
- 8 Logger Window

NOTE

Other MIPI C-PHY parameters can be accessed by toggling the “Signal Modifications & Impairments” category buttons. When a button is toggled, the corresponding parameters associated with the button category are displayed on the “Parameters” panel.

NOTE

The “Parameters” panel is normally present on the right side of the User Interface unless the user has chosen to move it to different region of the User Interface.

Toolbar

The toolbar provides the following shortcuts:

Table 9 Toolbar

Icons	Name	Description
	Apply/Abort	The “Apply” command calculates the waveform based on the configured settings. During a waveform calculation, this waveform changes to “Abort”. Pressing “Abort” causes an internal abort of the current calculation and a new waveform calculation starts. After the waveform calculation finishes, the AWG is put into run mode. A change in the MIPI C-PHY settings does not cause a waveform recalculation. To apply changes in MIPI C-PHY settings apply, you must press “Apply” button.
	Start	The “Start” command puts the AWG into run mode. This command is available only after a waveform is applied.
	Trigger	The “Trigger” command enables a jump from the initial loop block to next block. This option is available only if a pattern with the option “triggered startup” was programmed into the AWG.
	Restart	The “Restart” command stops and starts the AWG. It does not reprogram any waveform on the AWG. It is available only if a waveform is applied on the AWG.
	Stop	The “Stop” command puts the AWG into stop mode. The command is available only after a waveform is applied and if the AWG is running.
	Reset	The “Reset” command resets the settings to the default values and calculates the waveform with the default settings.

Data Pattern Selection

“Data Pattern” includes activating “Triggered Start” and for different “Mode” defined as following:

- Run Mode: The Run Mode enables switching between continuous (ping-pong) mode and interrupted mode.
 - In the Continuous mode, if the waveform calculation has already been performed, when you press “Apply” for the new settings, the AWG is not stopped. In this case, the AWG outputs the previous waveforms. After the new waveform is calculated, the AWG then activates the new waveform, without stopping the AWG.

NOTE

It is mandatory that for some combinations of settings, you must ensure that the AWG is always stopped (for example, HS Symbol Range change, etc.).

- In the Interrupted mode, the functionality continues to be the same, that is, applying a new waveform always triggers the AWG to stop.
- Idle Voltage: The Idle voltage sets the offset on the AWG output amplifiers. This is the output offset voltage when the AWG is in the Stop state.
- Activating “Triggered Start”: The purpose of the “Triggered Startup” is to enable the MIPI C-PHY plug-in to “Triggered Start” mode with a static LP STOP signal.

To configure a trigger sequence on the MIPI C-PHY Editor user interface,

- select the “Triggered Start” check-box in the Data Pattern area.
- select the Low Power state from the drop-down options that must be used in the initial looped block. See [Figure 24](#).

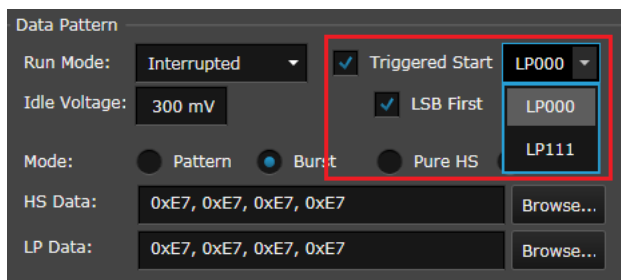


Figure 24 Activating Triggered Start on the user interface

- LSB First: Select this check-box to transmit the Least Significant Bit (instead of the Most Significant Bit) first in the Data Pattern.

The data is looped until you send a trigger that breaks from the loop (the trigger is sent using the MIPI C-PHY Editor Toolbar). After breaking from the initial loop, the second loop that contains the data starts. The “Triggered Start” is especially important for DUTs that require special initialization to properly receive data.

- Modes: The MIPI C-PHY Editor plug-in supports four modes; “Pattern”, “Burst”, “Pure HS” and “Frames”, which are as following:

- Pattern Mode: The Pattern mode and the Burst mode are same. In this mode you can load pattern files with the extension '*.ptrn' written in terms of line states. A *.ptrn file contains LP and/or HS data definitions.

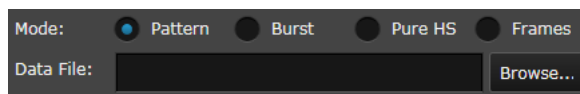


Figure 25 Pattern Mode

- Burst Mode: In the Burst mode a block of data is repeated infinitely. This block can contain either LP data, HS data or both, which can be typed directly on the user interface or they can also be loaded by data files with the extension '*.dat'.

NOTE

The data must be formatted in hexadecimal bytes separated by commas as shown in [Figure 26](#).

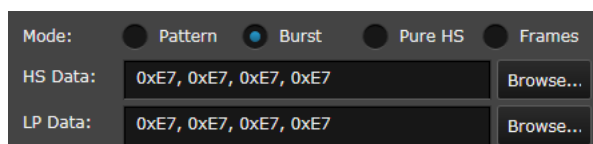


Figure 26 Burst Mode

- Pure HS Mode: The Pure HS mode is also similar to the Burst mode expect that no LP111 transitions are included and all the LP data is neglected.

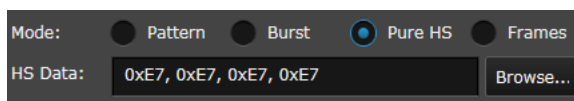


Figure 27 Pure HS Mode

- **Frames:** In Frames mode a complex sequence file can be loaded (*.seq). In this sequence file the blocks containing the data can be specified and the sequence behavior can also be specified. For more information, refer to [Chapter 2 Sequence and Data Files](#) on page 27.

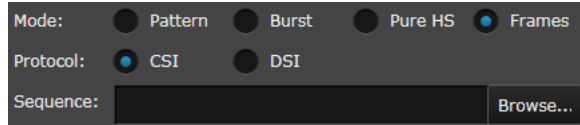


Figure 28 Frames Mode

NOTE

With Mode: selected as Frames, the DSI protocol has two version options (either V1.0 or V1.1) to choose from.

Setting Pattern and Sequence

Along with processing simple pattern files (*.ptrn) and data files (*.dat) for waveform calculation, the MIPI C-PHY Editor plug-in is also capable of generating valid video frame data, in compliance with either the CSI protocol or the DSI protocol. The MIPI C-PHY Editor user interface enables you to select the type of protocol (CSI or DSI) because the long and short packet structure of high speed mode is different in CSI and DSI specification respectively. Selecting the protocol becomes imperative in order to generate the proper long and short packet structure. The section below briefly explains the structures of simple and complex sequence files.

You may also run [SCPI Commands for Data Pattern Group](#) on page 156.

Loading complex sequences for CSI and DSI video frames

The Data Pattern Mode option Frames enables using complex sequence files for generating valid video frame data in CSI or DSI format.

To load a sequence file into the MIPI C-PHY Editor user interface,

- in the **Data Pattern** section, from the options for **Mode:**, select **"Frames"**.
- from the options for **Protocol:**, select either **"CSI"** or **"DSI"**, depending on the protocol compliance testing you require to perform.

- For the DSI protocol, the MIPI C-PHY Editor user interface displays two drop-down options to choose the DSI version from. By default, V1.0 is selected. The other option is V1.1. The difference in both DSI versions is that the sequence structure of DSI V1.0 contains the SSS block whereas this block is unavailable in the sequence structure of DSI V1.1.
- in the “**Sequence**” field that appears, click “**Browse...**” to navigate to the sequence file on your computer. See [Figure 29](#).

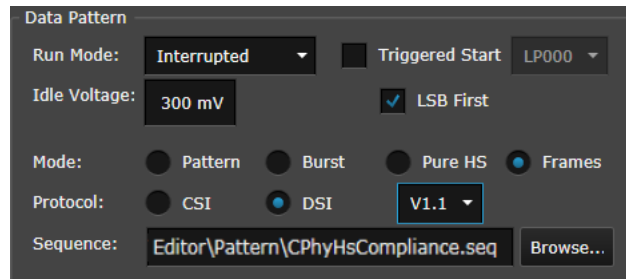


Figure 29 Loading a sequence file in the Frame mode of the user

The MIPI C-PHY Editor plug-in supports complex sequence files and is not limited to the sequence structure as shown in [Figure 30](#) and [Figure 31](#).



Figure 30 Pattern Loop Type

To define a sequence, declare the pattern blocks and sequence instructions in a text file.

To achieve a Pattern Loop type of the MIPI C-PHY Editor, which is represented in [Figure 30](#), you may create a Pattern Loop Type Sequence text file that has the following text:

```
Blocks:
Init: <Pattern>
Loop: <Pattern>

Sequence:
1. Init, 1;
2. Data, 1;
LoopTo 2;
```

The MIPI C-PHY Editor also supports loading pattern files that contain line states.

To load pattern files that contain line states, you may create a sequence file (in the text format) by defining the folder navigation path to the pattern files, as shown below.

```
Blocks:
Init: P"<pathtofile>InitPattern.ptn"
Data: P"<pathtofile>LoopPattern.ptn"

Sequence:
1. Init, 1;
2. Data, 1;
LoopTo 2;
```

To achieve the PRBS Loop Type, which is represented in [Figure 31](#), you may use the same sequence file shown above along with declaring a “Data” block with a PRBS9 pattern

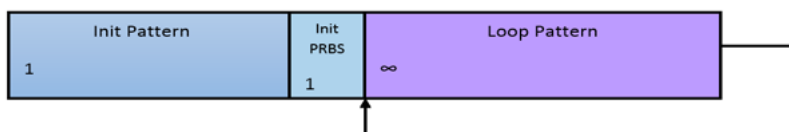


Figure 31 PRBS Loop Type

The PRBS can be initialized with a custom seed, e.g. 0x2789a. An example of a sequence file containing a PRBS9 looped block is shown below. Similarly, PRBS11 or PRBS18 patterns can be generated by replacing PRBS9 with PRBS11 or PRBS18 respectively.

```
Blocks:
Init : <Pattern>, LPHS
Data: PRBS9(0x2789a)

Sequence:
1. Init, 1;
2. Data, 1;
LoopTo 2;
```

Since the MIPI C-PHY Editor plug-in supports complex sequences to achieve a CSI or DSI video frame data, you require a video frame loop type, represented in [Figure 32](#).

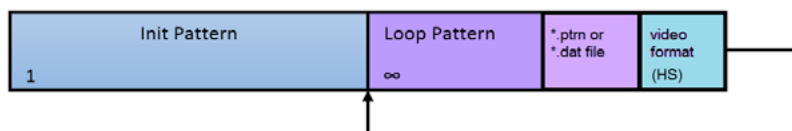


Figure 32 Complex HS video data loop type

Structure of the CSI and DSI Sequence File

For details on the structure of the sequence file for CSI and DSI, refer to [CSI and DSI Sequences](#) on page 38.

NOTE

D-PHY Editors cannot use a sequence that contains MIPI C-PHY specific macros such as three wire LP states (for example, LP000, LP011, etc.) or the P-Macro (for example, P"Patternfile.ptn"). Otherwise, both MIPI C-PHY and MIPI D-PHY Editors use the same sequence structure.

Data Rates and Transition Times



Figure 33 Data Rates and Transition Times

The Data Rates & Transition Times provide the following options:

- LP Data Rate and HS Symbol Rate: The LP and HS stand for Low Power and High Speed respectively, which defines the data rates. In these fields you can adjust the low power data rate and the high speed symbol rate. However, in some specific situations, the exact data rate or symbol rate cannot be fulfilled. For these situations, you can use the “Calculated” data rates (as shown in [Figure 33](#)).

NOTE

Beginning with version 2.7 onwards, the M8085A MIPI C-PHY Editor plug-in supports a minimum LP Data Rate value of 1Mbps. However, using such lower data rate values can impact the pattern calculation time and memory usage. Therefore, the default LP Data Rate is set to 10 Mbps.

- LP TT and HS TT: The LP TT and HS TT stands for Low Power transition time and High Speed transition time respectively. They allow you to adjust the rise and fall time (both) of the Low Power and High Speed modes independently.

AUX Channel

For M8195A, the 4th channel does not normally generate any data. However, you may utilize this 'unused channel' at times to have some data generated for the purposes of debugging or any other miscellaneous functions. Considering this point, the M8085A plug-in has the AUX Channel setting for the M8195A AWG in the waveform generation user interface of the plug-in, as shown in [Figure 34](#).

AUX Channel settings

As shown in [Figure 34](#), you may click the drop-down options to select the desired Pattern:

- **Pattern Trigger:** Select this option to generate a stable triggered pattern on oscilloscope. Note that the trigger is synchronous with the infinite loop generated. As a result, the oscilloscope displays the same part of the pattern within the loop.
- **Clock Pattern:** Select this option to initiate eye trigger on high speed data.
- **Mirror Line A / B / C:** Select the desired option to mirror the data from one of the lines to either Line A, B or C.
- **Off:** Select this option to disable the auxiliary channel.

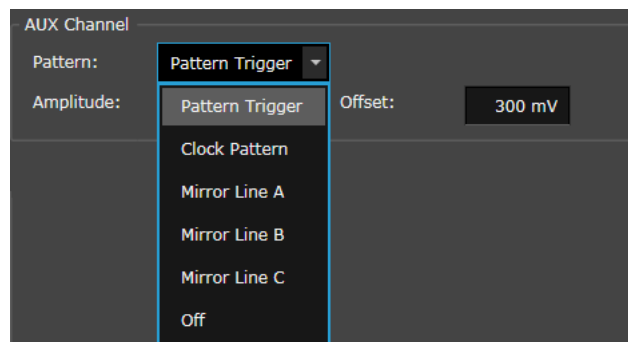


Figure 34 AUX Channel

Use the Amplitude and Offset settings to change the amplitude and offset, respectively, of the signal output on the 4th channel of the AWGs, as this channel is not normally used to generate the MIPI C-PHY Signals. These settings are available only when the Pattern is set to either "Pattern Trigger" or "Clock Pattern".

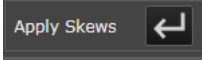
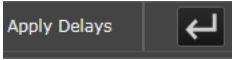
Signal Modifications & Impairments

All the parameters of MIPI C-PHY editor are accessible by the Signal Modification & Impairments panel. By selecting the parameter group from the “Signal Modification & Impairments” panel (identified with the number 1 in [Figure 35](#)), the corresponding parameters of the group appears in the “Parameters” panel (identified with the number 2 in [Figure 35](#)).

Changing the value of any parameter on the user interface does not perform waveform recalculation automatically anymore. You must explicitly request for a waveform recalculation by clicking the “Apply” button on the user interface.

NOTE

Changing the values of “Skews” and “Delays” group parameters are not applied immediately. To apply the made changes

- Press  button for Skews
- Press  button for Delays
- or you can also press global “Apply” button from the toolbar (for both Skews and Delays).

For M8195A pressing the “Apply” button (or Apply Skews/Delays button), restarts the sequence in all cases.

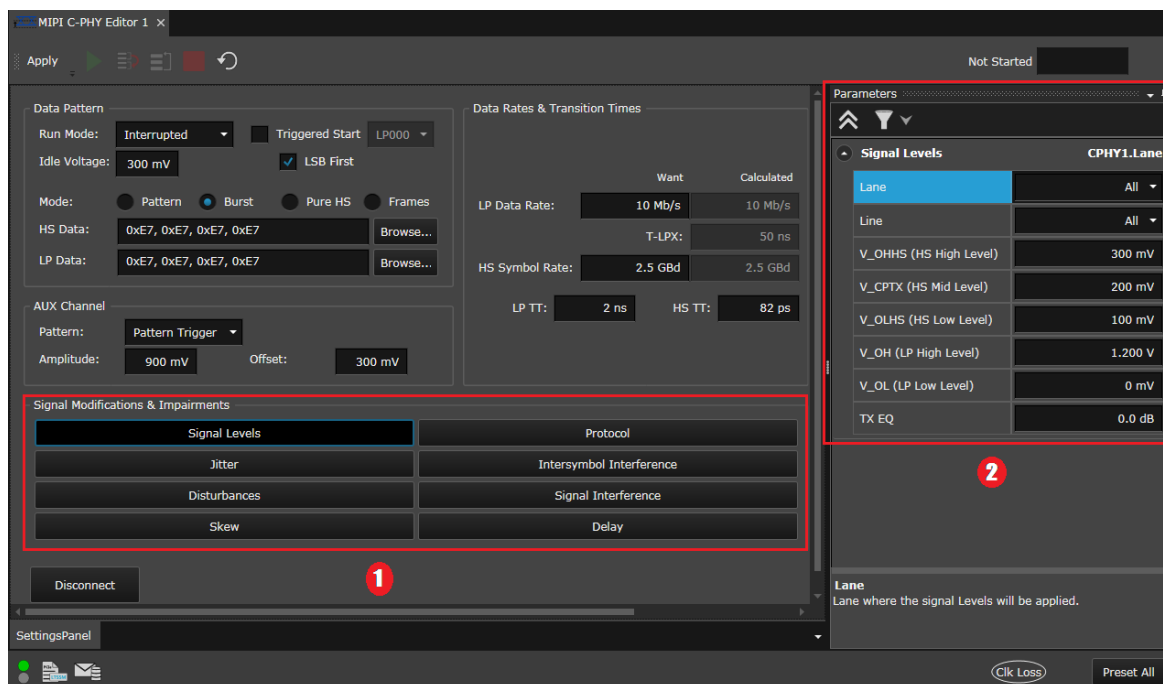


Figure 35 Signal Modifications & Impairments

Parameters

The Parameters window allows you to set the parameters for MIPI C-PHY Editor. It provides the following parameters group:

- Signal Levels Group
- Protocol Group
- Jitter Group
- Intersymbol Interference Group
- Disturbances Group
- Signal Interference Group
- Skew Group
- Delay Group

For detail description of parameter refer the section [Setting up MIPI C-PHY Editor Parameters](#) on page 97.

NOTE

To initiate waveform calculation, click the “Apply” button on the toolbar. Also, any change in the MIPI C-PHY settings does not trigger waveform calculation automatically anymore. Changes made to the MIPI C-PHY settings are only taken into account after you click the “Apply” button.

Status Indicator

The status indicator shows the current state of a waveform calculation based on the configured settings. It shows the following type of status:

- Not Started: Indicates that the waveform calculation activity is not started.
- Running: Indicates that the waveform calculation activity is currently running.
- Finished: Indicates that the waveform calculation activity is finished and the pattern file is sent.

The following figure shows the status indicator while the waveform calculation activity is in progress:



Figure 36 MIPI C-PHY Status Indicator

Logger Window

The Logger Window displays description of errors, warnings and information messages along with the applications from where they were generated and their time stamps.

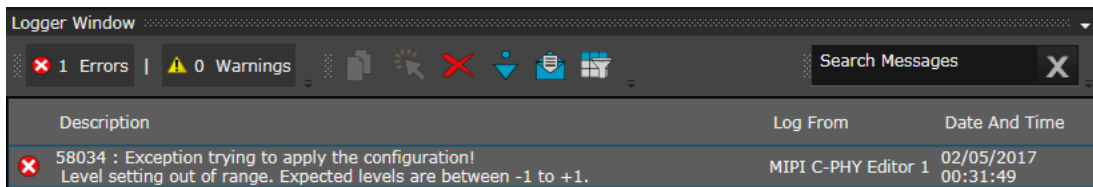


Figure 37 Logger Window

Setting up MIPI C-PHY Editor Parameters

To change the value of any parameter that belongs to the 'double' data type, click the field displaying the value of the parameter as shown in figure (red box indicated by number 1). It opens a parameter dialog box, in which you can write the new desired value of parameter within the acceptable range (red box indicated by number 2).

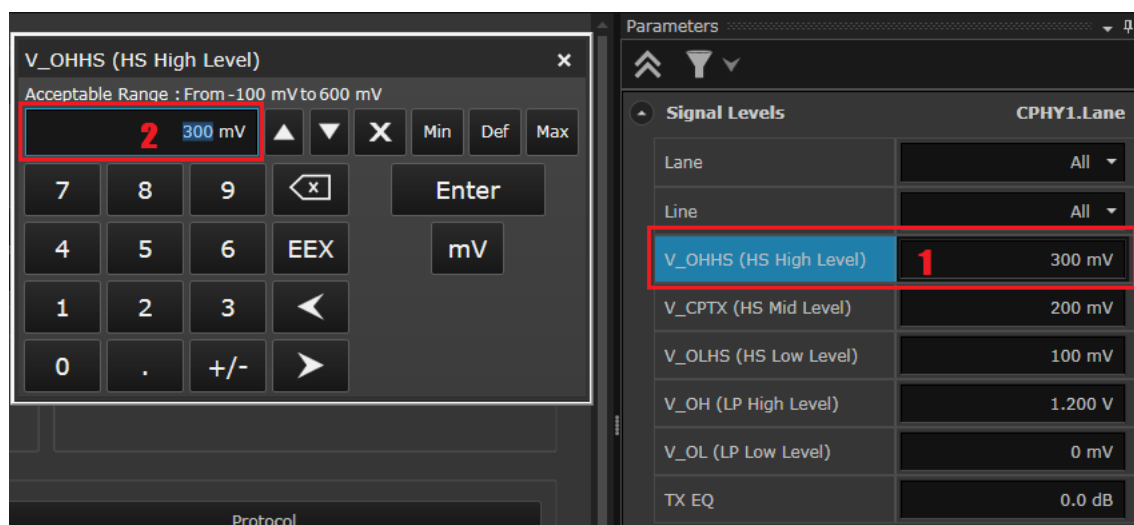


Figure 38 Setting values for parameters

NOTE

If you want to change the value of any parameter, every-time you must explicitly request for a waveform calculation by clicking the “Apply” button on the user interface. The user interface does not perform waveform calculation automatically.

Signal Levels Group

To change the voltage levels of the signal, click the Signal Levels button in the “Signal Modifications & Impairments” area such that the corresponding settings are visible on the Parameters panel (see [Figure 39](#)).

Other than setting values for Signal Levels in the Parameters panel of the user interface, you may also run the [SCPI Commands for Signal Levels Group](#) on page 197.

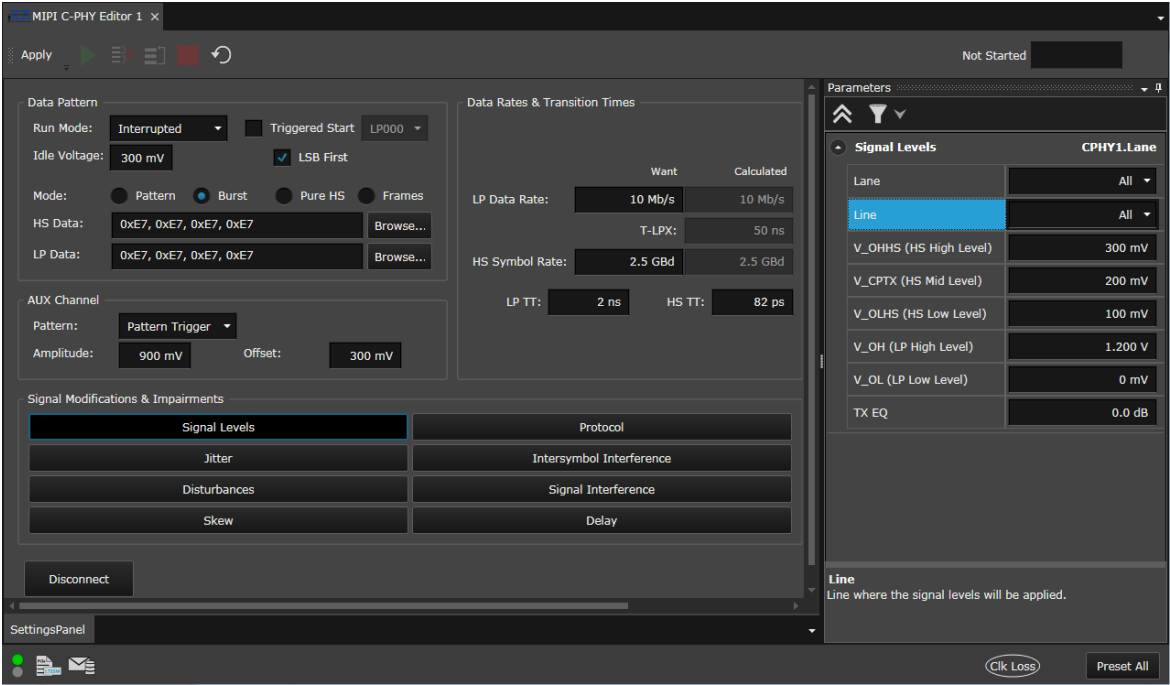


Figure 39 Setting the Signal Levels

Setting different signal levels for lanes and lines

On the MIPI C-PHY Editor user interface, you may modify the voltage level for specific lanes, lines or both while keeping the signal levels for the rest of the data lanes and lines the same. This feature is helpful when you are running compliance tests on the DUT for M8195A AWG configuration, where you may want to understand the impact of different voltage levels on specific data lanes or wires.

Referring to [Figure 39](#), notice in the “Parameters” panel that by default, common voltage levels are assigned for all lines and lanes in a single instance of the MIPI C-PHY Editor plug-in. That is, with the “Lane” and “Line” set to “All”, the values set for the signal level settings (V_OHHS, V_CPTX, V_OLHS, V_OH, V_OL) are the same for each data lane and line.

However, for waveform calculation, if required, you can modify the voltage levels for specific Lanes, Lines, or both without impacting the voltage levels on other Lanes or Lines. The MIPI C-PHY Editor plug-in saves the values for the voltage levels that you modify for a certain combination of Lane and Line and does not change them automatically. However, if you perform a waveform calculation for “All” option in Lanes, Lines or both in the same MIPI C-PHY Editor instance, the modified values for specific lanes or lines are reset to the voltage level values set for the “All” option in Lanes, Lines or both.

As mentioned earlier, you may use a single instance of the MIPI C-PHY Editor plug-in to connect to multiple lanes on the M8195A AWG. This means that by default, the MIPI C-PHY Editor plug-in sets a common signal level on all lines and lanes on M8195A AWG, due to configuration using a single instance. However, for waveform calculation, there may be instances where you may find a need to configure different signal levels for one of the following combinations:

Data0 - Line A, Data0 - Line B, Data0 - Line C or Data0 - All Lines

Data1 - Line A, Data1 - Line B, Data1 - Line C or Data1 - All Lines

Data2 - Line A, Data2 - Line B, Data2 - Line C or Data2 - All Lines

All Lanes - Line A, All Lanes - Line B, All Lanes - Line C or All Lanes - All Lines

Moreover, because of the high number of combinations, it may be difficult to remember the values set for different signal levels for each combination.

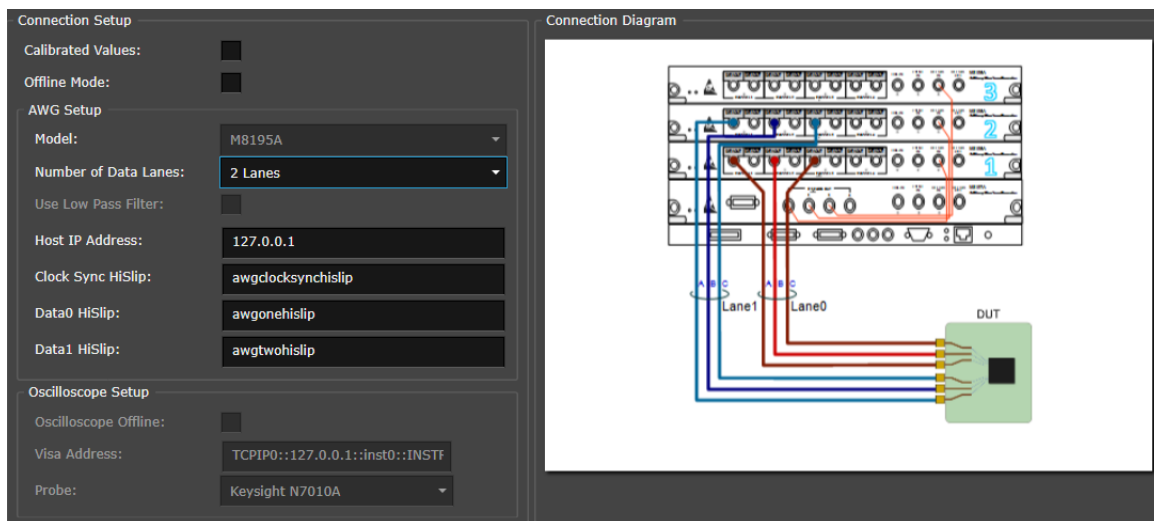
To accomplish such configuration where you may have to set values that are deviated from the default standard values, the **Parameter** area has been modified in such a manner that selecting **Signal Levels** allows you to select one of the combinations of data **Lane** and **Line** (as mentioned above) and set a signal voltage level for that combination.

The value for the voltage levels that you set for either of the combinations does not change by itself. That is, the MIPI C-PHY Editor plug-in stores the set value and does not change them except for two situations –

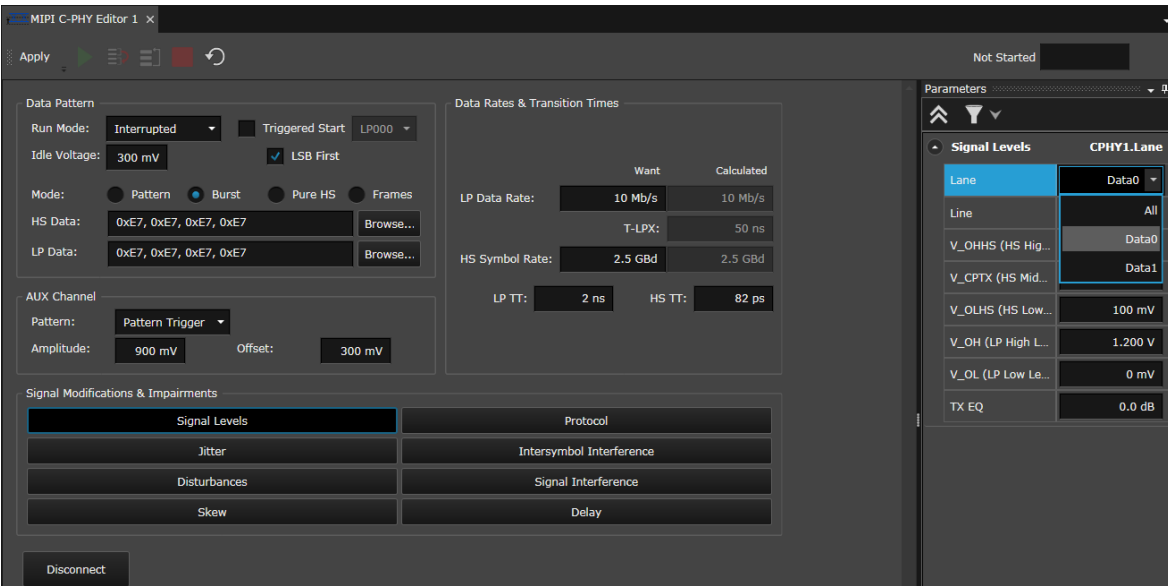
- when you manually change the values for those combinations, or
- you select All Lanes - All Lines combination for another connection setup and run it in the same instance.

To understand this concept in a better manner, consider the following example:

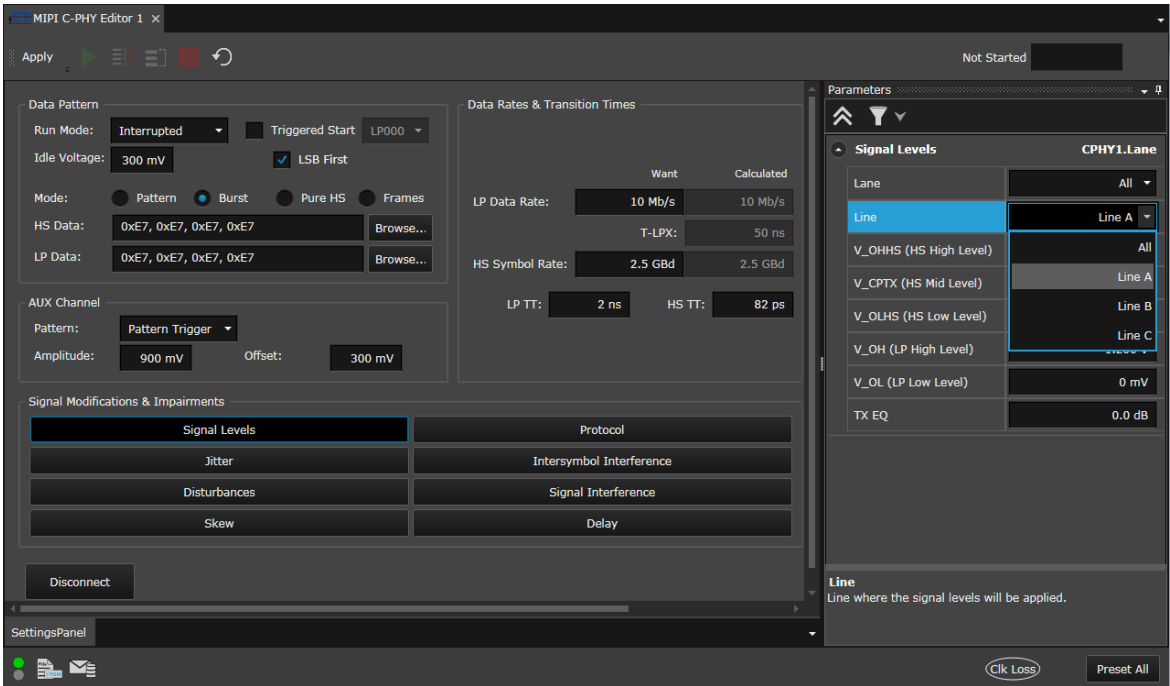
- 1 In the **Connection Setup** section, connect the MIPI C-PHY Editor plug-in to 2-Lanes of M8195A AWG.



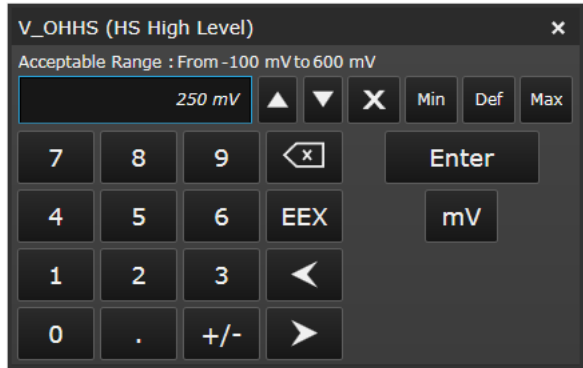
- 2 In the **Signal Modifications & Impairments** section, select **Signal Levels** and from the drop-down options for **Lane**, select a lane whose signal level you wish to modify.



- 3 Similarly, from the drop-down options for **Line**, select a line whose signal level you wish to modify.



4 Modify one or more signal values as per your requirements. Note that by default, each field contains the values compliant to the MIPI C-PHY standards.



V_CPTX (HS Mid Level) ✕

Acceptable Range : From -100 mV to 600 mV

100 mV ▲ ▼ ✕ Min Def Max

7	8	9	⌫	Enter	
4	5	6	EEX		mV
1	2	3	⬅		
0	.	+/-	➡		

V_OLHS (HS Low Level) ✕

Acceptable Range : From -100 mV to 600 mV

50 mV ▲ ▼ ✕ Min Def Max

7	8	9	⌫	Enter	
4	5	6	EEX		mV
1	2	3	⬅		
0	.	+/-	➡		

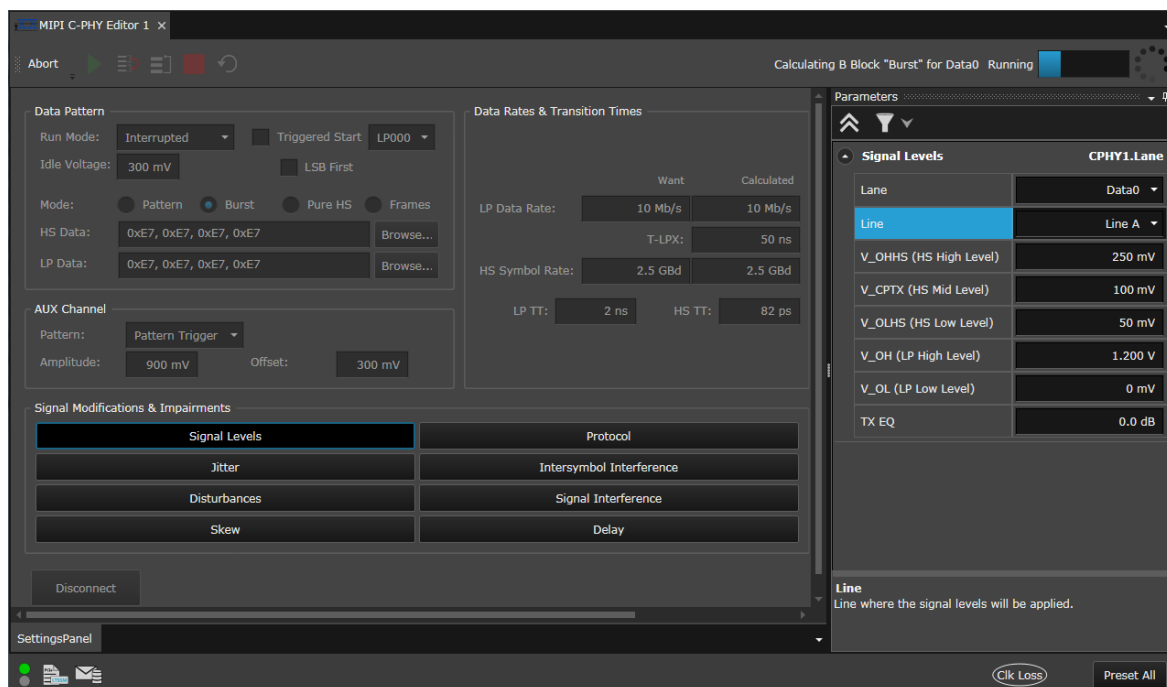
In the following image, the values for “V_OHHS”, “V_CPTX” and “V_OLHS” for Lane “Data0” and “Line A” have been modified to 250mV, 100mV and 50mV, respectively.

Parameters

Signal Levels CPHY1.Lane

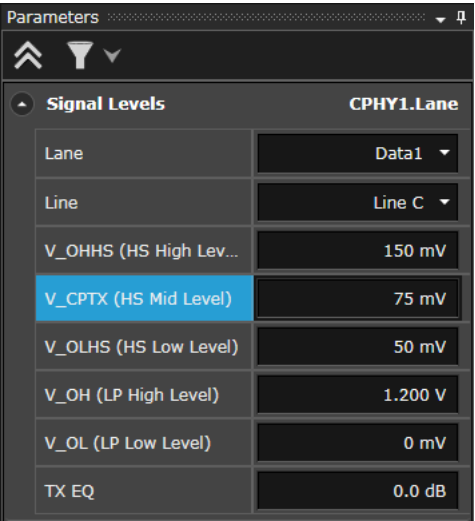
Lane	Data0
Line	Line A
V_OHHS (HS High Level)	250 mV
V_CPTX (HS Mid Level)	100 mV
V_OLHS (HS Low Level)	50 mV
V_OH (LP High Level)	1.200 V
V_OL (LP Low Level)	0 mV
TX EQ	0.0 dB

- 5 You may either click **Apply** to perform a waveform calculation or skip to step 8 if you wish to continue working in the waveform generation interface of the same MIPI C-PHY Editor plug-in instance.

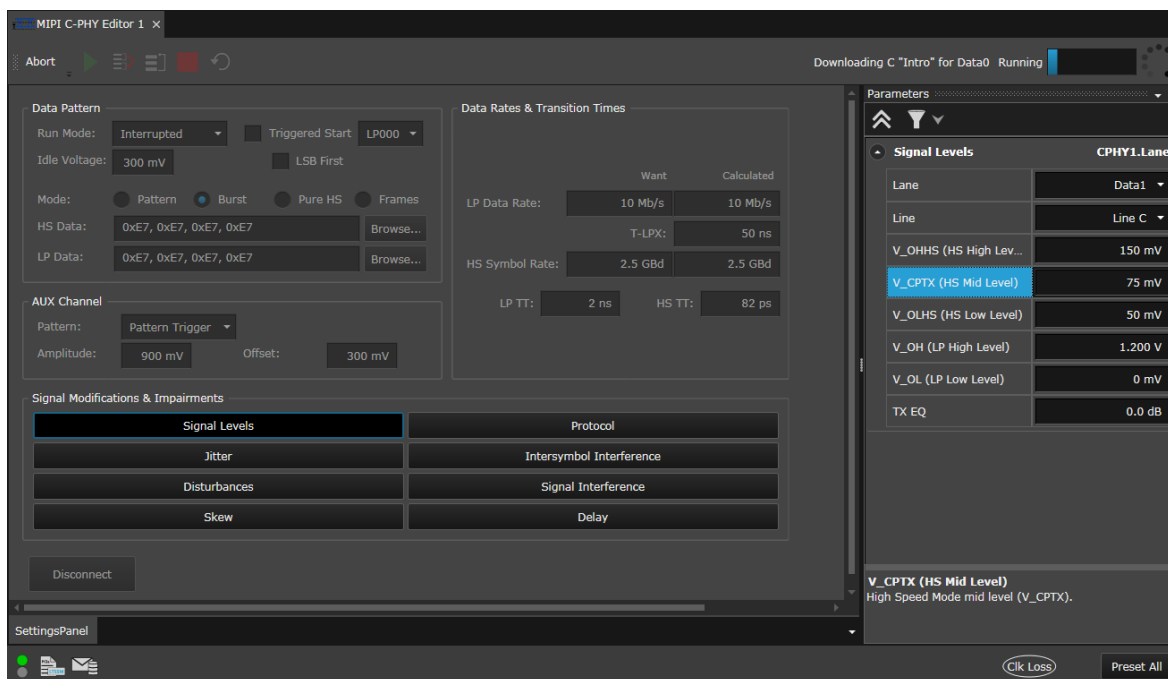


- 6 After modifying the configuration of the signal levels for the desired lanes and lines, you may optionally click **Disconnect** to return to the **Connection Setup** interface to make any further modifications. Otherwise, skip to step 7.
- 7 Repeat step 1 to connect the MIPI C-PHY Editor plug-in to 3-Lanes of M8195A AWG in the same instance.
- 8 Repeat step 2 to 4 to modify the Lane, Line and desired signal levels, such that the options you select are different from the previous configuration.

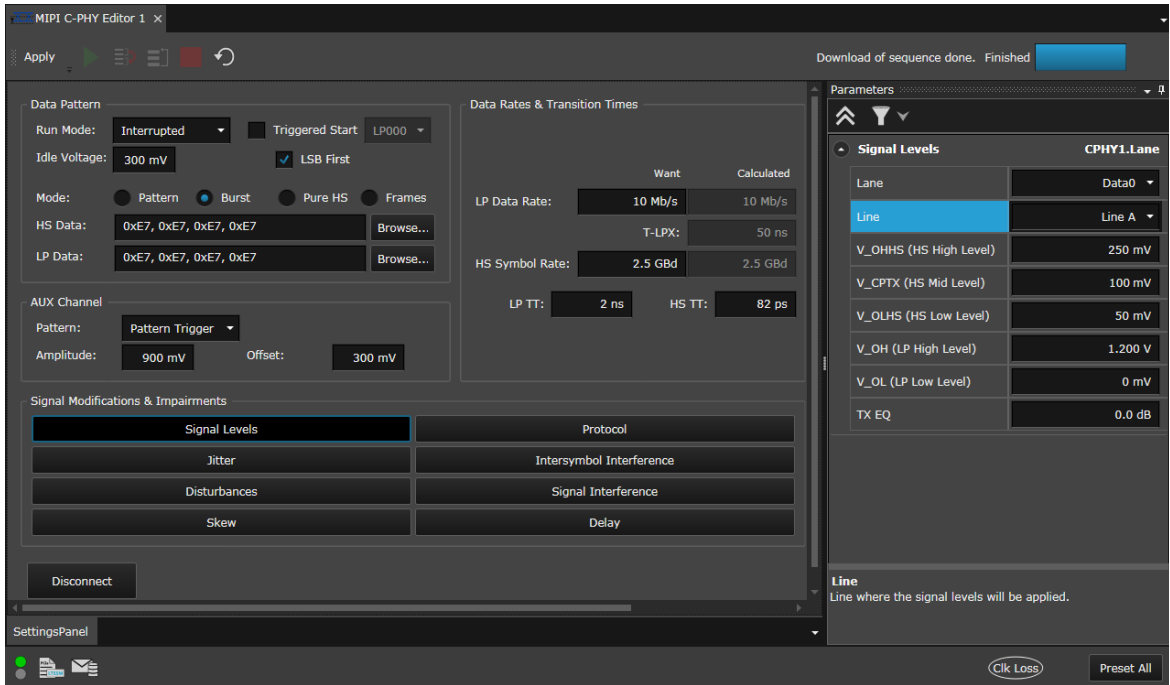
For example, in the following image, the values for "V_OHHS", "V_CPTX" and "V_OLHS" for Lane "Data1" and "Line C" have been modified to 150mV, 75mV and 50mV, respectively.



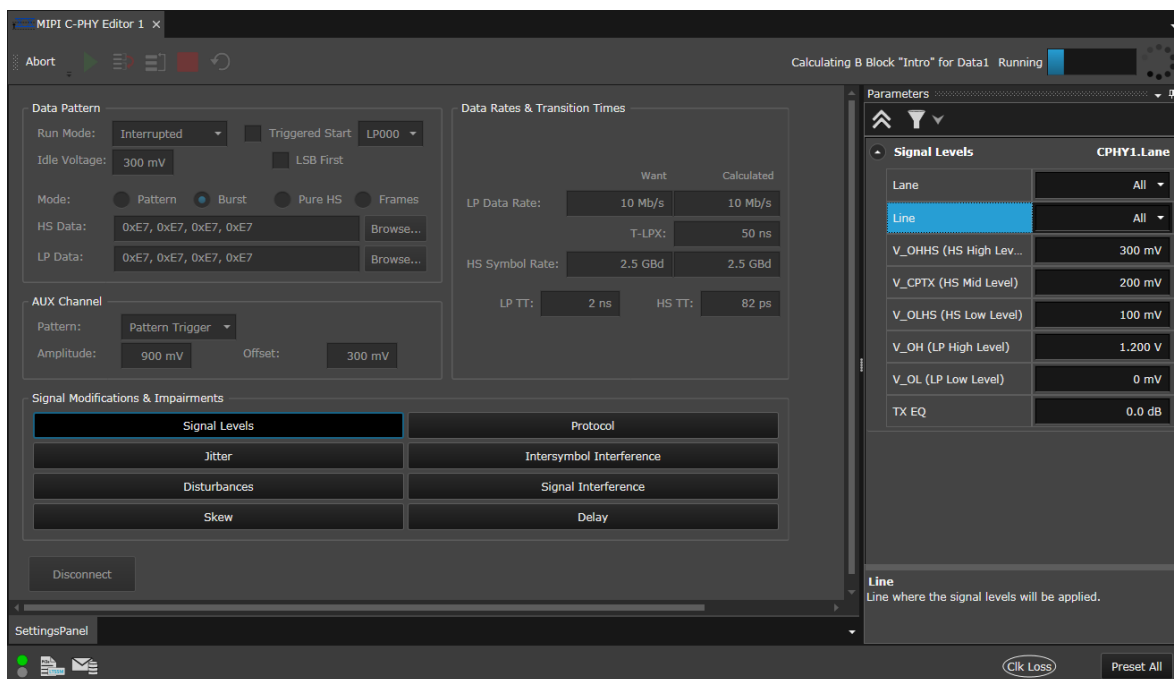
- 9 Click **Apply** to perform a waveform recalculation for the second configuration.



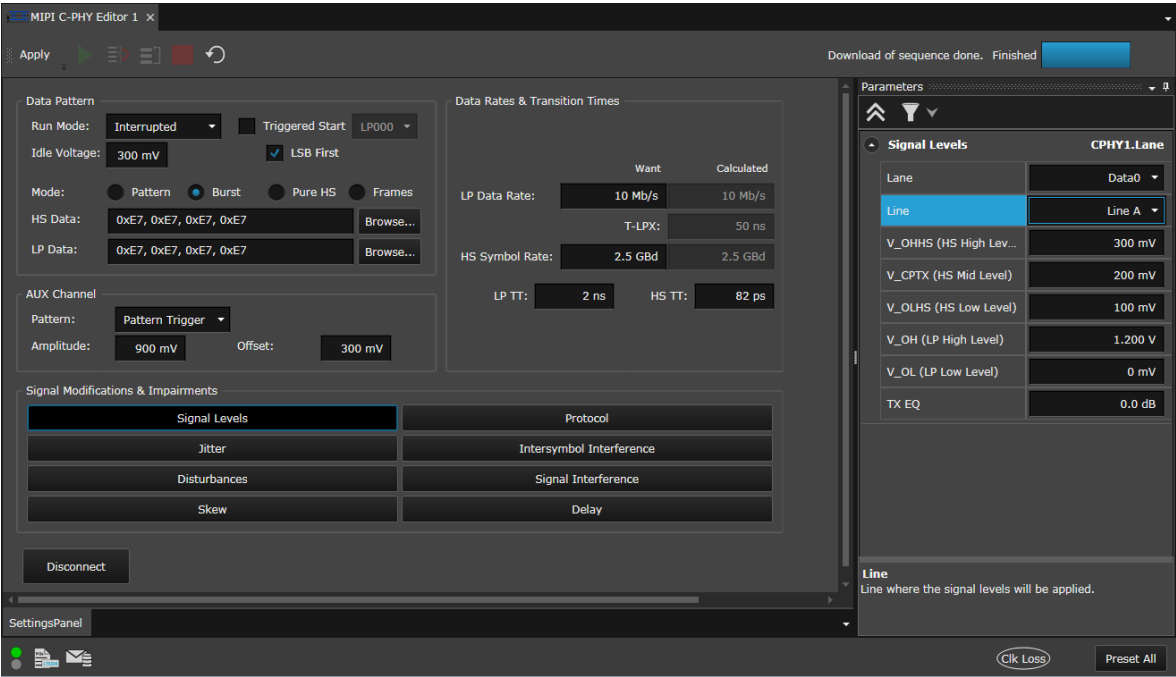
- 10 Once the waveform generation is complete, revert to the Parameter configuration for Signal Levels as defined in steps 2 to 5. In the example shown above, it is Lane Data0 and Line A. Notice that this new modification in the MIPI C-PHY Editor plug-in does not allow any change in the Signal Level values that you had customized for the first configuration, unless you change the values for another setup in the same instance.



11 If, at any stage after modifying the Signal Level values for any configuration, you may select **All** for either **Lane**, **Line** or both options and perform a waveform recalculation.

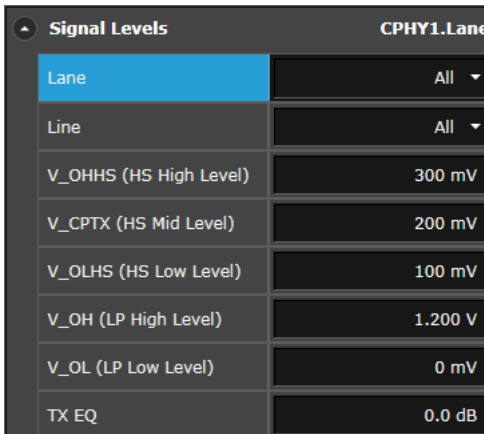


- 12 Notice that all combinations of Data lanes and lines are reset to the values that had been set for the option All in Lane and Line. For example, switch to Lane Data0 and Line A as shown in the image below.



13 You may repeat the steps mentioned previously to customize any voltage level under Signal Levels.

The following figure shows the parameters for signal level group:



Signal Levels		CPHY1.Lane
Lane		All ▼
Line		All ▼
V_OHHS (HS High Level)		300 mV
V_CPTX (HS Mid Level)		200 mV
V_OLHS (HS Low Level)		100 mV
V_OH (LP High Level)		1.200 V
V_OL (LP Low Level)		0 mV
TX EQ		0.0 dB

Figure 40 Signal Levels Group

Signal Levels group parameters:

- Lane: Selection of the lane where the voltage levels are to be applied.
- Line: Selection of the line where the voltage levels are to be applied.
- V_OHHS (HS High Level): High Speed Mode high voltage level.
- V_CPTX (HS Mid Level): High Speed Mode mid voltage level.
- V_OLHS (HS Low Level): High Speed Mode low voltage level
- V_OH (LP High Level): Low Power Mode high voltage level.
- V_OL (LP Low Level): Low Power Mode low voltage level.
- TX EQ: Displays the Transmitter Equalization value in dB. The TX EQ values range from 0 to -10 dB.

Protocol Group

To change the protocol settings, click the “Protocol” button in the “Signal Modifications & Impairments” area such that the corresponding settings are visible on the “Parameters” panel. Other than setting values for “Protocol” in the “Parameters” panel of the user interface, you may also run the [SCPI Commands for Protocol Group](#) on page 187.

The parameters provided by Protocol setting are shown in the following figure:

Protocol	CPHY1.Link
T3-PREPARE Duration	50.0 ns
T3-PREBEGIN	3333333
T3-PREBEGIN Multiplier	1
T3-PROGSEQ	
T3-PREEND	3333333
T3-SYNC	3444443
T3-POST	4444444
T3-POST Multiplier	1
TX-HS-EXIT Duration	200.0 ns
TX-WAKEUP Duration	1.0000 ms
TX-INIT Duration	100.00 us
Set to Default	

Figure 41 Protocol Group

Protocol group parameter:

- T3-PREPARE Duration: Sets the time that the transmitter drives a LP-000 state immediately before the start of high speed transmission.

- **T3-PREBEGIN:** Sets the T3-PREBEGIN pattern in high-speed MIPI C-PHY pattern format before the beginning of the HS data transmission. The pattern should be a multiple of 7 UI from a minimum of 7UI to a maximum of 448 UI. By default, an HS pattern of 7UI length is defined. To define a pattern of more than 7UI, use the parameter “T3-PREBEGIN Multiplier” to define a multiplier.
- **T3-PREBEGIN Multiplier:** Define a multiplier value between 1 and 64 to set the required HS pattern length from 7UI to 448UI. By default, the multiplier is set to 1, which indicates that a T3-PREBEGIN pattern length of (1 x 7UI) 7UI is defined. You may modify the multiplier value up to 64 to achieve the maximum T3-PREBEGIN pattern length of up to 448UI.
- **T3-PROGSEQ:** Sets the optional T3-PROGSEQ pattern in high-speed MIPI C-PHY pattern format.
- **T3-PREEND:** Sets the T3-PREEND pattern in high-speed MIPI C-PHY pattern format. By default, this pattern contains a total of 7 symbols of type 3.
- **T3-SYNC:** Sets the T3-SYNC which is sent immediately before starting high-speed transmission. By default, the pattern is 3444443.
- **T3 POST:** Sets the T3-POST pattern which is sent immediately after starting high-speed transmission. The pattern should be a multiple of 7 UI from a minimum of 7UI to 224UI. By default, an HS pattern of 7UI length is defined. To define a pattern of more than 7UI, use the parameter “T3-POST Multiplier” to define a multiplier.
- **T3-POST Multiplier:** Define a multiplier value between 1 and 32 to set the required T3-POST pattern length from 7UI to 224UI. By default, the multiplier is set to 1, which indicates that a T3-POST pattern length of (1 x 7UI) 7UI is defined. You may modify the multiplier value up to 32 to achieve the maximum T3-POST pattern length of up to 224UI.
- **TX-HS-EXIT Duration:** Sets the length of LP-111 state following a high-speed burst.
- **TX-WAKEUP Duration:** Time that the transmitter drives a Mark-1 state prior to a Stop state in order to initiate an exit from ULPS.
- **TX-INIT Duration:** Time that the transmitter drives a Stop State (LP-111).
- **Set to Default:** Allows to reset all the Protocol settings to the default value.

Jitter Group

To change the jitter settings, click the “Jitter” button in the “Signal Modifications & Impairments” area such that the corresponding settings are visible on the “Parameters” panel. Other than setting values for “Jitter” in the “Parameters” panel of the user interface, you may also run the **SCPI Commands for Jitter Group** on page 182.

The Jitter group contains the parameters to generate Sinusoidal Jitter and Bounded Uncorrelated Jitter (Random Jitter). For both types of jitter, it is possible to select the target lane were it needs to be generated.

The following figure shows parameters for jitter group:

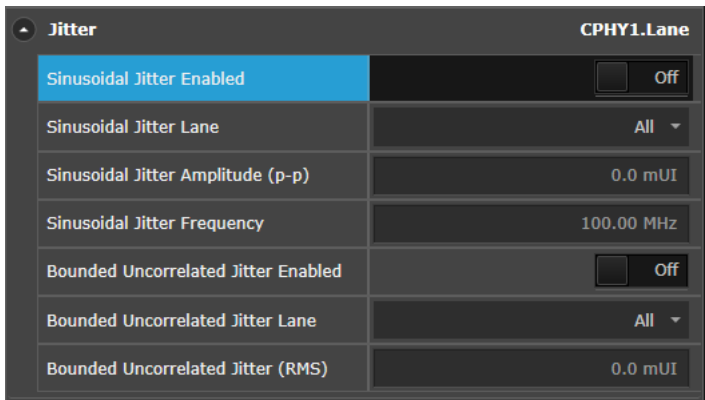


Figure 42 Jitter Group

Jitter Group Parameters:

- Sinusoidal Jitter Enabled – Enables or disables the Sinusoidal Jitter generation on all lanes.
- Sinusoidal Jitter Lane– Lane selection where the Sinusoidal Jitter is applied. It supports Sinusoidal Jitter on the clock lane, on all data lanes or alternatively on a specific data lane.
- Sinusoidal Jitter Amplitude (p-p) – Amplitude of the Sinusoidal Jitter in peak-to-peak.
- Sinusoidal Jitter Frequency– Frequency of the Sinusoidal Jitter.
- Bounded Uncorrelated Jitter Enabled – Enables or disables the Bounded Uncorrelated Jitter generation on all lanes. It emulates random jitter.

- Bounded Uncorrelated Jitter Lane– Lane selection where the Bounded Uncorrelated Jitter is applied. It supports Bounded Uncorrelated Jitter on the clock lane, on all data lanes or alternatively on a specific data lane.
- Bounded Uncorrelated Jitter (RMS) – Root Mean Square (RMS) Amplitude of the Bounded Uncorrelated Jitter.

Intersymbol Interference Group

To change the intersymbol interference settings, click the “Intersymbol Interference” button in the “Signal Modifications & Impairments” area such that the corresponding settings are visible on the “Parameters” panel. Other than setting values for “Intersymbol Interference” in the “Parameters” panel of the user interface, you may also run the **SCPI Commands for InterSymbol Interference Group** on page 178.

Intersymbol Interference can also be generated by the MIPI C-PHY Editor plug-in by using an S-Parameter file that modulates the targeted physical channel. After selecting an S-Parameter file some additional changes are allowed.

The following figure shows parameters for intersymbol interference group:

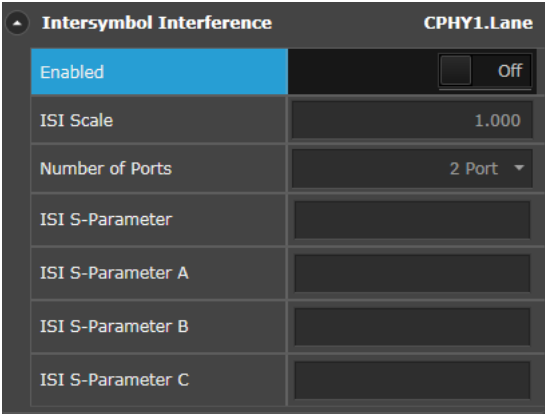


Figure 43 Intersymbol Interference Group

Intersymbol Interference Group Parameters:

- Enabled: Enables or disables the Intersymbol Interference.
- ISI Scale: Sets the intersymbol interference factor.
- Number of Ports: Allows to load a S-parameter file of either 2-Port or 6 Ports. The Editor needs just one parameter to modify ISI because in a s-parameter file a model of a fixed trace is given. Internally it is converted into a time domain filter and applied to the waveform. By this factor the time domain filter is stretched or compressed in time domain, which emulates a longer or shorter trace than the original. It can be used to tune the resulting ISI.

- ISI S-Parameter File: S-Parameter file that emulates the target ISI channel. It is considered only when the “6-Port” is selected from the “Number of Ports”.
- ISI S-Parameter A / B / C: These are ISI 2 Port S-parameter file. They are considered only when the “2-Port” is selected from the “Number of Ports”.

Disturbances Group

To change the disturbances settings, click the “Disturbances” button in the “Signal Modifications & Impairments” area such that the corresponding settings are visible on the “Parameters” panel. Other than setting values for “Disturbances” in the “Parameters” panel of the user interface, you may also run the [SCPI Commands for Disturbances Group](#) on page 172.

The Disturbances Group parameters allow you to insert the e_{SPIKE} in the middle of a symbol with editable area and input voltages of logic 1 (or 0). Also, these parameters allow you to modify the LP Pulse Width and the pattern disturbances on the high-speed entry and exit sequences of a burst. To clearly understand the functionality of the various parameters under the Disturbances group, a conceptual understanding of e-Spikes is required.

e_{SPIKE}

Spikes are defined as transient signals, clearly distinguishable from the background activity with a pointed peak.

The e_{Spike} is described as an LP (low power) receiver’s ability to reject short-term glitches, i.e., narrow pulses with voltage levels outside of the current Logic state, but that must not change the receiver state, as their widths are sufficiently shorter than the nominal T_{LPX} interval.

The LP receiver should reject any e_{Spike} up to the limit defined in the specification.

Activating e_{Spike} generation introduces a single voltage spike per LP state. It is located in the center of the LP-symbol and has an area as defined in the corresponding property. Area calculation of e_{Spike} also depends on the receiver high/low level thresholds called “Logic 1 Input Voltage” and “Logic 0 Input Voltage” respectively. When having e_{Spike} in logic 1 states the glitch will go from LP high level to LP low level. As area calculation of the e_{Spike} in a high state begins below the selected “Logic 1 Input Voltage” the width of the e_{Spike} increases when increasing this value as it will also be the case when increasing the area directly. “Logic 0 Input Voltage” has no effect on high level e_{Spike} . Behavior of the e_{Spike} in low levels is similar to the high level case with the difference that “Logic 0 Input Voltage” is then a width defining parameter. During LP $\leftrightarrow$ HS transitions “TX-HS-Request Duration”, “TX-HS-Prepare Duration” and “TX-HS-Exit Duration” are treated as a single LP symbol and therefore only contain a single e_{Spike} each.

The conformance limit for e_{Spike} is defined in units of Volts times picoseconds, (V*ps), which allows for multiple potential test cases (high voltage/short-duration, vs. low voltage/long duration).

The following diagram shows e_{Spike} glitches.

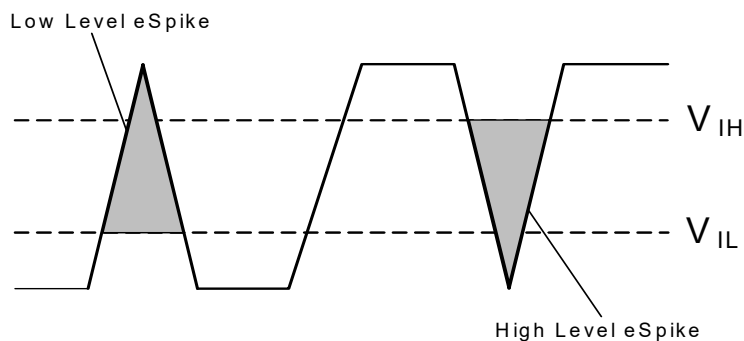


Figure 44 Input Glitch Rejection of Low-Power Receivers

The following table shows the LP Receiver DC specifications:

Table 10 LP Receiver DC Specifications

Parameter	Description	Min	Nom	Max	Units	Notes
V_{IH}	Logic 1 input voltage	740			mV	
V_{IL}	Logic 0 input voltage, not in ULP State			550	mV	
$V_{IL-ULPS}$	Logic 0 input voltage, ULP State			300	mV	

The following table shows the LP Receiver AC specifications:

Table 11 LP Receiver AC Specifications

Parameter	Description	Min	Nom	Max	Units	Notes
e_{Spike}	Input pulse rejection			300	V.ps	1, 2, 3
$T_{\text{MIN-RX}}$	Minimum pulse width response	20			ns	4

The Low power receivers must have an ability to reject short-term glitches (any input signal smaller than e_{SPIKE}), which are the narrow pulses with voltage levels outside of the current logic state. However, the receiver state does not change as their widths are sufficiently shorter than the nominal T_{LPX} interval as shown in [Figure 45](#).

Activating e_{SPIKE} generation introduces a single voltage spike per LP state. It is located in the center of the LP-state. Area calculation of e_{SPIKE} depends on the receiver high/low level thresholds called “Logic 1 Input Voltage” and “Logic 0 Input Voltage” respectively.

The following figure shows e_{SPIKE} glitches where V_{IL} and V_{IH} represent e_{SPIKE} Logic 0 Input Voltage and e_{SPIKE} Logic 1 Input Voltage, respectively.

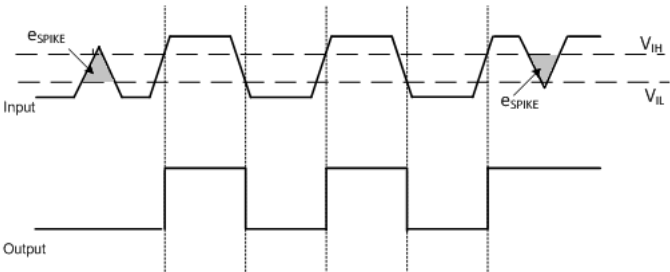


Figure 45 Input Glitch Rejection of Low-Power Receiver

The following figure shows parameters for Disturbances group:

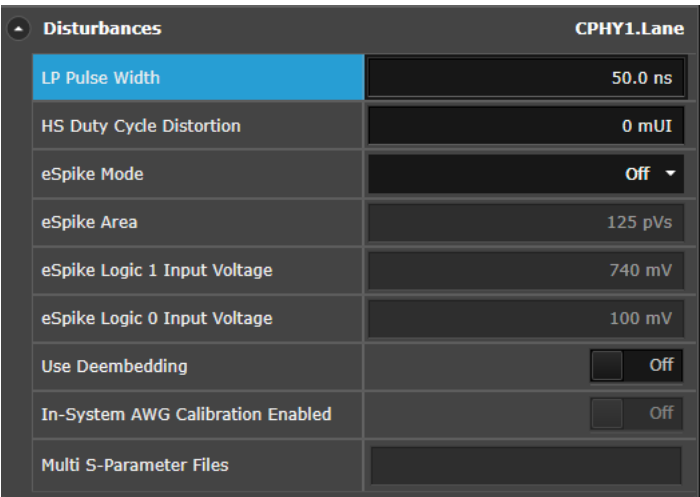


Figure 46 Disturbances Group

Disturbances Group Parameters:

- LP Pulse Width – Allows you to change the pulse width at the Low Power mode within the defined ranges. Changing the value of LP pulse width does not affect the data rate. The data rate remains constant because increasing the pulse width by this option applies to LP High state, which changes the duty cycle and simultaneously, decreases the width of LP low state.
- HS Duty Cycle Distortion – Allows to change the Duty Cycle of the HS signal.
- eSpike Mode – Allows you to enable or disable eSPIKE generation on the high levels or on the low levels of the pulse. The eSPIKE mode is used to test the ability of LP receiver to reject any input signal smaller than the eSPIKE.
- eSpike Area – Defines the area covered by the eSPIKE. It can be calculated as following:
 - Area calculation – With eSPIKE in High Levels, the glitch goes from LP high level to LP low level, the area calculation begins below the selected “Logic 1 Input Voltage to the peak of the glitch”. The width of the eSPIKE increases when you increase the value of Logic 1 Input Voltage, which simultaneously increases the area. “Logic 0 Input Voltage” has no effect on high level eSPIKE. Similarly, you can calculate the area of eSPIKE in Low Levels.
- eSpike Logic 1 Input Voltage – Specifies the lower receiver detection threshold level for a logic 1. It is the voltage level just below the LP high level voltage and is represented by V_{IH} (refer to [Figure 45](#)).
- eSpike Logical 0 Input Voltage – Specifies the upper receiver detection threshold level for a logic 0. It is the voltage level just above the LP low level voltage and is represented by V_{IL} (refer to [Figure 45](#)).
 - Some examples are (refer to [Figure 47](#)):
 - If you select eSpike Logic 1 Input Voltage = 740 mV and eSpike Logic 0 Input Voltage = 300 mV, eSpike Area = 350 pVs
 - If you select eSpike Logic 1 Input Voltage = 740 mV and eSpike Logic 0 Input Voltage = 100 mV, eSpike Area = 500 pVs

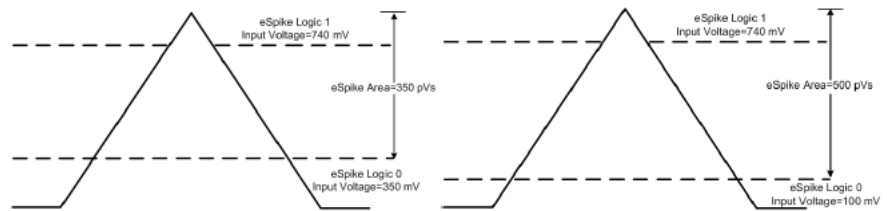


Figure 47 Examples of eSPIKE

- Use Deembedding – If set to 'On', this parameter applies a software filter to compensate impairments of the AWG output amplifier, thereby, improving the MIPI C-PHY signal quality. It also enables the parameter to enable/disable the In-System AWG Calibration and to define multi S-Parameter files.
- In-System AWG Calibration Enabled – If set to 'On', this parameter enables the In-System AWG Calibration data for de-embedding. While the process of a normal de-embedding achieves de-embedding of the output amplifier only, using In-System Calibration achieves de-embedding of cables and probes as well (that is, the entire signal path), if applied. For more information regarding how to perform In-System AWG Calibration, see [Performing In-System AWG Calibration with Keysight IQ Tools](#) on page 128.
- Multi S-Parameter Files – Define the s-parameter files to be de-embedded from the calculated waveform.

To select one or more S-Parameter files, click this field and navigate to *C:\ProgramData\BitfEye\ValiFrame\SPParameter\CPhy*. Select the required files and click Open. The selected file locations appear on the field and the de-embedding attributes are loaded.

Signal Interference Group

To change the signal interference settings, click the “Signal Interference” button in the “Signal Modifications & Impairments” area such that the corresponding settings are visible on the “Parameters” panel. Other than setting values for “Signal Interference” in the “Parameters” panel of the user interface, you may also run the [SCPI Commands for Signal Interference Group](#) on page 194.

The Signal Interference settings indicate the lanes, lines, mode of the interference that is to be introduced along with amplitude and frequency, of the interfering signal.

The different values of interferences in amplitude and frequency for different lanes and lines can be applied in the same manner as described in section [Setting different signal levels for lanes and lines](#) on page 98.

By default, the Signal Interference settings is disabled and you must enable it only when the waveform generation must be performed along with an available interfering signal.

The Signal Interference group allows to generate signal interference that is super-imposed on the MIPI C-PHY waveform. This is particular useful to emulate external interferences on the MIPI C-PHY devices.

The following figure shows parameters for signal interference group:

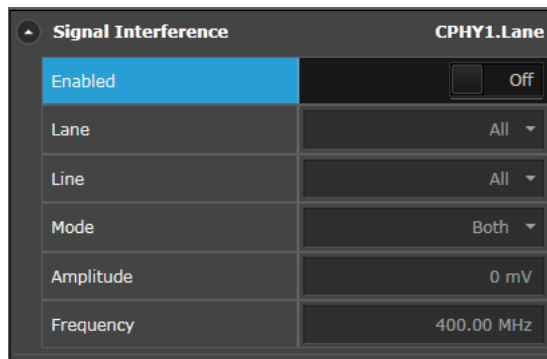


Figure 48 Signal Interference Group

Signal Interference Group Parameters:

- Enabled – Enables or Disables the sinusoidal Signal Interference generation.
- Lane – Represents the target lane where the Signal Interference are added. It allows the interference on the all data lanes simultaneously or on a specific data lane.
- Line – Represents the target line where the Signal Interference will be added. It allows interference on all line simultaneously or on a specific line.
- Mode – Allows to add interference on the High Speed signal, Low Speed signal or on both.
- Amplitude– Amplitude of the generated interference. Interference Amplitude is always measured 0Vpk terminated, regardless of whether it is inserted in HS mode or LP mode.
- Frequency– Frequency of the generated interference.

Skew Group

To change the skew settings, click the “Skew” button in the “Signal Modifications & Impairments” area such that the corresponding settings are visible on the “Parameters” panel. Other than setting values for “Skew” in the “Parameters” panel of the user interface, you may also run the **SCPI Commands for Skew Group** on page 201.

The Skew Group parameter gives the ability to change the skew in seconds (usually pico seconds) of a specific channel on the AWGs. This feature is useful for de-skewing the influences of cables. Depending on the configured setup, some of the parameters do not apply and they are disabled.

The following figure shows parameters for skew group:

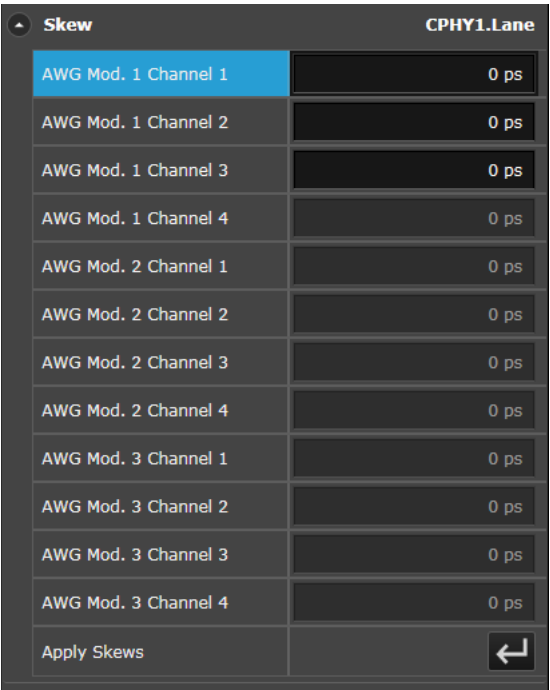


Figure 49 Skew Group

NOTE

The “Apply Skews” buttons is bounded with the “Apply Delays”. So when the “Apply Skews” is pressed the delays parameters are also applied. Similarly, when the “Apply Delays” is pressed the skews parameters are also applied.

Delay Group

To change the delay settings, click the “Delay” button in the “Signal Modifications & Impairments” area such that the corresponding settings are visible on the “Parameters” panel. Other than setting values for “Delay” in the “Parameters” panel of the user interface, you may also run the **SCPI Commands for Delay Group** on page 168.

The Delay Group allows you to change the delays among the lanes. If the Enable Intra-Line Delay is activated it is also possible to apply delays among the lines. In order to apply the delays, click the “Apply Delays” button or alternatively click the “Apply” button of the toolbar. When the “Apply Delays” button is clicked the MIPI C-PHY Editor tries to use the hardware resources to generate the delays. If the hardware resources are not enough, the delays need to be embedded in the pattern and therefore the waveform needs to be recalculated.

NOTE

The “Apply Delays” buttons is bounded with the “Apply Skews”. So when the “Apply Delays” is pressed the skews parameters are also applied. Similarly, when the “Apply Skews” is pressed the delays parameters are also applied.

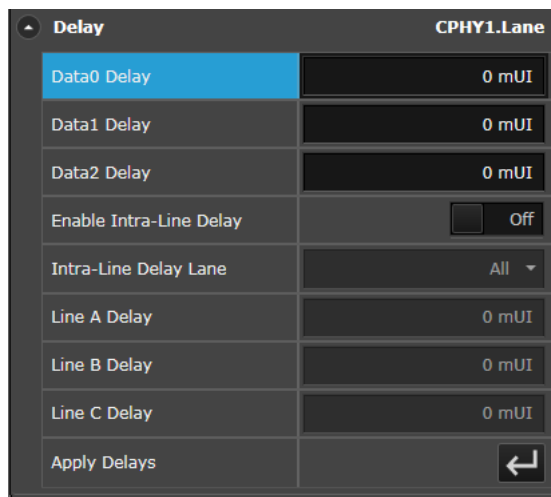


Figure 50 Delay Group

Performing In-System AWG Calibration with Keysight IQ Tools

To improve the quality of the MIPI C-PHY signal, you may enable the calibrated values obtained via the In-System Calibration of the connected AWGs. The In-System Calibration can be performed using the Keysight IQ Tools software, which you may download from www.keysight.com.

To perform In-System Calibration:

- 1 Launch the *Keysight IQ Tools* software.
- 2 In the **Configuration** area of the main window, click **Configure instrument connection**.

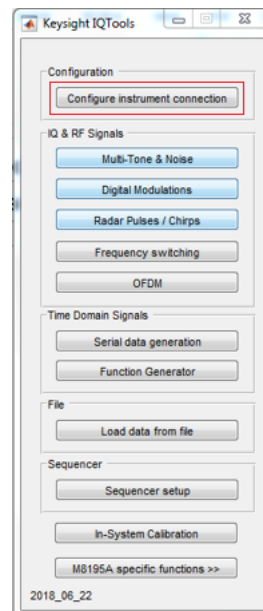


Figure 51 Main window of the IQ Tools software

- 3 In the **Arbitrary Waveform Generator Configuration** section of the **Instrument Configuration** window,
 - For 4-Channel mode:
 - a Select **Instrument model** as *M8195A*
 - b Select **Mode** as *4 ch, deep mem, 16 GSa/s*
 - c Select **Connection type** as *visa*
 - d In the **VISA Address** field, type the address of the M8195A module

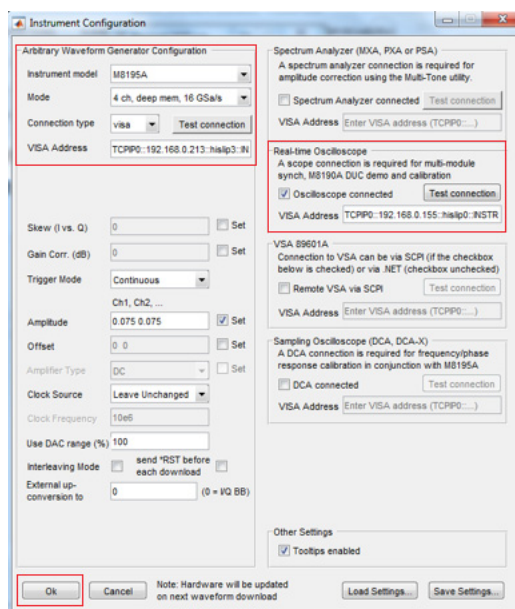


Figure 52 Settings on the Instrument Configuration window for 4-Channel

- For Dual Channel mode:
 - a Select **Instrument model** as *M8195A*
 - b Select **Mode** as *2 ch, deep mem, 32 GSa/s*
 - c Select **Connection type** as *visa*
 - d In the **VISA Address** field, type the address of the M8195A module

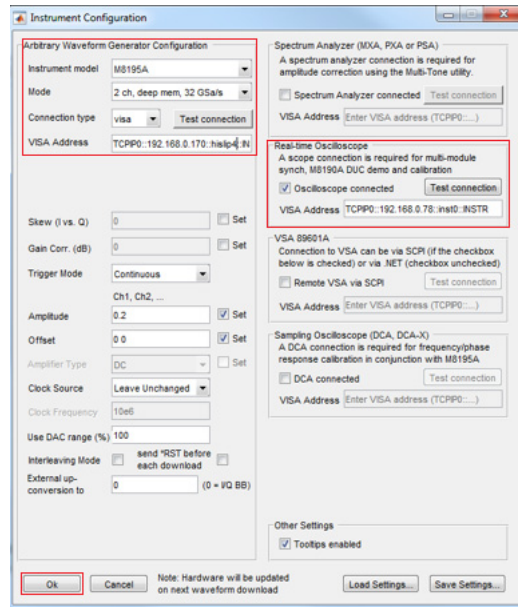


Figure 53 Settings on the Instrument Configuration window for Dual Channel

- 4 In the **Real-time Oscilloscope** section of the **Instrument Configuration** window,
 - a Select the **Oscilloscope connected** check box, if not checked already
 - b In the **VISA Address** field, type the address of the Oscilloscope.
- 5 Click **OK** on the Instrument Configuration window.

- 6 In the **Time Domain Signals** area of the main window, click **Serial data generation**.

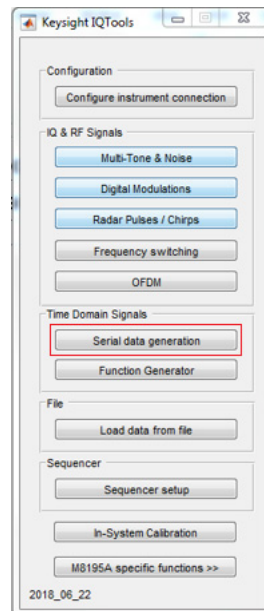


Figure 54 Accessing Serial data generation option

- 7 In the **Serial Data** window that appears, click **Show Correction**.

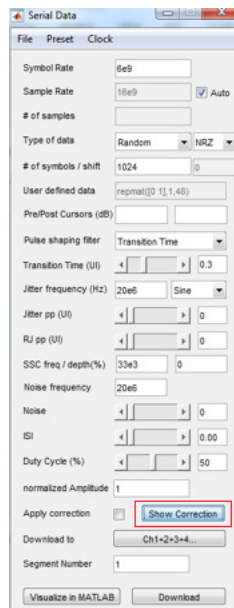


Figure 55 Accessing the Correction Management window

- 8 From the **Correction Management** window that appears, click **In-System Calibration**.

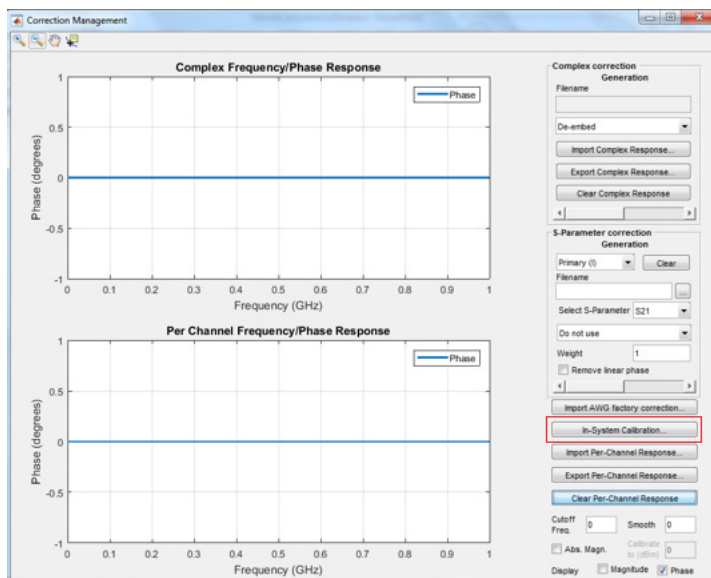


Figure 56 Selecting In-System Calibration option

- 9 On the **Frequency/Phase response calibration** window that appears,
 - For 4-Channel mode
 - a Ensure that the physical channels of the AWG and the Oscilloscope are connected as per the configuration shown in the **Channel Mapping** area (AWG Ch1 to Oscilloscope Ch1 and so on).

NOTE

On the AWG complement outputs, ensure that terminations are connected.

- b In the **Settings** area,
 - **Sample Rate** value remains as-is
 - If a Low-Pass Filter (LPF) is connected, the **Max. tone frequency** value should not exceed 8e9
- c Click **Run** to begin calibration measurements.

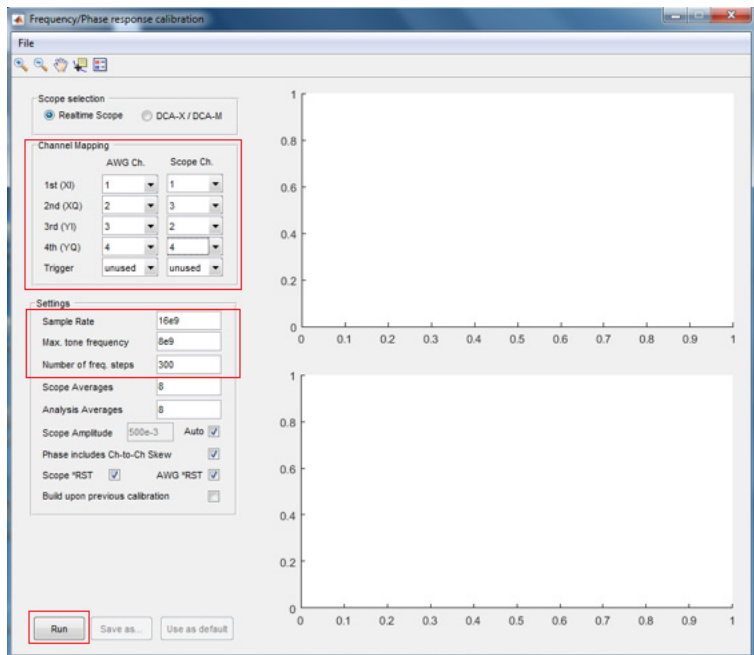


Figure 57 Validating settings to perform In-System Calibration

- d If error messages are displayed,
 - Validate the AWG-Oscilloscope Channel Mapping
 - Reduce the Max. tone frequency value and try again
- e If one or more outliers are found in the measurement, repeat the measurement a few more times.
- f Once measurements are performed, overwrite the results.

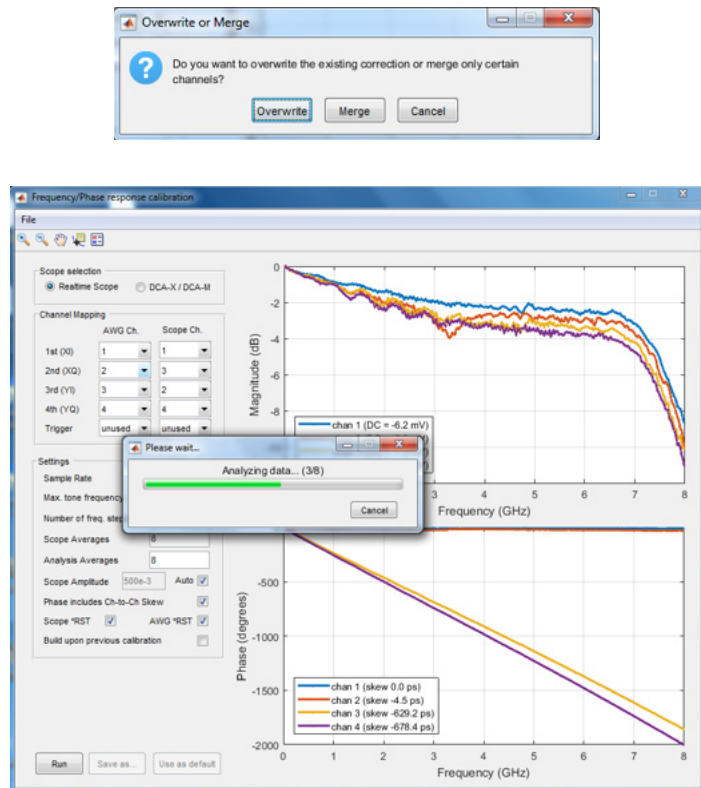


Figure 58 Viewing In-System Calibration results

- For Dual-Channel mode
 - a Ensure that the physical channels of the AWG and the Oscilloscope are connected as per the configuration shown in the **Channel Mapping** area (AWG Ch1 to Oscilloscope Ch1 and so on).

NOTE

On the AWG complement outputs, ensure that terminations are connected.

- b In the **Settings** area,
 - **Sample Rate** value remains as-is
 - No Low-Pass Filter (LPF) is connected, therefore, the **Max. tone frequency** value should be $16e9$
- c Click **Run** to begin calibration measurements.

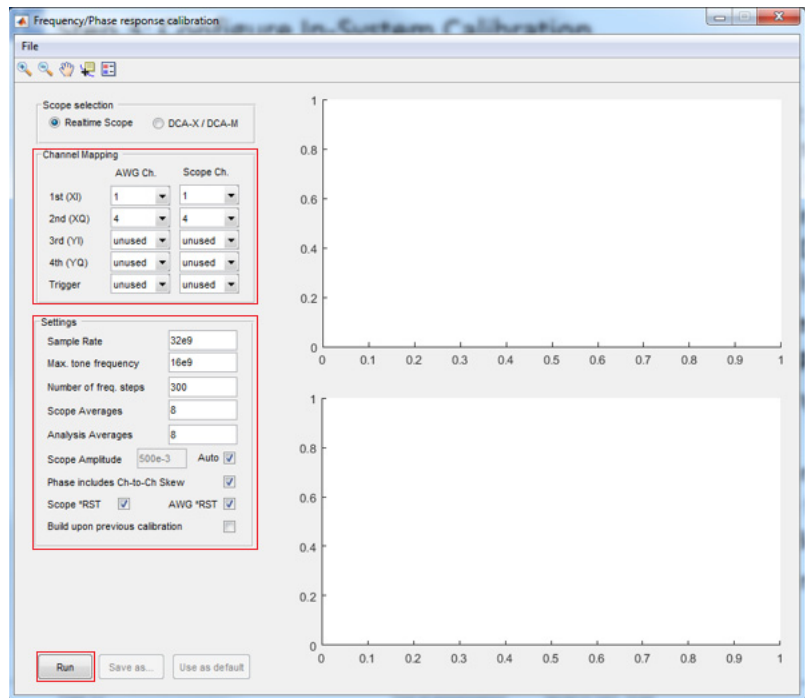


Figure 59 Validating settings to perform In-System Calibration

- d If error messages are displayed,
 - Validate the AWG-Oscilloscope Channel Mapping
 - Reduce the Max. tone frequency value and try again
- e If one or more outliers are found in the measurement, repeat the measurement a few more times.
- f Once measurements are performed, overwrite the results.

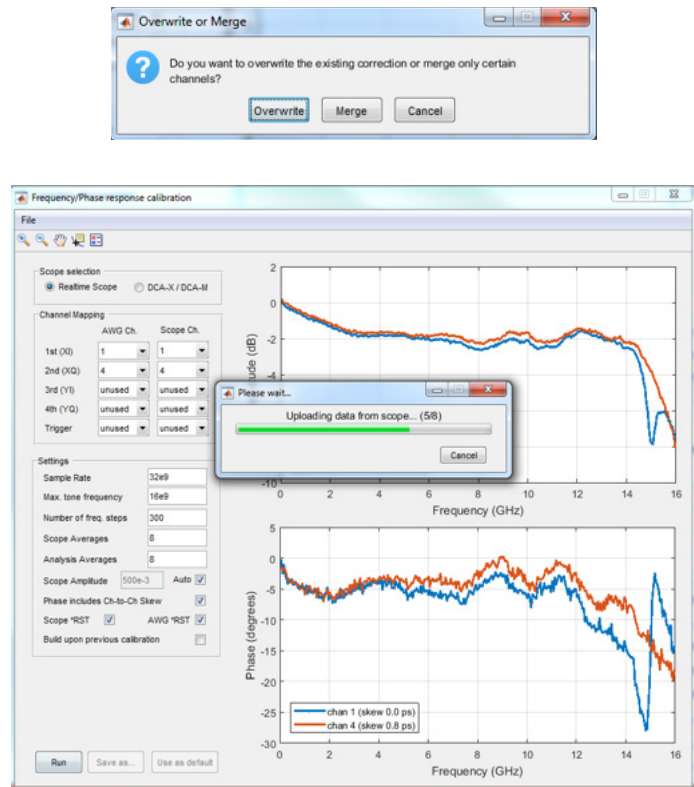


Figure 60 Viewing In-System Calibration results

- 10 To view the updated results, return to the **Correction Management** window.
- For 4-Channel mode

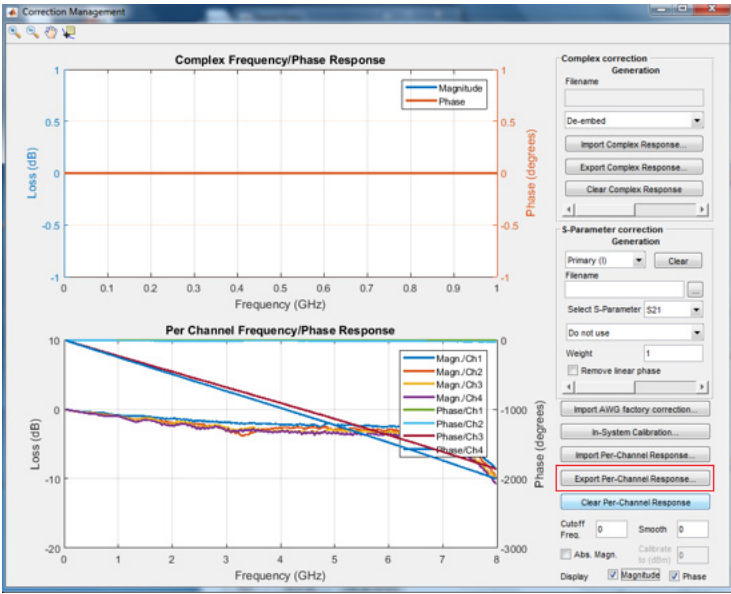


Figure 61 Exporting the Per-Channel Response

- For Dual-Channel mode

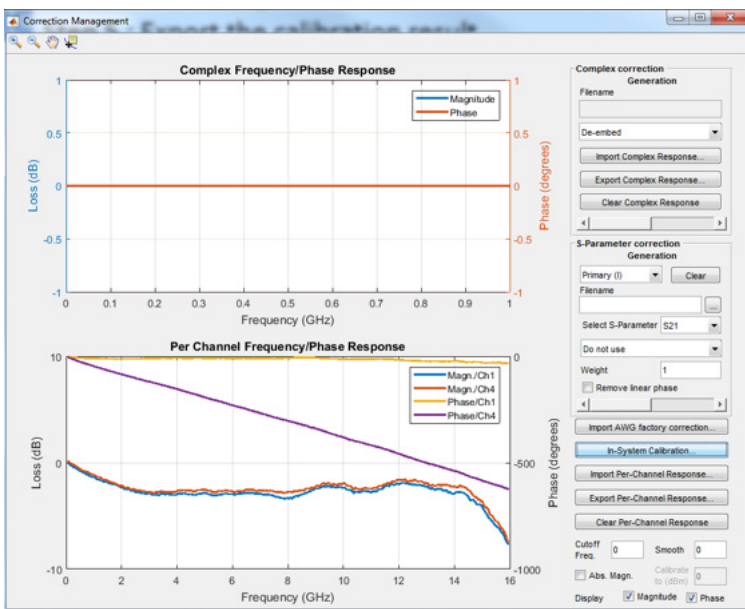


Figure 62 Exporting the Per-Channel Response

11 On the same window, click **Export Per-Channel Response...**

- 12 On the **Save Frequency Response As...** window,
- For 4-Channel mode

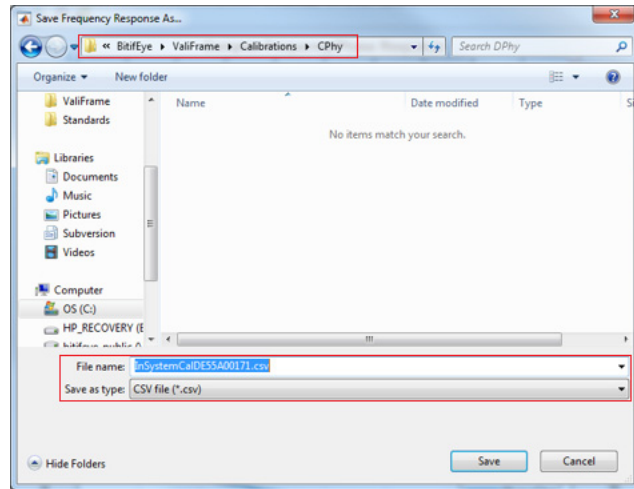


Figure 63 Saving the In-System Calibration values

- In the **Save as type:** drop-down field, select *CSV file (*.csv)*
- Navigate to the location *C:\ProgramData\BitfEye\ValiFrame\Calibrations\Cphy*
- Save the file with the following naming convention:
InSystemCal<AWG-Serial-Number>.csv

- For Dual-Channel mode

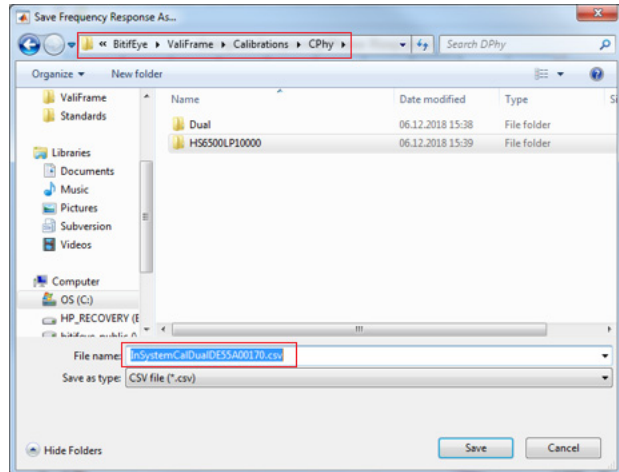


Figure 64 Saving the In-System Calibration values

- In the **Save as type:** drop-down field, select *CSV file (*.csv)*
- Navigate to the location *C:\ProgramData\BitifEye\ValiFrame\Calibrations\Cphy*
- Save the file with the following naming convention:
InSystemCalDual<AWG-Serial-Number>.csv

- d To obtain the AWG Serial Number, go to **About Keysight M8195A**.

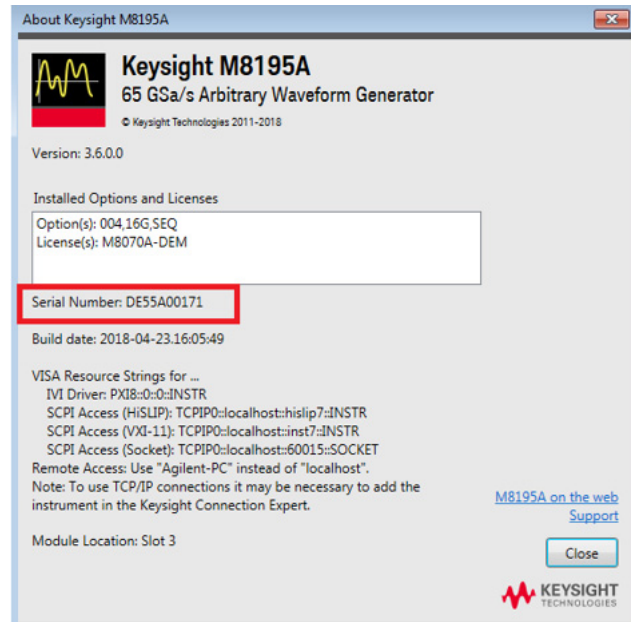


Figure 65 Identifying the AWG Serial Number

- 13 Repeat the entire process for each AWG that is physically connected.
- 14 Ensure to save the generated calibration output in the correct folder location with the CSV file format and the correct naming convention, as described earlier.

4 SCPI Programming

[SCPI Command Language](#) / 144

[SCPI Command Reference](#) / 150

SCPI Command Language

The M8020A is compatible with the standard language for remote control of instruments. Standard Commands for Programmable Instruments (SCPI) is the universal programming language for instrument control.

SCPI can be subdivided into the following command sets:

- SCPI Common Commands
- SCPI Instrument Control Commands
- IEEE 488.2 Mandatory Commands

For more details on command sets, refer to *M8020A Programming Guide*.

Data Types

The M8020A has the capability of receiving and returning data in the following formats:

STRING

A string of human-readable ASCII characters, either quoted or non-quoted.

NUMERIC

The M8020A handles the following numeric formats:

- <NR1>: Integer (0, 1, 2, - 1, etc.)
- <NR2>: Number with an embedded decimal point (0.1, 0.001, 3.3, etc.)
- <NR3>: Number with an embedded decimal point and exponent (1e33, 1.3e- 12, etc.)
- <NRf>: Represents <NR1>, <NR2>, and <NR3>
- Binary preceded by #b (#B010101, #b011111, etc.)
- Octal preceded by #q (#Q777111, #q7331777, etc.)
- Hex preceded by #h (#haff, #h8989ffff, etc.)

BOOLEAN

Boolean values can be sent to the M8020A as either ON | OFF or 0 | 1. The M8020A answers queries with 0 | 1.

Definite Length Arbitrary Block Data

Block data is used when a large quantity of related data is being returned. A definite length block is suitable for sending blocks of 8-bit binary information when the length is known beforehand. An indefinite length

block is suitable for sending blocks of 8-bit binary information when the length is not known beforehand or when computing the length beforehand is undesirable.

It has the following format:

```
#<Length of length><Length of data><data>
```

<Length of length> is a single integer that contains the number of digits in <Length of data>, which in turn contains the length of the data. For example, a 512-byte pattern would be defined as:

```
#3512<data>
```

Important Points about SCPI

There are a number of key areas to consider when using SCPI for the first time. These are as follows:

- Instrument Model
- Command Syntax
- Query Response
- Command Separators
- SCPI Command Structure

Instrument Model

SCPI guidelines require that the M8020A is compatible with an instrument model. This ensures that when using SCPI, functional compatibility is achieved between instruments that perform the same tasks. For example, if two different instruments have a programmable clock frequency setting, then both instruments would use the same SCPI commands to set their frequency. The instrument model is made up of a number of subsystems.

The sub-system defines a group of functions within a module and has a unique identifier under SCPI, which is called the Root Keyword.

Command Syntax

Commands may be up to twelve characters long. A short-form version is also available which has a preferred length of four characters or less. In this document the long-form and short-form versions are shown as a single word with the short-form being shown in upper-case letters.

For example, the long-form node command VOLTage has the short-form VOLT. Using the short form saves time when entering a program; however, using the long form makes a program more descriptive and easier to understand.

SCPI commands may be commands only, commands and queries, or queries only. A question mark at the end of a command indicates that it is a query. If the question mark appears in brackets ([?]), the command has a command and query form.

Query Responses

It is possible to cross-examine the individual settings and status of a device using query commands. Retrieving data is a two-stage operation.

The query command is sent from the controller using the OUTPUT statement and the data is read from the device using the ENTER statement. A typical example is the SCPI IEEE 488.2 Common Command *IDN? which queries the identity of a device.

NOTE

When sending strings to the instrument, either the double quote (") or the single quote (') may be used ('), the latter being more suited to PASCAL programs, which make use of a single quote; the former being more suited to use in BASIC programs, which use a double quote as a delimiter.

Command Separators

The SCPI command structure is hierarchical and is governed by commas, semicolons and colons:

- Commas are used to separate parameters in one command.
- Colons are used to separate levels.
- Semicolons are used to send more than one command to the instrument at a time.

It is possible to send several commands in one pass, as long as the commands all belong to the same node in the SCPI tree. The commands have to be separated by colons.

The following SCPI commands provide examples of this.

```
:PLUGin:CPHYplugin:PATtern:TRIGstart:ENable 'MIPI C-PHY Editor 1',  
OFF
```

These commands can also be sent as follows:

```
:PLUG:CPHY:PATT:TRIG:ENAB 'MIPI C-PHY Editor 1', OFF
```

SCPI Command Structure Example

The SCPI command structure can be best examined by means of an example. For example, the following command is used to enable or disable the Triggered Start:

```
:PLUGin:CPHYplugin:PATtern:TRIGstart:ENABLE 'MIPI C-PHY Editor 1',  
OFF
```

NOTE

Any optional commands are enclosed in square brackets [] and any optional characters are shown in lower case.

A colon indicates a change of level in the command hierarchy. Commands at the same level in the hierarchy may be included in the same command line, if separated by a semi-colon.

The bar symbol (|) indicates mutually exclusive commands.

To translate this syntax into a command line, follow the convention described above. Remember, however, that the command line can be created in several different ways. It can be created with or without optional keywords, and in a long or short form. The following example gives possible forms of the command line; all are acceptable:

In long form:

```
:PLUGin:CPHYplugin:PATtern:TRIGstart:ENABLE 'MIPI C-PHY Editor 1',  
OFF
```

In short form:

```
:PLUG:CPHY:PATT:TRIG:ENAB 'MIPI C-PHY Editor 1', OFF
```

The long form is the most descriptive form of programming commands in SCPI.

SCPI Editor

The **SCPI Editor** lists all SCPI that can be used to program M8020A and also provides a platform to execute them.

The following figure depicts the M8020A **SCPI Editor** user interface:

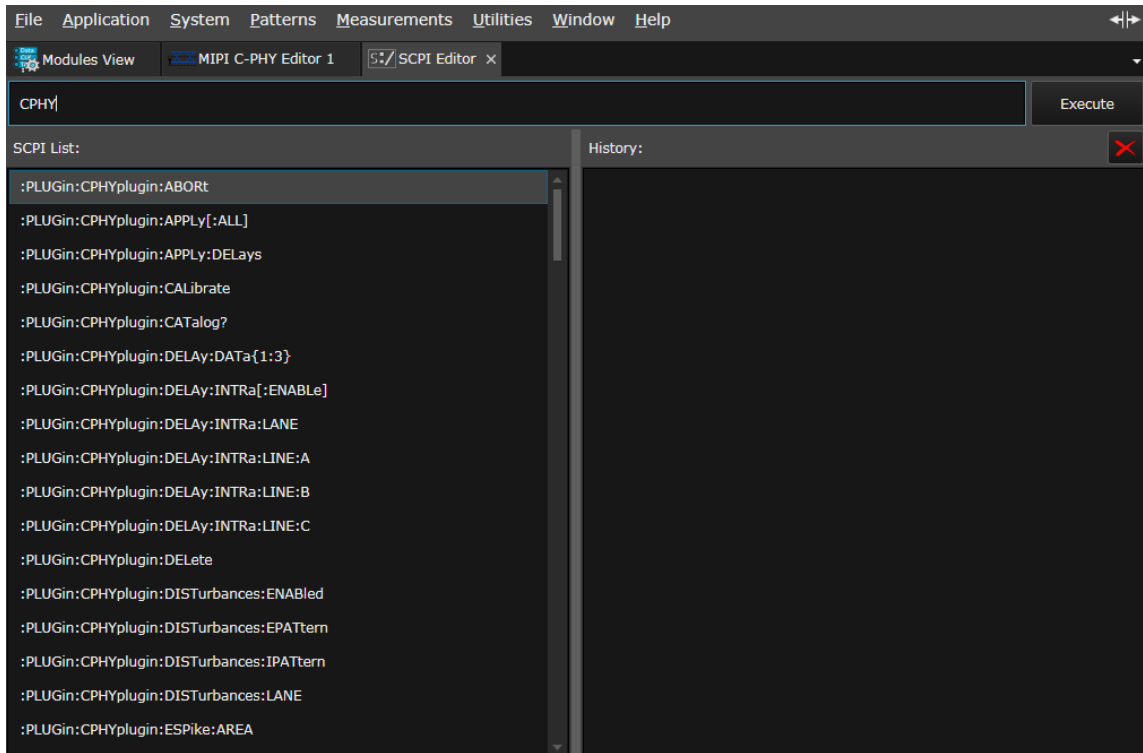


Figure 66 M8020A SCPI Editor

For complete details, refer to section “SCPI Editor” in the *Keysight M8000 Series of BER Test Solutions User Guide*.

Executing SCPI Command

To execute a SCPI command, follow the given steps:

- Select the SCPI from the given list. You can also type the SCPI in the provided text box to expedite the command search.
- Use the proper SCPI command syntax along with the command separators.

For example, the following command is used to enable or disable the Triggered Start:

```
:PLUGin:CPHYplugin:PATtern:TRIGstart:ENABLE 'MIPI C-PHY Editor 1',  
OFF
```

- Click **Execute**. The output of the SCPI command will be displayed in the **History** pane.

A command is invalid and will be rejected if:

- It contains a syntax error.
- It cannot be identified.
- It has too few or too many parameters.
- A parameter is out of range.
- It is out of context.

Sending Commands using VISA

The following is a list of the available hardware interfaces for sending commands to the M8020A firmware:

SCPI Access (HiSLIP): TCPIP0::localhost::hislip0::INSTR (High-Speed LAN Instrument Protocol)

SCPI Access (VXI-11): TCPIP0::localhost::inst0::INSTR (VXI-11 is a TCP/IP instrument protocol defined by the VXIbus Consortium)

SCPI Access (Socket): TCPIP0::localhost::5025::SOCKET (Standard SCPI-over-sockets port)

SCPI Access (Telnet): telnet localhost 5024 (Communication with LAN instrument through SCPI Telnet port)

For further details on SCPI command language, refer to *M8020A Programming Guide*.

SCPI Command Reference

The C-PHY Editor has the following commands:

SCPI Commands for connection to instruments

Table 12 SCPI Commands for connection to instruments

Command	Description under
:PLUGin:CPHYplugin:LINK:NLANes[?]	For details, see :PLUGin:CPHYplugin:LINK:NLANes[?] on page 150.
:PLUGin:CPHYplugin:LINK:OFFLine[?]	For details, see :PLUGin:CPHYplugin:LINK:OFFLine[?] on page 151.
:PLUGin:CPHYplugin:LINK:CALibrated[?]	For details, see :PLUGin:CPHYplugin:LINK:CALibrated[?] on page 151.
:PLUGin:CPHYplugin:LINK:AWG:MODEl[?]	For details, see :PLUGin:CPHYplugin:LINK:AWG:MODEl[?] on page 151.
:PLUGin:CPHYplugin:LINK:AWG:LPFilter[?]	For details, see :PLUGin:CPHYplugin:LINK:AWG:LPFilter[?] on page 152.
:PLUGin:CPHYplugin:LINK:CLKSync[?]	For details, see :PLUGin:CPHYplugin:LINK:CLKSync[?] on page 152.
:PLUGin:CPHYplugin:LINK:SYNCmodule:ADDRes[?]	For details, see :PLUGin:CPHYplugin:LINK:SYNCmodule:ADDRes[?] on page 153.
:PLUGin:CPHYplugin:LINK:AWGA[?]	For details, see :PLUGin:CPHYplugin:LINK:AWGA[?] on page 153.
:PLUGin:CPHYplugin:LINK:AWGB[?]	For details, see :PLUGin:CPHYplugin:LINK:AWGB[?] on page 153.
:PLUGin:CPHYplugin:LINK:AWGC[?]	For details, see :PLUGin:CPHYplugin:LINK:AWGC[?] on page 154.
:PLUGin:CPHYplugin:LINK:OSCilloscope:OFFLine[?]	For details, see :PLUGin:CPHYplugin:LINK:OSCilloscope:OFFLine[?] on page 154.
:PLUGin:CPHYplugin:LINK:OSCilloscope[:ADDRes][?]	For details, see :PLUGin:CPHYplugin:LINK:OSCilloscope[:ADDRes][?] on page 155.
:PLUGin:CPHYplugin:LINK:CONNEct:STATus?	For details, see :PLUGin:CPHYplugin:LINK:CONNEct:STATus? on page 155.

:PLUGin:CPHYplugin:LINK:NLANes[?]

Syntax	:PLUGin:CPHYplugin:LINK:NLANes 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:LINK:NLANes? 'pluginidentifier'
Parameters	LANE1 LANE2 LANE3
Description	This command is used to set the number of lanes in C-PHY Plug-in. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:LINK:NLANes 'MIPI C-PHY Editor 1', LANE2

Query:

```
:PLUGin:CPHYplugin:LINK:NLANes? 'MIPI C-PHY Editor 1'
```

:PLUGin:CPHYplugin:LINK:OFFLine[?]

Syntax :PLUGin:CPHYplugin:LINK:OFFLine 'pluginidentifier', <parameter>
:PLUGin:CPHYplugin:LINK:OFFLine? 'pluginidentifier'

Parameters ON | OFF | 1 | 0

Description This command is used to enable or disable the offline mode. In offline mode, the instruments allow the editor to execute without real instruments.

This query returns the present setting.

Example Command:

```
:PLUGin:CPHYplugin:LINK:OFFLine 'MIPI C-PHY Editor 1', 1
```

Query:

```
:PLUGin:CPHYplugin:LINK:OFFLine? 'MIPI C-PHY Editor 1'
```

:PLUGin:CPHYplugin:LINK:CALibrated[?]

Syntax :PLUGin:CPHYplugin:LINK:CALibrated 'pluginidentifier', <parameter>
:PLUGin:CPHYplugin:LINK:CALibrated? 'pluginidentifier'

Parameters ON | OFF | 1 | 0

Description This command is used to enable or disable the editor to use or not to use calibrated values. When this option is enabled the editor checks if the C-PHY CTS calibrations are available. If no calibrations are available the editor uses the uncalibrated values.

This query returns the present setting.

Example Command:

```
:PLUGin:CPHYplugin:LINK:CALibrated 'MIPI C-PHY Editor 1', ON
```

Query:

```
:PLUGin:CPHYplugin:LINK:CALibrated? 'MIPI C-PHY Editor 1'
```

:PLUGin:CPHYplugin:LINK:AWG:MODEl[?]

Syntax :PLUGin:CPHYplugin:LINK:AWG:MODEl 'pluginidentifier', <parameter>
:PLUGin:CPHYplugin:LINK:AWG:MODEl? 'pluginidentifier'

Parameters M8195

Description This command allows you to set the AWG modules. Currently, the software supports M8195A AWG modules only.

This query returns the present setting.

Example Command:

```
:PLUGin:CPHYplugin:LINK:AWG:MODEl 'MIPI C-PHY Editor 1', M8195
```

Query:

```
:PLUGin:CPHYplugin:LINK:AWG:MODEl? 'MIPI C-PHY Editor 1'
```

```
:PLUGin:CPHYplugin:LINK:AWG:LPFilter[?]
```

Syntax :PLUGin:CPHYplugin:LINK:AWG:LPFilter 'pluginidentifier', <parameter>

```
:PLUGin:CPHYplugin:LINK:AWG:LPFilter? 'pluginidentifier'
```

Parameters ON | OFF | 1 | 0

Description This command enables or disables the Low-Pass Filter option in the Connection Setup.

This query returns the present setting.

Example Command:

```
:PLUGin:CPHYplugin:LINK:AWG:LPFilter 'MIPI C-PHY Editor 1', ON
```

Query:

```
:PLUGin:CPHYplugin:LINK:AWG:LPFilter? 'MIPI C-PHY Editor 1'
```

```
:PLUGin:CPHYplugin:LINK:CLKSync[?]
```

Syntax :PLUGin:CPHYplugin:LINK:CLKSync 'pluginidentifier', <parameter>

```
:PLUGin:CPHYplugin:LINK:CLKSync? 'pluginidentifier'
```

Parameters VISA Address (string format)

Description This command is used to set the VISA Address of the Clock Sync Module.

This query returns the present setting.

Example Command:

```
:PLUGin:CPHYplugin:LINK:CLKSync 'MIPI C-PHY Editor 1',  
'TCPIP0::121.0.0.1::hisip0::INSTR'
```

Query:

```
:PLUGin:CPHYplugin:LINK:CLKSync? 'MIPI C-PHY Editor 1'
```

:PLUGin:CPHYplugin:LINK:SYNCmodule:ADDRess[?]

Syntax	:PLUGin:CPHYplugin:LINK:SYNCmodule:ADDRess 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:LINK:SYNCmodule:ADDRess? 'pluginidentifier'
Parameters	VISA Address (string format)
Description	This command is used to set the VISA Address of the Clock Sync Module. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:LINK:SYNCmodule:ADDRess 'MIPI C-PHY Editor 1', 'TCPIP0::121.0.0.1::hislip0::INSTR' Query: :PLUGin:CPHYplugin:LINK:SYNCmodule:ADDRess? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:LINK:AWGA[?]

Syntax	:PLUGin:CPHYplugin:LINK:AWGA 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:LINK:AWGA? 'pluginidentifier'
Parameters	VISA Address (string format)
Description	This command specifies the Visa connection string so as to establish a connection with AWG 1. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:LINK:AWGA 'MIPI C-PHY Editor 1', 'TCPIP0::121.0.0.1::hislip1::INSTR' Query: :PLUGin:CPHYplugin:LINK:AWGA? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:LINK:AWGB[?]

Syntax	:PLUGin:CPHYplugin:LINK:AWGB 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:LINK:AWGB? 'pluginidentifier'
Parameters	VISA Address (string format)
Description	This command specifies the Visa connection string so as to establish a connection with AWG 2. This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:LINK:AWGB 'MIPI C-PHY Editor 1'
'TCPIPO::121.0.0.1::hislip2::INSTR'
Query:
:PLUGin:CPHYplugin:LINK:AWGB? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:LINK:AWGC[?]

Syntax :PLUGin:CPHYplugin:LINK:AWGC 'pluginidentifier', <parameter>
:PLUGin:CPHYplugin:LINK:AWGC? 'pluginidentifier'

Parameters VISA Address (string format)

Description This command specifies the Visa connection string so as to establish a connection with AWG 3.
This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:LINK:AWGC 'MIPI C-PHY Editor 1',
'TCPIPO::121.0.0.1::hislip3::INSTR'
Query:
:PLUGin:CPHYplugin:LINK:AWGC? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:LINK:OSCilloscope:OFFLine[?]

Syntax :PLUGin:CPHYplugin:LINK:OSCilloscope:OFFLine 'pluginidentifier',
<parameter>
:PLUGin:CPHYplugin:LINK:OSCilloscope:OFFLine? 'pluginidentifier'

Parameters ON | OFF | 1 | 0

Description This command is used to enable or disable the oscilloscope. When this option is enabled the editor does not connect to an oscilloscope. It is not possible to calibrate the skews when no oscilloscope is connected.
This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:LINK:OSCilloscope:OFFLine 'MIPI C-PHY Editor 1', 1
Query:
:PLUGin:CPHYplugin:LINK:OSCilloscope:OFFLine? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:LINK:OSCilloscope[:ADDRess][?]

Syntax	:PLUGin:CPHYplugin:LINK:OSCilloscope[:ADDRess] 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:LINK:OSCilloscope[:ADDRess]? 'pluginidentifier'
Parameters	VISA Address (string format)
Description	This command is used to set the VISA Address of the oscilloscope. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:LINK:OSCilloscope:ADDRess 'MIPI C-PHY Editor 1', 'TCPIP0::121.0.0.1::inst0::INSTR' Query: :PLUGin:CPHYplugin:LINK:OSCilloscope? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:LINK:CONNect:STATus?

Syntax	:PLUGin:CPHYplugin:LINK:CONNect:STATus? 'pluginidentifier'
Description	This query returns the value '1' if the Editor is currently connected to the instruments otherwise returns '0'.
Example	Query: :PLUGin:CPHYplugin:LINK:CONNect:STATus? 'MIPI C-PHY Editor 1'

SCPI Commands for Data Pattern Group

Table 13 SCPI Commands for Data Pattern Group

Command	Description under
:PLUGin:CPHYplugin:PATtern:TRIGstart[:ENABLE][?]	For details, see :PLUGin:CPHYplugin:PATtern:TRIGstart[:ENABLE][?] on page 157.
:PLUGin:CPHYplugin:PATtern:TRIGstart:PATtern[?]	For details, see :PLUGin:CPHYplugin:PATtern:TRIGstart:PATtern[?] on page 157.
:PLUGin:CPHYplugin:RUN:MODE[?]	For details, see :PLUGin:CPHYplugin:RUN:MODE[?] on page 157.
:PLUGin:CPHYplugin:PATtern:IDLe[:VOLTage][?]	For details, see :PLUGin:CPHYplugin:PATtern:IDLe[:VOLTage][?] on page 158.
:PLUGin:CPHYplugin:PATtern:MODE[?]	For details, see :PLUGin:CPHYplugin:PATtern:MODE[?] on page 158.
:PLUGin:CPHYplugin:PATtern:LSBFirst[?]	For details, see :PLUGin:CPHYplugin:PATtern:LSBFirst[?] on page 159.
:PLUGin:CPHYplugin:PATtern:PROTOcol[?]	For details, see :PLUGin:CPHYplugin:PATtern:PROTOcol[?] on page 159.
:PLUGin:CPHYplugin:PATtern:DSIVersion[?]	For details, see :PLUGin:CPHYplugin:PATtern:DSIVersion[?] on page 159.
:PLUGin:CPHYplugin:PATtern:DATA:FIleName[?]	For details, see :PLUGin:CPHYplugin:PATtern:DATA:FIleName[?] on page 160.
:PLUGin:CPHYplugin:PATtern:HSDATA:FIleName[?]	For details, see :PLUGin:CPHYplugin:PATtern:HSDATA:FIleName[?] on page 160.
:PLUGin:CPHYplugin:PATtern:HSDATA[:BYTeS][?]	For details, see :PLUGin:CPHYplugin:PATtern:HSDATA[:BYTeS][?] on page 161.
:PLUGin:CPHYplugin:PATtern:LPDATA:FIleName[?]	For details, see :PLUGin:CPHYplugin:PATtern:LPDATA:FIleName[?] on page 161.
:PLUGin:CPHYplugin:PATtern:LPDATA[:BYTeS][?]	For details, see :PLUGin:CPHYplugin:PATtern:LPDATA[:BYTeS][?] on page 162.
:PLUGin:CPHYplugin:PATtern:SEQuence[:FIleName][?]	For details, see :PLUGin:CPHYplugin:PATtern:SEQuence[:FIleName][?] on page 162.
:PLUGin:CPHYplugin:AUX:CHANnel:PATtern[:MODE][?]	For details, see :PLUGin:CPHYplugin:AUX:CHANnel:PATtern[:MODE][?] on page 163.
:PLUGin:CPHYplugin:AUX:CHANnel:VOLTage:AMPLitude[?]	For details, see :PLUGin:CPHYplugin:AUX:CHANnel:VOLTage:AMPLitude[?] on page 163.
:PLUGin:CPHYplugin:AUX:CHANnel:VOLTage:OFFSet[?]	For details, see :PLUGin:CPHYplugin:AUX:CHANnel:VOLTage:OFFSet[?] on page 164.

:PLUGin:CPHYplugin:PATtern:TRIGstart[:ENABle][?]

Syntax	:PLUGin:CPHYplugin:PATtern:TRIGstart[:ENABle] 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:PATtern:TRIGstart[:ENABle]? 'pluginidentifier'
Parameters	ON OFF 1 0
Description	This command is used to enable or disable the Triggered Start. Triggered Start means that an initial looped block is inserted at the beginning of the sequence. The sequence only breaks from the looped block after an explicit Trigger event.
Example	Command: :PLUGin:CPHYplugin:PATtern:TRIGstart:ENABle 'MIPI C-PHY Editor 1', OFF Query: :PLUGin:CPHYplugin:PATtern:TRIGstar? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:PATtern:TRIGstart:PATtern[?]

Syntax	:PLUGin:CPHYplugin:PATtern:TRIGstart:PATtern 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:PATtern:TRIGstart:PATtern? 'pluginidentifier'
Parameters	LP000 LP111
Description	This command sets the type of Low Power Mode state used for the initial looped block when Triggered Start is enabled. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:PATtern:TRIGstart:PATtern 'MIPI C-PHY Editor 1', LP111 Query: :PLUGin:CPHYplugin:PATtern:TRIGstart:PATtern? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:RUN:MODE[?]

Syntax	:PLUGin:CPHYplugin:RUN:MODE 'pluginidentifier', <mode> :PLUGin:CPHYplugin:RUN:MODE? 'pluginidentifier'
Parameter	'Identifier': 'MIPI C-PHY Editor 1', INTerrupted CONTinuous
Description	This command changes the run mode of the plugin.

This query returns the present setting.

Example

Command:

```
:PLUGin:CPHYplugin:RUN:MODE 'MIPI C-PHY Editor 1', INT
```

Query:

```
:PLUGin:CPHYplugin:RUN:MODE? 'MIPI C-PHY Editor 1'
```

```
:PLUGin:CPHYplugin:PATtern:IDLe[:VOLTage][?]
```

Syntax :PLUGin:CPHYplugin:PATtern:IDLe[:VOLTage] 'pluginidentifier', <Enum>

```
:PLUGin:CPHYplugin:PATtern:IDLe[:VOLTage]? 'pluginidentifier'
```

Parameter 'Identifier': 'MIPI C-PHY Editor 1'

Description This command sets the voltage on the AWGs when they are in Stop state.

This query returns the present setting.

Example

Command:

```
:PLUGin:CPHYplugin:PATtern:IDLe[:VOLTage] 'MIPI C-PHY Editor 1',  
3.000e-001
```

Query:

```
:PLUGin:CPHYplugin:PATtern:IDLe[:VOLTage]? 'MIPI C-PHY Editor 1'
```

```
:PLUGin:CPHYplugin:PATtern:MODE[?]
```

Syntax :PLUGin:CPHYplugin:PATtern:MODE 'pluginidentifier', <parameter>

```
:PLUGin:CPHYplugin:PATtern:MODE? 'pluginidentifier'
```

Parameters PATtern | Burst | PUREhs | Frames

Description This command is used to set the Pattern Mode selection. The pattern mode allows do define the structure of the pattern data that is provided to the Editor.

This query returns the present setting.

Example

Command:

```
:PLUGin:CPHYplugin:PATtern:MODE 'MIPI C-PHY Editor 1', Burst
```

Query:

```
:PLUGin:CPHYplugin:PATtern:MODE? 'MIPI C-PHY Editor 1'
```

:PLUGin:CPHYplugin:PATtern:LSBFirst[?]

Syntax	:PLUGin:CPHYplugin:PATtern:LSBFirst 'pluginidentifier', <Boolean> :PLUGin:CPHYplugin:PATtern:LSBFirst? 'pluginidentifier'
Parameters	'Identifier': 'MIPI C-PHY Editor 1', 1 0
Description	This command defines the byte transmission order. When LSB is selected, the least significant bit is sent first. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:PATtern:LSBFirst 'MIPI C-PHY Editor 1', Burst Query: :PLUGin:CPHYplugin:PATtern:LSBFirst? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:PATtern:PROTOcol[?]

Syntax	:PLUGin:CPHYplugin:PATtern:PROTOcol 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:PATtern:PROTOcol? 'pluginidentifier'
Parameters	CSI DSI
Description	This command is used to set the CSI and the DSI protocol. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:PATtern:PROTOcol 'MIPI C-PHY Editor 1', CSI Query: :PLUGin:CPHYplugin:PATtern:PROTOcol? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:PATtern:DSIVersion[?]

Syntax	:PLUGin:CPHYplugin:PATtern:DSIVersion 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:PATtern:DSIVersion? 'pluginidentifier'
Parameters	V10 V11
Description	This command is used to set the DSI2 version either as V1.0 or V1.1. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:PATtern:DSIVersion 'MIPI C-PHY Editor 1', V11

Query:

```
:PLUGIn:CPHYplugin:PATtern:DSIVersion? 'MIPI C-PHY Editor 1'
```

:PLUGIn:CPHYplugin:PATtern:DATA:FiLename[?]

Syntax :PLUGIn:CPHYplugin:PATtern:DATA:FiLename 'pluginidentifier',
<parameter>

```
:PLUGIn:CPHYplugin:PATtern:DATA:FiLename? 'pluginidentifier'
```

Parameters 'filepath'

Description This command is used to set the file path. Th file contains the data patterns, which are expected to be encoded in terms of line states.

This query returns the present setting.

Example Command:

```
:PLUGIn:CPHYplugin:PATtern:DATA:FiLename 'MIPI C-PHY Editor 1', 'C:\Users\user1\Documents\Keysight\M8070B\Workspaces\Default\Factory\Settings\C-PHY\CPHY_PatternFile.ptrn'
```

Query:

```
:PLUGIn:CPHYplugin:PATtern:DATA:FiLename? 'MIPI C-PHY Editor 1'
```

:PLUGIn:CPHYplugin:PATtern:HSData:FiLename[?]

Syntax :PLUGIn:CPHYplugin:PATtern:HSData:FiLename 'pluginidentifier',
<parameter>

```
:PLUGIn:CPHYplugin:PATtern:HSData:FiLename? 'pluginidentifier'
```

Parameters 'filepath'

Description This command is used to set the file path with the High Speed data pattern. It is expected that file contains the data in hexadecimal bytes separated by commas (e.g 0xE7, 0xE7, 0xE7, 0xE7).

This query returns the present setting.

Example Command:

```
:PLUGIn:CPHYplugin:PATtern:HSData:FiLename 'MIPI C-PHY Editor 1', 'C:\Users\user1\Documents\Keysight\M8070B\Workspaces\Default\Factory\Settings\C-PHY\CPHY_HS_DataFile.dat'
```

Query:

```
:PLUGIn:CPHYplugin:PATtern:HSData:FiLename? 'MIPI C-PHY Editor 1'
```

:PLUGIn:CPHYplugin:PATtern:HSData[:BYtes][?]

Syntax	:PLUGIn:CPHYplugin:PATtern:HSData[:BYtes] 'pluginidentifier', <parameter> :PLUGIn:CPHYplugin:PATtern:HSData[:BYtes]? 'pluginidentifier'
Parameters	Hexadecimals bytes (string format)
Description	This command is used to set the High Speed data pattern. It is expected that the data is given in hexadecimal bytes separated by commas (e.g 0xE7, 0xE7, 0xE7, 0xE7). This query returns the present setting.
Example	Command: :PLUGIn:CPHYplugin:PATtern:HSData 'MIPI C-PHY Editor 1','0xE7, 0xE7, 0xE7, 0xE7' Query: :PLUGIn:CPHYplugin:PATtern:HSData:BYtes? 'MIPI C-PHY Editor 1'

:PLUGIn:CPHYplugin:PATtern:LPData:FIleName[?]

Syntax	:PLUGIn:CPHYplugin:PATtern:LPData:FIleName 'pluginidentifier', <parameter> :PLUGIn:CPHYplugin:PATtern:LPData:FIleName? 'pluginidentifier'
Parameters	'filepath'
Description	This command is used to set the file path with Low Power data pattern. It is expected that file contains the data in hexadecimal bytes separated by commas (e.g 0xE7, 0xE7, 0xE7, 0xE7). This query returns the present setting.
Example	Command: :PLUGIn:CPHYplugin:PATtern:LPData:FIleName 'MIPI C-PHY Editor 1', 'C:\Users\user1\Documents\Keysight\M8070B\Workspaces\Default\Factory\Settings\C-PHY\CPHY_LP_DataFile.dat' Query: :PLUGIn:CPHYplugin:PATtern:LPData:FIleName? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:PATtern:LPData[:BYtes][?]

Syntax	:PLUGin:CPHYplugin:PATtern:LPData[:BYtes] 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:PATtern:LPData[:BYtes]? 'pluginidentifier'
Parameters	LP pattern in hexadecimal bytes (string format)
Description	This command is used to set the Low Power data pattern. It is expected that the data is given in hexadecimal bytes separated by commas (e.g 0xE7, 0xE7, 0xE7, 0xE7). This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:PATtern:LPData:BYtes 'MIPI C-PHY Editor 1', '0xE7, 0xE7, 0xE7, 0xE7' Query: :PLUGin:CPHYplugin:PATtern:LPData? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:PATtern:SEquence[:FILename][?]

Syntax	:PLUGin:CPHYplugin:PATtern:SEquence[:FILename] 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:PATtern:SEquence[:FILename]? 'pluginidentifier'
Parameters	'filepath'
Description	This command is used to set the Sequence file when Data Pattern Mode is set to Frame. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:PATtern:SEquence:FILename 'MIPI C-PHY Editor 1','C:\Users\user1\Documents\Keysight\M8070B\Workspaces\Default\Factory\Settings\C-PHY\CPHY_LP_SeqFile.seq' Query: :PLUGin:CPHYplugin:PATtern:SEquence? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:AUX:CHANnel:PATtern[:MODE][?]

Syntax	:PLUGin:CPHYplugin:AUX:CHANnel:PATtern[:MODE] 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:AUX:CHANnel:PATtern[:MODE]? 'pluginidentifier'
Parameters	TRIGger CLOCK AMIRror BMIRror CMIRror OFF
Description	This command is used to set the Pattern Mode selection. The pattern mode allows do define the structure of the pattern data that is provided to the Editor. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:AUX:CHANnel:PATtern:MODE 'MIPI C-PHY Editor 1', TRIG Query: :PLUGin:CPHYplugin:AUX:CHANnel:PATtern:MODE? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:AUX:CHANnel:VOLtag:AMPLitude[?]

Syntax	:PLUGin:CPHYplugin:AUX:CHANnel:VOLtag:AMPLitude 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:AUX:CHANnel:VOLtag:AMPLitude? 'pluginidentifier'
Parameters	Aux Channel voltage in the scientific notation.
Description	This command is used to set the amplitude of the auxiliary channel. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:AUX:CHANnel:VOLtag:AMPLitude 'MIPI C-PHY Editor 1', 9.000e-001 Query: :PLUGin:CPHYplugin:AUX:CHANnel:VOLtag:AMPLitude? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:AUX:CHANnel:VOLTage:OFFSet[?]

Syntax	:PLUGin:CPHYplugin:AUX:CHANnel:VOLTage:OFFSet 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:AUX:CHANnel:VOLTage:OFFSet? 'pluginidentifier'
Parameters	Aux Channel Offset in the scientific notation.
Description	This command is used to set the offset on the auxiliary channel. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:AUX:CHANnel:VOLTage:OFFSet 'MIPI C-PHY Editor 1', 0.000e+000 Query: :PLUGin:CPHYplugin:AUX:CHANnel:VOLTage:OFFSet? 'MIPI C-PHY Editor 1'

SCPI Commands for Data Rate and Transition Time Group

Table 14 SCPI Commands for Data Rate and Transition Time Group

Command	Description under
:PLUGin:CPHYplugin:HSMoDe:SYMRate[?]	For details, see : PLUGin:CPHYplugin:HSMoDe:SYMRate[?] on page 165.
:PLUGin:CPHYplugin:HSMoDe:TT[?]	For details, see : PLUGin:CPHYplugin:HSMoDe:TT[?] on page 165.
:PLUGin:CPHYplugin:HSMoDe:TT:CALibrated[?]	For details, see : PLUGin:CPHYplugin:HSMoDe:TT:CALibrated[?] on page 166.
:PLUGin:CPHYplugin:LPMoDe:DAtRate[?]	For details, see : PLUGin:CPHYplugin:LPMoDe:DAtRate[?] on page 166.
:PLUGin:CPHYplugin:LPMoDe:TT?	For details, see : PLUGin:CPHYplugin:LPMoDe:TT[?] on page 167.
:PLUGin:CPHYplugin:LPMoDe:TT:CALibrated[?]	For details, see : PLUGin:CPHYplugin:LPMoDe:TT:CALibrated[?] on page 167.

:PLUGin:CPHYplugin:HSMoDe:SYMRate[?]

Syntax	:PLUGin:CPHYplugin:HSMoDe:SYMRate 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:HSMoDe:SYMRate? 'pluginidentifier'
Parameters	Symbol Rate (double)
Range	Refer to M8195A AWG specifications.
Description	This command is used to set the High Speed Mode Symbol Rate. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:HSMoDe:SYMRate 'MIPI C-PHY Editor 1', 2.500e+009 Query: :PLUGin:CPHYplugin:HSMoDe:SYMRate? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:HSMoDe:TT[?]

Syntax	:PLUGin:CPHYplugin:HSMoDe:TT 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:HSMoDe:TT? 'pluginidentifier'
Parameters	Transition Time (double)
Range	20 ps to 500 ns
Description	This command is used to set the High Speed Mode Transition Time. The Transition Time is applied on both fall and rise times. This query returns the present setting.

Example Command:
:PLUGIn:CPHYplugin:HSMODE:TT 'MIPI C-PHY Editor 1', 8.200e-011
Query:
:PLUGIn:CPHYplugin:HSMODE:TT? 'MIPI C-PHY Editor 1'

:PLUGIn:CPHYplugin:HSMODE:TT:CALibrated[?]

Syntax :PLUGIn:CPHYplugin:HSMODE:TT:CALibrated 'pluginidentifier',
<parameter>
:PLUGIn:CPHYplugin:HSMODE:TT:CALibrated? 'pluginidentifier'

Parameters ON | OFF | 1 | 0

Description This command is used to enable or disable the HS Mode calibrated values.
This query returns the present setting.

Example Command:
:PLUGIn:CPHYplugin:HSMODE:TT:CALibrated 'MIPI C-PHY Editor 1', 1
Query:
:PLUGIn:CPHYplugin:HSMODE:TT:CALibrated? 'MIPI C-PHY Editor 1'

:PLUGIn:CPHYplugin:LPMODE:DAtRate[?]

Syntax :PLUGIn:CPHYplugin:LPMODE:DAtRate 'pluginidentifier', <parameter>
:PLUGIn:CPHYplugin:LPMODE:DAtRate? 'pluginidentifier'

Parameters Data Rate (double)

Range 5 Mb/s to 150 Mb/s

Description This command is used to set the Low Power Mode Data Rate.
This query returns the present setting.

Example Command:
:PLUGIn:CPHYplugin:LPMODE:DAtRate 'MIPI C-PHY Editor 1', 1.000e+007
Query:
:PLUGIn:CPHYplugin:LPMODE:DAtRate? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:LPMODE:TT[?]

Syntax	:PLUGin:CPHYplugin:LPMODE:TT 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:LPMODE:TT? 'pluginidentifier'
Parameters	Transition Time (double)
Range	60 ps to 25 ns
Description	This command is used to set the Low Power Transition Time. The Transition Time is applied on both fall and rise times. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:LPMODE:TT 'MIPI C-PHY Editor 1', 2.000e-009 Query: :PLUGin:CPHYplugin:LPMODE:TT? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:LPMODE:TT:CALibrated[?]

Syntax	:PLUGin:CPHYplugin:LPMODE:TT:CALibrated 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:LPMODE:TT:CALibrated? 'pluginidentifier'
Parameters	ON OFF 1 0
Description	This command is used to enable or disable the LP Mode calibrated values. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:LPMODE:TT:CALibrated 'MIPI C-PHY Editor 1', 1 Query: :PLUGin:CPHYplugin:LPMODE:TT:CALibrated? 'MIPI C-PHY Editor 1'

SCPI Commands for Delay Group

Table 15 SCPI Commands for Delay Group

Command	Description under
:PLUGin:CPHYplugin:DELAy:DATA{1:3}[?]	For details, see :PLUGin:CPHYplugin:DELAy:DATA{1:3}[?] on page 168.
:PLUGin:CPHYplugin:DELAy:INTRa[:ENABLe][?]	For details, see :PLUGin:CPHYplugin:DELAy:INTRa[:ENABLe][?] on page 169.
:PLUGin:CPHYplugin:DELAy:INTRa:LANE[?]	For details, see :PLUGin:CPHYplugin:DELAy:INTRa:LANE[?] on page 169.
:PLUGin:CPHYplugin:DELAy:INTRa:LINE:A[?]	For details, see :PLUGin:CPHYplugin:DELAy:INTRa:LINE:A[?] on page 170.
:PLUGin:CPHYplugin:DELAy:INTRa:LINE:B[?]	For details, see :PLUGin:CPHYplugin:DELAy:INTRa:LINE:B[?] on page 170.
:PLUGin:CPHYplugin:DELAy:INTRa:LINE:C[?]	For details, see :PLUGin:CPHYplugin:DELAy:INTRa:LINE:C[?] on page 170.

:PLUGin:CPHYplugin:DELAy:DATA{1:3}[?]

Syntax	:PLUGin:CPHYplugin:DELAy:DATA1 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:DELAy:DATA1? 'pluginidentifier' :PLUGin:CPHYplugin:DELAy:DATA2 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:DELAy:DATA2? 'pluginidentifier' :PLUGin:CPHYplugin:DELAy:DATA3 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:DELAy:DATA3? 'pluginidentifier'
Parameters	Delay in seconds (double)
Range	-5UI to +5UI
Description	This command is used to apply the delay in all lines either of Data0, of Data1 or of Data2. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:DELAy:DATA1 'MIPI C-PHY Editor 1', 0.000e+000 Query: :PLUGin:CPHYplugin:DELAy:DATA1? 'MIPI C-PHY Editor 1' Command: :PLUGin:CPHYplugin:DELAy:DATA2 'MIPI C-PHY Editor 1', 0.000e+000 Query: :PLUGin:CPHYplugin:DELAy:DATA2? 'MIPI C-PHY Editor 1'

Command:

```
:PLUGin:CPHYplugin:DELAy:DATa3 'MIPI C-PHY Editor 1', 0.000e+000
```

Query:

```
:PLUGin:CPHYplugin:DELAy:DATa3? 'MIPI C-PHY Editor 1'
```

:PLUGin:CPHYplugin:DELAy:INTRa[:ENABLe][?]

Syntax :PLUGin:CPHYplugin:DELAy:INTRa[:ENABLe] 'pluginidentifier',
<parameter>
:PLUGin:CPHYplugin:DELAy:INTRa[:ENABLe]? 'pluginidentifier'

Parameters ON | OFF | 1 | 0

Description This command is used to enable or disable the delay between lines. When this options is enabled it is possible to apply delay on each line individually.

This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:DELAy:INTRa[:ENABLe] 'MIPI C-PHY Editor 1', OFF
Query:
:PLUGin:CPHYplugin:DELAy:INTRa[:ENABLe]? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:DELAy:INTRa:LANE[?]

Syntax :PLUGin:CPHYplugin:DELAy:INTRa:LANE 'pluginidentifier', <parameter>
:PLUGin:CPHYplugin:DELAy:INTRa:LANE? 'pluginidentifier'

Parameters ALL | DATA0 | DATA1 | DATA2

Description This command is used to set the lane where the intra-delay will be applied.

This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:DELAy:INTRa:LANE 'MIPI C-PHY Editor 1', ALL
Query:
:PLUGin:CPHYplugin:DELAy:INTRa:LANE? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:DELAy:INTRa:LINE:A[?]

Syntax	:PLUGin:CPHYplugin:DELAy:INTRa:LINE:A 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:DELAy:INTRa:LINE:A? 'pluginidentifier'
Parameters	Delay in seconds (double)
Range	0mUI to 5mUI
Description	This command is used to apply delay on Line A. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:DELAy:INTRa:LINE:A 'MIPI C-PHY Editor 1', 0.000e+000 Query: :PLUGin:CPHYplugin:DELAy:INTRa:LINE:A? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:DELAy:INTRa:LINE:B[?]

Syntax	:PLUGin:CPHYplugin:DELAy:INTRa:LINE:B 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:DELAy:INTRa:LINE:B? 'pluginidentifier'
Parameters	Delay in seconds (double)
Range	0mUI to 5mUI
Description	This command is used to apply delay on Line B. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:DELAy:INTRa:LINE:B 'MIPI C-PHY Editor 1', 0.000e+000 Query: :PLUGin:CPHYplugin:DELAy:INTRa:LINE:B? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:DELAy:INTRa:LINE:C[?]

Syntax	:PLUGin:CPHYplugin:DELAy:INTRa:LINE:C 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:DELAy:INTRa:LINE:C? 'pluginidentifier'
Parameters	Delay in seconds (double)
Range	0mUI to 5mUI
Description	This command is used to apply the delay on Line C.

This query returns the present setting.

Example

Command:

```
:PLUGin:CPHYplugin:DELAy:INTRa:LINE:C 'MIPI C-PHY Editor 1',  
0.000e+000
```

Query:

```
:PLUGin:CPHYplugin:DELAy:INTRa:LINE:C? 'MIPI C-PHY Editor 1'
```

SCPI Commands for Disturbances Group

Table 16 SCPI Commands for Disturbance Group

Command	Description under
:PLUGin:CPHYplugin:LPPulse[?]	For details, see: PLUGin:CPHYplugin:LPPulse[?] on page 172.
:PLUGin:CPHYplugin:IMPairments:HISpeed:DCD[?]	For details, see: PLUGin:CPHYplugin:IMPairments:HISpeed:DCD[?] on page 173.
:PLUGin:CPHYplugin:IMPairments:HISpeed:DCD:CALibrated[?]	For details, see: PLUGin:CPHYplugin:IMPairments:HISpeed:DCD:CALibrated[?] on page 173.
:PLUGin:CPHYplugin:ESPike:MODE[?]	For details, see : PLUGin:CPHYplugin:ESPike:MODE[?] on page 173.
:PLUGin:CPHYplugin:ESPike:AREA[?]	For details, see : PLUGin:CPHYplugin:ESPike:AREA[?] on page 174.
:PLUGin:CPHYplugin:ESPike:LHIGhinvol[?]	For details, see : PLUGin:CPHYplugin:ESPike:LHIGhinvol[?] on page 174.
:PLUGin:CPHYplugin:ESPike:LLOWintvol[?]	For details, see : PLUGin:CPHYplugin:ESPike:LLOWintvol[?] on page 175.
:PLUGin:CPHYplugin:ESPike:CALibrated[?]	For details, see : PLUGin:CPHYplugin:ESPike:CALibrated[?] on page 175.
:PLUGin:CPHYplugin:EMBedding:ENABLEd[?]	For details, see : PLUGin:CPHYplugin:EMBedding:ENABLEd[?] on page 176.
:PLUGin:CPHYplugin:INSCal:ENABLEd[?]	For details, see : PLUGin:CPHYplugin:INSCal:ENABLEd[?] on page 176.
:PLUGin:CPHYplugin:EMBedding:PATH:SPArAmeters[?]	For details, see : PLUGin:CPHYplugin:EMBedding:PATH:SPArAmeters[?] on page 176.

:PLUGin:CPHYplugin:LPPulse[?]

Syntax	:PLUGin:CPHYplugin:LPPulse 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:LPPulse? 'pluginidentifier'
Parameters	Time in seconds (double)
Range	1ns to 100ns
Description	This command is used to set the pulse width of low power mode, which in turn changes the LP duty cycle. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:LPPulse 'MIPI C-PHY Editor 1', 6.000000000000000E-08 Query: :PLUGin:CPHYplugin:LPPulse? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:IMPairments:HISpeed:DCD[?]

Syntax	:PLUGin:CPHYplugin:IMPairments:HISpeed:DCD', <parameter> :PLUGin:CPHYplugin:IMPairments:HISpeed:DCD
Parameters	Time in seconds (double)
Range	0mUI to 800mUI
Description	This command is used to set the high speed mode duty cycle. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:IMPairments:HISpeed:DCD 'MIPI C-PHY Editor 1', 0.001000000000000E+00 Query: :PLUGin:CPHYplugin:IMPairments:HISpeed:DCD? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:IMPairments:HISpeed:DCD:CALibrated[?]

Syntax	:PLUGin:CPHYplugin:IMPairments:HISpeed:DCD:CALibrated 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:IMPairments:HISpeed:DCD:CALibrated? 'pluginidentifier'
Parameters	ON OFF 1 0
Description	This command is used to enable or disable the HS DCD calibrated values. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:IMPairments:HISpeed:DCD:CALibrated, 'MIPI C-PHY Editor 1', 1 Query: :PLUGin:CPHYplugin:IMPairments:HISpeed:DCD:CALibrated? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:ESPike:MODE[?]

Syntax	:PLUGin:CPHYplugin:ESPike:MODE 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:ESPike:MODE? 'pluginidentifier'
Parameters	HLEvels LLEvels OFF

Description This command is used to enable or disable the e-Spike mode. For enabled state it can be selected whether the spikes should occur inside high levels only or low levels. The e-Spike is inserted in the middle of the symbol with the defined area.

This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:ESpike:MODE' MIPI C-PHY Editor 1', LLEvels
Query:
:PLUGin:CPHYplugin:ESpike:MODE? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:ESpike:AREA[?]

Syntax :PLUGin:CPHYplugin:ESpike:AREA 'pluginidentifier', <parameter>
:PLUGin:CPHYplugin:ESpike:AREA? 'pluginidentifier'

Parameters Area in pVs

Range 0pVs to 5nVs

Description This command is used define the area where the e-Spike is generated. Take into consideration that the granularity of the area is limited by the number of samples available in the effective duration of the e-Spike.

This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:ESpike:AREA 'MIPI C-PHY Editor 1',
2.25000000000000E-10
Query:
:PLUGin:CPHYplugin:ESpike:AREA? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:ESpike:LHIGHinvol[?]

Syntax :PLUGin:CPHYplugin:ESpike:LHIGHinvol 'pluginidentifier', <parameter>
:PLUGin:CPHYplugin:ESpike:LHIGHinvol? 'pluginidentifier'

Parameters Voltage in mV

Range -2V to +6.6V

Description For logic 1, this command sets the lower receiver detection threshold, which is the voltage level where the e-Spike area inside this logic 1 is to be calculated from.

This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:ESpike:LHIGhinvol 'MIPI C-PHY Editor 1',
2.000e-001
Query:
:PLUGin:CPHYplugin:ESpike:LHIGhinvol? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:ESpike:LLOWintvol[?]

Syntax :PLUGin:CPHYplugin:ESpike:LLOWintvol 'MIPI C-PHY Editor 1',
<parameter>
:PLUGin:CPHYplugin:ESpike:LLOWintvol? 'MIPI C-PHY Editor 1'
Parameters Voltage in mV
Range -2V to +6.6V
Description For logic 0, this command sets the upper receiver detection threshold
which is the voltage level where the e-Spike area inside this logic 0 is to be
calculated from.
This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:ESpike:LLOWintvol 'MIPI C-PHY Editor 1',
3.000e-001
Query:
:PLUGin:CPHYplugin:ESpike:LLOWintvol? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:ESpike:CALibrated[?]

Syntax :PLUGin:CPHYplugin:ESpike:CALibrated 'pluginidentifier', <parameter>
:PLUGin:CPHYplugin:ESpike:CALibrated? 'pluginidentifier'
Parameters ON | OFF | 1 | 0
Description This command is used to enable or disable the e-Spike calibrated values.
This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:ESpike:CALibrated 'MIPI C-PHY Editor 1', 0
Query:
:PLUGin:CPHYplugin:ESpike:CALibrated? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:EMBedding:ENABLEd[?]

Syntax	:PLUGin:CPHYplugin:EMBedding:ENABLEd 'identifier', <Boolean> :PLUGin:CPHYplugin:EMBedding:ENABLEd? 'identifier'
Parameter	'Identifier': 'MIPI C-PHY Editor 1', ON OFF 1 0
Description	This command enables the Use Deembedding parameter for signal disturbance. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:EMBedding:ENABLEd 'MIPI C-PHY Editor 1', ON Query: :PLUGin:CPHYplugin:EMBedding:ENABLEd? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:INSCal:ENABLEd[?]

Syntax	:PLUGin:CPHYplugin:INSCal:ENABLEd 'identifier', <Boolean> :PLUGin:CPHYplugin:INSCal:ENABLEd? 'identifier'
Parameter	'Identifier': 'MIPI C-PHY Editor 1', ON OFF 1 0
Description	This command enables the In-System Calibration Enabled parameter for signal disturbance, with the Use Deembedding parameter active. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:INSCal:ENABLEd 'MIPI C-PHY Editor 1', ON Query: :PLUGin:CPHYplugin:INSCal:ENABLEd? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:EMBedding:PATH:SPArAmeters[?]

Syntax	:PLUGin:CPHYplugin:EMBedding:PATH:SPArAmeters 'identifier', '<parameter>' :PLUGin:CPHYplugin:EMBedding:PATH:SPArAmeters? 'identifier'
Parameter	'Identifier': 'MIPI C-PHY Editor 1', "file path"
Description	This command helps you to define either the s-parameter (s6p) files or data (dat) files to de-embed the Output Amplifier from the AWG. You may define more than one file path locations, separated by a semi-colon (;). This query returns the present setting.

Example Command:

```
:PLUGin:CPHYplugin:EMBedding:PATH:SPArAmeters 'MIPI C-PHY Editor  
1', "C:\ProgramData\BitifEye\ValiFrame\SPArAmeter\CPhy\  
CPHY_LegacyChannel.s6p;C:\ProgramData\BitifEye\ValiFrame\  
SPArAmeter\CPhy\CPHY_StandardISChannel.s6p"
```

Query:

```
:PLUGin:CPHYplugin:EMBedding:PATH:SPArAmeters? 'MIPI C-PHY Editor  
1'
```

SCPI Commands for InterSymbol Interference Group

Table 17 SCPI Commands for InterSymbol interference Group

Command	Description under
:PLUGin::PLUGin:CPHYplugin:ISI:ENABLE[?]	For details, see :PLUGin:CPHYplugin:ISI:ENABLE[?] on page 178.
:PLUGin:CPHYplugin:ISI:NUMPorts[?]	For details, see :PLUGin:CPHYplugin:ISI:NUMPorts[?] on page 178.
:PLUGin:CPHYplugin:ISI:PATH[?]	For details, see :PLUGin:CPHYplugin:ISI:PATH[?] on page 179.
:PLUGin:CPHYplugin:ISI:PATH:A[?]	For details, see :PLUGin:CPHYplugin:ISI:PATH:A[?] on page 179.
:PLUGin:CPHYplugin:ISI:PATH:B[?]	For details, see :PLUGin:CPHYplugin:ISI:PATH:B[?] on page 180.
:PLUGin:CPHYplugin:ISI:PATH:C[?]	For details, see :PLUGin:CPHYplugin:ISI:PATH:C[?] on page 180.
:PLUGin:CPHYplugin:ISI:SCALE[?]	For details, see :PLUGin:CPHYplugin:ISI:SCALE[?] on page 180.

:PLUGin:CPHYplugin:ISI:ENABLE[?]

Syntax	:PLUGin:CPHYplugin:ISI:ENABLE 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:ISI:ENABLE? 'pluginidentifier'
Parameters	ON OFF 1 0
Description	This command enables or disables the ISI feature. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:ISI:ENABLE 'MIPI C-PHY Editor 1', ON Query: :PLUGin:CPHYplugin:ISI:ENABLE? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:ISI:NUMPorts[?]

Syntax	:PLUGin:CPHYplugin:ISI:NUMPorts 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:ISI:NUMPorts? 'pluginidentifier'
Parameters	TWOPort SIXPort
Description	This command sets the number of ports. Number of port defines number of input and output ports i.e. for 2 ports, there will be 2 input and 2 output ports. This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:ISi:NUMPorts 'MIPI C-PHY Editor 1', SIXPort
Query:
:PLUGin:CPHYplugin:ISi:NUMPorts? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:ISi:PATH[?]

Syntax :PLUGin:CPHYplugin:ISi:PATH 'pluginidentifier', <parameter>
:PLUGin:CPHYplugin:ISi:PATH? 'pluginidentifier'

Parameters S-Parameter File Path (string)

Description This command sets the file path for a file containing a 6 port (S6P) S-Parameter information.
This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:ISi:PATH 'MIPI C-PHY Editor 1', "C:\Users\user1\Documents\Keysight\M8070B\Workspaces\Default\Factory\Settings\C-PHY\SParameterFile.s6p"
Query:
:PLUGin:CPHYplugin:ISi:PATH? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:ISi:PATH:A[?]

Syntax :PLUGin:CPHYplugin:ISi:PATH:A 'pluginidentifier', <parameter>
:PLUGin:CPHYplugin:ISi:PATH:A? 'pluginidentifier'

Parameters S-Parameter File Path (string)

Description This command sets the file path for a file containing a S-Parameter information for line A. To access this folder, write "%USERPROFILE%\Documents" into a file explorer window
This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:ISi:PATH:A 'MIPI C-PHY Editor 1', 'SParameterFile.s2p'
Query:
:PLUGin:CPHYplugin:ISi:PATH:A? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:ISI:PATH:B[?]

Syntax	:PLUGin:CPHYplugin:ISI:PATH:B 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:ISI:PATH:B? 'pluginidentifier'
Parameters	S-Parameter File Path (string)
Description	This command sets the file path for a file containing a S-Parameter information for line B. To access this folder, write “%USERPROFILE%\Documents” into a file explorer window This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:ISI:PATH:B 'MIPI C-PHY Editor 1', 'SParameterFile.s2p' Query: :PLUGin:CPHYplugin:ISI:PATH:B? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:ISI:PATH:C[?]

Syntax	:PLUGin:CPHYplugin:ISI:PATH:C 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:ISI:PATH:C? 'pluginidentifier'
Parameters	S-Parameter File Path (string)
Description	This command sets the file path for a file containing a S-Parameter information for line C. To access this folder, write “%USERPROFILE%\Documents” into a file explorer window This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:ISI:PATH:C 'MIPI C-PHY Editor 1', 'SParameterFile.s2p' Query: :PLUGin:CPHYplugin:ISI:PATH:C? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:ISI:SCALE[?]

Syntax	:PLUGin:CPHYplugin:ISI:SCALE 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:ISI:SCALE? 'pluginidentifier'
Parameters	ISI Scale Factor (double)
Range	0 to 10

Description	This command sets the scaling parameter value that increases or decreases the effect of S-parameter measurement freely. This query returns the present setting.
Example	Command: <pre>:PLUGin:CPHYplugin:ISI:SCALE 'MIPI C-PHY Editor 1', 2.000e+000</pre> Query: <pre>:PLUGin:CPHYplugin:ISI:SCALE? 'MIPI C-PHY Editor 1'</pre>

SCPI Commands for Jitter Group

Table 18 SCPI Commands for Jitter Group

Command	Description under
:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:CALibrated[?]	For details, see :PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:CALibrated[?] on page 182.
:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:LANE[?]	For details, see :PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:LANE[?] on page 183.
:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:RMS[?]	For details, see :PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:RMS[?] on page 183.
:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:ENABled[?]	For details, see :PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:ENABled[?] on page 184.
:PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:ENABled[?]	For details, see :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:ENABled[?] on page 184.
:PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:LANE[?]	For details, see :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:LANE[?] on page 185.
:PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:AMPlitude[?]	For details, see :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:AMPlitude[?] on page 185.
:PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:FREQuency[?]	For details, see :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:FREQuency[?] on page 186.
:PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:CALibrated[?]	For details, see :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:CALibrated[?] on page 186.

:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:CALibrated[?]

Syntax	:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:CALibrated 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:CALibrated? 'pluginidentifier'
Parameters	ON OFF 1 0
Description	This command is used to enable or disable the Random Jitter calibrated values. This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:CALibrated 'MIPI C-PHY Editor 1', OFF
Query:
:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:CALibrated? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:LANE[?]

Syntax :PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:LANE 'pluginidentifier', <parameter>
:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:LANE? 'pluginidentifier'

Parameters ALL | DATA0 | DATA1 | DATA2

Description This command sets the lane where the Bounded Uncorrelated Jitter will be applied.
This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:LANE 'MIPI C-PHY Editor 1', DATA0
Query:
:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:LANE? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:RMS[?]

Syntax :PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:RMS 'pluginidentifier', <parameter>
:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:RMS? 'pluginidentifier'

Parameters Jitter amplitude (double)

Description This command sets the bounded uncorrelated jitter (BUJ, Random Jitter Simulation) applied to the lane in RMS.
This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:RMS 'MIPI C-PHY Editor 1', 1.00000000000000E-09

Query:

```
:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:RMS? 'MIPI C-PHY Editor 1'
```

```
:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:ENABLEd[?]
```

Syntax	:PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:ENABLEd 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:ENABLEd? 'pluginidentifier'
Parameters	ON OFF 1 0
Description	This command is used to enable or disable the Bounded Uncorrelated Jitter (Random Jitter) This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:ENABLEd 'MIPI C-PHY Editor 1', 0 Query: :PLUGin:CPHYplugin:IMPairments:JITTer:BUJ:ENABLEd? 'MIPI C-PHY Editor 1'

```
:PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:ENABLEd[?]
```

Syntax	:PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:ENABLEd 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:ENABLEd? 'pluginidentifier'
Parameters	ON OFF 1 0
Description	This command is used to enable or disable the Sinusoidal Jitter. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:ENABLEd 'MIPI C-PHY Editor 1', 1 Query: :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:ENABLEd? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:LANE[?]

Syntax	:PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:LANE 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:LANE? 'pluginidentifier'
Parameters	ALL DATA0 DATA1 DATA2
Description	This command is used to set the lane where the sinusoidal jitter will be applied. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:LANE 'MIPI C-PHY Editor 1', ALL Query: :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:LANE? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:AMPlitude[?]

Syntax	:PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:AMPlitude 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:AMPlitude? 'pluginidentifier'
Parameters	Jitter amplitude (double)
Description	This command is used to set the amount of sinusoidal jitter peak-peak applied to the lane. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:AMPlitude 'MIPI C-PHY Editor 1', 1.000e+000 Query: :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:AMPlitude? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:FREQuency[?]

Syntax	:PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:FREQuency 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:FREQuency? 'pluginidentifier'
Parameters	Frequency (double)
Range	30 KHz to 3 GHz
Description	This command is used to set the frequency of sinusoidal jitter applied to the lane. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:FREQuency 'MIPI C-PHY Editor 1', 2.000e+008 Query: :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:FREQuency? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:CALibrated[?]

Syntax	:PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:CALibrated 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:CALibrated? 'pluginidentifier'
Parameters	ON OFF 1 0
Description	This command is used to enable or disable the sinusoidal jitter calibrated values. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:CALibrated 'MIPI C-PHY Editor 1', ON Query: :PLUGin:CPHYplugin:IMPairments:JITTer:SINusoidal:CALibrated? 'MIPI C-PHY Editor 1'

SCPI Commands for Protocol Group

Table 19 SCPI Commands for Protocol Group

Command	Description under
:PLUGin:CPHYplugin:PROTOCOL:SET:DEFault?	For details, see :PLUGin:CPHYplugin:PROTOCOL:SET:DEFault on page 187
:PLUGin:CPHYplugin:PROTOCOL:PREPare:DURation[?]	For details, see :PLUGin:CPHYplugin:PROTOCOL:PREPare:DURation[?] on page 188
:PLUGin:CPHYplugin:PROTOCOL:PREAmble:BEGin:PATtern[?]	For details, see :PLUGin:CPHYplugin:PROTOCOL:PREAmble:BEGin:PATtern[?] on page 188
:PLUGin:CPHYplugin:PROTOCOL:PREAmble:BEMultiplier:PATtern[?]	For details, see :PLUGin:CPHYplugin:PROTOCOL:PREAmble:BEMultiplier:PATtern[?] on page 189
:PLUGin:CPHYplugin:PROTOCOL:PREAmble:PROGsequence:PATtern[?]	For details, see :PLUGin:CPHYplugin:PROTOCOL:PREAmble:PROGsequence:PATtern[?] on page 189.
:PLUGin:CPHYplugin:PROTOCOL:PREAmble:END:PATtern[?]	For details, see :PLUGin:CPHYplugin:PROTOCOL:PREAmble:END:PATtern[?] on page 190.
:PLUGin:CPHYplugin:PROTOCOL:SYNCword:PATtern[?]	For details, see :PLUGin:CPHYplugin:PROTOCOL:SYNCword:PATtern[?] on page 190.
:PLUGin:CPHYplugin:PROTOCOL:POST:PATtern[?]	For details, see :PLUGin:CPHYplugin:PROTOCOL:POST:PATtern[?] on page 191.
:PLUGin:CPHYplugin:PROTOCOL:POMultiplier:PATtern[?]	For details, see :PLUGin:CPHYplugin:PROTOCOL:POMultiplier:PATtern[?] on page 191.
:PLUGin:CPHYplugin:PROTOCOL:EXIT:DURation[?]	For details, see :PLUGin:CPHYplugin:PROTOCOL:EXIT:DURation[?] on page 192.
:PLUGin:CPHYplugin:PROTOCOL:WAKEup:DURation[?]	For details, see :PLUGin:CPHYplugin:PROTOCOL:WAKEup:DURation[?] on page 192.
:PLUGin:CPHYplugin:PROTOCOL:INIT:DURation[?]	For details, see :PLUGin:CPHYplugin:PROTOCOL:INIT:DURation[?] on page 192.

:PLUGin:CPHYplugin:PROTOCOL:SET:DEFault

Syntax	:PLUGin:CPHYplugin:PROTOCOL:SET:DEFault 'pluginidentifier'
Parameters	No parameters
Description	This command is used to set the protocol settings to the default values.
Example	Command: :PLUGin:CPHYplugin:PROTOCOL:SET:DEFault 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:PROTOCOL:PREPare:DURation[?]

Syntax	:PLUGin:CPHYplugin:PROTOCOL:PREPare:DURation 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:PROTOCOL:PREPare:DURation? 'pluginidentifier'
Parameters	Time in seconds (double) (for example 5.000e-008)
Range	100ps to 10ms
Description	This command is used to set the time that the transmitter drives a LP-000 state immediately before the start of the High Speed transmission. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:PROTOCOL:PREPare:DURation 'MIPI C-PHY Editor 1',7.000000000000000E-08 Query: :PLUGin:CPHYplugin:PROTOCOL:PREPare:DURation? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:PROTOCOL:PREAmble:BEGin:PATtern[?]

Syntax	:PLUGin:CPHYplugin:PROTOCOL:PREAmble:BEGin:PATtern 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:PROTOCOL:PREAmble:BEGin:PATtern? 'pluginidentifier'
Parameters	Line states (string)
Description	This command is used to define the T3-PREBEGIN pattern in high speed C-PHY pattern format. It should be a multiple of 7 UI from a minimum of 7 UI up to 448 UI. The T3-PREBEGIN pattern needs to be written in Symbols (e.g. 0, 1, 2, 3 and 4). This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:PROTOCOL:PREAmble:BEGin:PATtern 'MIPI C-PHY Editor 1', '4444444' Query: :PLUGin:CPHYplugin:PROTOCOL:PREAmble:BEGin:PATtern? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:PROTOCOL:PREAmble:BEMultiplier:PATtern[?]

Syntax	:PLUGin:CPHYplugin:PROTOCOL:PREAmble:BEMultiplier:PATtern 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:PROTOCOL:PREAmble:BEMultiplier:PATtern? 'pluginidentifier'
Parameters	Numeric value from 1 to 64
Description	This command is used to define the multiplier value between 1 and 64 to set the required HS pattern length from 7UI to 448UI for the T3-PREBEGIN pattern in high speed C-PHY pattern format. The multiplier value is a number from 1 (7UI) to 64 (448UI). This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:PROTOCOL:PREAmble:BEMultiplier:PATtern 'MIPI C-PHY Editor 1', 63 Query: :PLUGin:CPHYplugin:PROTOCOL:PREAmble:BEMultiplier:PATtern? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:PROTOCOL:PREAmble:PROGsequence:PATtern[?]

Syntax	:PLUGin:CPHYplugin:PROTOCOL:PREAmble:PROGsequence:PATtern 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:PROTOCOL:PREAmble:PROGsequence:PATtern? 'pluginidentifier'
Parameters	Line states (string)
Description	This command is used to define the optional T3-PROGSEQ pattern in High Speed C-PHY pattern format. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:PROTOCOL:PREAmble:PROGsequence:PATtern 'MIPI C-PHY Editor 1', "44444444" Query: :PLUGin:CPHYplugin:PROTOCOL:PREAmble:PROGsequence:PATtern? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:PROTOCOL:PREAmble:END:PATtern?]

Syntax	:PLUGin:CPHYplugin:PROTOCOL:PREAmble:END:PATtern 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:PROTOCOL:PREAmble:END:PATtern? 'pluginidentifier'
Parameters	Line states (string)
Description	This command is used to define the T3-PREEND pattern in High Speed C-PHY pattern format. As a default this pattern contains a total of 7 symbols of type 3. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:PROTOCOL:PREAmble:END:PATtern 'MIPI C-PHY Editor 1', '1234231' Query: :PLUGin:CPHYplugin:PROTOCOL:PREAmble:END:PATtern? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:PROTOCOL:SYNCword:PATtern[?]

Syntax	:PLUGin:CPHYplugin:PROTOCOL:SYNCword:PATtern 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:PROTOCOL:SYNCword:PATtern? 'pluginidentifier'
Parameters	Line states (string)
Description	This command is used to define the T3-SYNC pattern and is sent immediately before the beginning of High Speed transmission. By default, this pattern is 3444443. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:PROTOCOL:SYNCword:PATtern 'MIPI C-PHY Editor 1', '2134123' Query: :PLUGin:CPHYplugin:PROTOCOL:SYNCword:PATtern? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:PROTOCOL:POST:PATtern[?]

Syntax	:PLUGin:CPHYplugin:PROTOCOL:POST:PATtern 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:PROTOCOL:POST:PATtern? 'pluginidentifier'
Parameters	Line states (string)
Description	This command is used to define the T3-POST pattern and is sent immediately after the High Speed data transmission. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:PROTOCOL:POST:PATtern 'MIPI C-PHY Editor 1', '3411233' Query: :PLUGin:CPHYplugin:PROTOCOL:POST:PATtern? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:PROTOCOL:POMultiplier:PATtern[?]

Syntax	:PLUGin:CPHYplugin:PROTOCOL:POMultiplier:PATtern 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:PROTOCOL:POMultiplier:PATtern? 'pluginidentifier'
Parameters	Numeric value from 1 to 32
Description	This command is used to define the multiplier value between 1 and 32 to set the required pattern length from 7UI to 224UI for the T3-POST pattern in high speed C-PHY pattern format. The multiplier value is a number from 1 (7UI) to 32 (224UI). This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:PROTOCOL:POMultiplier:PATtern 'MIPI C-PHY Editor 1', 25 Query: :PLUGin:CPHYplugin:PROTOCOL:POMultiplier:PATtern? 'MIPI C-PHY Editor 1'

:PLUGIn:CPHYplugin:PROTOCOL:EXIT:DURATION[?]

Syntax	:PLUGIn:CPHYplugin:PROTOCOL:EXIT:DURATION 'pluginidentifier', <parameter> :PLUGIn:CPHYplugin:PROTOCOL:EXIT:DURATION? 'pluginidentifier'
Parameters	Time in seconds (double)
Range	100ps to 10ms
Description	This command is used to define the time of a LP-111 state following a High Speed burst. This query returns the present setting.
Example	Command: :PLUGIn:CPHYplugin:PROTOCOL:EXIT:DURATION 'MIPI C-PHY Editor 1', 5.00000000000000E-07 Query: :PLUGIn:CPHYplugin:PROTOCOL:EXIT:DURATION? 'MIPI C-PHY Editor 1'

:PLUGIn:CPHYplugin:PROTOCOL:WAKEUP:DURATION[?]

Syntax	:PLUGIn:CPHYplugin:PROTOCOL:WAKEUP:DURATION 'pluginidentifier',<parameter> :PLUGIn:CPHYplugin:PROTOCOL:WAKEUP:DURATION? 'pluginidentifier'
Parameters	Time in seconds (double)
Description	This command is used to set the time that the transmitter drives a Mark-1 state prior to a stop state in order to initiate an exit from ULPS. This query returns the present setting.
Example	Command: :PLUGIn:CPHYplugin:PROTOCOL:WAKEUP:DURATION 'MIPI C-PHY Editor 1', 20.00000000000000E-03 Query: :PLUGIn:CPHYplugin:PROTOCOL:WAKEUP:DURATION? 'MIPI C-PHY Editor 1'

:PLUGIn:CPHYplugin:PROTOCOL:INIT:DURATION[?]

Syntax	:PLUGIn:CPHYplugin:PROTOCOL:INIT:DURATION 'pluginidentifier', <parameter> :PLUGIn:CPHYplugin:PROTOCOL:INIT:DURATION? 'pluginidentifier'
Parameters	Time in seconds (double)

Range	100ps to 1ms
Description	This command is used to set the time that the transmitter drives a stop State (LP-111). This query returns the present setting.
Example	Command: <pre>:PLUGin:CPHYplugin:PROTocol:INIT:DURation 'MIPI C-PHY Editor 1', 5.000000000000000E-06</pre> Query: <pre>:PLUGin:CPHYplugin:PROTocol:INIT:DURation? 'MIPI C-PHY Editor 1'</pre>

SCPI Commands for Signal Interference Group

Table 20 SCPI Commands for Signal Interference Group

Command	Description under
:PLUGin:CPHYplugin:INTerference:ENABled[?]	For details, see :PLUGin:CPHYplugin:INTerference:ENABled[?] on page 194
:PLUGin:CPHYplugin:INTerference:LANE[?]	For details, see :PLUGin:CPHYplugin:INTerference:LANE[?] on page 194
:PLUGin:CPHYplugin:INTerference:LINE[?]	For details, see :PLUGin:CPHYplugin:INTerference:LINE[?] on page 195
:PLUGin:CPHYplugin:INTerference:MODE[?]	For details, see :PLUGin:CPHYplugin:INTerference:MODE[?] on page 195.
:PLUGin:CPHYplugin:INTerference:AMPLitude[?]	For details, see :PLUGin:CPHYplugin:INTerference:AMPLitude[?] on page 195.
:PLUGin:CPHYplugin:INTerference:FREQuency[?]	For details, see :PLUGin:CPHYplugin:INTerference:FREQuency[?] on page 196.

:PLUGin:CPHYplugin:INTerference:ENABled[?]

Syntax	:PLUGin:CPHYplugin:INTerference:ENABled 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:INTerference:ENABled? 'pluginidentifier'
Parameters	ON OFF 1 0
Description	This command is used to enable or disable the signal interference. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:INTerference:ENABled 'MIPI C-PHY Editor 1', 0 Query: :PLUGin:CPHYplugin:INTerference:ENABled? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:INTerference:LANE[?]

Syntax	:PLUGin:CPHYplugin:INTerference:LANE 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:INTerference:LANE? 'pluginidentifier'
Parameters	ALL DATA0 DATA1 DATA2
Description	This command is used to set the lane where the signal interference will be applied. This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:INTerference:LANE 'MIPI C-PHY Editor 1', DATA0
Query:
:PLUGin:CPHYplugin:INTerference:LANE? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:INTerference:LINE[?]

Syntax :PLUGin:CPHYplugin:INTerference:LINE 'pluginidentifier', <parameter>
:PLUGin:CPHYplugin:INTerference:LINE? 'pluginidentifier'

Parameters ALL | A | B | C

Description This command is used to set the line where the signal interference will be applied.
This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:INTerference:LINE 'MIPI C-PHY Editor 1', C
Query:
:PLUGin:CPHYplugin:INTerference:LINE? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:INTerference:MODE[?]

Syntax :PLUGin:CPHYplugin:INTerference:MODE 'pluginidentifier', <parameter>
:PLUGin:CPHYplugin:INTerference:MODE? 'pluginidentifier'

Parameters BOTH | LPower | HSpeed

Description This command is used to set the mode where the signal interference will be applied.
This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:INTerference:MODE 'MIPI C-PHY Editor 1', HSpeed
Query:
:PLUGin:CPHYplugin:INTerference:MODE? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:INTerference:AMPLitude[?]

Syntax :PLUGin:CPHYplugin:INTerference:AMPLitude 'pluginidentifier',
<parameter>
:PLUGin:CPHYplugin:INTerference:AMPLitude? 'pluginidentifier'

Parameters	Signal interference amplitude in Volts (double)
Range	0mV to 3V
Description	This command is used to set the amplitude of signal interference. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:INTerference:AMPLitude 'MIPI C-PHY Editor 1', 2.00000000000000E+00 Query: :PLUGin:CPHYplugin:INTerference:AMPLitude? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:INTerference:FREQuency[?]

Syntax	:PLUGin:CPHYplugin:INTerference:FREQuency 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:INTerference:FREQuency? 'pluginidentifier'
Parameters	Signal interference frequency in Hertz (double) (for example 4.000e+008)
Range	0Hz to 2GHz
Description	This command is used to set the frequency of signal interference This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:INTerference:FREQuency 'MIPI C-PHY Editor 1', 5.00000000000000E+08 Query: :PLUGin:CPHYplugin:INTerference:FREQuency? 'MIPI C-PHY Editor 1'

SCPI Commands for Signal Levels Group

Table 21 SCPI Commands for Signal Level Group

Command	Description under
:PLUGin:CPHYplugin:VOLTage:LANE[?]	For details, see :PLUGin:CPHYplugin:VOLTage:LANE[?] on page 197
:PLUGin:CPHYplugin:VOLTage:LINE[?]	For details, see :PLUGin:CPHYplugin:VOLTage:LINE[?] on page 197
:PLUGin:CPHYplugin:HSMoDe:VOLTage:HIGH[?]	For details, see :PLUGin:CPHYplugin:HSMoDe:VOLTage:HIGH[?] on page 198
:PLUGin:CPHYplugin:HSMoDe:VOLTage:MID[?]	For details, see :PLUGin:CPHYplugin:HSMoDe:VOLTage:MID[?] on page 198.
:PLUGin:CPHYplugin:HSMoDe:VOLTage:LOW[?]	For details, see :PLUGin:CPHYplugin:HSMoDe:VOLTage:LOW[?] on page 199.
:PLUGin:CPHYplugin:LPMoDe:VOLTage:HIGH[?]	For details, see :PLUGin:CPHYplugin:LPMoDe:VOLTage:HIGH[?] on page 199.
:PLUGin:CPHYplugin:LPMoDe:VOLTage:LOW[?]	For details, see :PLUGin:CPHYplugin:LPMoDe:VOLTage:LOW[?] on page 199.
:PLUGin:CPHYplugin:EQUalization[:VALue][?]	For details, see :PLUGin:CPHYplugin:EQUalization[:VALue][?] on page 200
:PLUGin:CPHYplugin:LEVelS:CALibrated[?]	For details, see :PLUGin:CPHYplugin:LEVelS:CALibrated[?] on page 200

:PLUGin:CPHYplugin:VOLTage:LANE[?]

Syntax	:PLUGin:CPHYplugin:VOLTage:LANE 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:VOLTage:LANE? 'pluginidentifier'
Parameters	ALL DATA0 DATA1 DATA2
Description	This command is used to set the lane where the signal Levels will be applied. This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:VOLTage:LANE 'MIPI C-PHY Editor 1', ALL Query: :PLUGin:CPHYplugin:VOLTage:LANE? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:VOLTage:LINE[?]

Syntax	:PLUGin:CPHYplugin:VOLTage:LINE 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:VOLTage:LINE? 'pluginidentifier'
Parameters	A B C ALL
Description	This command is used to set the line where the signal Levels will be applied.

This query returns the present setting.

Example

Command:

```
:PLUGin:CPHYplugin:VOLTage:LINE 'MIPI C-PHY Editor 1', ALL
```

Query:

```
:PLUGin:CPHYplugin:VOLTage:LINE? 'MIPI C-PHY Editor 1'
```

```
:PLUGin:CPHYplugin:HSMoDe:VOLTage:HIGH[?]
```

Syntax :PLUGin:CPHYplugin:HSMoDe:VOLTage:HIGH 'pluginidentifier',
<parameter>

```
:PLUGin:CPHYplugin:HSMoDe:VOLTage:HIGH? 'pluginidentifier'
```

Parameters Voltage (double)

Range 0mV to 600mV

Description This command is used to set the High Speed Mode high level (V_OHHS).

This query returns the present setting.

Example

Command:

```
:PLUGin:CPHYplugin:HSMoDe:VOLTage:HIGH 'MIPI C-PHY Editor 1',  
0.006
```

Query:

```
:PLUGin:CPHYplugin:HSMoDe:VOLTage:HIGH? 'MIPI C-PHY Editor 1'
```

```
:PLUGin:CPHYplugin:HSMoDe:VOLTage:MID[?]
```

Syntax :PLUGin:CPHYplugin:HSMoDe:VOLTage:MID 'pluginidentifier',
<parameter>

```
:PLUGin:CPHYplugin:HSMoDe:VOLTage:MID? 'pluginidentifier'
```

Parameters Voltage (double)

Range 0mV to 600mV

Description This command is used to set the High Speed Mode mid level (V_CPTX).

This query returns the present setting.

Example

Command:

```
:PLUGin:CPHYplugin:HSMoDe:VOLTage:MID 'MIPI C-PHY Editor 1', 0.003
```

Query:

```
:PLUGin:CPHYplugin:HSMoDe:VOLTage:MID? 'MIPI C-PHY Editor 1'
```

:PLUGin:CPHYplugin:HSMoDe:VOLTage:LOW[?]

Syntax	:PLUGin:CPHYplugin:HSMoDe:VOLTage:LOW 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:HSMoDe:VOLTage:LOW? 'pluginidentifier'
Parameters	Voltage (double)
Range	0mV to 600mv
Description	This command is used to set the High Speed Mode low level (V_OLHS). This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:HSMoDe:VOLTage:LOW 'MIPI C-PHY Editor 1', 0.002 Query: :PLUGin:CPHYplugin:HSMoDe:VOLTage:LOW? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:LPMoDe:VOLTage:HIGH[?]

Syntax	:PLUGin:CPHYplugin:LPMoDe:VOLTage:HIGH 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:LPMoDe:VOLTage:HIGH? 'pluginidentifier'
Parameters	Voltage (double)
Range	600mv to 1.3mV
Description	This command is used to set the High Speed Mode low level (V_OLHS). This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:LPMoDe:VOLTage:HIGH 'MIPI C-PHY Editor 1', 0.002 Query: :PLUGin:CPHYplugin:LPMoDe:VOLTage:HIGH? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:LPMoDe:VOLTage:LOW[?]

Syntax	:PLUGin:CPHYplugin:LPMoDe:VOLTage:LOW 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:LPMoDe:VOLTage:LOW? 'pluginidentifier'
Parameters	Voltage (double)
Range	-200mv to 600mV

Description This command is used to set the Low Power Mode low level (V_OL).
This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:LPMODE:VOLTage:LOW 'MIPI C-PHY Editor 1',
0.004e+000
Query:
:PLUGin:CPHYplugin:LPMODE:VOLTage:LOW? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:EQualization[:VALue][?]

Syntax :PLUGin:CPHYplugin:EQualization[:VALue] 'pluginidentifier', <parameter>
:PLUGin:CPHYplugin:EQualization[:VALue]? 'pluginidentifier'

Parameters Transmitter Equalization value in dB (double)

Range 0 dB to -10 dB

Description This command is used to set the Transmitter Equalization value.
This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:EQualization 'MIPI C-PHY Editor 1', 0.000e+000
Query:
:PLUGin:CPHYplugin:EQualization:VALue? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:LEvels:CALibrated[?]

Syntax :PLUGin:CPHYplugin:LEvels:CALibrated 'pluginidentifier', <parameter>
:PLUGin:CPHYplugin:LEvels:CALibrated? 'pluginidentifier'

Parameters ON | OFF | 1 | 0

Description This command is used to enable or disable the use of calibrated signal levels.

This query returns the present setting.

Example Command:
:PLUGin:CPHYplugin:LEvels:CALibrated'MIPI C-PHY Editor 1', 1
Query:
:PLUGin:CPHYplugin:LEvels:CALibrated? 'MIPI C-PHY Editor 1'

SCPI Commands for Skew Group

Table 22 SCPI Commands for Skew Group

Command	Description under
:PLUGin:CPHYplugin:SKEW:AWGA:CH{1:4}[?]	For details, see :PLUGin:CPHYplugin:SKEW:AWGA:CH{1:4}[?] on page 201
:PLUGin:CPHYplugin:SKEW:AWGB:CH{1:4}[?]	For details, see :PLUGin:CPHYplugin:SKEW:AWGB:CH{1:4}[?] on page 201
:PLUGin:CPHYplugin:SKEW:AWGC:CH{1:4}[?]	For details, see :PLUGin:CPHYplugin:SKEW:AWGC:CH{1:4}[?] on page 202

:PLUGin:CPHYplugin:SKEW:AWGA:CH{1:4}[?]

Syntax	:PLUGin:CPHYplugin:SKEW:AWGA:CH{1:4} 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:SKEW:AWGA:CH{1:4}? 'pluginidentifier'
Parameters	Skew in seconds (double)
Description	This command is used to set the skew in AWGA module on channels{1:4} This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:SKEW:AWGA:CH3 'MIPI C-PHY Editor 1', 4.00000000000000E-12 Query: :PLUGin:CPHYplugin:SKEW:AWGA:CH1? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:SKEW:AWGB:CH{1:4}[?]

Syntax	:PLUGin:CPHYplugin:SKEW:AWGB:CH{1:4} 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:SKEW:AWGB:CH{1:4}? 'pluginidentifier'
Parameters	Skew in seconds (double)
Description	This command is used to set the skew in AWGB module on channels{1:4} This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:SKEW:AWGB:CH3 'MIPI C-PHY Editor 1', 5.00000000000000E-12 Query: :PLUGin:CPHYplugin:SKEW:AWGB:CH1? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:SKEW:AWGC:CH{1:4}[?]

Syntax	:PLUGin:CPHYplugin:SKEW:AWGC:CH{1:4} 'pluginidentifier', <parameter> :PLUGin:CPHYplugin:SKEW:AWGC:CH{1:4}? 'pluginidentifier'
Parameters	Skew in seconds (double)
Description	This command is used to set the skew in AWGC module on channels{1:4} This query returns the present setting.
Example	Command: :PLUGin:CPHYplugin:SKEW:AWGC:CH3 'MIPI C-PHY Editor 1', 1.00000000000000E-12 Query: :PLUGin:CPHYplugin:SKEW:AWGC:CH1? 'MIPI C-PHY Editor 1'

Other SCPI Commands

Table 23 Other SCPI Commands

Command	Description under
:PLUGin:CPHYplugin:LINK:CONNect[:EXECute]	For details, see :PLUGin:CPHYplugin:LINK:CONNect[:EXECute] on page 204
:PLUGin:CPHYplugin:LINK:DISConnect	For details, see :PLUGin:CPHYplugin:LINK:DISConnect on page 204
:PLUGin:CPHYplugin:ABORT	For details, see :PLUGin:CPHYplugin:ABORT on page 204.
:PLUGin:CPHYplugin:TRIGger[:EXECute]	For details, see :PLUGin:CPHYplugin:TRIGger[:EXECute] on page 204.
:PLUGin:CPHYplugin:REStart	For details, see :PLUGin:CPHYplugin:REStart on page 204.
:PLUGin:CPHYplugin:PARAmeters:RESet	For details, see :PLUGin:CPHYplugin:PARAmeters:RESet on page 205.
:PLUGin:CPHYplugin:StARt	For details, see :PLUGin:CPHYplugin:StARt on page 205.
:PLUGin:CPHYplugin:StOP	For details, see :PLUGin:CPHYplugin:StOP on page 205.
:PLUGin:CPHYplugin:RuN:StAtus?	For details, see :PLUGin:CPHYplugin:RuN:StAtus? on page 205.
:PLUGin:CPHYplugin:RuN:PRoGress?	For details, see :PLUGin:CPHYplugin:RuN:PRoGress? on page 206.
:PLUGin:CPHYplugin:RuN:MESSage?	For details, see :PLUGin:CPHYplugin:RuN:MESSage? on page 206.
:PLUGin:CPHYplugin:RuN:LoG?	For details, see :PLUGin:CPHYplugin:RuN:LoG? on page 206.
:PLUGin:CPHYplugin:NEw	For details, see :PLUGin:CPHYplugin:NEw on page 206.
:PLUGin:CPHYplugin:DELeTe	For details, see :PLUGin:CPHYplugin:DELeTe on page 207.
:PLUGin:CPHYplugin:CATalog?	For details, see :PLUGin:CPHYplugin:CATalog? on page 207.
Non Blocking Commands	
:PLUGin:CPHYplugin:APPLy[:ALL]	For details see, :PLUGin:CPHYplugin:APPLy[:ALL] on page 209.
:PLUGin:CPHYplugin:APPLy:DELays	For details, see :PLUGin:CPHYplugin:APPLy:DELays on page 209.
:PLUGin:CPHYplugin:RESet	For details, see :PLUGin:CPHYplugin:RESet on page 209.

:PLUGin:CPHYplugin:LINK:CONNect[:EXECute]

Syntax	:PLUGin:CPHYplugin:LINK:CONNect[:EXECute] 'pluginidentifier'
Parameters	No parameter
Description	This command is used to connect the C-PHY Editor to the instruments.
Example	Command: :PLUGin:CPHYplugin:LINK:CONNect:EXECute 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:LINK:DISConnect

Syntax	:PLUGin:CPHYplugin:LINK:DISConnect 'pluginidentifier'
Parameters	No parameter
Description	This command is used to disconnect the C-PHY Editor from the instruments.
Example	Command: :PLUGin:CPHYplugin:LINK:DISConnect 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:ABORt

Syntax	:PLUGin:CPHYplugin:ABORt 'pluginidentifier'
Parameters	No parameter
Description	This command is used to abort the current running operation.
Example	Command: :PLUGin:CPHYplugin:ABORt 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:TRIGger[:EXECute]

Syntax	:PLUGin:CPHYplugin:TRIGger[:EXECute] 'pluginidentifier'
Parameters	No parameter
Description	This command sends a trigger to start the sequence.
Example	Command: :PLUGin:CPHYplugin:TRIGger:EXECute 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:REStart

Syntax	:PLUGin:CPHYplugin:REStart 'pluginidentifier'
Parameters	No parameter
Description	This command restarts the sequence by stopping and starting the AWGs.

Example Command:
:PLUGIn:CPHYplugin:REStArT 'MIPI C-PHY Editor 1'

:PLUGIn:CPHYplugin:PARAmeters:RESet

Syntax :PLUGIn:CPHYplugin:PARAmeters:RESet 'pluginidentifier'
Parameters No parameter
Description This command resets the parameters to the default value without causing a waveform recalculation.
Example Command:
:PLUGIn:CPHYplugin:PARAmeters:RESet 'MIPI C-PHY Editor 1'

:PLUGIn:CPHYplugin:STARt

Syntax :PLUGIn:CPHYplugin:STARt 'pluginidentifier'
Parameters No parameter
Description This command starts the AWGs and enables the outputs.
Example Command:
:PLUGIn:CPHYplugin:STARt 'MIPI C-PHY Editor 1'

:PLUGIn:CPHYplugin:STOP

Syntax :PLUGIn:CPHYplugin:STOP 'pluginidentifier'
Parameters No parameter
Description This command stops the AWGs and disables the outputs.
Example Command:
:PLUGIn:CPHYplugin:STOP 'MIPI C-PHY Editor 1'

:PLUGIn:CPHYplugin:RUN:STATus?

Syntax :PLUGIn:CPHYplugin:RUN:STATus? 'pluginidentifier'
Parameters No parameter
Description This query returns the value '1' or '0' if some waveform calculation or calibration is currently occurring. If "1" is returned, it means that some operation is currently begin executed by the plug-in. If "0" is returned, it means no operations is currently begin executed.
Example Command:
:PLUGIn:CPHYplugin:RUN:STATus? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:RUN:PROGress?

Syntax	:PLUGin:CPHYplugin:RUN:PROGress? 'pluginidentifier'
Parameters	No parameter
Description	The query returns the progress of the current active operation begin executed by the plug-in (waveform calculation or calibration). The progress return is contained between 0 and 1.
Example	Command: :PLUGin:CPHYplugin:RUN:PROGress? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:RUN:MESSage?

Syntax	:PLUGin:CPHYplugin:RUN:MESSage? 'pluginidentifier'
Parameters	No parameter
Description	The query returns the current status message of the plug-in. The possible status message can be "Running", "Finished", "NotStarted", "Error" and "Stopped".
Example	Command: :PLUGin:CPHYplugin:RUN:MESSage? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:RUN:LOG?

Syntax	:PLUGin:CPHYplugin:RUN:LOG? 'pluginidentifier'
Parameters	No parameter
Description	This query returns the logs for the identifier 'pluginidentifier'.
Example	Command: :PLUGin:CPHYplugin:RUN:LOG? 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:NEW

Syntax	:PLUGin:CPHYplugin:NEW 'pluginidentifier'
Parameters	No parameter
Description	This command creates a new plug-in instance with the instance name given by 'pluginidentifier'.
Example	Command: :PLUGin:CPHYplugin:NEW 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:DELeTe

Syntax	:PLUGin:CPHYplugin:DELeTe 'pluginidentifier'
Parameters	No parameter
Description	This command deletes the plug-in instance with name given by 'pluginidentifier'.
Example	Command: :PLUGin:CPHYplugin:DELeTe 'MIPI C-PHY Editor 1'

:PLUGin:CPHYplugin:CATalog?

Syntax	:PLUGin:CPHYplugin:CATalog?
Parameters	No parameter
Description	This query returns the list of created plug-in instances.
Example	Command: :PLUGin:CPHYplugin:CATalog? 'MIPI C-PHY Editor 1'

Non Blocking Commands

A non-blocking command is one that runs asynchronously when you enter the command request. You must use other methods to achieve synchronization. After requesting a non-blocking SCPI command, you must run a query on the SCPI Editor to check the progress (:PLUGin:CPHYplugin:RUN:PROGress?) and to obtain the run status (:PLUGin:CPHYplugin:RUN:STATus?). If the progress is not 100% (where 100% is represented as 1.000000000E+00) and if the run status is 1, it means that the Editor is still processing the non-blocking command request. Running non-blocking commands does not allow parallel processing, that is, only one non-blocking command is processed at a given time.

To stop a non-blocking command from running further, enter a request for the abort command (:PLUGin:CPHYplugin:ABORT).

For example:

Run the non-blocking command :PLUGin:CPHYplugin:APPLY to initiate the process of waveform calculation (see [Figure 67](#)).

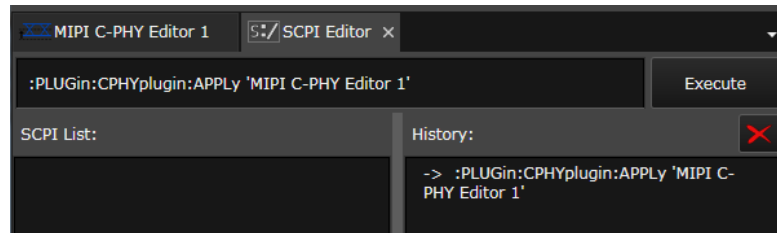


Figure 67 SCPI Command for the waveform calculation

Since the non-blocking command runs asynchronously, if you switch to the C-PHY Editor, notice that the process of waveform calculation is running and you cannot simultaneously run any other command on the SCPI Editor (see [Figure 68](#)).

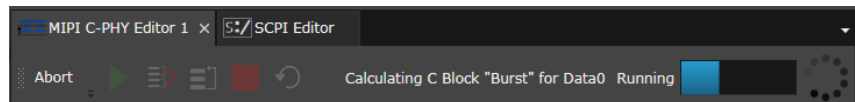


Figure 68 Process of waveform calculation

Run the :PLUGin:CPHYplugin:ABORT command if you wish to stop the :PLUGin:CPHYplugin:APPLY command.

Following commands come under the category of non-blocking SCPI commands:

:PLUGIn:CPHYplugin:APPLy[:ALL]

Syntax	:PLUGIn:CPHYplugin:APPLy[:ALL] 'pluginidentifier'
Parameters	No parameter
Description	This command is used to trigger a waveform calculation with the current settings and applies the waveform on the AWGs. The AWGs are automatically started.
Example	Command: :PLUGIn:CPHYplugin:APPLy:ALL 'MIPI C-PHY Editor 1'

:PLUGIn:CPHYplugin:APPLy:DELays

Syntax	:PLUGIn:CPHYplugin:APPLy:DELays 'pluginidentifier'
Parameters	No parameter
Description	This command is used to apply the delays and skews. It tries to apply the delays and skews using the hardware resources first. If it is not possible to allocate the delay using the hardware resources, the waveform is recalculated.
Example	Command: :PLUGIn:CPHYplugin:APPLy:DELays 'MIPI C-PHY Editor 1'

:PLUGIn:CPHYplugin:RESet

Syntax	:PLUGIn:CPHYplugin:RESet 'pluginidentifier'
Parameters	No parameter
Description	This command is used to reset the plug-into its default values.
Example	Command: :PLUGIn:CPHYplugin:RESet' MIPI C-PHY Editor 1'

Remote Queries to find Parameter ranges (maximum and minimum limits)

The C-PHY Editor plug-in consists of several parameters, where you may define a value only within a permissible range. The C-PHY Editor plug-in automatically configures the ranges for such parameters based on certain options you may select on the user interface or remotely.

On the C-PHY Editor plug-in GUI, you may simply hover the mouse cursor over a specific parameter's value field to find its maximum and minimum permissible values. However, when working remotely, you may have to check the permissible ranges before configuring the value of a parameter.

While running a query for a parameter returns its current value, you can add the **MAX** and **MIN** identifiers in your query (separated by a comma after the plug-in identifier) to find the maximum and minimum value, respectively, for that parameter.

Consider the following examples to understand how to use these identifiers:

1 Checking **TX EQ** parameter range under **Signal Levels**:

Running the **:PLUGin:CPHYplugin:EQualization[:VALue]?** query returns the current value configured for the **TX EQ** parameter under **Signal Levels**.

```
-> :PLUGin:CPHYplugin:EQualization:VALue? 'MIPI C-PHY Editor 1'
<- 0.000000000000000E+00
```

To find the maximum value allowed for the **TX EQ** parameter, add the **MAX** identifier to the query separated by a comma.

```
-> :PLUGin:CPHYplugin:EQualization:VALue? 'MIPI C-PHY Editor 1',
MAX
<- 0.000000000000000E+00
```

Similarly, to find the minimum value allowed for the **TX EQ** parameter, add the **MIN** identifier to the query separated by a comma.

```
-> :PLUGin:CPHYplugin:EQualization:VALue? 'MIPI C-PHY Editor 1',
MIN
<- -1.000000000000000E+01
```

When converted from the scientific notation to numeric values, the maximum and minimum limit for the **TX EQ** parameter are 0dB to -10dB.

You may now configure the value for the **TX EQ** parameter remotely using **:PLUGin:CPHYplugin:EQualization[:VALue]** command within the permissible values.

2 Checking **LP Data Rate** parameter range under **Data Rates & Transition Times**:

Running the **:PLUGin:CPHYplugin:LPMoDe:DAtRate?** query returns the current value configured for the **LP Data Rate** parameter under **Data Rates & Transition Times**.

```
-> :PLUGin:CPHYplugin:LPMoDe:DAtRate? 'MIPI C-PHY Editor 1'
<- 10.000000000000000E+07
```

To find the maximum value allowed for the **LP Data Rate** parameter, add the **MAX** identifier to the query separated by a comma.

```
-> :PLUGin:CPHYplugin:LPMoDe:DAtRate? 'MIPI C-PHY Editor 1',
MAX
<- 1.500000000000000E+08
```

Similarly, to find the minimum value allowed for the **LP Data Rate** parameter, add the **MIN** identifier to the query separated by a comma.

```
-> :PLUGin:CPHYplugin:LPMoDe:DAtRate? 'MIPI C-PHY Editor 1', MIN
<- 1.000000000000000E+06
```

When converted from the scientific notation to numeric values, the minimum and maximum values for the **LP Data Rate** parameter are 1 Mb/s and 150 Mb/s.

You may now configure the value for the **LP Data Rate** parameter remotely using **:PLUGin:CPHYplugin:LPMoDe:DAtRate** command within the permissible values.

Therefore, to find out the permissible ranges for parameters with numeric values, add the **MAX** and **MIN** identifiers in your query (separated by a comma after the plug-in identifier) to find the maximum and minimum value, respectively, for that parameter.

