

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*

M5302A PXIe LVDS Digital IO Module

User's Guide

KEYSIGHT TECHNOLOGIES

Notices

Copyright Notice

© Keysight Technologies 2021

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

UNIX is a registered trademark of UNIX System Laboratories in the U.S.A. and other countries. Target is copyrighted by Thru-Put Systems, Inc.

Manual Part Number

M5302-91003

Edition

1.0, March 2021

Available in electronic format only

Published by

Keysight Technologies, Inc.
1900 Garden of the Gods Road
Colorado Springs, CO 80907 USA

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.

Declaration of Conformity

Declarations of Conformity for this product and for other Keysight products may be downloaded from the Web. Go to http://www.keysight.com/go/conformity and click on “Declarations of Conformity.” You can then search by product number to find the latest Declaration of Conformity.

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 Agreement (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 Information

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 Understanding M5302A LVDS Digital IO Module's Theory of Operation

Overview on M5302A LVDS Digital IO modules / 8

Using M5302A pins / 9

LVDS IO / 9

SMB IOs / 9

PXIe triggers / 9

External DDR4 Memory / 9

Power Supply / 10

2 Using the KS2201A PathWave Test Sync Executive software

Licensing for KS2201A PathWave Test Sync Executive software / 12

Understanding operation flow in KS2201A HVI API / 13

Working with KS2201A PathWave Test Sync Executive software / 14

Overview on HVI technology / 14

Understanding the HVI elements used in M5302x API / 15

Description of various HVI elements / 15

HVI Engine 15

HVI Actions 15

HVI Events 15

HVI Trigger 15

HVI Instructions 15

FPGA sandbox registers 16

HVI timings in M5302A / 17

HVI-Engine Clock / 17

HVI Sync Resources timing / 17

HVI FPGA User Sandbox instructions timing / 17

HVI Trigger timings / 18

Trigger groups 18

TriggerWrite instruction timings 19

*Trigger Write Groups 19

Understanding 'Ceiling' of values 19

HVI Action timings / 20

Action groups 20

ActionExecute instruction timings 20

HVI Event timings / 21

Event groups 21

HVI Local Instructions Timing for M5302A / 23

Arithmetic Logic Unit (ALU) Instructions 23

3 Using the PathWave FPGA Board Support Package (BSP)

Licensing for PathWave FPGA BSP support / 26

Features in PathWave FPGA / 27

Working with PathWave FPGA software / 28

Understanding Partial Configuration (PR) / 29

Using BSP with PathWave FPGA software / 31

Understanding BSP composition / 31

Generating a k7z file using PathWave FPGA BSP / 33

Loading k7z file onto modules / 40

4 Using Keysight M5302A PXIe Module SFP Software

Installing the M5302A PXIe Module SFP Software / 42

Launching the M5302A PXIe Module SFP software / 43

Familiarizing with the SFP software / 45

Menus / 45

I/O Banks / 46

Status Indicators / 46

Understanding Menu options / 47

File menu options / 47

To save a connection: 47

To delete a connection: 48

View menu options / 48

Utilities menu options / 49

Tools menu options / 49

To clear the driver call logs: 50

To copy and paste call history from the driver call log: 50

To save a copy of the Driver Call Log to a file: 51

Help menu options / 51

Using the I/O Banks / 52

LVDS I/O / 53

SMB IO / 53

Low Speed General Purpose IO / 54

PXI IO / 54

Performing specific tasks in the M5302x SFP Software / 55

Connecting to Modules / 55

Connecting via Simulation mode 56
Checking Instrument Status / 57
Resetting to Default Factory Settings / 57
Viewing Errors log / 58
Conducting a Self Test / 59
Setting Calibration Options / 60
Setting Passphrase 61
Toggling Signal state / 61
Updating Firmware / 63
Upgrading Hardware Options / 64
Redeeming instructions 64
Understanding Host Information 65

5 Using M5302x API functions in sample programs

Implementing M5302x API functions – Sample Programs / 68
Sample program to perform & check toggling on SMB IO Channels / 68
Sample program to load a Sandbox design onto an M5302A module / 70
Sample program to create an HVI sequence for an SMB trigger / 73

6 Documentation References

Accessing related Help documents for M5302A / 78
Links to other documents / 79

Index

1. Understanding M5302A LVDS Digital IO Module's Theory of Operation

Overview on M5302A LVDS Digital IO modules 8
Using M5302A pins 9

Section 1.1: Overview on M5302A LVDS Digital IO modules

The M5302A is a digital IO PXIe card with a large Xilinx UtrascalePlus FPGA. The FPGA includes a reconfigurable sandbox region, which you can program using the Keysight KF9000A PathWave FPGA software. The FPGA contains a Hardware Virtual Instrument (HVI) interface that can be used to control the M5302A module in a time deterministic fashion. For more information on the HVI technology, refer to the *KS2201A PathWave Test Sync Executive User Guide*.

The core clock is 300MHz and is locked to the PXIE Clk100 interface.

The module has 28 LVDS bidirectional IOs located on the front panel, 8 LVCMOS SMB bidirectional IOs located on the front panel, 8 PXI bidirectional backplane triggers, and the PXI STAR backplane trigger.

The SMB, PXI, and STAR signals are routed directly to the HVI engine in the FPGA and can be controlled through HVI sequences, the soft front panel, or the module API.

The LVDS IOs are routed though a user-reconfigurable sandbox region. By default, the LVDS IOs are controlled through HVI sequences, the soft front panel, or the module API. Because the LVDS IOs a routed through the sandbox, they can also be controlled by circuits designed in the PathWave FPGA environment.

The module has 1 GB of external DDR4 memory.

The M5302A digital IO modules can be controlled using either the M5302x SFP software or the M5302x API.

To know about the functionality of the M5302x SFP, refer to "Using Keysight M5302A PXIe Module SFP Software" on page 41.

To know about the various API functions related to M5302A module, refer to one or more Help files, as listed in "Accessing related Help documents for M5302A" on page 78.

For a sample program on how the M5302x API works, see "Sample program to perform & check toggling on SMB IO Channels" on page 68.

Section 1.2: Using M5302A pins

1.2.1: LVDS IO

The M3202 has 28 LVDS IOs on the front panel. Each IO has a 4:1 SERDES. So, with the 300 MHz core clock, the LVDS IOs can transmit and receive data at 1.2 Gbps. The LVDS IOs instantiate Xilinx IDELAYs and ODELAYs along with a bit slip circuit that can slip up to seven 300 MHz clocks. The delays and bit slips give the LVDS IOs a delay adjustment range of 10 ns.

The LVDS lines are 100 Ω differential pairs connected directly from the FPGA pins to the 100-pin front panel connector.

The LVDS IO connector provides 500 mA of 5 V along with four low-speed control lines that are used to power and control an external user design board.

An IO protection scheme is implemented on the LVDS IOs to help you prevent damage to the FPGA pins. To prevent damage, the FPGA must be powered up when the inputs are driven. When stage 2 is loaded, the FPGA drives its *LVDS_TX_DISABLEen_OUT* pin high indicating that it is ready to accept inputs. The FPGA will not drive its LVDS outputs until *LVDS_TX_DISABLEen_IN* is driven high by the device receiving the FPGA LVDS outputs. The protection feature can be disabled from the front panel or by setting the 'differential_channels.output_safety' attribute to 'DISABLED' in Python.

The LVDS IOs go back and forth across the sandbox, which you can intercept and control in a PathWave FPGA design. By default, the LVDS IOs are connected directly to HVI in the static region of the FPGA.

1.2.2: SMB IOs

The IOs are connected from the FPGA to the front panel IOs through a Dual-Bit Dual-Supply Bus Transceiver. Refer to https://www.ti.com/lit/ds/symlink/sn74lvc2t45.pdf for more information about this transceiver. Note that the SMB IOs have a 10 ns delay adjustment range.

1.2.3: PXIe triggers

The PXIe triggers are connected from the FPGA to the backplane thorough a CPLD and have no delay adjustment.

1.2.4: External DDR4 Memory

The M5302 has 8 Gbits of DDR4 memory. The memory is accessed via the DMA controller and AXI crossbars on the FPGA.

1.2.5: Power Supply

The M5302A FPGA has a large sandbox region, which allows you to possibly build the sandbox to such an extent, where it can dissipate more power than the chassis cooling system can handle. This may cause the M5302A module to shut down. Power dissipation is design dependent. The power dissipation of the FPGA must be limited to 42 W. This allows 30 W to be dissipated in the sandbox.

Refer to the *M5302A PXIe Digital IO Module Data Sheet* to know the values of power consumption at 12 V and 3.3 V ratings.

For typical sandbox designs, the maximum power is achieved when the utilization in the sandbox is about 30%. Utilization can be measured with the following procedure:

1 In Vivado, open the *post_impl.dcp* design checkpoint generated by PathWaveFPGA. The checkpoint is located in the build checkpoints directory.

2 Run the tcl command: *report_utilization -cells [get_cells instr_part/inst_application_partition/inst_sandbox].*

3 Refer to the CLB Logic Utilization table, as shown in Table 1. If the *CLB Register* utilization is above 30%, the power dissipation will most likely exceed the limits.

Table 1 CLB Logic Utilization table

Site type	Used	Fixed	Available	Utilization percentage (%)
CLB LUTs	106821	0	521456	20.49
▪ LUT as Logic	106297	0	521456	20.38
▪ LUT as Memory	524	0	160928	0.33
▪ LUT as Distributed RAM	0	0		
▪ LUT as Shift Register	524	0		
CLB Registers	330756	0	1042912	31.71
▪ Register as Flip Flop	330756	0	1042912	31.71
▪ Register as Latch	0	0	1042912	0.00
CARRY8	101	0	65182	0.15
F7 Muxes	153	0	260728	0.06
F8 Muxes	64	0	130364	0.05
F9 Muxes	0	0	65182	0.00

The module power dissipation can be measured with the KtM5302xCurrentDraw() and KtM5302xModule.Temperature APIs, respectively.

2. Using the KS2201A PathWave Test Sync Executive software

Licensing for KS2201A PathWave Test Sync Executive software 12
Understanding operation flow in KS2201A HVI API 13
Working with KS2201A PathWave Test Sync Executive software 14
Understanding the HVI elements used in M5302x API 15
HVI timings in M5302A 17

This chapter provides an introduction to the KS2201A PathWave Test Sync Executive Software and describes its implementation in the M5302x API. For detailed information about the elements in KS2201A PathWave Test Sync Executive Software and its API, refer to the *KS2201A PathWave Test Sync Executive User Guide*.

Section 2.1: Licensing for KS2201A PathWave Test Sync Executive software

Hardware license option (-HV2) on the M5302A modules

The M5302A modules support HVI technology. During procurement, you may choose to procure the *-HV2* hardware option. The hardware license option *-HV2* enables programming the module using the PathWave Test Sync Executive software.

Each M5302A card is shipped with the newest version of firmware, which supports the KS2201A PathWave Test Sync Executive software. For more information regarding the supported firmware and software versions, refer to the *M5302A Software Startup Guide*.

Software license option for the KS2201A software

Refer to the *KS2201A PathWave Test Sync Executive User Guide* to know about the licenses that you must procure for the KS2201A PathWave Test Sync Executive software.

Section 2.2: Understanding operation flow in KS2201A HVI API

Table 2 summarizes the main operations necessary in an HVI design as performed in the KS2201A HVI API use model.

Table 2 Operation flow in KS2201A HVI API

Operations	KS2201A HVI API Use Model
HVI Design Flow	Application code must import the "keysight_pathwave_hvi" library to use the HVI API. HVI sequences can be created using programs directly into the application code without importing external files.
HVI Sequence	KtHviSequence class enables you to create a Local HVI sequence using programs that run "locally" on a specific HVI engine in a specific instrument. Local Sequences are accessed using 'SyncMultiSequenceBlock' statement placed in a SyncSequence (KtHviSyncSequence). The HVI top sequence is a SyncSequence that contains SyncStatements.
HVI SyncSequence	KtHviSyncSequence class enables you to add synchronized operations (Sync Statements) common to all HVI engines within the HVI instance. The HVI top sequence is a SyncSequence that contains SyncStatements. Local instructions are added and executed within Local Sequences that can be accessed by adding a 'SyncMultiSequenceBlock' in a SyncSequence.
HVI Resources (Chassis, Triggers, M9031A modules, and so on)	HVI resources can be configured using "KtHviPlatform" class and all the classes inside it.
Program HVI Sequences	You may program both HVI SyncSequences and HVI (Local) Sequences with the API methods add_XXX(), where 'XXX' is the statement name.
HVI Compile, Load, Run	API SW methods can compile the sequence using (hvi.compile()), load it to hardware using (hvi.load_to_hw()), run the sequence using (hvi.run()).

Section 2.3: Working with KS2201A PathWave Test Sync Executive software

The KS2201A PathWave Test Sync Executive software has been introduced to enhance the functionalities of one or more PXIe modules both individually and interactively. This software has an API based environment for developing and running programs with a new generation of Keysight's Hard Virtual Instrument (HVI) technology. The KS2201A software enables programmatic development and execution of synchronous real-time operations across multiple instruments. It enables you to program multiple instruments together so they can act together with other instruments, like one instrument.

2.3.1: Overview on HVI technology

Keysight's Hardware Virtual Instrumentation (HVI) technology provides the capability to create time-deterministic execution sequences with precise synchronization by deploying FPGA hardware simultaneously among the constituent instruments. This makes the technology a powerful tool in MIMO systems, such as massive-scale quantum control networks.

A virtual instrument may be considered to function like any other instrument in the system; its main objective being to digitally sequence events and instructions in the application while synchronizing multiple modules. This instrument, (referred to in this document as the HVI instrument), accomplishes this by running one or more "engines" synchronously by referencing a common digital clock that all instruments (engines) operate on.

The KS2201A PathWave Test Sync Executive software provides you with the capability of designing HVI sequences using an Application Programming Interface (API) available in Python coding language. The HVI Application Programming Interface (API) is the set of programming classes and methods that allows you to create and program an HVI instance. HVI API currently supports Python v3.7. The HVI core functionality is extended by the PXIe M5302A modules using the M5302x API. The core HVI features and the API extensions that are specific to M5302A, allow a heterogeneous array of instruments and resources to coexist on a common framework.

All the PXIe M5302A modules support HVI technology. When Keysight M5302A software is installed on a PXI system, it installs the drivers required to interact with the M5302x series modules. The classes in the Keysight M5302x API Library contain HVI add-on interfaces provided as an extension of the instrument.

The primary HVI elements defined for the M5302x API and the corresponding API functions are described in the following sections.

Section 2.4: Understanding the HVI elements used in M5302x API

The HVI Core API exposes all HVI functions and defines base interfaces and classes, which are used to create an HVI, control the hardware execution flow, and operate with data, triggers, events and actions, but it alone does not include the ability to control operations specific to the M5302x product family. It is the HVI instrument extensions specific to M5302A modules that enable instrument functionalities in an HVI. Such functions are exposed by the module specific add-on HVI definitions. The M5302x API describes the instrument specific resources and operations that can be executed or used within HVI sequences. See "Sample program to create an HVI sequence for an SMB trigger" on page 73 for an example, where HVI is implemented in M5302x API.

Refer to the *KS2201A PathWave Test Sync Executive User Guide* to know more about the HVI Python API.

2.4.1: Description of various HVI elements

HVI Engine

An HVI Engine block controls the functions of the instrument and the timing of operations. For HVI to control an M5302A module, the latter requires an HVI Engine. The HVI Engine is included directly in the instrument hardware or it can be programmed using the M5302x API into the Field programmable Gate Array (FPGA) on each module. The HVI Engine executes sequences, which are made up of Statements.

HVI Actions

An HVI Action is defined for module-specific operations.

HVI Events

An HVI Event is defined to occur when specific conditions are met during module-specific operations.

HVI Trigger

An HVI Trigger is defined to activate the logic signal triggering source and perform various triggering operations, which are shared between instruments to initiate module-related operations, communicate states or other information.

HVI Instructions

An HVI Instruction is defined to configure various settings related to the module. There are two types of HVI instructions:

- Product specific (custom) HVI instructions—can change a module's setting (such as amplitude, frequency, etc.) or trigger a functionality in the module (such as output a waveform, trigger a data acquisition, etc.).
- HVI core instructions (general purpose)—provide global, non-module specific or custom functions, such as register arithmetic, read/write general purpose I/O triggers, execution actions, etc.

FPGA sandbox registers

For the modules that contain an FPGA with a user-configurable sandbox, HVI can, potentially (if the configuration of the module allows it), access (read/write) the registers that you define in that sandbox. To accomplish this, you must obtain the ".k7z" file for the FPGA sandbox, generated by the PathWave FPGA application. This file contains all the necessary information to access the registers by name.

Section 2.5: HVI timings in M5302A

This section displays the HVI actions, events and instructions along with the corresponding latency measured for M5302A modules. For the description of each term listed in the tables below, refer to *Chapter 7: HVI Time Management and Latency* in the *KS2201A PathWave Test Sync Executive User Manual*.

2.5.1: HVI-Engine Clock

The HVI Engine clock period is key parameter for the sequence execution timing inside a module. All HVI-native instructions are documented in clock cycles and the actual execution timing in each instrument depends on its value.

For the M5302A module, the HVI engine clock is:

HVI Engine Clock Frequency (MHz)	Period (cycles)	Period (ns)
300	1	$3.\overline{3}$

2.5.2: HVI Sync Resources timing

To calculate latency, this timing parameter in the HVI statements include the "*Instrument_SyncResources_Latency*" value, which is specific for each instrument/module being used.

For the M5302A module, the "*Instrument_SyncResources_Latency*" value is:

Name	Timing value (ns)
Instrument_SyncResources_Latency	$3.\overline{3}$

2.5.3: HVI FPGA User Sandbox instructions timing

The FPGA registers and Memory map access latency from HVI for the M5302A modules is measured using the *Instrument_HVI_FPGA_Latency* parameter.

For the M5302A module, the "*Instrument_HVI_FPGA_Latency*" value is:

Name	Timing value (ns)
Instrument_HVI_FPGA_Latency	$6.\overline{6}$

Table 3 summarizes the timing for all FPGA read/write instructions. Refer to *Chapter 7: HVI Time Management and Latency* in the *KS2201A PathWave Test Sync Executive User Manual* for more information on the formula used to derive at these values.

Table 3 Minimum delay required (in nanoseconds) for FPGA User Sandbox instructions

HVI Instructions	Execution time (ns)	Fetch time (ns)
FpgaArrayRead	26.66	3.3
FpgaArrayRead (Address from HviRegister)	33.33	3.3
FpgaArrayWrite	13.33	3.3
FpgaArrayWrite (Address or data from HviRegister)	20	3.3
FpgaRegisterRead	26.66	3.3
FpgaRegisterWrite	13.33	3.3
FpgaRegisterWrite (Address or data from HviRegister)	20	3.3

NOTE

Consecutive FPGA read instructions must be issued with at least 1 cycle of delay between them.

If an FPGA instruction that uses an HVI register is issued before an FPGA instruction that does not use an HVI register, the delay between both instructions must be at least 3 cycles.

2.5.4: HVI Trigger timings

When writing HVI instructions, any number of TriggerIOs can be written at the same time. TriggerIOs are organized in groups of 16. The Execution time and Fetch time of the instruction depends on the number of different TriggerIO groups included in the instruction (#TriggerIOGroups).

Trigger groups

Table 4 Elements in Trigger IO Groups

IO Groups	Signals
Group 1	PXI_TRIG0, PXI_TRIG1, PXI_TRIG2, PXI_TRIG3, PXI_TRIG4, PXI_TRIG5, PXI_TRIG7, PXI_TRIG7, PXI_STAR0, SMB1, SMB2, SMB3, SMB4, SMB5, SMB6, SMB7
Group 2	SMB8, LVDS0 – LVDS14
Group 3	LVDS15 – LVDS27

TriggerWrite instruction timings

Table 5 Minimum delay required (in nanoseconds) for TriggerWrite instructions

HVI TriggerWrite Instructions	Execution time (ns)	Fetch time (ns)
trigger_write to SMB outputs	33.3 + 3.3 * (#TriggerWriteGroups - 1)	3.3 * (#TriggerWriteGroups)
trigger_write to LVDS output	33.3 + 3.3 * (#TriggerWriteGroups - 1)	3.3 * (#TriggerWriteGroups)
trigger_write to PXI trigger	32 + 3.3 * (#TriggerWriteGroups - 1)	3.3 * (#TriggerWriteGroups)
trigger_write to PXI Star	32 + 3.3 * (#TriggerWriteGroups - 1)	3.3 * (#TriggerWriteGroups)

The following table provides some examples for M5302A:

Table 6 Examples of distribution of instructions into TriggerIOGroupsON and TriggerIOGroupsOFF

Example#	Triggers ON	Triggers OFF	#TriggerIOGroupsON	#TriggerIOGroupsOFF	#TriggerWriteGroups*	Execution time (ns)	Fetch time (cycles)
1	SMB1, SMB2		1	0	1	33.3	3.3
2	SMB1, SMB2		2	0	1	33.3	3.3
	LVDS0, LVDS1					33.3	
3	SMB1	SMB2, SMB7	1	1	1	33.3	3.3
4	SMB1	SMB2, SMB8	1	2	2	36.63	6.6
5	SMB1	SMB2	2	1	2	36.63	6.6
	LVDS0, LVDS1					36.63	
6	SMB1	SMB2	2	2	2	36.63	6.6
	LVDS0, LVDS1	LVDS2, LVDS3				36.63	

*Trigger Write Groups

Any number of TriggerIOs can be written at the same time, but the timing is impacted by the number of TriggerWriteGroups. The number of TriggerWriteGroups is determined by the Trigger IO groups and the combination of ON or OFF values.

#TriggerWriteGroups = ceil[(TriggerIOGroupsON + TriggerIOGroupsOFF)/2]

where,

#TriggerIOGroupsON is the number of TriggerIOGroups that contain values set to ON.

#TriggerIOGroupsOFF is the number of TriggerIOGroups that contain values set to OFF.

Understanding ‘Ceiling’ of values

If you notice, the formula above results in fractional values. The ceiling of values rounds off the fraction to the next whole number. For example,

Ceil(0) = 0

Ceil(0.5) = 1

Ceil(1) = 1

Ceil(1.5) = 2

Therefore, when calculating the Execution and Fetch times, you must consider the ceiling value.

2.5.5: HVI Action timings

The ActionExecute HVI instruction synchronously runs a list of HVI actions that you have defined. HVI actions are organized in groups that can contain up to 16 actions.

Each instrument defines its own groups of actions, and the M5302A module has '8 actions', which are included in 'one group'.

Action groups

Table 7 Elements in Action Groups for M5302A

Action Groups	Signals
Group 1	Action1, Action2, ..., Action8

Any number of HVI actions can be run synchronously, regardless of the group to which each action, which you have specified, belongs. However, the number of action groups included in the action-execute instruction (#ActionGroups) will affect both the Fetch time and the Execution time of the instruction.

The execution time for M5302A actions is calculated as:

$$[1 + \{2 + \text{Ceil}(\#\text{ActionGroups} / 2)\}] * (3.3 \text{ ns} / \text{cycle})$$

The fetch time for M5302A actions is calculated as:

$$[\text{Ceil}(\#\text{ActionGroups} / 2)] * (3.3 \text{ ns} / \text{cycle})$$

Here, Ceil(#ActionGroups / 2) is interpreted as Ceil(1 / 2) or Ceil(0.5) = 1

ActionExecute instruction timings

Table 8 Minimum delay required (in nanoseconds) for ActionExecute instructions

HVI ActionExecute Instruction Groups	Execution time (ns)	Fetch time (ns)
Group 1	10	$3.\overline{3}$

2.5.6: HVI Event timings

Event groups

M5302A has '8 events' that originate in the sandbox, organized in 'one group'.

Table 9 Elements in Event Groups for M5302A

Event Groups	Signals
Group 1	FPGA_UserEvent0, FPGA_UserEvent1, ..., FPGA_UserEvent7

The "*Instrument_Event_Latency*" is the delay from the event source until the event state is available inside the HVI Engine and is measured as 1 HVI clock cycle.

Events can be used in event conditions and HVI statements in general, like the WaitForEvent statement. Refer to *Chapter 7: HVI Time Management and Latency* in the *KS2201A PathWave Test Sync Executive User Manual* for more information on WaitForEvent and other statement timing using events. The following table list the event latency for the M5302A events.

Table 10 Elements in Event Groups for M5302A

Event Category	HVI Events	Instrument_Event_Latency (ns)
External Events (TriggerIO)	SMB Input	28.33
	LVDS Input	TBD*
	PXI_TRIG Input	9
	PXI_STAR Input	TBD*
Internal Events	FPGA_UserEvent0, FPGA_UserEvent1, ...	$3.\overline{3}$
	Other internal events	$3.\overline{3}$

* Instrument Event Latency value shall be updated in a future release.

For the M5302A modules, the TriggerIO additional condition evaluation latency is:

Table 11 TriggerIO additional condition evaluation latency in M5302A

Name	Timing value (ns)
Instrument_TrigerIO_Condition_Latency	$3.\overline{3}$

To understand how the Event latencies are measured, let us consider few definitions:

1 *Event_arrival_time* is:
   - For Internal Events
     - *Event_Arrival_Time = Internal_Event_Generation_Time - WaitForEvent_Start_Time*
   - For External Events
     - *Event_Arrival_Time = Event_At_Module_Connector_Time – WaitForEvent_Start_Time*

   The event time can be measured at the front panel or PXIe backplane connector depending on the event.

   The "Internal_Event_Generation_Time" for Internal Events and "Event_At_Module_Connector_Time" for External Events are defined based on your FPGA design.

2 *Instrument_Event_Latency* is the delay from the event source until the event state is available inside the HVI Engine. Events sources can be the Trigger IOs, or internal to the product (including User Sandbox Events). It is an instrument and event specific value.

3 *Instrument_Event_Condition_Latency* is the time needed for the condition evaluation to be executed once the event has settled inside the HVI Engine.

Table 12 Instrument Event Condition latency in M5302A

Event type	Instrument Event Condition Latency (ns)
Internal Event	3.3
Trigger IO	6.6

NOTE The Event_Arrival_Time can be a negative value if the event enters the module before the Wait-For-Event instruction Start Time.

A Local Wait-for-event statement blocks HVI execution in a Local sequence until an event occurs. The timing of this statement depends on the actual event used, as documented in Table 10.

Events sources can be any of the Trigger IOs, or internal M5302A events described above.

The following diagrams shows two scenarios where the execution time of a Wait-For-Event statement can vary:

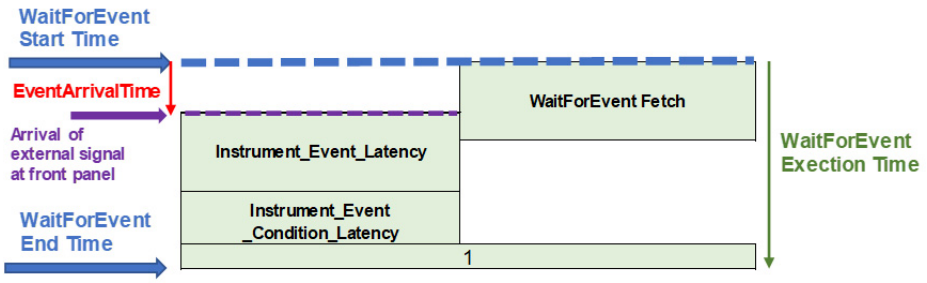


Figure 1 Case 1: Event arrival + propagation delay after Fetch-Time completion

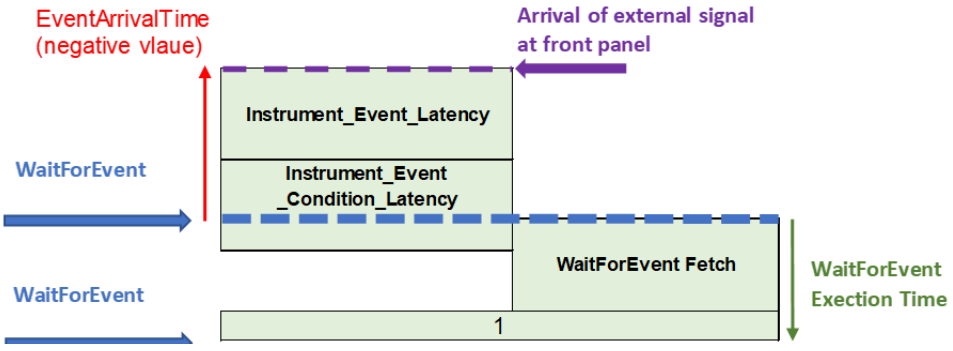


Figure 2 Case 2: Event arrival + propagation delay completes before Fetch-Time completion

Here, *EventArrivalTime* is the time from the start of the wait-for-event until the event is present at the instrument input, or for internal events when generated. If the Event happens before the wait-for-event start, it can be negative down to *Instrument_Event_Latency* or (*Instrument_Event_Latency* + *Instrument_TrigerIO_Condition_Latency*) in the case of TriggerIOs.

Table 13 Local Wait-for-event statement timing values for M5302A

Events		Execution time (ns)	Fetch time (ns)
External Events (TriggerIO)	SMB Input	MAX [(Event_Arrival_Time + 35.0), 10] + 3.$\overline{3}$	10
	LVDS Input	MAX [(Event_Arrival_Time + TBD), 10] + 3.$\overline{3}$	10
	PXI_TRIG Input	MAX [(Event_Arrival_Time + 16), 10] + 3.$\overline{3}$	10
	PXI_STAR Input	MAX [(Event_Arrival_Time + TBD), 10] + 3.$\overline{3}$	10
Internal Events	FPGA_UserEvent0, FPGA_UserEvent1, ...	MAX [(Event_Arrival_Time + 6.$\overline{6}$), 10] + 3.$\overline{3}$	10
	Other internal events	MAX [(Event_Arrival_Time + 6.$\overline{6}$), 10] + 3.$\overline{3}$	10

Table 14 Local Wait-for-event latency timing values for M5302A

Parameter	Time (ns)
Start-Latency	0
End-Latency	3.$\overline{3}$

2.5.7: HVI Local Instructions Timing for M5302A

Arithmetic Logic Unit (ALU) Instructions

Arithmetic Logic Unit (ALU) instructions are the register add, subtract or assign operations that are available in the HVI-native instruction set.

Table 15 Minimum delay required (in nanoseconds) for ALU instructions

HVI ALU instructions	Execution time (ns)	Fetch time (ns)
Add	26.66	3.$\overline{3}$
Subtract	26.66	3.$\overline{3}$
Assign	16.66	3.$\overline{3}$

3. Using the PathWave FPGA Board Support Package (BSP)

Licensing for PathWave FPGA BSP support 26
Features in PathWave FPGA 27
Working with PathWave FPGA software 28
Using BSP with PathWave FPGA software 31
Generating a k7z file using PathWave FPGA BSP 33
Loading k7z file onto modules 40

This chapter provides an introduction to the PathWave FPGA Board Support Package (BSP) and describes its implementation in the M5302x API. For detailed information about using the *PathWave FPGA 2020 Update 1.1* software along with BSP for supported modules, refer to the *PathWave FPGA Customer Documentation* and the *BSP guides* for the respective modules.

Section 3.1: Licensing for PathWave FPGA BSP support

All instruments that can be programmed with PathWave FPGA BSP will enable programming using the licensing option *-PWP*. You are required to procure an *-PWP* license option for the instrument that must be programmed. For example, if you are purchasing ten M5302A cards, each card requires an individual *-PWP* license option enabled.

These modules may also be purchased to include the following Xilinx FPGA:

1 xcku15p-ffva1156-1-e

The following table outlines the Sandbox FPGA resources, which are available for custom logic for each module.

Table 16 Available FPGA Resources (Sandbox) per module

Resource Type	Resource Information
CLB LUTs	251160
CLB Registers	502320
DSP Blocks	960
Block RAM Tiles	480
Ultra RAM	80

Section 3.2: Features in PathWave FPGA

The primary features available in the KF9000A PathWave FPGA environment are:

- Support for larger number of Keysight developed boards
- Uses Xilinx Vivado Design Suite locally on Host Machine to generate the output bitstream
- Generates “k7z” file, which is loaded on the FPGA of the new modules
- IPs can be defined using IP-XACT file. This gives you information about the IP name, version, interfaces, category, and so on.
- A repository of IPs that are using IP-XACT can be loaded concurrently.
- Block designs are now called sub-modules.
- Multiple boards are supported using the BSP technology.

Section 3.3: Working with PathWave FPGA software

The KF9000A PathWave FPGA Programming Environment (commonly known as PathWave FPGA), powered by *Xilinx Vivado Design Suite*, is used for FPGA designing on supported Keysight PXIe modules.

Some applications require the use of custom on-board real-time processing, which might not be covered by the comprehensive off-the-shelf functionalities of standard hardware products. For these applications, Keysight supplies Option *-PWP* (Enabled FPGA Programming) that provide the capability to program the on-board FPGA. With PathWave FPGA, development time is dramatically reduced and you can focus exclusively on expanding the functionality of the standard instrument, instead of developing a complete new one.

PathWave FPGA is a graphical environment that provides a complete FPGA design flow for rapid FPGA development from design creation to simulation to Gateware deployment to Hardware/Gateware verification on Keysight hardware with Open FPGA. This environment provides a design flow from schematic to bitstream file generation with the click of a button.

NOTE **The PathWave FPGA 2020 is a licensed software. Contact Keysight Support for more information on procuring the respective licenses.**

Once you have installed the PathWave FPGA 2020 software, you can launch its user interface from the **Start** menu. Alternatively, you may use the corresponding command line or scripts to launch the application.

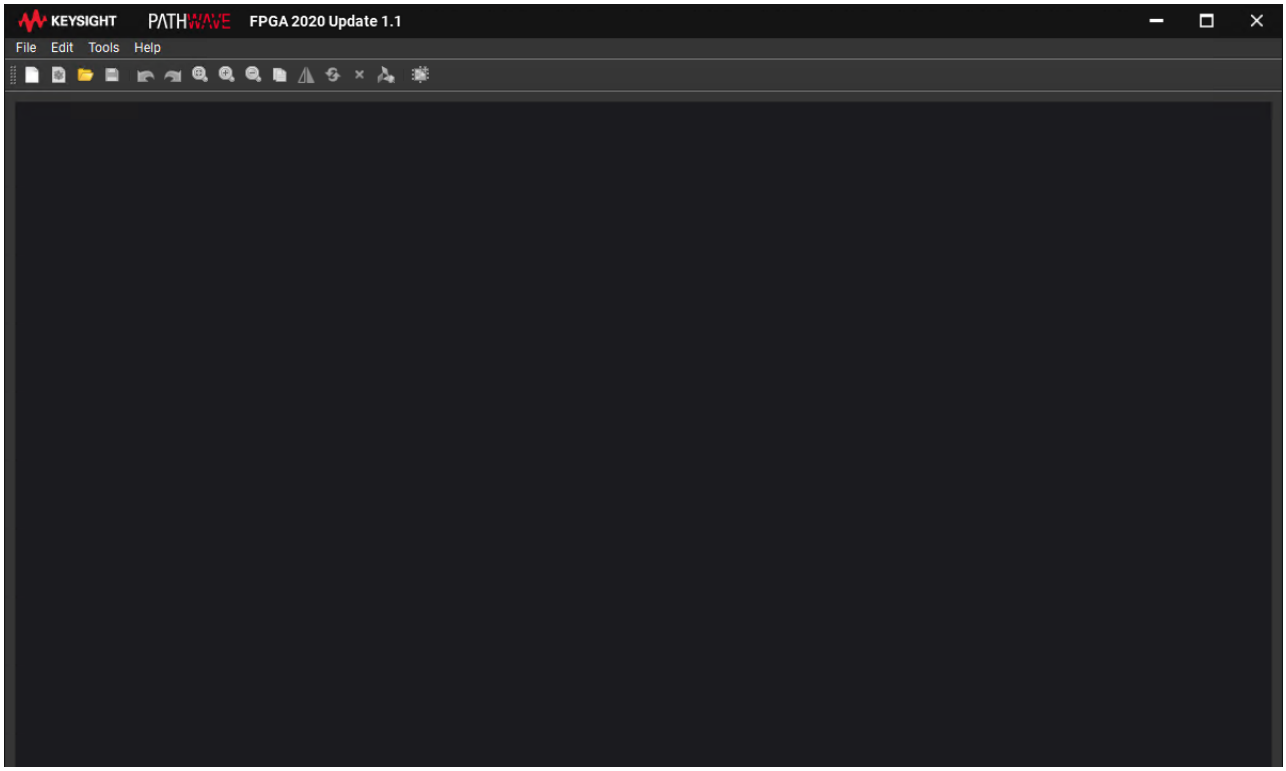


Figure 3 Default window for PathWave FPGA 2020 Update 1.1 software

In Keysight PathWave, FPGA code is represented as boxes (called blocks) with IO ports. An empty project contains the “Default Product Blocks”, and the “Design IO Blocks” that provide the outer interface of the design. You can add/remove blocks from the Keysight Block Library, External Blocks, or Xilinx IP cores.

To know about the required prerequisite software, system requirements and licensing information for the PathWave FPGA 2020 Update 1.1 software, refer to the “*Getting Started*” section of the *PathWave FPGA Customer Documentation*. To view the installation procedure, refer to the “*Installing PathWave FPGA 2020 Update 1.1 software*” section and to view how to launch the software, refer to the “*Launching the PathWave FPGA BSP*” section in the *M5302x Software Startup Guide*.

3.3.1: Understanding Partial Configuration (PR)

The FPGA configurable region utilizes Xilinx FPGA partial reconfiguration technology to allow you to design and configure a defined section of the supported M5302A FPGA without the need to power down or reboot the module or host computer, respectively.

PathWave FPGA supports the design flow in the Partial Reconfiguration (PR) technology. In a PR flow, a full FPGA reconfiguration is only necessary once for a given static region version. The sandboxes can be reconfigured anytime, without a full reconfiguration, and without stopping the current operation of the FPGA.

Partial Reconfiguration enables performing dynamic change of modules within an active design. By implementing multiple configurations in this flow, you can achieve full bitstream for each configuration and partial bitstream for each reconfigurable module.

While FPGA technology facilitates direct programming and re-programming of modules without going through re-fabrication with a modified design; the partial reconfiguration technique allows the modification of an operating FPGA design by loading a partial configuration file. The logic in the FPGA design is divided into two different types, reconfigurable logic and static logic. The static logic remains functional and is unaffected by the loading of a partial bitstream file, whereas the reconfigurable logic is replaced by the contents of that bitstream file.

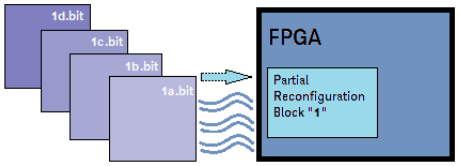


Figure 4 Block diagram depicting partial reconfiguration using bitstream files

Partial Reconfiguration (PR) is modifying a subset of logic in an operating FPGA design by downloading a partial bitstream. In some cases, a complete FPGA reconfiguration might be preferable (better routing, timing closure, and so on). This is also supported by PathWave FPGA, as long as a reboot is not required after the reconfiguration process.

The PathWave FPGA software does not, by itself, provide access to the controls for the supported PXIe modules. You must install the module-specific Board Support Package (BSP) to leverage the features within the PathWave FPGA software for FPGA designing and for loading your customized code onto the Keysight instrument.

The design part in Keysight FPGA consists of two regions: the static region and the sandbox region. The static region for each supported module is defined within BSP and cannot be modified. This region defines the implementation of the FPGA interfaces to external resources, and defines the interfaces to the sandbox. A static region implementation can define one or more sandbox regions in an FPGA design. The sandbox region contains the user specific FPGA design. The interface of the sandbox depends on the static region implementation.

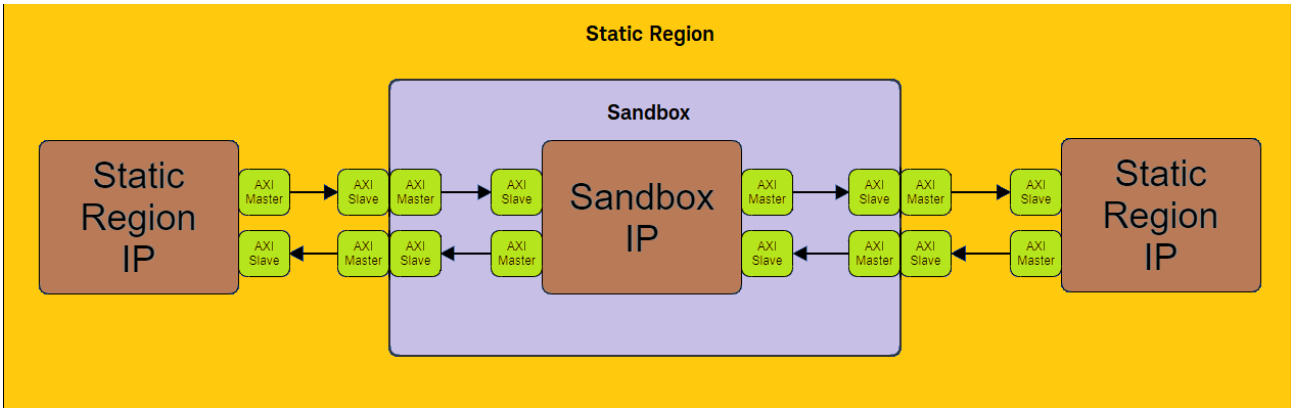


Figure 5 Block diagram representing layout of Static and Sandbox regions

The block diagram shown in Figure 6 depicts a simple view of M5302A IOs, sandbox ports, HVI connections, and data paths to and from the PCIe interface and DDR4 memory.

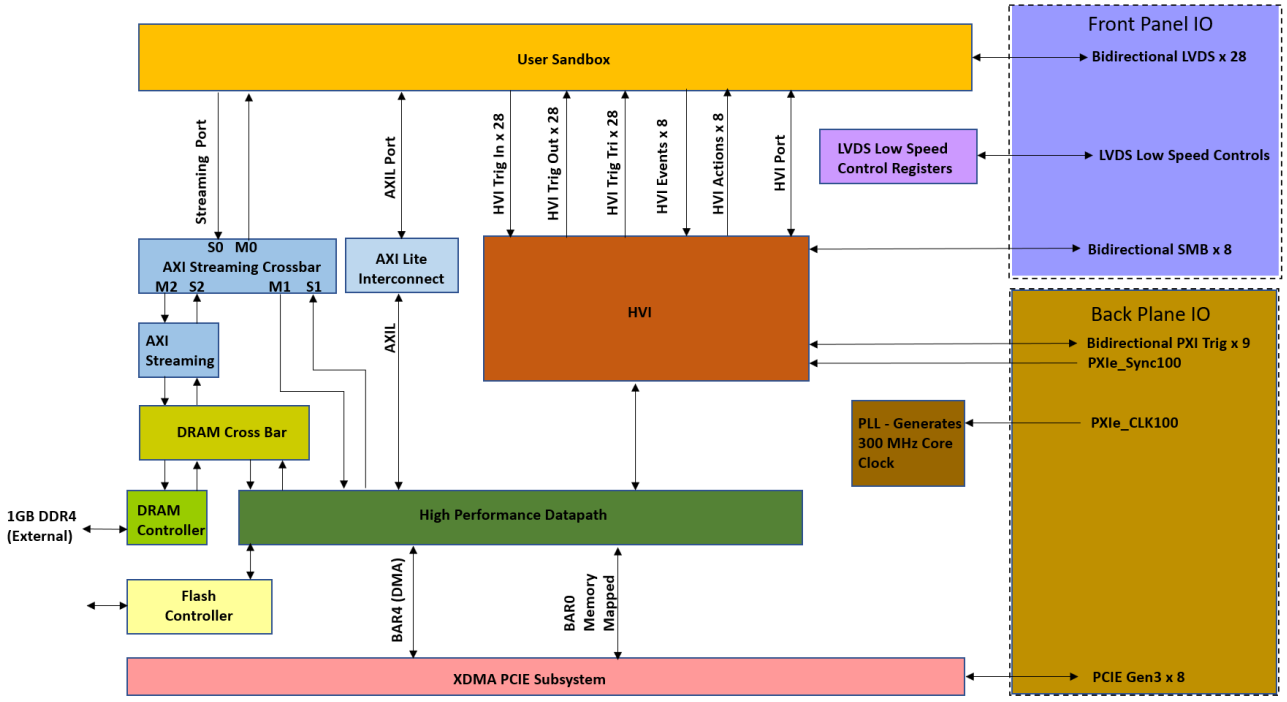


Figure 6 Block diagram depicting implementation of various ports in M5302A

Section 3.4: Using BSP with PathWave FPGA software

3.4.1: Understanding BSP composition

In embedded systems, the board support package (BSP) is the layer of software containing hardware-specific drivers and other routines that allow a particular operating system (traditionally a real-time operating system, or RTOS) to function in a particular hardware environment (a computer or CPU card), integrated with the RTOS itself. Third-party hardware developers, who wish to support a particular RTOS must create a BSP that allows that RTOS to run on their platform. In most cases the RTOS image and license, the BSP containing it, and the hardware are bundled together by the hardware vendor. BSPs are typically customizable, allowing you to specify the drivers and routines, which should be included in the software build based on their selection of hardware and software options. (*source: Wikipedia*)

The Board Support Package (BSP), installed separately from PathWave FPGA, comprises of two parts—an FPGA Support Package (FSP) and a Runtime Support Package (RSP). A BSP configuration file contains all the necessary information to identify the configuration, which includes static region implementation, an RSP implementation, build scripts, examples, project templates, and documentation.

- The FSP is that portion of the BSP that allows you to build a bitstream file for the target FPGA. It is consumed by PathWave FPGA to support design creation and sandbox compilation; everything that is performed without the physical hardware. The FPGA Support Package allows you, as a PathWave FPGA user, to create a design targeting your instrument. It includes descriptions of the sandbox interfaces, template PathWave FPGA projects, and files required for compiling a sandbox image from HDL sources. An FSP fulfills the 'design-time requirements' of PathWave FPGA, which covers everything prior to loading a bit image onto the instrument.
- The RSP is that portion of the BSP that allows you to control your target FPGA. It provides a 'C' API that you can use to load design images onto hardware, verify your FPGA bitstream image and perform simple register and streaming accesses to the sandbox. An RSP requires a hardware driver with the following capabilities:
  - Hardware discovery/enumeration
  - Program the FPGA sandbox regions
  - Read and write registers inside the sandbox
  - Read and write to streaming interfaces inside the sandbox (if the sandbox has streaming interfaces)

Both PathWave FPGA software and the BSP work together and cannot be used individually.

The block diagram shown in Figure 7 indicates how the bitstream file is generated using both the PathWave FPGA software and BSP together for any specific supported PXIe module.

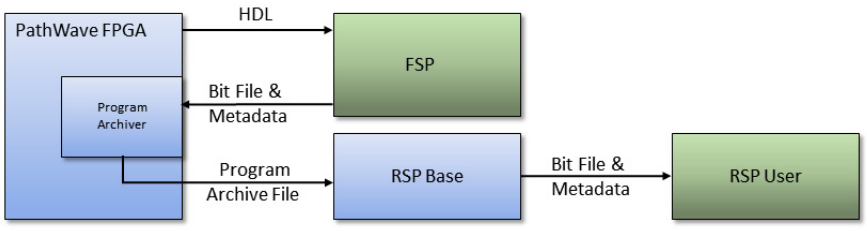


Figure 7 Bitstream file compilation flow using PathWave FPGA and BSP

PathWave FPGA manages bitstream file generation using a scripted flow of the FPGA tools. The scripts for building a specific design are provided by the FSP. The FSP build script can also add metadata, which are delivered to the RSP. The bitstream file and metadata are packaged into a Keysight PathWave FPGA program archive file with the filename extension *.k7z*.

To load a bitstream image on the FPGA, PathWave FPGA supplies the program archive file to the RSP. The RSP unpacks the contents of the program archive file and checks that the image is compatible. The original bitstream file and the metadata are then passed to the instrument-specific portion of the RSP, which eventually configures the FPGA with the bitstream file.

The output from a PathWave FPGA design compilation is saved in a Keysight PathWave FPGA program archive file, with a *k7z* extension. To understand how to generate the k7z file onto a module, see "Generating a k7z file using PathWave FPGA BSP" on page 33.

Hereafter, you may load the k7z file onto the required hardware using the corresponding M5302x API functions. To understand how to load the k7z file onto a module, see "Loading k7z file onto modules" on page 40.

Section 3.5: Generating a k7z file using PathWave FPGA BSP

After you install BSP for one or more modules, you can proceed with creating a new project to design the sandbox region for the corresponding modules.

Following steps give you a quick glance into creating a new sandbox project using PathWave FPGA 2020 software for FPGA designing followed by generating a bitstream file for one or more supported PXIe modules.

You can also perform most of the steps shown below using command line arguments. Refer to the "*Advanced Features*" section of the *PathWave FPGA Customer Documentation*.

1 On the main window of the PathWave FPGA 2020 software, click the icon for new project.

A new sandbox project dialog appears.

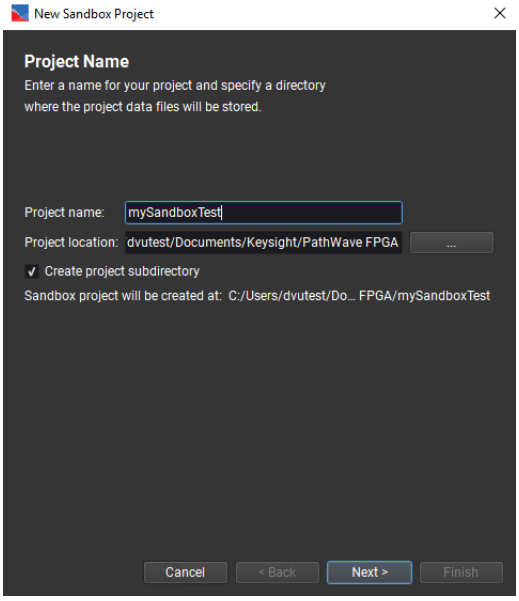


Figure 8 Creating a new Sandbox project

2 Click **Next >**.

3 For the Project Type, choose the BSP corresponding to one of the modules whose FPGA you wish to update.

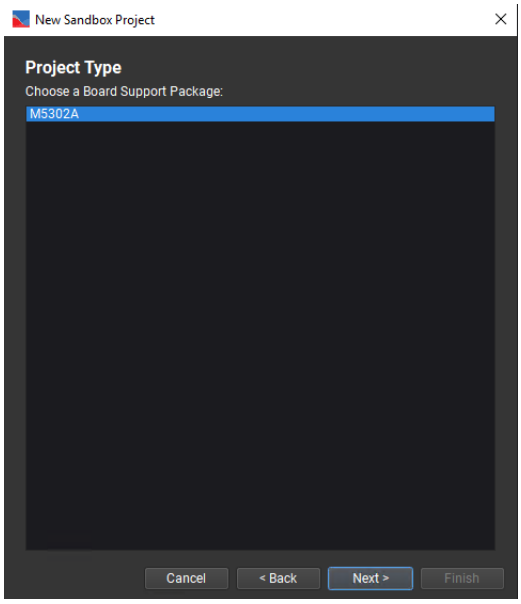


Figure 9 Selecting the module-specific BSP

4 Click **Next >**.

5 For the Project Template, either choose the default/sample template offered within the PathWave FPGA design environment or choose blank to start a new custom design logic.

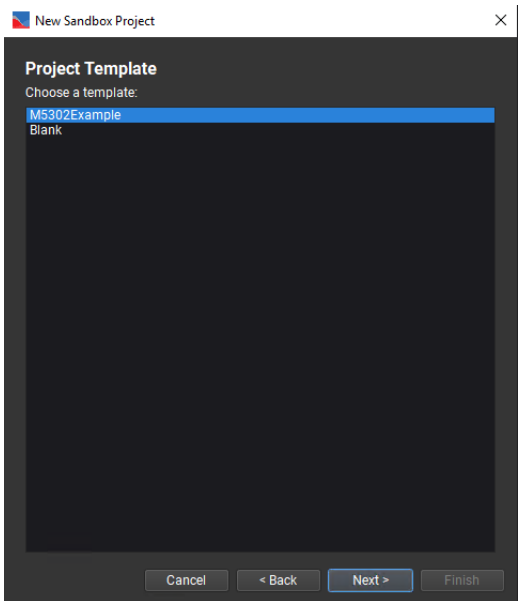


Figure 10 Selecting a project template for the selected BSP

6 Click **Next >**.

7 Verify all information displayed in the Project Summary. To make any amendments, click **Back**. If the selected options for the corresponding BSP are satisfactory, click **Finish**.

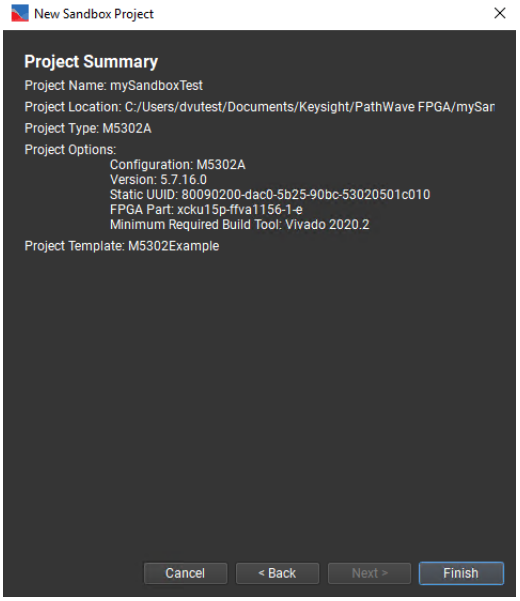


Figure 11 Viewing the project summary information

Various blocks are displayed, which are part of the default template in the PathWave FPGA 2020 software.

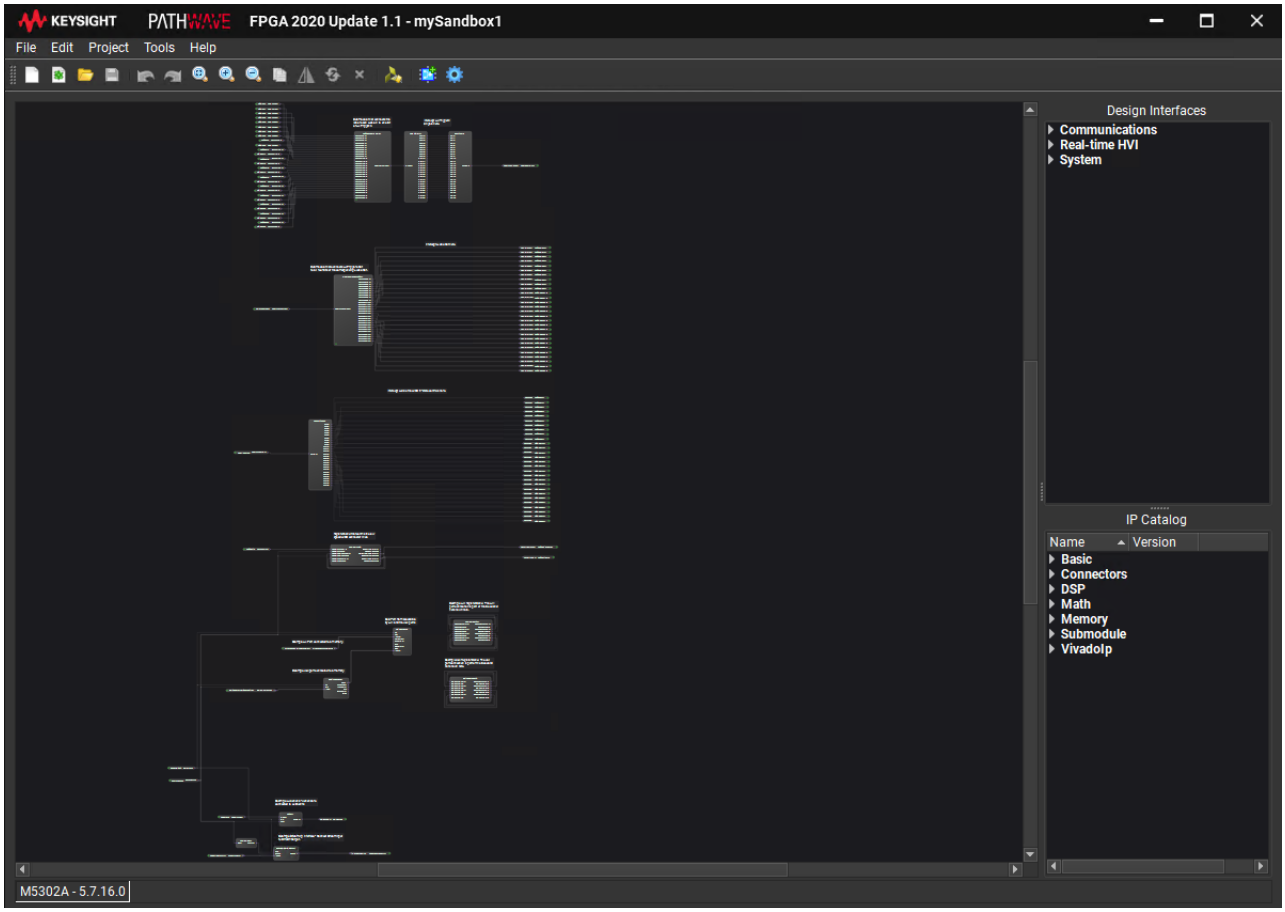


Figure 12 Viewing FPGA design blocks in the sample template

8 Customize the configuration as per your requirements using one or more elements from the Design Interfaces and IP Catalog panels.

For more information regarding the configurable region of the M5302A FPGA interfaces and BSP IP Repository, refer to the BSP User Guide for the corresponding modules, which can be accessed via **Help** > **BSPs Help** menu options in the PathWave FPGA 2020 software.

9 After you have finished designing your logic, you can proceed with the k7z file generation. Click the **Generate Bit File...** icon as shown in Figure 13.

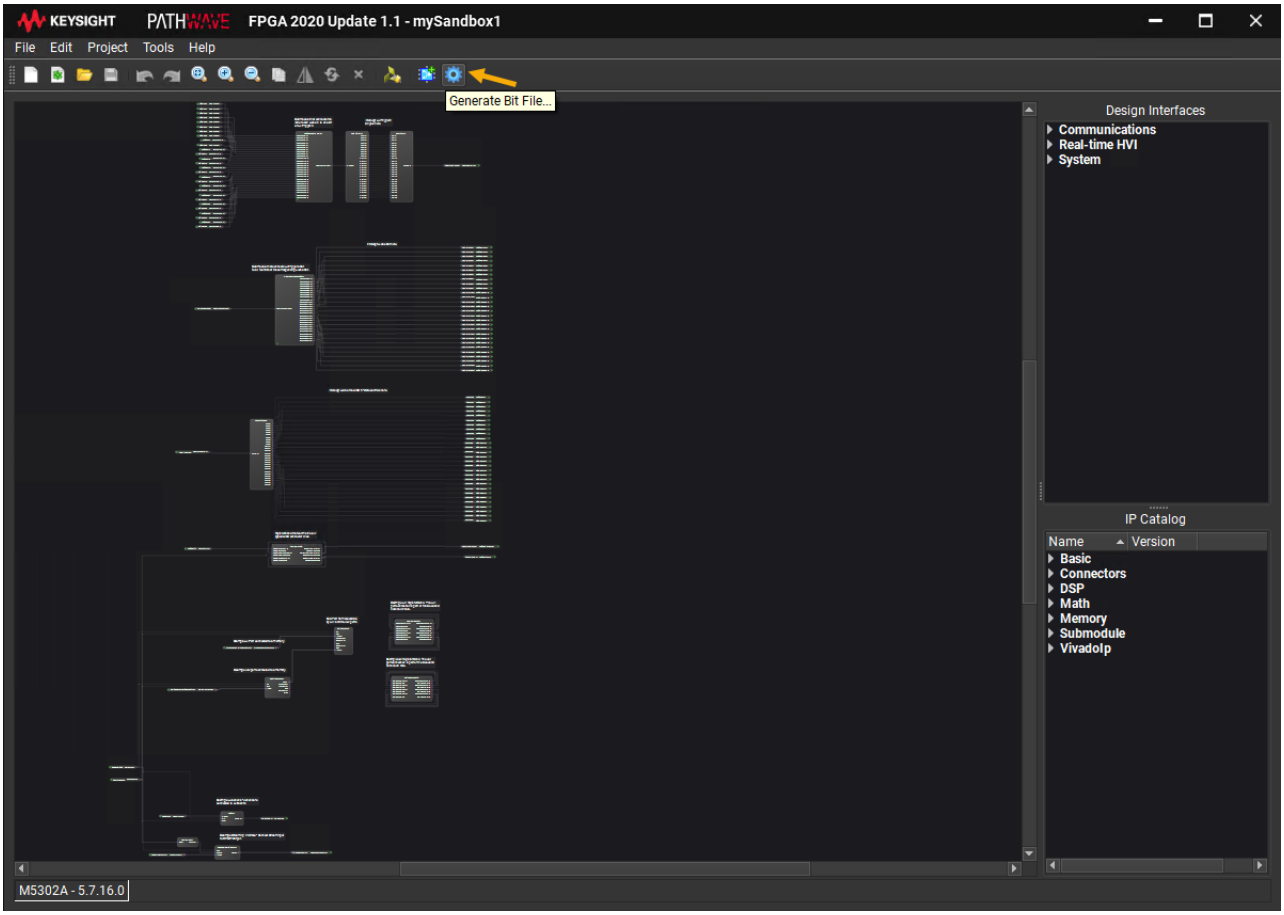
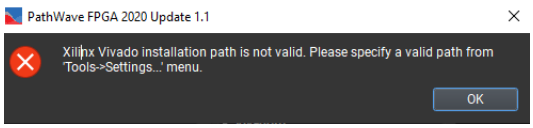


Figure 13 Initiating Bit File generation for the selected module

If you do not have the *Xilinx Vivado Design Suite* on the same machine where PathWave FPGA 2020 software and BSP are installed, the following error is prompted when you click the **Generate Bit File...** icon.



Refer to *PathWave FPGA Customer Documentation* for more information on installing the software and the licenses for the *Xilinx Vivado Design Suite*.

10 On the FPGA Hardware Build window that appears, click **Run**.

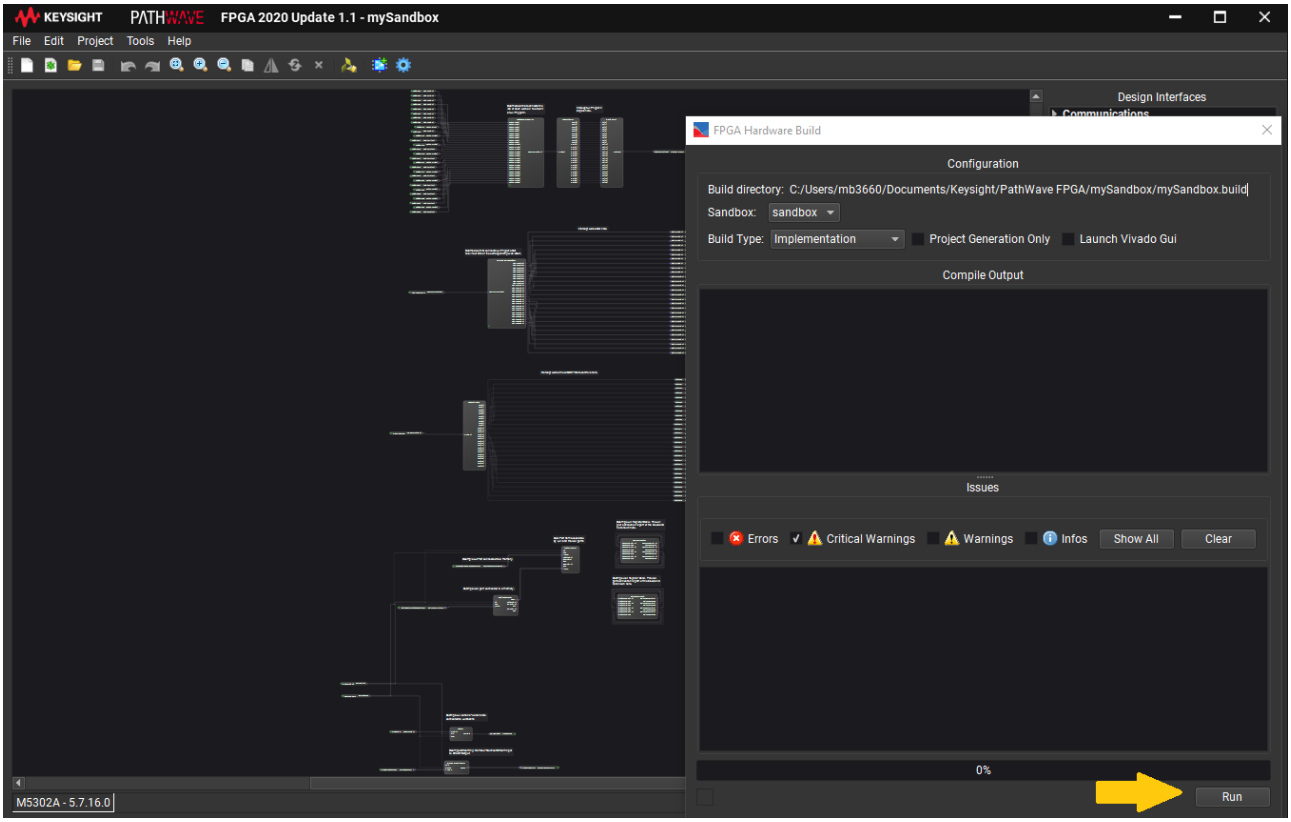


Figure 14 Generating FPGA Hardware Build

Depending on the configuration, the software takes some time before finishing the process of *k7z* file generation.

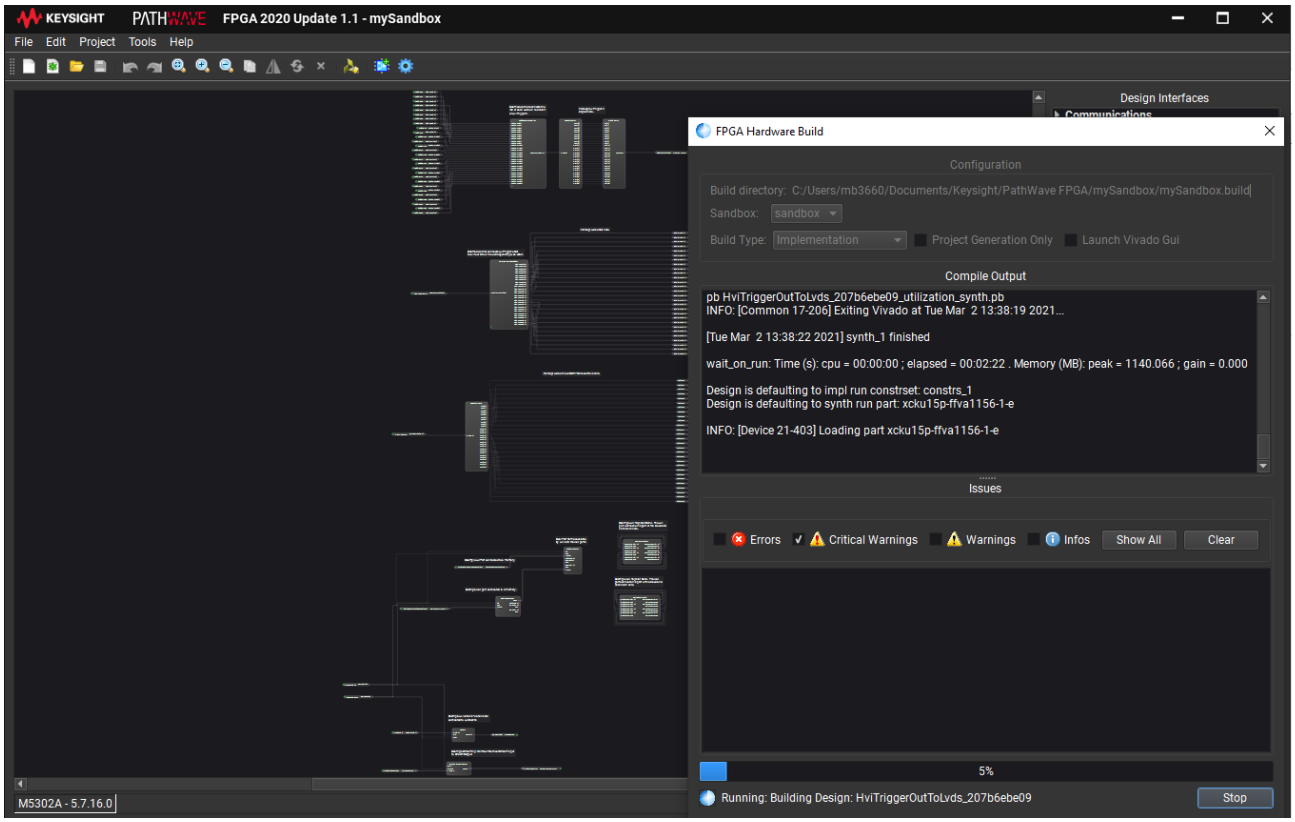


Figure 15 Progress status for the k7z file generation

Once the Bitstream file (which is the *k7z* file) is generated, your custom FPGA logic is ready to be loaded on the selected module using the M5302x API.

Other than configuring and designing your FPGA Logic using the *PathWave FPGA 2020 Update 1.1* software, you can perform one or more of the following operations:

- Build your FPGA Logic
  - Generate the Bit File (covered briefly in this section)
  - Verify the Bit File
- Simulate your FPGA Logic
  - Simulation Testbench Designing
  - Test Bench Address Mapping
- Work with Advanced features
  - Using Command Line Arguments
  - Migrating a design to a new BSP
  - Changing a Submodule Project Target Hardware
  - Debugging in Hardware

For detailed instructions on performing these steps and to understand the features of the PathWave FPGA 2020 software, refer to the PathWave FPGA Help file accessible via the Help menu of the software. For information about the features and various elements along with understanding the workflow associated with the PathWave FPGA 2020 Update 1.1 software, refer to the "*User Guide*" section of the *PathWave FPGA Customer Documentation*.

3.5.1: Loading k7z file onto modules

After a bitstream file (k7z) is generated using the PathWave FPGA software, use the M5302x API functions to load the FPGA on the corresponding module and also, verify design compatibility on that module.

To access the FPGA user design in the sandbox using M5302x API,

1 The generated *k7z* file may be loaded with the "configure_from_k7z" function

2 Retrieve the registers added to Pathwave FPGA design using "get_register_entry" function.

This API returns a tuple containing the address, length and access type.

3 (Optional) Write / Read to a 32-bit register using the Peek / Poke and writeRegisterBuffer / readRegisterBuffer functions, respectively.

See "Sample program to load a Sandbox design onto an M5302A module" on page 70 to understand the workflow of loading a sandbox design onto an M5302A module.

4. Using Keysight M5302A PXIe Module SFP Software

Installing the M5302A PXIe Module SFP Software 42
Launching the M5302A PXIe Module SFP software 43
Familiarizing with the SFP software 45
Understanding Menu options 47
Using the I/O Banks 52
Performing specific tasks in the M5302x SFP Software 55

This chapter describes how to use the Keysight M5302A PXIe Module Soft Front Panel (SFP) software.

Section 4.1: Installing the M5302A PXIe Module SFP Software

The Keysight M5302A PXIe Module SFP Software Package includes:

- Keysight M5302A PXIe Module SFP Software
- KF9000A PathWave FPGA Programming Environment (commonly known as PathWave FPGA software)—required for FPGA logic designing
- PathWave FPGA Board Support Package (BSP)—required for FPGA logic designing
- KS2201A PathWave Test Sync Executive Software—required for integration with HVI technology

Prior to installing the software package components listed above, the following software must be installed on your machine that runs a Windows 10 (64-bit) Operating System.

Prerequisites

- Keysight IO Libraries Suite 2018 (version 18.0 or later)
- Microsoft .NET 3.5 or later
- Any API interface
  - Python 64-bit version 3.7
  - Any C# /.NET Compiler
  - Any C/C++ Compiler
- *Xilinx Vivado Design Suite* (required with PathWave FPGA software)
- License options
  - Option *-PWP* (to enable FPGA programming through PathWave FPGA BSP)
  - Option *-HV2* (to implement HVI technology using the PathWave Test Sync Executive software)

Refer to the *M5302A PXIe Module SFP Software Startup Guide* for download links and installation instructions.

Section 4.2: Launching the M5302A PXIe Module SFP software

After the driver is installed, launch the SFP from the **Start** menu. Click **Start** > **Keysight M5302x PCIe Module** > **M5302x SFP (x64)**.

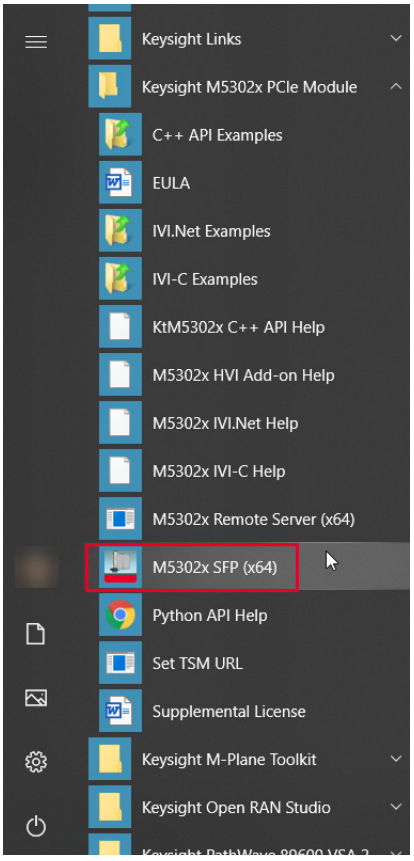


Figure 16 Launching M5302x SFP software from the Start menu

The Connect to Instrument window in the Keysight M5302x SFP software is displayed, as shown in Figure 17 and Figure 18. Refer to “Connecting to Modules” on page 55 for more information about establishing connection in the ‘online mode’ and ‘simulation mode’.

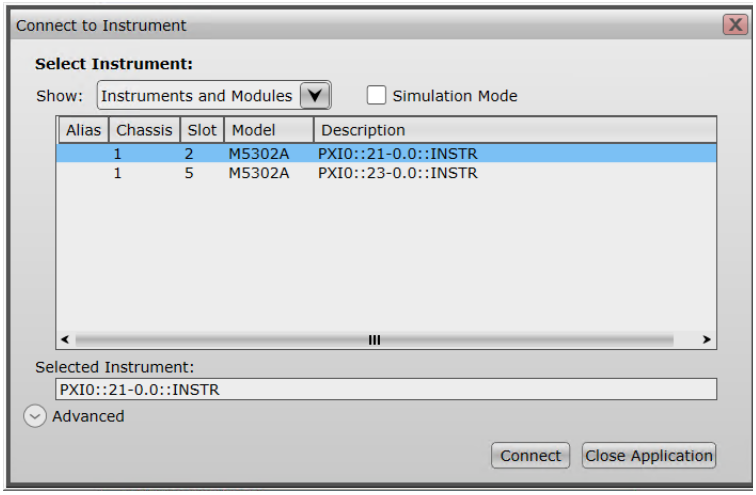


Figure 17 M5302x SFP software Connect to Instrument window (with one or more modules connected)

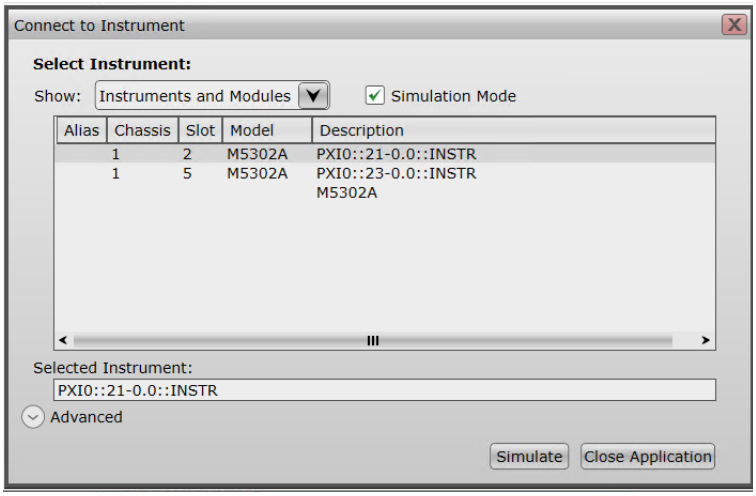


Figure 18 M5302x SFP software Connect to Instrument window (Simulation Mode)

Section 4.3: Familiarizing with the SFP software

The Soft Front Panel (SFP) interface for the M5302A Digital I/O module provides a simple tool to help you accomplish one or more of the following tasks:

- Check the module's connection status
- Save and, if required, delete (saved) module connections
- Control single-ended and differential IO signals
- Perform firmware updates and hardware upgrades
- Monitor Driver calls

The SFP has the following elements in its user interface:

- Menus
- I/O Banks
- Status Indicators

4.3.1: Menus



Figure 19 Menu options in the SFP

To understand the features that appear in the Menu options shown in Figure 19, see:

- File menu options
- View menu options
- Utilities menu options
- Tools menu options
- Help menu options

4.3.2: I/O Banks



Figure 20 I/O Banks in the SFP

To understand the features that appear in the I/O Banks shown in Figure 20, see:

- LVDS I/O
- SMB IO
- Low Speed General Purpose IO
- PXI IO

4.3.3: Status Indicators



Figure 21 Status indicator tab in the SFP

To understand the status information that appear on this tab, as shown in Figure 21, see:

- Checking Instrument Status

Section 4.4: Understanding Menu options

4.4.1: File menu options

The **File** menu provides the following options.

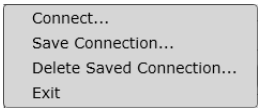


Figure 22 Options in the File menu

- **Connect...**: This option is used to connect the soft front panel (SFP) to a module in the chassis or to run in simulation mode. See Connecting to Modules.
- **Save Connection...**: This option is used to save an instrument configuration in the IVI Configuration Store. This process may also be referred to as Creating a Connection Alias.
- **Delete Saved Connection...**: This option is used to delete an instrument configuration from the IVI Configuration Store.
- **Exit**: This option is used to close the SFP session.

To save a connection:

1 Select **File** > **Save Connection...**.

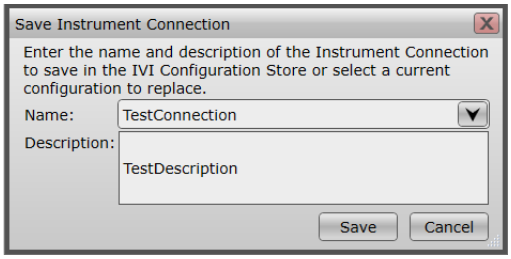


Figure 23 Saving instrument connection

2 In the **Name** field, specify a name for the connection to be saved. Optionally, you may add an appropriate description for the connection in the **Description** field.

3 Click **Save**. If the name specified for the connection (in the **Save Instrument Connection** dialog box) exists for another pre-saved connection, a prompt is displayed before the connection is replaced (overwritten) with the new connection.

To delete a connection:

1 Select **File** > **Delete Saved Connection...**.

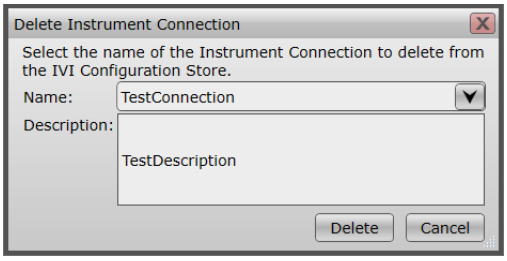


Figure 24 Deleting instrument connection

2 From the **Name** drop-down list on the **Delete Instrument Connection** dialog box, select the name of the connection, which is to be deleted.

3 Click **Delete**.

4 On the **Confirm Delete** prompt, click “Yes” to proceed with deleting the connection.

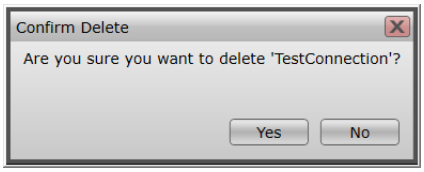


Figure 25 Prompt to confirm deletion

4.4.2: View menu options

The **View** menu provides the following options.

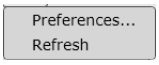


Figure 26 Options in the View menu

- **Preferences...**: This option opens the **Preferences** window, which is used to adjust the color scheme and the current font size.

  These preference values can be changed anytime between SFP sessions, so you do not have to set them each time the SFP is opened.

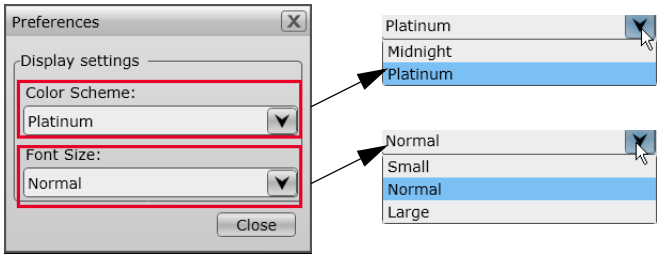


Figure 27 Options in the Preferences sub-menu

- **Refresh**: Use this option to manually refresh the module state.

  In addition to overall status, the **Refresh** control synchronizes the SFP with the current hardware / driver settings.

4.4.3: Utilities menu options

The **Utilities** menu provides the following options.

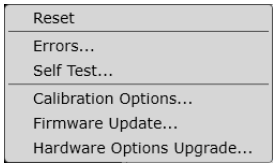


Figure 28 Options in the Utilities menu

- **Reset**: Resets the module to its default settings.

  For more information, refer to Resetting to Default Factory Settings.
- **Errors...**: Resets the module to its default settings.

  For more information, refer to Viewing Errors log.
- **Self Test...**: Displays the **Self Test** window, where you can run a Self Test on the module.

  For more information, refer to Conducting a Self Test.
- **Calibration Options...**: Displays the **Calibration Information and Preferences** window, where you can modify the calibration schedule, which allows you to manage and set the calibration intervals and warnings.

  For more information, refer to Setting Calibration Options.
- **Firmware Update...**: Displays the **Firmware Update** window, where you can perform firmware update on the M5302A module.

  For more information, refer to Updating Firmware.
- **Hardware Options Upgrade...**: Displays the **Hardware Options Upgrade** window, where you can upgrade the options on the hardware being used in the M5302A module configuration.

  For more information, refer to Upgrading Hardware Options.

4.4.4: Tools menu options

The **Tools** menu provides the following options.



Figure 29 Options in the Tools menu

- **Monitor Driver Calls**: This option opens the **Driver Call Log** window, where a list of the driver calls are displayed in the **Call History** area. These driver calls correspond to each action you perform on the SFP.

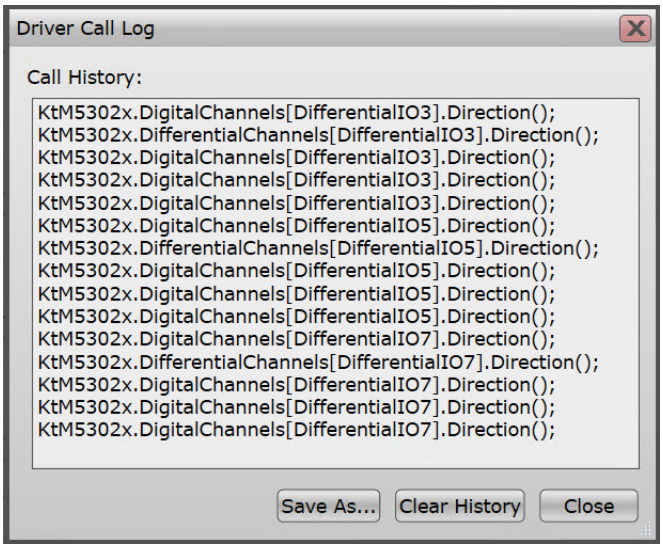


Figure 30 Messages as they appear on the Driver Call Log window

To learn more about the IVI commands associated with the driver calls listed in the driver call log,

- For IVI-C Driver Help, go to the folder *C:\Program Files\IVI Foundation\IVI\Drivers\KtM5302x*.
- For IVI-.NET Driver Help, go to the folder *C:\Program Files\IVI Foundation\IVI\Microsoft.NET\Framework64\v4.5.50709\Keysight.KtM5302x <build-version>\Help*.

To clear the driver call logs:

1 On the **Driver Call Log** window, click **Clear History**.

This clears the displayed driver call logs and refreshes the **Driver Call Log** window.

To copy and paste call history from the driver call log:

1 In the **Driver Call Log** window, highlight the call history to be copied. You may also use the right-click menu option **Select All Ctrl+A** to select the entire log displayed in the active instance.

2 Right-click in the text area and click **Copy Ctrl+C**.

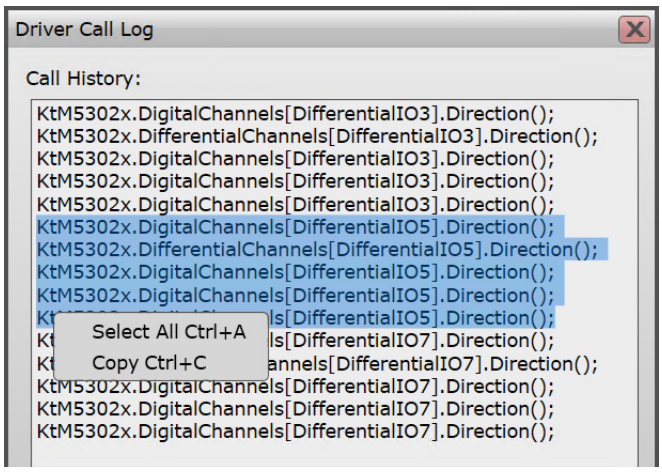


Figure 31 Right-click menu options in the Driver Call Log window

3 Paste the call log entries from the system's memory on to any editor, such as Notepad, and so on.

To save a copy of the Driver Call Log to a file:

1 Click **Save As...**.

A file system browser opens, which enables you to navigate any folder and save the Driver Call Log.

2 Navigate to a folder, specify a file name, and click **Save**.

4.4.5: Help menu options

The **Help** menu provides the following options.

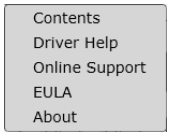


Figure 32 Options in the Help menu

- **Contents**: Opens the Online Help system related to SFP.
- **Driver Help**: Opens the Online Help system related to IVI.NET driver.
- **Online Support**: Opens a browser web page to Keysight Technologies support.
- **EULA**: Opens the Keysight software's end user license agreement, associated with the SFP software.
- **About**: Opens a window that displays the driver version information and module information, such as the serial number, hardware revision, and license options.

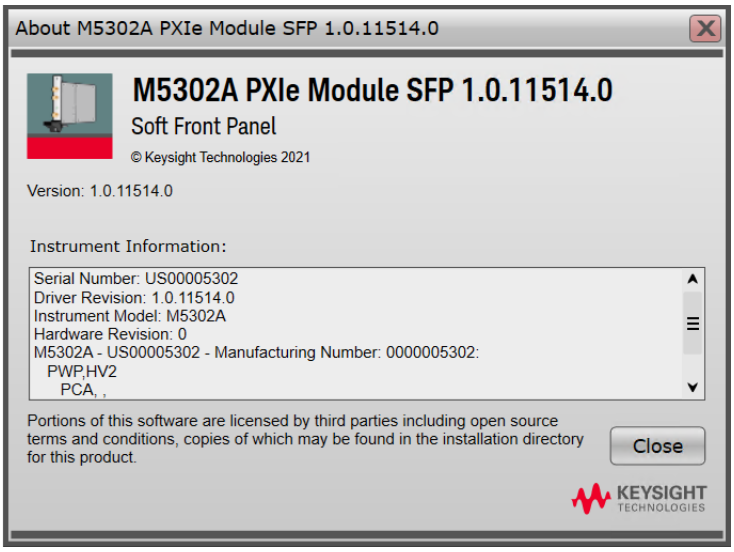


Figure 33 About window in SFP

Section 4.5: Using the I/O Banks

The M5302A is a single-slot PXIe module with single-ended and differential digital IO connectors.

The SFP allows you to control the digital (input-output) signals on these connectors using the IO Banks.

NOTE The SFP provides limited control of the M5302A module. For enhanced control and for additional configuration of the M5302A module, Keysight recommends using the M5302A module's API. For more information on the C, C++, .NET and Python APIs, refer to the Help available in the *Keysight M5302x PCIe Module* folder in the *Start* menu of your machine.



Figure 34 I/O Banks in the M5302x SFP

The description of the various I/O banks is covered in:

- LVDS I/O
- SMB IO
- Low Speed General Purpose IO
- PXI IO

4.5.1: LVDS I/O

The M5302A module has 28 bi-directional differential IO on the 100-pin connector. The differential signals are LVDS (Low Voltage Differential Signals), which can drive 100-ohm differential load and can toggle up to 1.2 Gbps (600MHz clock speed). The connector signals are called *USER_LVDSn_P/N*, where 'n' denotes 0 to 27. For example, the first differential pair is *USERIO_LVDS0_P* and *USERIO_LVDS0_N*.

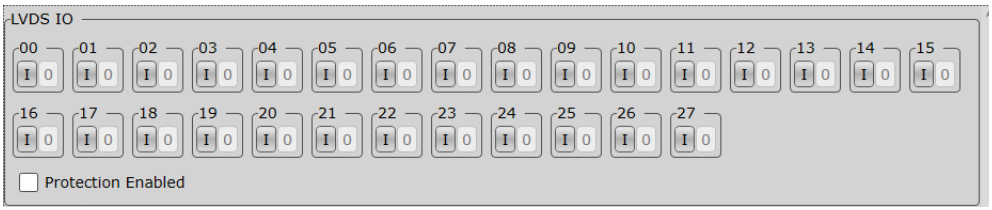


Figure 35 LVDS I/O Banks in the M5302x SFP

The LVDS I/O bank on the SFP lets you control the input-output differential signal state on one or more pins. Each pin is denoted by a double-digit number (00 to 27). By default, the pins are set to drive an input (I) signal in a low (0) state. You can manually switch to output (O) and toggle the signal state (low-to-high or vice-versa). To view instructions regarding toggling states, see Toggling Signal state.

Currently, the LVDS port on the M5302A module uses an Output Safety feature, where the output will not be driven on this port, unless one of the following conditions is met.

1 The output safety input pin is driven high by some other device. This scenario occurs when you use two M5302A modules along with cables recommended by Keysight only.

2 The output safety circuit is either disabled programmatically using the related API or by clearing the **Protection Enabled** check box in the SFP. This scenario is useful, if you wish to use the LVDS connectors to connect your M5302A modules to external devices / circuitry and in case of a power loss, you do not need to protect that circuitry, whereas the M5302A modules do not.

4.5.2: SMB IO

The M5302A module comprises of eight connectors for single-ended IO signals, which are denoted as *TRIG1* through *TRIG8*.

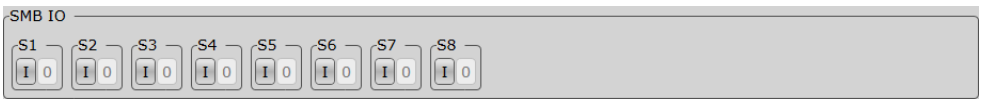


Figure 36 SMB IO Banks in the M5302x SFP

The SMB IO bank on the SFP lets you control the input-output single-ended signal state on one or more connectors. Each connector *TRIGn* is represented by *Sn* on the SFP, where n = 1 to 8. By default, the connectors are set to drive an input (I) signal in a low (0) state. You can manually switch to output (O) and toggle the signal state (low-to-high or vice-versa). To view instructions regarding toggling states, see Toggling Signal state.

4.5.3: Low Speed General Purpose IO

The M5302A module comprises of four connectors for Low Speed IO signals, which are denoted as *L0* through *L3*.



Figure 37 Low Speed General Purpose IO Banks in the M5302x SFP

The Low Speed General Purpose IO bank on the SFP lets you control the input-output differential state on one or more connectors for all such signals that operate at a much lower speed than the LVDS or SMB signals. Note that while the other signals can be driven using the HVI registers, the Low Speed General Purpose IO signals are programmed using the registers specific to the M5302x API only. Each connector is represented by *Ln* on the SFP, where n = 0 to 3. By default, the connectors are set to drive an input (I) signal in a low (0) state. You can manually switch to output (O) and toggle the signal state (low-to-high or vice-versa). To view instructions regarding toggling states, see Toggling Signal state.

4.5.4: PXI IO

The PXI backplane of the M5302A module provides eight bus-trigger lines (*PXI_TRIG[7:0]*) and one point-point trigger line (*PXI_STAR*). These lines allow communication between cards (for example, one card communicates to another card to perform an action, when there is a rising edge on a trigger line).

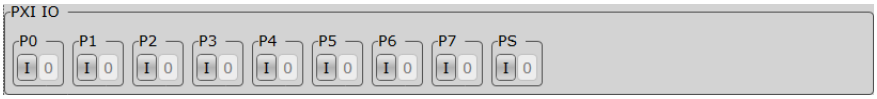


Figure 38 PXI IO Banks in the M5302x SFP

The PXI IO bank on the SFP lets you control the states on these PXI trigger lines. Each line (*PXI_TRIG[7:0]*) is represented by *Pn* on the SFP, where n = 0 to 7. By default, the lines are set to drive an input (I) signal in a low (0) state. You can manually switch to output (O) and toggle the signal state (low-to-high or vice-versa). To view instructions regarding toggling states, see Toggling Signal state.

Section 4.6: Performing specific tasks in the M5302x SFP Software

This section provides how to information of various features in M5302A SFP. These features are explained in details in the following topics:

- Connecting to Modules
- Checking Instrument Status
- Resetting to Default Factory Settings
- Viewing Errors log
- Conducting a Self Test
- Setting Calibration Options
- Toggling Signal state
- Updating Firmware
- Upgrading Hardware Options

As you use the SFP to perform various actions, Keysight recommends to use **Tools** > **Monitor Driver Calls** to bring up the **Driver Call Log** window. This enables you to view the methods and properties that are used to program the module.

4.6.1: Connecting to Modules

The **Connect to Instrument** dialog enables you to select an M5302A module connected to a specific slot with a unique VISA address.

The selected configuration performs either of the following operations:

- Connects to the SFP of the physical hardware in a chassis
- Simulates the SFP session but not with an actual hardware connection

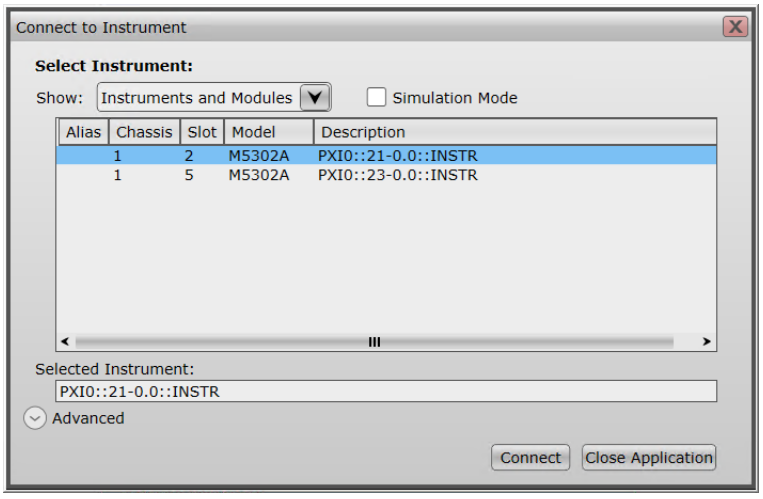


Figure 39 Connect to Instrument window in the M5302x SFP

To connect to a Module or Saved Instruments:

- Select one of the following options from the **Show** drop-down list. The displayed modules / configuration is filtered by the option you select in the **Show** drop-down list.
  - **Modules** – This element shows modules found through VISA discovery.

- Saved Instruments – This element shows the known instruments, which are defined by saving a configuration into the **IVI Configuration Store** after the SFP is up and running.

- The selected item is highlighted and displayed in the **Selected Instrument** entry box. An IVI logical instrument, a module, or Saved Instrument may also be selected by typing its name directly into the **Selected Instrument** entry box.

Connecting via Simulation mode

- **Simulation Mode**: When enabled, you can select the type of hardware which is being simulated. This works in conjunction with the Simulate button.
  - **Connect** / **Simulate** – When this button is labeled as **Connect**, it initiates the connection to the selected instrument or hardware module. If **Simulation Mode** is checked, this button is labeled as **Simulate**, as shown in Figure 40. Click the button to initiate a session with only the driver(s) for the selected instrument or module (not an actual hardware connection).
  - **Advanced** – Select this control to view the **Options**: dialog.

If you select the **Simulation Mode** check box and click the **Simulate** button after selecting a module / configuration, the underlying IVI driver (that the SFP is using) is opened in simulation mode. In this case, the selected resources may be ignored.

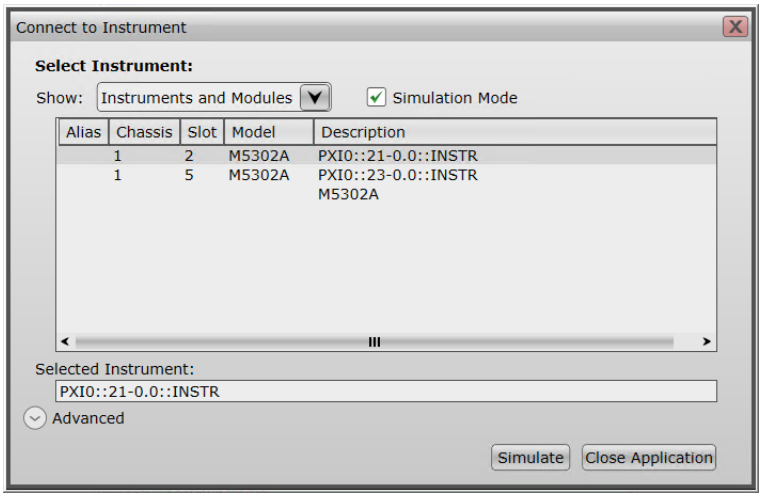


Figure 40 Simulation Mode in the Connect to Instrument window

NOTE An IVI driver can operate without the presence of an actual instrument by using simulation mode. When simulation is enabled, the driver does not communicate with the instrument but attempts to generate results, which allow client applications to run reasonably.

Simulation is designed to allow the driver to be used without an instrument or any I/O software or hardware. However, simulation is not intended to fully emulate the instrument. There is often no parameter range checking or coercion unless done in the driver. Most drivers depend on the instrument to perform range checking and coercion of parameter values. In simulation mode, this does not occur.

4.6.2: Checking Instrument Status

The status tab, which is displayed at the bottom of the SFP window, shows the status of the module in your Keysight instrument. You may also click **Refresh** to update the status (**View** > **Refresh**).



Figure 41 Status indicator when module is connected in the M5302x SFP



Figure 42 Status indicator in Simulation Mode in the M5302x SFP

The bottom-left corner of the SFP displays the following operational mode:

- **Simulation Mode**: This mode is displayed when the SFP is being used in "Simulation Mode" and no real hardware is being communicated with or controlled.
- **Not Connected**: This mode is displayed when no connections have been established with the module. This is typically displayed when the dialog box first opens instead of the **Connect to Instrument** window.
- **Connected**: This mode is displayed when the SFP has established a connection with a module.
  - If a single module is connected, the bottom-left corner shows "Connected" along with the VISA address.
  - If an alias is selected as the instrument to connect to, the bottom-left corner shows the alias.
  - The current drawn at Voltage values of 3.3V and 12V are also displayed.
- If you perform a factory reset on the connected module, the status tab displays "Reset complete" upon successful reset of the hardware settings. See Resetting to Default Factory Settings.

The bottom-right corner of the SFP displays the current status of the Calibration and Error Log:

- **Calibration status** – Displays the calibration type to which the module is subjected to.

NOTE **Calibration is not supported in the current version of the M5302x SFP. Therefore, the status shall display as "Not Subject to Periodic Calibration".**

- **No Error** – Displayed when there are no errors to report and the "Error Log" is empty.
- **Error** – Displayed when errors have occurred. You may click "Error" on the status bar or click **Utilities** > **Errors** to view the **Errors** window. Click **Get Instrument Errors** to view any errors encountered on the instrument.

4.6.3: Resetting to Default Factory Settings

- From the **Utilities** menu, select **Reset**.

The module's default factory settings are restored.

4.6.4: Viewing Errors log

This section provides the details of the reported errors.

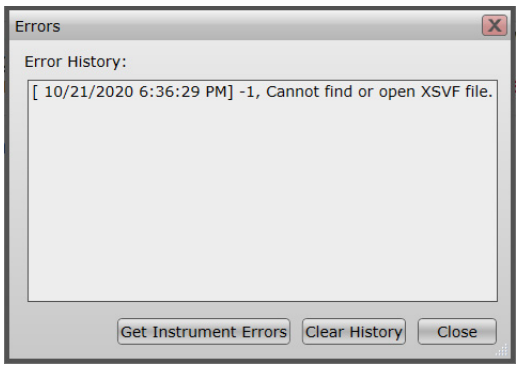


Figure 43 Viewing Errors log in M5302x SFP

To view the error log:

1 Select **Utilities** > **Errors**.

2 Click **Get Instrument Errors**.

All errors are displayed in the **Errors** dialog window.

NOTE If errors occur, the bottom-right corner of the SFP displays the word “Error”. Clicking the word “Error” opens the **Errors** dialog box. To learn more about instrument status, refer to Checking Instrument Status.

To clear the error log history:

- Click **Clear History**.

4.6.5: Conducting a Self Test

This section provides the details of conducting a Self Test.

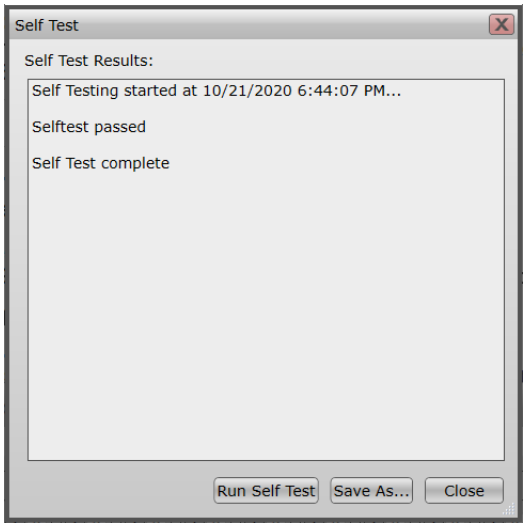


Figure 44 Running Self Test in M5302x SFP

To run Self Test on the module:

1 Click **Utilities** > **Self Test...**.

2 On the **Self Test** window that appears, click **Run Self Test**.

All Self Test errors are displayed in the **Self Test Results** area. If your Keysight M5302A Digital IO module passes its Self Test, the message “Selftest passed” is displayed.

NOTE Performing a Self Test may take several minutes.

To clear the self-test results window:

1 Close the **Self Test** window.

2 Re-open it by clicking **Utilities** > **Self Test...**.

This clears the Self Test results log, and refreshes the **Self Test Results** area.

To save Self Test results:

1 Click **Save As...**.

2 On the **Save As** dialog that appears, save the results of your Self Test as a text (.txt) file in a folder of your choice. If multiple files are saved in the same folder, the latest file is at the bottom of the stack.

4.6.6: Setting Calibration Options

The **Calibration Information and Preferences** window helps you with the following tasks:

- Displays the Calibration status
- Provides control for modifying a schedule

NOTE Calibration is not supported in the current version of the M5302x SFP. Therefore, the status shall display as “Not Subject to Periodic Calibration”.

Your Keysight M5302A Digital IO module comes fully calibrated from the factory.

Notice that the **Subject to Periodic Cal** check box and the **Module Cal Warnings** check boxes are enabled (checked) at the factory. The recommended calibration interval is 365 days for all modules that are subjected to periodic calibration.

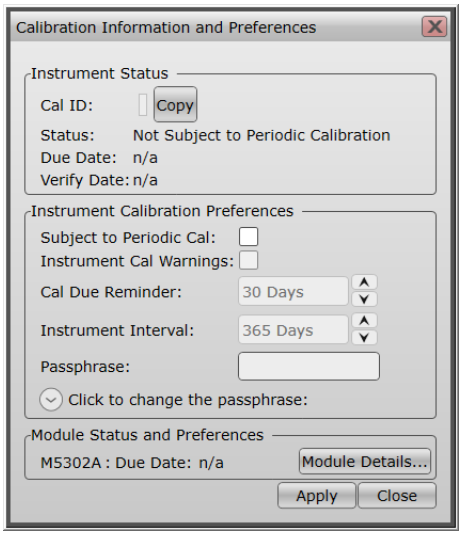


Figure 45 Calibration options in M5302x SFP

M5302A - Calibration Status

- **Status**: Displays the current calibration status.

  The status can be any of the following:

  - **Modules Calibrated**: Indicates that the calibration Due Date or Cal Due Reminder date has not yet reached
  - **Calibration Due**: Indicates that the calibration Due Date is within Cal Due Reminder days
  - **Calibration Expired**: Indicates that the calibration Due Date is expired
- **Due Date**: It is determined on the basis of Verify Date and Module Cal Interval dialogs.
- **Verify Date**: Set at the factory.

M5302A - Calibration Preferences

- **Subject to Periodic Cal**: Set at the factory, you can disable if not needed.
- **Module Cal Warnings**: When Module Cal Warnings is enabled, and Due Date is within Cal Due Reminder days, the Status field displays Calibration Due.
- **Cal Due Reminder**: Set at the factory, based on 365 days cycle.

- **Module Cal Interval**: Set at the factory, based on 365 days cycle.
- **Passphrase**: See the section below.

Setting Passphrase

A passphrase can be set if you want to protect Calibration Preferences settings. Once a passphrase is set, it is required when making additional changes to these settings. If a passphrase has never been set, the value is an empty string. This value is not affected by **Reset** or power cycle.

The values protected by Passphrase are:

- EnablePeriodicCalChecks
- EnableModuleCalWarnings
- ModuleCalInterval
- Passphrase

4.6.7: Toggling Signal state

NOTE The steps described in this section are applicable to all the banks. By default, the state of the input signal is set to '0' (low). You can toggle states for output signal only. However, you may set the state to '1' (high) and switch the signal from output to input.

Consider toggling the signal state on SMB connectors. The following image shows the default setting on each connector.



Figure 46 Default SMB IO connector states in M5302x SFP

To toggle the state for a specific pin / connector / line on the same or another IO bank:

1 Click the button displaying '**I**' (Input) so that it displays '**0**' (Output).
The button for the low (0) and high (1) states are enabled.
2 Click the button to toggle between '**0**' and '**1**', as required.
3 Repeat steps 1 and 2 for another pin / connector / line on the same or another IO bank.

Figure 47 shows the modified IO configuration for the SMB connectors.

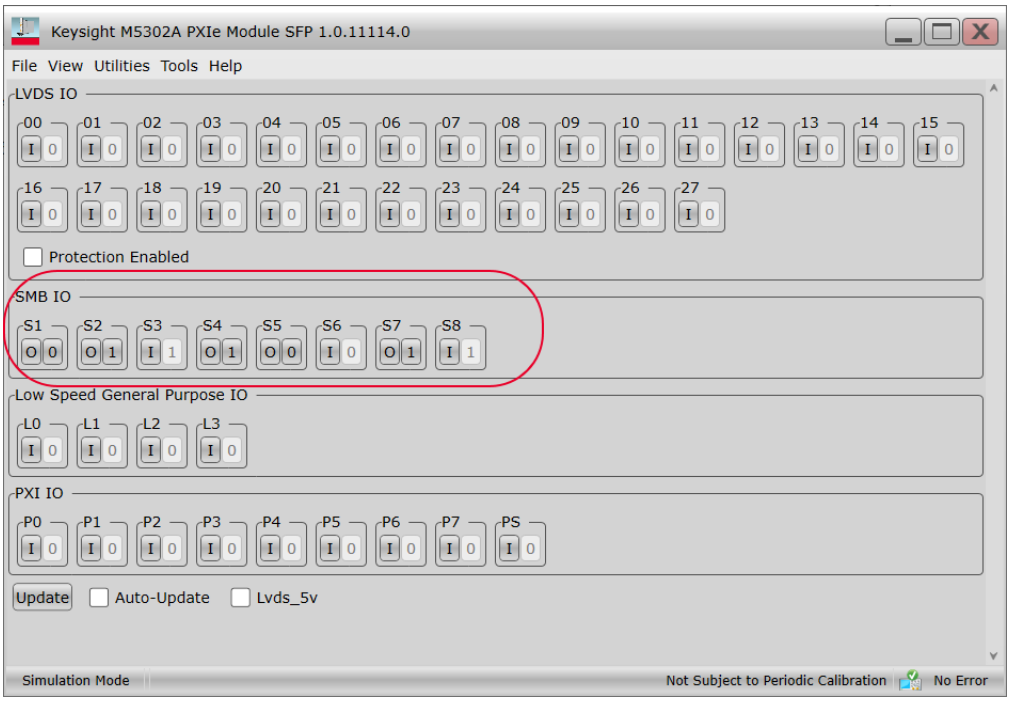


Figure 47 Modified SMB IO connector states in M5302x SFP

4 Once you have finished defining the required IO configuration, click **Update** for the changes to take effect on the M5302A module.

5 Optionally, you may select the **Auto-Update** check box, so that the SFP automatically modifies the configuration on the module.

6 Select **Lvds_5v** to drive a DC voltage of 5V amplitude as IO signal.

4.6.8: Updating Firmware

This section provides the details of updating the firmware.

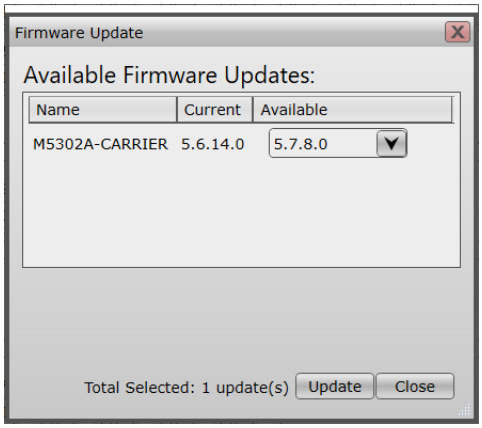


Figure 48 Performing Firmware update in M5302x SFP

Use the **Firmware Update...** option in the **Utilities** menu to update the firmware on a module.

To Update the Firmware in a Module:

1 Select a version of firmware for the module using the drop-down.

2 Click **Update** to install the selected Firmware version.

NOTE Updating the Firmware takes several minutes to complete. You can also select and install a previous version of the firmware.

4.6.9: Upgrading Hardware Options

This section provides the details of upgrading the options of the hardware that is being used in the M5302A Digital IO module.

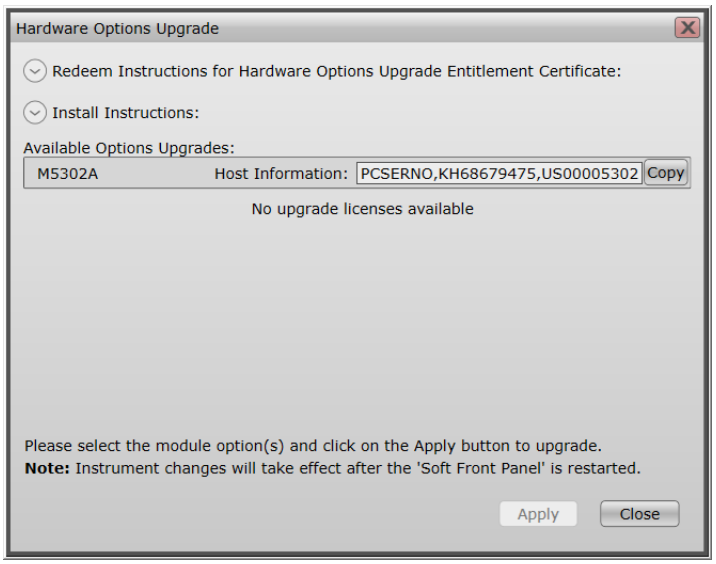


Figure 49 Performing Firmware update in M5302x SFP

Redeeming instructions

1 Once you have purchased a hardware options upgrade, you will receive a “Hardware Options Upgrade Entitlement Certificate”.

The entitlement certificate guarantees you access to the licensed capabilities of your Keysight hardware options upgrade.

2 From the list of displayed “Available Options Upgrades”, select **Copy** next to the module you plan to upgrade.

This saves the Host Information to the Microsoft Windows clipboard.

3 Access the Keysight Software Manager (KSM) website and use the entitlement certificate you received to add licenses to your profile.

Once added to KSM, the website shows how many licenses you are entitled to assign to hosts.

- Log onto the Keysight Software Manager website www.keysight.com/find/softwaremanager.
  - If you are a ‘First Time User’, you must Register with an email address and select a password.

    This allows you to create a profile, which is needed before you can add entitlement certificates or assign a license to a host.
  - If you are a ‘Returning User’, enter your email, password, and select **Login**.
- Select the **Licenses** tab.

  You should see all of your currently assigned licenses and each of their corresponding hosts.

  If you have never entered information from an entitlement certificate before, the table shows that there are no licenses available.

- Select **Get more licenses...**.

  You will be prompted with, “Do you want to add another certificate to your profile?”

  The steps are determined based on your response:

Selecting Yes (if you have entitlement certificates that you would like to add to your profile)

1 Select the “Yes” check box and then click **Continue**.

2 Enter the **Order Number** and **Certificate Number** located on the top of your “Hardware Options Upgrade Entitlement Certificate.”

  The entitlement certificate number is not case sensitive, but must include the dash.

  This will add your entitlement certificate information to the Keysight Software Manager database. After completing your registration, you can assign these licenses to a host.

3 Click **Continue** to Add Licenses.

4 Click **Add to My Profile** to add these licenses to your profile.

5 You will then be prompted with, “Do you want to add another certificate to your profile?”

  You can repeat this process again by checking “Yes” or you can check “No” and follow the process for assigning licenses.

Selecting No (if you do not have additional entitlement certificates, but you are ready to assign licenses)

1 Select the “No” check box and then click **Continue**.

2 Assign a License to a Host.

  - Select a Host from the **Select Host** drop-down menu.

  (If the host is not in the list, select **Add a new host...** and you are prompted for the Host Information associated with the host you are trying to add.)

  - Select one of the Licenses to be assigned to the selected Host.
  - Click **Assign Licenses**. (This assigns one License to the selected Host.)
  - Click **Continue**.
  - When finished assigning Licenses to selected Hosts, check “No” and select **Continue**.

3 Click **Request Licenses** to request the license assignment between the selected License and the selected Host.

  This will trigger an email to be sent to you with a license file of the form: *filename.lic*.

4 Check your email system and retrieve the email with the license file attachment.

5 Save the license file on your file system.

6 Start the Keysight License Manager program by selecting **Start** > **All Programs** > **Keysight** > **Keysight License Manager** > **Keysight License Manager**.

  - Select **File** > **Install** and using a Windows browser, navigate to *filename.lic*.
  - Start the Soft Front Panel of the Keysight PXI module associated with the license capability and verify the new capability functions.

Understanding Host Information

The *Host Information* is a permanent, unique identifier to Keysight instrument or application that locks licenses to that particular instrument or application. If the Host Information provided at the time the license is generated is wrong, the license will not enable the features purchased.

NOTE It is important to enter the correct Host Information. Your license will not enable the new features if the Host Information does not exactly match your instrument or application.

Host Information for PXIe modules is a character string and differs depending on the process being used.

Host Information is the: Host ID, Serial Number

For Example: *PCSERNO, FS67755648, US12345678*

- uses Hardware Options Upgrade Entitlement Certificate

5. Using M5302x API functions in sample programs

Implementing M5302x API functions – Sample Programs 68

This chapter describes how to use the Keysight SD1 Programming Libraries with Python.

Section 5.1: Implementing M5302x API functions – Sample Programs

The M5302x Programming Libraries can be located in *C:\Program Files\Keysight\M5302x*.

The following sample programs, using Python, give you an understanding of the work flow for one or more use cases. While the work flow is the same for other programming languages, the syntax vary based on the programming language being used.

5.1.1: Sample program to perform & check toggling on SMB IO Channels

```
"""
keysight_ktm5302x Python API Example Program
Creates a driver object, toggles a SMB output channel, and checks for the toggle
on an SMB input channel.
Requires Python 3.6 or newer and keysight_ktm5302x Python extension source module.
"""

import sys
import os
import keysight_ktm5302x
import numpy as np # For keysight_ktm5302x arrays

def main():
    """
   Edit resource_name and options as needed.
   resource_name is ignored if option Simulate=true
    For this example, resource_name may be a VISA address(e.g.
"PXI0::CHASSISchassis::SLOTslot::INSTR")
    or a VISA alias.
   For more information on using VISA aliases, refer to the Keysight IO Libraries Connection
Expert documentation.
    """
    resource_name = "PXI0::11-0.0::INSTR"

    #  Edit the initialization options as needed
    idQuery = False
    reset   = False
    options = "Simulate=False"

    try:
        print("\n  keysight_ktm5302x Python API Example1\n")

        # Call driver constructor with options
```

```
    global driver # May be used in other functions
    driver = None
    driver = keysight_ktm5302x.KtM5302x(resource_name, idQuery, reset, options)
    print("Driver Initialized")

    #  Print a few identity properties
    print('  model:      ', driver.identity.instrument_model)
    print('  resource:   ', driver.driver_operation.io_resource_descriptor)
    print('  options:    ', driver.driver_operation.driver_setup)
    print('  simulate:   ', driver.driver_operation.simulate)

   # Toggle SMB output and validate on SMB input.
   # Assumes SMB 1 is connected to SMB 2.
    print("\nSMB channels count: ", driver.smb_channels.count)
    source  = 0
    destination = 1
    print("  Source Name: ", driver.smb_channels[source].name)
    print("  Destination Name: ", driver.smb_channels[destination].name)

    driver.smb_channels[source].direction = keysight_ktm5302x.ChannelDirection.OUT
    print("  Source Direction: ", driver.smb_channels[source].direction)

    driver.smb_channels[destination].direction = keysight_ktm5302x.ChannelDirection.IN
    print("  Destination Direction: ", driver.smb_channels[destination].direction)

    driver.smb_channels[source].state = keysight_ktm5302x.ChannelState.LOGIC_LOW
    print("  Destination State: ", driver.smb_channels[destination].state)

    driver.smb_channels[source].state = keysight_ktm5302x.ChannelState.LOGIC_HIGH
    print("  Destination State: ", driver.smb_channels[destination].state)
    print()

    # Check instrument for errors
    print()
    while True:
        outVal = ()
        outVal = driver.utility.error_query()
        print("  error_query: code:", outVal[0], " message:", outVal[1])
        if(outVal[0] == 0):
            break

except Exception as e:
```

```
            print("\n  Exception:", e.__class__.__name__, e.args)

        finally:
            if driver is not None: # Skip close() if constructor failed
                driver.close()
            input("\nDone - Press Enter to Exit")

if __name__ == "__main__":
    main()

# © Keysight Technologies, 2021
# All rights reserved.
# You have a royalty-free right to use, modify, reproduce and distribute
# this Sample Application (and/or any modified # version) in any way you find useful,
# provided that you agree that Keysight Technologies has no warranty, obligations or liability
# for any Sample Application Files.
#
# Keysight Technologies provides programming examples for illustration only.
# This sample program assumes that you are familiar with the programming language
# being demonstrated and the tools used to create and debug procedures.
# Keysight Technologies support engineers can help explain the functionality
# of Keysight Technologies software components and associated commands,
# but they will not modify these samples to provide added functionality or
# construct procedures to meet your specific needs.
# ----------
```

5.1.2: Sample program to load a Sandbox design onto an M5302A module

```
"""
keysight_ktm5302x Python BSP Example Program
Creates a driver object, loads a sandbox design, and performs read/write operations.
Requires Python 3.6 or newer and keysight_ktm5302x Python extension source module.
"""

import sys
import os
import keysight_ktm5302x
import numpy as np # For keysight_ktm5302x arrays

def main():
    """
    Edit resource_name and options as needed.  resource_name is ignored if option Simulate=true
```

```
    For this example, resource_name may be a VISA address(e.g.
"PXI0::CHASSISchassis::SLOTslot::INSTR")
    or a VISA alias.  For more information on using VISA aliases, refer to the Keysight IO
    Libraries Connection Expert documentation.
    """
    resource_name = "PXI0::11-0.0::INSTR"

    #  Edit the initialization options as needed
    idQuery = False
    reset   = False
    options = "Simulate=False"

    try:
        print("\n  keysight_ktm5302x Python BSP Example1\n")

        # Call driver constructor with options
        global driver # May be used in other functions
        driver = None
        driver = keysight_ktm5302x.KtM5302x(resource_name, idQuery, reset, options)
        print("Driver Initialized")

        # Print some FPGA properties
        sandbox = driver.fpga_sandboxes['Application']
        print("\nDevice UUID : ", sandbox.static_region_id) # Unique ID
        # of the currently loaded static region, 128 bits, formatted as a string
        # of 32 hex characters.
        print("Sandbox index : ", sandbox.sandbox_index)
        print("Sandbox name : ", sandbox.name)
        print("Kernel id : ", sandbox.kernel_id) # Unique ID
        # of the currently loaded sandbox content identifying the algorithm.
        print("FPGA name : ", sandbox.fpga_name)
        print("FPGA version : ", sandbox.fpga_version)

        # Load FPGA sandbox with default example k7z file
        k7z_file = "example.k7z"
        sandbox.configure_from_k7z(k7z_file)

        # Given a register name, return address and other information.
        name = 'AxilRegisterBank_AxilReg_0' # modify to register name in your bsp design.
        access_type = 'RW' # or 'Memory Map'
        register_entry = ()
        register_entry = sandbox.get_register_entry(name, access_type)
```

```
            address = register_entry[0]
            length =  register_entry[1]
            access_type = register_entry[2]
            print("Register address : ", address)
            print("Register length : ", length)
            print("Register access type : ", access_type)

            # Poke / Peek a 32-bit register in the sandbox
            print("\nPEEK/POKE test...")
            address_offset = 0
            data = 100
            print("Poke : ", data)
            sandbox.poke(address, data)
            ret_val = sandbox.peek(address + address_offset)
            print("Peek : ", ret_val)

            # Check instrument for errors
            print()
            while True:
                outVal = ()
                outVal = driver.utility.error_query()
                print("  error_query: code:", outVal[0], " message:", outVal[1])
                if(outVal[0] == 0):
                    break

        except Exception as e:
            print("\n  Exception:", e.__class__.__name__, e.args)

        finally:
            if driver is not None: # Skip close() if constructor failed
                driver.close()
            input("\nDone - Press Enter to Exit")

if __name__ == "__main__":
    main()

# © Keysight Technologies, 2021
# All rights reserved.
# You have a royalty-free right to use, modify, reproduce and distribute
# this Sample Application (and/or any modified # version) in any way you find useful,
# provided that you agree that Keysight Technologies has no warranty, obligations or liability
# for any Sample Application Files.
```

```
#
# Keysight Technologies provides programming examples for illustration only.
# This sample program assumes that you are familiar with the programming language
# being demonstrated and the tools used to create and debug procedures.
# Keysight Technologies support engineers can help explain the functionality
# of Keysight Technologies software components and associated commands,
# but they will not modify these samples to provide added functionality or
# construct procedures to meet your specific needs.
# ----------
```

5.1.3: Sample program to create an HVI sequence for an SMB trigger

```
"""
keysight_ktm5302x Python API Example Program
Creates an HVI sequence that waits for smb trigger, and sets the HVI register to true
when trigger condition is met.
Requires Python 3.7 or newer and keysight_ktm5302x Python extension source module
installed using Package Installer for Python (pip install <fileName>.tar.gz).
"""

    import sys
    import os
    import keysight_hvi
    import keysight_ktm5302x as denim
    from datetime import timedelta
    import time

    def main():
            try:
"""
Edit resource_name and options as needed.
resource_name is ignored if option Simulate=true
For this example, resource_name may be a VISA address(e.g.
"PXI0::CHASSISchassis::SLOTslot::INSTR") or a VISA alias.
For more information on using VISA aliases, refer to the Keysight IO
Libraries Connection Expert documentation.
"""
                    # Change to False to use real module hardware
                    #simulate = False

                    # Open Keysight module with HVI capability
                    resource = 'PXI0::9-0.0::INSTR'
```

```
resource1 = 'PXI0::13-0.0::INSTR'
query = False
reset = False
options = 'Simulate=0, LogLevel=Info'
module = denim.KtM5302x(resource, query, reset, options)
module1 = denim.KtM5302x(resource1, query, reset, options)

# create System Definition
system_def = keysight_hvi.SystemDefinition("MyHviSystem")

# add clocks
system_def.non_hvi_system_clocks = [100.0e6]

# add the module engine to the System Definition
engine_name = 'My Engine'

cmi = module.get_instance()
ktm5302xhvi = denim.KtM5302xHvi(cmi)

print("HVI Engine ID in slot 2:  ", ktm5302xhvi.engines.primary)

system_def.engines.add(ktm5302xhvi.engines.primary, engine_name)
if len(system_def.engines) == 0:
        raise Exception('System Definition has no engines')

def_engine = system_def.engines[0]
 # Configure a smb denim trigger
 # that follows the pxi event (PXI event->SMB6 toggle)
smb3_trigger_id = ktm5302xhvi.triggers.smb2
smb3_trigger = def_engine.triggers.add(smb3_trigger_id, 'smb3')
smb3_trigger.config.sync_mode = keysight_hvi.SyncMode.IMMEDIATE
smb3_trigger.config.direction = keysight_hvi.Direction.INPUT

# Make HVI sequencer
hvi_sequencer = keysight_hvi.Sequencer("myName", system_def)

 # Create sequence
 sync_sequence = hvi_sequencer.sync_sequence # Obtain main sequence
 # from the created HVI instrument
 sequence = sync_sequence.add_sync_multi_sequence_block
 ('SyncExec', 10).sequences[engine_name]
```

```
hvi_done = sync_sequence.scopes[engine_name].registers.add
("hvi_done", keysight_hvi.RegisterSize.SHORT)

wait_condition = keysight_hvi.Condition.trigger(smb3_trigger)
wait_event = sequence.add_wait('wait external_trigger',
10, wait_condition)
wait_event.set_mode(keysight_hvi.WaitMode.TRANSITION,
keysight_hvi.SyncMode.IMMEDIATE)

instruction = sequence.add_instruction("hvi_done = 1",
10, sequence.instruction_set.assign.id)
instruction.set_parameter(sequence.instruction_set.assign.destination.id,
hvi_done)
instruction.set_parameter(sequence.instruction_set.assign.source.id, 1)

print("Compiling and running Sequence!")

# Compile and run the HVI sequence
hvi = hvi_sequencer.compile()       # Compile the instrument sequence(s)

hvi.load_to_hw()    # Load the instrument sequence(s) to HW

hvi.run(hvi.no_wait)       # Execute the instrument sequence(s)

print("Waiting for external trigger")

while True:
        # Wait for HVI sequence loop to finish
        time.sleep(.5)

    if hvi.sync_sequence.scopes[engine_name].registers['hvi_done'].read()
== 1:
                print("Trigger received")
                break

        print("Press any key to send smb trigger")
        input()

        #send smb trigger
        module1.smb_channels[2].direction = denim.ChannelDirection.OUT
        module1.smb_channels[2].state = denim.ChannelState.LOGIC_HIGH
        module1.smb_channels[2].state = denim.ChannelState.LOGIC_LOW
```

```
                        module1.smb_channels[2].state = denim.ChannelState.LOGIC_HIGH

                hvi.release_hw()
                module.close()

                print("Test complete!")

        except Exception as ex:
                print(ex)
                print('Exiting')

if __name__ == '__main__':
        main()

# © Keysight Technologies, 2021
# All rights reserved.
# You have a royalty-free right to use, modify, reproduce and distribute
# this Sample Application (and/or any modified # version) in any way you find useful,
# provided that you agree that Keysight Technologies has no warranty, obligations or liability
# for any Sample Application Files.
#
# Keysight Technologies provides programming examples for illustration only.
# This sample program assumes that you are familiar with the programming language
# being demonstrated and the tools used to create and debug procedures.
# Keysight Technologies support engineers can help explain the functionality
# of Keysight Technologies software components and associated commands,
# but they will not modify these samples to provide added functionality or
# construct procedures to meet your specific needs.
# ----------
```

6. Documentation References

Accessing related Help documents for M5302A 78
Links to other documents 79

This section provides references to the links where you can access one or more documents, as required.

Section 6.1: Accessing related Help documents for M5302A

The installer for the M5302x software creates a shortcut in the *Keysight M5302x PCIe Module* folder in the *Start* menu, for the files shown in Table 17.

Table 17 Related Help files for M5302A software and respective folder locations

Name	Filename	Folder location
KtM5302x C++ API Help	index.html	C:\Program Files\Keysight\M5302x\Help
M5302x HVI Add-on Help	Keysight_KtM5302xHvi_Fx45.chm	
M5302x IVI.Net Help	Keysight_KtM5302x_Fx45.chm	
M5302x IVI-C Help	KtM5302x.chm	
Python API Help	index.html	C:\Program Files\Keysight\M5302x\python\html

Section 6.2: Links to other documents

Document Reference	Reference location
M5302A PXIe Digital IO module Startup Guide	https://www.keysight.com/find/M5302A-TechSupport
KS2201A PathWave Test Sync Executive User Manual	https://www.keysight.com/main/techSupport.jspx?cc=IN&lc=eng&nid=-33330.1294480&pid=3069213&pageMode=PL
KF9000A PathWave FPGA (Home Page)	https://www.keysight.com/us/en/products/software/pathwave-test-software/pathwave-fgpa-software.html *PathWave FPGA Customer Documentation* can be accessed from the **Help** menu of the design environment.
BSP Guide	
M5302A PXIe Digital IO module BSP Help	Accessed from the Help menu of the *PathWave FPGA 2020 Update 1.1* design environment.
Data Sheet	
M5302A PXIe LVDS Digital IO Data Sheet	https://www.keysight.com/find/M5302A-TechSupport

Index

B

board support package, 31

F

FPGA code, 29
FPGA configurable region, 29
FSP, 31

H

Hardware Virtual Instrumentation, 14
HVI Core API, 15

I

IP-XACT, 27

K

KtHviPlatform, 13
KtHviSequence, 13

P

Partial Reconfiguration, 29

R

RSP, 31

S

sandbox region, 29
SD1 API, 15
static region, 29

X

Xilinx FPGAs, 26
Xilinx Vivado Design Suite, 28

Index

This information is subject to
change without notice.
© Keysight Technologies 2020-2021
Edition 1.0, March 2021

M5302-91003
www.keysight.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP