PathWave
FPGA 2020

M3102A PXIe Digitizer

KEYSIGHT
TECHNOLOGIES

Notice

Table of Contents

This section is a supplement for https://literature.cdn.keysight.com/litweb/pdf/M3100-90002.pdf

Note: The above link refers to an old version of Keysight M31XXA/M33XXA Digitizer User Guide. An updated version of this document will be released in the future. (Link to be updated soon)

Installation Requirements

In addition to the PathWave FPGA system requirements, this BSP requires the following software:

Vendor	Software / Feature	Beta Release Supported	May work, but not supported	Release Explicitly not-supported
Keysight	HVI-2 Design Environment (optional)	0.68.0/alpha	Older versions	
Keysight	SD1 SFP Drivers, programming libraries and Software Front Panel for the following modules: M3102A, M3201A, M3202A	3.00.82 to 3.xx.xx	Older versions	2.xx.xx

Overview

The M3102A devices are high performance, high-bandwidth digitizers with an advanced data acquisition system (DAQ).

The following document outlines the performance specifications for the M3102A product family: https://literature.cdn.keysight.com/litweb/pdf/5992-1805EN.pdf

Note: The above link refers to an old version of Keysight M3102A Data Sheet. An updated version of this document will be released in the future. (Link to be updated soon)

The M31/M33XXA User's Guide provides a Theory of Operation and Software Guide: https://literature.cdn.keysight.com/litweb/pdf/M3100-90002.pdf

Note: The above link refers to an old version of Keysight M31XXA/M33XXA Digitizer User Guide. An updated version of this document will be released in the future. (Link to be updated soon)

Scope

This document is a technical reference of the M3102A digitizer with regard to the configurable region of the M3102A FPGA. The FPGA configurable region utilizes Xilinx FPGA partial reconfiguration technology to allow customers to design and configure a defined section of the M3102A FPGA without the need to power down or reboot the M3102A or host computer, respectively.

Available FPGA Features and Resources

The M3102A may be purchased as a 4 channel configuration, which has fixed sampling clock.

An M3102A may also be purchased to include one of two possible Xilinx FPGAs:

1. xc7k410tffg676-2
2. xc7k325tffg676-2

The xc7k410tffg676-2 part offers a substantial increase in the amount of FPGA resources available to the user for custom logic. The following table outline the FPGA resources which are available for custom logic for each part.

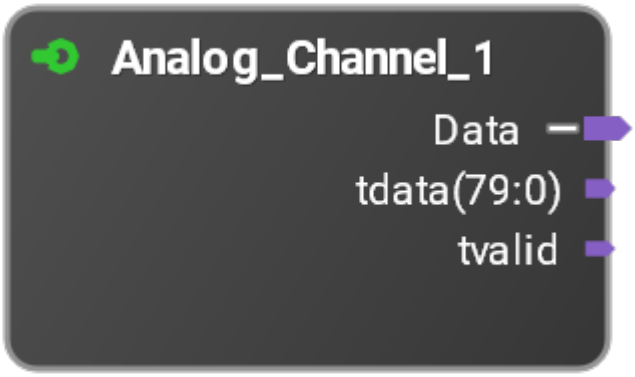
	xc7k325tffg676-2	**xc7k410tffg676-2**
Resource Type	4 Channel	4 Channel
Slice LUTs	82800	102000
Slice Registers	165600	204000
DSP Blocks	360	660
Block Ram (RAMB18)	360	660

FPGA Configurable Region Interfaces

The following sections describe the physical interfaces which are available within an M3102A FPGA configurable region.

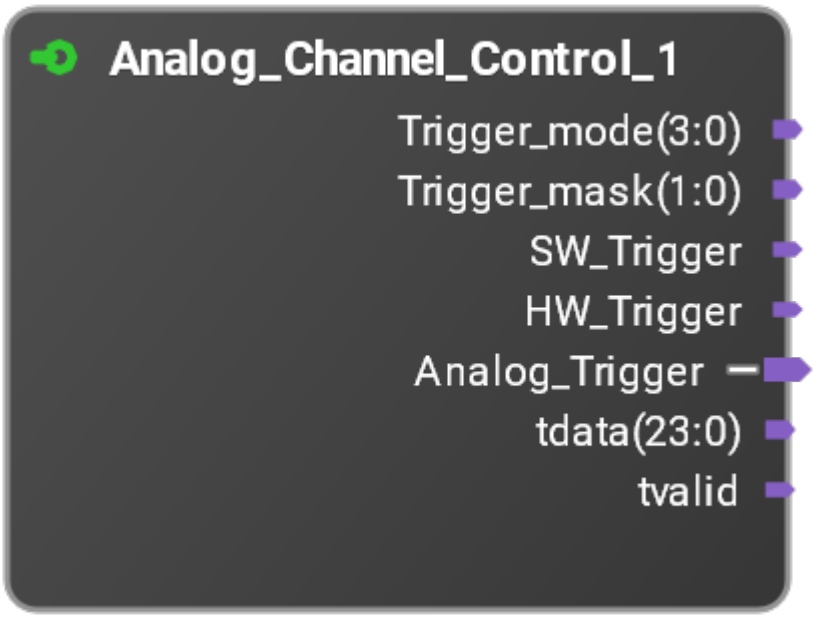
The DAQ, Host interfaces, trigger, analog_trigger, and the HVI interfaces are synchronous to CLK100. The triggerIn, triggerOut, User_trigger, and PXItriggers are synchronous to other clocks. Please see the module's User's Guide for a more detailed discussion of the trigger clocking system before using these signals.

Analog Channel Input (4x)



Interface name	Signal name	Width (bits)	Bit field	Description
Data		80		AXI streaming interface for data captured from the digitizer
	tdata	15:0	data_0	Sample 0
		31:16	data_1	Sample 1
		47:32	data_2	Sample 2
		63:48	data_3	Sample 3
		79:64	data_4	Sample 4
	tvalid	1		

Analog Channel Control (4x)



Interface name	Signal name	Width (bits)	Bit field	Description
Trigger_mode		4		0: Auto trigger 1: Software trigger (trigger_sw) 2: Hardware trigger (trigger_hw) 3: Analog trigger (from ADC data) Others: unused
Trigger_mask		4		Trigger Mask: enable analog triggering on specified channel
		3:3	Mask 3	Enable channel 3 triggering
		2:2	Mask 2	Enable channel 2 triggering
		1:1	Mask 1	Enable channel 1 triggering
		0:0	Mask 0	Enable channel 0 triggering
SW_Trigger		1		Software trigger signal used when trigger_mode==1
HW_Trigger		1		Hardware trigger signal used when trigger_mode==2
Analog_Trigger		24		AXI streaming interface for trigger control
	tdata	15:0	Threshold	Trigger threshold value
		17:16	Trigger Mode	0: No trigger 1: Positive 2: Negative 3: Both
		23:18	Unused	

Interface name	Signal name	Width (bits)	Bit field	Description
	tvalid	1		

Analog trigger output (4x)

Interface name	Width (bits)	Description
Trigger_Out	5	Analog trigger output

Data acquisition (4x)

Interface name	Signal name	Width (bits)	Bit field	Description
DaqX		80		AXI streaming interface for data acquisition interface
	tdata	15:0	data_0	Sample 0
		31:16	data_1	Sample 1
		47:32	data_2	Sample 2
		63:48	data_3	Sample 3
		79:64	data_4	Sample 4
	tvalid	1		

Host interface

The M3102A provides a host interface for reading and writing registers from a host PC. The user can select between a memory-mapped host interface or a register host interface. If a memory-mapped interface is selected, the user can choose between an AXI-Lite, AXI, or Mem interface.

PXI trigger input (8x)

Interface name	Width (bits)	Description
PXItriggerInX	1	A PXI backplane trigger signal sent from the backplane to the sandbox.

PXI trigger output (x8)

Interface name	Width (bits)	Description
PXItriggerOutX	1	A PXI backplane trigger signal sent from the sandbox to the backplane.

Trigger Output (4x)

Interface name	Width (bits)	Description
Trigger_Out	5	Trigger output for the DAQ.

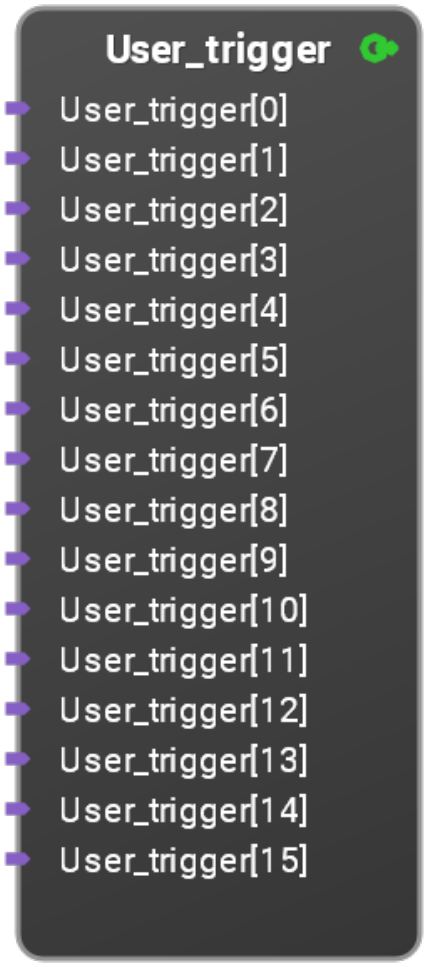
Front panel trigger input (1x)

Interface name	Width (bits)	Description
triggerIn	0:0	Trigger input from the M3102A front panel trigger connector.

Front panel trigger output (1x)

Interface name	Width (bits)	Description
triggerOut	0:0	Trigger output to the M3102A front panel trigger connector.

User Trigger (1x)

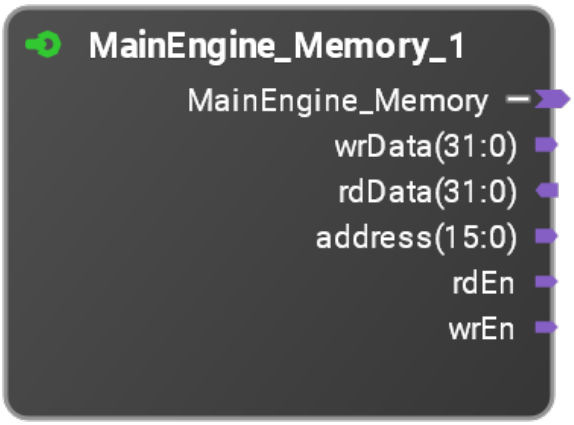


Interface name	Width (bits)	Description
User_trigger[x]	1	User triggers that are sent to the HVI engine (not fully supported).

Real-time HVI interface

The M3102A provides a real-time HVI interface. The user can select between a memory-mapped HVI interface or a register HVI interface. The user can use HVI actions input to sandbox and drive HVI events going tout of sandbox.

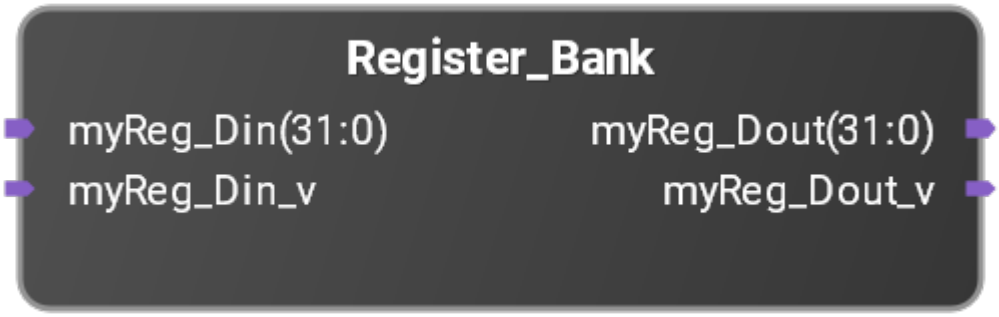
HVI Memory-Mapped interface



Interface name	Signal name	Width (bits)	Description
MainEngine_memory		32	
	wrData	32	Write data
	rdData	32	Read data
	address	16	Address
	rdEn	1	Read enable
	wrEn	1	Write enable

The MainEngine_memory works at a main system clock of 100 MHz.

HVI Register interface



Input Signals

Interface name	Width (bits)	Description
myReg_X_Din	32	Data from this port is written to a register that can be read by the host PC.
myReg_X_Din_valid	1	Latches Din into the internal register

Output Signals

Interface name	Width (bits)	Description
myReg_X_Dout	32	Data from this port comes from a host accessible register and can be read in the sandbox.
myReg_X_Dout_valid	1	

HVI_UserAction (8x)

Interface name	Width (bits)	Description
HVI_UserActionX	1	An HVI user action input signal in the sandbox.

HVI_UserEvent(x8)

Interface name	Width (bits)	Description
HVI_UserEventX	1	An HVI user event output signal from the sandbox.

100 MHz clock source (main system clock)

Interface name	Width (bits)	Description
clock	1	100 MHz clock source (main system clock)

200 MHz clock source

Interface name	Width (bits)	Description
Clk200	1	200 MHz clock source

Active low reset

Interface name	Width (bits)	Description
nRst	1	Active low reset

Sync 100

Interface name	Width (bits)	Description
sync100	1	Sync signal from the PXI backplane.

Sync Clock

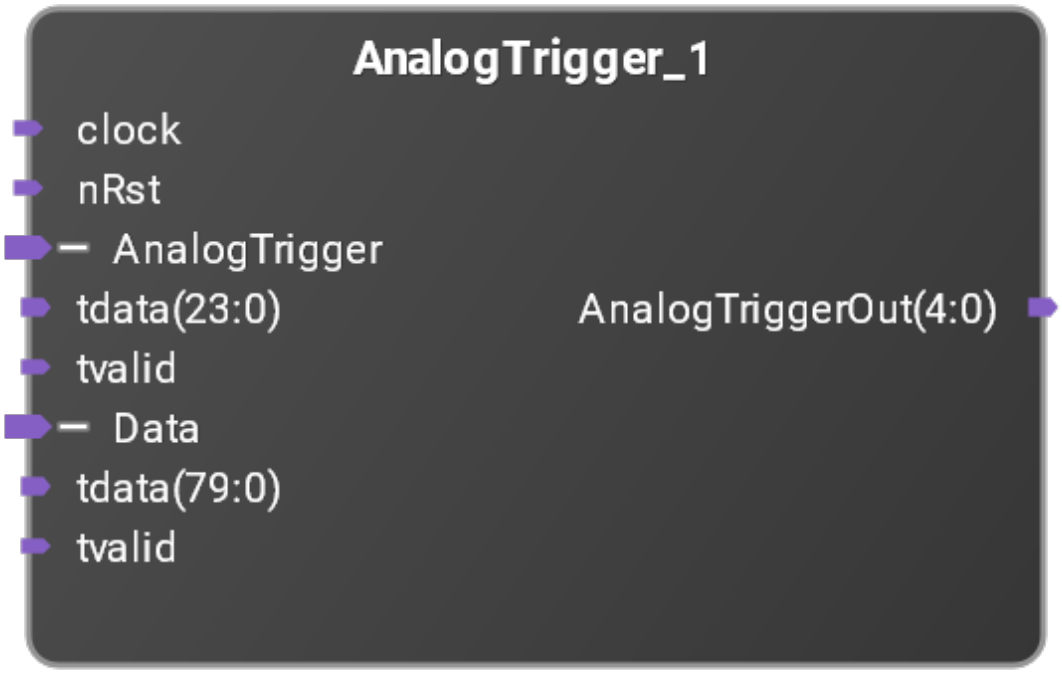
Interface name	Width (bits)	Description
syncClk	1	Sync signal referenced to the FPGA clock.

DDR Memory(1x)

The interface for accessing external DDR memory is an AXI4 Memory Mapped Slave interface.

BSP IP Repository

Analog Trigger block



Input Signals

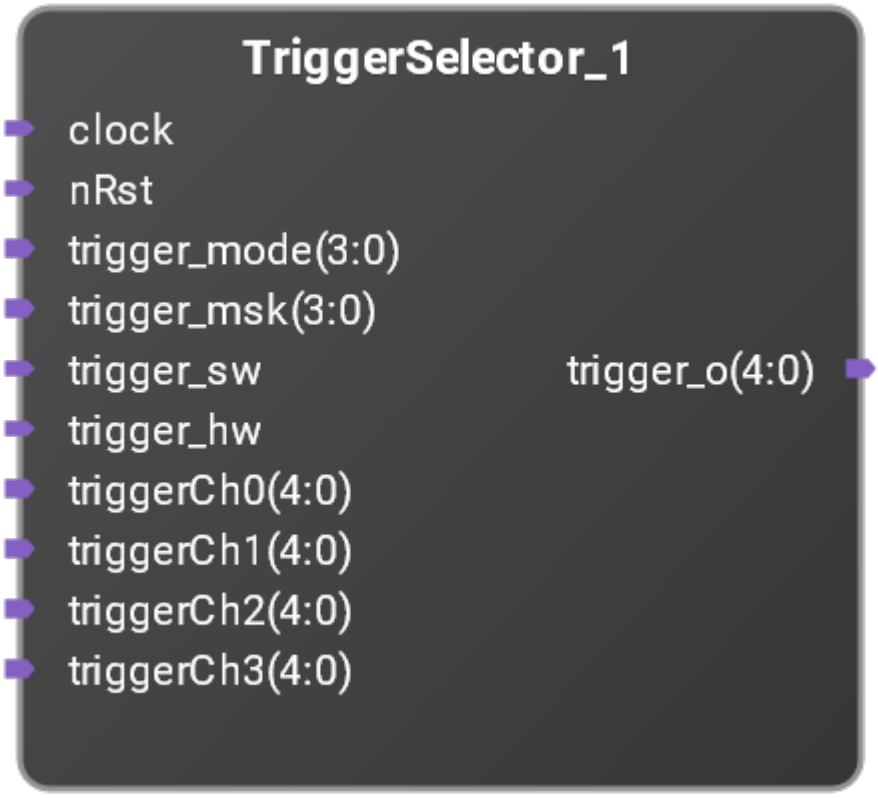
Interface name	Signal name	Width (bits)	Bit Field	Description
clock		1		Clock input
nRst		1		Reset input (active low)
AnalogTrigger		24		AXI streaming interface for trigger control
	tdata	15:0	Threshold	Trigger threshold value
		17:16	Trigger Mode	0: No trigger 1: Positive 2: Negative 3: Both
		23:18	Unused	
	tvalid	1		
Data		80		AXI streaming interface for ADC data input. (5x supersampled)
	tdata	15:0	data_0	Sample 0
		31:16	data_1	Sample 1

		47:32	data_2	Sample 2
		63:48	data_3	Sample 3
		79:64	data_4	Sample 4
	tvalid	1		

Output signals

Signal name	Width (bits)	Bit Field	Description
AnalogTriggerOut	5		Analog Trigger Result Output
	0:0	Trigger 0	Trigger output for sample 0
	1:1	Trigger 1	Trigger output for sample 1
	2:2	Trigger 2	Trigger output for sample 2
	3:3	Trigger 3	Trigger output for sample 3
	4:4	Trigger 4	Trigger output for sample 4

Trigger Selector block (4 channel)



Input Signals

Signal name	Width (bits)	Description
clock	1	Clock input
nRst	1	Reset input (active low)
trigger_mode	4	0: Auto trigger 1: Software trigger (trigger_sw) 2: Hardware trigger (trigger_hw) 3: Analog trigger (from ADC data) Others: unused
trigger_msk	4	Trigger Mask: enable analog triggering on specified channel
	3:3	Enable channel 3 triggering
	2:2	Enable channel 2 triggering
	1:1	Enable channel 1 triggering
	0:0	Enable channel 0 triggering
trigger_sw	1	Software trigger signal used when trigger_mode==1
trigger_hw	1	Hardware trigger signal used when trigger_mode==2
triggerCh0	5	Analog trigger signal for channel 0 used when trigger_mode==3 and trigger_msk[0]==1. Supersampled by 5 (one bit per sample)
triggerCh1	5	Analog trigger signal for channel 1 used when trigger_mode==3 and trigger_msk[1]==1. Supersampled by 5 (one bit per sample)
triggerCh2	5	Analog trigger signal for channel 2 used when trigger_mode==3 and trigger_msk[2]==1. Supersampled by 5 (one bit per sample)
triggerCh3	5	Analog trigger signal for channel 3 used when trigger_mode==3 and trigger_msk[3]==1. Supersampled by 5 (one bit per sample)

Output Signal

Signal name	Width (bits)	Description
trigger_o	5	Trigger output signal selected by trigger_mode and trigger_msk. Supersampled by 5 (one bit per sample)

How to use the SD1 and RSP API

PathWave FPGA does not provide access to the waveform and digitizer controls in the Keysight M3XXXA PXIe modules. To use front panel analog IO with these modules, it is necessary to use the SD1 API to control the hardware.

A second API is provided to work with designs created with PathWave FPGA. This API provides convenient access to the program archive file (the "k7z" file) created by PathWave FPGA.

The user must decide whether to use both API's in the program or to use only the Keysight SD1 API.

Using SD1 API with PathWave FPGA User Design and API

The advantages of using the PathWave FPGA API include symbolic name access to registers and buses in the user design. Because addresses can change when a design is recompiled, using symbolic names is more robust than using hard-coded addresses.

- The PathWave FPGA API loads the FPGA image from the program archive file. It is not appropriate to use the SD1 soft front panel or the SD1 API to load the FPGA image directly from the k7z file.
- On some AWG modules, it may be necessary to set the output amplitude on channels after the FPGA image is loaded.
- PathWave FPGA API calls to access the user design and SD1 API calls to access the analog IO may be intermixed.

Using SD1 API with PathWave FPGA User Design

It is possible to use only the SD1 API. There are some additional steps for accessing the user design in the sandbox.

- The generated .k7z file may be loaded with either the SD1 soft front panel, or the SD1 API "`FPGAload`" function.
- Retrieve all the registers added to Pathwave FPGA design using SD1 API "**FPGAgetSandBoxRegisters**". This API would return the list of SD_SandBoxRegister objects. Get the register name using **Name** property of SD_SandBoxRegister object.
- It is also possible to retrieve a single register using the SD1 API "**FPGAgetSandBoxRegister**" and Pathwave FPGA register name.
- The SD1 APIs **writeRegisterInt32/readRegisterInt32** and **writeRegisterBuffer/readRegisterBuffer** can be used to to write/read single or multiple DWords

Usage Example

The following example demonstrates how to open a k7z file, load the FPGA image binary file into the module, and read and write named registers:

```
import sys

sys.path.append('C:\Program Files (x86)\Keysight\SD1\Libraries\Python')
import keysightSD1


#Set product details
product=''
chassis=1
slot=9
```

```
# open a module
module=keysightSD1.SD_Module()
moduleID=module.openWithSlot(product,chassis,slot)

if moduleID < 0:
   print ("Module open error: ",moduleID)
else:
   print("Module is: ",moduleID)

#Loading FPGA sandbox using .K7z file
error = module.FPGAload(r'..\myHardwareTest.k7z')

numRegisters = 4
#Get Registers list
registers = module.FPGAgetSandBoxRegisters(numRegisters)

#Print the register properties in register list
for register in registers:
print(register.Name);
print(register.Length);
print(register.Address);
print(register.AccessType);

registerName = 'Register_Bank_A'
#Get Sandbox Register with name "Register_Bank_A"
registerA = module.FPGAgetSandBoxRegister(registerName)

#Write data to Register_Bank_A
error = registerA.writeRegisterInt32(9)

registerNameB = 'Register_Bank_B'
#Get Sandbox Register with name "Register_Bank_B"
registerB = module.FPGAgetSandBoxRegister(registerNameB)
```

```
#Write data to Register_Bank_B
error = registerB.writeRegisterInt32(9)


registerNameC = 'Register_Bank_C'

#Get Sandbox Register with name "Register_Bank_C"
sandbox_register_C = module.FPGAgetSandBoxRegister(registerNameC)


#Read data from Register_Bank_B
error = sandbox_register_C.readRegisterInt32()


memoryMap = 'Host_mem_1'

#Get Sandbox memoryMap with name "Host_mem_1"
memory_Map = module.FPGAgetSandBoxRegister(memoryMap)


#Write buffer to memory map
memory_Map.writeRegisterBuffer(0, [1,2,3, 4, 5, 6],
keysightSD1.SD_AddressingMode.AUTOINCREMENT, keysightSD1.SD_AccessMode.DMA)


#Read buffer from memory map
c_value = memory_Map.readRegisterBuffer(0, 6,
keysightSD1.SD_AddressingMode.AUTOINCREMENT,
keysightSD1.SD_AccessMode.NONDMA)
```