

User's
Guide

Keysight
M3602A
FPGA Design
Environment



Notices

Copyright Notice

© Keysight Technologies 2017

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, Inc. as governed by United States and international copyright laws.

Manual Part Number

M3602-90001

Published By

Keysight Technologies
1400 Fountaingrove Parkway
Santa Rosa
CA 95405

Edition

Edition 1, March 2017
Printed In USA

Regulatory Compliance

This product has been designed and tested in accordance with accepted industry standards, and has been supplied in a safe condition. To review the Declaration of Conformity, go to <http://www.keysight.com/go/conformity>.

Warranty

THE MATERIAL CONTAINED IN THIS DOCUMENT IS PROVIDED "AS IS," AND IS SUBJECT TO BEING CHANGED, WITHOUT NOTICE, IN FUTURE EDITIONS. FURTHER, TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, KEYSIGHT DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, WITH REGARD TO THIS MANUAL AND ANY INFORMATION CONTAINED HEREIN, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. KEYSIGHT SHALL NOT BE LIABLE FOR ERRORS OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE

FURNISHING, USE, OR PERFORMANCE OF THIS DOCUMENT OR OF ANY INFORMATION CONTAINED HEREIN. SHOULD KEYSIGHT AND THE USER HAVE A SEPARATE WRITTEN AGREEMENT WITH WARRANTY TERMS COVERING THE MATERIAL IN THIS DOCUMENT THAT CONFLICT WITH THESE TERMS, THE WARRANTY TERMS IN THE SEPARATE AGREEMENT SHALL CONTROL. KEYSIGHT TECHNOLOGIES DOES NOT WARRANT THIRD-PARTY SYSTEM-LEVEL (COMBINATION OF CHASSIS, CONTROLLERS, MODULES, ETC.) PERFORMANCE, SAFETY, OR REGULATORY COMPLIANCE, UNLESS SPECIFICALLY STATED.

Technology Licenses

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

U.S. Government Rights

The Software is "commercial computer software," as defined by Federal Acquisition Regulation ("FAR") 2.101. Pursuant to FAR 12.212 and 27.405-3 and Department of Defense FAR Supplement ("DFARS") 227.7202, the U.S. government acquires commercial computer software under the same terms by which the software is customarily provided to the public. Accordingly, Keysight provides the Software to U.S. government customers under its standard commercial license, which is embodied in its End User License 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.

CAUTION ces

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.

The following safety precautions should be observed before using this product and any associated instrumentation.

This product is intended for use by qualified personnel who recognize

shock hazards and are familiar with the safety precautions required to avoid possible injury. Read and follow all installation, operation, and maintenance information carefully before using the product.

WARNING

If this product is not used as specified, the protection provided by the equipment could be impaired. This product must be used in a normal condition (in which all means for protection are intact) only.

The types of product users are:

- Responsible body is the individual or group responsible for the use and maintenance of equipment, for ensuring that the equipment is operated within its specifications and operating limits, and for ensuring operators are adequately trained.
- Operators use the product for its intended function. They must be trained in electrical safety procedures and proper use of the instrument. They must be protected from electric shock and contact with hazardous live circuits.
- Maintenance personnel perform routine procedures on the product to keep it operating properly (for example, setting the line voltage or replacing consumable materials). Maintenance procedures are described in the user documentation. The procedures explicitly state if the operator may perform them. Otherwise, they should be performed only by service personnel.
- Service personnel are trained to work on live circuits, perform safe installations, and repair products. Only properly trained service personnel may perform installation and service procedures.

WARNING

Operator is responsible to maintain safe operating conditions. To ensure safe operating conditions, modules should not be operated beyond the full temperature range specified in the Environmental and physical specification. Exceeding safe operating conditions can result in shorter lifespans, improper module

performance and user safety issues. When the modules are in use and operation within the specified full temperature range is not maintained, module surface temperatures may exceed safe handling conditions which can cause discomfort or burns if touched. In the event of a module exceeding the full temperature range, always allow the module to cool before touching or removing modules from chassis.

Keysight products are designed for use with electrical signals that are rated Measurement Category I and Measurement Category II, as described in the International Electrotechnical Commission (IEC) Standard IEC 60664. Most measurement, control, and data I/O signals are Measurement Category I and must not be directly connected to mains voltage or to voltage sources with high transient over-voltages. Measurement Category II connections require protection for high transient over-voltages often associated with local AC mains connections. Assume all measurement, control, and data I/O connections are for connection to Category I sources unless otherwise marked or described in the user documentation.

Exercise extreme caution when a shock hazard is present. Lethal voltage may be present on cable connector jacks or test fixtures. The American National Standards Institute (ANSI) states that a shock hazard exists when voltage levels greater than 30V RMS, 42.4V peak, or 60VDC are present. A good safety practice is to expect that hazardous voltage is present in any unknown circuit before measuring.

Operators of this product must be protected from electric shock at all times. The responsible body must ensure that operators are prevented access and/or insulated from every connection point. In some cases, connections must be exposed to potential human contact. Product operators in these circumstances must be trained to protect themselves from the risk of electric shock. If the circuit is capable of operating at or above 1000V,

no conductive part of the circuit may be exposed.

Do not connect switching cards directly to unlimited power circuits. They are intended to be used with impedance-limited sources. NEVER connect switching cards directly to AC mains. When connecting sources to switching cards, install protective devices to limit fault current and voltage to the card.

Before operating an instrument, ensure that the line cord is connected to a properly-grounded power receptacle. Inspect the connecting cables, test leads, and jumpers for possible wear, cracks, or breaks before each use.

When installing equipment where access to the main power cord is restricted, such as rack mounting, a separate main input power disconnect device must be provided in close proximity to the equipment and within easy reach of the operator.

For maximum safety, do not touch the product, test cables, or any other instruments while power is applied to the circuit under test. ALWAYS remove power from the entire test system and discharge any capacitors before: connecting or disconnecting cables or jumpers, installing or removing switching cards, or making internal changes, such as installing or removing jumpers.

Do not touch any object that could provide a current path to the common side of the circuit under test or power line (earth) ground. Always make measurements with dry hands while standing on a dry, insulated surface capable of withstanding the voltage being measured.

The instrument and accessories must be used in accordance with its specifications and operating instructions, or the safety of the equipment may be impaired.

Do not exceed the maximum signal levels of the instruments and accessories, as defined in the specifications and operating information, and as shown on the instrument or test fixture panels, or switching card.

When fuses are used in a product, replace with the same type and rating for continued protection against fire hazard.

Chassis connections must only be used as shield connections for measuring circuits, NOT as safety earth ground connections.

If you are using a test fixture, keep the lid closed while power is applied to the device under test. Safe operation requires the use of a lid interlock.

Instrumentation and accessories shall not be connected to humans.

Before performing any maintenance, disconnect the line cord and all test cables.

To maintain protection from electric shock and fire, replacement components in mains circuits – including the power transformer, test leads, and input jacks – must be purchased from Keysight. Standard fuses with applicable national safety approvals may be used if the rating and type are the same. Other components that are not safety-related may be purchased from other suppliers as long as they are equivalent to the original component (note that selected parts should be purchased only through Keysight to maintain accuracy and functionality of the product). If you are unsure about the applicability of a replacement component, call an Keysight office for information.

WARNING

No operator serviceable parts inside. Refer servicing to qualified personnel. To prevent electrical shock do not remove covers. For continued protection against fire hazard, replace fuse with same type and rating.

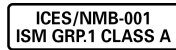
PRODUCT MARKINGS:



The CE mark is a registered trademark of the European Community.



Australian Communication and Media Authority mark to indicate regulatory compliance as a registered supplier.



This symbol indicates product compliance with the Canadian Interference-Causing Equipment Standard (ICES-001). It also identifies the product is an Industrial Scientific and Medical Group 1 Class A product (CISPR 11, Clause 4).



South Korean Class A EMC Declaration. This equipment is Class A suitable for professional use and is for use in electromagnetic environments outside of the home. A 급 기기 (업무용 방송통신기자재) 이 기기는 업무용 (A 급) 전자파적합기기로서 판매자 또는 사용자는 이 점을 주의하시기 바라며, 가정외의 지역에서 사용하는 것을 목적으로 합니다.



This product complies with the WEEE Directive marketing requirement. The affixed product label (above) indicates that you must not discard this electrical/electronic product in domestic household waste. **Product Category:** With reference to the equipment types in the WEEE directive Annex 1, this product is classified as “Monitoring and Control instrumentation” product. Do not dispose in domestic household waste. To return unwanted products, contact your local Keysight office, or for more information see

<http://about.keysight.com/en/companyinfo/environment/takeback.shtml>.



This symbol indicates the instrument is sensitive to electrostatic discharge (ESD). ESD can damage the highly sensitive components in your instrument. ESD damage is most likely to occur as the module is being installed or when cables are connected or disconnected. Protect the circuits from ESD damage by wearing a grounding strap that provides a high resistance path to ground. Alternatively, ground yourself to discharge any built-up static charge by touching the outer shell of any grounded instrument chassis before touching the port connectors.



This symbol on an instrument means caution, risk of danger. You should refer to the operating instructions located in the user documentation in all cases where the symbol is marked on the instrument.



This symbol indicates the time period during which no hazardous or toxic substance elements are expected to leak or deteriorate during normal use. Forty years is the expected useful life of the product.

Contents

1 Introduction	1
1.1 M3602A a Graphical FPGA Design Environment	1
1.1.1 Workflow	3
1.2 M3602A Basics	6
1.2.1 Hardware Project	6
1.2.2 FPGA Block Project	6
1.2.3 GUI Menu Selections	6
1.2.3.1 Hardware Projects and FPGA Block GUI Selections	6
1.2.3.2 Keysight M3602A Menu Selections	6
1.2.3.3 Keysight M3602A Icons Selections	7
1.2.4 Creating the Main Hardware Project	8
1.2.5 Hardware/FW Management	11
1.2.5.1 Assigning Hardware	11
1.2.5.2 Generate Firmware	13
1.2.5.3 Load Firmware	13
1.2.5.4 Hardware and Library Components	14
1.2.5.5 Other IP Blocks	17
1.2.6 Creating an FPGA Block Project	17
1.2.7 Ports and Interfaces	19
1.2.7.1 Connection Rules	21
1.2.7.2 Interface Roles	23
1.2.8 Importing User IPs	23
1.2.9 Importing Vivado IPs	26
1.2.10 Generating FPGA Bitstream	28
1.2.10.1 Build Errors and Warnings	31
1.2.11 Project Directory Structure	31
1.3 Editing Actions	33
1.3.1 Basic Controls	33
1.3.1.1 Zooming in and out	33
1.3.1.2 Span	33
1.3.1.3 Fit in Window	33
1.3.1.4 Multiple Selections	33
1.3.1.5 Copy Action	34
1.3.1.6 Move Items	34
1.3.1.7 Undo/Redo Action	34
1.3.2 Connecting Ports	35
1.3.3 Adding and Editing Blocks	38
1.3.3.1 M3602A Different IPs	38
1.3.3.2 Built-In Blocks	38

1.3.3.3 Library Ports (FPGA Block Project)	40
1.3.3.4 Adding User Blocks	40
1.3.3.5 Adding an FPGA Block	40
1.3.4 Adding and Editing Integers	41
1.3.4.1 Connecting Decimals, Hexadecimals, and Binaries	43
1.3.5 Adding and Editing Comments	43
2 Software Programming Guide: PC-FPGA Interaction	45
2.1 Programming Functions	45
2.1.1 SW Programming Overview Programming Libraries	45
2.1.2 SD_Module Configurations	45
2.1.2.1 Addressing Mode	45
2.1.2.2 Accessing Mode	45
2.1.2.3 Reset Mode	46
2.1.3 SD_Module Functions	46
2.1.3.1 FPGAwritePCport	46
2.1.3.2 FPGAreadPCport	47
2.1.3.3 FPGAload	48
2.1.3.4 FPGAreset	49
2.1.4 Error Codes	50
2.1.4.1 Open and Close Errors	50
2.1.4.2 ID Errors	51
3 Addendum: Keysight Technology and Software Overview	57
3.1 Programming Tools	57
3.1.1 SW Programming	57
3.1.1.1 Keysight Programming Libraries	58
3.1.2 HW Programming	58
3.1.2.1 HVI Technology: Keysight M3601A	58
3.1.2.2 FPGA Programming: Keysight M3602A	62
3.1.2.3 Keysight M3602A: An FPGA Design Environment	63
3.2 Design Process: Customization vs. Complete Design	64
3.3 Application Software	64
3.3.1 Keysight SD1 SFP	64
3.4 M3602A Initial Setup	65
3.4.1 Add License to Product	65
3.4.2 Vivado 2015.2 Installation	67
3.4.3 Cloud Server Connection	69

1 Introduction

Most of the modular PXIe products adopt an FPGA based architecture, implementing the digital interfaces to the underlined hardware such as ADC/DAC, clock and memory management, trigger and sync, etc. The off-the-shelf feature-rich functionalities of the modules and the control (e.g. IQ modulator, function generators or queue system for the AWGs or advanced triggers and DAQs for the digitizers) are also implemented in the logic.

However, standard products typically do not have available resources for additional processing functions. 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 the standard hardware products. For these applications, Keysight supplies hardware products that provide the capability to program the on-board FPGA.

M3602A FPGA programming software is compatible with **M3xxxA** PXIe modular hardware products. M3602A provides the necessary tools to design, compile and program the FPGA of the module. **M3xxxA** offering is made of **AWGs** and **digitizers** that allow engineers to build onto the instrument's core capabilities or insert custom algorithms into the on-board FPGAs. This graphical design environment makes it easy to add customization required for emerging technologies, research and design while accessing the full performance and speed of the FPGA.

1.1 M3602A a Graphical FPGA Design Environment

The M3602A Design Environment requires and is dependent on the Vivado 2015.2 WebPACK license being present.

The M3602A FPGA Design Environment provides a graphical FPGA programmable environment that allows for customization capability. To take advantage of the M3602A software, it is important to order the -FP1 hardware option when purchasing the hardware. When configuring the hardware with option -FP1, select which FPGA wanted to be loaded on the hardware. FPGAs that needs to work with option -FP1 are bigger than those that come with the standard hardware.

The M3602A software enables non-expert and expert users to achieve high-performant solutions (e.g. loop control for DRFM, DPD* filtering for AWGs, data compression*, frequency masking, for data acquisition, etc.) based on the true parallelism of FPGA via a simple graphical interface.

The graphical interface:

- Is easy to use
- Enables fast FPGA hardware programming (see [Figure 1](#))
- Draws on a large body of FPGA solving building blocks (see [Figure 2](#) and [Figure 3](#))
- Verifies the FPGA solution at each step
- Is an end-to-end solution resulting in compiled code being loaded into the target FPGA

* DPD Digital pre-distortion is used to pre-distort the signal going into a power amplifier to accommodate for non-linearity as the chip approaches the max power. Data compression is needed due to the high data throughput of current ADC and limitations of Data bandwidth from digitizer to host using PCIe x4 Gen2

Fig1



Figure1 Avoid FPGA labyrinth

By design, the control of the main digital interfaces is kept out of the reach of the customizable area of the FPGA. This enables non-expert and expert users to be able to focus on the following tasks.

Non-expert users can (non-HDL programmers):

- Take benefit of the ready-to-use M3602A block library and the IP Core blocks of the library yet achieving full FPGA bandwidth performance (non-performance penalty).
- Access custom processing functions or algorithms with M3601A HVIs, bringing off-the-shelf, inter-module synchronization, phase coherence channel capabilities, and time deterministic real-time processing.
- It is also possible to free-up FPGA programming space for customer code so that users can build on the products core capabilities or add in their custom code into the FPGA with faster project deployment and no FPGA performance penalty.

Expert-users can:

- Access custom processing functions or algorithms with M3601A HVIs, bringing off-the-shelf, inter-module synchronization, phase coherence channel capabilities, and time deterministic real-time processing.

- Import existing IP developed in expert low-level FPGA programming tools such as VHDL, Verilog or Xilinx® VIVADO/ISE projects, Xilinx CORE Generator IP Cores or include MATLAB/SIMULINK® code.



- Take benefit of the ready-to-use M3602A block library and the IP Core blocks of the library yet achieving full FPGA bandwidth performance (non-performance penalty).

1.1.1 Workflow

In addition to the faster compiling, it features hot programming capability where a module can be hot programmed with existing compiled code via PCIe without rebooting the instrument.

Referring to figure 2:

1. User opens an M3602A hardware project
2. Assembles the solution (see also figure 3)
3. Compiles using a cloud server and generates the binary
4. Hot programming: binary is loaded into the FPGA of the hardware

Fig2



Figure2 M3602A workflow

Figure3 shows the relation between the different types of blocks available in M3602A. The control of the main digital interfaces is kept out of the reach (locked FPGA features in the FPGA project) of the customizable area of the FPGA

A **Hardware project** integrates some locked blocks such as the Hardware blocks and Hardware ports which are integrated into the control system of the digital interface. A description of the different functional blocks is included below:

Hardware block: These blocks are an integral part of every hardware project. They can be used with the M3602A system design environment to create custom FPGA solutions. These blocks come along with the purchased hardware module. They have been designed to work together with the colored blocks described as follows when creating a custom FPGA solution.

Associated with the hardware blocks are **hardware ports** as shown in Figure 3. These ports are locked so that the user cannot remove them. They are only available in hardware projects and are responsible for communications between the internally configurable FPGA part (the FPGA customizable space which user can edit) and the rest of FPGA.

- **Library block:** These blocks are provided with every new project. They are generally simple and provide commonly used FPGA functionality.
- **User block:** These blocks are created and saved by the user using different FPGA vendor tools or from source code directly. These blocks can be imported to the project using the Block Editor. The user is the owner of these blocks which can be used them for extending or creating specific functionalities to create the FPGA solution.
- **FPGA user block:** This block is designed in the M3602A system design environment using the FPGA Block Project application.

- **FPGA vendor block**: This block is available for purchase from third party vendors. Powerful functional blocks can be added to the solution considerably reducing development time. The M3602A integrates their importation into the project and facilitates their management.
- Associated with the FPGA Block Projects are **Library ports**. These ports are configurable so as to interface with other blocks in a hardware or other FPGA Block project.

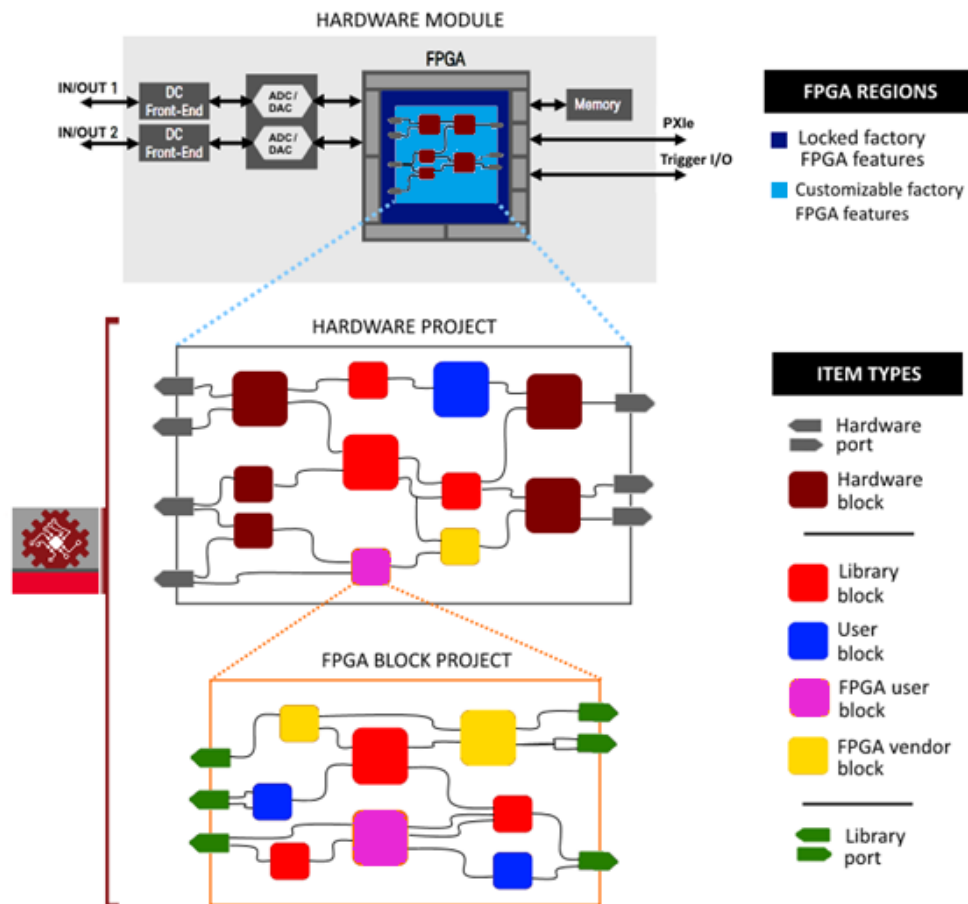


Figure3 Ports, blocks and Project relations

1. 2 M3602A Basics

There are two project types:

- Hardware project designed for a specific M3xxxA hardware product
- FPGA block project for user generated IPs

1. 2. 1 Hardware Project

See [Creating the Main Hardware Project \(page 8\)](#)

A hardware project contains the customizable resources of the programmable FPGA of an M3xxxA hardware module. When selecting a target module, the project is opened with the factory settings of a standard module. The custom on-board solution is developed within this hardware project and is saved, compiled and loaded into the hardware module (the binary can be loaded into multiple identical modules). As per Figure 3, part of the hardware project can be made up of FPGA user block projects, which are created as follows.

1. 2. 2 FPGA Block Project

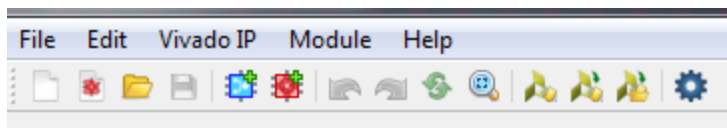
See [Creating an FPGA Block Project \(page 17\)](#)

The purpose of an FPGA Block project is to create independent user FPGA blocks. They can be designed to provide custom functionality to other projects. These blocks once created, can be saved and reused in other hardware projects, and in other FPGA Block projects as shown in Figure 3.

1. 2. 3 GUI Menu Selections

1. 2. 3. 1 Hardware Projects and FPGA Block GUI Selections

When a Hardware Project or FPGA Block editor window is launched, the following GUI choices are available. See tables 1 and 2 for more information.



1. 2. 3. 2 Keysight M3602A Menu Selections

Menu Item	Description (see Table 2 for Icon numbers)
File	Includes Menu Icons 1 - 6 and Save As ..., Recent Project, Close, Settings, and Exit
Edit	Includes Menu Icons 7 - 8, and Select All
Vivado IP	Includes Menu Icons 11, 12, and 13
Module	Enables assigning hardware, generating, and loading firmware
Help	License Management... and About...

Table:1

1. 2. 3. 3 Keysight M3602A Icons Selections















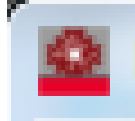
	Icon	Description
1		Creates a new hardware project
2		Creates a new FPGA Block
3		Open a file, such as a project file
4		Saves a project file
5		Adds an External Block
6		Adds an FPGA Block
7		Undoes changes
8		Redoes changes
9		Redraw Connections
10		Fit In Window
11		Launches the Vivado Tool
12		Imports Vivado IPs
13		Imports Vivado IP Locations
14		Generates firmware

Table: 2

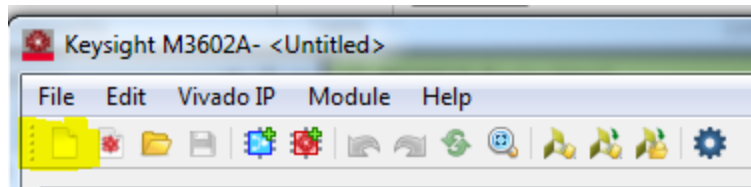
1.2.4 Creating the Main Hardware Project

As per section 1.2.1 a hardware project contains the customizable resources of the programmable FPGA of an M3xxxA hardware module. When created, the project can be edited, saved, compiled and loaded into the hardware module (the binary can be loaded into multiple identical modules). These are the steps to create a hardware project:

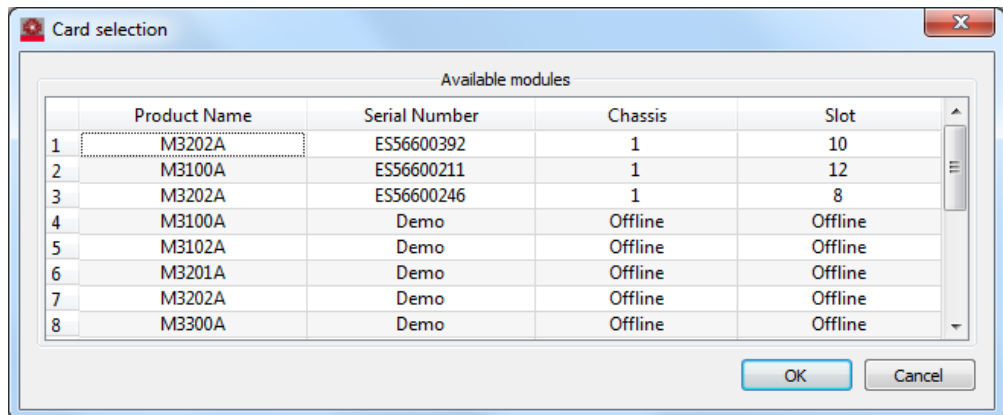
1. Launch the application program by clicking on the M3602A software icon.



2. To create a new hardware project, click on the blank icon shown highlighted as shown as follows.



3. The card selection dialog box appears. Provided that M3602A is installed in the chassis controller, the cards plugged into the chassis will be presented. Also, a comprehensive collection of the M3xxxA demo modules is selectable.



When one of the populated chassis cards is selected, M3602A gathers the hardware information, and option buttons are grayed (hardware options are defined at the time of purchase and are not upgradable).

Options

CH: CH8 CLK: CLF FPGA: FP1-K32 RAM: M20 DMod: None HVI: None

Card selection

Available modules

	Product Name	Serial Number	Chassis	Slot
1	M3202A	ES56600392	1	10
2	M3100A	ES56600211	1	12
3	M3202A	ES56600246	1	8
4	M3100A	Demo	Offline	Offline
5	M3102A	Demo	Offline	Offline
6	M3201A	Demo	Offline	Offline
7	M3202A	Demo	Offline	Offline
8	M3300A	Demo	Offline	Offline

Options

CH: CH8 CLK: CLF FPGA: FP1-K32 RAM: M20 DMod: None HVI: None

OK Cancel

Click **OK**.

Alternatively, Demo modules can be loaded with multiple options.

Card selection

Available modules

	Product Name	Serial Number	Chassis	Slot
1	M3100A	Demo	Offline	Offline
2	M3102A	Demo	Offline	Offline
3	M3201A	Demo	Offline	Offline
4	M3202A	Demo	Offline	Offline
5	M3300A	Demo	Offline	Offline

Options

CH: CH2 CLK: CLF FPGA: FP1-K32 RAM: M20 DMod: DM1 HVI: HV1

OK Cancel

For populated real hardware modules, step 4 can be skipped.

4. Configure the scroll down buttons options.

Options

CH: CH2 CLK: CLF FPGA: FP1-K32 RAM: M20 DMod: DM1 HVI: HV1

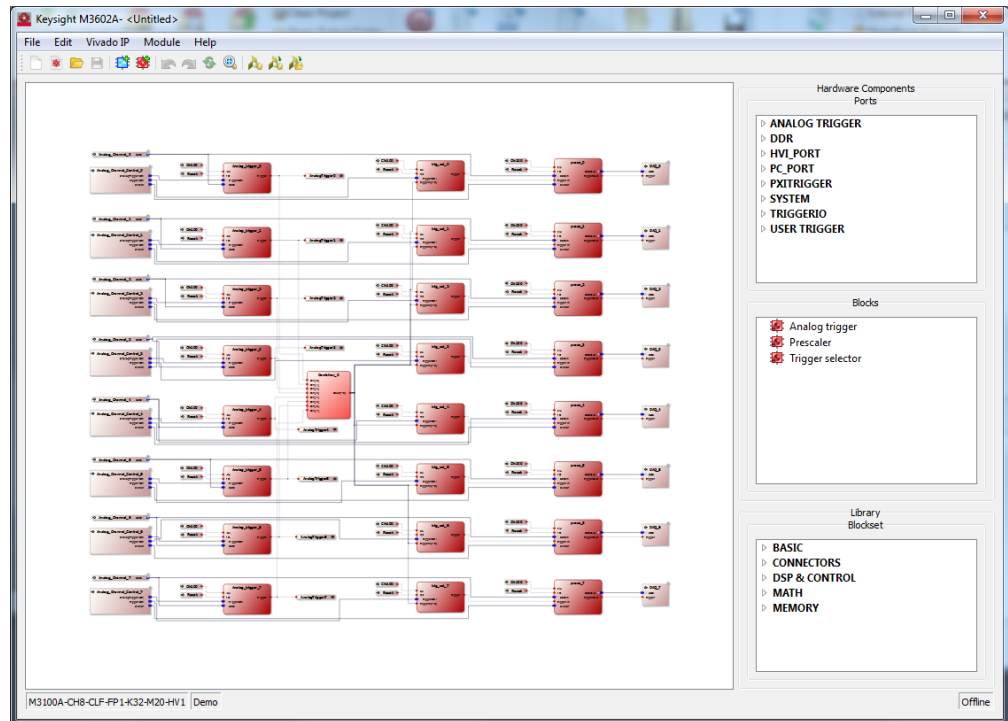
Where:

Option	Description
CH:	Select the number of channels: for example, CH4 or CH8
CLK:	Select the clock: CLF or CLV
FPGA:	Select the FPGA, for example, FP1-K32 or FP1-K41
RAM:	Select the RAM, for example, M20,
DMod:	Select the DMod, for example, DM1 or None
HVI	Select the HVI, for example, HV1 or None

NOTE More information can be found in the M31/M33XX Digitizer User's Guide and the M32/M33XX AWG User's Guide.

5. If the module has been configured correctly, a new “untitled” hardware project will be pulled.

Populated chassis hardware will be pulled with its product name.



6. Review the default hardware configuration. Default functionality for each hardware module comes with ports, blocks and their connections. The default configuration can be edited for additional processing functions. See [Editing Actions \(page 33\)](#)
7. Add hardware and library components from the selections available on the right side of the GUI, see [Hardware and Library Components \(page 14\)](#)
8. Add component blocks as needed to create the hardware project, see: [Adding and Editing Blocks \(page 38\)](#)

[Importing User IPs \(page 23\)](#)

[Importing Vivado IPs \(page 26\)](#)

9. Connect the signals between the blocks, see [Connecting Ports \(page 35\)](#)
10. Once the hardware project has been completed, save it, and give it an appropriate title, using **File > Save As**

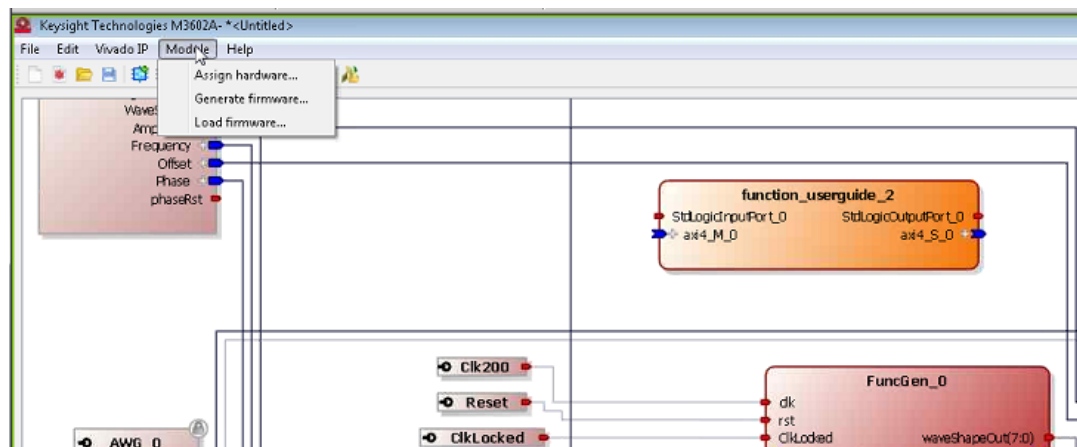
1. 2. 5 Hardware/FW Management

When a project has been created, it can be saved. Once saved, it can be reloaded later to make further changes to it. The associated hardware and its firmware can be managed within a hardware project.

As can be seen in the following image, there are three selections in the Module menu drop-down to manage the project work flow:

Module menu drop-down

- Assign hardware...
- Generate firmware...
- Load firmware...

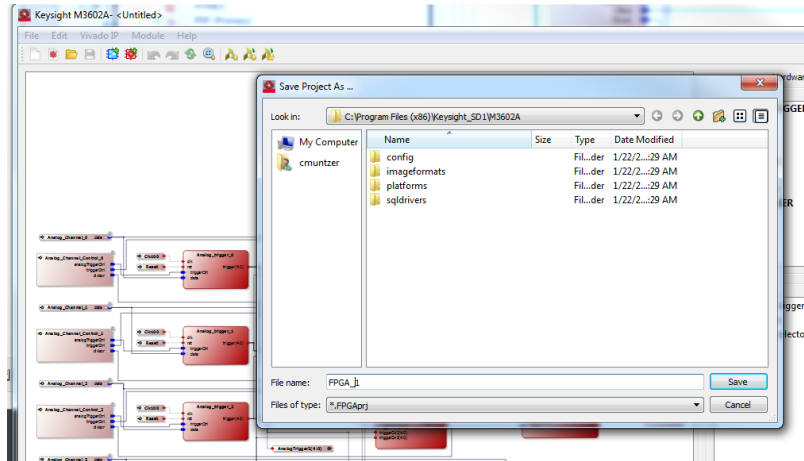


1. 2. 5. 1 Assigning Hardware

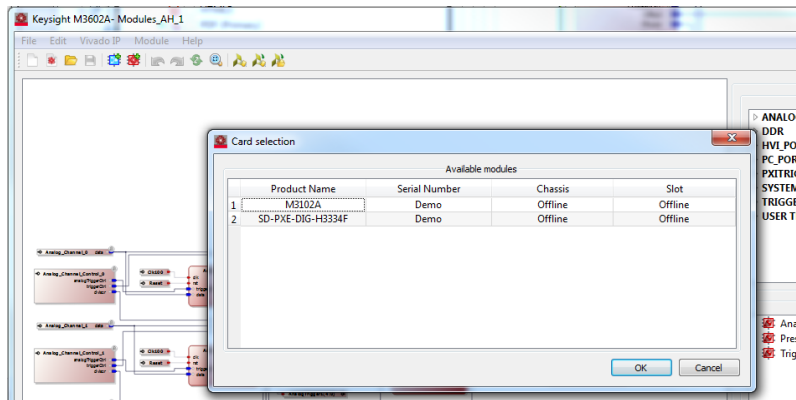
Select **Module > Assign Hardware...**

Before hardware can be assigned, the project needs to be saved.

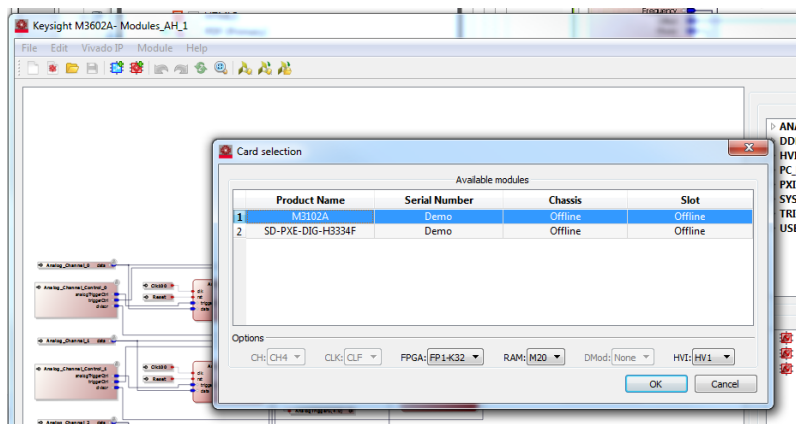
1 Introduction



Click Save



Select the module to assign the software project to.



Click **OK** to assign the selected module to the project.

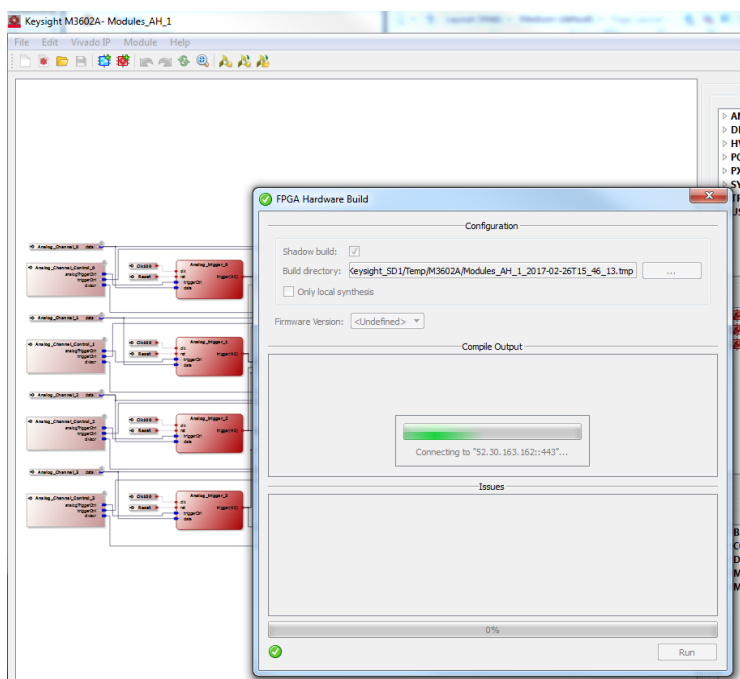
NOTE

In this dialog, the selectable number of cards is minimized in comparison with the options available for the creation of a new hardware project ([section 1.2.4](#)). This is because the associated card to the hardware project can only be changed to a compatible card. In M3602A, the compatibility of cards depends on having the same product name, the same number of channels, and the same clock type. The other options may differ.

1.2.5.2 Generate Firmware

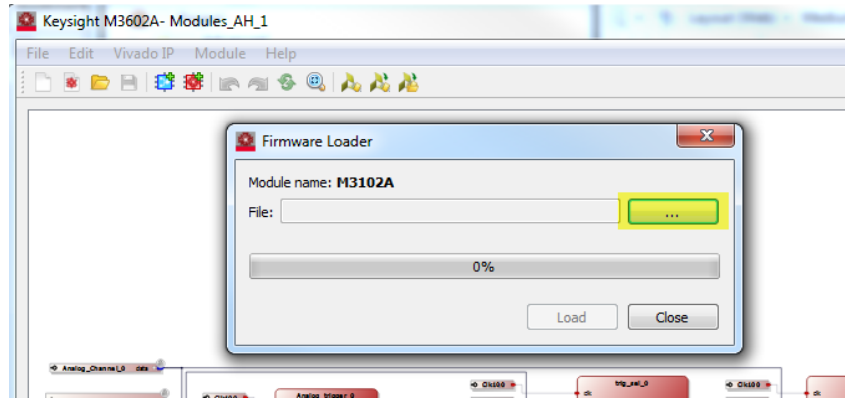
Select **Module > Generate Firmware...**

The M3602A software will connect to the cloud server, and start the firmware generation process. If the generation finishes successfully, the firmware file will be created inside the project folder. For more information and details, please see [Generating FPGA Bitstream \(page 28\)](#)



1.2.5.3 Load Firmware

Select **Module > Load Firmware...**



Select the "." button highlighted above and navigate to the firmware file to be loaded.

Once selected, click the Load button to load the firmware into the hardware module.

NOTE

For this action to be successful, a real card has to be connected to the chassis and assigned to the current hardware project.

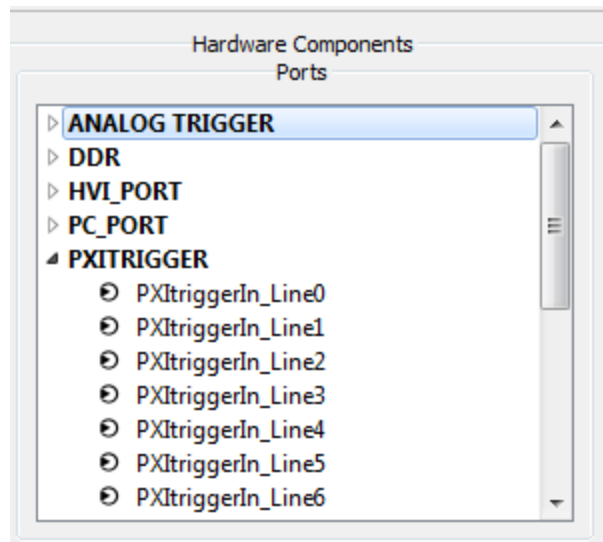
1. 2. 5. 4 Hardware and Library Components

Much of the hardware project will be created by combining components from the three sections displayed on the right side of the GUI. These are divided into Hardware components and Library components.

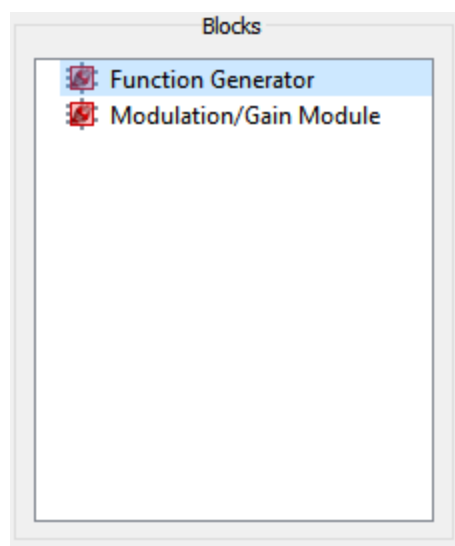
Hardware components

- Hardware ports

These ports enable communication between the hardware project and the rest of the resources of the FPGA. The available type of ports are dependent of the hardware project. In the list below, each port has a different function. For example, the PC_Port selection allows for programming communications to the PC.



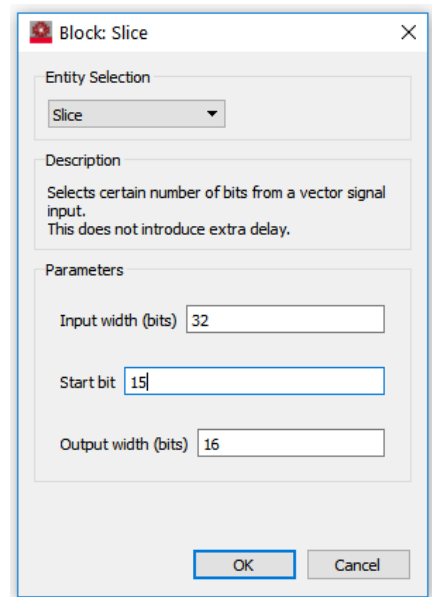
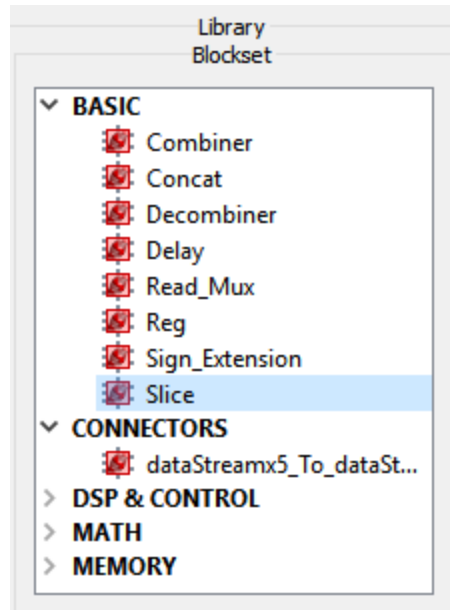
- Hardware blocks
These are IP blocks which are a core part of the default factory setting configuration of the hardware modules. Blocks in this section can be selected by double-clicking on them. A window with a description will pop-up, and the block will be added to the hardware project, ready to be connected to other components in the project. These blocks have a dark red color, and they reflect the functionality of the card associated with the hardware project. As an example, an M320xA or M330xA AWG might include an additional Function Generator to combine multiple tones in the generation path.



Library components

- Blockset IP
This section includes the library of components that can be selected to

complete a hardware project. Blocks are grouped in different generic families (i.e. Basic, Connectors, DSP & Control, etc.). The available blocksets of a family will be displayed after a single-click, and a double-click on a selection will pop up a dialog message with description and configurable parameters, which upon OK selection will be added to the project.



1.2.5.5 Other IP Blocks

In addition to these components, IP blocks can be brought in from the following sources:


- New blocks can be created from scratch and have an orange color. [Creating an FPGA Block Project \(page 17\)](#)
- Blocks imported into a hardware or FPGA Block project as External IP have a blue color.
- Blocks imported from Vivado IP manager tool (using Vivado menu actions of UI) are dark yellow. [Importing Vivado IPs \(page 26\)](#)

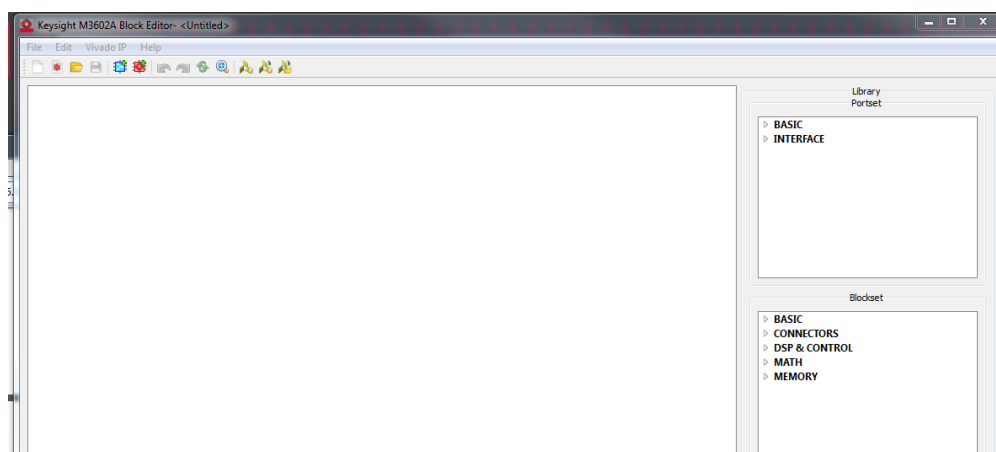
1.2.6 Creating an FPGA Block Project

The hardware project is used for assembling blocks and connections to achieve a hardware FPGA project solution, see [Creating the Main Hardware Project \(page 8\)](#)

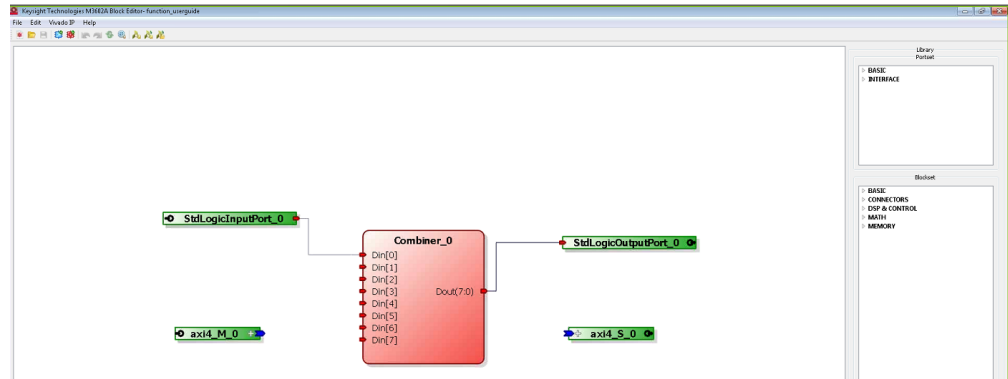
As part of that solution, it will no doubt be necessary to create some new custom blocks to add to the IP blocks in the library. An example of a new custom block would be to add some custom signal delay based on the different logical combination between two existing blocks in the hardware project. This would be done as follows:

To create a new FPGA Block project, do the following:

1. Click on the  icon in the Menu bar.
2. A new FPGA Block project window appears as shown.



3. A block can be created using components from the Library Portset and Blocksets.

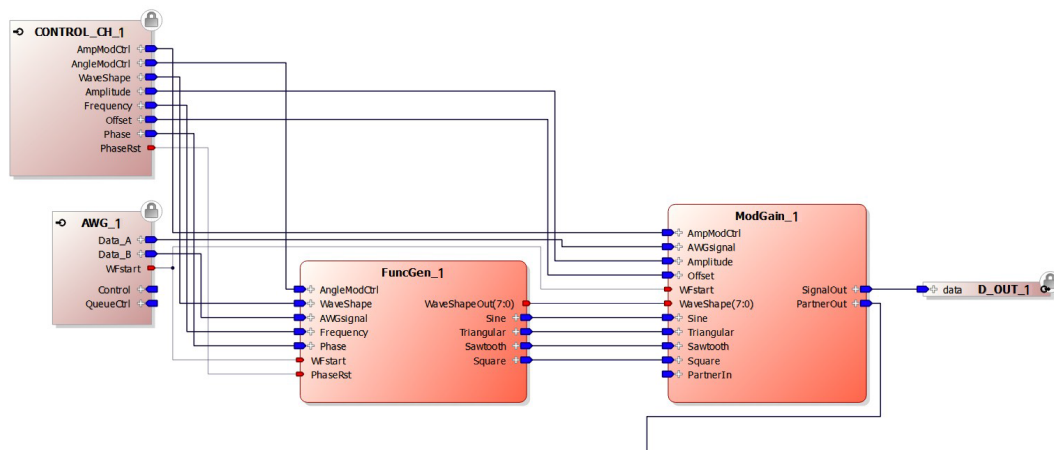


4. Add component blocks as needed to create the project, see:
 - [Adding and Editing Blocks \(page 38\)](#)
 - [Importing User IPs \(page 23\)](#)
 - [Importing Vivado IPs \(page 26\)](#)
5. Connect the signals between the blocks, see [Connecting Ports \(page 35\)](#)
6. Once the FPGA block project has been completed, it can be saved (File name is editable) using **File > Save As**

1.2.7 Ports and Interfaces

After hardware has been set up using the dialog box (see section 1.2.4), the associated hardware project for the card is opened. The contents of the software window will reflect the factory default functionalities of the hardware module.

For example, here is the functionality of an AWG hardware module. To the left side of the image are two input port blocks: **Control_CH_1** and **AWG_1**. To the right side of the image is an output port block: **DOUT_1**.



The icons to the far left of the input port blocks and far right of the output port block show the direction of the port signal. At these input and output ports are images of a locked padlock. This signifies that these ports cannot be removed from project. However, their connections can be edited. They have been designed to communicate with the module hardware.

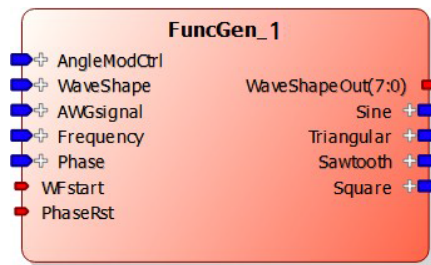


These ports and blocks are represented by the gray (hardware ports) and brown (hardware blocks) icons in [Keysight M3602A Design Environment.htm](#)

In the modules above, different connector types can be identified. There are red connectors and blue connectors. Red connectors represent ports, and blue connectors represent interfaces. An interface is a set of ports.

The interfaces available for the hardware module are listed in the Ports section of the Hardware Components to the right side of the project window. To add them to the design, drag and drop them into the project window. Once in place, they can be configured and connected to other blocks within the project.

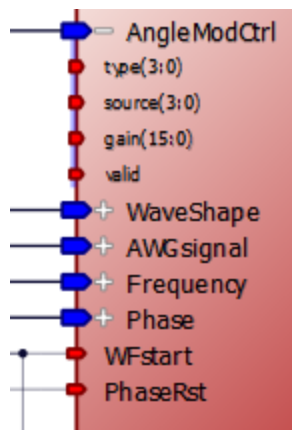
Here is an example of one of the project blocks.



Blocks have ports and interfaces. It can be seen from above that this block has inputs to the left (connectors pointing into the block), and outputs to the right side of the block (connectors out of the block).

This block has three ports (red connectors), and the other connectors are interfaces (blue connectors). The ports can represent one bit of data or a vector of bits. If the port represents a vector of bits, size can be identified next to its name. In the example block above, there are two ports with one bit, "WFstart" and "PhaseRst." The third port, "WaveShapeOut", has a width of 8-bits, where next to its name the "(7:0)" notation is seen. This notation describes its bit ordering, where bit 7 is the most significant bit, and bit 0 is the least significant bit.

When clicking on the "+" sign of an interface, such as "AngleModctrl" in the above image, the internal ports of the interface appear shown below. Notice also that the "+" sign has changed to a "-" sign. Clicking on the "-" sign hides the ports again.



When the interface "AngleModctrl" is connected to another interface, the internal ports are also connected. Although these cannot normally be seen, the individual internal connections between the interface ports are connected. Please, keep in mind that the internal interface ports are connected while their parent interface is connected. Each interface knows how it has to be connected. The internal connections between ports are made automatically, the user only needs to connect the different interfaces and not worry about how it is done internally.

1.2.7.1 Connection Rules

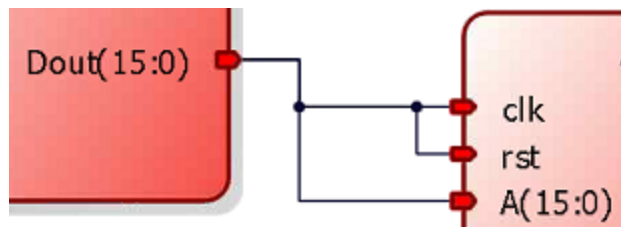
Ports:

There are input ports and output ports.

The input ports can only have one connection to an output port. Here Din(15:0) has one connection



The output ports can be connected to many input ports. Here Dout(15:0) output is connected to three inputs.



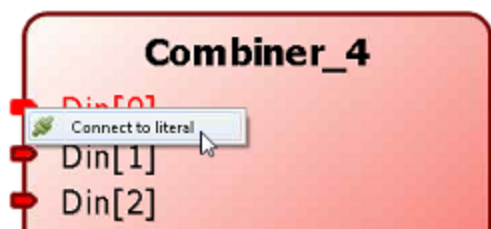
Integers:

An integer represents a source of data, so works as another output port. For this reason, integers can only be connected to input ports.

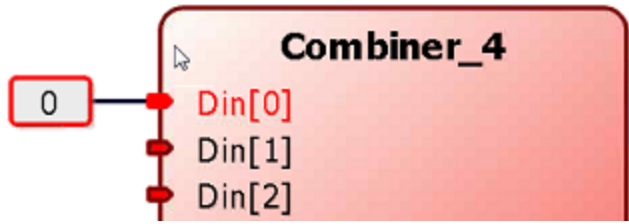
Integer Connection

Adding an integer signal to an input port.

With the mouse, click on the input port, and click on "Connect to literal" as shown below.



An integer box automatically appears. This can be edited to put in binary, hex or decimal to produce fixed value signals.



This technique can be applied to more than one input ports, such as to **Din[1]** and **Din[2]** above.

Interfaces:

Interfaces with the same type can be connected. Therefore, interfaces of like protocols can be put together with a single connection.



Clicking on the "+" sign for either interface the number and naming of the ports will be the same. By connecting the interface to interface, as shown above, all the ports shown are connected, one to one for each interface. This removes the chore of having to connect each interface port as shown below.



1.2.7.2 Interface Roles

Interfaces can have two unique roles, one role is termed a **Master**, and the other role is termed a **Slave**. The Master interface can be connected to a Slave interface with the same type and vice versa. A Master interface controls the connected Slave interface.


1.2.8 Importing User IPs

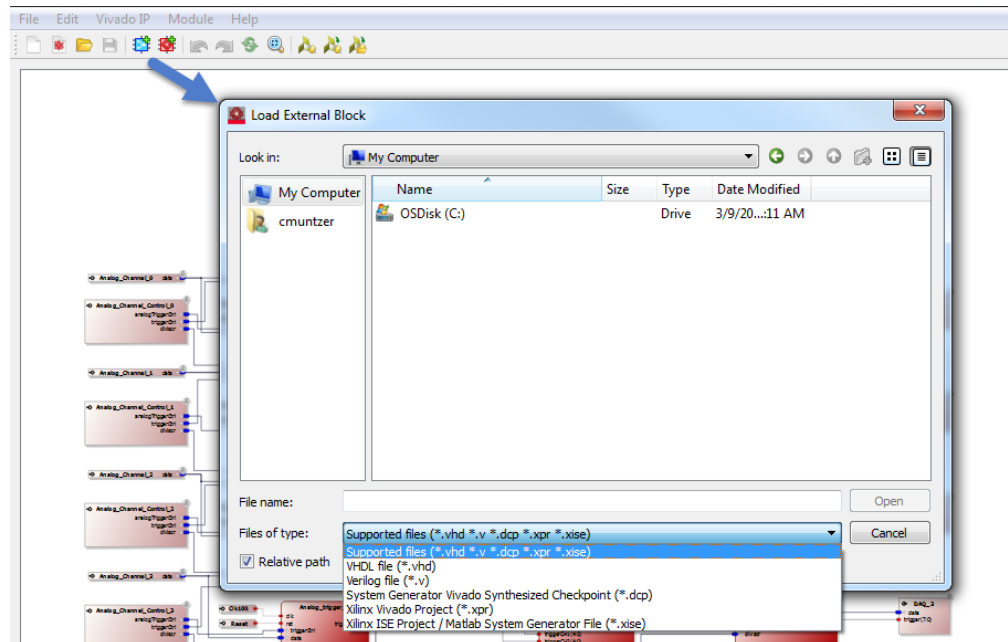
In addition to IPs developed with the Library tools, the M3602A software allows for importing and integration of user custom IPs into a project. These different user IPs have to be developed by the user, using external FPGA tools. The M3602A software is not designed for developing IPs from scratch. However, once the user has finished creating an IP (synthesis and simulate it for example), the IP is ready for being imported to the M3602A software.

The user can import IPs from different source files:

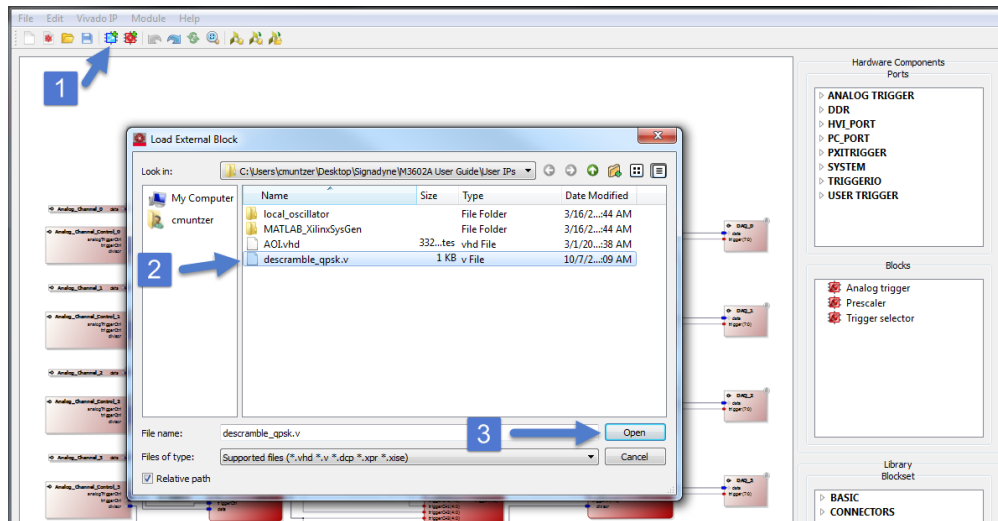
- VHDL source files.
- Verilog source files.
- Xilinx ISE projects.
- Xilinx Vivado projects.
- Matlab Simulink IPs designed using Xilinx System Generator.

To import a user IP, follow this process:

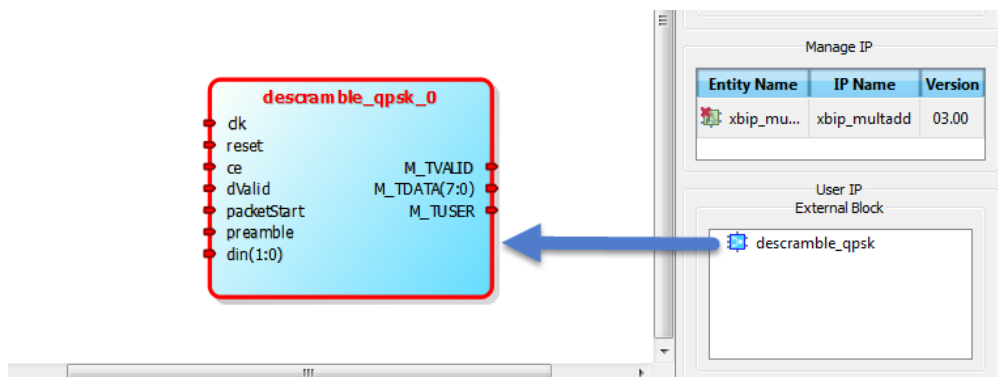
1. Click the  icon in the toolbar, or find it in the **File** drop-down menu. In the image below, notice the file types that are available for importing. Notice also, the **Relative path** check box is checked by default.



2. Following the numbering in the image below, select the User IP icon, navigate to the file to be imported, and select it for importing into the project. Click **OK** to open and import the file.



3. The IP will be inserted into the project where it can be connected to other blocks. The block will have a blue background, see the blue block in the following image.



4. The file name will appear in the User IP External Block region for reuse as shown above. Note, these files can be removed by right-clicking the file name and choosing remove.

NOTE If the Relative Path check box is deselected, the path will become an absolute path.

NOTE If the User IP file is moved, an "X" will appear at the top of the block indicating the file cannot be found. Once the file is moved back, or the path changed, right-clicking the block will allow for reloading the IP and removing the "X" on the block.

NOTE

If the underlying code for the code is changed, a "!" can appear to signify an alert condition. Once the code is corrected, the block can be reloaded to remove the "!" on the block.


1. 2. 9 Importing Vivado IPs

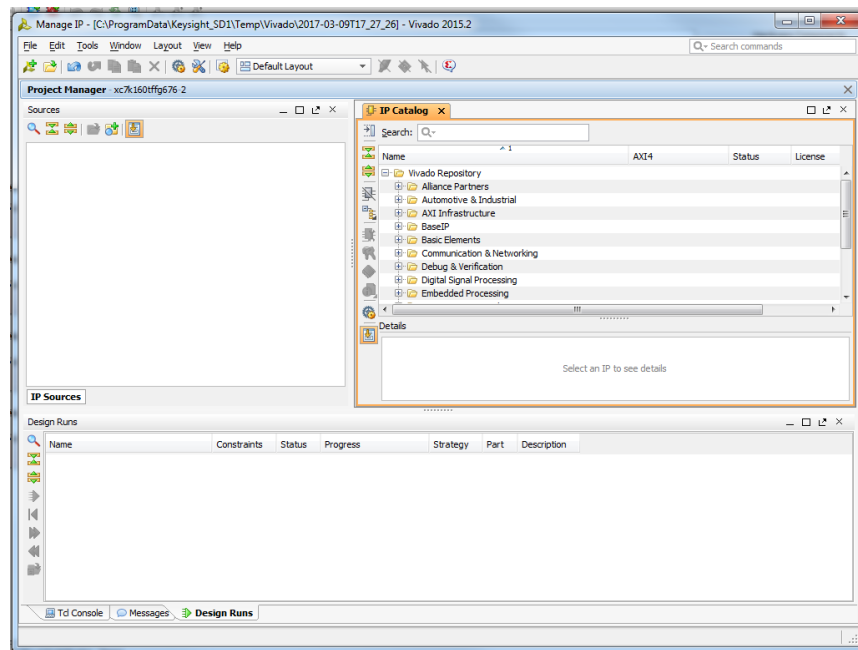
NOTE

For first time users of M3602A software, refer to [M3602A Initial Setup \(page 65\)](#)

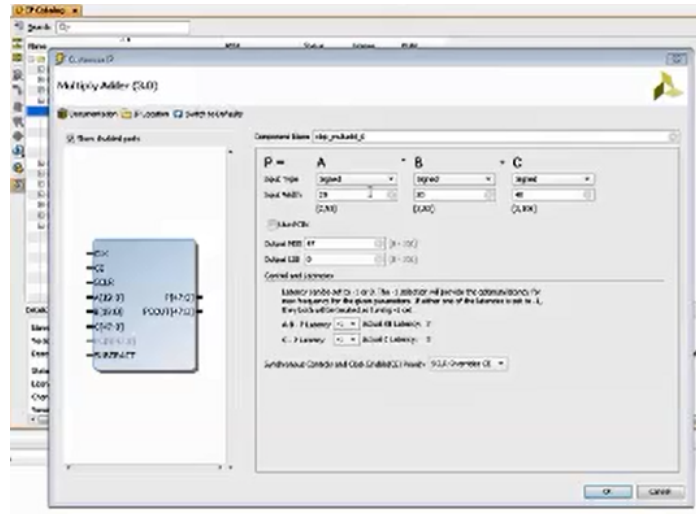
The M3602A software also allows the user to import Vivado IPs from the Xilinx Vivado IP Catalog. The available Vivado IPs can be imported from the catalog and integrated to project.

To import a Vivado IP, do the following:

1. Open the M3602A software.
2. Open the existing project that needs to be imported from Vivado IP and make sure it is named and saved.
3. Click on the **Launch Vivado IP Tool** icon .
4. Search the desired IP inside "IP Catalog" and select it for it to be imported.



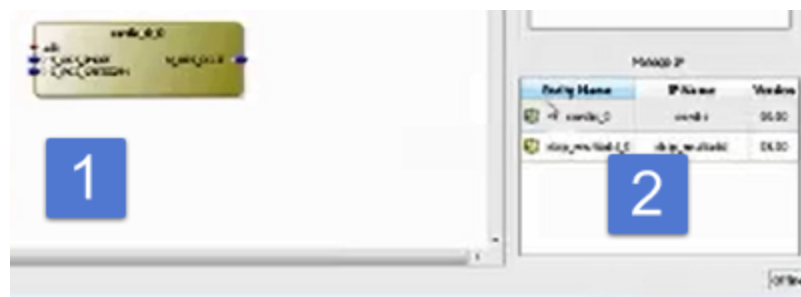
5. Configure the IP properties according to its purpose and then click **OK**. After importing the IP, properties can always be reconfigured again



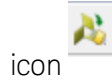
6. The Vivado IP will appear in the bottom right of the project window as shown.



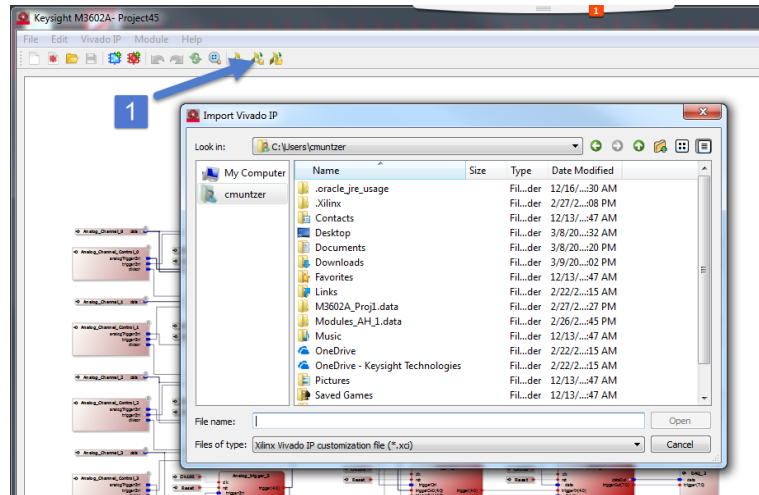
7. The Vivado IP can be selected and appears in the project as shown.



8. Vivado IPs can be shared between different projects by selecting the following




icon



9. The Vivado IP will then be available for reuse in other projects.



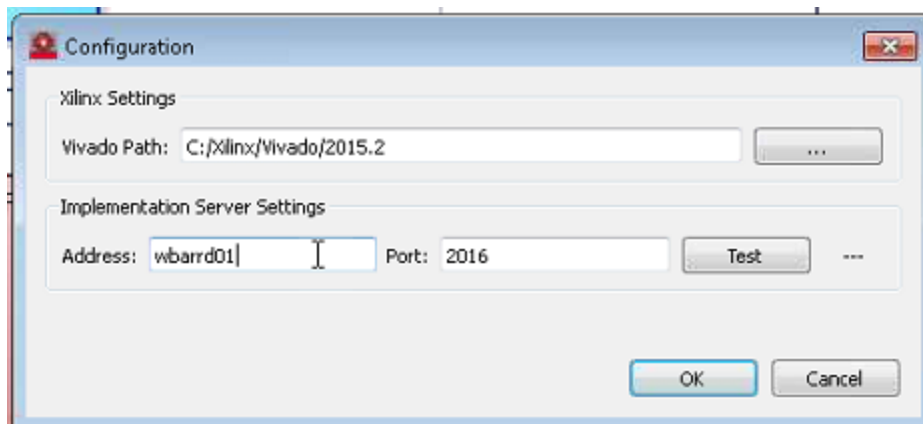
10. By clicking the third Vivado icon, , more than one Vivado IPs can be imported from the same folder.

1. 2. 10 Generating FPGA Bitstream

This section shows how to generate the bitstream of the M3602A FPGA hardware projects.

1. Open the M3602A FPGA software.
2. Add IPs and connect them up within the hardware project.
3. Save the project.

4. Check the configuration connections to the server. **File > Settings**. The Configuration dialog box appears:



Configure the server connection settings:

- **Vivado Path** to the Vivado 2015.2 software
- **Implementation Server Settings** (check these settings with information received on purchasing the M3602A FPGA software):
 - **Address**
 - **Port**

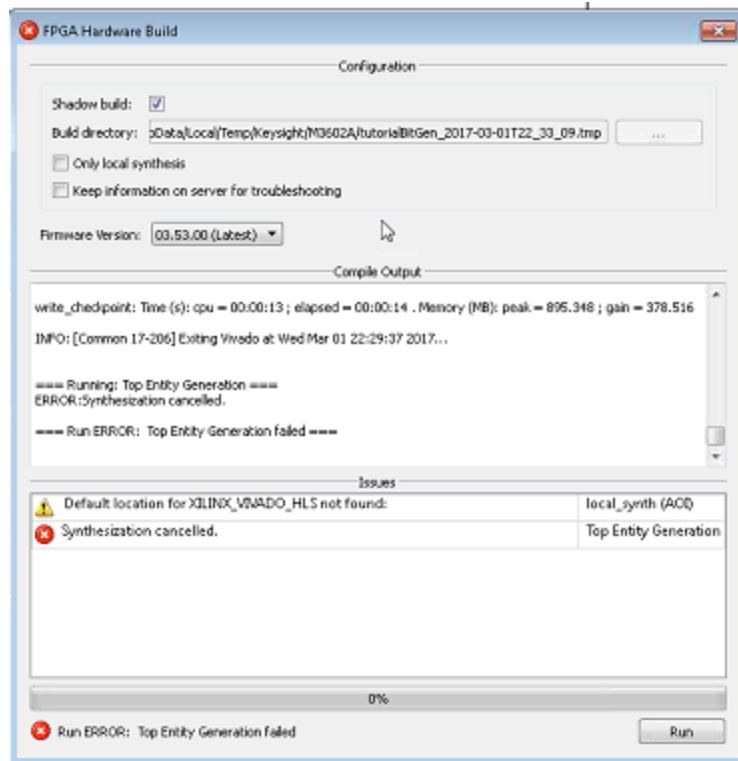
5. Click the **Test** button to confirm that the connection is OK.

6. Launch the Generate Firmware process by clicking the  icon.

NOTE

The FPGA compilation is a two step process. The compilation is carried out at first on the local PC, after which the compilation is continued on the cloud server.

7. The FPGA Hardware Build dialog box appears.



The dialog box has several settings:

- **Shadow build:** Check this if wishing to create a shadow build of the hardware project. Shadow build means the software will use a build folder inside the temporal path of the system. However, it can be specified to a different folder by deselecting this option. By default, it will be created within the hardware project folder.
- **Build Directory:** Make sure the path to the folder shown is where it is wished to have the firmware build file deposited. It can be edited by deselecting the shadow build option. The build directory path is shown in red text to alert that the target folder does not exist and that it will be created when the compilation runs.
- **Only Local Synthesis:** Check this box only if wishing to launch the local synthesis step. It means the software will only synthesize the hardware project on the current computer, and will not try to connect to the cloud compilation server for compiling the whole project. So, with this option, it will never be receiving a bitstream reset file. However, this option could be useful if it is only wanted to check if the project is synthesized successfully. A correct local synthesis result is necessary for continuing the compilation using the cloud.
- **Keep information on server for troubleshooting:** Check this if wishing to have a copy of the compilation results saved on the server. After compilation, the build files on the server are erased. Checking this retains those files for troubleshooting.

- **Firmware Version.** This shows the versions of firmware available. It is important to have the same firmware version as the module that the FPGA bitstream was created for.

8. After these settings have been checked and applied, click the **Run** button.

During the compilation process, messages are seen in the **Compile Output** window of the **FPGA hardware Build** dialog box. Messages can also be seen in the Issues window of the **FPGA hardware Build** dialog box. At the bottom of the dialog box is the compilation progress bar. When this gets to 20%, the compilation is taken over by the server. As the compilation proceeds, some common errors may be observed in the **Issues** window. See [Generating FPGA bitstream.htm](#).

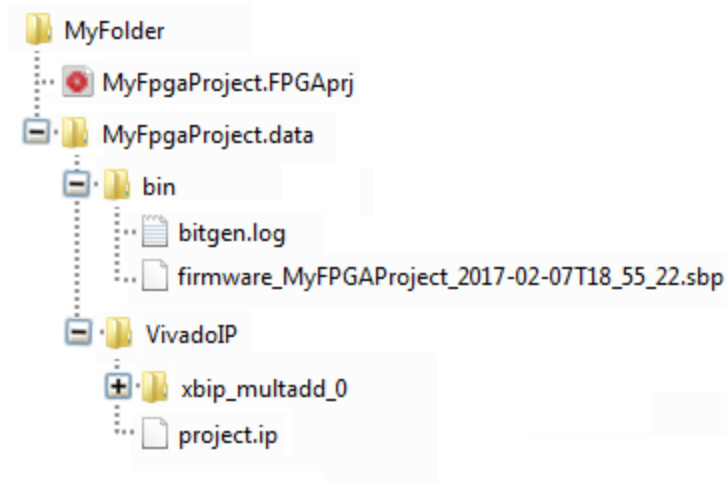
9. If the compilation is successful, the bitstream files with extension “sbp” will be deposited in the “bin” folder inside the data project folder. See [1.2.11 Project directory](#) structure for more details.

1. 2. 10. 1 Build Errors and Warnings

- [3] **"Not supported product"**: It is possible the server does not support the product under compilation. This error should not happen when using the software available for the products.
- [4] **"Project incompatible: "**: The hardware project version is incompatible with the current firmware version of the module. It will be needed to create a new hardware project for implementing hardware projects with this module.
- [5] **"Firmware version of card is not compatible"**: It is needed to update the firmware of the card being used.
- [6] **"The implementation server is temporarily unavailable. Please try it later. We apologize for the inconveniences"**: This message may be seen if the server has many implementations in progress.
- [7] **"You have reached the maximum number of simultaneous implementations"**: For example, if two compilations were run at the same time, this error would be seen as only one simulation implementation is allowed.

1. 2. 11 Project Directory Structure

When a new project is created, a new folder named “<project_name>.data” is created near of the project file for managing project information such as imported IPs or compilation result files. The different folders and files inside of this data folder are created depending on software needs. This folder contents should not be changed by user. For example, if there is a project named “MyFpgaProject”, the following disk structure can be found:



Directory Name, File Name or File Type	Description
MyFPGAProject.FPGAproj	M3602A hardware project file
MyFPGAProject.data	Data folder of “MyFPGAProject” project
bin	Folder for saving the compilation result files.
bin/bitgen.log	Output compilation log file.
bin/*.sbp	Bitstream result files
VivadoIP	Folder for saving the imported Vivado IPs to project

1.3 Editing Actions

1.3.1 Basic Controls

1.3.1.1 Zooming in and out

Using the mouse button

To make the blocks larger: Control plus Left-click on a block as many times as needed to get the image size required.

To make the blocks smaller: Control plus Right-click on a block as many times as needed to get the image size required.

Using the mouse wheel

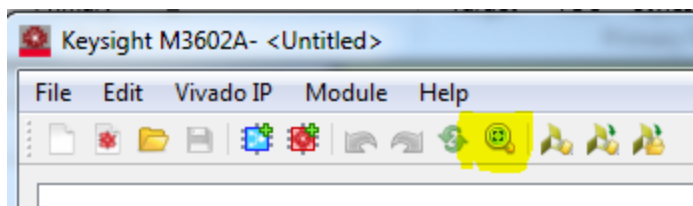
To make the blocks larger or smaller: Control plus move the mouse wheel one direction or the other to get the image size required.

1.3.1.2 Span

Use the Alt+left mouse button to move the project view with the mouse cursor.

1.3.1.3 Fit in Window

Use the highlighted icon below to fit the project within the window if it spills outside the window. This acts as an auto-zoom-out feature to bring all project elements within the window for viewing.



1.3.1.4 Multiple Selections

To make multiple selections, with the left mouse button held down, drag a rectangle around the multiple items wished to be selected.

Alternatively, the Shift key can be pressed and click on the items for multiple selections as well.

1.3.1.5 Copy Action

To copy a block or element, select the item with the mouse, and use the Ctrl key plus the C key to copy it. Once the item is copied, it can be dragged to the desired location.

An alternative way to copy an element is by clicking Ctrl key, click on the desired item and move the mouse. Then, the newly copied item can be seen below the mouse cursor, and the item can be drag and drop to the desired location.

Another way to copy items is to press the Right-click mouse button on the item and select the “Copy” option in the options menu.

1.3.1.6 Move Items

To move an item, select the item with the left-button on the mouse. Holding the left-button down, drag the selected item to the desired location.

1.3.1.7 Undo/Redo Action

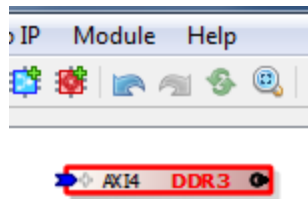
From the keyboard:

To **Undo** an action, select the Ctrl key + Z key

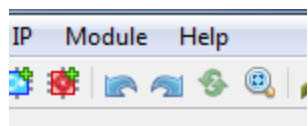
To **Redo** an action, select the Ctrl key + Y key

From the GUI

Here a port (AXI4) has been added to the project causing the **Undo** menu icon to be highlighted.



Clicking on the highlighted **Undo** icon removes the port and highlights the **Redo** icon as shown here.

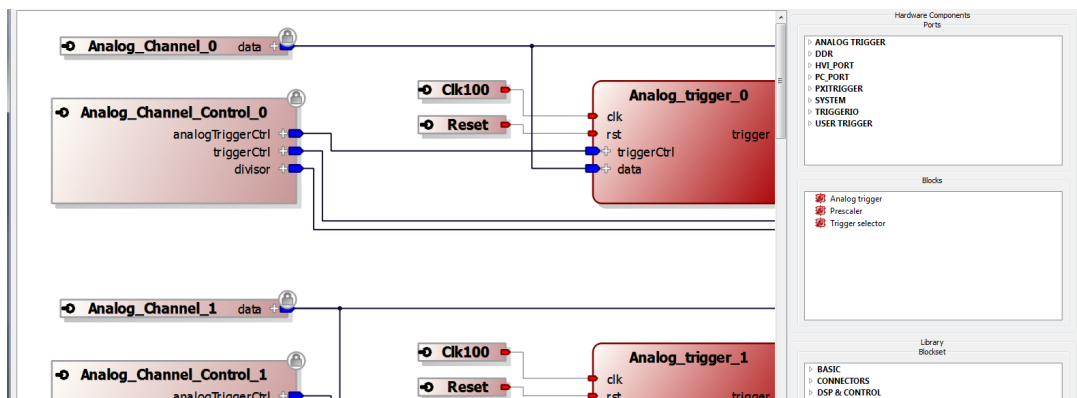


Now both icons are highlighted, use these **Undo**, **Redo** icons to step back and forward through changes to the project.

1.3.2 Connecting Ports

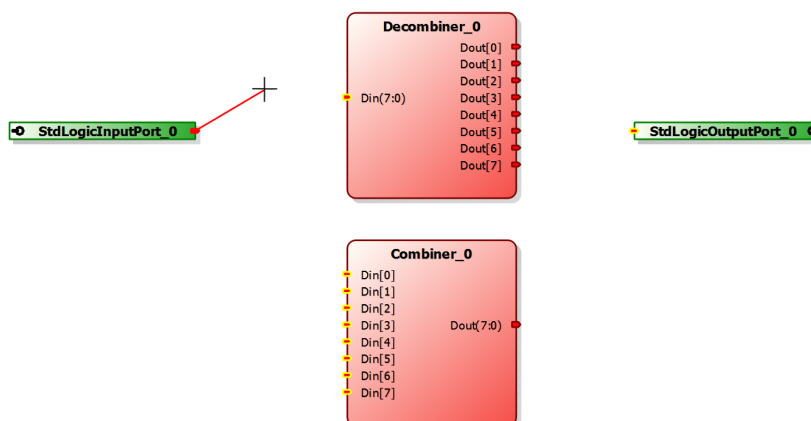
Connections can be created using the mouse control.

In the image below, connections are made by clicking on an output port and then dragging the line from it to a suitable input.

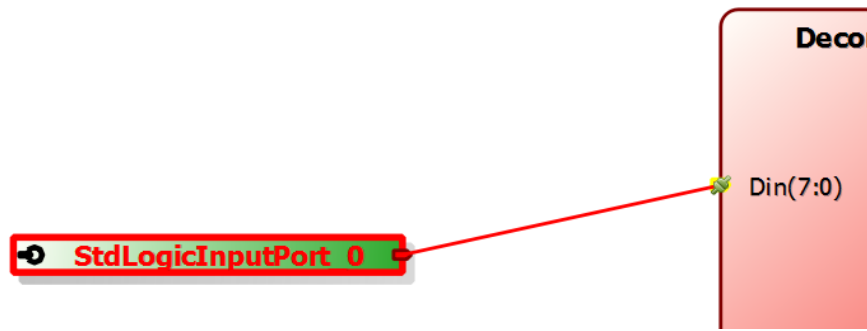


Connections can be created according to connection rules, which are explained in [Connecting Ports \(page 35\)](#)

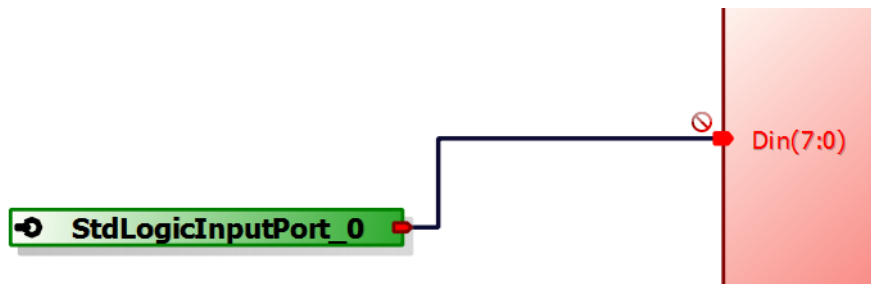
If a connection can be made from a connector, a new line appears from this connector to mouse and the mouse cursor changes to the axis icon as shown below. Furthermore, the possible target connectors are highlighted in yellow color for showing the different connection possibilities. See the input ports on the lower block "Combiner_0" shown below.



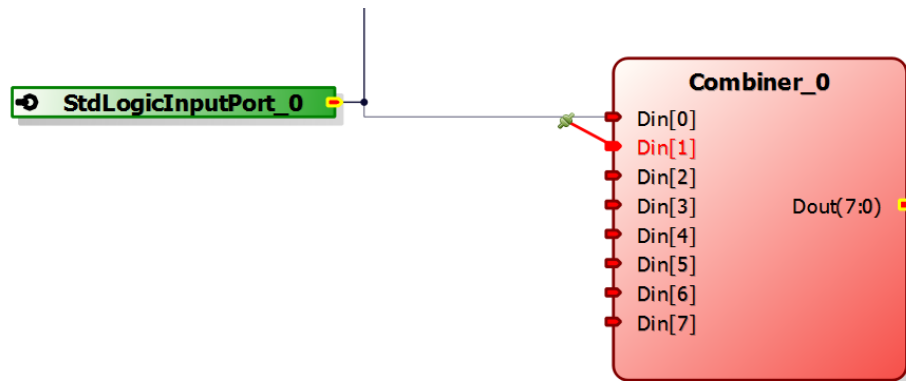
For finishing the connection, the end of the connection line is dragged by the mouse to a compatible target connector. In this case, the mouse icon changes to green connection icon can be seen.



When the mouse button is released, the new connection is created. Otherwise, attempting to create a connection from a connector which does not accept new connections, the forbidden icon as mouse cursor appears as follows.



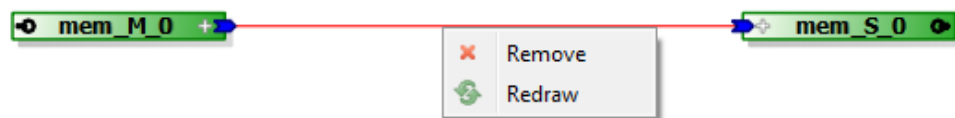
When making multiple connections, it is not needed to connect directly for example from a common source to multiple input ports. The ports can be connected together at the target. For example, in the following image, **Din[0]** is connected to **StdLogicInputPort_0**. If it is wanted to have both **Din[0]** and **Din[1]** connected to **StdLogicInputPort_0**, **Din[1]** can be joined to **Din[0]** locally by selecting the **Din[1]** port with the mouse and dragging its connection line to the existing line already connected to **Din[0]**. Notice the green acceptable connection icon below. Once the left-hand button of the mouse is released, the connection is made locally between **Din[0]** and **Din[1]**.



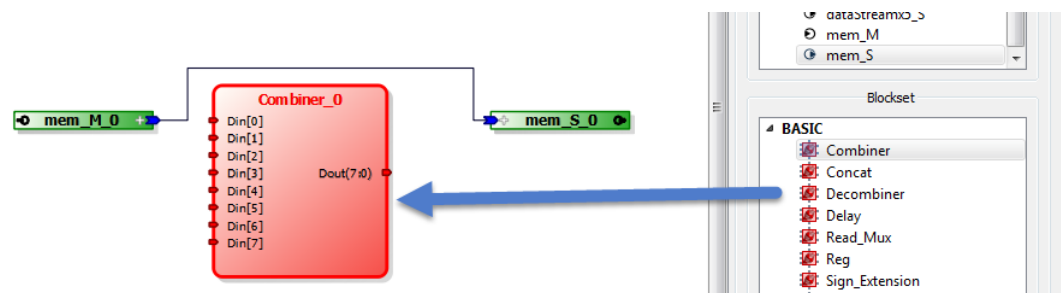
Remove and Redraw operations

For example, select the **mem_M** and **mem_S** ports and drag them individually into the FPGA Block project. Connect the output of the master denoted with an "M" to the input of the slave, denoted with an "S."

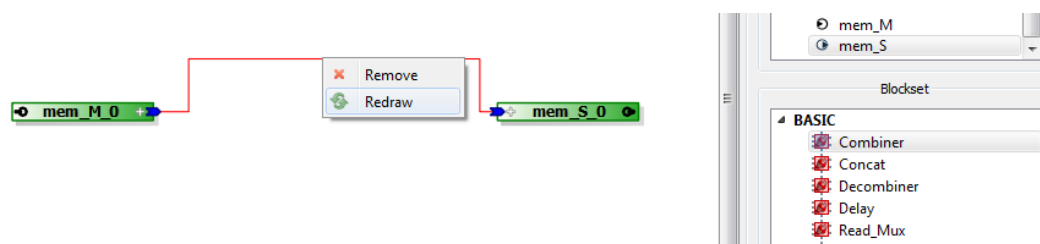
Right-click the line connecting the two ports to see two options: **Remove** and **Redraw**. Remove will delete the connecting line.



By way of an example, add a block between the two ports. Notice the line connecting the ports is now no longer straight.

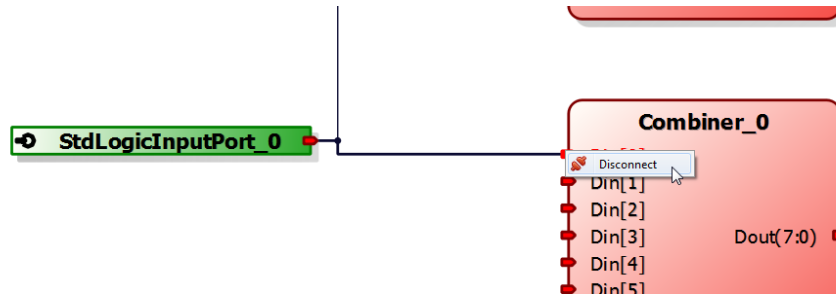


Delete the block that was just added and notice that the connecting line stays unchanged. Right-click the line and select **Redraw**. The line will become straight again.



Disconnecting a connection

Once a connection is created, the connection can be disconnected by right-clicking with the mouse button on the connector, which brings up the **Disconnect** option.



1.3.3 Adding and Editing Blocks

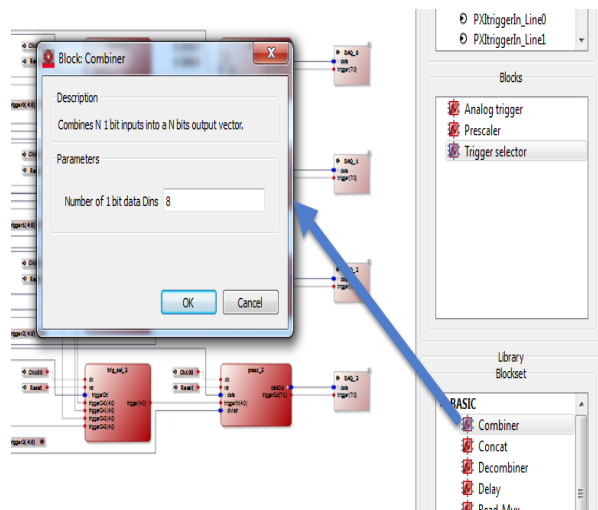
1.3.3.1 M3602A Different IPs

- Built-in blocks:
 - Hardware blocks and ports (hardware project)
 - Library ports (FPGA block project)
 - Library blocks
- User blocks
 - External Blocks
 - Vivado blocks
- FPGA block

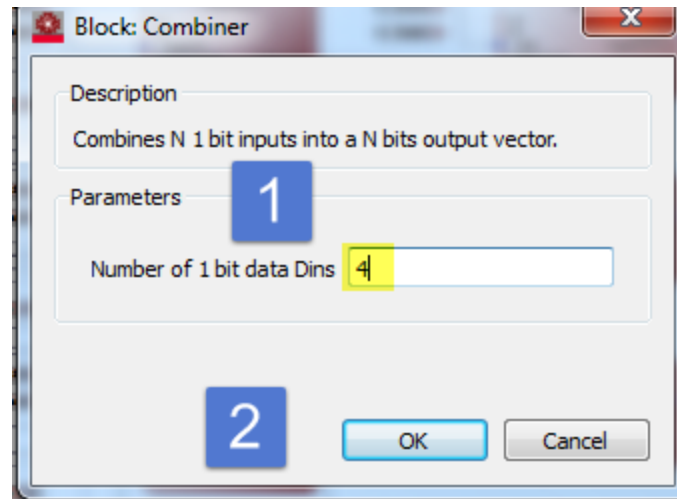
1.3.3.2 Built-In Blocks

When a hardware project is opened, both the hardware and library can be seen, blocks and ports that are available for the particular hardware module. These blocks and ports can be selected, dragged into the project, configured, and connected to other blocks in the project.

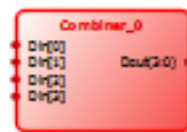
1. For example:



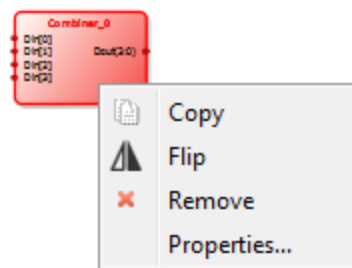
- The selected block can now be (1) configured and (2) saved.



- Notice there are now only four data lines as opposed to the previous eight data lines that came as the default value.



- When a block is selected and right-click the block, the following options are available:



- **Copy** allows making a copy of this block.
- **Remove** deletes the block from the project.
- **Properties...** provides the configuration dialog box shown above.

- Selecting "Flip" will drive the following action:



1.3.3.3 Library Ports (FPGA Block Project)

To import Hardware Ports, Blocks, and Library Blocks, see [M3602A Basics.htm](#)

1.3.3.4 Adding User Blocks

External Blocks


To import Hardware Ports, Blocks, and Library Blocks, see [Importing User IPs \(page 23\)](#)

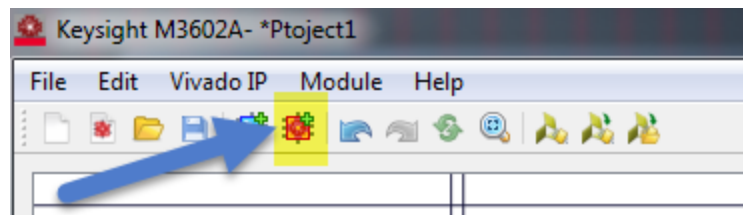
Vivado Blocks

To import Hardware Ports, Blocks, and Library Blocks, see [Importing Vivado IPs \(page 26\)](#)

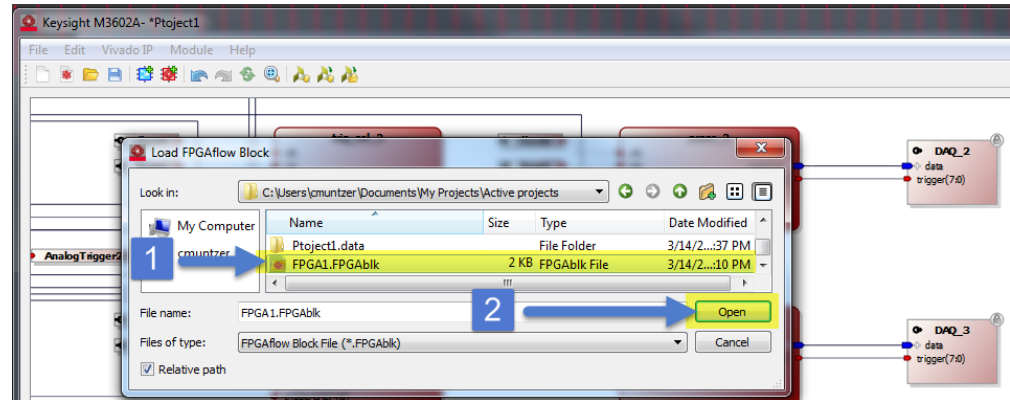
1.3.3.5 Adding an FPGA Block

To add an FPGA Block, do the following:

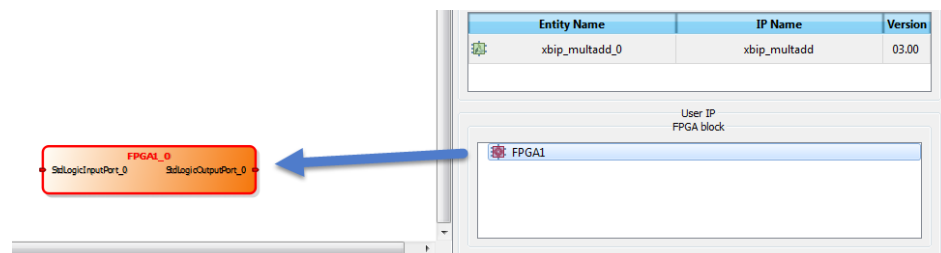
1. Click on the **Adding an FPGA block**  icon



2. Navigate to, select and open the desired FPGA block.



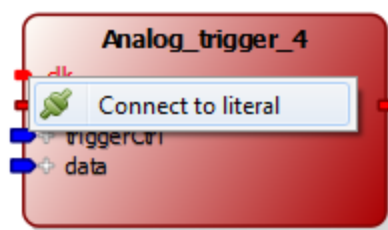
3. Notice the selected block appears in the **User IP FPGA block** area of the project, where it can be selected and dragged into the project window. Once there, it can be connected to other project ports and blocks.



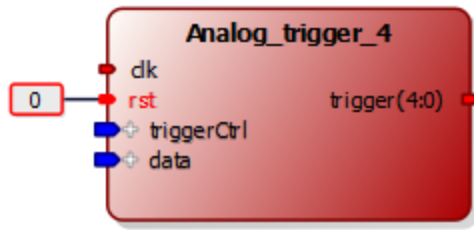
1.3.4 Adding and Editing Integers

To add and edit an integer, do the following:

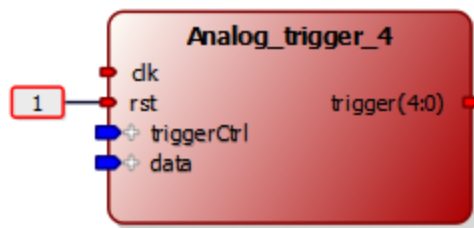
Make sure there is no input already on the input port intended to have an integer connected to it, and Right-click the port.



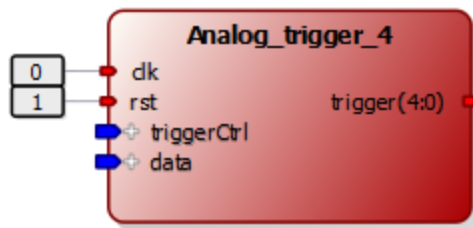
Accept the **Connect to literal** message. A text box with a decimal value of 0 is connected to the port. See below for **connecting decimals, hexadecimals and binaries**.



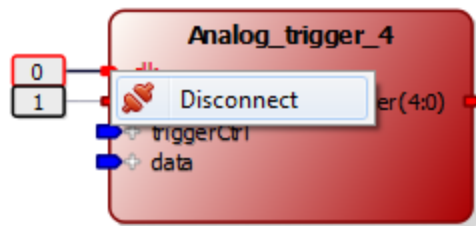
If need be, edit the integer value. In this case from 0 to 1.



Other integers can be added to ports in a similar manner.



To disconnect an integer, Right-click the input port to be disconnected, and accept the Disconnect message.



The integer is removed.



1.3.4.1 Connecting Decimals, Hexadecimals, and Binaries

When the literal is added to the connector, as shown above, the text box appearing can be edited to allow a connecting decimal, hexadecimal, or binary value to be applied to the connector.

Decimal: The literal by default is the decimal. In the above example, only a **0** or **1** is shown. However, a larger decimal could be entered, for example, **10**. A maximum of ten decimal numbers can be used in sequence.

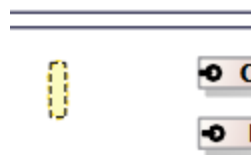
Hexadecimal: A hexadecimal number between the range of 0 - F can be entered, followed by an **h**; for example, **Ah**. A maximum of seven hexadecimal numbers can be used in sequence.

Binary: Binary numbers can be added followed by a **b**, for example, **1010b**. A maximum of thirty binary numbers can be used in sequence.

1.3.5 Adding and Editing Comments

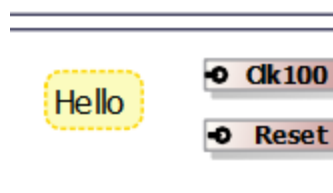
To add a comment, do the following:

1. Position the cursor within the project where the comment is wished to made.
2. Double-click the left button on the mouse and a text box appears with focus and a yellow background.



to be made

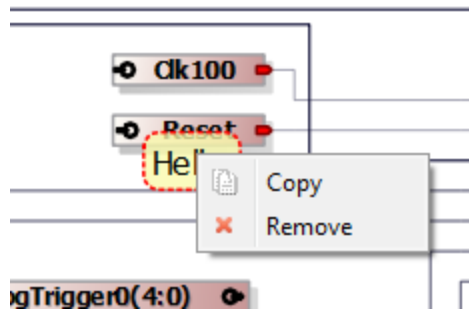
3. Add text to the comment text box.



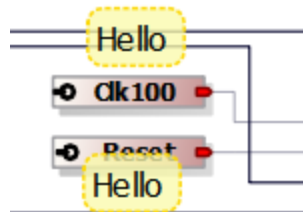
- 4. The comment can be moved by dragging it with the mouse. Notice the comment is in the foreground and sits above the project elements.



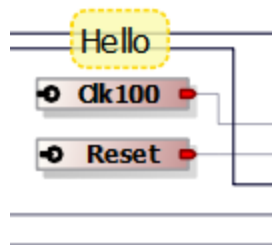
- 5. By Right-clicking the comment, a choice of copying or removing the comment is provided.



- 6. Choosing **Copy**, a duplicate comment is created close to the original.



- 7. Choosing **Remove**, the comment can be deleted.



2 Software Programming Guide: PC-FPGA Interaction

Thanks to the Keysight Programming Libraries, FPGAs created with Keysight M3602A integrate seamlessly with the user application, commonly referred in Keysight documentation as VI (Virtual Instrument, Section [SW Programming](#)).

Within M3602A, an FPGA Project can be compiled into an FPGA binary file (File ⇒ Generate FPGA...). With this file, the FPGA can be executed and controlled from the user application, e.g. the FPGA can be launched, paused, stopped, etc. VIs and FPGAs can also exchange data and signals.

This Section describes the programming functions designed to control FPGAs.

2.1 Programming Functions

2.1.1 SW Programming Overview Programming Libraries

Keysight provides highly optimized programming libraries to operate the Keysight M3602A FPGA Design Environment which is supported by the following modules: M3100A, M3102A, M3201A, M3202A, M3300A and M3302A with -FPGA option enabled.

2.1.2 SD_Module Configurations

The following configurations are used as parameters of the SD_Module functions (in the next section).

2.1.2.1 Addressing Mode

The addressing mode configuration is used for selecting how the provided address is used in an Input/Output transaction.

Option	Description	Programming Definitions Name	Value
Auto Increment	Initial address is incremented after each access	ADDRESSING_AUTOINCREMENT	0
Fixed	Initial address is used for the whole access	ADDRESSING_FIXED	1

In object-oriented languages, this configuration is implemented inside the SD_AddressMode namespace.

2.1.2.2 Accessing Mode

The accessing mode configuration is used for using DMA technology or not in memory Input/Output operations.

Option	Description	Programming Definitions Name	Value
Non DMA	Memory access are split in several one double word accesses	NONDMA	0
DMA	memory access is done with one DMA transaction	DMA	1

In object-oriented languages, this configuration is implemented inside the `SD_AccessMode` namespace.

2. 1. 2. 3 Reset Mode

The reset mode configuration is used for selecting the reset signal mode type of FPGA.

Option	Description	Programming Definitions Name	Value
Low	Low active reset	RESET_LOW	0
High	High active reset	RESET_HIGH	1
Pulse	Pulse reset	RESET_PULSE	2

In object-oriented languages, this configuration is implemented inside the `SD_ResetMode` namespace.

2. 1. 3 SD_Module Functions

2. 1. 3. 1 FPGAwritePCport

This function writes data at the PCport FPGA Block.

Properties

Name	Description
Inputs	
moduleID	(Non-object-oriented languages only) Module identifier, returned by function open in Section 2.4.2.1.
nPCport	PCport number
data	Data buffer to write through PC port to FPGA
dataSize	Number of 32-bit words to write (maximum is 128 words)
address	Address that will appear on the PCport interface
addressMode	Write addressing mode desired Section 2.1.2.1
accessMode	Write accessing mode desired Section 2.1.2.2
errorIn	(LabVIEW only) If it contains an error, the function will not be executed, and errorIn will be passed to errorOut

Name	Description
Outputs	
moduleIDout	(LabVIEW only) A copy of moduleID
errorOut	Error Codes

C

```
int SD_Module_FPGAwritePCport(int moduleID, int nPCport, int* data, int dataSize, int address,
int addressMode, int accessMode);
```

C++

```
int SD_Module::FPGAwritePCport(int nPCport, int* data, int dataSize, int address, SD_
AddressingMode addressMode, SD_AccessMode accessMode);
```

Visual Studio .NET, MATLAB

```
int SD_Module::FPGAwritePCport(int nPCport, int [] data, int address, SD_AddressingMode
addressMode, SD_AccessMode accessMode);
```

Python

```
int SD_Module::FPGAwritePCport(int nPCport, int [] data, int address, int addressMode, int
accessMode);
```

LabVIEW

FPGAwritePCport.vi

2.1.3.2 FPGAreadPCport

This function reads data at the PCport FPGA Block.

Properties

Name	Description
Inputs	
moduleID	(Non-object-oriented languages only) Module identifier, returned by function open in Section 2.4.2.1.
nPCport	PCport number
dataSize	Number of 32-bit words to read (maximum is 128 words)
address	Address that will appear on the PCport interface
addressMode	Read addressing mode desired Section 2.1.2.1

Name	Description
accessMode	Read accessing mode desired Section 2.1.2.2
errorIn	(LabVIEW only) If it contains an error, the function will not be executed, and errorIn will be passed to errorOut
Outputs	
data	Rx data buffer
moduleIDout	(LabVIEW only) A copy of moduleID
errorOut	Error Codes

C

```
int SD_Module_FPGAreadPCport(int moduleID, int nPCport, int* data, int dataSize, int address, int addressMode, int accessMode);
```

C++

```
int SD_Module::FPGAreadPCport(int nPCport, int* data, int dataSize, int address, SD_AddressingMode addressMode, SD_AccessMode accessMode);
```

Visual Studio .NET, MATLAB

```
int SD_Module::FPGAreadPCport(int nPCport, int address, int[] data, SD_AddressingMode addressMode, SD_AccessMode accessMode);
```

Python

```
{int[], int} SD_Module::FPGAreadPCport(int nPCport, int dataSize, int address, int addressMode, int accessMode);
```

*Returned data array is a NumPy array.

LabVIEW

FPGAreadPCport.vi

2.1.3.3 FPGAload

This function loads a bitstream file generated using M3602A software to FPGA.

Properties

Name	Description
Inputs	
moduleID	(Non-object-oriented languages only) Module identifier, returned by function open in Section 2.4.2.1.
fileName	File to load
errorIn	(LabVIEW only) If it contains an error, the function will not be executed, and errorIn will be passed to errorOut
Outputs	
moduleIDout	(LabVIEW only) A copy of moduleID
errorOut	Error Codes

C

```
int SD_Module_FPGAload(int moduleID, const char *fileName);
```

C++

```
int FPGAload(const char *fileName);
```

Visual Studio .NET, MATLAB

```
int FPGAload(string fileName);
```

Python

```
int FPGAload(string fileName);
```

LabVIEW

```
load.vi
```

2.1.3.4 FPGAReset

This function sends a reset signal to FPGA.

Properties

Name	Description
Inputs	
moduleID	(Non-object-oriented languages only) Module identifier, returned by function open in Section 2.4.2.1.
mode	Reset mode desired Section 2.1.2.3

Name	Description
errorIn	(LabVIEW only) If it contains an error, the function will not be executed, and errorIn will be passed to errorOut
Outputs	
moduleIDout	(LabVIEW only) A copy of moduleID
errorOut	Error Codes

C

```
int SD_Module_FPGAreset(int moduleID, int mode);
```

C++

```
int FPGAreset(SD_ResetMode::SD_ResetMode mode);
```

Visual Studio .NET, MATLAB

```
int FPGAreset(SD_ResetMode mode);
```

Python

```
int FPGAreset(int mode);
```

LabVIEW

```
reset.vi
```

2.1.4 Error Codes

error codes	
#define SD_STATUS_OK	0
#define SD_ERROR_NONE	0
#define SD_STATUS_DEMO	1
#define SD_ERROR_OFFSET	-8000

2.1.4.1 Open and Close Errors

Open and Close Errors	
//Open and Close Errors	
#define SD_ERROR_OPENING_MODULE	-8000
#define SD_ERROR_OPENING_MODULE_STRING	"Keysight Error: Opening module"

Open and Close Errors

#define SD_ERROR_CLOSING_MODULE	-8001
#define SD_ERROR_CLOSING_MODULE_STRING	"Keysight Error: Closing module"
#define SD_ERROR_OPENING_HVI	-8002
#define SD_ERROR_OPENING_HVI_STRING	"Keysight Error: Opening HVI"
#define SD_ERROR_CLOSING_HVI	-8003
#define SD_ERROR_CLOSING_HVI_STRING	"Keysight Error: Closing HVI"
#define SD_ERROR_MODULE_NOT_OPENED	-8004
#define SD_ERROR_MODULE_NOT_OPENED_STRING	"Keysight Error: Module not opened"
#define SD_ERROR_MODULE_NOT_OPENED_BY_USER	-8005
#define SD_ERROR_MODULE_NOT_OPENED_BY_USER_STRING	"Keysight Error: Module not opened by user"
#define SD_ERROR_MODULE_ALREADY_OPENED	-8006
#define SD_ERROR_MODULE_ALREADY_OPENED_STRING	"Keysight Error: Module already opened"
#define SD_ERROR_HVI_NOT_OPENED	-8007
#define SD_ERROR_HVI_NOT_OPENED_STRING	"Keysight Error: HVI not opened"

2.1.4.2 ID Errors

ID Errors

#define SD_ERROR_INVALID_OBJECTID	-8008
#define SD_ERROR_INVALID_OBJECTID_STRING	"Keysight Error: Invalid ObjectID"
#define SD_ERROR_INVALID_MODULEID	-8009
#define SD_ERROR_INVALID_MODULEID_STRING	"Keysight Error: Invalid ModuleID"
#define SD_ERROR_INVALID_MODULEUSERNAME	-8010
#define SD_ERROR_INVALID_MODULEUSERNAME_STRING	"Keysight Error: Invalid Module User Name"
#define SD_ERROR_INVALID_HVIID	-8011
#define SD_ERROR_INVALID_HVIID_STRING	"Keysight Error: Invalid HVI"
#define SD_ERROR_INVALID_OBJECT	-8012
#define SD_ERROR_INVALID_OBJECT_STRING	"Keysight Error: Invalid Object"

ID Errors	
#define SD_ERROR_INVALID_NCHANNEL	-8013
#define SD_ERROR_INVALID_NCHANNEL_STRING	"Keysight Error: Invalid channel number"
#define SD_ERROR_BUS_DOES_NOT_EXIST	-8014
#define SD_ERROR_BUS_DOES_NOT_EXIST_STRING	"Keysight Error: Bus doesn't exist"
#define SD_ERROR_BITMAP_ASSIGNED_DOES_NOT_EXIST	-8015
#define SD_ERROR_BITMAP_ASSIGNED_DOES_NOT_EXIST_STRING	"Keysight Error: Any input assigned to the bitMap does not exist"
#define SD_ERROR_BUS_INVALID_SIZE	-8016
#define SD_ERROR_BUS_INVALID_SIZE_STRING	"Keysight Error: Input size does not fit on this bus"
#define SD_ERROR_BUS_INVALID_DATA	-8017
#define SD_ERROR_BUS_INVALID_DATA_STRING	"Keysight Error: Input data does not fit on this bus"
#define SD_ERROR_INVALID_VALUE	-8018
#define SD_ERROR_INVALID_VALUE_STRING	"Keysight Error: Invalid value"
#define SD_ERROR_CREATING_WAVE	-8019
#define SD_ERROR_CREATING_WAVE_STRING	"Keysight Error: Creating Waveform"
#define SD_ERROR_NOT_VALID_PARAMETERS	-8020
#define SD_ERROR_NOT_VALID_PARAMETERS_STRING	"Keysight Error: Invalid Parameters"
#define SD_ERROR_AWG	-8021
#define SD_ERROR_AWG_STRING	"Keysight Error: AWG function failed"
#define SD_ERROR_DAQ_INVALID_FUNCTIONALITY	-8022
#define SD_ERROR_DAQ_INVALID_FUNCTIONALITY_STRING	"Keysight Error: Invalid DAQ functionality"
#define SD_ERROR_DAQ_POOL_ALREADY_RUNNING	-8023
#define SD_ERROR_DAQ_POOL_ALREADY_RUNNING_STRING	"Keysight Error: DAQ buffer pool is already running"
#define SD_ERROR_UNKNOWN	-8024
#define SD_ERROR_UNKNOWN_STRING	"Keysight Error: Unknown error"
#define SD_ERROR_INVALID_PARAMETERS	-8025

ID Errors	
#define SD_ERROR_INVALID_PARAMETERS_STRING	"Keysight Error: Invalid parameter"
#define SD_ERROR_MODULE_NOT_FOUND	-8026
#define SD_ERROR_MODULE_NOT_FOUND_STRING	"Keysight Error: Module not found"
#define SD_ERROR_DRIVER_RESOURCE_BUSY	-8027
#define SD_ERROR_DRIVER_RESOURCE_BUSY_STRING	"Keysight Error: Driver resource busy"
#define SD_ERROR_DRIVER_RESOURCE_NOT_READY	-8028
#define SD_ERROR_DRIVER_RESOURCE_NOT_READY_STRING	"Keysight Error: Driver resource not ready"
#define SD_ERROR_DRIVER_ALLOCATE_BUFFER	-8029
#define SD_ERROR_DRIVER_ALLOCATE_BUFFER_STRING	"Keysight Error: Cannot allocate buffer in driver"
#define SD_ERROR_ALLOCATE_BUFFER	-8030
#define SD_ERROR_ALLOCATE_BUFFER_STRING	"Keysight Error: Cannot allocate buffer"
#define SD_ERROR_RESOURCE_NOT_READY	-8031
#define SD_ERROR_RESOURCE_NOT_READY_STRING	"Keysight Error: Resource not ready"
#define SD_ERROR_HARDWARE	-8032
#define SD_ERROR_HARDWARE_STRING	"Keysight Error: Hardware error"
#define SD_ERROR_INVALID_OPERATION	-8033
#define SD_ERROR_INVALID_OPERATION_STRING	"Keysight Error: Invalid Operation"
#define SD_ERROR_NO_COMPILED_CODE	-8034
#define SD_ERROR_NO_COMPILED_CODE_STRING	"Keysight Error: No compiled code in the module"
#define SD_ERROR_FW_VERIFICATION	-8035
#define SD_ERROR_FW_VERIFICATION_STRING	"Keysight Error: Firmware verification failed"
#define SD_ERROR_COMPATIBILITY	-8036
#define SD_ERROR_COMPATIBILITY_MODULE_STRING	"Keysight Error: Compatibility error"
#define SD_ERROR_INVALID_TYPE	-8037
#define SD_ERROR_INVALID_TYPE_STRING	"Keysight Error: Invalid type"
#define SD_ERROR_DEMO_MODULE	-8038

ID Errors	
#define SD_ERROR_DEMO_MODULE_STRING	"Keysight Error: Demo module"
#define SD_ERROR_INVALID_BUFFER	-8039
#define SD_ERROR_INVALID_BUFFER_STRING	"Keysight Error: Invalid buffer"
#define SD_ERROR_INVALID_INDEX	-8040
#define SD_ERROR_INVALID_INDEX_STRING	"Keysight Error: Invalid index"
#define SD_ERROR_INVALID_NHISTOGRAM	-8041
#define SD_ERROR_INVALID_NHISTOGRAM_STRING	"Keysight Error: Invalid histogram number"
#define SD_ERROR_INVALID_NBINS	-8042
#define SD_ERROR_INVALID_NBINS_STRING	"Keysight Error: Invalid number of bins"
#define SD_ERROR_INVALID_MASK	-8043
#define SD_ERROR_INVALID_MASK_STRING	"Keysight Error: Invalid mask"
#define SD_ERROR_INVALID_WAVEFORM	-8044
#define SD_ERROR_INVALID_WAVEFORM_STRING	"Keysight Error: Invalid waveform"
#define SD_ERROR_INVALID_STROBE	-8045
#define SD_ERROR_INVALID_STROBE_STRING	"Keysight Error: Invalid strobe"
#define SD_ERROR_INVALID_STROBE_VALUE	-8046
#define SD_ERROR_INVALID_STROBE_VALUE_STRING	"Keysight Error: Invalid strobe value"
#define SD_ERROR_INVALID_DEBOUNCING	-8047
#define SD_ERROR_INVALID_DEBOUNCING_STRING	"Keysight Error: Invalid debouncing"
#define SD_ERROR_INVALID_PRESCALER	-8048
#define SD_ERROR_INVALID_PRESCALER_STRING	"Keysight Error: Invalid prescaler"
#define SD_ERROR_INVALID_PORT	-8049
#define SD_ERROR_INVALID_PORT_STRING	"Keysight Error: Invalid port"
#define SD_ERROR_INVALID_DIRECTION	-8050
#define SD_ERROR_INVALID_DIRECTION_STRING	"Keysight Error: Invalid direction"
#define SD_ERROR_INVALID_MODE	-8051
#define SD_ERROR_INVALID_MODE_STRING	"Keysight Error: Invalid mode"
#define SD_ERROR_INVALID_FREQUENCY	-8052

ID Errors	
#define SD_ERROR_INVALID_FREQUENCY_STRING "	Keysight Error: Invalid frequency"
#define SD_ERROR_INVALID_IMPEDANCE	-8053
#define SD_ERROR_INVALID_IMPEDANCE_STRING	"Keysight Error: Invalid impedance"
#define SD_ERROR_INVALID_GAIN	-8054
#define SD_ERROR_INVALID_GAIN_STRING	"Keysight Error: Invalid gain"
#define SD_ERROR_INVALID_FULLSCALE	-8055
#define SD_ERROR_INVALID_FULLSCALE_STRING	"Keysight Error: Invalid fullscale"
#define SD_ERROR_INVALID_FILE	-8056
#define SD_ERROR_INVALID_FILE_STRING	"Keysight Error: Invalid file"
#define SD_ERROR_INVALID_SLOT	-8057
#define SD_ERROR_INVALID_SLOT_STRING	"Keysight Error: Invalid slot"
#define SD_ERROR_INVALID_NAME	-8058
#define SD_ERROR_INVALID_NAME_STRING	"Keysight Error: Invalid product name"
#define SD_ERROR_INVALID_SERIAL	-8059
#define SD_ERROR_INVALID_SERIAL_STRING	"Keysight Error: Invalid serial number"
#define SD_ERROR_INVALID_START	-8060
#define SD_ERROR_INVALID_START_STRING	"Keysight Error: Invalid start"
#define SD_ERROR_INVALID_END	-8061
#define SD_ERROR_INVALID_END_STRING	"Keysight Error: Invalid end"
#define SD_ERROR_INVALID_CYCLES	-8062
#define SD_ERROR_INVALID_CYCLES_STRING	"Keysight Error: Invalid number of cycles"
#define SD_ERROR_HVI_INVALID_NUMBER_MODULES	-8063
#define SD_ERROR_HVI_INVALID_NUMBER_MODULES_STRING	"Keysight Error: Invalid number of modules on HVI"
#define SD_ERROR_DAQ_P2P_ALREADY_RUNNING	-8064
#define SD_ERROR_DAQ_P2P_ALREADY_RUNNING_STRING	"Keysight Error: DAQ P2P is already running"

3 Addendum: Keysight Technology and Software Overview

This is an overview of the M3100A, M3102A, M3201A, M3202A, M3300A and M3302A family of PXIe modules.

3.1 Programming Tools

The diagram shown in Figure 11 summarizes the programming tools available to control the compatible Keysight Hardware.

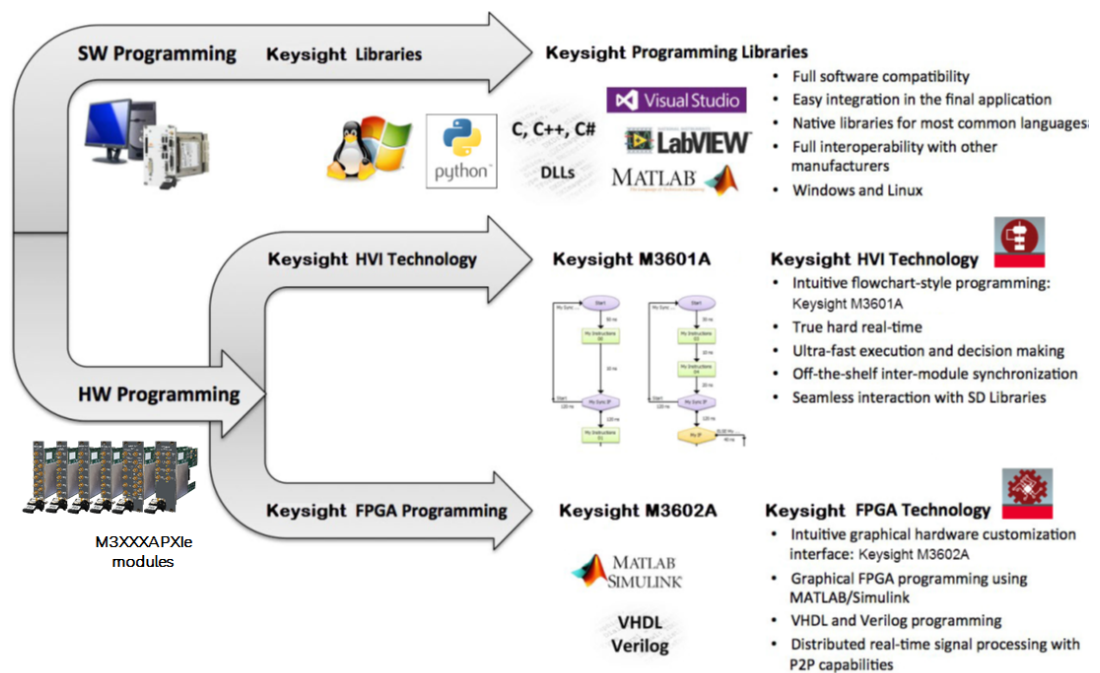


Figure 11: Programming tools for the following Keysight hardware: M3100A, M3102A, M3201A, M3202A, M3300A and M3302A

3.1.1 SW Programming

A comprehensive set of highly optimized software instructions controls the off-the-shelf functionalities of the compatible Keysight hardware. These instructions are compiled into the Programming Libraries. The use of customizable software to create a user-defined control, test and measurement systems is commonly referred as Virtual Instrumentation. In all documentation associated with the M3100A, M3102A, M3201A, M3202A, M3300A or M3302A modules, the concept of a Virtual Instrument (or VI) describes user software that uses programming libraries and is executed by a computer.

3. 1. 1. 1 Keysight Programming Libraries

Keysight provides native programming libraries for a comprehensive set of programming languages, such as C, C++, Visual Studio (VC++, C#, VB), MATLAB, National Instruments LabVIEW, Python, etc., ensuring full software compatibility and seamless multivendor integration. Keysight also provides dynamic libraries, e.g. DLLs, which can be used in virtually any programming language.

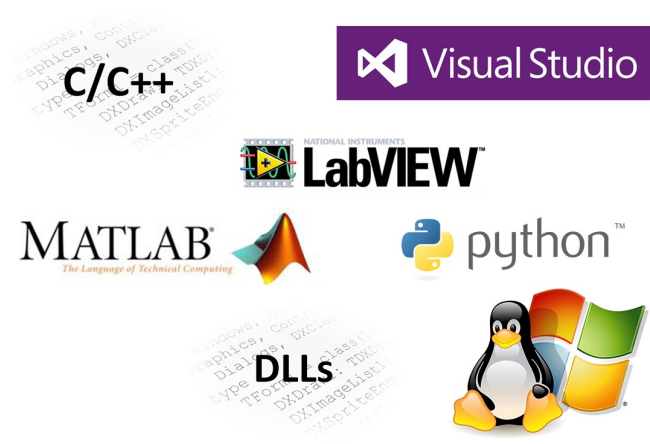


Figure 12: Keysight native programming libraries ensure full compatibility, providing effortless and seamless software integration

3. 1. 2 HW Programming

3. 1. 2. 1 HVI Technology: Keysight M3601A

According to the definition mentioned in the software programming introduction (Section 3.1.1 on the facing page), a VI is a piece of user- defined software executed by a computer, and therefore its real-time performance (speed, latency, etc.) is limited by the computer and by its operating system. In many cases, this real-time performance might not be enough for the application, even with a real-time operating system. Also, many modern applications require tight triggering and precise intermodule synchronization, making the development of final systems very complex and time-consuming.

For all these applications, Keysight has developed an exclusive technology called Hard Virtual Instrumentation. In a Hard Virtual instrument (or HVI), the user application is executed by the hardware modules independently of the computer, which stays free for other VI tasks, like visualization, user interaction, etc. The I/O modules run in parallel, completely synchronized, and exchange data and decisions in real-time. The result is a set of modules that behave like a single integrated real-time instrument.

NOTE

HVIs vs. VIs: Keysight’s HVI technology provides the capability to create time-deterministic execution sequences with the same instructions available in the Keysight Programming Libraries.

HVIs are programmed with Keysight M3601A [2], an HVI programming environment with a user-friendly flowchart-style interface, compatible with all the following Keysight PXle modules: M3100A, M3102A, M3201A, M3202A, M3300A and M3302A with -HV1 option enabled.



M3601A

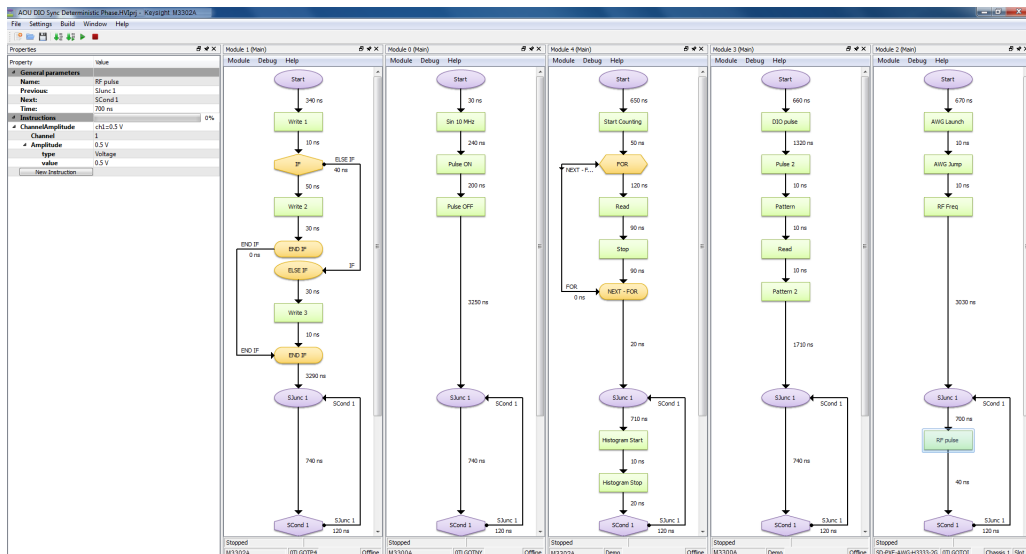


Figure 13: Keysight M3601A, a user-friendly flowchart-style HVI programming Environment
Keysight’s Hard Virtual Instrumentation technology provides:

- Ultra-fast hard real-time execution, processing, and decision making: Execution is hardware-timed and can be as fast as 1 nanosecond, matching very high-performance FPGA-based systems and outperforming any real-time operating system.
- User-friendly flowchart-style programming interface: Keysight M3601A provides an intuitive flowchart-style programming environment that makes HVI programming extremely fast and easy (Figure 14). Using M3601A and its set of built-in instructions (the same instructions available for VIs), the user can

program the hardware modules without any knowledge in FPGA technology, VHDL, etc.

- Off-the-shelf intermodule synchronization and data exchange: Each HVI is defined by a group of hardware modules which work perfectly synchronized, without the need of any external trigger or additional external hardware (Figure 15). Also, Keysight modules exchange data and decisions for ultra-fast control algorithms.
- Complete robustness: Execution is performed by hardware, without an operating system, and independently of the user PC.
- Seamless integration with custom FPGA functions (modules with option -FP1 Enabled FPGA Programming and -K32 or -K41 logic): HVIs can interact with user-defined FPGA functions, making the real-time processing capabilities of HVIs unlimited.
- Seamless integration with Keysight Programming Libraries: In a complex control or test system, there are still some non-time-critical tasks that can only be performed by a VI, like for example user interaction, visualization, or processing and decisions tasks which are too complex to be implemented by hardware. Therefore, in a real application, the combination of VIs and HVIs is required. This task can be performed seamlessly with the Keysight programming tools, e.g. the user can have many HVIs and can control them from a VI using instructions like start, stop, pause, etc.

New hardware functionalities without FPGA programming:
Keysight's HVI technology is the perfect tool to create new hardware functionalities with FPGA-like performance and without any FPGA programming knowledge. Users can create a repository of HVIs that can be launched from VIs using the Keysight Programming Libraries.

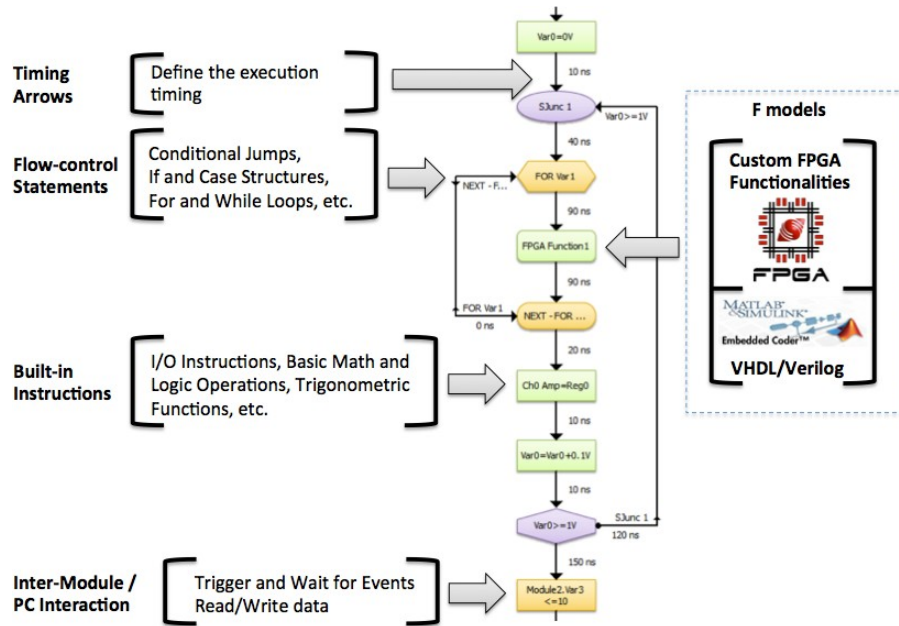


Figure 14: HVI flowchart elements. Keysight M3601A is based on flowchart programming, providing an easy-to-use environment to develop hard real-time applications

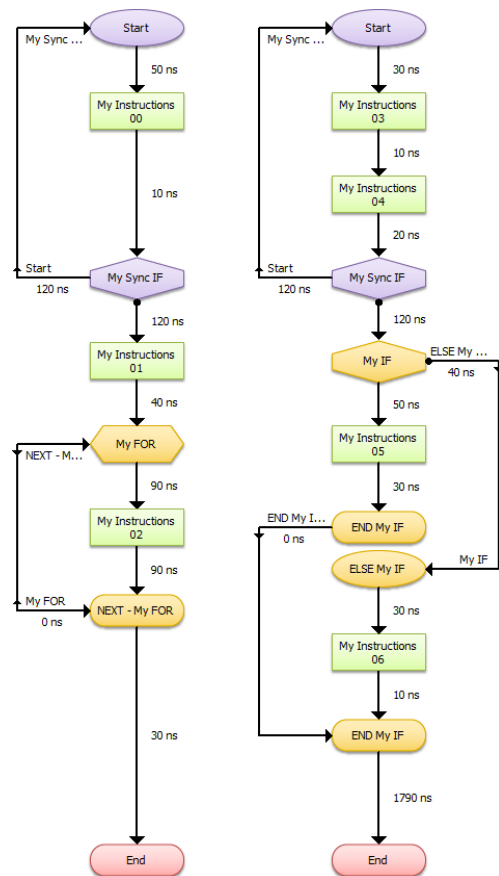


Figure 15: HVI example with two hardware modules. In an HVI, all Keysight modules run in parallel and completely synchronized, executing one flowchart per module. This results in simpler systems without the need of triggers.

3.1.2.2 FPGA Programming: Keysight M3602A

Some applications require the use of custom onboard real-time processing which might not be covered by the comprehensive off-the-shelf functionalities of the standard hardware products. For these applications, Keysight supplies hardware products that provide the capability to program the onboard FPGA.

Keysight M3100A, M3102A, M3201A, M3202A, M3300A and M3302A PXie modular hardware family of products have as an option the -FP1 Enabled FPGA Programming capability with -K32 or -K41 logic and they keep the same built-in functionalities of their standard counterparts, giving the users more time to focus on their specific functionalities. For example, using the -FP1 enabled FPGA Programming with -K32 or -K41 logic version of a Keysight digitizer, the user has all the off-the-shelf functionalities of the hardware (data capture, triggering, etc.), but custom real-time FPGA processing can be added in the data path, between the acquisition and the transmission of data to the computer.

Keysight FPGA programming technology is managed with Keysight M3602A, an intuitive graphical environment to customize Keysight compatible hardware.



M3602A

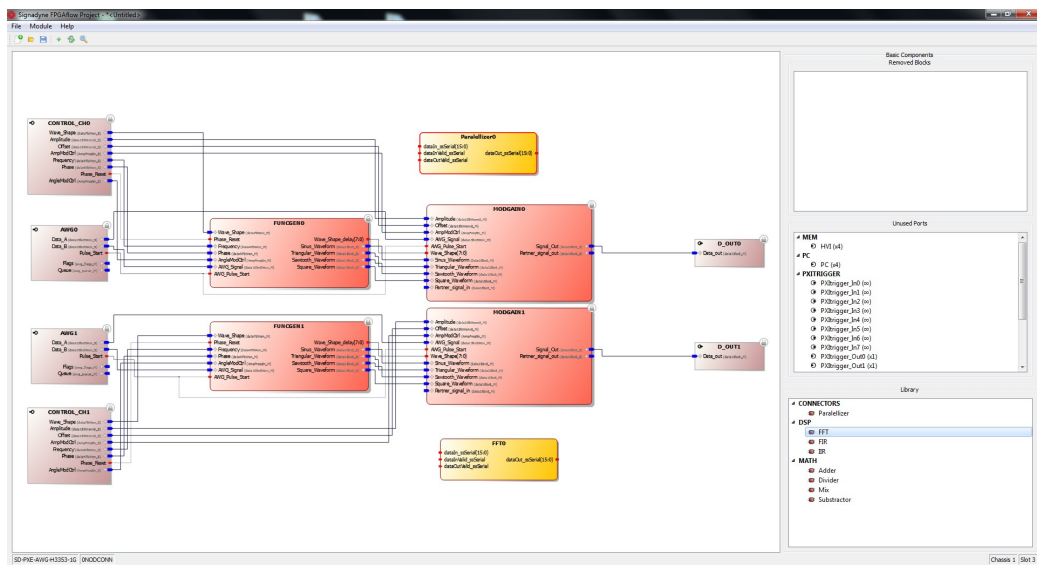


Figure 16: Keysight M3602A provides an intuitive graphical FPGA customization interface

NOTE FPGA programming made simple: Full language compatibility (including the graphical environment MATLAB/Simulink) and an easy-to-use FPGA graphical IDE, make Keysight FPGA programming extremely simple.

3.1.2.3 Keysight M3602A: An FPGA Design Environment

Keysight M3602A is a complete FPGA design environment that allows the user to customize M3XXXA PXle hardware products. M3602A provides the necessary tools to design, compile and program the FPGA of the module (Figure 15).

Keysight M3602A provides the following features:

- User-friendly graphical FPGA programming environment:
- Complete platform, from design to FPGA programming: Keysight M3602A provides the necessary tools to design, compile and program the FPGA of the module (Figure 15)
- 5x faster project development
- Graphical environment without performance penalty
- FPGA know-how requirement minimized: The graphical environment provides a tool which does not require an extensive know-how in FPGA technology, improving the learning curve drastically.

Streamlined design process

- Ready-to-use Keysight Block Library: M3602A provides a continuously-growing library of blocks which reduces the need for custom FPGA-code development.
- Include VHDL, Verilog, or Xilinx VIVADO/ISE projects: Experienced FPGA users can squeeze the power of the onboard FPGA.
- Include MATLAB/Simulink projects: MATLAB/Simulink in conjunction with Xilinx System Generator for DSP provides a powerful tool to implement Digital Signal Processing. The user can go from the design/simulation power of MATLAB/Simulink to M3602A code in just a few clicks.
- Include Xilinx CORE Generator IP cores: Xilinx CORE Generator can be launched by M3602A to create IP cores that can be seamlessly included in the design.
- Add and remove built-in resources to free up space: The user can remove unused built-in resources to free up more FPGA space.

One-click compiling and programming:

- 3x faster ultra-secure cloud FPGA compiling: An ultra-fast cloud compiling system provides up to 3 times faster compiling. An ultra-secure TLS encrypted

communication protects the IP of the user.

- 100x faster hot programming via PCI Express without rebooting: Hardware can be reprogrammed without external cables and without rebooting the system.

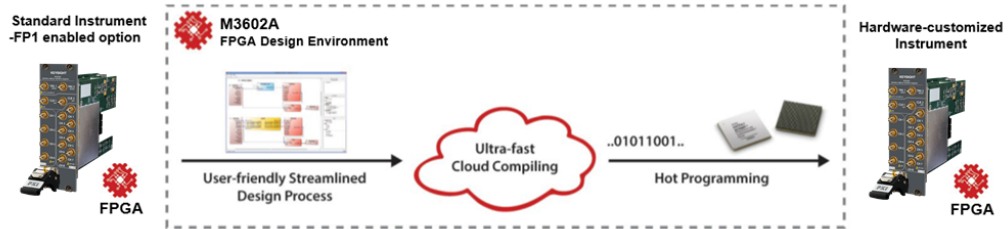


Figure 17: Keysight M3602A: a platform that provides the complete flow from design to FPGA programming

3.2 Design Process: Customization vs. Complete Design

The M3602A FPGA Design Environment simplifies the development of custom processing functions for the following -FP1 enabled FPGA programming PXIe modules: M3100A, M3102A, M3201A, M3202A, M3300A, M3302A. These products are delivered with all the off-the-shelf functionalities of the standard products, and therefore the development time is dramatically reduced. The user can focus exclusively on expanding the functionality of the standard instrument, instead of developing a completely new one.

In Keysight M3602A, FPGA code is represented as boxes (called blocks) with IO ports. An empty project contains the "Default Product Blocks" (off-the-shelf functionalities), and the "Design IO Blocks" that provide the outer interface of the design. The user can then add/remove blocks from the Keysight Block Library, External Blocks or Xilinx IP cores

.....

3.3 Application Software

3.3.1 Keysight SD1 SFP

All Keysight modules can be operated as classical workbench instruments using Keysight SD1 SFP [3], a ready-to-use software front panels for live operation. When SD1 SFP opens, it identifies all Keysight hardware connected to the computer, opening the corresponding front panels.



SD1 SFP

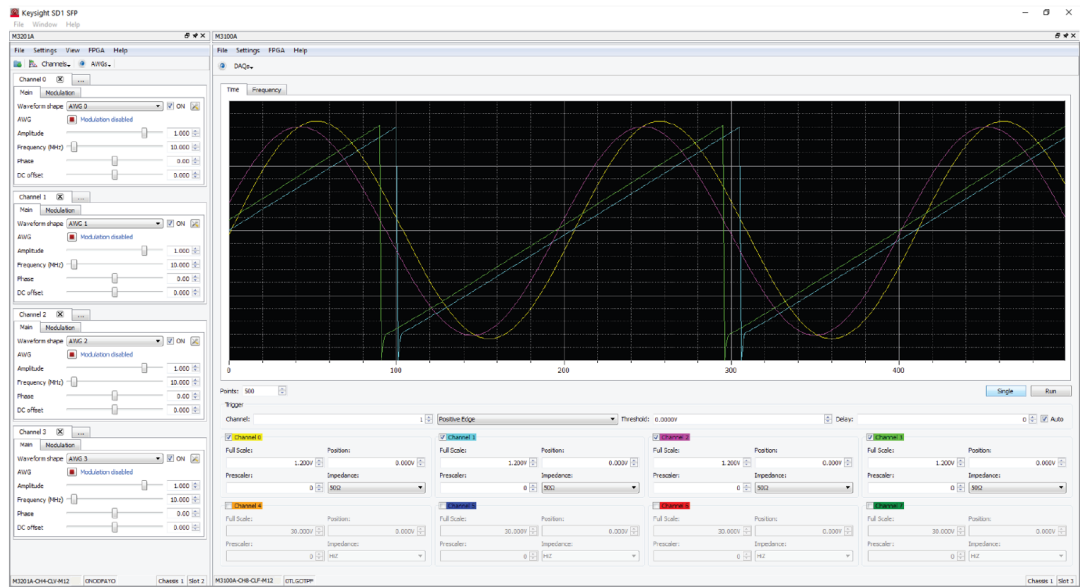


Figure 18: Keysight SD1 SFP provides software front panels, a fast and intuitive way of operating any of the following modules: M3100A, M3102A, M3201A, M3202A, M3300A and M3302A

3. 4 M3602A Initial Setup

This topic covers the M3602A license and software configuration:

- [Add License to product](#)
- [Vivado 2015.2 installation](#)
- [Cloud server connection](#)

3. 4. 1 Add License to Product

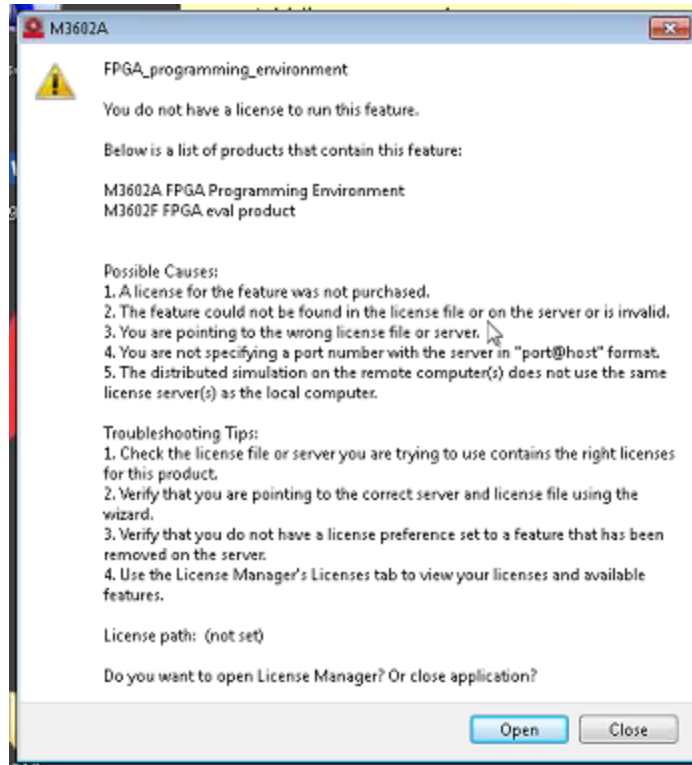
NOTE

It is assumed that the M3602A software has been downloaded and installed onto your PC.

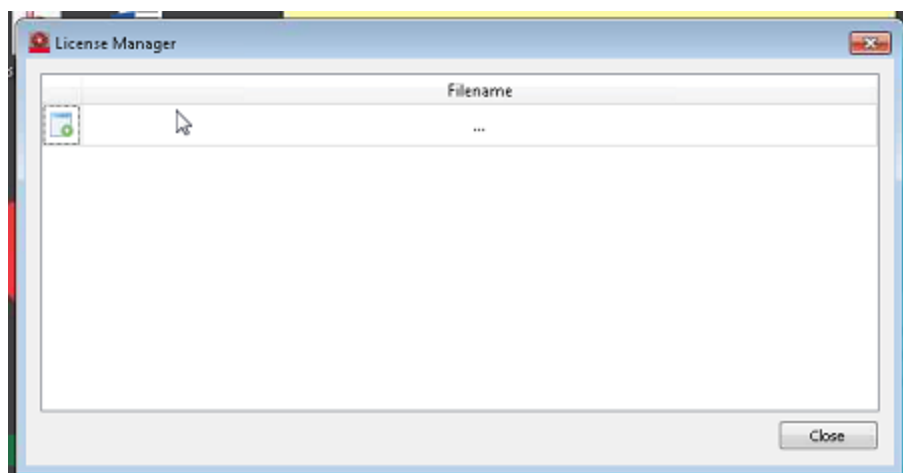
1. Navigate on the PC to where the M3602A software has been installed. A common folder for the software is labeled, Keysight M3602A. Within that folder will

be the Keysight M3602A.exe, which can be launched.

2. When the software is first launched, a dialog opens requesting that the software license associated with the software be linked to it.

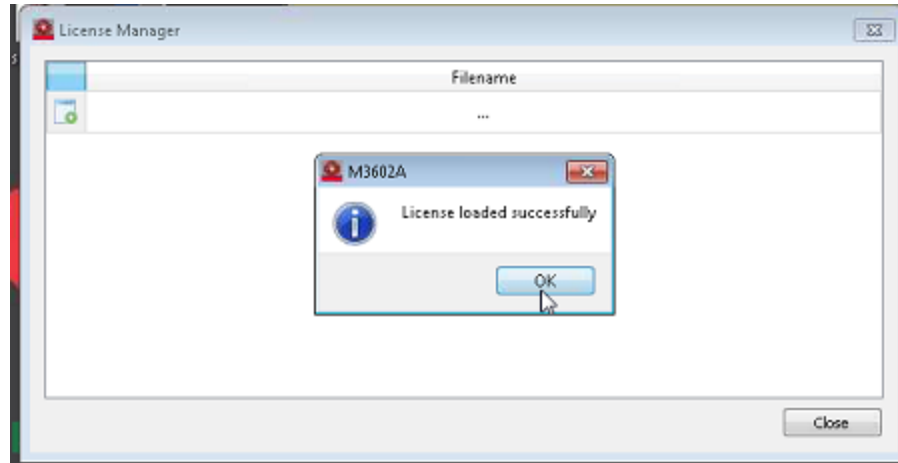


3. In the dialog box, open the License Manager.



4. Click the icon to the top left of the License Manager window.
5. Navigate to where the software license is located on the PC.

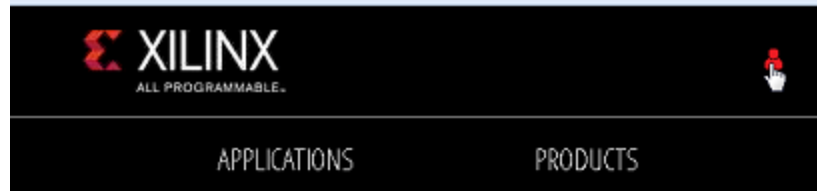
6. Highlight the license and click Open.
7. A "License loaded successfully" message appears, click OK to dismiss it.



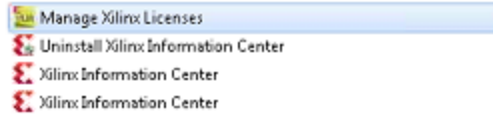
8. The license will appear in the License Manager window, which can now be closed.
9. The software will try again to find the license. Having now found it, the M3602A software will open.

3. 4. 2 Vivado 2015.2 Installation

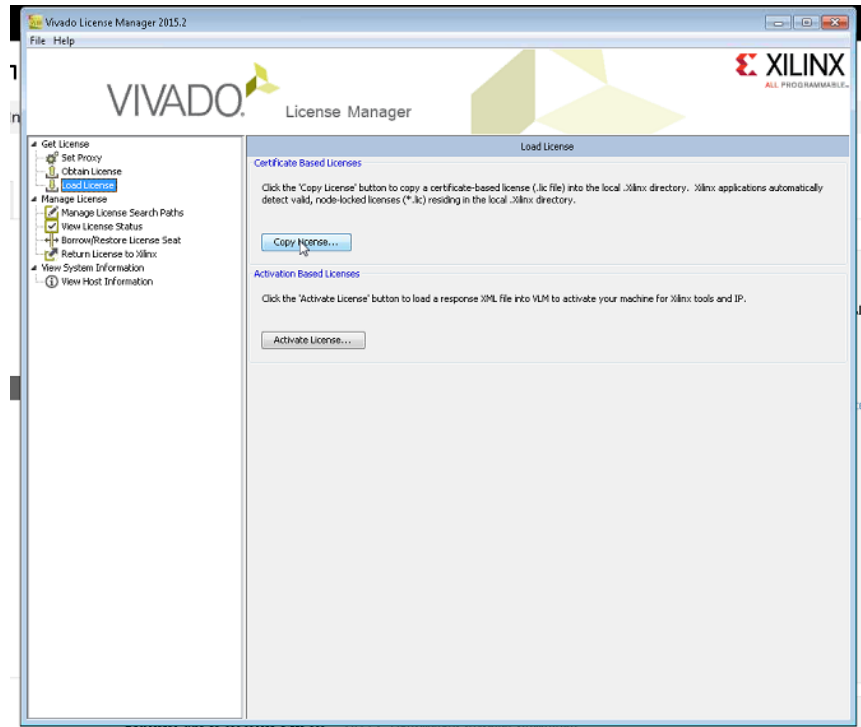
1. To download the Vivado 2015.2 software, go to <https://www.xilinx.com>.
2. Go to "Products"
3. In the search bar enter "Vivado", and from the drop down, select "Vivado 2015.2 download", or search on that if it is not among the selections.
4. Follow directions for installing the software on the PC.
5. Once installed, a Xilinx Web Pack license needs to be obtained to associate with the software. To do this, create a new user by clicking the new user icon as shown below.



6. Navigate to the Xilinx License Manager on the PC. For example:

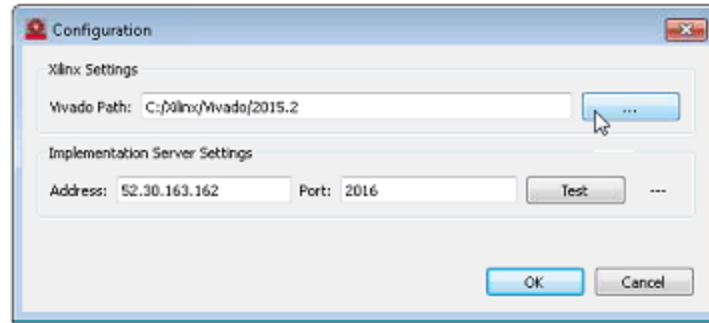


7. Once the Xilinx License Manager has been opened, the license can be associated with the software:



8. Once the Xilinx License is associated with the Vivado 20152 software, open the Keysight M3602A software to check the configuration settings.

9. Navigate **File > Settings** to check that the Vivado path is set correctly. For example:



10. Check the path shown in the Vivado Path: text box is correct. If incorrect, select the Browse button and navigate to the correct folder.

3. 4. 3 Cloud Server Connection

1. Navigate File > Settings as in step 9 above.
2. Enter the IP address of the server and port provided with the M3602A software carefully in the Address and Port text boxes.
3. After clicking the Test button, an OK icon will appear to the right of the Test button.



This information is subject to change without notice.

© Keysight Technologies 2017

Edition 1, March 2017

M3602-90001

www.keysight.com