Summary

This document aims to assist the user in the identification and use of the various components involved in the functionality of the product.

Products

This documentation pertains to the following products

Category	Elk Name	Avnet Name	Avnet P/N
System-on-Module (SOM)	DAQ16	XRF16 Gen2 SOM	AES-XRF16-ZU39-G
	RTX16	XRF16 Gen3 SOM	AES-XRF16-ZU49-G
	DAQ8	XRF8 Gen3 SOM	AES-XRF8-ZU47-G
SOM Carrier	DAQ16 Carrier	XRF16™ Carrier	AES-XRF16-CC-G
	DAQ8 Carrier	XRF8 Carrier	AES-XRF8-CC-G
Adapter	Isorate Breakout	XRF-ISORATEBB-G	AES-XRF-ISORATEBB-G

IP Repositories

The entirety of the soft deliverables, including source code for the firmware, software and some hardware elements are provided via BSP packaging and Gitlab repositores. Contact your local Avnet FAE to obtain access credentials to reach the Gitlab repositories.

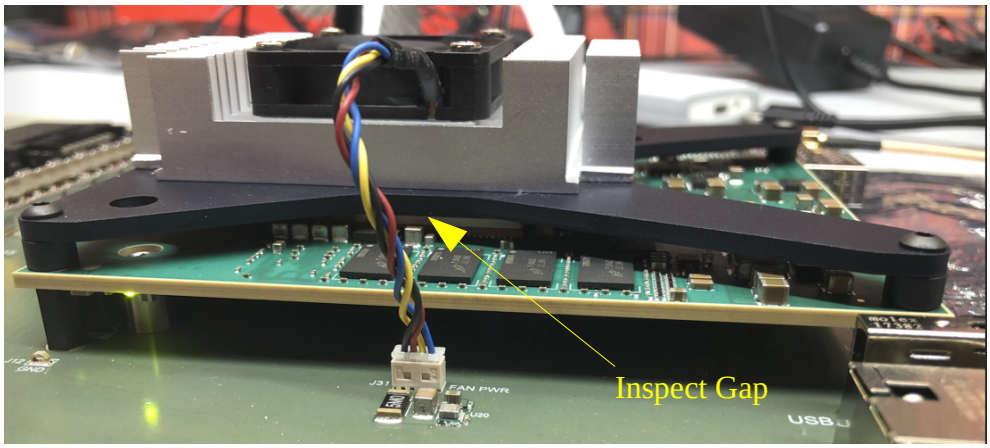
Quick Start: Initial Use

The SOM ships pre-programmed with firmware and a bootable Petalinux image. Follow the instructions below to install and operate the product before attempting to reprogram it with custom firmware or software.

Unbox the carrier and SOM at a suitable workstation exercising good anti-static protocols. Follow the general guidance in this installation video during hardware setup.

Important Note:

Before powering up for the first time, visually inspect the interface between the graphite shim+ 1mm thermal pad atop the RFSoC and the bottom of the fansink. No visible gap should exist. Use back-illumination to highlight the gap, if present. The bottom of the fansink should *slightly depress* the thermal pad to obtain adequate heat transfer from the RFSoC.



Visually verify FPGA to thermal pad contact

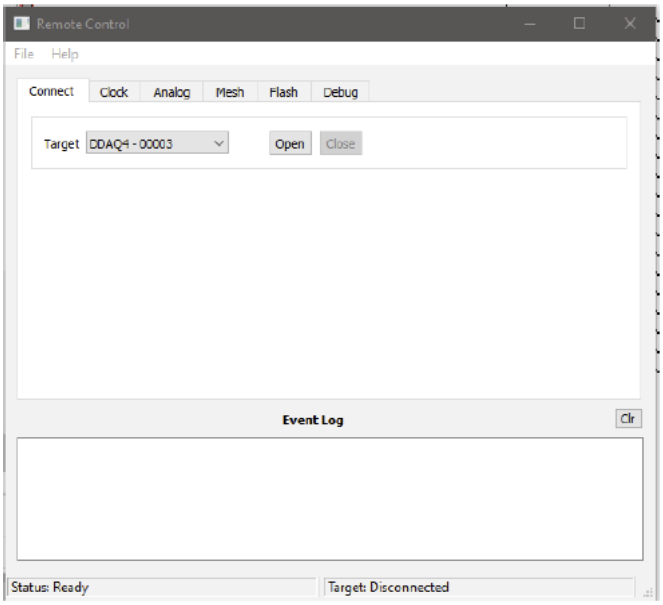
If additional thickness is required, 0.5mm thermal pad material is available from Amazon:

Iceberg Thermal DRIFTIce Thermal Pad 80mm x 40mm x 0.5mm	Hand cut to 40x40mm

Install a micro USB cable between the UART connector on the carrier and the host development PC. Install an ethernet cord between the carrier RJ45 jack and your local network. Install the 12V barrel

jack to the carrier (adapter needed for DAQ8 carrier). Power on the carrier and boot the SOM. Launch Putty to observe the Linux boot process. During the Linux boot process, a green LED will blink, stopping after the Talker.elf target executable executes which occurs automatically at the conclusion of the target Petalinux boot process.

Run the pre-built RemoteControl executable. The target should auto-detect, as illustrated below. Refer to the board-specific Avalon Library User's Manual chapter 3 (Creating Host Applications) for details on the RemoteControl applet. Use the provided target Talker.cpp/h and host RemoteControl application source codes as templates for your custom application development.



The RemoteControl executable communicates with the SOM via ethernet and illustrates PLL clock programming, RF signal generation, triggering, capture, up/down conversion, flash ROM access and SERDES communications. Familiarize yourself with this software prior to reprogramming the target and host.

Precompiled images of the factory software and firmware are available in the Gitlab repositories, so that you may revert to the factory state if your custom firmware renders the board unusable or if you question is fundamental operation.

Install the tools below on a Windows development PC to allow control of the SOM/carrier after hardware installation.

Windows Host Software Installation

Software Tool
Avalon runtime libraries – Run self-installer *SetupAvalonRed.exe* to install essential support runtime binaries (IPP and graphics DLLs).
Avalon development sources – Run self-installer *SetupAvalonHost.exe* to install the Windows port of Avalon libraries. Needed to recompile RemoteControl executable under QtCreator.
Binview – Clone this repo to to a utilities directory. Add Win64/ subdir to Windows path. Waveform time/frequency domain visualization tool. Binview.exe will launch successfully only if the Avalon runtime libraries are installed.
QtCreator – Clone this repo to to a scratch directory. Execute QtInstall.bat to install Windows port of cross-platform C++ development environment needed to rebuild RemoteControl.exe example.
RemoteControl – Clone this repo to a development directory. QtCreator project with Avalon dependencies. Pre-compiled binary RemoteControl.exe needed to interact with target Talker.elf to demonstrate SOM functionality.
Putty – Terminal emulator needed to interact with target Petalinux console. Configure for 115,200 baud communications and the serial port associated with the target SOM.

Install the tools below on a development PC to allow cross-compilation of Petalinux and Elk example programs.

Platform-agnostic Host Software Installation

Software Tool
Xilinx Vitis - Self-installer for ARM SDK and cross-compiler
Putty – Terminal emulator needed to interact with target Petalinux console. Configure for 115,200 baud communications and the serial port associated with the target SOM.

Linux -specific Host Software Installation

Software Tool
Petalinux – Self installer for Petalinux cross compiler
sh self-extracting archive – Extract the Petalinux sdk archive into a directory named `sdk2021.2/` in the root of the Vitis workspace.

From a Linux command line, navigate into the Vitis workspace. Create a symbolic link named sdk which points to sdk2021.2/sysroots/cortexa72-cortexa53-xilinx-linux/ sub-folder using the command:

```
ln -s sdk2021.2/sysroots/cortexa72-cortexa53-xilinx-linux/ sdk
```

Windows -specific Host Software Installation

Software Tool
sdk2021.2.tar.gz – Gzipped tarball. **Using 7zip running in administrator mode**, open this file, descend to the sdk2021.2 folder, then extract its contents into the root of the Vitis workspace to create a folder named sdk2021.2\ containing the sdk files.

From a Windows command line (with administrator priviledges), navigate into the Vitis workspace.

Create a symbolic link named sdk which points to sdk2021.2\sysroots/cortexa72-cortexa53-xilinx-linux\ sub-folder using the command:

```
mklink /D sdk sdk2021.2\sysroots\cortexa72-cortexa53-xilinx-linux\
```

This *sdk* link is implicitly referenced by the supplied target example projects when compiling applications using the xrfdc driver and other key Petalinux/Xilinx features.

Quick Start: Firmware Update

SOM-specific firmware or Petalinux images have been pre-built and the Gitlab repositories updated accordingly. These repositories contain the following image files:

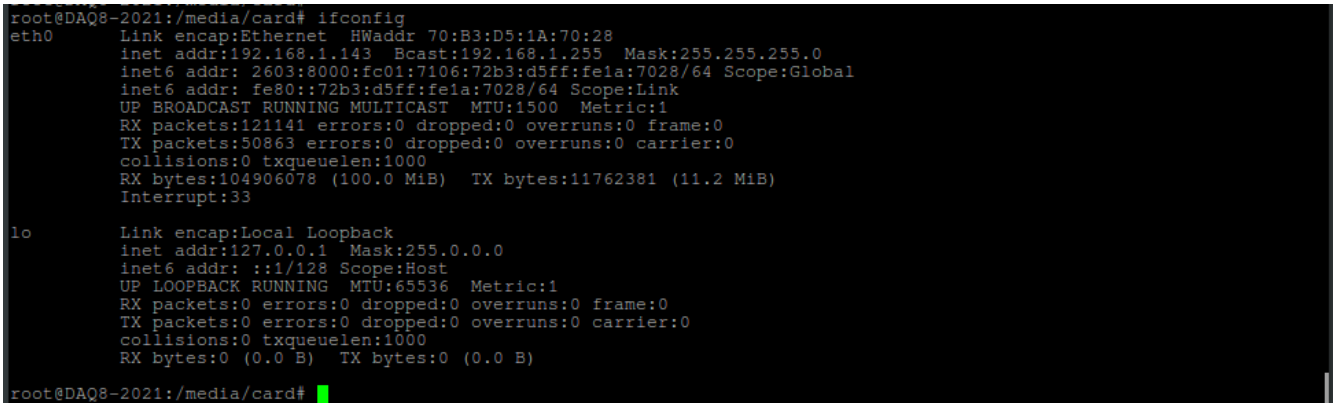
File	Function
BOOT.BIN	Vivado-generated firmware image resulting from build of SOM-specific firmware
boot.scr	Default uboot boot-script
image.ub	Firmware image resulting from build of SOM-specific Petalinux-SoM
zynqmp_fsbl.elf	Xilinx first-stage boot loader.

The latest such images are accessible via the links below:

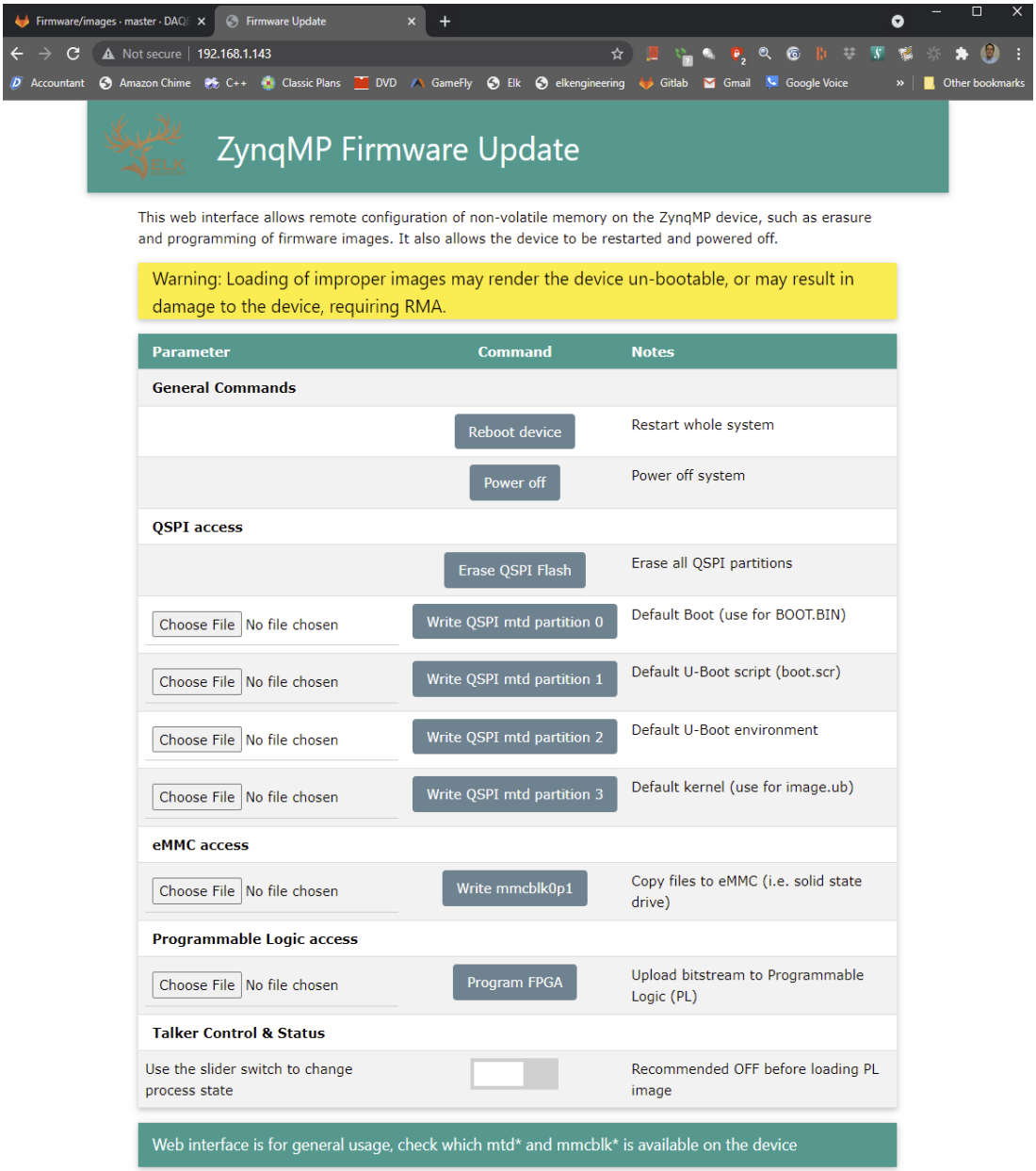
SOM
DAQ16 Firmware Images
DAQ8 Firmware Images

RTX16 Firmware Images

To upgrade the firmware boot the SOM to the Petalinux prompt then note its IP address by typing the Linux command *ifconfig*:



Launch the board web GUI by entering the IP address of the SOM (discovered above) into a browser window:



Sequentially, use the Choose File buttons within the QSPI access group on the web GUI to program *BOOT.BIN* into QSPI partition 0, *boot.scr* into QSPI partition 1 and *image.ub* into QSPI partition 3. Programming partition 3 is the most time-consuming. Once complete, power-cycle the SOM to put all

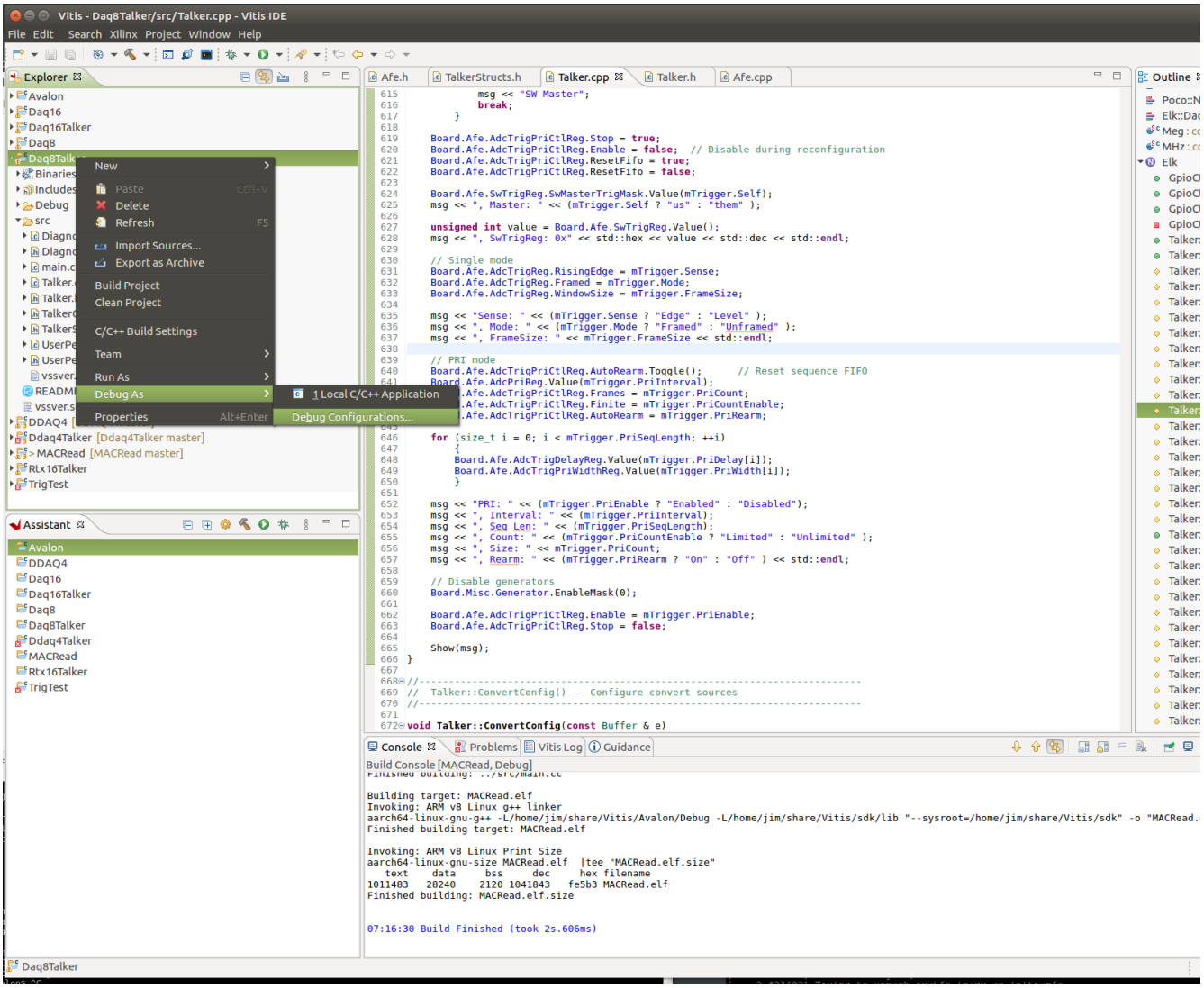
images into effect. If only one of the images has been altered, only its corresponding partition need be reprogrammed.

Quick Start: Talker Update

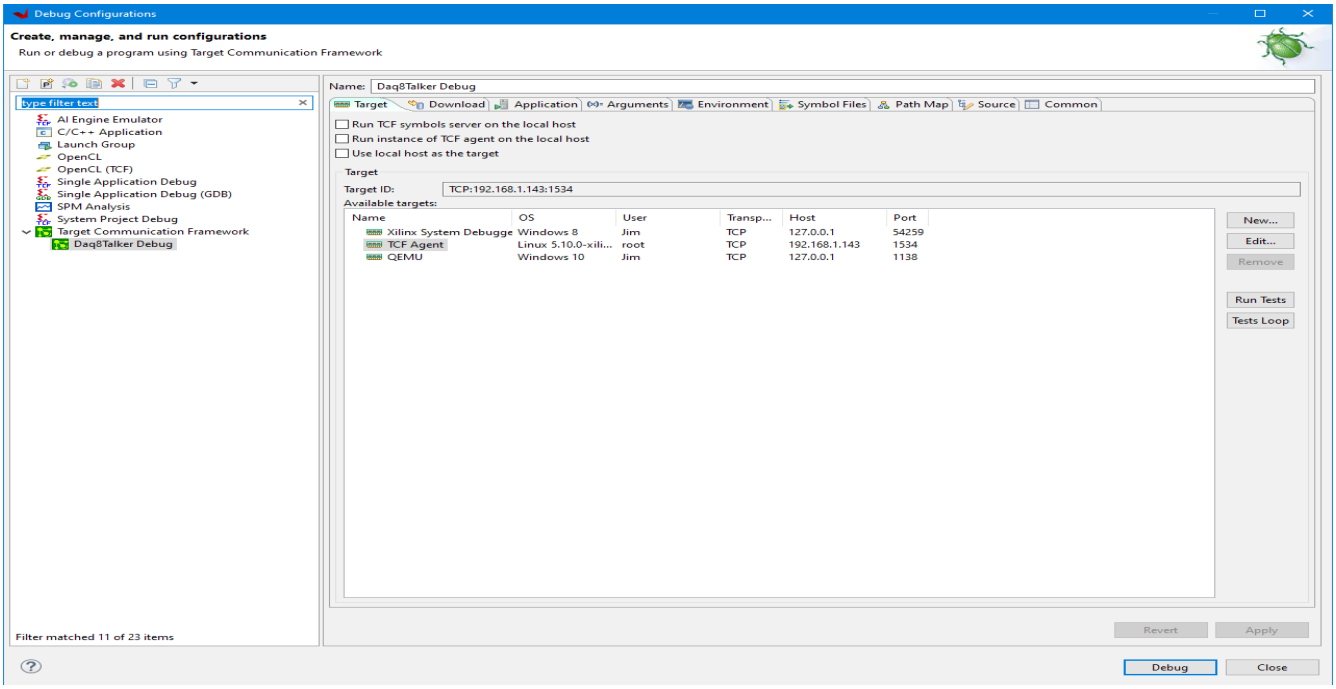
The Talker.elf image is placed into /media/card/Talker.elf during factory initialization. If present in this folder during Petalinux boot, this application is automatically run in the background as a daemon. The daemon may be stopped/started using the *Talker Control and Status* button at the bottom of the web GUI. Should a new Talker.elf image become available on Gitlab, use the web GUI to stop the daemon, then use *Choose File* within the eMMC access on the web GUI to load the new Talker.elf image. The most recent images are stored in Gitlab at the locations listed below.

File
DAQ16 Talker executable
DAQ8 Talker executable
RTX16 Talker executable

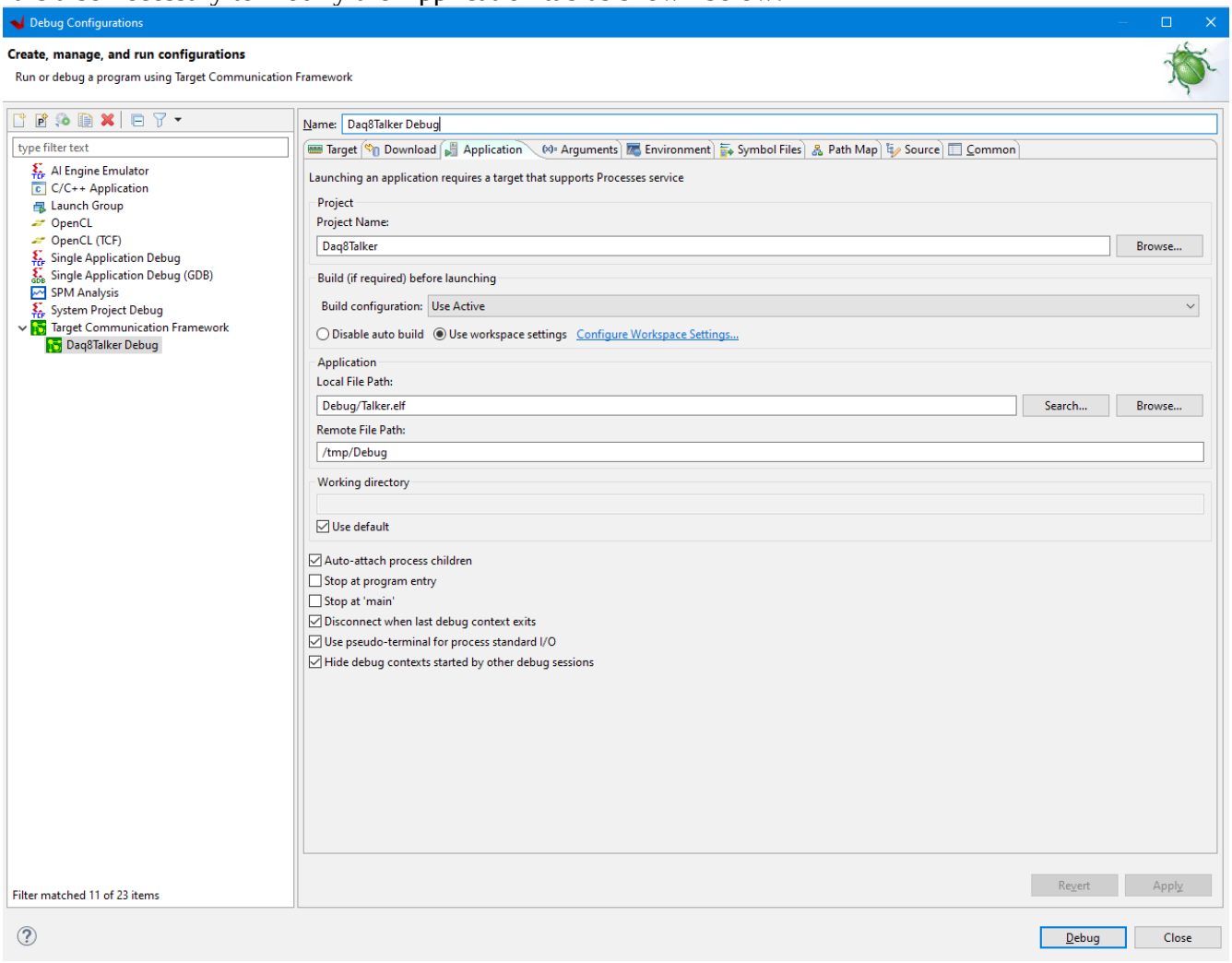
During application development it is expedient to create a copy of the SOM-specific Talker project and modify it to meet custom application requirements. During such development, stop the Talker daemon and rename the */media/card/Talker.elf* image to */media/card/TalkerOld.elf*. Cross compile and launch the custom Talker application from with the SDK via:



The target communications framework (TCF) is running on the target. The SOM will be listed as an available target running *TCF Agent* on which the Talker can be executed as shown highlighted below:


Debug Configurations
Create, manage, and run configurations
Run or debug a program using Target Communication Framework
type filter text
AI Engine Emulator
C/C++ Application
Launch Group
OpenCL
OpenCL (TCF)
Single Application Debug
Single Application Debug (GDB)
SPM Analysis
System Project Debug
Target Communication Framework
Daq8Talker Debug
Filter matched 11 of 23 items
Name: Daq8Talker Debug
Target
Download
Application
Arguments
Environment
Symbol Files
Path Map
Source
Common
Run TCF symbols server on the local host
Run instance of TCF agent on the local host
Use local host as the target
Target
Target ID:
TCP:192.168.1.143:1534
Available targets:
Name
OS
User
Transp...
Host
Port
Xilinx System Debugge
Windows 8
Jim
TCP
127.0.0.1
54259
TCF Agent
Linux 5.10.0-xili...
root
TCP
192.168.1.143
1534
QEMU
Windows 10
Jim
TCP
127.0.0.1
1138
New...
Edit...
Remove
Run Tests
Tests Loop
Revert
Apply
Debug
Close

It is also necessary to modify the Application tab as shown below:



This allows Vitis to locate the Talker.elf build on the local machine and from where it will run on the remote target.

Click Debug to download and cross-debug the custom Talker.elf. Once development is complete, copy the custom Talker.elf to the /*media*/card folder so that it can autorun at the completion of Petalinux boot during a subsequent cold-start.

Quick Start: Analog Connections

RFSoC analog signals egress via the carrier to Samtec Isorate connectors (P/N P/N IP5-08-01-S-S-RA1-L-TR). Adapter cable P/N IJ5H-08-0610-S-2-01SP1 provides conversion to eight SMA male.

Figure 1: Samtec P/N IJ5H-08-0610-S-2-01SP1

Alternately, Avnet adapter AES-XRF-ISORATEBB-G converts to SMA female via a compact PCB.

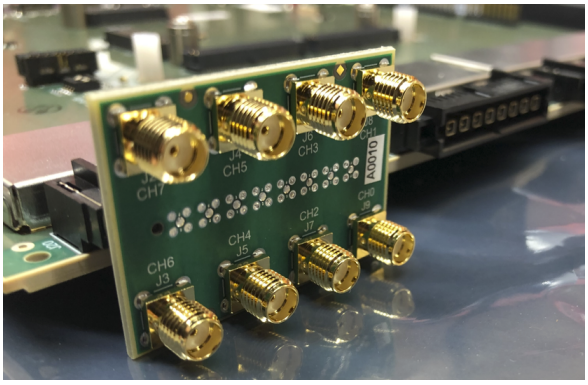
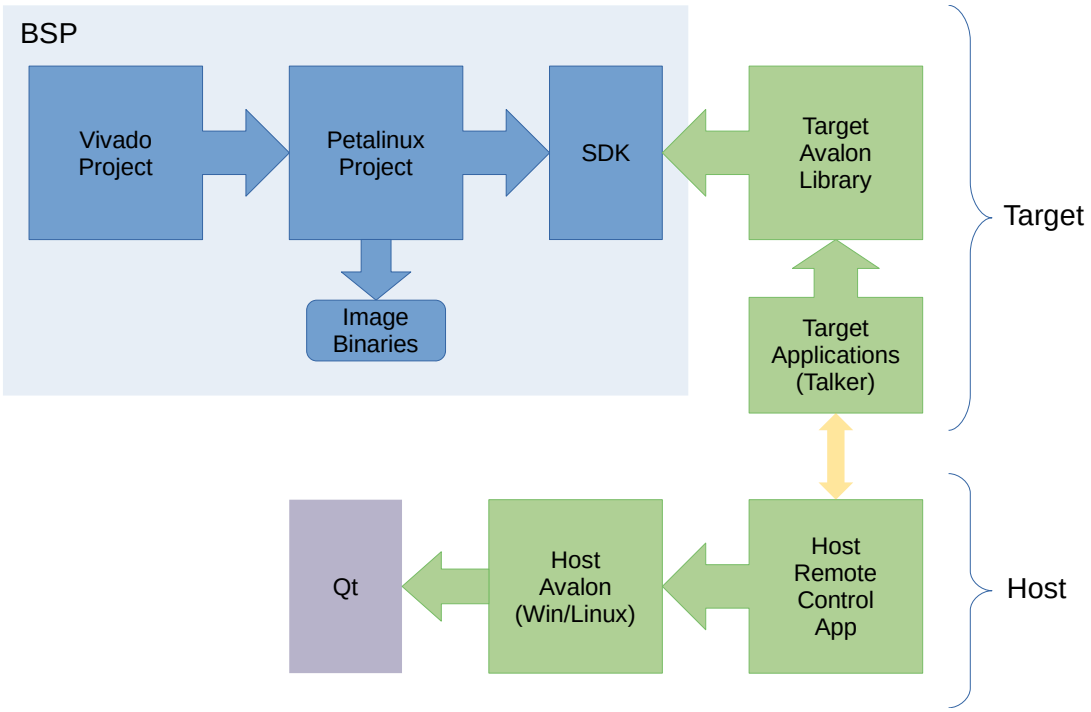


Figure 2: Adapter AES-XRF-ISORATEBB-G

Isorate Adapter Pinout details the pinout of this adapter when mated with the DAQ16 or DAQ8 Carrier products.

Development Flow

The following diagram shows the main software components and their relationship. The components shaded in blue are provided in a packaged BSP format, whereas the shaded green are in Gitlab repos.



Tools

A number of tools are required to compile and generate binaries to be loaded and run on the hardware.

Vitis. The installer is called *Xilinx Unified Installer* and includes Vivado, Vitis Core Development Kit, and optional packages such as Vitis HLS and Vitis Model Composer. The latest supported and recommended version is 2021.2.

For Vitis/Vivado Operating System requirements please refer to ug973.

Petalinux. The installer is a separate package and can be downloaded from Xilinx Petalinux installer from Xilinx website. The Operating System requirements vary, as seen in ug1144. It's recommended to

be installed on a supported Linux host for cross-development. For a Windows host, either a Virtual Machine (VM) such as VirtualBox, or Docker can be used to install a target Linux OS.

QtCreator and Qt SDK. For host applications, these are required and may be downloaded from Qt website. The precompiled static version of the Qt libraries and toolchain for a Windows host used to buid the supplied Windows host example is available here. The Qt website contains the latest tools and libraries.

Getting the source code

The following table contains the various links to obtain the source code of the project's components. The packages are split into *common* for all products, and *product-specific*, in the tables below.

Common Packages	Source
Board Support Package*	Target
Readme BSP*	Host
Petalinux helper scripts*	Host
Petalinux project	Target OS
Avalon	Library
MAC reader app	Target
RemoteControl app	Host
Vivado common library	Firmware

* Liquidfiles Login credentials @ liquidfiles.elkengineering.com:
email: support@elkengineering.com password: RFSoC_SOM

DAQ16 Packages	Source
Talker app	Target
Vivado project	Firmware
DAQ8 Packages	**Source**
Talker app	Target
Vivado project	Firmware
RTX16	**Source**
Talker app	Target
Vivado project	Firmware

Tools Documentation

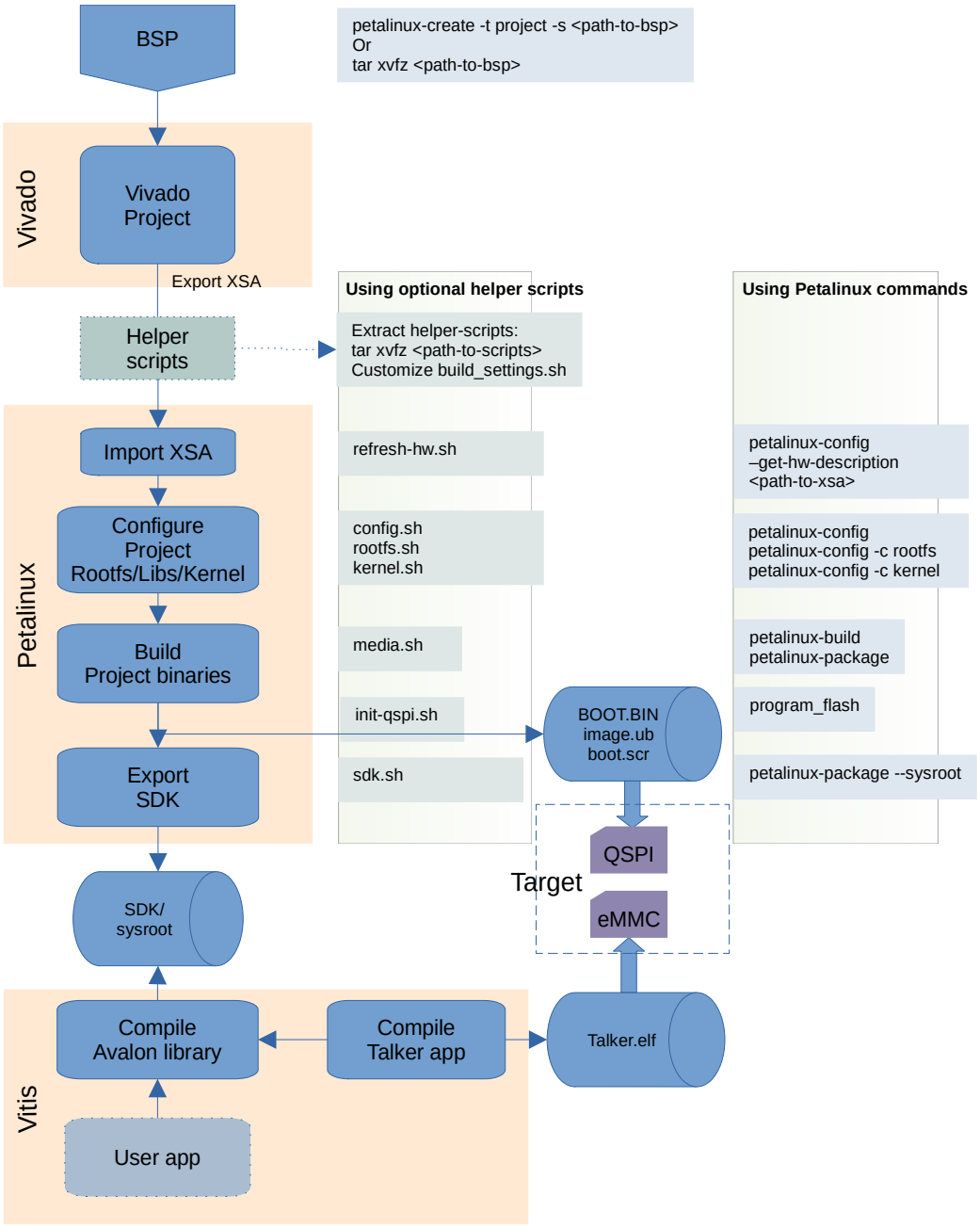
Most of the Xilinx documentation can be found using their DocNav application, that comes as an option during the Vitis installation. Here's a summary of a few documents that should be reviewed for insight.

Tool	Document
Vivado	
ug910	Getting started
ug1046	Ultrafast design methodology guide
ug994	Designing IP subsystems using IP integrator
ug898	Embedded processory hardware design
ug908	Programming and debugging
ug1037	AXI reference guide
Zynq RFSoC	
ug1085	Zynq Ultrascale+ MPSoC Technical Reference Manual (TRM)
pg269	Zynq Ultrascale+ RFSoC RF Data Converter Product Guide
ug1087	Zynq Ultrascale+ MPSoC Register Reference
Petalinux	
ug1144	Petalinux Tools Documentation: Reference Guide

Vitis	
ug643	Vitis Embedded Software Development

Process flow

The following flowchart describes the complete lifespan of a project with the various sources and tools that make it possible.



SoM Documentation

Product	Document
DAQ16	Firmware User Guide
	Firmware Memory Map
	Hardware Manual
	Device Specifications
	Clock Topology
	SoM Installation
	Avalon User Guide
	Avalon Reference
	Communications Architecture
DAQ8	Firmware User Guide
	Firmware Memory Map
	Hardware Manual
	Device Specifications
	Avalon User Guide
	Avalon Reference
	Communications Architecture
RTX16	Firmware User Guide
	Firmware Memory Map
	Hardware Manual
	Device Specifications
	Avalon User Guide
	Avalon Reference
	Communications Architecture

Tutorials

The following videos and documents attempt to clarify installation, multi-tile sync functionality, clocking and other key capabilities.

Topic
10 Gb Ethernet application Note Addendum
GbE phy debugging
Client/Server communications
Multi-board synchronization
Petalinux MMC boot notes
Booting from eMMC notes
QSPI reprogramming
Rebuilding Petalinux from scratch - Part 1
Rebuilding Petalinux from scratch – Part II
Reference clock calculator worksheet
Remote debugging via ethernet
SOM installation/Assembly
SmartLynq Pod connections
Dual SOM synchronization demo
Updating SOM firmware
Using the DDC
Using Linux’ libgpiod library
Multiboard sync ref plan