VectorBlox™ Video Kit Demo Guide v2.0.3

Introduction

This demo demonstrates CoreVectorBlox neural network acceleration on PolarFire® Field Programmable Gate Array (FPGA) devices. This document provides instructions for using the associated reference design and describes how to run the CoreVectorBlox Neural Network using the PolarFire Video Kit, the Dual Camera sensor module, an HDMI monitor and optionally an HDMI source such as a laptop. The demo design features a fully integrated solution developed using Microchip Libero® SoC software to help customers evaluate PolarFire FPGA in Neural Network Vision applications and to build prototypes quickly. For more information, see Smart Embedded Vision.

The demo demonstrates the following functions:

- MIPI CSI-2 RX to read one of the cameras
- HDMI display controller
- VectorBlox CNN acceleration of piplined network demos
- Image enhancements such as contrast, brightness and color balance

The PolarFire Video Kit (MPF300-VIDEO-KIT-NS) includes the following components:

- A 300K LE FPGA (MPF300T, FCG1152)
- HDMI 1.4 transmitter (ADV7511) chipset and corresponding connector
- HDMI 2.0 with rail clamps, ReDrivers and corresponding connectors
- Dual camera sensor featuring IMX334 Sony image sensor
- Image sensor interface to support up to two MIPI CSI-2 cameras
- Display Serial Interface (DSI)
- NVIDIA® Jetson interface (MIPI CSI-2 TX connector)
- A High Pin Count (HPC) FMC connector to connect to high-speed interfaces (such as 12G-SDI and USXGMII)

For more information about the video kit, see PolarFire FPGA Video and Imaging Kit.

Known Issue

There is a known Reset bug where sometimes the frame buffer management comes out of Reset incorrectly. If the output video is out of sync, try power cycling the board.

Intended Audience

This demo guide is intended for:

- FPGA designers
- Firmware designers
- System level designers
- Data scientists

References

The following documents are referred in this demo guide.

- *CoreVectorBlox SDK Programmer's Guide*
- *CoreVectorBlox IP Handbook*

Table of Contents

1. Design Requirements

The following table lists the hardware and software required to run the demo.

Design Requirements	Description
Hardware Requirements	
PolarFire® Video Kit Development Board	MPF300-VIDEO-KIT-NS
USB A to mini-B cable[1]	Required for the following: • FPGA programming and SPI Flash programming • Running the modified Mi-V C code from SoftConsole
HDMI cable[1]	HDMI A Male to Male cable
Power adapter[1]	12V, 5A
HDMI monitor	A 1920 x1080 60 Hz resolution monitor for the HDMI 1.4 TX port
Host PC	A host PC with a USB port and HDMI output
Software Requirements	
Libero® System-on-Chip (SoC) v2024.2	You must install the full Libero SoC software and not just the programming tools to program the SPI Flash, which cannot be done from FPExpress. A Libero license is necessary; the video kit comes with a Gold license or an evaluation license that can be obtained from the **Licensing** tab of the following page. Libero SoC v12.0 and later
CoreVectorBlox License	To configure and synthesize the CoreVectorBlox IP, a license is required. It is available at SoC portal. Follow these steps to obtain the license: 1. Click on the **Request Free License** option available on the top right corner. 2. Click on the **Register** button for the required free license option. 3. Enter the Disk ID or MAC ID based on the selected license option and click on the **Register** button. 4. The license is send through an email. You can also download the license from the MicrochipDirect site.
SoftConsole 2021.1	SoftConsole is required to modify the application Mi-V C code. For more details, see SoftConsole.

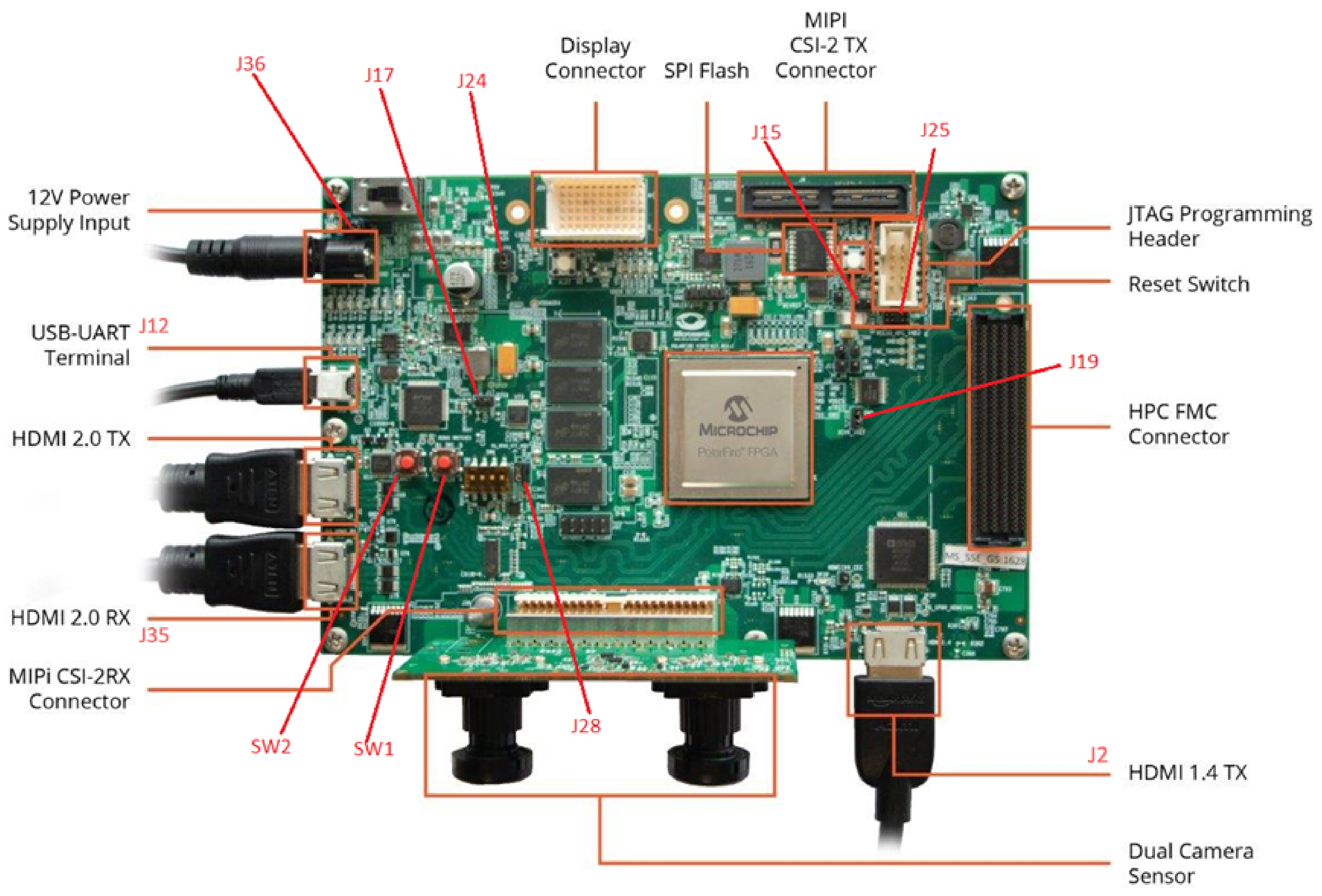
Note:

1. Included with the PolarFire Video Kit.

2. Development Kit for Demo

The following figure highlights the features of PolarFire® Video Kit.

Figure 2-1. PolarFire Video Kit Features



The following table provides the jumper position and functionality for the jumper settings.

Table 2-1. Jumper Description

Jumper/Switch	Default Position	Functionality
J15	Open	SPI Slave and Master mode selection Default: SPI master
J17	Open	100K PD for TRSTn Default: 1K PD is connected
J19	Pin1 and 2	Default: XCVR_VREF is connected to GND
J28	Pin 1 and 2	Default: Programming through the FTDI
J24	Pin 2 and 4	Default: VDDAUX4 voltage is set to 3V3
J25	Pin 5 and 6	Default: Bank4 voltage is set to 1V8
J36	Pin 1 and 2	Default: Board power-up through SW4

Table 2-1. Jumper Description (continued)

Jumper/Switch	Default Position	Functionality
SW1	—	Opens model selection menu and loads highlighted model from menu.
SW2	—	Cycles through model selection menu if open. When the face demo is running and the model menu is not open, pressing this button toggles the GenderAge network on/off.

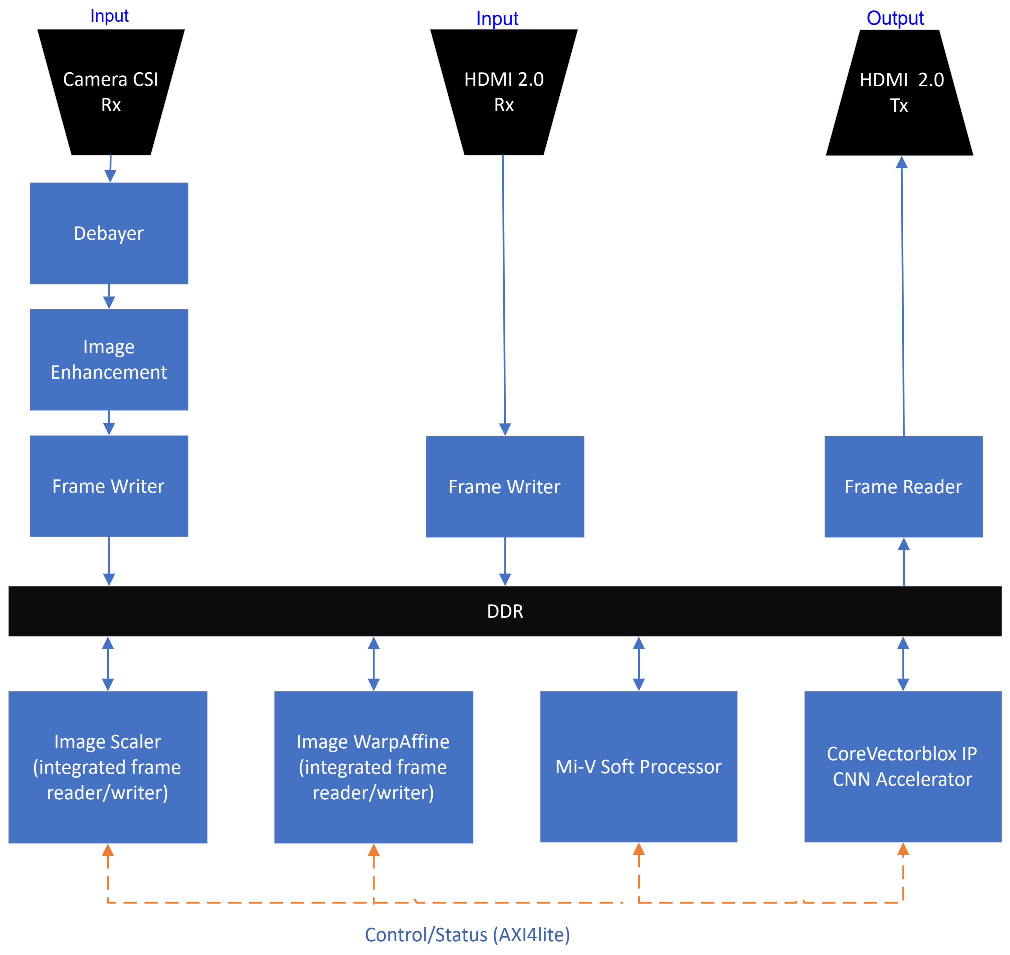
3. Demo Design Description

The following section provides an overview of the dataflow in the demo design.

3.1. System Design

The following diagram illustrates an overview of the dataflow in the design.

Figure 3-1. System Dataflow


Input
Camera CSI Rx
Input
HDMI 2.0 Rx
Output
HDMI 2.0 Tx
Debayer
Image Enhancement
Frame Writer
Frame Writer
Frame Reader
DDR
Image Scaler (integrated frame reader/writer)
Image WarpAffine (integrated frame reader/writer)
Mi-V Soft Processor
CoreVectorblox IP CNN Accelerator
Control/Status (AXI4lite)

The following sequence of data flow shown in the preceding figure:

1. Input
   a. Received from MIPI CSI 2: Video frame data is passed through a debayer and Image Enhancement block to a Frame Writer which writes the video data to DDR.
   b. Received from HDMI Rx: If a source is connected to J35, the video frame data is sent directly to a Frame Writer which writes the video data to DDR.
2. Based on demo requirements, Mi-V instructs the Image Scaler and/or the Image WarpAffine block to process the input frame appropriately for CNN(s). If an HDMI source is available, the input frame is captured from HDMI. If not, the frame is captured from the camera. The result of scaling and warping is written to DDR.
3. Mi-V instructs CoreVectorBlox IP to run the appropriate CNN using the scaled and warped image as input to CNN. The result of CNN is written back to DDR by CoreVectorBlox IP.
4. Mi-V reads the result of CNN from DDR and runs post-processing software routines. The result is then drawn on the original Input Frame Buffer.
5. The Frame Reader reads the frame and streams the frame data to the HDMI TX block.

4. Setting Up the Demo

The following steps describe how to setup the demo:

1. Setting up the Hardware
2. Programming the PolarFire® Device
3. Programming the SPI Flash

4.1. Setting Up the Hardware

Setting up the hardware involves interfacing the dual camera sensor module and the HDMI monitor with the PolarFire® Video Kit and verifying the jumper settings.

Perform the following steps.

1. Connect the J1 connector of the dual camera sensor module to the J5 interface of the video kit. The video kit is already shipped with this.
2. Connect the Full HD HDMI (1080P) monitor to J2 (HDMI 1.4 TX port) of the video kit using the HDMI cable.
3. Connect the host PC to J12 of the video kit using the USB mini cable.
4. Connect the power supply cable to J20 of the video kit.
5. Ensure that the jumper settings are set on the video kit. The video kit is shipped in this configuration. For jumper position and functionality, see Table 2-1.
6. Power-up the HDMI monitor.
7. Power-up the board using the SW4 slide switch.
8. Optionally, if opting to use an HDMI input feed, connect HDMI source playing `SampleFaces.mp4` or `SamplePlates.mp4`, respectively (for more information, see the github.com/Microchip-Vectorblox/assets/releases/tag/assets) fullscreen to the HDMI 2.0 RX port (J35).

The PolarFire dual camera video and imaging hardware setup is completed.

4.2. Programming the PolarFire Video Kit

The following section describes how to program the PolarFire® device and run the demo.

4.2.1. Creating and Opening the Project

To create the demo, follow the instructions found in the `readme.md` file on the VectorBlox website: github.com/Microchip-Vectorblox/VectorBlox-Video-Kit-Demo.

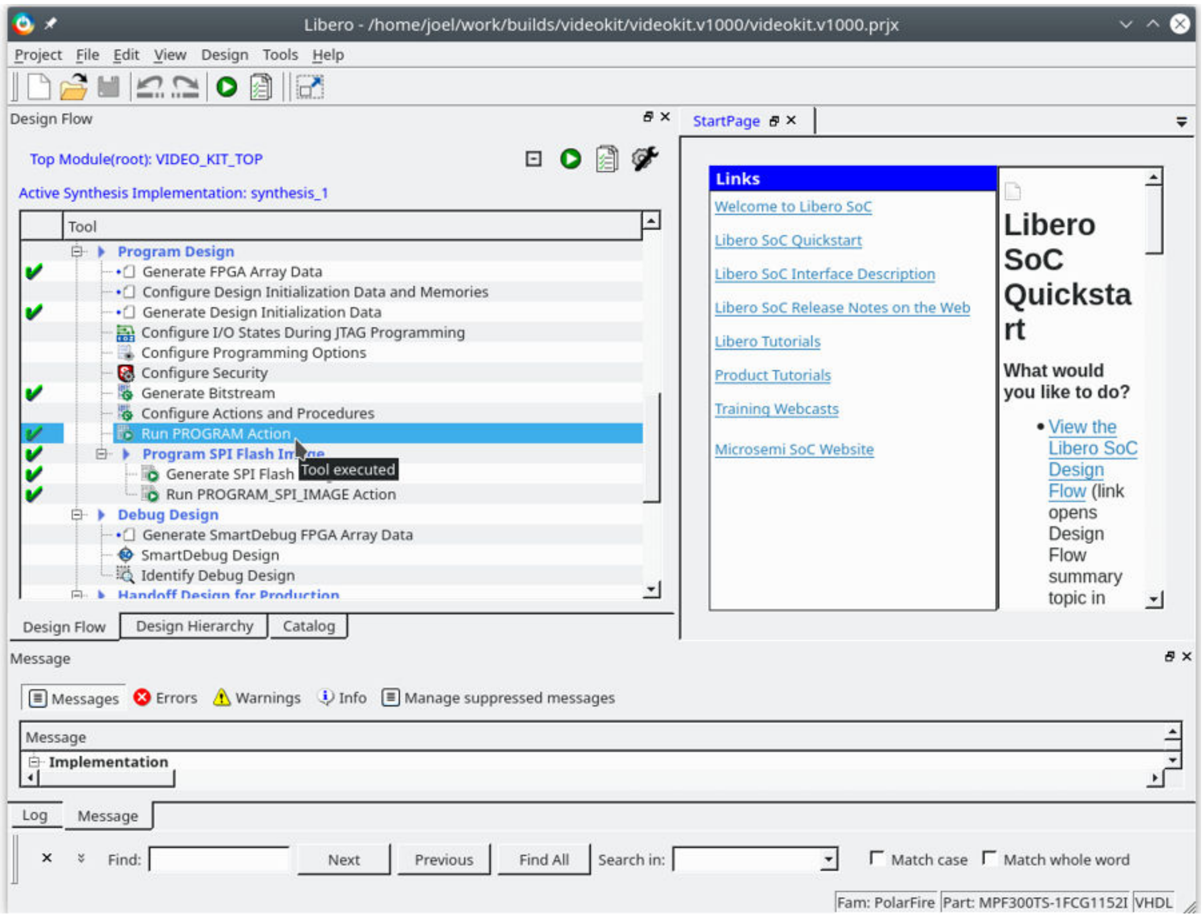
Note: You may be prompted to update Libero SoC or individual IP cores, ignore these prompts. If you are building the project from GitHub, ensure that a tutorial can be generated from the SDK. For more instructions, see github.com/Microchip-Vectorblox/VectorBlox-Video-Kit-Demo.

4.2.2. Programming the Device

Once the project is successfully generated, to program the device, perform the following steps:

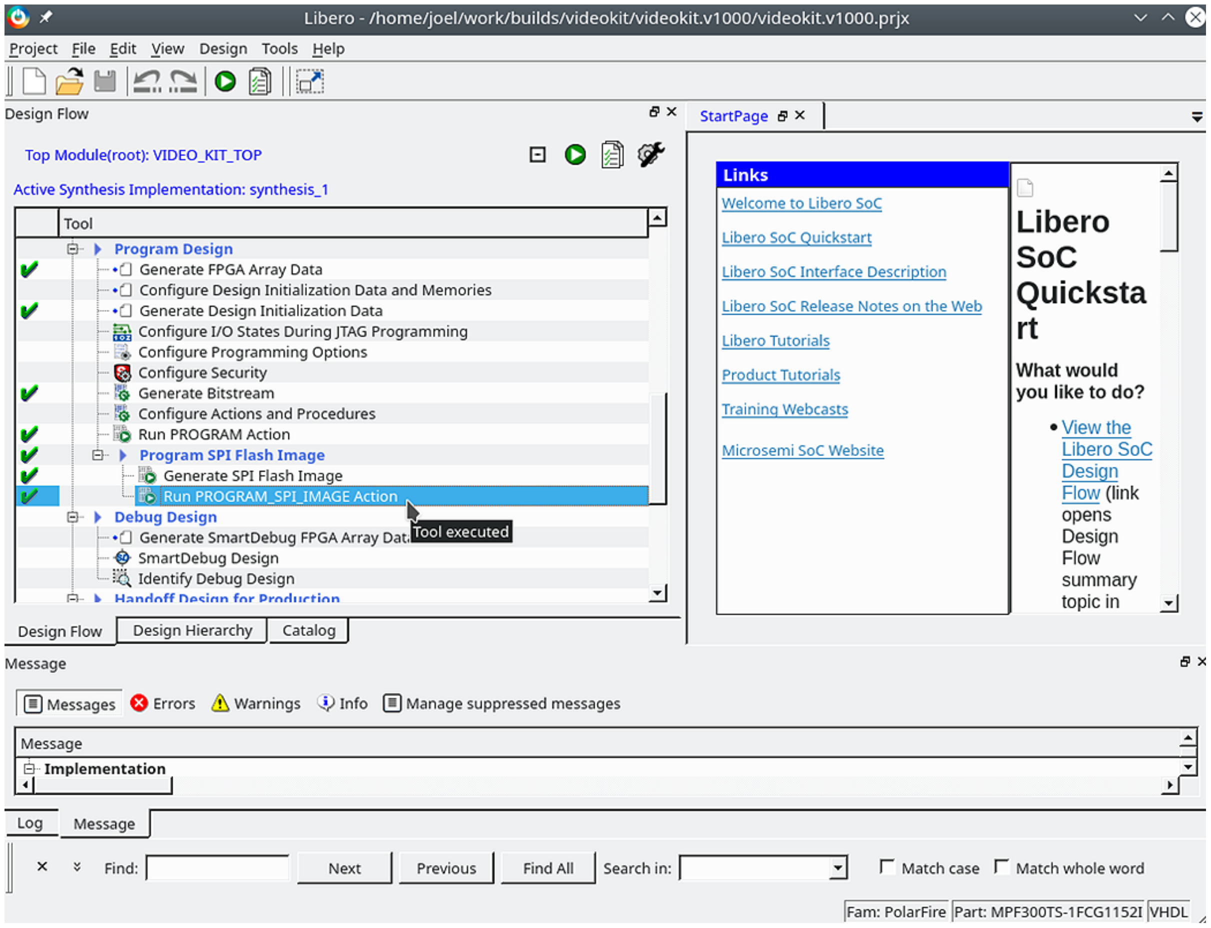
1. In the **Design Flow** window, double-click **Run PROGRAM Action**.

Figure 4-1. Run PROGRAM ACTION


Libero - /home/joel/work/builds/videokit/videokit.v1000/videokit.v1000.prjx
Project File Edit View Design Tools Help
Design Flow
Top Module(root): VIDEO_KIT_TOP
Active Synthesis Implementation: synthesis_1
Tool
Program Design
Generate FPGA Array Data
Configure Design Initialization Data and Memories
Generate Design Initialization Data
Configure I/O States During JTAG Programming
Configure Programming Options
Configure Security
Generate Bitstream
Configure Actions and Procedures
Run PROGRAM Action
Program SPI Flash Image
Generate SPI Flash
Tool executed
Run PROGRAM_SPI_IMAGE Action
Debug Design
Generate SmartDebug FPGA Array Data
SmartDebug Design
Identify Debug Design
Handoff Design for Production
Design Flow
Design Hierarchy
Catalog
StartPage
Links
Welcome to Libero SoC
Libero SoC Quickstart
Libero SoC Interface Description
Libero SoC Release Notes on the Web
Libero Tutorials
Product Tutorials
Training Webcasts
Microsemi SoC Website
Libero SoC Quickstart
What would you like to do?
View the Libero SoC Design Flow (link opens Design Flow summary topic in
Message
Messages
Errors
Warnings
Info
Manage suppressed messages
Message
Implementation
Log
Message
Find:
Next
Previous
Find All
Search in:
Match case
Match whole word
Fam: PolarFire
Part: MPF300TS-1FCG1152I
VHDL

2. Double-click **Run PROGRAM_SPI_IMAGE Action** and wait. This takes some time.
   Notes:
   - In Windows® operating system, you might be prompted with a firewall popup.
   - Ignore any warnings about misaligned sectors.

Figure 4-2. Run PROGRAM_SPI_IMAGE Action


Libero - /home/joel/work/builds/videokit/videokit.v1000/videokit.v1000.prjx
Project File Edit View Design Tools Help
Design Flow
Top Module(root): VIDEO_KIT_TOP
Active Synthesis Implementation: synthesis_1
Tool
Program Design
Generate FPGA Array Data
Configure Design Initialization Data and Memories
Generate Design Initialization Data
Configure I/O States During JTAG Programming
Configure Programming Options
Configure Security
Generate Bitstream
Configure Actions and Procedures
Run PROGRAM Action
Program SPI Flash Image
Generate SPI Flash Image
Run PROGRAM_SPI_IMAGE Action
Tool executed
Debug Design
Generate SmartDebug FPGA Array Dat
SmartDebug Design
Identify Debug Design
Handoff Design for Production
Design Flow
Design Hierarchy
Catalog
StartPage
Links
Welcome to Libero SoC
Libero SoC Quickstart
Libero SoC Interface Description
Libero SoC Release Notes on the Web
Libero Tutorials
Product Tutorials
Training Webcasts
Microsemi SoC Website
Libero SoC Quicksta rt
What would you like to do?
View the Libero SoC Design Flow (link opens Design Flow summary topic in
Message
Messages
Errors
Warnings
Info
Manage suppressed messages
Message
Implementation
Log
Message
Find:
Next
Previous
Find All
Search in:
Match case
Match whole word
Fam: PolarFire
Part: MPF300TS-1FCG1152I
VHDL

4.2.3. Running the Demo

Power cycle the board with SW4 to start the demo.

The startup takes a few minutes. The following events occur during the startup: the camera is calibrated to the brightness of the environment, the firmware and models are read from the Flash into DDR.

After the startup is completed, the demo must start with the preloaded models.

Note: Sometimes the demo might appear to start correctly without a power cycle but actually be in an Incoherent state.

You can select the source of the frames that needs to be processed by the network. The source is determined by connecting to the J35 connector. An HDMI cable connecting a video source (for example, from a laptop) takes the priority over the image sensor coming with the video kit. If the HDMI cable is not connected to J35, the image sensor is selected as the source.

4.2.4. Switching Models From Model Menu

After start-up, the currently running network can be swapped out for another network programmed in the SPI Flash. By default, the demo is programmed with the MobileNet V2 and YOLOv8n models. To switch to the other networks, perform the following steps:

1. To open the Model menu, press the SW1 push button.
2. To cycle downward through the Model menu, use the SW2 push button.
3. When highlighting the Model you desired to load, press the SW1 push button again.

After the network is loaded from the SPI Flash first time, it is not re-loaded again from the SPI Flash until the board is powercycled.

5. Running Alternate Models

By default, the demo runs a few select demos with supported postprocessing. However, it is capable of running many other networks. In this document, you can see an example of adding in a new network, Tiny YOLOv3.

Postprocessing exists for certain models such as detection, classifiers, pose estimation and segmentation. Users must write additional post-processing code to run other networks.

The following sections describe how to run the alternate models.

5.1. Obtaining Model File

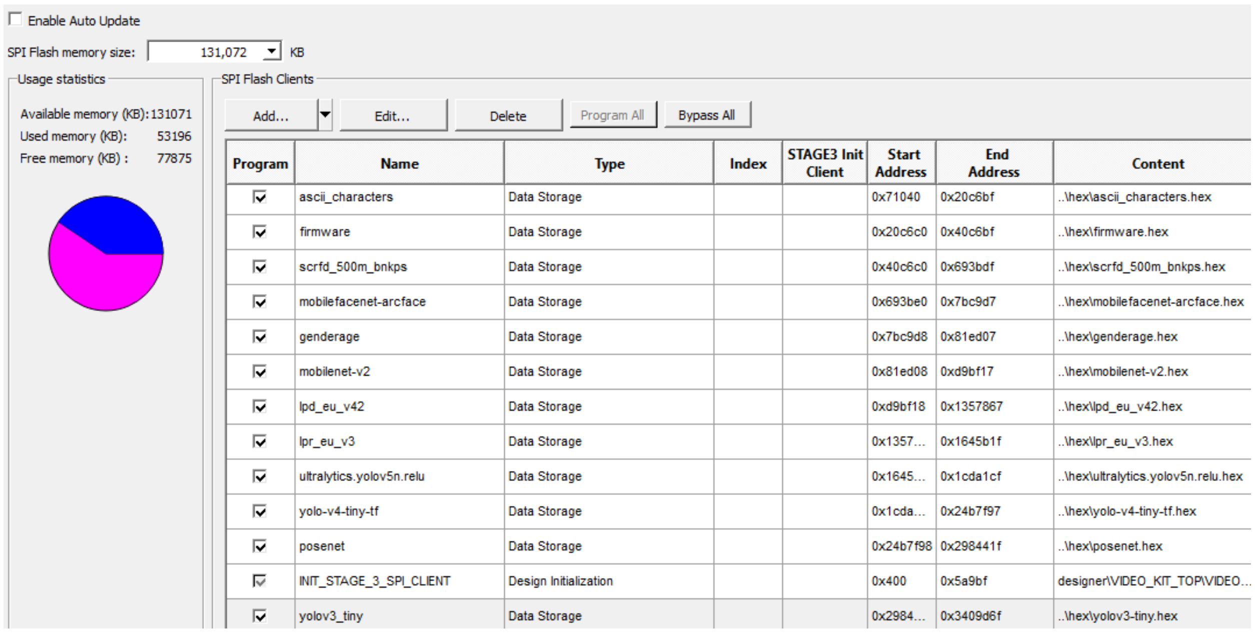
The model files can be obtained by running the tutorial available in the VectorBlox SDK in github.com/Microchip-Vectorblox/VectorBlox-SDK. Instructions for running the tutorials can be found in the Programmer's Guide available as part of the SDK documentation. Generate the model using this tutorial: `tutorials/darknet/yolov3-tiny`. The artifact generated from the tutorial that needs to be stored is `yolov3-tiny.hex`. This hex file is added to the SPI Flash on the board.

5.2. Modifying the SPI Flash Configuration

Perform the following steps in Libero.

1. Invoke the “Configure Design Initialization Data and Memories” tool in the Libero Design Flow.
2. To view all the SPI Flash clients, click the **SPI Flash** tab available at the top of the “Configure Design Initialization Data and Memories” tool.
3. To add a new model, click **Add** and select the **Add Data Storage Client** option. Change the path to point to `yolov3-tiny.hex` (file described in the preceeding section, Obtaining Model File).
   Note: Ensure the address range does not overlap with other clients in the Flash memory.
4. Click **Apply**.
5. In the **Design Flow** window, double-click **Run PROGRAM_SPI_FLASH Action** (see Step 2 in the Programming the Device section).
6. Click **Yes** to ignore warnings about memory alignment.
7. After the SPI programming is complete, power cycle the board using SW4.

Figure 5-1. SPI Configuration after Modification



Available memory (KB): 131071
Used memory (KB): 53196
Free memory (KB) : 77875

Program	Name	Type	Index	STAGE3 Init Client	Start Address	End Address	Content
✓	ascii_characters	Data Storage			0x71040	0x20c6bf	..\hex\ascii_characters.hex
✓	firmware	Data Storage			0x20c6c0	0x40c6bf	..\hex\firmware.hex
✓	scrfd_500m_bnkps	Data Storage			0x40c6c0	0x693bdf	..\hex\scrfd_500m_bnkps.hex
✓	mobilefacenet-arcface	Data Storage			0x693be0	0x7bc9d7	..\hex\mobilefacenet-arcface.hex
✓	genderage	Data Storage			0x7bc9d8	0x81ed07	..\hex\genderage.hex
✓	mobilenet-v2	Data Storage			0x81ed08	0xd9bf17	..\hex\mobilenet-v2.hex
✓	lpd_eu_v42	Data Storage			0xd9bf18	0x1357867	..\hex\lpd_eu_v42.hex
✓	lpr_eu_v3	Data Storage			0x1357...	0x1645b1f	..\hex\lpr_eu_v3.hex
✓	ultralytics.yolov5n.relu	Data Storage			0x1645...	0x1cda1cf	..\hex\ultralytics.yolov5n.relu.hex
✓	yolo-v4-tiny-tf	Data Storage			0x1cda...	0x24b7f97	..\hex\yolo-v4-tiny-tf.hex
✓	posenet	Data Storage			0x24b7f98	0x298441f	..\hex\posenet.hex
✓	INIT_STAGE_3_SPI_CLIENT	Design Initialization			0x400	0x5a9bf	designer\VIDEO_KIT_TOP\VIDEO...
✓	yolov3_tiny	Data Storage			0x2984...	0x3409d6f	..\hex\yolov3-tiny.hex

5.3. SoftConsole Project

Before the new model runs on the FPGA, the software running on Mi-V must be modified as described in this section:

1. A SoftConsole project is located in the Libero Design zip archive at `Download_Directory/vectorblox_videokit_v/softconsole`. In either SoftConsole 2022 (or 2021), open the project in your workspace.

   Figure 5-2. Open Projects in SoftConsole

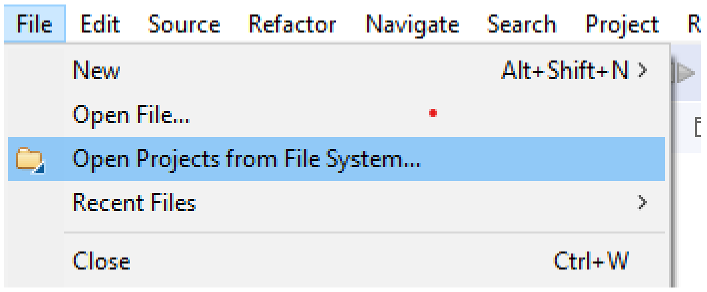
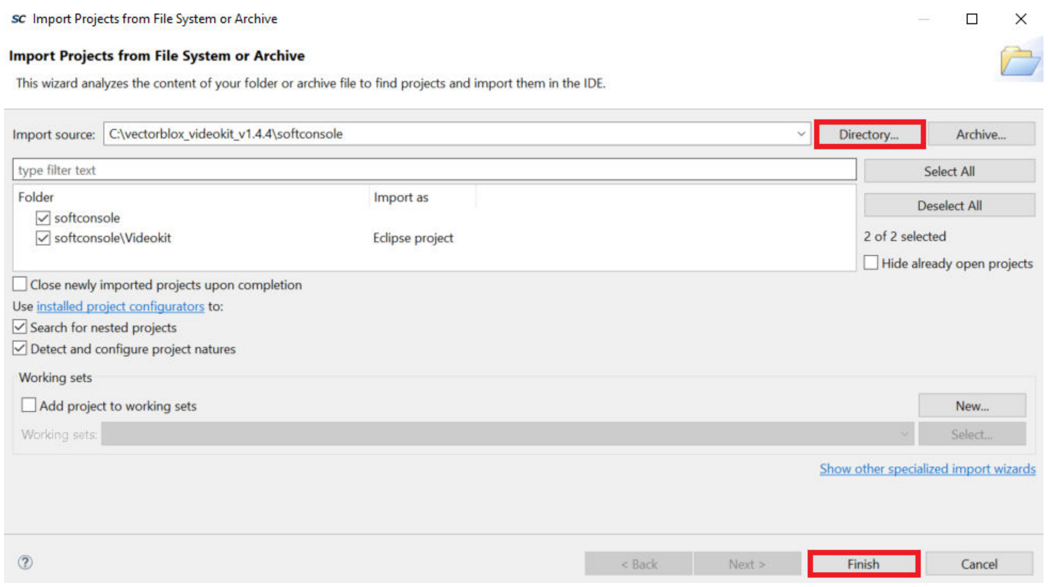
   

   Figure 5-3. Import Projects from File System or Archive

   Import Projects from File System or Archive

   Import Projects from File System or Archive

   This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.

   Import source: C:\vectorblox_videokit_v1.4.4\softconsole

Folder	Import as
☑ softconsole	
☑ softconsole\Videokit	Eclipse project

   2 of 2 selected

   ☐ Hide already open projects

   ☐ Close newly imported projects upon completion

   Use installed project configurators to:

   ☑ Search for nested projects

   ☑ Detect and configure project natures

   Working sets

   ☐ Add project to working sets

   Working sets:

   Show other specialized import wizards

2. In the VideoKit project, locate and open `main.c`. Modify the `struct model_descr_t model[]` similar to the following example.

```
struct model_descr_t models[] = {
                                {"SCRFD","",0x40c6c0, "SCRFD"},
                                {"ArcFace","",0x693be0, "ARCFACE"},
                                {"GenderAge","",0x7bc9d8, "GENDERAGE"},
                                {"LPD","",0xd9bf18, "LPD"},
                                {"LPR","",0x1357868, "LPR"},
                                {"MobileNet V2","",0x81ed08, "IMAGENET"},
                                {"Yolo V5 Nano","",0x1645b20, "YOLOV5"},
                                {"Tiny Yolo V4 COCO","",0x1cda1d0, "YOLOV4"},
                                {"Posenet","",0x24b7f98, "POSENET"},
};
```

3. Change the code to the following.

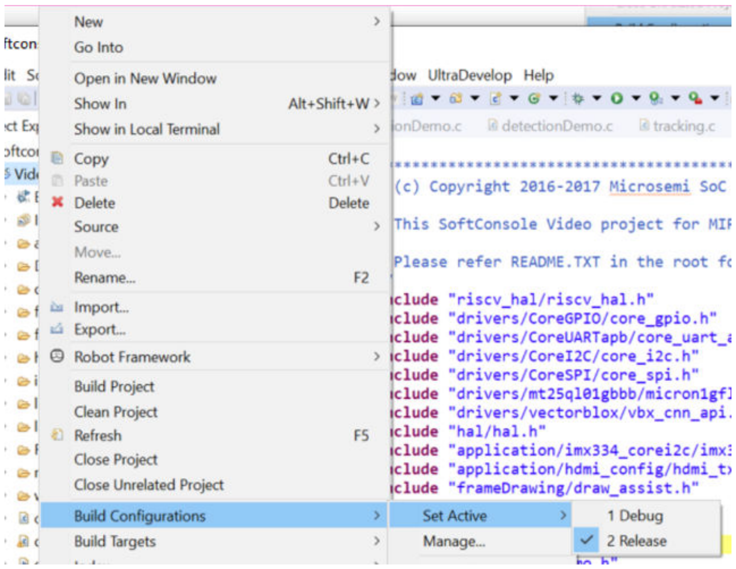
```
struct model_descr_t models[] = {
                                {"SCRFD","",0x40c6c0, "SCRFD"},
                                {"ArcFace","",0x693be0, "ARCFACE"},
                                {"GenderAge","",0x7bc9d8, "GENDERAGE"},
                                {"LPD","",0xd9bf18, "LPD"},
                                {"LPR","",0x1357868, "LPR"},
                                {"MobileNet V2","",0x81ed08, "IMAGENET"},
                                {"Yolo V5 Nano","",0x1645b20, "YOLOV5"},
                                {"Tiny Yolo V4 COCO","",0x1cda1d0, "YOLOV4"},
                                {"Posenet","",0x24b7f98, "POSENET"},
                                {"Tiny Yolo V3 COCO","",0x2984420, "YOLOV3"},
};
```

Where, parameters in the structure are as follows:

- Display name of the model
- Address in the SPI Memory in which the model is stored.
- The type of postprocessing for displaying the network. See Running Alternate Models for the supported postprocessing implementations.

4. After these modifications are performed, save the `main.c` file and build the project. The software is now ready to run and models can be executed.
5. Ensure that the Build Configurations are set to Release Mode. To do this, on VideoKit, right click.

Figure 5-4. Build Configuration

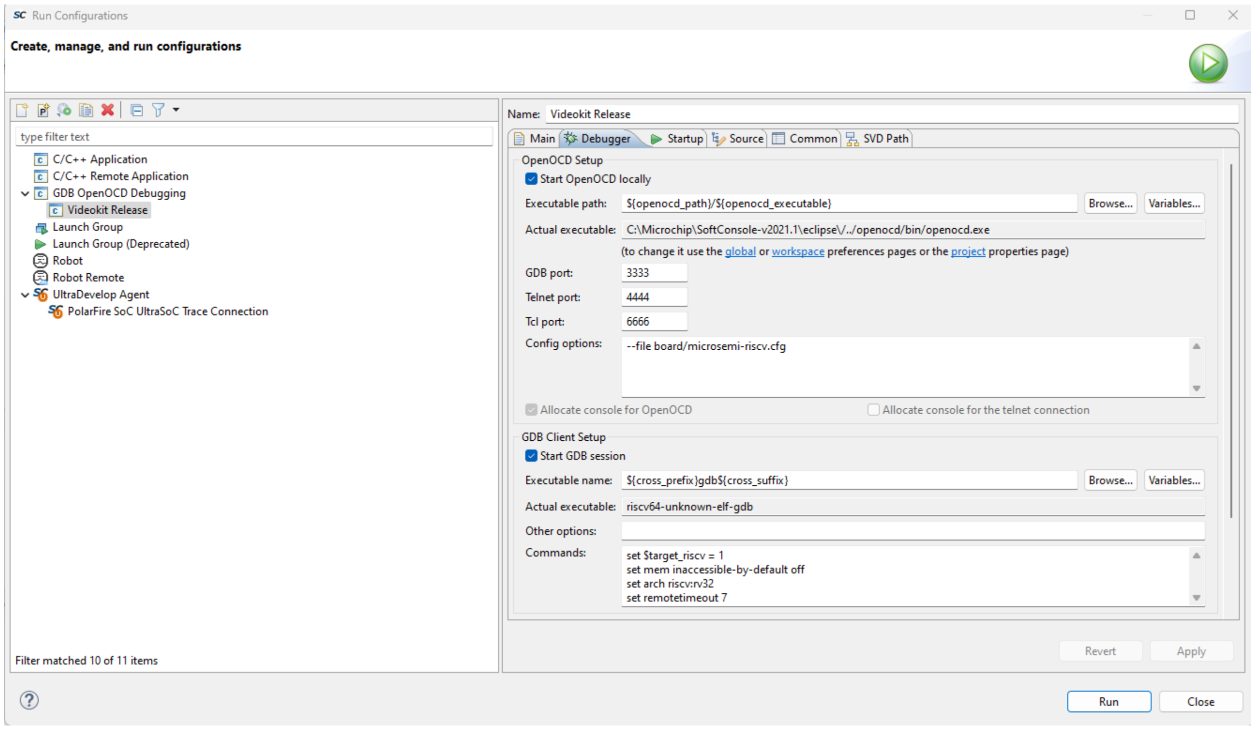


```
(c) Copyright 2016-2017 Microsemi SoC
This SoftConsole Video project for MIF
Please refer README.TXT in the root fo
iclude "riscv_hal/riscv_hal.h"
iclude "drivers/CoreGPIO/core_gpio.h"
iclude "drivers/CoreUARTapb/core_uart_a
iclude "drivers/CoreI2C/core_i2c.h"
iclude "drivers/CoreSPI/core_spi.h"
iclude "drivers/mt25ql01gbbb/micron1gfl
iclude "drivers/vectorblox/vbx_cnn_api.
iclude "hal/hal.h"
iclude "application/imx334_corei2c/imx3
iclude "application/hdmi_config/hdmi_tx
iclude "frameDrawing/draw_assist.h"
```

5.4. Running the Mi-V Program

Make sure to use FlashPro5 and jumper J28 connects Pin 1 to Pin 2. To run the Mi-V program, click the button in the toolbar. Set the Run Configuration settings as shown in the following figure.

Figure 5-5. Run Configuration



Name: Videokit Release

OpenOCD Setup

☑ Start OpenOCD locally

Executable path: ${openocd_path}/${openocd_executable}

Actual executable: C:\Microchip\SoftConsole-v2021.1\eclipse\/../openocd/bin/openocd.exe

(to change it use the global or workspace preferences pages or the project properties page)

GDB port: 3333

Telnet port: 4444

Tcl port: 6666

Config options: --file board/microsemi-riscv.cfg

☑ Allocate console for OpenOCD ☐ Allocate console for the telnet connection

GDB Client Setup

☑ Start GDB session

Executable name: ${cross_prefix}gdb${cross_suffix}

Actual executable: riscv64-unknown-elf-gdb

Other options:

Commands:

```
set $target_riscv = 1
set mem inaccessible-by-default off
set arch riscv:rv32
set remotetimeout 7
```

5.5. Flashing New Software

Alternatively, to avoid having to manually run the Mi-V program on every start-up if making changes to the project in SoftConsole, the Fabric RAM can be configured to load your changes automatically.

If the build succeeds, perform the following steps:

1. Verify the timestamp of `softconsole/Videokit/Release/Videokit.hex` is up to date.
2. In Libero, click **Configure Design Initialization Data and Memories**.
3. Click the **Fabric RAMs** tab.
4. Click edit on the entry that uses `softconsole/Videokit/Release/Videokit.hex`. This entry is named `PROC_SUBSYSTEM_0/PF_SRAM_AHBL_AXI_C1_0`.
5. Select your modified `Videokit.hex` file, click **OK**, then click **Apply** tab.
6. To update the required steps, double-click **Run PROGRAM_SPI_IMAGE Action**.

6. Error Debugging

If an error is encountered, both of the relevant error message and error code are written out over the HDMI output display and to UART. For more information and to proceed with next steps, consult the CoreVectorBlox IP Handbook.

7. Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

Revision	Date	Description
K	08/2025	Updated to 2.0.3 Release
J	06/2024	Updated to 1.4.5 Release
H	09/2023	Updated to 1.4.4 Release
G	04/2023	Updated to 1.4.3 Release
F	09/2022	Updated to 1.4 Release
E	02/2022	Updated to 1.3 Release
D	09/2021	Updated to 1.2 Release
C	05/2021	Updated to 1.1 Release
B	11/2020	Updated to 1.0 Release
A	08/2020	Initial Revision

Microchip Information

Trademarks

ISBN: 979-8-3371-1814-7

Legal Notice

Microchip Devices Code Protection Feature