PolarFire Ethernet Sensor Bridge Quickstart Guide

Introduction

The PolarFire® Ethernet Sensor Bridge is part of Nvidia's Holoscan Ecosystem and extends multi-protocol signal conversion to NVIDIA® Jetson™ Orin™ AGX and IGX developer kits via Ethernet.

The sensor bridge is based on Microchip's power-efficient PolarFire Field-Programmable Gate Array (FPGA), MPF200T-FCG784. It has two 10G SFP+ Ethernet ports that connect to the Jetson AGX Orin and IGX developer kits and two MIPI CSI-2 receive ports for connecting cameras. The included FMC slot provides expansion options for protocols like Scalable Low-Voltage Signaling with Embedded Clock (SLVS-EC), CoaXPress, JESD 204B, Serial Digital Interface (SDI), and so on. The sensor bridge also has DDR4 for frame buffering and SPI Flash for enabling field upgrades.

The following table lists the Ethernet Sensor Bridge (ESB) kit contents.

Table 1. Kit Contents

Quantity	Description
1	PolarFire® ESB board
1	12.3 MP IMX477M Camera Module from Arducam Part Number: B0466R
1	10 Gb SFP+ to RJ45 Module Part Number: SFP-10G-T-S
1	10G Ethernet Cable
1	12V/5A AC Adapter
1	12V Power Cord
1	Type-C USB cable
1	QuickStart Card

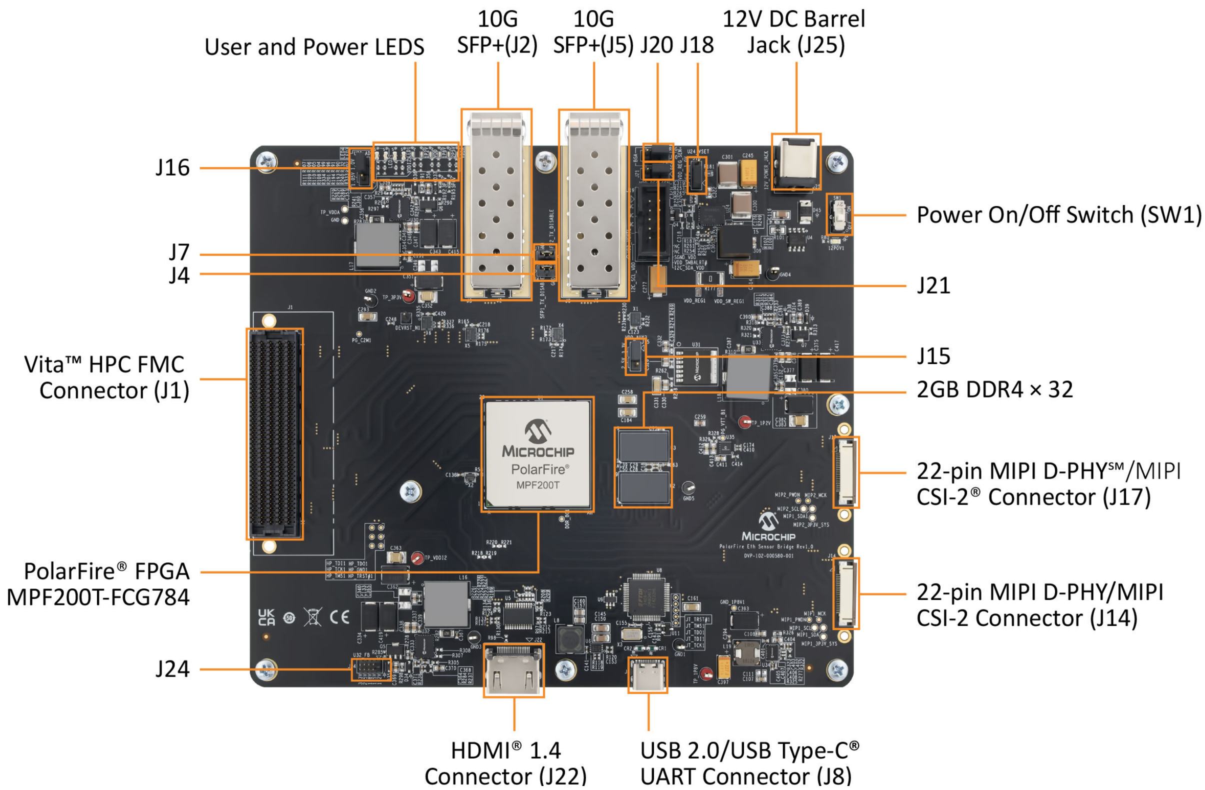
The following figure shows the PolarFire ESB Kit contents.

Figure 1. PolarFire® Ethernet Sensor Bridge Kit Contents

1. Hardware Features

The following figure shows the board components.

Figure 1-1. Board Components



2. Demo Requirements

Table 2-1. Pre-requisites for the Demo

Requirement	Description
Hardware and Accessories	
PolarFire® ESB	MPF200-ETH-SENSOR-BRIDGE
NVIDIA® Jetson AGX Orin™ Developer Kit[1]	A MIPI CSI-2 camera is attached to the sensor bridge and connected to the AGX Orin Devkit through Ethernet. This kit has to be bought separately.
One MIPI CSI-2 camera module	IMX477-based Arducam camera module is included in the kit
One 10G Ethernet cable	Included in the kit
SFP+ to RJ45 converter	Included in the kit
12V/5A power supply	Included in the kit
Monitor with DisplayPort input	Display for the AGX Orin Devkit
Keyboard and mouse	Required to configure the AGX Orin Devkit.

Note: The Quick Start Guide provides setup instructions for use with a Jetson Orin AGX Developer Kit. If you are using an IGX Developer Kit, follow the specific steps intended for the IGX kit. We highlight the sections where instructions differ for each kit.

3. Running the Demo

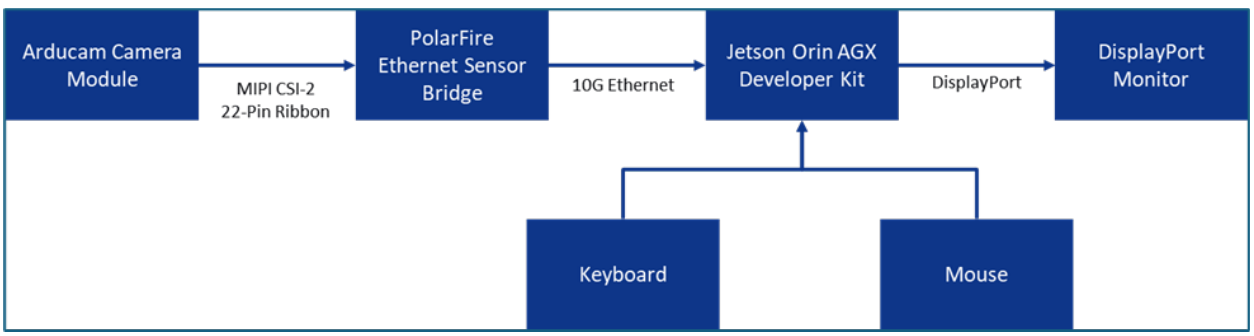
Important:
1. Ensure that the AGX Orin system is running on the latest Jetpack release (see Jetson AGX Orin Developer Kit Setup Host Setup).
2. Update the FPGA bitstream to the newest version—see Programming the Board for instructions.

The scope of this quick start guide is to get you set up and run a single MIPI CSI-2 camera streaming video via 10G Ethernet to the NVIDIA Jetson AGX Orin Developer Kit, which connects to a monitor through DisplayPort.

The PolarFire ESB is pre-programmed to support two IMX477 MIPI CSI-2 cameras from Arducam. However, only one Camera module is provided in the box.

The following figure shows the functional block diagram.

Figure 3-1. Functional Block Diagram



3.1. Setting up the Demonstration

The following table list the setup summary.

Steps	What	Description
Step 1	PolarFire® ESB	Steps covering connecting the image sensor to the sensor bridge and ethernet cable between sensor bridge and the AGX Orin devkit.
Step 2	AGX Orin Devkit setup	Steps covering AGX Orin devkit setup, updating packages and doing a ping test on the sensor bridge.
Step 3	Running examples	Running examples.

3.2. Setting up the PolarFire ESB

The following table lists the jumpers and their default position, ensure that the jumpers in the ESB are set correctly.

Table 3-1. Jumper setting for ESB

Jumper	Default Position
J4	Closed
J7	Closed
J18	Close pins 2 and 3

Table 3-1. Jumper setting for ESB (continued)

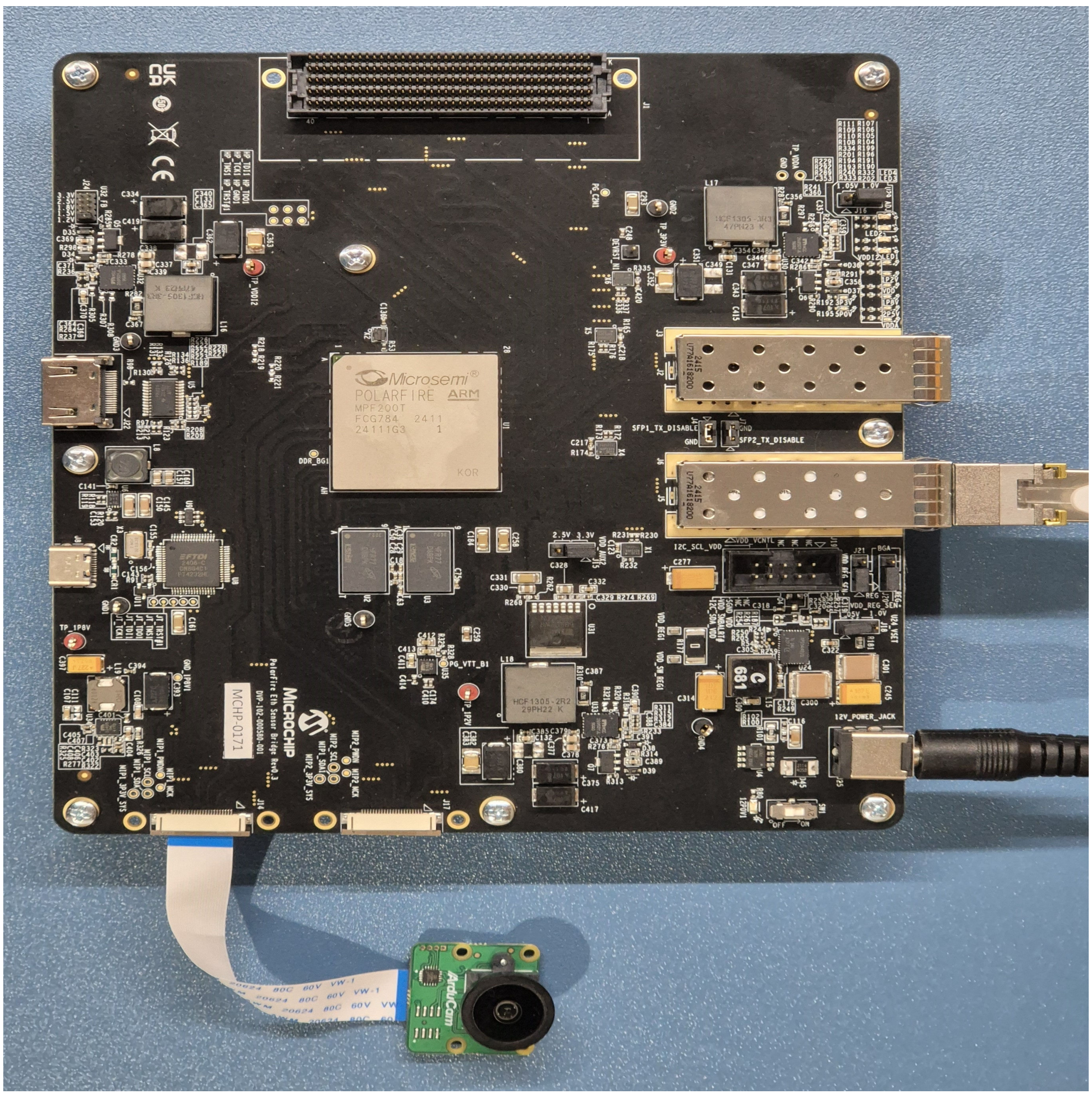
Jumper	Default Position
J21	Close pins 2 and 3
J15	Close pins 1 and 2 (3.3V)
J20	Close pins 2 and 3
J16	Close pins 2 and 3
J24	Close pins 9 and 10 (3.3V)

3.3. Camera Setup

To set up the camera, perform the following steps:

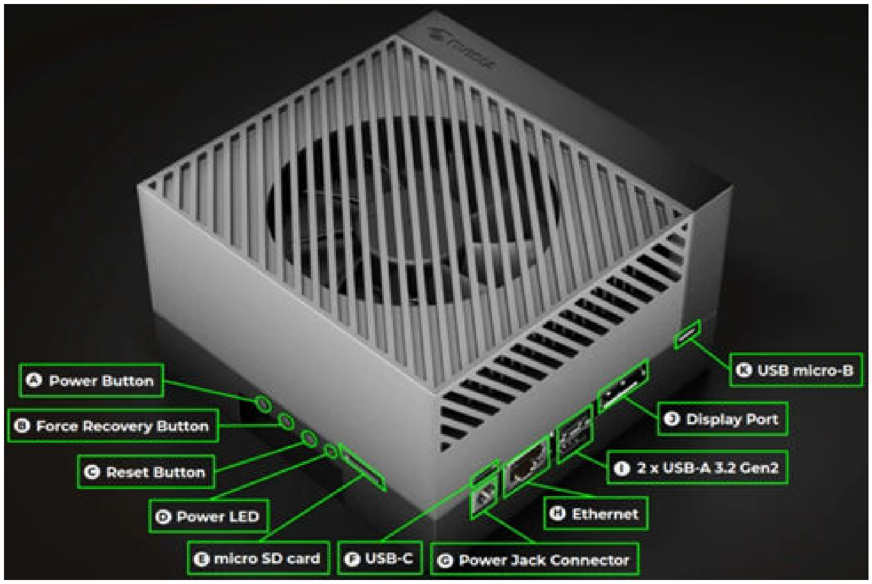
1. Make sure the MPF200-ETH-SENSOR-BRIDGE board is powered OFF.
2. Connect the IMX477 Camera module to J14 MIPI connector using the 22-pin to 22-pin camera cable, as shown in the following figure.

Figure 3-2. Connecting the Image Sensor to the MIPI CSI-2 Port of the Sensor Bridge



3. Insert SFP+ to RJ45 converter into the SFP cage on J5.
4. Connect the Ethernet cable from the SFP+ RJ45 port to the Ethernet port on NVIDIA Jetson AGX Orin Developer Kit, as shown in the following figure.

Figure 3-3. Jetson AGX Orin Developer Kit Callout



5. Connect the 12V power adapter to the J25 power input port.
6. To turn on the board, slide the switch SW1 to the ON position.

3.4. Setting up the AGX Orin Developer Kit

To set up the AGX Orin developer kit, perform the following steps:

1. Run the steps in the Getting Started with Jetson AGX Orin Developer Kit.
2. While on the getting started page, choose the "**Default setup flow**" instead of the "**Optional Setup Flow**" and when scrolling down, choose "**Initial setup with display attached**" instead of the "**Initial setup in a headless configuration**".
   Note: This step can take a long time. Make sure you have a stable internet connection.

3.5. Jetson AGX Orin Developer Kit Setup Host Setup

The PolarFire sensor bridge is supported on AGX Orin systems running Jetpack (**JP6.0 release 2**). In this configuration, the on-board Ethernet controller is used with the Linux kernel network stack for data I/O; all network I/O is performed by the CPU without network acceleration.

After the PolarFire Ethernet Sensor Bridge board is set up, configure a few prerequisites in your host system. While sensor bridge applications run in a container, these commands are all to be executed outside the container, on the host system directly. These configurations are remembered across power cycles and therefore only need to be set up once.

1. Install git-lfs.
   Some data files in the sensor bridge source repository use GIT LFS.

```
sudo apt-get update
sudo apt-get install -y git-lfs
```

2. Grant your user permission to the docker subsystem:

```
$ sudo usermod -aG docker $USER
```

   Reboot the computer to activate this setting.

Demos and examples in this package assume a sensor bridge device is connected to eth0, which is the RJ45 connector on the AGX Orin.

3. Linux sockets require a larger network receiver buffer.
   Most sensor bridge self-tests use Linux's loopback interface; if the kernel starts dropping packets due to out-of-buffer space then these tests will fail.

```
echo 'net.core.rmem_max = 31326208' | sudo tee /etc/sysctl.d/52-hololink-rmem_max.conf
sudo sysctl -p /etc/sysctl.d/52-hololink-rmem_max.conf
```

4. Configure eth0 for a static IP address of `192.168.0.101`.
   L4T uses NetworkManager to configure interfaces; by default, interfaces are configured as DHCP clients. Use the following command to update the IP address to **192.168.0.101**. For more information about configuring your system, see **Holoscan sensor bridge IP address configuration** (If you cannot use the `192.168.0.0/24` network in this way).

```
sudo nmcli con add con-name hololink-eth0 ifname eth0 type ethernet ip4 192.168.0.101/24
sudo nmcli connection up hololink-eth0
```

   Apply power to the sensor bridge device, ensure that it's properly connected, then ping **192.168.0.2** to check connectivity.

5. For the Linux socket-based examples, isolating a processor core from Linux kernel is recommended. For high bandwidth applications, like 4k video acquisition, isolation of the network receiver core is required. When an example program runs with processor affinity set to that isolated core, performance is improved, and latency is reduced. By default, sensor bridge software runs the time-critical background network receiver process on the third processor core. If that core is isolated from Linux scheduling, no processes will be scheduled on that core without an explicit request from the user, and reliability and performance is greatly improved.
   Isolating that core from Linux can be achieved by editing `/boot/extlinux/extlinux.conf`. Add the setting **isolcpus=2** to the end of the line that starts with **APPEND**. Your file should look like something like this:

```
TIMEOUT 30
DEFAULT primary

MENU TITLE L4T boot options

LABEL primary
      MENU LABEL primary kernel
      LINUX /boot/Image
      FDT /boot/dtb/kernel_tegra234-p3701-0000-p3737-0000.dtb
      INITRD /boot/initrd
      APPEND ${cbootargs} root=/dev/mmcblk0p1 rw rootwait ...<other-settings>... isolcpus=2
```

   Sensor bridge applications can run the network receiver process on another core by setting the environment variable HOLOLINK_AFFINITY to the core it should run on. For example, to run on the first processor core,

```
HOLOLINK_AFFINITY=0 python3 examples/linux_imx477_player.py
```

   Setting **HOLOLINK_AFFINITY** to blank will skip any core affinity settings in the sensor bridge code.

6. Run the "jetson_clocks" tool on startup, to set the core clocks to their maximum.

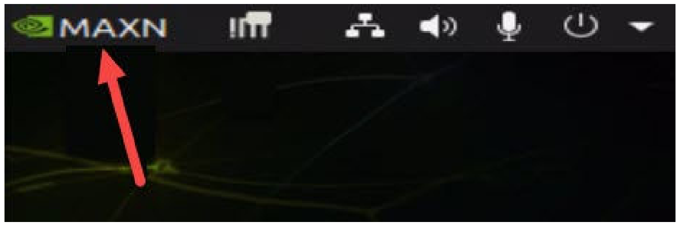
```
JETSON_CLOCKS_SERVICE=/etc/systemd/system/jetson_clocks.service
cat <<EOF | sudo tee $JETSON_CLOCKS_SERVICE >/dev/null
[Unit]
Description=Jetson Clocks Startup
After=nvpmodel.service

[Service]
Type=oneshot
ExecStart=/usr/bin/jetson_clocks

[Install]
```

```
WantedBy=multi-user.target
EOF
sudo chmod u+x $JETSON_CLOCKS_SERVICE
sudo systemctl enable jetson_clocks.service
```

7. Set the AGX Orin power mode to ‘MAXN’ for optimal performance, as shown in the following figure. The setting can be changed via L4T power drop down setting found on the upper left corner of the screen:



8. Restart the AGX Orin. This allows core isolation and performance settings to take effect. If configuring for ‘MAXN’ performance doesn’t request that you reset the unit, then execute the reboot command manually:

```
reboot
```

9. Log in to Nvidia GPU Cloud (NGC) with your developer account:
   a. If you don’t have a developer account for NGC please register at NVIDIA.
   b. Create an API key for your account, thru API Key.
   c. Use your API key to log in to nvcr.io:

```
$ docker login nvcr.io
Username: $oauthtoken
Password: <Your token key to NGC>
WARNING! Your password will be stored unencrypted in /home/<user>/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

3.6. Build and Test the Sensor Bridge Demo Container

Holoscan sensor bridge host software includes instructions for building a demo container. This container is used to run all holoscan tests and examples.

1. Fetch the sensor bridge source code from GitHub.

```
$ git clone https://github.com/nvidia-holoscan/holoscan-sensor-bridge
```

2. Build the sensor bridge demonstration container. For systems with iGPU,

```
$ cd holoscan-sensor-bridge
$ sh docker/build.sh --igpu
```

Note: igpu is appropriate for systems running on a system with iGPU (e.g. AGX or IGX without a dGPU). This requires an OS installed with iGPU support (For example: for AGX: JetPack 6.0, and for IGX: IGX OS with iGPU configuration).

3.7. Run Tests in the Demo Container

To run the sensor bridge demonstration container, from a terminal in the GUI,

```
xhost +
sh docker/demo.sh
```

This brings you to a shell prompt inside the Holoscan sensor bridge demo container.
Note: iGPU configurations, when starting the demo container it will display the message “**Failed to detect NVIDIA driver version**”: this can be ignored.

Now you're ready to run sensor bridge applications.

3.8. Sensor Bridge Software Loopback Tests

Sensor bridge host software includes a test fixture that runs in loopback mode, where no sensor bridge equipment is necessary. This test works by generating UDP messages and sending them over the Linux loopback interface.

In the shell in the demo container:

```
pytest
```

Note: The test fixture intentionally introduces errors into the software stack. If pytest indicates that all tests have passed, any error messages published by individual tests can be ignored.

4. Programming the Board

The board can be programmed in two ways.

Programming Method	Equipment Required	Approximate Programming Time
Programming using Holoscan Sensor Bridge Software	• Nvidia Jetson Orin platform with Holoscan SDK	~ 50–55 minutes
Programming using FlashPro Express	• USB-C Cable • Host PC with FlashPro Express	~2–3 minutes

4.1. Programming using Holoscan Sensor Bridge Software

The FPGA bit-version from the factory is compatible with the Holoscan Ethernet Sensor Bridge SDK v2407.

To update the FPGA bit-version to the latest version, perform the following steps:

1. Connect the Ethernet cable to J6 connector on MPF200-ETH-SENSOR-BRIDGE to AGX/IGX Ethernet port.
   Note: Perform a ping test to Check if Nvidia AGX/IGX is connected to Holoscan board by issuing the following command.
   `ping 192.168.0.2`
2. In new terminal. Issue the following command to change directory to `hololink-sensor-bridge` folder.
   `cd <PATH/TO/hololink-sensor-bridge>`
3. Run the following commands to transfer `.spi`(job/RTL design) file to the on board SPI Flash of the MPF200-ETH-SENSOR-BRIDGE board.
   `xhost + <enter>`
   a. The following command runs holoscan-sensor-bridge docker container.
      `sh docker/demo.sh <enter>`
   b. The following command flashes (Transfers to on board SPI flash) 2412 version bit file.
      `polarfire_esb flash --fpga-bit-version 2412`
   c. The following command flashes (Transfers to on board SPI flash) 2412 version bit file.
      `polarfire_esb --force flash --fpga-bit-version 2412`

   Notes:
   a. When FPGA is running the older version of bit file (like 2407), use command switch `--force` (as shown in 3c).
   b. Step 3.b(or 3.c) downloads FPGA design file from internet and takes around 50 minutes to transfer the design file to on-board flash.
4. To program the FPGA with the `.spi` that is in the on-board SPI flash, run the following command within the holoscan-sensor-bridge docker container.
   a. The following command programs FPGA with design file in SPI Flash. It takes around one minute to program the FPGA
      `polarfire_esb --program`
   b. The following command programs FPGA with design file in SPI Flash. It takes around one minute to program the FPGA.
      `polarfire_esb --force --program`

Notes:

a. When FPGA is running the older version of bit file (like 2407), use command switch `--force` .

b. If (new/latest) Holoscan Ethernet sensor bridge software is not unable to detect Ethernet packets, use the new `--force` switch . This situation arise if FPGA is running older bit file and NVIDIA AGX/IGX running the newer Holoscan sensor bridge software.

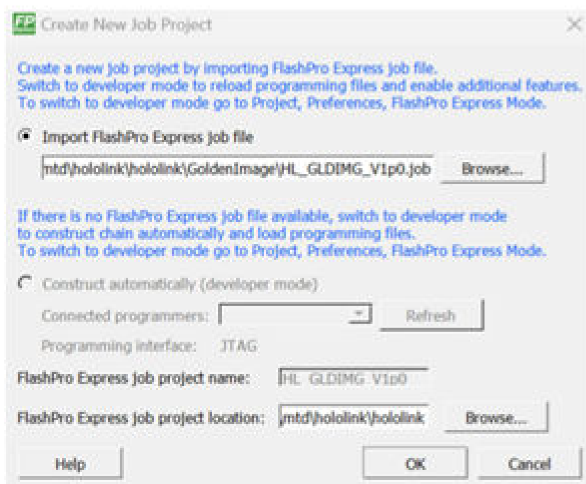
4.2. Programming using FlashPro Express

This is a faster method to program the board. This section describes how to program the PolarFire® device using a .job file included in the design files by utilizing the FlashPro Express software. The `mpf_an5522_v2024p2_jb` file is available at the following location: `<download_folder>\mpf_an5522_v2024p2_jb.zip`.

To program the PolarFire device with the `.job` file, perform the following steps:

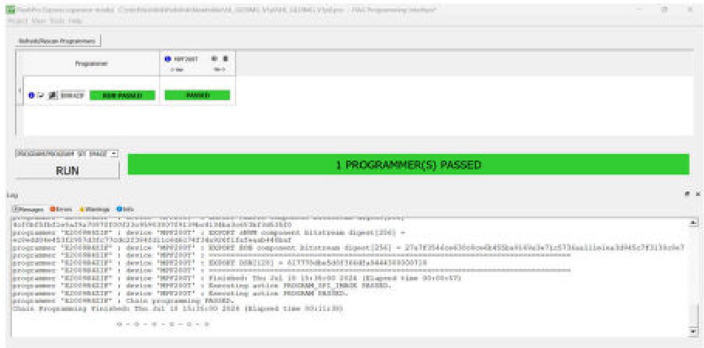
1. Connect the USB Type-C cable from Host PC to **J8** of MPF200-ETH-SENSOR-BRIDGE.
2. On the Host PC, start the FlashPro Express software from its installation directory.
3. To create a new job project on the Project menu, click **New** or **New Job Project from FlashPro Express Job**.
4. In the **New Job Project from FlashPro Express Job** dialog box, perform the following steps:
   a. **Programming job file:** Click **Browse**, navigate to the location where the `mpf_an5522_v2024p2_jb` file is located, and select the file.
   b. **FlashPro Express job project location:** Click **Browse** and navigate to the location where you want to save the project.

Figure 4-1. New Job Project from FlashPro Express Job



5. Click **OK**. The required programming file is selected and ready to be programmed in the device. The FlashPro Express window appears.
6. Verify that a programmer number appears in the **Programmer** box. If it does not, verify the board connections and click **Refresh/Rescan Programmers**.
7. To program the device, click **Run**. When the device is programmed successfully, a "Run" status is displayed, as shown in the following figure.

Figure 4-2. FlashPro Express—RUN PASSED



8. To Close FlashPro Express, click **Project > Exit**. The PolarFire device is programmed.

5. Running Examples

Two examples are described in this section

1. Streaming camera video
2. Running a pose estimation demo

5.1. Streaming the Video on AGX Developer Kit

This demonstration shows the output of the IMX477 camera module on the monitor connected via displayport. To run the high-speed video player with IMX477 setup, perform the following steps:

1. Open a new terminal and navigate to the `holoscan-sensor-bridge` folder by using the following command.

   ```
   cd <PATH/TO/holoscan-sensor-bridge>
   ```

2. To run the sensor bridge demonstration container, from a terminal in the GUI
   Note: Ignore the step if the docker is running already

   ```
   xhost +
   sh docker/demo.sh
   ```

   It runs the holoscan-sensor-bridge docker container.

3. Set up the camera and run the high-speed video player (Holoviz) with live video using the following command:

   ```
   python examples/linux_imx477_player.py
   ```

4. To close the Holoviz application and exit the docker,

   ```
   exit
   ```

5.2. Running Pose Estimation on GPU

To run this example, perform the following steps:

1. Open a new terminal and navigate to the `holoscan-sensor-bridge` folder by using the following command.

   ```
   cd <PATH/TO/holoscan-sensor-bridge>
   ```

2. The next step involves downloading `ffmpeg` and `ultralytics` packages to run the pose estimation demo from Running Holoscan Sensor Bridge examples - NVIDIA Docs. Rather than going to the link above, type the following in the console

   ```
   apt-get update && apt-get install -y ffmpeg
   pip3 install ultralytics onnx
   cd examples
   yolo export model=yolov8n-pose.pt format=onnx
   cd -
   ```

   Note:
   This conversion step only needs to be executed once; the `yolov8n-pose.onnx` file contains the converted model and is all that's needed for the demo to run. The installed components will be forgotten when the container is exited; those do not need to be present in future runs of the demo.

3. To run the sensor bridge demonstration container, from a terminal in the GUI,
   Note: Ignore the step if the docker is running already

   ```
   xhost +
   sh docker/demo.sh
   ```

4. To run the pose estimation demo,

```
python examples/linux_imx477_pose_estimation.py
```

5. Close the Holoviz application and exit the docker to terminate the application

6. Documentation Resources

For more information on the PolarFire ESB, including schematics and user's guides, see the MPF200-Eth-sensor-bridge.

Microchip FPGA Support

Microchip FPGA products group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

- From North America, call **800.262.1060**
- From the rest of the world, call **650.318.4460**
- Fax, from anywhere in the world, **650.318.8044**

Microchip Information

Trademarks

ISBN: 979-8-3371-1544-3

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.