XeroHID

Introduction

I am a mentor for the team Error Code Xero (1425) out of Wilsonville Oregon. Every year we build an "OI" that is used by the gunner. This OI is based on the TI Launchpad board as this is really the only choice for building this OI devices. The TI Launchpad can be used in one of three configurations. But none of these three configurations were a great fit for what we wanted. The biggest issue was we were not interested in the analog inputs and were more interested in the largest number of digital inputs and output. Specifically, we wanted at least 20 digital inputs and at least 8 digital outputs. None of the TI Launchpad configurations met this need.

This led to the creation of XeroHID. XeroHID is a USB based HID device with an on-board bootloader to provide the required HID functionality for an FRC OI. XeroHID provides …

- 6 analog inputs
- 24 digital inputs
- 16 digital outputs
- A bootloader and Windows based bootload utility to update the firmware

XeroHID is based on the Infineon CY8CPROTO-062-4343W development kit. This development kit was chosen because I work for Infineon and not only have easy access to the development kit, but I also understand the device and the development tools very well. In addition, this development kit was designed so that the peripherals that are usually available on microcontroller development kits can be removed providing almost unfettered access to many device I/Os.

This kit was also chosen because it is inexpensive. These can be found from multiple vendors for about $30 per kit.

Using XeroHID

Step 1: Get the XeroHID Firmware and Software

Run the clone command below to retrieve the XeroHID firmware and software. This retrieves whatis required to run the XeroHID firmware as well as everything needed to customize XeroHID for your own purposes. If the command below does not make sense, talk to someone on your software subteam. I generally do all of this on a Windows machine and use Cygwin for running git, but any git solution should work.

```
git clone https://github.com/sjcbulldog/frchid2.git
```

Step 2: Download and Install the Cypress Programmer

Navigate to the Infineon developer center (https://softwaretools.infineon.com/welcome) and select tools in the top bar. In the "Filter results" text box, type "Cypress Programmer", click the download link, and select Windows (x64). Download and install the "Cypress Programmer" tool. Note, you will be required to have an account on the Infineon site to download the Cypress Programmer.

Step 3: Connect the Development Kit

Attach the CY8CPROTO-062-4343W development kit to your computer using the provided cable. Note, the development kit has two USB connectors. You want to use the port shown in the picture below.

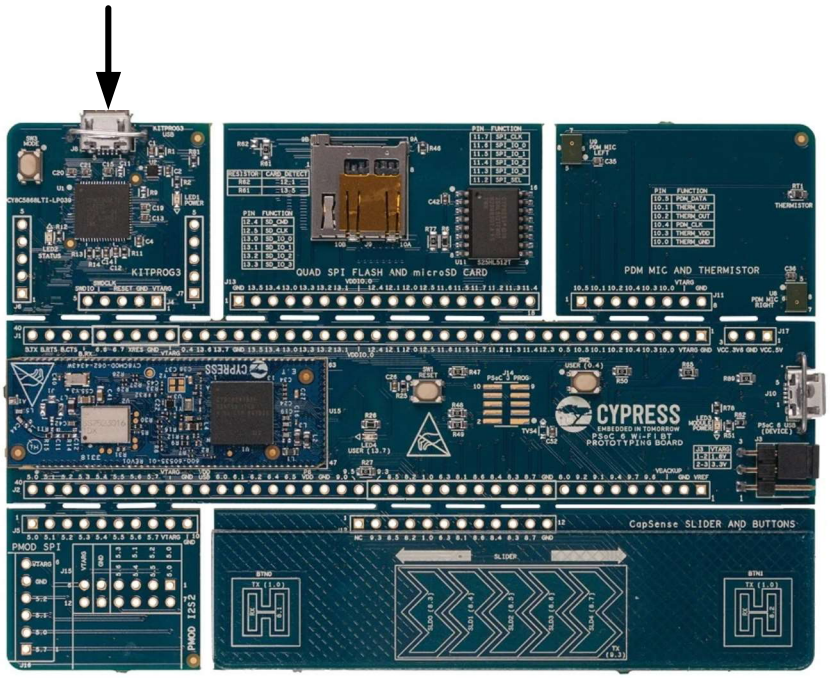


Figure 1: "MiniProg" USB Port Location

Step 4: Program The Firmware

Start the Cypress Programmer and follow these steps:

- In the Probe/Kit field select the entry that starts with CY8CPROTO-062-4343W
- In the “File” line under Program Settings, push the “…” button on the right and navigate to where you cloned the contents from github. Select the file named ‘release/bootloader_cm0p.hex’.
- Press the ‘Connect’ button at the top and wait until the programmer is connected to the development kit.
- Press the ‘Program’ button and wait until you see ‘Device programmed successfully’ in the bottom status bar. There will be text describing what is happening in the log window.
- Press the ‘Disconnect’ button at the top.
- In the “File” line under Program Settings, push the “…” button on the right and navigate to where you cloned the contents from github. Select the file named ‘release/frchid.hex’.
- Press the ‘Connect’ button at the top and wait until the programmer is connected to the development kit.
- Press the ‘Program’ button and wait until you see ‘Device programmed successfully’ in the bottom status bar. There will be text describing what is happening in the log window.

Step 5: Verify The Firmware

Move the USB cable from the programming port on the development port to the device port on the development kit. See the picture below.

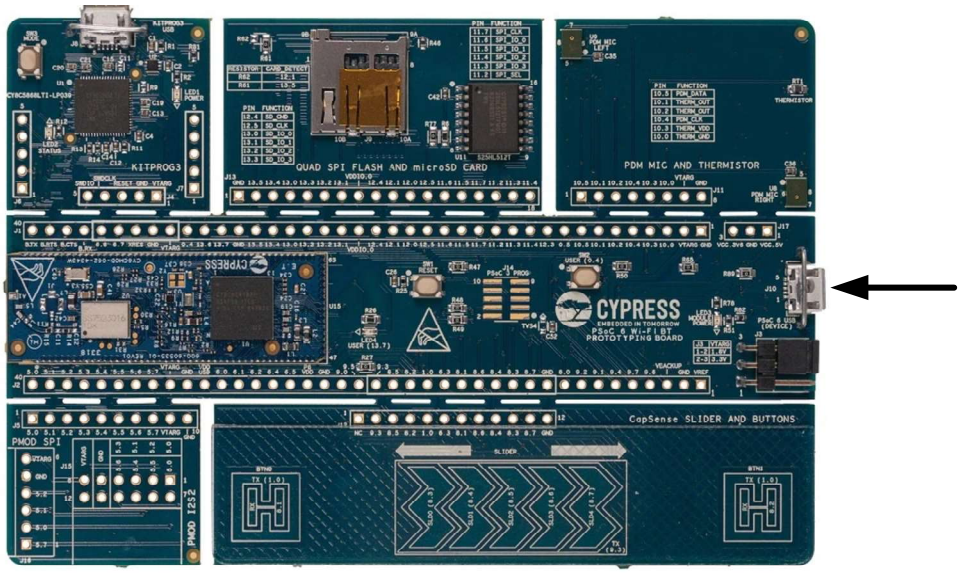


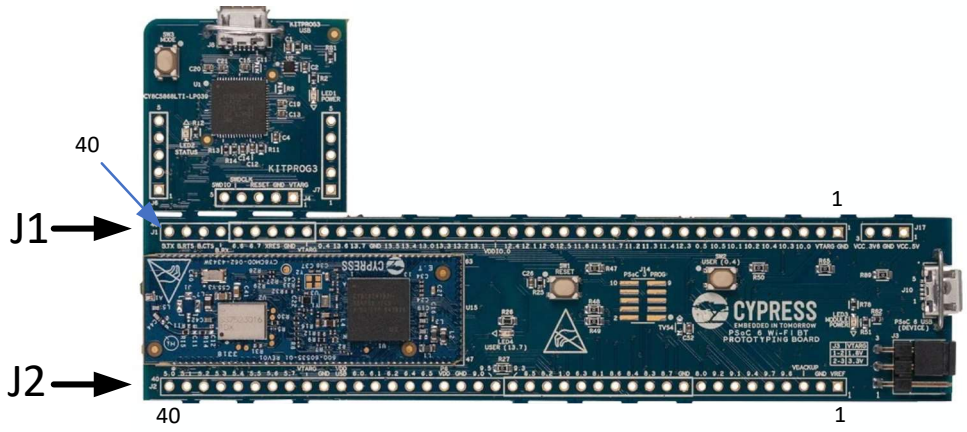
Figure 2: Device USB Port Location

Start the FRC Driver Station and on the left side select the USB button. You should see the PSoC 6 Panel Device in the FRC Driver Station. If not, you might need to press the Rescan button.

Note, if the RED LED is blinking, this indicates an error. A one second blink cycle indicates a hardware error initializing the digital IOS or the ADC channels. A longer three second blink cycle indicates an issue with USB initialization.

Step 6: Construct Your OI Gunners Panel

As seen in the picture above, the board has the unique feature that many of the attached peripherals can be snapped off. This makes the I/O pins available for purposes other than those intended when using the complete board. Carefully snap off all of the side boards except the one shown in the picture below. Note, this is the "MiniProg3" board which provides an interface between the computer and the programming and debug features of the device on the main board. This board can be snapped off eventually as well because the XeroHID application contains a bootloader, but for now leave it attached in case something goes wrong.



From here using this board is just like using the TI Launchpad. The only difference is knowing where various signals connect. This table below shows the connections available for the various signals and how these are referenced in the FRC robot software.

Device Pin	Connector	FRC Type	FRC Index
P10_0	J1-3	Joystick	1
P10_1	J1-7	Joystick	2
P10_2	J1-6	Joystick	3
P10_3	J1-4	Joystick	4
P10_4	J1-5	Joystick	5
P10_5	J1-8	Joystick	6
P0_5	J1-9	Button	1
P1_0	J2-17	Button	2
P11_2	J1-13	Button	3
P11_3	J1-12	Button	4
P11_4	J1-11	Button	5
P11_5	J1-15	Button	6

P11_6	J1-16	Button	7
P11_7	J1-14	Button	8
P12_0	J1-18	Button	9
P12_1	J1-19	Button	10
P12_3	J1-10	Button	11
P12_4	J1-20	Button	12
P12_5	J1-17	Button	13
P13_0	J1-25	Button	14
P13_1	J1-22	Button	15
P13_2	J1-23	Button	16
P13_3	J1-24	Button	17
P13_4	J1-26	Button	18
P13_5	J1-27	Button	19
P13_6	J1-30	Button	20
P5_4	J2-36	Button	21
P5_5	J2-35	Button	22
P5_6	J2-34	Button	23
P5_7	J2-33	Button	24
P6_2	J2-27	Output	1
P6_3	J2-16	Output	2
P8_1	J2-15	Output	3
P8_2	J2-18	Output	4
P8_3	J2-12	Output	5
P8_4	J2-13	Output	6
P8_5	J2-19	Output	7
P8_6	J2-14	Output	8
P8_7	J2-11	Output	9
P9_0	J2-22	Output	10
P9_1	J2-7	Output	11
P9_2	J2-8	Output	12
P9_3	J2-20	Output	13
P9_4	J2-6	Output	14
P9_5	J2-21	Output	15
P9_6	J2-4	Output	16

Debugging XeroHID

The steps above should be straight forward and provide for a working HID device. If something is not working, having access to the output of the firmware is very useful. As long as the “MiniProg3” USB port is connected to the computer, there is also USB Serial Port that can be used to see messages from the firmware.

Using the Window Device Manager, look for the “KitProg3 USB-UART (COMn)” entry where the letter ‘n’ is replaced with a number. This is the number of the Windows COM port. Use your favorite Windows serial terminal (e.g. Putty) and select the given serial port (e.g. COMn). Set the speed to 115200, 8 data bits, 1 stop bit, and no parity. Now you should be able to see any messages from the firmware.

Note, both “MiniProg3” USB port and the device USB port can be connected concurrently to the computer and in fact this is the common configuration while developing the XeroHID software and firmware.

You should first see sample output from the bootloader like shown below.

```
[INF] MCUBoot Bootloader Started
[INF] Primary image: magic=unset, swap_type=0x1, copy_done=0x3, image_ok=0x3
[INF] Scratch: magic=unset, swap_type=0x1, copy_done=0x3, image_ok=0x3
[INF] Boot source: primary slot
[INF] boot_swap_type_multi: Primary image: magic=unset, swap_type=0x1, copy_done=0x3, image_ok=0x3
[INF] boot_swap_type_multi: Secondary image: magic=unset, swap_type=0x1, copy_done=0x3, image_ok=0x3
[INF] Swap type: none
[INF] User Application validated successfully
[INF] Starting User Application (wait)...
[INF] Start slot Address: 0x10018400
[INF] MCUBoot Bootloader finished
[INF] Deinitializing hardware...
```

This will be followed by output from the FRCHID firmware which will clear the screen first if you are using a terminal program that understand clear screen ANSI sequences. This means the above bootloader text may disappear quickly.

Then the following text should appear.

```
FRCHID: firmware version 1.0.0
FRCHID: bsp/retarget io initialization completed
FRCHID: upgrade is 0
FRCHID: hardware initialization completed
0:000 USBD_Start
FRCHID: USBHID initialization completed
```

If instead you see the following text, you were pressing the user button on the board while the board was booting. This text means you are in bootloader mode. In this mode, just press the reset button to return to the FRCHID mode.

```
FRCHID: firmware version 1.0.0
FRCHID: bsp/retarget io initialization completed
FRCHID: upgrade is 0
FRCHID: firmware upgrade mode
0:000 USBD_Start
FRCHID: info: firmware upgrade CDC device configured
```

Finally, if you see the following text, the bootloader was programmed but the FRCHID firmware was not. Got back to step 4 above and repeat the steps to program the firmware ensuring that both the ‘bootloader_cm0p.hex’ and ‘frchid.hex’ are both programmed.

```
[INF] MCUBoot Bootloader Started
[INF] Primary image: magic=unset, swap_type=0x1, copy_done=0x3, image_ok=0x3
[INF] Scratch: magic=unset, swap_type=0x1, copy_done=0x3, image_ok=0x3
[INF] Boot source: primary slot
[INF] boot_swap_type_multi: Primary image: magic=unset, swap_type=0x1, copy_done=0x3, image_ok=0x3
[INF] boot_swap_type_multi: Secondary image: magic=unset, swap_type=0x1, copy_done=0x3, image_ok=0x3
[INF] Swap type: none
[ERR] MCUBoot Bootloader found none of bootable images
```

Building XeroHID

The XeroHID consists of a single ModusToolbox application with two projects and a Windows bootloader program. The Windows bootloader program relies on Qt.

ModusToolbox can be installed by following the links provided above for Cypress Programmer but searching for ModusToolbox instead of Cypress Programmer. Multiple packages will show up so pick the package that is labeled “ModusToolbox Tools Package”. This software was developed using version 3.1 but any later version should work.

The bootloader application requires Visual Studio 2022 Community Edition which can be installed from this link (https://visualstudio.microsoft.com/vs/community/).

The bootloader application requires Qt can be installed from this link (https://www.qt.io/). The bootloader was developed against Qt 6.3.1, but any Qt 6.x version should work.

Firmware

It is beyond the scope of this document to walk through installation and usage of ModusToolbox. There are many resources on these topics available through the ModusToolbox dashboard that is available when ModusToolbox is installed. While ModusToolbox is built to run in many different environments, my environment of choice ModusToolbox plus Visual Studio Code plus the Visual Studio Code extension. The text below assumes this configuration.

This section talks about portions of the Visual Studio Code screen. This website (https://code.visualstudio.com/docs/getstarted/userinterface) shows an image of the screen and what the various pieces are called.

Start Visual Studio code and navigate to the 'fw' directory under the location where you previously performed the git clone in step 1 above. Note in the bottom right corner status bar in Visual Studio Code it should tell you that ModusToolbox is running a series of steps to be sure the application is ready for Visual Studio Code. This includes running 'make getlibs' and 'make vscode'. Wait until you see the prompt 'ModusToolbox: Ready' before you proceed.

Building the bootloader project:

Select the menu item 'Terminal/Run Build Task'
Select 'Build bootloader_cm0p'

Building the FRCHID project:

Select the menu item 'Terminal/Run Build Task'
Select 'Build frchid'

Building the application (both firmware projects):

Select the menu item 'Terminal/Run Build Task'
Select 'Build fw'

Programming the Complete Application

Select the debug icon[1] in the Activity Bar on the left side of the Visual Studio Code window.
In the dropdown at the top of the Side Bar select 'Program Application (KitProg3_MiniProg4)'
Press the green arrow next to the dropdown menu.

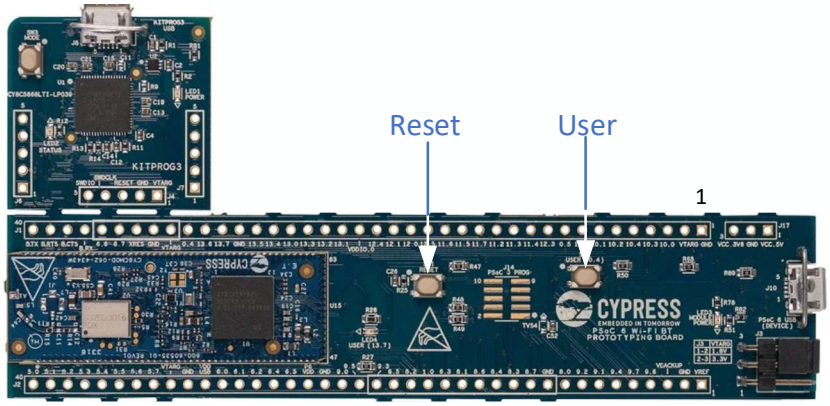
Windows Bootloader

To use the bootloader program, you must build it. This requires Visual Studio 2022 and Qt release 6.3.1 or more recent. Once both are installed, the bootloader is located in the loader directory. The Visual Studio solution file, frchidloader.sln, should be loaded and the bootloader can be built in Visual Studio.

[1] If you are unsure, hover over the icons until you see the icon that says 'Run and Debug'

Running the bootloader takes two arguments. The first is the name of the serial port that is running the bootloader. The second argument is the name of the image to program.

You enter the bootloader mode by pressing and holding both buttons. Then release the reset button and continue to hold the User button until the red user LED on the board lights and remains lit. Note, any blinking pattern of the LED indicates an initialization error. See Step 5 above for more details.



Bootloadable Firmware

TBD – instructions for building a new firmware image to be bootloaded