CAERUS User Manual

Last revision: 13 May 2019

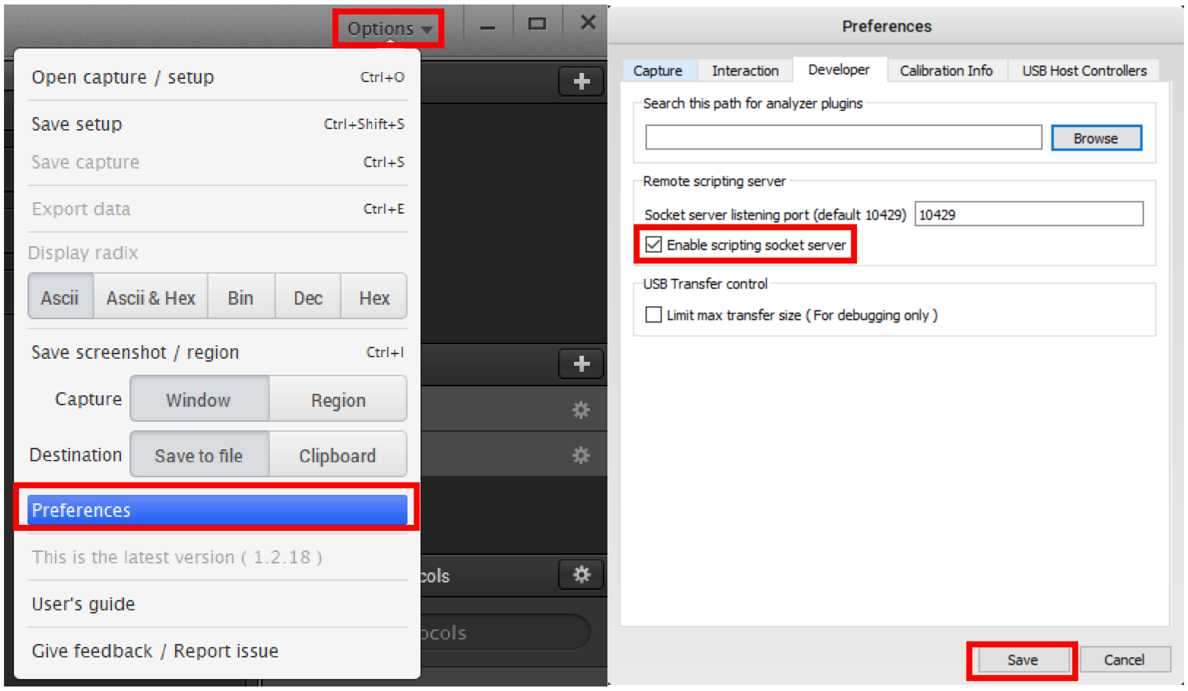
Installing CAERUS Software

I. Pre-requisite software

- Windows (recommended)
- Python 3 (https://www.python.org/downloads/)
    - Check "Add Python to PATH" in the installer
    - Ensure Python's pip package manager is installed (enabled by default under "Customize Installation"
- Saleae Logic Software (https://www.saleae.com/downloads/)
- Git (optional)

II. Installing CAERUS

1. Clone the CAERUS controller software repository using **git**, or download a zipped archive from the repository
2. Navigate to the top-level of the controller software repository
3. Run the command `pip install -r requirements.txt` to install the required Python libraries



Socket server listening port (default 10429) 10429

☑ Enable scripting socket server

☐ Limit max transfer size (For debugging only)

Figure 1: Enabling Saleae Logic Software scripting socket server

4. Open the Saleae Logic Software
5. Open the "Options" menu, select "Preferences", select the "Developer" tab, and check the "Enable scripting socket server" check box, shown in Figure 1: Enabling Saleae Logic Software scripting socket server

6. Click the “Save” button

Using CAERUS

I. Hardware Setup

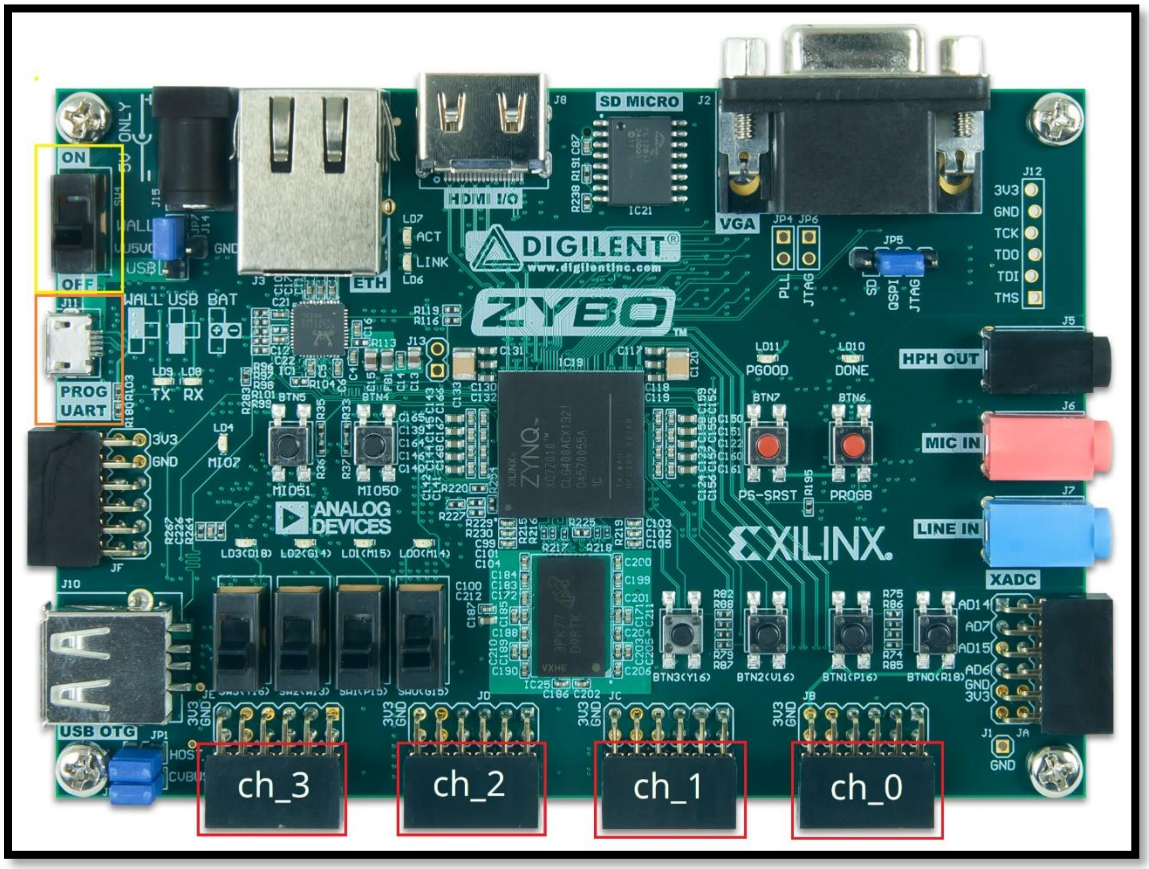


Figure 2: Zybo Board

1. Ensure the board power switch is set to **ON** position (Switch outlined in Yellow in Figure 2: Zybo Board)
2. Connect **micro side** of the USB-A to USB-micro cable (Port outlined in Orange in Figure 2: Zybo Board)
3. Connect **USB-A side** of the USB-A to USB-micro to an available USB port on PC

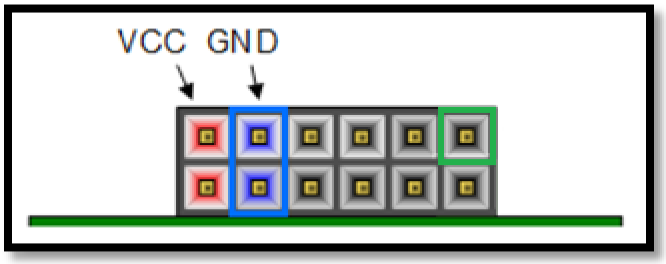


*Figure 3: Zybo Pmod Port Interface (Outlined in **Red** in Figure 2: Zybo Board)*

4. CAERUS produces signals on the pin outlined in **Green** in Figure 3: Zybo Pmod Port Interface (Outlined in **Red** in Figure 2: Zybo Board) for each channel in Figure 2: Zybo Board; this pin should be used to connect to target device inputs.
5. Connect the GND pin outlined in **Blue** in Figure 3: Zybo Pmod Port Interface (Outlined in **Red** in Figure 2: Zybo Board) to the target device for a common ground
6. Connect the Saleae Logic Analyzer via USB to an available USB port on PC
7. When connecting the Saleae Logic Analyzer to a target device, ensure a ground pin is connected

II. Software Startup

1. Open the Saleae Logic Software
2. Ensure the Zybo board has been programmed. See section titled "Ephemerally Programming the Zybo board"
3. Open the Controller Software
   - Open the Windows command prompt or other terminal emulator
   - Navigate to the top-level of the controller software repository
   - Run with the command `python -m controller [saved profile]`

III. Software Usage

Tips & Tricks

- If the software crashes, it will attempt to save a profile in the current directory named `_crash_[current date].profile`. This profile can be loaded to resume work.
- When defining a signal, it uses the global sample rate setting. Set the global sample rate setting with the `sample` command before defining any signals; it defaults to 25 MHz in a new profile, which is too fast for most signals.

Errata

This section describes bugs and work-arounds to avoid them.

- A bug in the HDL implementation currently results in a hang when any channel other than **I1** is selected as the "stop channel", or the channel whose address is observed as the condition to stop playback. This results in the software waiting forever to playback to finish. To avoid this, ensure the test set-up always has a non-looping signal with the shortest-duration signal on **I1**.
- A bug exists where using sample rates below 200 Hz usually results in incorrect playback. Use higher sample rates (> 300 Hz?) to avoid this.

Ephemerally Programming the Zybo board

I. Pre-requisite software

- Vivado Design Suite (https://www.xilinx.com/support/download.html)
- Git

II. Hardware Setup

1. Connect **micro side** of the USB-A to USB-micro cable (Port outlined in **Orange** in Figure 2: Zybo Board)
2. Connect **USB-A side** of the USB-A to USB-micro to an available USB port on PC
3. Set the Zybo board power switch to the **ON** position (Switch outlined in **Yellow** in Figure 2: Zybo Board)

III. Loading Firmware and FPGA Bitstream

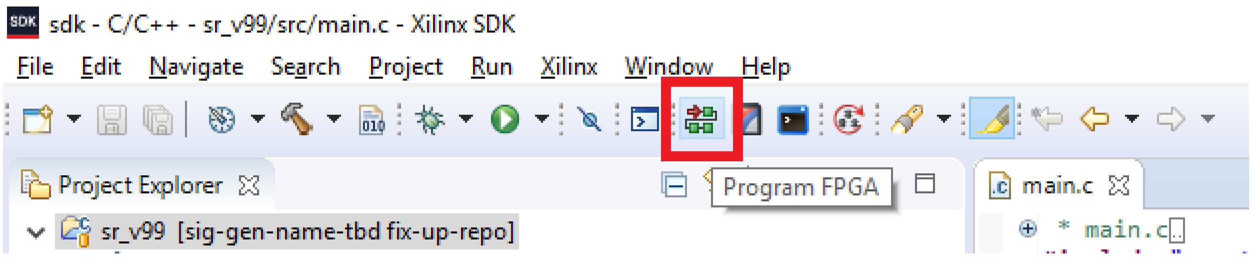

sdk - C/C++ - sr_v99/src/main.c - Xilinx SDK
File Edit Navigate Search Project Run Xilinx Window Help
Project Explorer
Program FPGA
main.c
sr_v99 [sig-gen-name-tbd fix-up-repo]
main.c

Figure 4: Program FPGA Button

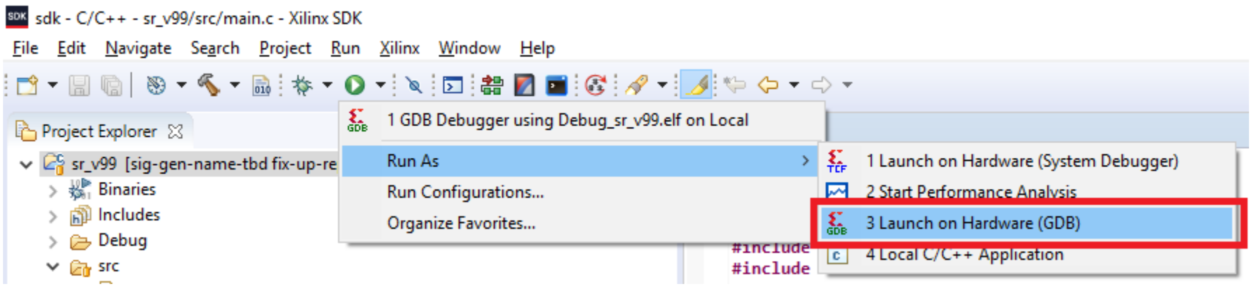

sdk - C/C++ - sr_v99/src/main.c - Xilinx SDK
File Edit Navigate Search Project Run Xilinx Window Help
Project Explorer
sr_v99 [sig-gen-name-tbd fix-up-re
Binaries
Includes
Debug
src
1 GDB Debugger using Debug_sr_v99.elf on Local
Run As
Run Configurations...
Organize Favorites...
1 Launch on Hardware (System Debugger)
2 Start Performance Analysis
3 Launch on Hardware (GDB)
4 Local C/C++ Application

Figure 5: Launch on Hardware Button

1. Load the bitstream to the FPGA with the "Program FPGA" button in the Xilinx SDK, shown in Figure 1.
2. Open the `main.c` file, and ensure its window is selected.
3. Load the Firmware with the “Run As…”, “Launch on Hardware (GDB)” cascaded menu button, shown in Figure 2.
4. This loads the firmware and FPGA bitstream ephemerally; it needs to be performed again each time the FPGA is disconnected from power. (For loading the firmware and booting from an SD card, refer to **section 4-1 of the Configuration *and Booting*** document located in the **User Manual folder** of the project USB drive.)

Example Usage: Minimum Button Duration Detection

I. Required Parts

- PIC16F887 DIP package
- 2× 0.1 µF ceramic capacitor
- 5V power source
- PICKit 3

II. Required Software

- MPLABX IDE 5.5
- XC8 2.00 Compiler

III. Target hardware setup

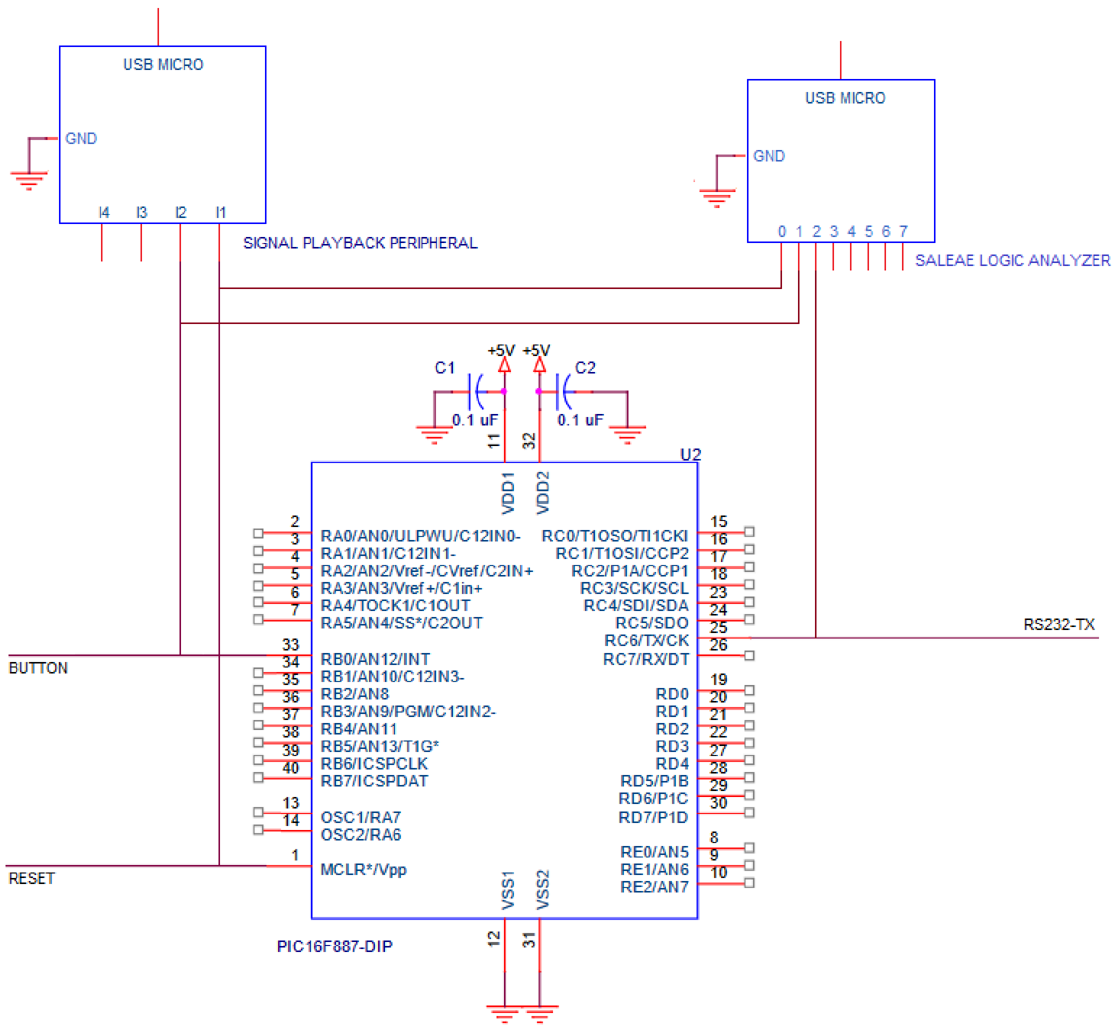
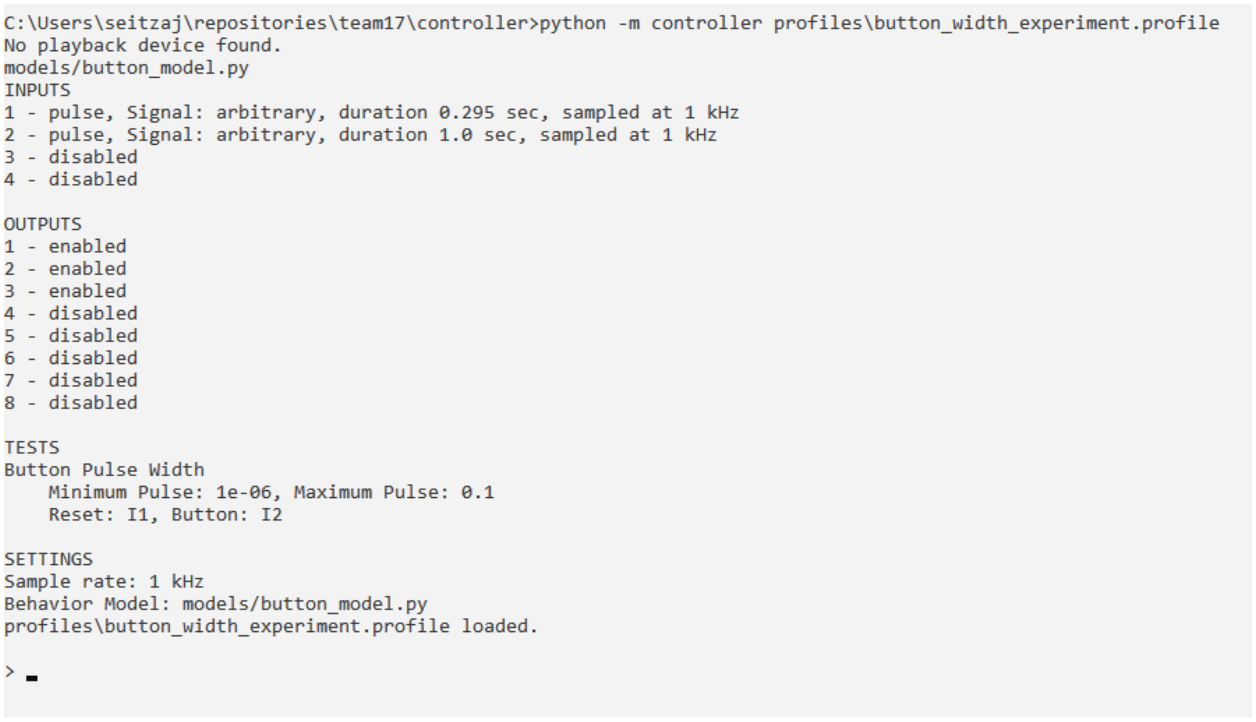


Figure 6: Circuit diagram for button duration experiment example

1. Download the target device code from the pic-button-duration-device repository
2. Assemble the target device circuit shown in Figure 6: Circuit diagram for button duration experiment example
3. Program the PIC16F887 with the code from Step 1 using the PICKit3 programmer. Ensure the Zybo board is disconnected from the Reset during programming, so that the PICKit can drive the Reset signal appropriately

IV. Software usage



```
C:\Users\seitzaj\repositories\team17\controller>python -m controller profiles\button_width_experiment.profile
No playback device found.
models/button_model.py
INPUTS
1 - pulse, Signal: arbitrary, duration 0.295 sec, sampled at 1 kHz
2 - pulse, Signal: arbitrary, duration 1.0 sec, sampled at 1 kHz
3 - disabled
4 - disabled

OUTPUTS
1 - enabled
2 - enabled
3 - enabled
4 - disabled
5 - disabled
6 - disabled
7 - disabled
8 - disabled

TESTS
Button Pulse Width
    Minimum Pulse: 1e-06, Maximum Pulse: 0.1
    Reset: I1, Button: I2

SETTINGS
Sample rate: 1 kHz
Behavior Model: models/button_model.py
profiles\button_width_experiment.profile loaded.

> ▬
```

Figure 7: Loaded profile for button experiment

1. Set up the CAERUS hardware and software according to the “Using CAERUS” section, loading the existing button duration profile from the repository with the following command, which should result in the settings shown in Figure 7:

```
python -m controller profiles/button_width_experiment.profile
```

2. Start the test by issuing the command `run` in the software console
3. Wait for results to be displayed in the software console