FemtoClock2 Python Driver Setup

The FemtoClock2 Python driver allows customers to create or load a device setting file. The Python environment uses an Aardvark or FTDI USB to I2C device to program or OTP the device. It comes with the FTDI, Aardvark drive, and other Python library.

Contents

1. Initial Installation

The FemtoClock2 driver requires Python 3.6.8. Aardvark currently will not work on Python 3.7, and the driver will fail to load in Python 3.7 even if Aardvark functionality is not being used.

1. Create a Python 3.6.8 virtual environment.

```
c:\> python -m venv fc2_venv
```

2. Launch a command prompt in Admin mode and activate the virtual environment.

```
fc2_venv\Scripts\activate
```

3. Upgrade pip.

```
pip install --upgrade pip
```

4. Download the driver .tar.gz file (e.g., r_drv_fc2-6.0.0-win_amd64.tar.gz for 64-bit) from Renesas Application Engineering.

5. Extract the .tar.gz file to a folder (e.g., c:\temp\fc2), then unzip .gz and .tar file.
   You should see all the .whl file.

6. Run the following command (in the activated virtual environment) to install the driver. Replace “c:\temp\fc2” with the folder where you extracted the .tar.gz file in the previous step.

```
pip install --find-links=c:\temp\fc2  r_drv_fc2
```

2. Driver Update

When a driver is already installed, the driver can be upgraded to the latest version as follows:

```
pip install --find-links=c:\temp\fc2  r_drv_fc2  --upgrade
```

3. Using the Command Line Interface (CLI)

3.1 Run the CLI

From within the virtual environment, run fc2_cli:

```
(py36_x86) c:\deviceNet\framework>fc2_cli
rbcli V3.1.0 rbcore V7.1.0 driver V0.0.0.dev75174
? What would you like to do? (Use arrow keys)
 » New Product
   Load Settings
```

Use the up/down arrow to select “New Product” or “Load Settings”. For “New Product”, it will list the complete product name.

```
(py36_x86) c:\deviceNet\framework>fc2_cli
? What would you like to do?  New Product
? Select a product variant (Use arrow keys)
   RC32504A HW: b
   RC32514A HW: b
   8P49N344 HW: b
 » RC22504A HW: b
   RC22514A HW: b
   DIVLDO HW: b
```

3.2 Create Setting using “smart config”

Use the set command to enable smart config and set metadata fields. For example, you can use the following command to create a simple synthesizer configuration with two outputs:

```
set smart_config true
set operation_mode synth
set xtal_frequency 49.152    (Note: 49.152 is in MHz)
set output_goal_frequency0 122.88
set output_goal_frequency1 61.44
```

The session will look like the following:

```
? What would you like to do?  New Product
? Select a product variant  RC22504A HW: b
WARN 3 > set smart_config true
smart_config = True [LOCKED to true]
> set operation_mode synth
operation_mode = synth (Synthesizer) [LOCKED]
> set xtal_frequency 49.152
xtal_frequency = 49.152MHz [LOCKED to 49.152]
> set output_goal_frequency0 122.88
```

```
output_goal_frequency0 = 122.88MHz [LOCKED to 122.88]
output_actual_frequency0 = 122.88MHz
> set output_goal_frequency1 61.44
output_goal_frequency1 = 61.44MHz [LOCKED to 61.44]
output_actual_frequency1 = 61.44MHz
>
```

You can then save the setting using the save command:

```
>save "c:\temp\mysetting.tcs"
```

Note that the setting file created with the driver *cannot* be used with the FemtoClock2 Timing Commander GUI because the field names are different. However, you can load the settings file back into the driver.

3.3 Connect to a Board and Perform I/O

Type commands to connect, read, write, etc. Note that this example assumes that the default I2C slave address is correct for communicating with the attached device. If not, you can override with the -a argument to the read/write commands.

```
> connect ftdi i2c -c A
Choose which FTDI device you wish to use (listed by serial number)
1 - s/n: "A"
choice: 1
Connected [ftdi (channel A) I2C 0x09]
> write fc 0 c0 10 20
Wrote 4 byte(s) to offset 0xFC
> read 0 -l A0 --table
  OFFSET  |  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
----------------------------------------------------------
 00000000 | 33 10 4A 30 32 02 00 00 04 00 30 00 00 00 00 00
 00000010 | 0B 00 19 9A 00 00 00 00 00 00 00 F0 00 00 00 00
 00000020 | 00 00 F0 01 00 00 00 00 00 00 00 00 00 00 00 00

 00000030 | 03 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00000040 | 03 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00000050 | 03 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00000060 | 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00000070 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00000080 | 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00000090 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
>

> read --all --output myfile.txt
> disconnect
Disconnected
> exit
```

3.3.1. Example of “connect” Command

```
connect aardvark i2c -p 0     // Connect Aardvark device
connect ftdi i2c -c A         // Connect FTDI device
read 0 -l A0                  // All number is Hex
                // read ADDRESS -l LENGTH --all --table target
write 0 33 10 00 00
                //  write ADDRESS target data [data ...]
disconnect                    // Disconnect I2C device
exit                          // Exit the CLI
```

3.3.2. Positional Arguments for “connect”

- adapter – Type of the adapter providing connectivity to the device (e.g., FTDI)
- Protocol – Serial protocol for the connection (e.g., I2C, SPI)

3.3.3. Optional Arguments

- -h, --help show this help message and exit
- -c, --channel CHANNEL
  Specific channel to connect to on the adapter, if required (e.g., A or B for 2-channel FTDI)
- -p, --port-number PORT_NUMBER
  Specific port_number to connect to on the adapter, if required (e.g., 0 for Aardvark)

4. Using the Python Driver from Python Script

4.1 Create Settings using “smart config”

```
from r_drv_fc2.device_abstraction.femtoclock2 import Femtoclock2
Import  r_drv_fc2.device_abstraction.config as cfg
from rbcore.io import DeviceConnection, SerialProtocol

# STEP 1: Create the device and connect
fc2 = Femtoclock2(Femtoclock2.create_device('8p49n344', 'A'))
fc2.device.connection = DeviceConnection.connect(
fc2.device.connection_info, 'ftdi', SerialProtocol.I2C, channel=0)
7
# STEP 2: Create a configuration
# Note the use of "extra" argument to specify additional arbitrary bitsets that
need to be set.

ja_config = {
    'ja_config': cfg.Config(
        cfg.Xo(frequency=49.152e6),
        cfg.Apll(vco_frequency=9.345703125e9, dsm_order=3, cp_up=3, cp_dn=3,
cp_offset=1),
        [
            cfg.Output(index=0, disabled=False, goal_frequency=322.265625e6),
            cfg.Output(index=1, disabled=False, goal_frequency=322.265625e6),
            cfg.Output(index=2, disabled=False, goal_frequency=322.265625e6),
            cfg.Output(index=3, disabled=False, goal_frequency=322.265625e6)
```

```
        ],
        dpll=cfg.Dpll(enabled=True, input_frequency=31.25e6,
mode=cfg.DpllMode.JitterAttenuator.value),
        extra=[
            (fc2.bitsets.TOP.APLL.LPF_3RD_CNFG.cnf_lpf_r3(), 0x05),
            (fc2.bitsets.TOP.APLL.LPF_CNFG.cnf_lpf_cp(), 0x03),
            (fc2.bitsets.TOP.APLL.LPF_CNFG.cnf_lpf_res(), 0x05)]),
    'config2': [
        (fc2.metadata.operation_mode, 'ja'),
        (fc2.metadata.refclk_input_frequency, '25MHz'),
        (fc2.metadata.output_goal_frequency[0], '156.25MHz'),
        (fc2.bitsets.TOP.APLL.CP_CNFG.cp_up(), 5),
    ]
}

# STEP 3: Program the device with the configuration
fc2.program(ja_config)

# STEP 4: Close the connection
fc2.device.connection = None
```

5. Revision History

Revision	Date	Description
1.0	May 10, 2021	Initial release.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.