

User Manual

Broadway
Version 2.3.5

June 6, 2024
Build 6
©FibreCode GmbH

Contents

1	History	3
1.1	Broadway2-Software	3
1.2	Broadway2-Firmware	4
2	Usage	6
3	Overview	6
4	Installation Windows	8
4.1	USB Broadway2	8
4.1.1	VLAN on Windows	8
4.2	USB Broadway2 API	9
4.3	Python3 Broadway2	9
5	Installation Linux	10
5.1	ip / ifconfig comparison	11
5.1.1	ip Basics cheat sheet	11
5.2	USB Broadway2 API	11
5.3	Python3 Broadway2	12
6	First Steps	13
7	Python samples	14
7.1	Migration from broadway	14
7.2	Changes on broadway2	15
7.3	fc_app1	16
7.4	get_hw_info	17
7.5	fc_control	18
7.6	phy_config	19
7.7	phy_control	21
7.8	Networking-Only	21
7.9	Raw-Only	22
7.9.1	raw_send	23
7.9.2	raw_receive	23
7.9.3	raw_loop	24
7.10	USB SMI/SPI Adapter-Only	25
7.10.1	tool_gui	25

7.10.2	smi_access	26
7.10.3	spi_access	26

1 History

Version history and overview of added features.

1.1 Broadway2-Software

Broadway2 software packages are deployed for Windows and Linux zip-files. The complete Broadway2-API is cross-compatible using Windows/Linux on C-API/SDK and python 3 `broadway_api` wrapper including multiple examples.

- V2.3.5
Requires FW V2.30.x on USB-Adapters to support new features
 - Updated `broadway2-python 3` to V2.4.11 for updated PCB-Revisions, added raw-socket samples for net. Changed install to wheel-files.
 - restructured and extended binaries for Raspberry Pi supporting 32/64-bit.
- V2.3.4
Requires FW V2.30.x on USB-Adapters to support new features
 - Update CDC-ECM driver V3.76.0 for improved Windows 10/11 support.
 - Windows VLAN feature to preserve VLAN-tags
 - Updated python 3 wrapper to V2.4.10 for new functionalities
- V2.3.3 - Internal Release only, not published
 - Update CDC-ECM driver V3.70.0 for improved Windows 10/11 support.
 - Updated python 3 wrapper to V2.4.9 for new functionalities
- V2.3.2
Requires FW V2.30.x on USB-Adapters to support new features (details see below)
 - Broadway2-API V2.30
 - Expanded API to control 10BASE-T1L short/long-distance easier

- Expanded functions for FC401 to set SMI/SPI to passive-mode
 - Updated python 3 wrapper to V2.4.8 for new functionalities
- V2.3.1
 - Requires FW V2.29.x on USB-Adapters to support new features (details see below)
 - Broadway2-API V2.29
 - Expanded API to get Frame-Statistic data (e.g. CRC-Errors on Rx)
 - Expanded support for SMI-Clause45 read-Increment
 - update python 3 wrapper to V2.4.5 for new PhyMultRead45 and statistics
 - added statistics python example
- V2.3.0
 - Broadway2-API V2.27
 - added CDC-ECM V2.56
 - added fix for StreamTo if bandwidth of video streams is bigger than available network speed
 - update python 3 wrapper to V2.4.2 for new external Trigger-In/Out Feature
 - added support for TJA1101BHN, ADIN1100 (FC621, FC631)
- V2.2.4
 - updated CDC-ECM V2.55
 - added support for FC401

1.2 Broadway2-Firmware

Broadway2 Firmware is rolled out as Windows zip including flash-tools for firmware updates using Windows PC environment on each device. For full featured use-cases, it's always recommended to update firmware and software to latest available packages.

- V2.30
 - extended FW for FC621/FC631 (10BASE-T1L) short/long range persistent mode
 - extended functions to control SMI/SPI active/passive modes
- V2.28
 - added support for IEEE802.1-Clause45 Read-Increment
- V2.27
 - added force-trigger using Trigger-Extension Board
- V2.26
 - added property to specify a min. interval between two consecutive TX frames
 - added support trigger output on UART-TX pin for FC611, FC612, FC631 and FC401
 - added support for TJA1101BHN
 - build-in read/write reg functions are deprecated
- V2.25
 - added trigger-in on UART-RX pin for FC611, FC612, FC631 and FC401
 - extended support for IEEE 802.3 clause 45
- V2.24
 - SPI-fix: wait until all bytes are transmitted/received through SPI before disabling the chip select signal
 - new API functions for external Phy-Access
 - optimized PHY connections on FC401: no PHY connected, SMI only connected
- V2.23
 - added support for SPI access on FC401

- added properties for SPI configuration: Protocol-Type and SPI-Speed
- optimized PHY connections on FC401: no PHY connected, SMI only connected

2 Usage

Starting with Broadway2, the initial software package is optimized and re-worked for higher throughput and much more features using USB-to-100BASE-T1 and 10BASE-T1L FibreCode Adapters.

Broadway2 supported adapters are:

- FC611 USB 100BASE-T1 Adapter Raw (TJA1100)
- FC602 USB 100BASE-T1 Adapter (TJA1101)
- FC612 USB 100BASE-T1 Adapter Raw (TJA1101)
- FC621 USB 10BASE-T1L Adapter (ADIN1100)
- FC631 USB 10BASE-T1L Adapter Raw (ADIN1100)
- FC401 USB SMI/SPI Adapter

Note: Broadway2 will only find and enumerate new Adapters like FC602, FC611, .. Broadway will only find and enumerate FC601.

If using FC601 adapters, latest CDC-ECM on Windows will also support this one. Difference is only relevant and important when accessing FC6xx/TJA110x using control-API.

Anyhow, FC601 and FC602 can be used within same python application. Broadway and Broadway2 can be both imported and used in parallel.

3 Overview

Broadway2 software package supports multiple USB-connection and abstraction layers for Control, CDC-ECM and Bulk-Raw interfaces. The complete software is cross-build and tested on following platforms:

- Windows 10 Standard PCs/Notebooks
- Linux Ubuntu 18.04 on x64 notebook and running VM using Virtual-Box
- Linux Ubuntu 20.04 on x64 notebook and running VM using Virtual-Box
- Linux Ubuntu 18.04 deployed for Nitrogen8M from Boundary Devices
- Raspberry Pi 3A+, 3B+ and 4B+

On Windows machines, CDC-ECM will install the standard network interfaces. In parallel, the `Broadway2_api.dll` enables access to NXP TJA110x MDIO registers for full read/write access. Using Raw-Ethernet adapters, the Broadway2-API includes complete functionality for USB-Bulk Rx/Tx Queues enabling high-speed raw Ethernet traffic in both directions.

Using USB Control, all properties and functions of the adapter itself and the NXP Ethernet Phy are accessible. In extension to these base functions, Raw-Ethernet Rx/Tx interfaces enables the usage of sending/receiving of Layer2 frames on 100Base-T1 networks.

Raw-Ethernet is only accessible using FC611 or FC612. FC602 supports standard network device.

Depending on setup and requirements, standard network devices can now be combined with Raw-Ethernet Adapters to get both: TCP/IP traffic using sockets and Multi-Instance Raw-Ethernet traffic access. All connected to same Windows or Linux host.

In case of having requirement to control complete functionality of the USB-Adapters, Broadway2 library needs to be added/installed. This will enable full access to the NXP Phy on all adapters supporting Broadway2. So, for FC602 this is optional as network device is working without Broadway2-API. For the USB-Raw Adapters it's mandatory, as all functions for control and raw access are implemented inside Broadway2-API.

For external devices like NXP evaluation boards, custom designs or complete devices, USB SMI/SPI Adapter FC401 brings an easy way to get same Broadway2-API extended with SMI/SPI access to external NXP TJA1100, TJA1101, TJA1102, SJA1105x and SJA1110x.

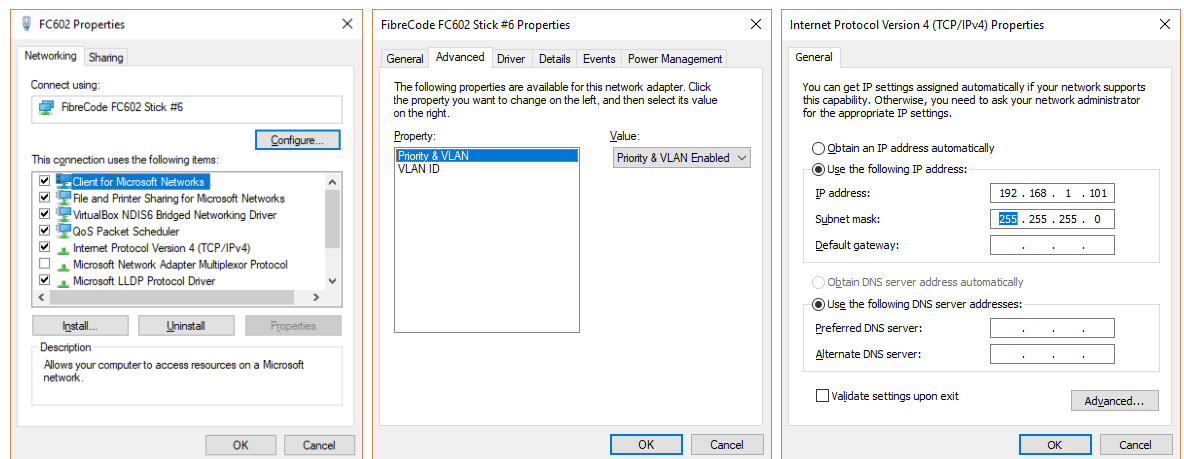
In difference to FC6xx, FC401 has no switch or phy assembled on board, but has 18-pins connector for easy and customizable connections.

4 Installation Windows

4.1 USB Broadway2

1. Install CDC-ECM V3.76.0 (or later)
2. run setup.exe
3. follow dialogs and complete installation
4. plug-in FC602, FC611 and/or FC612 to your PC (extended via USB-Hub if required)

Using FC602, enables standard network device which can be used as any other Ethernet network device.

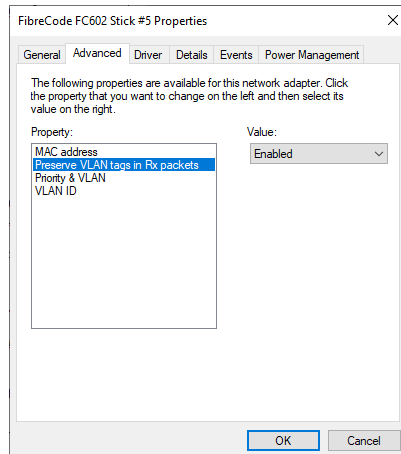


Using FC611, FC612 or FC631 will NOT have a standard network device installed. Instead, Broadway2-API comes with extended functions to access Ethernet Raw packets directly. Refer to `broadway2-API`, `SDK` and `raw_send.py` and `raw_receive.py` to get more details here.

4.1.1 VLAN on Windows

USB network drivers using CDC-ECM on Windows, has some issues in clear definition when using/forwarding VLAN-IDs. To preserve VLAN/Priority field for tests, analysis or using Wireshark, driver has custom property to

control this behavior:



By default, this is enabled to have 802.1Q field included, even no VLAN/Priority Ids are defined or used. If there are reasons to remove this field, like default implementation of driver versions before, please select Disabled here.

Note: Using Linux and Wireshark, VLAN/Priority was shown correctly also in versions before. This is standard Linux CDC-ECM implementation and no effect or relation to implementation on MS Windows.

4.2 USB Broadway2 API

1. copy `broadway2_api.dll` to windows system folder. 32-Bit version goes to SysWow64, 64-Bit version to System32 folder.
2. open console to verify access
3. run `broadway2_sample.exe info` to list FC6xx properties
4. information about stick contents and serial number is listed

4.3 Python3 Broadway2

1. Download and install python3-win from python.org. Recommended is using latest 3.x version. At this release, Python 3.6.3 was used.
2. Goto folder Python3 and locate latest version `python3/broadway2-2.x.y.z-py3-none-any.whl`

3. Open console here and run: `pip install broadway2-x.y.z-py3-none-any.whl`

For FC601 users, this will extend python packages to have Broadway and Broadway2 installed. Verify samples to see difference in importing Broadway for FC601 and Broadway2 for FC611. If mixed systems has high priority and is often used, we recommend to upgrade using FC602 for network device to only use Broadway2-API.

5 Installation Linux

To use FC611, FC602 or FC612 on Linux, no special kernel driver is required. On Linux systems like Ubuntu, Raspberry Pi, ..., Broadway2 API is enabled by just installing/using `libbroadway2_api.so` module.

FC602 implements a standard network device and is automatically found supporting USB CDC-ECM standard by Linux standard kernel modules. The new connected adapters needs to configured using `ifconfig` or `ip`. Refer to Linux user manual for all options.

Note: If ping is not working as expected, verify if default route is added for plugged-in USB devices: e.g. `sudo ip route add 192.168.1.0/24 dev eth0`

As the adapter supports a hardware switch to configure Master/Slave, there is basically no need to install any software. By installing the extension to Broadway2-API using `libbroadway2_api.so`, all samples and python 3 scripts can be used like on Windows platforms.

The following Linux-based environments are build and tested by shipping required binaries:

- Ubuntu 22.04 on standard PC (64-bit)
- Ubuntu 18.04 on standard PC (32-bit)
- Raspberry Pi 3x+ using Raspi OS armv6l image (Apr 2024)
- Raspberry Pi 4B+ using Raspi OS armv7l image (May 2024)
- Raspberry Pi 4B+ using Raspi OS aarch image (May 2024)

5.1 ip / ifconfig comparison

Comparison between old ifconfig and new ip command.

Description	ifconfig	ip
show network configuration	ifconfig	ip addr show and ip link show
activate network	ifconfig eth0 up	ip link set eth0 up
deactivate network	ifconfig eth0 down	ip link set eth0 down
set ip	ifconfig eth0 192.168.0.77	ip address add 192.168.0.77 dev eth0

Table 1: comparison ip / ifconfig

5.1.1 ip Basics cheat sheet

Former tool *ifconfig* should be not used anymore.

Description	iproute2 command
Show link status	ip link show
Show link status incl. statistics	ip -statistics link show
Show IP address	ip addr show
Set IP address	ip addr add IP/NETMASK dev DEVICE
Remove IP address	ip addr del IP/NETMASK dev DEVICE
Remove all IP addresses	ip addr flush dev DEVICE
Show routing table	ip route show
Set standard gateway	ip route add default via IP
Show ARP-Cache	ip neigh show
Show connections	ss -tcp -all -processes -extended -numeric
link up	ip link set dev DEVICE up
link down	ip link set dev DEVICE down

Table 2: ip Basics cheat sheet

5.2 USB Broadway2 API

1. open terminal and goto folder where `broadway2-xxxx.zip` file is located
2. extract files to folder

```
unzip LSP_Broadway_V_xx.yy.zz -d LSP_Broadway_V_xx.yy.zz
```

3. select folder matching your environment
4. run `sudo install.sh`.
This will perform the copy of the so and ldconfig to update library cache. This install Script performs theae two lines:
`cp libbroadway2.api.so /usr/local/lib`
`sudo ldconfig /usr/local/lib`
5. goto folder where test sample is located
e.g. `cd Linux/Raspbian/bin/linux`
6. change mode of `broadway_sample`
`chmod +x broadway2_sample`
7. run `broadway2_sample`
`sudo ./broadway2_sample info`
8. information about stick contents and serial number is listed

5.3 Python3 Broadway2

In the pretty same way like on Windows, python Broadway wrapper is installed and used. As Python is cross-platform, all samples are running on both systems, Windows and Linux based environments. In list below, we use pip as python3 package manager to get latest wheel installed from local file.

Important Note: To move from former installed packages using distutils, existing older `broadway2`-packages needs to be deleted first from `/usr/local/lib`:

Try to install:

```
sudo pip3 install broadway2-2.4.11-py3-none-any.whl
```

```
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
```

```
Processing ./broadway2-2.4.11-py3-none-any.whl
```

```
Installing collected packages: broadway2
```

```
Found existing installation: broadway2 2.4.10
```

```
Cannot uninstall 'broadway2'. It is a distutils installed project and thus we cannot
```

```
-----
```

```
cd /usr/local/lib/python3.7/dist-packages/
```

```
sudo rm -r broadway2-2.4.*
```

```
-----
```

```
sudo pip3 install broadway2-2.4.11-py3-none-any.whl
```

Looking in indexes: <https://pypi.org/simple>, <https://www.piwheels.org/simple>
Processing ./broadway2-2.4.11-py3-none-any.whl
Installing collected packages: `broadway2`
Successfully installed `broadway2-2.4.11`

1. Depends on used Linux environment:
e.g. `sudo apt-get install python3, venv, ...`
2. open folder `python3` which contains e.g. `broadway2-2.4.11-p3-none-any.whl`
3. `pip install broadway2-x.y.z-py3-none-any.whl`, where `x.y.z` needs to be replaced with latest version

To use the samples on Linux run:

`sudo python3 <scriptname>.py`. All examples can be started nearly the same way under Windows using: `python <scriptname>.py`.

6 First Steps

1. Use switch on side to configure Master/Slave on OABR network. Master will be signaled with Blue, Slave with Green LED. Slow-blinking for link-down, fast blinking for link-up is signaled using same LED.
2. Connect UTP-cable on stick by just adding the wires to TRX_P and TRX_N to the green cable connector and plug that into stick. When stick is held USB-A connector oriented down, TRX_P is the most left and TRX_N is the middle pin.
3. Connect USB-adaptor to USB-Host (Windows-PC, Linux Ubuntu, Raspberry Pi)
 - FC602: network device will appear to be used as standard network In parallel, USB-Bulk control device can be used with Broadway2-API to access TJA110x and the properties of the FC602
 - FC611/FC612: Only USB-Bulk driver is installed. As difference to FC602, there are now 3 devices to be used on Broadway2-API. Same control device as FC602. Additionally there is now a raw-tx and raw-rx device.

4. All applications using Broadway2-API control device are compatible for all USB Adapters: FC611, FC602 and FC612. If using the raw-devices, only FC611 and FC612 supports those.

7 Python samples

All python scripts are verified on Windows 10/11 (64-bit/Python 3.4 32-bit), latest Raspbian and Ubuntu 22.04 (Desktop 64-bit). For Windows machine, we recommend to download latest python 3.x (32-bit/64-bit) to start with identical environment. For Linux machine, debian based, `sudo apt-get install python3` will do the required installation. Some latest distributions already include python3. In addition, as pip is not part of standard installation, run: `sudo apt-get install python3-pip`

As the python Broadway installation package includes the complete abstraction of different OS-environments, the only difference in using the scripts is:

- Windows: `python <script-name.py>`
- Linux: `sudo python3 <script-name.py>`

In the following examples, `PYTHON3` is used as common call for Windows or Linux systems.

7.1 Migration from broadway

This section describes the small differences between Broadway and Broadway2. As FC611 already was delivered with Broadway2, migration is only

required from FC601 to FC602.

Broadway

```
from broadway import Broadway

base = Broadway()
base.scan_and_update()

for a in base.enumerate():
    print(a) # returns FCBProperties
    print(a.AdapterName)

# open control
dev = base.open_instance(a.Inst)

# access phy
phy = TJA1100()
phy.connect(dev)

..
base.close_instance()
```

Broadway2

```
from broadway2 import Broadway
from broadway2 import get_phy_access

base = Broadway()
# automatically done in constructor

for a in base.enumerate():
    print(a) # returns FC6xx
    print(a.AdapterName)

# open control
dev = base.open_control(a.Inst)
a.connect(dev)

# access phy
phy = get_phy_access(a) #
SMI-Access
# phy is already connected with dev

..
base.close()
```

Note: Phy-access is the same like on Broadway. FC6xx is used to access all adapter properties and returns automatically correct phy-instance (TJA1100 on FC611, TJA1101 on FC602/FC612).

The following samples uses the Broadway API to access control interface. This is working common for FC611, FC602 and FC612.

7.2 Changes on broadway2

With the extension to support new USB SMI/SPI-Adapter FC401, the API has changed a bit to support access to build-in Phy on FC6xx and also external Phy/Switch on FC401.

Changes are:

- Renamed FC6xx to FCAdapter

- Removed adapter.get_phy()
- Added new global function: get_phy_access(adapter)

With this optimization, get_phy_access() supports both: Internal and external Phys. Finally, typical python sample using Broadway2 with version V2.2.x.

New Broadway2 example

```
from broadway2 import Broadway, FCAdapter
from broadway2 import get_phy_access # Import new function

base = Broadway()
# automatically done in constructor

for a in base.enumerate():
    print(a) # returns FC6xx
    print(a.AdapterName)

# open contro device and connect adapter with this device
dev = base.open_control(a.Inst)
a.connect(dev)
# access phy
phyAccess = get_phy_access(a) # default is phyAddress=0

    print(phyAccess.read_reg(0x02)) # PhyId
..
    base.close(dev) # close dev for control access
```

Please refer also to following samples using get_phy_access(). Samples are the same like before, but adjusted to new Broadway2-Python3-API.

7.3 fc_app1

For simple status and overview of connected USB FC6xx Adapters, there is tool application installed within broadway2 package. This enables to list and print properties from anywhere having console open:

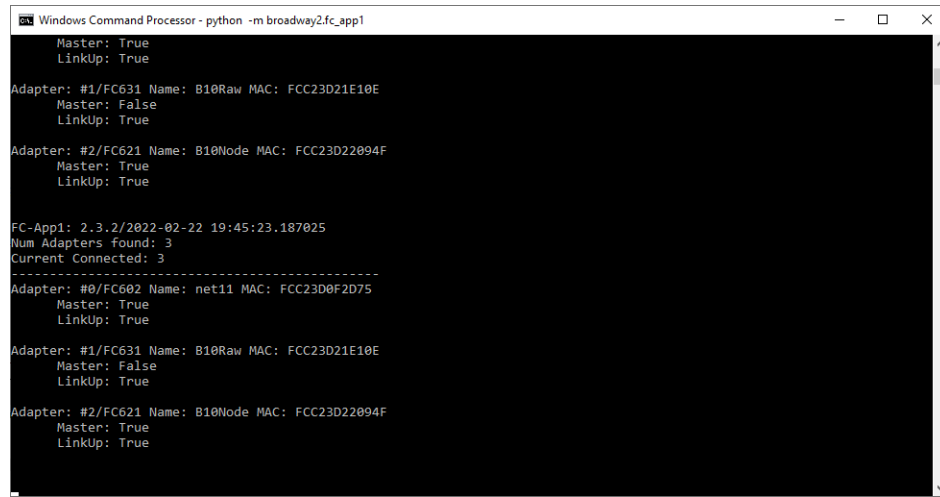
On Windows console enter:

```
python -m broadway2.fc_app1.py
```

Typically using Linux:

```
sudo python3 -m broadband2.fc_app1.py
```

Now, like using `fc_control.py`, enumerations are automatically updated and printed in a loop. Unplug/re-plug USB will automatically change current list.



```
Windows Command Processor - python -m broadband2.fc_app1
Master: True
LinkUp: True

Adapter: #1/FC631 Name: B10Raw MAC: FCC23D21E10E
Master: False
LinkUp: True

Adapter: #2/FC621 Name: B10Node MAC: FCC23D22094F
Master: True
LinkUp: True

FC-App1: 2.3.2/2022-02-22 19:45:23.187025
Num Adapters found: 3
Current Connected: 3
-----
Adapter: #0/FC602 Name: net11 MAC: FCC23D0F2D75
Master: True
LinkUp: True

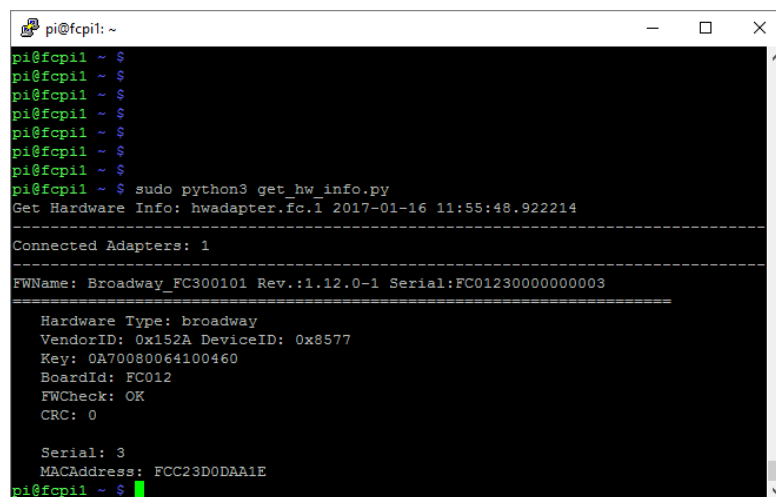
Adapter: #1/FC631 Name: B10Raw MAC: FCC23D21E10E
Master: False
LinkUp: True

Adapter: #2/FC621 Name: B10Node MAC: FCC23D22094F
Master: True
LinkUp: True
```

Pressing Ctrl-C will stop.

7.4 get_hw_info

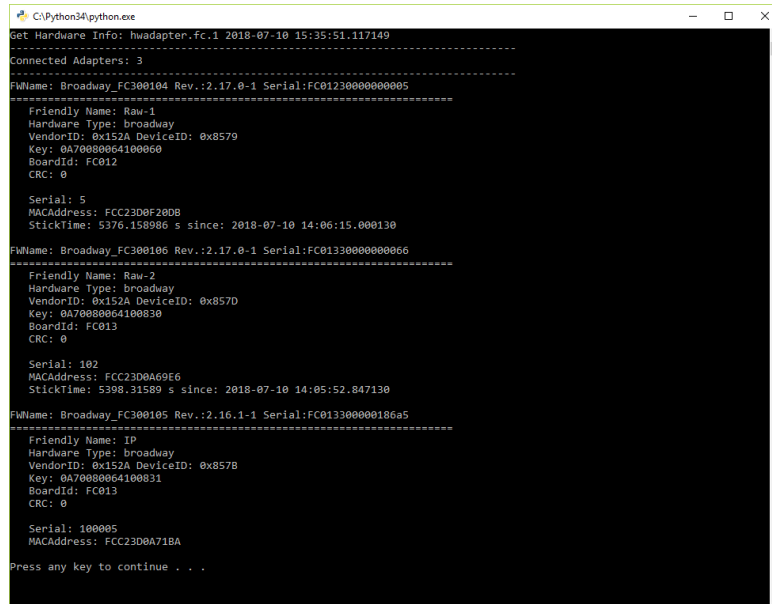
PYTHON3 `get_hw_info.py`



```
pi@fcpi1: ~
pi@fcpi1 ~ $
pi@fcpi1 ~ $
pi@fcpi1 ~ $
pi@fcpi1 ~ $
pi@fcpi1 ~ $
pi@fcpi1 ~ $
pi@fcpi1 ~ $
pi@fcpi1 ~ $
pi@fcpi1 ~ $ sudo python3 get_hw_info.py
Get Hardware Info: hwadapter.fc.1 2017-01-16 11:55:48.922214
-----
Connected Adapters: 1
-----
FWName: Broadway_FC300101 Rev.:1.12.0-1 Serial:FC01230000000003
-----
Hardware Type: broadband
VendorID: 0x152A DeviceID: 0x8577
Key: 0A70080064100460
BoardId: FC012
FWCheck: OK
CRC: 0

Serial: 3
MACAddress: FCC23D0DAA1E
pi@fcpi1 ~ $
```

This example runs on Raspberry Pi and lists all major properties of connected Broadway2 adapters.



```

C:\Python34\python.exe
Get Hardware Info: hwadapter.Fc.1 2018-07-10 15:35:51.117149
Connected Adapters: 3
=====
FWName: Broadway_FC300104 Rev.:2.17.0-1 Serial:FC01230000000005
=====
  Friendly Name: Raw-1
  Hardware Type: broadway
  VendorID: 0x152A DeviceID: 0x8579
  Key: 0A70080004100060
  BoardId: FC012
  CRC: 0

  Serial: 5
  MACAddress: FCC23D0F2008
  StickTime: 5376.158986 s since: 2018-07-10 14:06:15.000130
=====
FWName: Broadway_FC300106 Rev.:2.17.0-1 Serial:FC01330000000006
=====
  Friendly Name: Raw-2
  Hardware Type: broadway
  VendorID: 0x152A DeviceID: 0x857D
  Key: 0A70080004100030
  BoardId: FC013
  CRC: 0

  Serial: 102
  MACAddress: FCC23D0A69E6
  StickTime: 5398.31589 s since: 2018-07-10 14:05:52.847130
=====
FWName: Broadway_FC300105 Rev.:2.16.1-1 Serial:FC013300000186a5
=====
  Friendly Name: IP
  Hardware Type: broadway
  VendorID: 0x152A DeviceID: 0x857B
  Key: 0A70080004100031
  BoardId: FC013
  CRC: 0

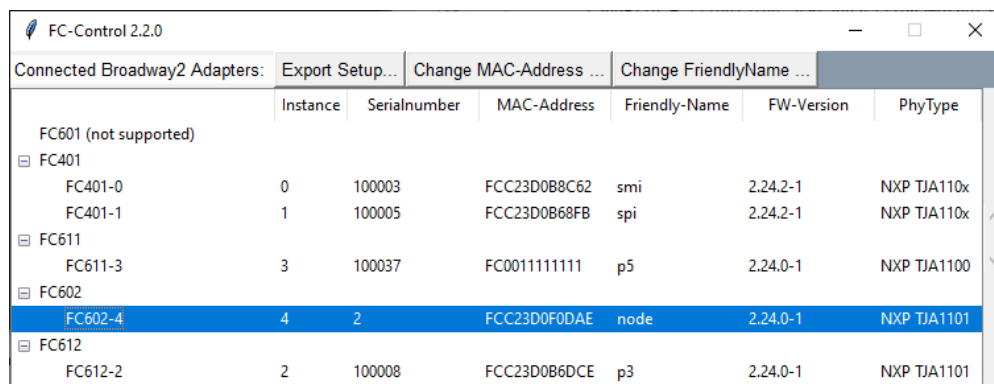
  Serial: 100005
  MACAddress: FCC23D0A71BA
Press any key to continue . . .

```

Another example shows content with same connected adapters like using `fc_control.py`, but here as console tool only running on Windows 10.

7.5 fc_control

PYTHON3 `fc_control.py`



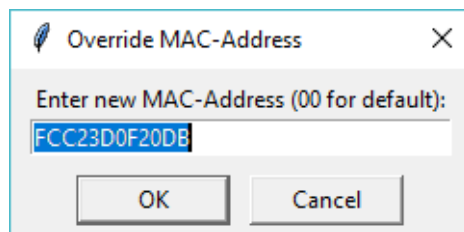
FC-Control 2.2.0						
Connected Broadway2 Adapters: Export Setup... Change MAC-Address ... Change FriendlyName ...						
	Instance	Serialnumber	MAC-Address	Friendly-Name	FW-Version	PhyType
FC601 (not supported)						
FC401						
FC401-0	0	100003	FCC23D0B8C62	smi	2.24.2-1	NXP TJA110x
FC401-1	1	100005	FCC23D0B68FB	spi	2.24.2-1	NXP TJA110x
FC611						
FC611-3	3	100037	FC0011111111	p5	2.24.0-1	NXP TJA1100
FC602						
FC602-4	4	2	FCC23D0F0DAE	node	2.24.0-1	NXP TJA1101
FC612						
FC612-2	2	100008	FCC23D0B6DCE	p3	2.24.0-1	NXP TJA1101

In extension to already shipped console tool `get_hw_info.py`, this python script uses TK for graphical user-interface. This requires on Linux to have

full graphical interface available, not only console like putty. The script supports USB-Pug-and-Play and enumerates all connected FibreCode Adapters supporting Broadway2-API. After each plug/unplug-event, complete tree is rebuild to show connected adapters only.

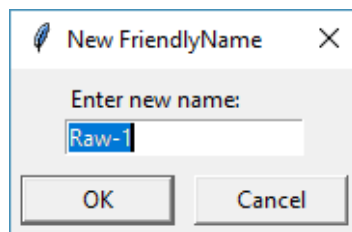
In addition to getting a tree-list overview of connected adapters, it supports the following features:

- Change MAC-Address



Select an adapter of choice to enable this button and open dialog to override default MAC-Address. After pressing OK, adapter is automatically reset to startup with new network device MAC-Address. The factory default is never lost. By just entering 00 as MAC-Address, the override is disabled and factory default is recovered.

- Change FriendlyName



Opens dialog to set a more friendly name for each adapter. This friendly name is stored in persistent memory and can be used to identify correct adapter when using multiple instances.

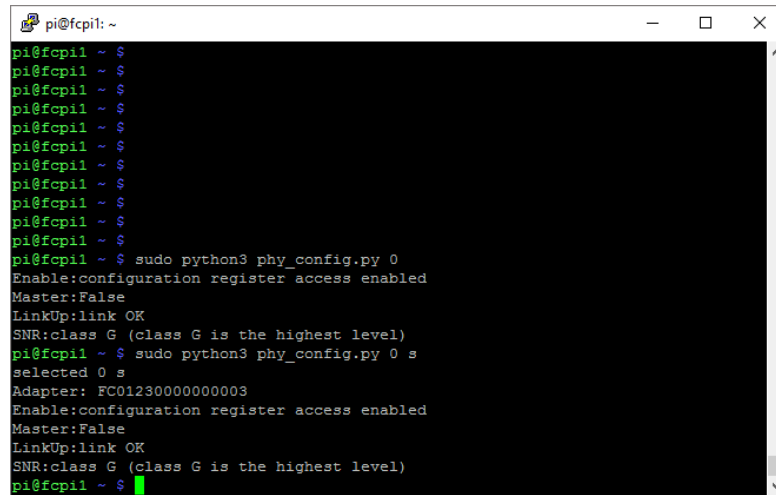
7.6 phy_config

`python phy_config.py <inst> <m|s>` inst: Instance of connected FC611/FC602/FC612.
Started with 0

- m: Configure FC61x as 100BASE-T1 Master

- s: Configure FC61x as 100BASE-T1 Slave

Note: Using software API to control OABR Master or Slave, overrides default setting used by hardware switch. In any case, after re-plug or re-power, FC61x always starts up with configuration set by the switch.

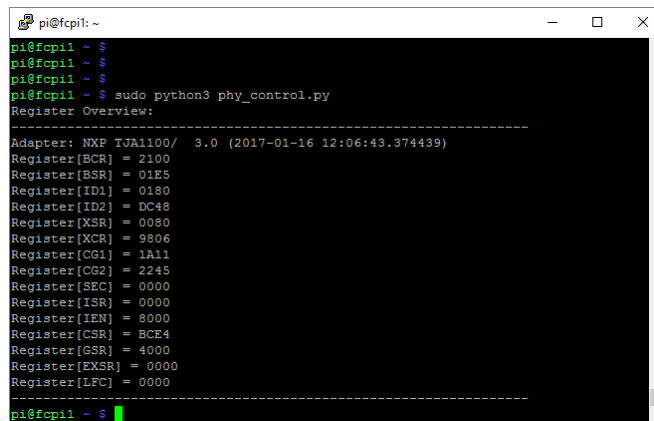


```
pi@fcpi1: ~  
pi@fcpi1 ~ $  
pi@fcpi1 ~ $  
pi@fcpi1 ~ $  
pi@fcpi1 ~ $  
pi@fcpi1 ~ $  
pi@fcpi1 ~ $  
pi@fcpi1 ~ $  
pi@fcpi1 ~ $  
pi@fcpi1 ~ $  
pi@fcpi1 ~ $  
pi@fcpi1 ~ $  
pi@fcpi1 ~ $  
pi@fcpi1 ~ $ sudo python3 phy_config.py 0  
Enable:configuration register access enabled  
Master:False  
LinkUp:link OK  
SNR:class G (class G is the highest level)  
pi@fcpi1 ~ $ sudo python3 phy_config.py 0 s  
selected 0 s  
Adapter: FC01230000000003  
Enable:configuration register access enabled  
Master:False  
LinkUp:link OK  
SNR:class G (class G is the highest level)  
pi@fcpi1 ~ $
```

7.7 phy_control

`python phy_control.py`

This sample just shows how to create register dumps of TJA110x.



```
pi@fcpi1: ~  
pi@fcpi1 ~ $  
pi@fcpi1 ~ $  
pi@fcpi1 ~ $ sudo python3 phy_control.py  
Register Overview:  
-----  
Adapter: NXP TJA1100/ 3.0 (2017-01-16 12:06:43.374439)  
Register[BCR] = 2100  
Register[BSR] = 01E5  
Register[ID1] = 0180  
Register[ID2] = DC48  
Register[XSR] = 0080  
Register[XCR] = 9806  
Register[CG1] = 1A11  
Register[CG2] = 2245  
Register[SEC] = 0000  
Register[ISR] = 0000  
Register[IEN] = 8000  
Register[CSR] = BCE4  
Register[GSR] = 4000  
Register[EXSR] = 0000  
Register[LFC] = 0000  
-----  
pi@fcpi1 ~ $
```

7.8 Networking-Only

Note: This example requires two standard USB-network devices FC602 connected point-to-point. In this setup, Raspberry Pi is used for the web-server and the Windows PC for the client running Firefox Web-Browser.

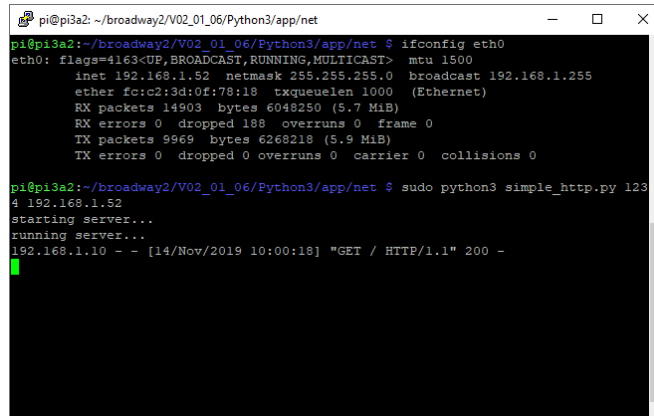
As FC602 is working like standard USB-Network Adapter, all tools like iperf, Wireshark, ... can be used.

In the Broadway-python package, there is a simple web server based on python network classes.

`python net/simple_http.py`

This example starts simple HTTP-Server with given portNumber and IP-Address. Run following cmd on the Raspberry Pi using FC602 on IP 192.168.1.52.

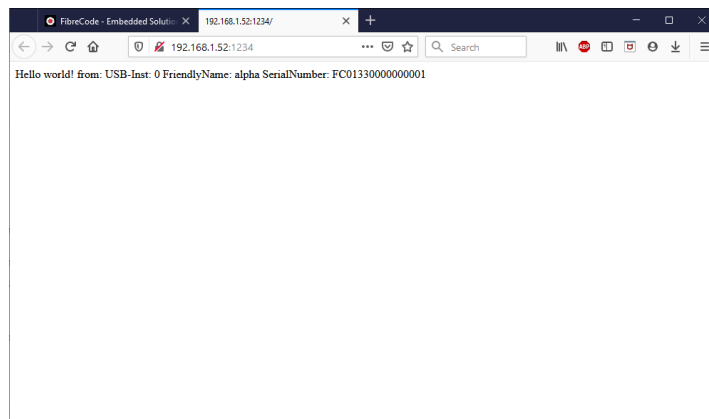
Example: `sudo python3 simple_http.py 1234 192.168.1.52.`



```
pi@pi3a2: ~/broadway2/V02_01_06/Python3/app/net
pi@pi3a2:~/broadway2/V02_01_06/Python3/app/net $ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.52 netmask 255.255.255.0 broadcast 192.168.1.255
    ether fc:c2:3d:0f:78:18 txqueuelen 1000 (Ethernet)
    RX packets 14903 bytes 6048250 (5.7 MiB)
    RX errors 0 dropped 188 overruns 0 frame 0
    TX packets 9969 bytes 6268218 (5.9 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

pi@pi3a2:~/broadway2/V02_01_06/Python3/app/net $ sudo python3 simple_http.py 123
4 192.168.1.52
starting server...
running server...
192.168.1.10 - - [14/Nov/2019 10:00:18] "GET / HTTP/1.1" 200 -
```

After running internet browser from Windows using FC602 with 192.168.1.10, output will look like:



Note: Address line in browser was entered here as: 192.168.1.52:1234

For each request, the simple-http-server will print some details for the HTTP-GET. Pressing F5 for refresh will trigger new HTTP-GET.

7.9 Raw-Only

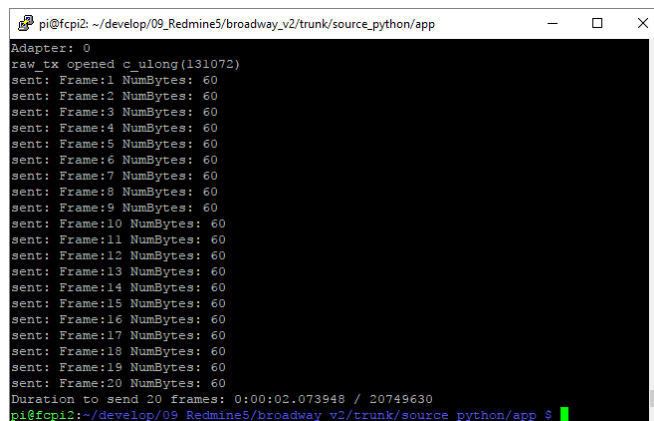
Note: Extension of Broadway2-API to access the raw-rx and raw-tx devices is only supported using FC611 and FC612!

7.9.1 raw_send

`python raw_send.py`

This sample just shows simple example to send raw ethernet data to UTP connected devices. Especially for all low-level tests bringing up own ECU node and switch designs, the FC611 fits perfect for all required stimulation and diagnosis of Layer2 traffic.

For easier usage of all Ethernet/IP protocol types, have also look on *pypacker* using `pip install pypacker`.



```
pi@fcpi2: ~/develop/09_Redmine5/broadway_v2/trunk/source_python/app
Adapter: 0
raw_tx opened c_ulong(131072)
sent: Frame:1 NumBytes: 60
sent: Frame:2 NumBytes: 60
sent: Frame:3 NumBytes: 60
sent: Frame:4 NumBytes: 60
sent: Frame:5 NumBytes: 60
sent: Frame:6 NumBytes: 60
sent: Frame:7 NumBytes: 60
sent: Frame:8 NumBytes: 60
sent: Frame:9 NumBytes: 60
sent: Frame:10 NumBytes: 60
sent: Frame:11 NumBytes: 60
sent: Frame:12 NumBytes: 60
sent: Frame:13 NumBytes: 60
sent: Frame:14 NumBytes: 60
sent: Frame:15 NumBytes: 60
sent: Frame:16 NumBytes: 60
sent: Frame:17 NumBytes: 60
sent: Frame:18 NumBytes: 60
sent: Frame:19 NumBytes: 60
sent: Frame:20 NumBytes: 60
Duration to send 20 frames: 0:00:02.073948 / 20749630
pi@fcpi2:~/develop/09_Redmine5/broadway_v2/trunk/source_python/app $
```

7.9.2 raw_receive

`python raw_receive.py`

This sample just shows how to receive raw ethernet data from UTP connected devices.

For easier usage of all Ethernet/IP protocol types, have also look on *pypacker*, *scapy*, ... using `pip install pypacker` or `pip install scapy`. Both APIs are matching perfect to compose/decode raw Ethernet traffic and transmit/receive Ethernet frames using Broadway2 raw-API.


```

pi@fcpi2: ~/develop/09_Redmine5/broadway_v2/trunk/source_python/app
Timestamp: 71740930 SeqCnt: 3015 Diff to last: 608
Dest: FFFFFFFF Src: FCC23D0DD471 Type:88A8 DataLen:168
Num Raw Frame Received: 3603
Timestamp: 82572530 SeqCnt: 3602 Diff to last: 587
Dest: FFFFFFFF Src: FCC23D0DD471 Type:88A8 DataLen:342
Num Raw Frame Received: 4213
Timestamp: 92203000 SeqCnt: 4212 Diff to last: 610
Dest: FFFFFFFF Src: FCC23D0DD471 Type:88A8 DataLen:144
Num Raw Frame Received: 4831
Timestamp: 102038990 SeqCnt: 4830 Diff to last: 618
Dest: FFFFFFFF Src: FCC23D0DD471 Type:88A8 DataLen:325
Num Raw Frame Received: 5455
Timestamp: 112379110 SeqCnt: 5454 Diff to last: 624
Dest: FFFFFFFF Src: FCC23D0DD471 Type:88A8 DataLen:1104
Num Raw Frame Received: 6064
Timestamp: 122031700 SeqCnt: 6063 Diff to last: 609
Dest: FFFFFFFF Src: FCC23D0DD471 Type:88A8 DataLen:1056
Num Raw Frame Received: 6675
Timestamp: 132555480 SeqCnt: 6674 Diff to last: 611
Dest: FFFFFFFF Src: FCC23D0DD471 Type:88A8 DataLen:694
Num Raw Frame Received: 7278
Timestamp: 142125530 SeqCnt: 7277 Diff to last: 603
Dest: FFFFFFFF Src: FCC23D0DD471 Type:88A8 DataLen:460
^Cpi@fcpi2:~/develop/09_Redmine5/broadway_v2/trunk/source_python/app $

```

7.9.3 raw_loop

python raw_loop_internal.py

This sample demonstrates complete handling to control NXP Phy using loopback mode and send/receive data using Broadway2 raw-API. In addition, demonstrates the precise timestamping for getting TX and RX timestamps direct from the LPC43333 MAC.

The measurement and tests are done here using new Nitrogen i.MX8M-EVB from Boundary Devices running Ubuntu-Bionic-x64.

```

ubuntu@bionic-dev64: ~/develop/09_Redmine5/broadway_v2/trunk/source_python/app/pu...
bllc/raw$ sudo python3 loop_local.py
submitted frame with frameId: 1
transmission completed: 1 / 101879.610292400
Received Frame: 1530
received: 0 time to send/receive:1360.0 ns
ubuntu@bionic-dev64:~/develop/09_Redmine5/broadway_v2/trunk/source_python/app/pu
bllc/raw$ sudo python3 loop_local.py
submitted frame with frameId: 1
transmission completed: 1 / 101885.782764760
Received Frame: 1530
received: 0 time to send/receive:1320.0 ns
ubuntu@bionic-dev64:~/develop/09_Redmine5/broadway_v2/trunk/source_python/app/pu
bllc/raw$ sudo python3 loop_local.py
submitted frame with frameId: 1
transmission completed: 1 / 101888.915673760
Received Frame: 1530
received: 0 time to send/receive:1360.0 ns
ubuntu@bionic-dev64:~/develop/09_Redmine5/broadway_v2/trunk/source_python/app/pu
bllc/raw$

```

7.10 USB SMI/SPI Adapter-Only

Python samples can be found in sub-folder tool. There are 2 Python classes to be used on the FC401:

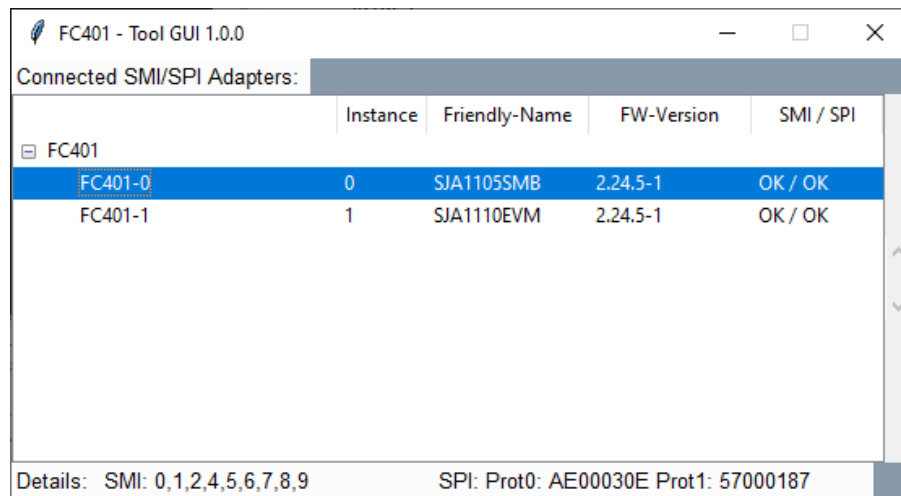
- SMIAccess to communicate with external NXP Phy Components like TJA1100, TJA1101 and TJA1102 using MDIO/MDC
- SpiAccess to communicate with external NXP Switch components using SPI in 32-bit block-mode

Both interfaces are available in parallel. If SPI and SMI is connected, Phy and switch components can be configured and tested.

7.10.1 tool_gui

This sample lists all available tool adapters FC401 and tries to read data on SMI and SPI interface.

To launch, start using `python tool_gui.py` on Windows or `sudo python3 tool_gui.py`



The screenshot shows a window titled "FC401 - Tool GUI 1.0.0". Below the title bar, it says "Connected SMI/SPI Adapters:". There is a table with the following columns: "Instance", "Friendly-Name", "FW-Version", and "SMI / SPI". The table lists two adapters under the "FC401" category:

Instance	Friendly-Name	FW-Version	SMI / SPI
0	SJA1105SMB	2.24.5-1	OK / OK
1	SJA1110EVM	2.24.5-1	OK / OK

At the bottom of the window, there is a status bar with the text: "Details: SMI: 0,1,2,4,5,6,7,8,9 SPI: Prot0: AE00030E Prot1: 57000187".

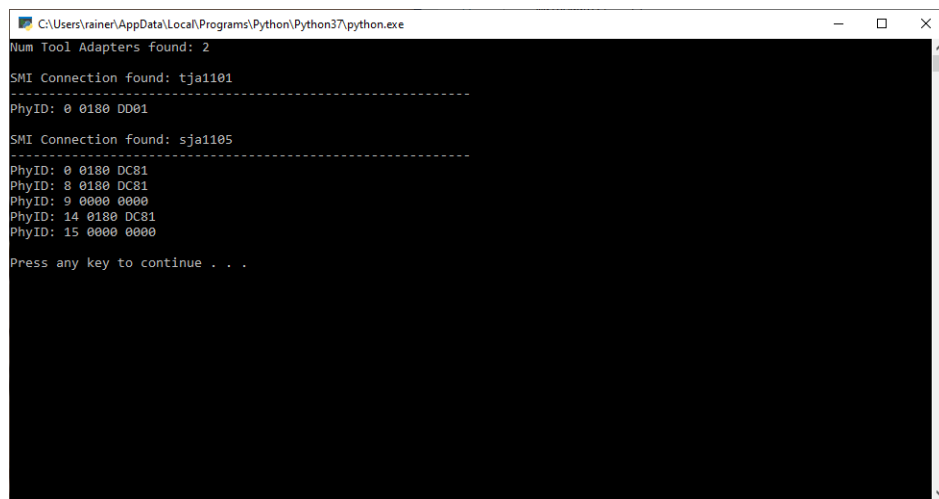
In this example, there are two FC401 connected to Windows Host. One is named as tja1101 which is connected to NXP TJA1101 Evaluation board. The second one is connected to SJA1105Q-EVB which includes 2 TJA1102 and one SJA1105Q. For the SMI, the tool found successful reads on PHYADDR 0, 8,9,14 and 15. On SPI, there are 2 modes supported for SPI protocol. As

the FC401 can't detect used mode on SPI-slave, this tool just reads both values by changing the protocol.

7.10.2 smi_access

`python read_phyid.py`

This sample demonstrates SMI access to external NXP Phys using MDIO/MDC of FC401.

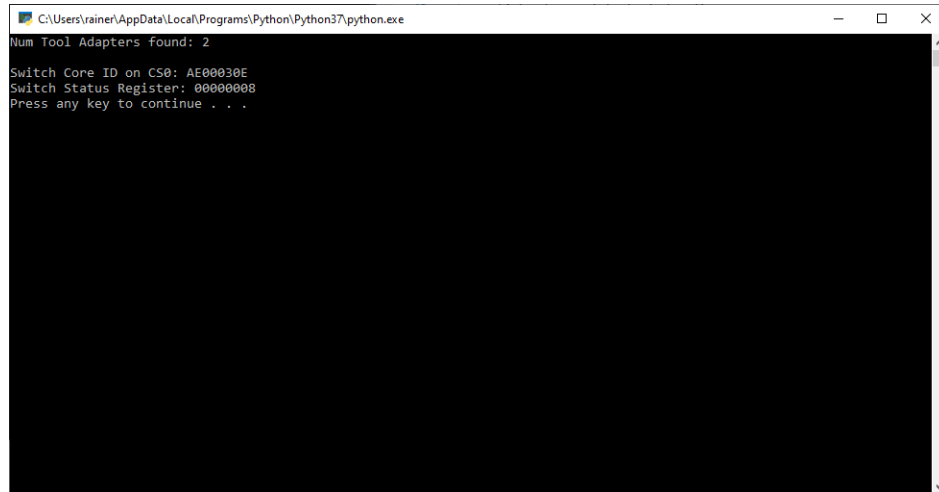


```
C:\Users\rainer\AppData\Local\Programs\Python\Python37\python.exe
Num Tool Adapters found: 2
SMI Connection found: tja1101
-----
PhyID: 0 0180 DD01
SMI Connection found: sja1105
-----
PhyID: 0 0180 DC81
PhyID: 8 0180 DC81
PhyID: 9 0000 0000
PhyID: 14 0180 DC81
PhyID: 15 0000 0000
Press any key to continue . . .
```

7.10.3 spi_access

`python read_deviceid.py`

This sample demonstrates 32-bit SPI access to external NXP Switch like SJA1105.



Screenshot of console running on MS-Windows 10.

Find short Python sample to show how easy SMI/SPI-access to NXP Eval-Boards can be implemented.

For easier usage, there are helper functions available for creating correct SPI or Phy-Access.

- `get_phy_access(a, phyAddr)` : returns instance of correct `PhyAccess`
- `get_spi_access(a, protocol, speed)` : returns instance of correct `SpiAccess` with given parameters
- `get_switch_access(spi, cs)` : return instance to `SJAXAccess` with given `cs`

generic example using `SmiAccess` to access all connected Phys

```
from broadway2 import Broadway, SpiAccess, SmiAccess, PhyAccess, SJAXAccess
from broadway2 import get_phy_access, get_spi_access, get_switch_access
```

```
base = Broadway()
```

```
aList = base.get_tool_adapters()
```

```
print('Num SMI/SPI Adapters found: {0}'.format(len(aList)))
```

```
print()

for a in aList:
    inst, friendlyName = a

    adapter = base.get_adapter(0)

    dev = base.open_control(inst)

    adapter.connect(dev) # connect adapter instance with USB-Control device

    spiAccess = get_spi_access(adapter)

    sjaAccess = get_switch_access(spiAccess, 0)
    # option for second switch on CS=1
    sjaAccess2 = get_switch_access(spiAccess, 1)

    switchCoreId = sjaAccess.read_reg(0)
    # read deviceId on second switch
    switchCoreId2 = sjaAccess2.read_reg(0)

    phyAccess4 = get_phy_access(adapter, 4)

    phyId1 = phyAccess4.read_reg(2) # PhyID register 1 on SmiAddr=0
    phyId2 = phyAccess4.read_reg(3) # PhyID register 2 on SmiAddr=0

    # finally close device
    base.close(dev)
```