Installing CIFX driver under linux Ubuntu [EN]

Global Support

Exported on 01/28/2023

Table of Contents

1 Installing NXDRV-Driver on Linux Ubuntu

This installing guide provides the installation requirements.

1.1 Overview

⚠ Support for Linux kernel >= 5.6

The cifX Linux driver runs as a library in userspace and accesses the card via a UIO kernel module (userspace I/O).

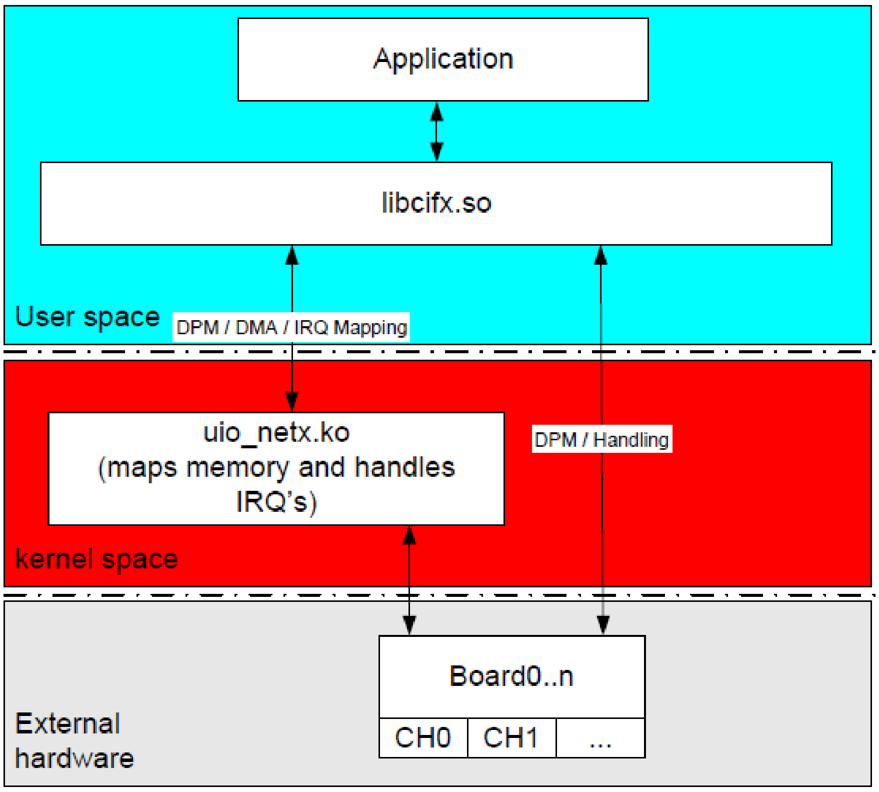


Figure 1: Linux cifX driver architecture (Linux DRV 13.EN)

1.2 Requirements

- cmake (min 2.8.9)
- libthread, librt
- libapciaccess-dev (tested with V0.10.2 / V0.13.1-2)
  - always needed for cifx PCI cards

1.3 Installing NXDRV-Driver

Install **cmake**.

```
$ apt-get install cmake
```

Install **libpciaccess-dev**.

```
$ apt-get install libpciaccess-dev
```

Extract the sources from the zip-file.

```
$ tar xf <driver_version>.tar.gz
```

Change to the driver directory.

```
$ cd <driver_version>/driver/
```

Run the installation script.

```
$ ./build_install_driver [optional pass the build folder]
```

Then follow the installation instructions

Deactivate the SPI interface if you do not use it.

In case of successful installation, the driver is ready to use.

In case of a system reboot, the kernel driver needs to be reloaded

```
$ modprobe uio_netx
```

1.4 Compiling the example programs

Go to the *examples* folder.

```
$ cd <driver_version>/examples
```

(Optional) Create a build directory.

```
$ mkdir build && cd build
```

Run cmake.

```
$ cmake ..
```

Build all source modules

```
$ make all
```

1.5 Configuration file storage

For a cifX device (especially CIFX 50 and CIFX 90), firmware and configuration files that are not stored on the hardware, must be loaded into the hardware each time the card is powered-up.

This part describes where and how these files have to be stored.

To allow device-specific configuration, the card needs to be identified, and the firmware must be stored on the host in a specific folder structure to create a unique relation between card and configuration.

These folders locate under a global base-folder. By default, it is '**/opt/cifx**' (can be changed during driver initialization).

There are four different types of configuration of a card, each with its specific folder structure. For more information, please see our Linux driver manual[1].

This guide describes the folder structure single directory.

Go to the *scripts* directory.

```
$ cd <driver_version>/driver/scripts
```

Run the installation firmware script.

```
$ ./install_firmware create_single_dir
```

This script will create a basic directory, it locates by default under */opt/cifx/deviceconfig/FW/channel0*

1.6 Installing driver module for the CIFX/M.2 netx90 based devices.

1.6.1 Building ax99100-pci-spi module

For using the CIFX/M.2 the **ax99100-pci-spi** module must be compiled and loaded to the kernel.

First, enable the **SPM_PLUGIN** option during compiling the driver. See Configuration file storage(see page 6).

⚠ If you are using linux driver version **V2.1.1.0 or lower**, the kernel module **ax99100-pci-spi** does not compile out of the box.
If you get the following compiler error: *-Werror=incompatible pointer-types* located in **moduleparam.h**, please change in **ax99100-pci-spi.c** the following calls of **module_param_named** from **ushort** to **uint**.

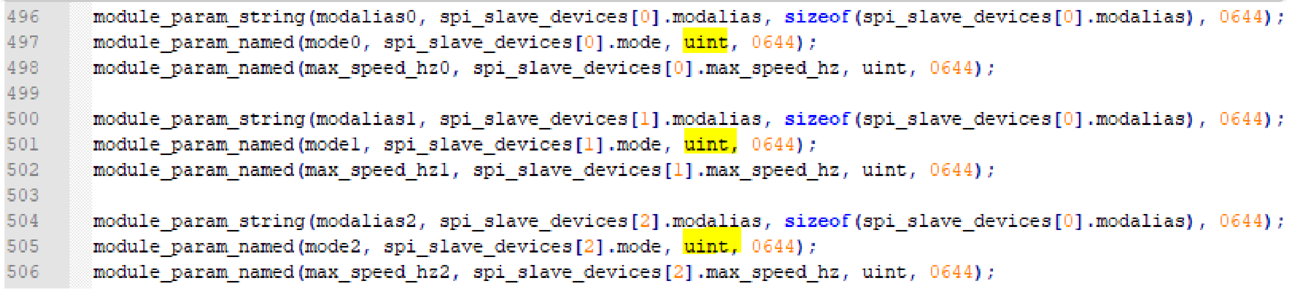
Go to cifx_m2 folder.

1 https://kb.hilscher.com/pages/viewpage.action?pageId=151627088

```
$ cd <driver_version>/driver/cifx_m2/
```

Change in **ax99100-pci-spi.c** (with e.g nano) the parameter in line 497, 501 and 505 from **ushort** to **uint**.

```
$ nano ax99100-pci-spi.c
```



```
496   module_param_string(modalias0, spi_slave_devices[0].modalias, sizeof(spi_slave_devices[0].modalias), 0644);
497   module_param_named(mode0, spi_slave_devices[0].mode, uint, 0644);
498   module_param_named(max_speed_hz0, spi_slave_devices[0].max_speed_hz, uint, 0644);
499
500   module_param_string(modalias1, spi_slave_devices[1].modalias, sizeof(spi_slave_devices[0].modalias), 0644);
501   module_param_named(mode1, spi_slave_devices[1].mode, uint, 0644);
502   module_param_named(max_speed_hz1, spi_slave_devices[1].max_speed_hz, uint, 0644);
503
504   module_param_string(modalias2, spi_slave_devices[2].modalias, sizeof(spi_slave_devices[0].modalias), 0644);
505   module_param_named(mode2, spi_slave_devices[2].mode, uint, 0644);
506   module_param_named(max_speed_hz2, spi_slave_devices[2].max_speed_hz, uint, 0644);
```

Then, in Makefile uncomment line 5

#obj-m += ax99100-pci-gpio.o

You cannot use Interrupt for now!

Install driver modules. This will copy the driver modules into the desired kernel library folder, which are compiled for.

```
$ make modules_install
```

Update the module dependencies in the modules.dep file.

```
$ depmod
```

Configure the SPI interface (at chip-select 0) for accessing the netX90 based cifX/M.2 card with mode 3 and a maximum speed of 25Mhz.

```
$ modprobe ax99100-pci-spi modalias0=spi-petra mode0=3 max_speed_hz0=25000000
```

(Optional) You can verify with **dmesg** if the **ax99100-pci-spi** module is successfully added to the kernel.

Run:

```
$ dmesg | grep "ax99"
```

You should see something similar to the following:

```
[  208.376412] ax99100-pci-spi: AX99100 PCIe to Multi I/O Controller - SPI driver
[  208.386582] ax99100-pci-spi 0000:06:00.3: spi0 successfully initialized!
[  208.386638] ax99100-pci-spi 0000:06:00.3: SPI slave (cs=0, modalias=spi-petra, 
mode=3, max_speed_hz=25000000) successfully added.
```

(Optional) Check if spi device is created.

```
$ ls /dev/spi*
```

Terminal output:

```
/dev/spidev0.0
```

1.6.2 Test

Then run a simple cifx example for the initial sanity check:

Go to the *example* folder and build the example programs. Please see Compiling the example programs(see page 5).

Then run the *cifxtcpserver* with *-a* as parameter to detect the installed CIFX/M.2 card.

An example:

```
$ cd <driver_version>/examples/build
$ ./cifxtcpserver -a
```

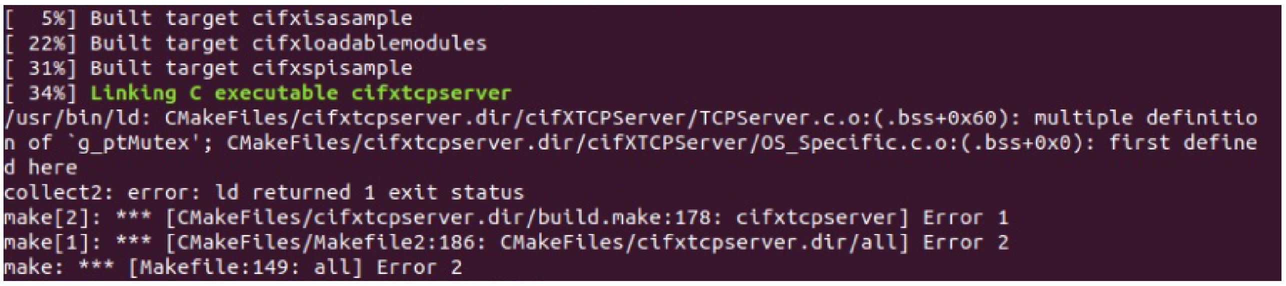
If a firmware (e.g. PROFINET) was downloaded to the device, you should get a similar output:

```
---------- Available Cards ----------
1.: cifX0
    DeviceNumber : 1443100
    SerialNumber : 20418
    - Channel 0:
      Firmware : PROFINET IO Device
      Version  : 5.4.3 build 0
    - Channel 1:
      Firmware : Network Services
      Version  : 1.0.0 build 0
----------------------------------------------------
```

1.7 Issues during build process

While compiling **cifxtcpserver** example, this error appears.

Error: multiple definition of 'g_ptMutex'



```
[  5%] Built target cifxisasample
[ 22%] Built target cifxloadablemodules
[ 31%] Built target cifxspisample
[ 34%] Linking C executable cifxtcpserver
/usr/bin/ld: CMakeFiles/cifxtcpserver.dir/cifXTCPServer/TCPServer.c.o:(.bss+0x60): multiple definitio
n of `g_ptMutex'; CMakeFiles/cifxtcpserver.dir/cifXTCPServer/OS_Specific.c.o:(.bss+0x0): first define
d here
collect2: error: ld returned 1 exit status
make[2]: *** [CMakeFiles/cifxtcpserver.dir/build.make:178: cifxtcpserver] Error 1
make[1]: *** [CMakeFiles/Makefile2:186: CMakeFiles/cifxtcpserver.dir/all] Error 2
make: *** [Makefile:149: all] Error 2
```

Figure 2: Error while compiling

Quick solution: In **OC_Specific.c** located under *<driver_version>examples/cifXTCPServer/*, line 32 add **extern** before pthread_mutex_t* g_ptMutex.

```
extern pthread_mutex_t* g_ptMutex;
```

1.8 Recommended content

- How to connect a CIFX Card in a Linux System to Sycon.net via TCP Server[2]
- Commissioning CIFX M223090AE as Profinet IO Device[3]
- Which Linux driver version do i need to use M.2 netX 90 based CIFX cards?[4]

2 https://kb.hilscher.com/display/GLOBALSUP/How+to+connect+a+CIFX+Card+in+a+Linux+System+to+Sycon.net+via+TCP+Server
3 https://kb.hilscher.com/display/GLOBALSUP/Commissioning+CIFX+M223090AE+as+Profinet+IO+Device
4 https://kb.hilscher.com/pages/viewpage.action?pageId=124831525