FELIX User Manual

ATLAS FELIX Group

Version 5.1.0-0-g6d9c9af-dirty

Table of Contents

0 :!table: 0

1. Welcome to the FELIX User Manual

1.1 Overview

This document is intended to support all users of the Phase-II FELIX readout infrastructure with installation, maintenance and operation of their system. The document covers all aspects of the FELIX system from recommended hardware to firmware and driver installation and maintenance. Finally the full suite of FELIX software will be presented, including useful test tools leading up to the primary 'felix-star' readout application. For more information users should consult the following locations for updates:

The FELIX users mailing list:

atlas-tdaq-felix-users@cern.ch

The FELIX Project Website:

https://atlas-project-felix.web.cern.ch/atlas-project-felix

The FELIX release distribution site:

https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/releases.html

This page includes compatibility information for different, firmware, software, driver and user manual versions. User support requests from users to the FELIX team should be made via the dedicated JIRA project:

https://its.cern.ch/jira/projects/FLXUSERS

Please report any broken links of obsolete material to help improve the overall quality of our documentation.

0 :!table: 1

2. Introduction to FELIX

FELIX is a detector readout component developed as part of the ATLAS upgrade effort. FELIX is designed to act as a data router, receiving packets from detector front-end electronics and sending them to programmable peers on a commodity high bandwidth network. Whereas previous detector readout implementations relied on diverse custom hardware platforms, the idea behind FELIX is to unify all readout across one well supported and flexible platform. Rather than the previous hardware implementations, detector data processing will instead be implemented in software hosted by commodity server systems subscribed to FELIX data. From a network perspective FELIX is designed to be flexible enough to support multiple technologies, including TCP/IP and RoCE.

The FELIX readout system is based on a custom PCIe "FELIX" card hosted on a commodity server. Front-end data is transferred from the FELIX card into the host memory into a so-called direct-memory-access (DMA) buffer. The host forwards the data to remote clients over a commodity network. Data can also flow in the opposite direction, from a remote client to a front-end.

2.1 FELIX Variants and Functionality

FELIX supports different link protocols for the transfer of data to and from front-end peers. Each is supported by the same hardware platform, with separate firmware revisions both based on the same core modules. A complete description of all the supported protocols and firmware flavours can be found in the Phase-II FELIX firmware specification document

---

2.1.1 Gigabit Transceiver (GBT) and the Versatile Link

The Gigabit Transceiver (GBT) chipset and associated technologies were developed as part of CERN's Radiation Hard Optical Link Project. GBT provides an interface an optical connectivity technology known as the Versatile link, which provides a radiation hard transport of data between GBT end points. The GBT transmission protocol is designed to aggregate multiple lower bandwidth links from front-end electronics components into one radiation hard high bandwidth data link (running at up to 5 Gb/s). The logical lower bandwidth links which make up a GBT link are known as E-links. The GBT protocol has been implemented both in the dedidated GBTX ASIC as well as directly on FPGA platforms, the latter of which has been built on for use by the FELIX project.

2.1.2 Low Power Gigabit Transceiver (lpGBT)

LpGBT is the evolution of the GBTX ASIC. LpGBT supports 2.56 Gb/s downlinks (from the readout system to the front-end) and 5.12 or 10.24 Gb/s uplinks (from the front-end to the readout system). Similarly to the GBT protocol, LpGBT also defines E-links.

2.1.3 FULL Mode

Within the context of the Phase-I ATLAS upgrade a requirement arose for a higher bandwidth data

link from detector to FELIX than was possible with GBT. These newer clients did not require radiation hardness, and were able to support a protocol which could be implemented in FPGAs on both sides of the link. The resulting development is known as 'FULL mode', referring to full link bandwidth.

The FULL mode protocol is a implemented as a single wide data stream with no handshaking or logical substructure (i.e. no E-links). The reduced constraint mean that FULL mode links can operate at a line transmission rate of 9.6 Gb/s, which accounting for 8b10b encoding means a maximum user payload of 7.68 Gb/s.

In the downstream direction FULL mode relays clocks and messages from the Local Trigger interface (LTI). LTI also operates at 9.6 Gb/s and is 8b10b encoded. The 40.079 MHz LHC BC clock is recovered from the link 240.474 MHz clock.

The Phase-I (rm-4) version of FULL-mode implemented GBT downlinks that transmitted the legacy TTC signal using a custom encoding.

2.1.4 Interlaken Mode

FELIX supports the Interlaken protocol [6] for transmission rates up to 25 Gb/s from the detector. From a functional point of view, Interlaken is similar to FULL Mode as links do not have a logical substructure. Downlinks are rated at 9.6 Gb/s and relay LTI messages.

2.1.5 ATLAS ITk Pixel and Strip

The ATLAS Phase-II Inner Tracker (ITk) adopted various data encodings, including Aurora and Endeavour. Dedicated firmware have been developed to support ITk pixel and strips sub-detectors.

0 :!table: 2

3. Hardware Setup

The FELIX readout system relies on a custom PCIe card. The card developed for the ATLAS Phase-I upgrade is called FLX-712 Phase-II cards are the FLX-182 (prototype) and FLX-155 (production card). The Xilinx VC-709 development board can also be used as FELIX card, but firmware support is being phased out.

Setting up PCIe bifurcation in the host BIOS is required for several configurations. Pay attention to PCIe bifurcation settings when mentioned in the text.

3.1 FLX-712

The FLX-712 card hosts a Xilinx® Kintex® UltraScale FPGA, capable of supporting 48 high speed optical links via MiniPOD transceivers, and a 16-lane PCIe Gen 3.0 interface. FLX-712 interfaces to the ATLAS Phase-I TTC system using a removable mezzanine. FLX-712 was produced also for Phase-II tests and can run all Phase-II firmware flavours but cannot interface with LTI nor support 25 Gb/s links. An image of the FLX-712 and its key features are presented below. More details, including the fibre mapping, can be found in the FLX-712 hardware manual.

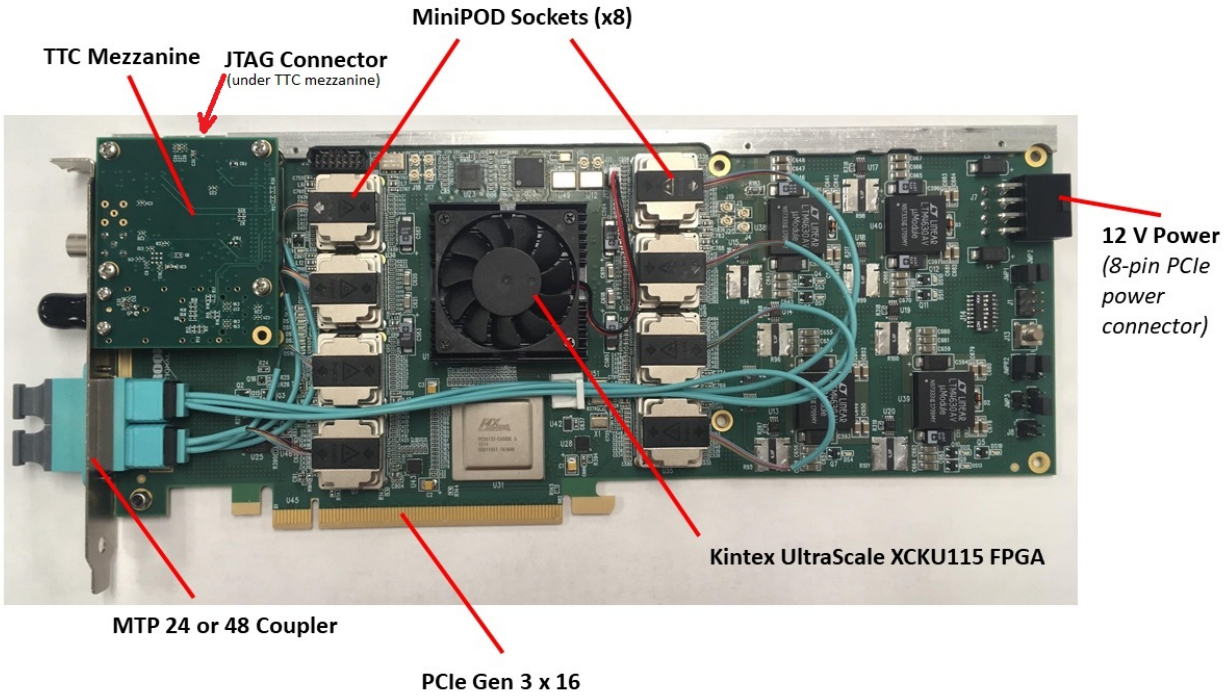

Figure 3.1 A 48-link FLX-712 card.

3.1.1 Installation

The FLX-712 should be installed in a 16-lane Gen >=3 PCIe slot on the host motherboard. The FLX-712 is equipped with a PCIe bridge which turns the two x8 PCIe endpoints supported by the FPGA into a single x16 device. This turned out to be a bottleneck for higher throughput, so the PCIe bridge has been set up in a transparent mode, exposing the two x8 endpoints directly. This has the implication that the PCIe slot must be set into x8+x8 bifurcation mode from the host BIOS setup. If this setting is not available for the host PC, the card can be set back into x16 PCIe mode by setting

the jumper J14:8 into "ON" position, the default production position is "OFF: bifurcation".

Table 3.1 The following firmware and hardware combinations use bifurcation

Version	FLX-709	FLX-711	FLX-712	FLX-182	FLX-155
rm4 < 4.14_35	x8	x16	x16	N/A	N/A
rm4 >= 4.14_35	x8	x16	x8+x8*	N/A	N/A
rm5 < 5.1_343	x8	N/A	x16	x8+x8	x8+x8
rm5 >= 5.1_343	x8	N/A	x8+x8*	x8+x8	x8+x8

From 4.14_35 and 5.1_343, the FLX712 can be set into x16 mode (no bifurcation) by setting the dipswitch J14:8 in "ON" position

The board must be connected to power from the system's internal ATA power supply via an 6-pin (recommended) or 8-pin PCIe power connector (of the type commonly used for graphics cards). Note that the board does not support use of Xilinx power connectors.

The FLX-712 provides a JTAG connector to which programmers can be connected for FPGA configuration. The Digilent® HS2 programmer is recommended for this purpose. While this programmer fits comfortably into the 4U chassis, the 2U chasses will need an additional flexible adapter. For the 2U chassis, the programmer can be used with a flexible cable fabricated from the Digilent XUP flywire assembly and a pin header.

Some card management features, such as programming the FPGA from FLASH (see Section 6.4.9), make use of the host motherboard's system management bus (SMBus) interface to on-card I2C. Please consult your motherboard manual and ensure that this feature is enabled (typically done by setting a jumper) in order to have full functionality.

3.2 FLX-182

FLX-182 is a Phase-II prototype card developed by BNL. FLX-182 is equipped with a SoC AMD Versal Prime VP-1802, FireFly transceivers that provide up to 24 25 Gb/s duplex links, LTI iterface, an electrical trigger interface and PCIe gen4. More details can be found in the I/O Card Hardware specification document.

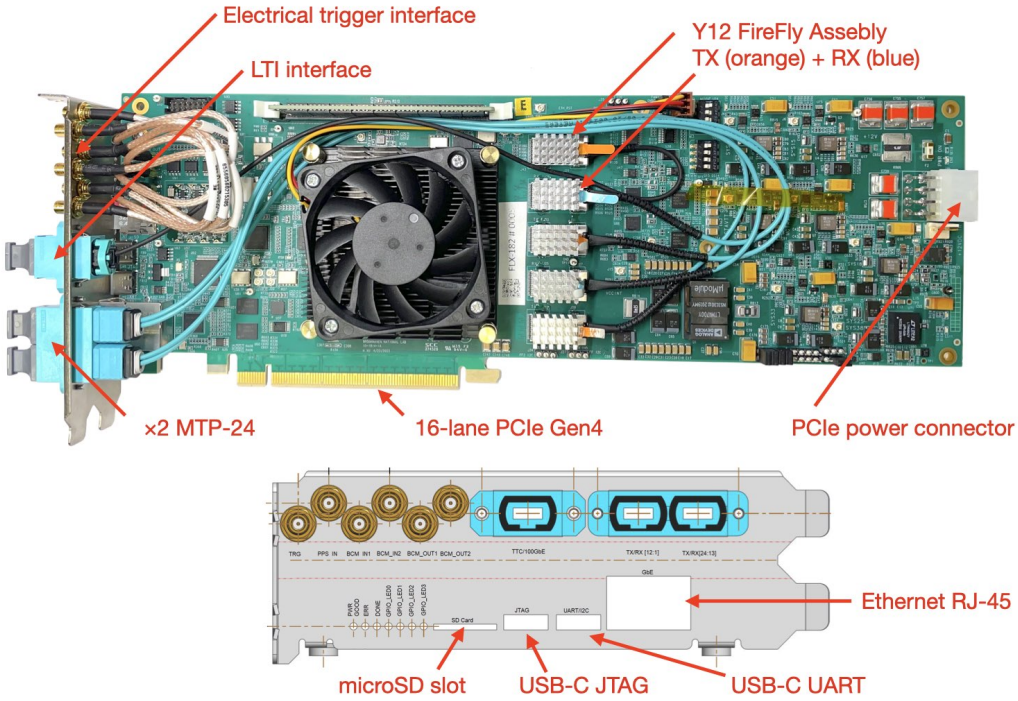

Figure 3.2 The FLX-182 card.

3.2.1 Installation

The FLX-182 card requires a 16-lane PCIe slot (preferably Gen4) configured as x8+x8 for all firmware flavours. One PICe 6+2 pin cable is required to power up the device. Contrarily to FLX-712, the PCIe bracket requires two adjacent PCIe slots. The card is provided with a micro-SD card containing the Petalinux OS for the SoC processing system and an image of the programmable logic (PL). The PL can be reprogrammed changing the content of the SD card or using the JTAG controller embedded on the board accessible via the USB-C port on the front panel.

3.3 FLX-155

FLX-155 is the candidate FELIX card for Phase-II ATLAS. FLX-155 uses an AMD Versal Premium VP-1552 SoC, supports PCIe gen5, and can serve up to 48 duplex links each rated up to 25 Gb/s via FireFly Y12 optical transceivers. One FireFly B04 module provides the LTI interface and a second optional one a 400 GbE interface (not used by ATLAS). More details can be found in the I/O Card Hardware specification document.

3.3.1 Installation

FLX-155 has the same physical dimensions of FLX-182 and it requires a 16-lane PCIe slot (possible Gen5) configured as x8+x8 for all firmware flavours. Al least one PICe 6+2 pin cable is required to power up the device. Similarly to FLX-182, FLX-155 is provided with an SD card containing both the Petalinux and PL images.

3.4 VC-709 (Commodity Platform)

The hardware platform initially recommended for small scale tests is based on the Xilinx® VC-709

Connectivity Kit. This platform provides 4 optical transceivers as well as a Xilinx® Virtex®-7 series FPGA and 8-lane PCIe Gen 3.0 interface. Because this card has only 8 lanes, PCIe bifurcation can be set to Auto or x8x8. All FELIX firmware variants are built for VC-709, but VC-709 does not support 25 Gb/s links. LTI can be connected using one of the four optical transceivers, thus reducing the data links to three. The Phase-I TTC interface for the system is provided by the TTCfx v3 FMC mezzanine card, An image of the VC-709 board and guide to features is presented below.

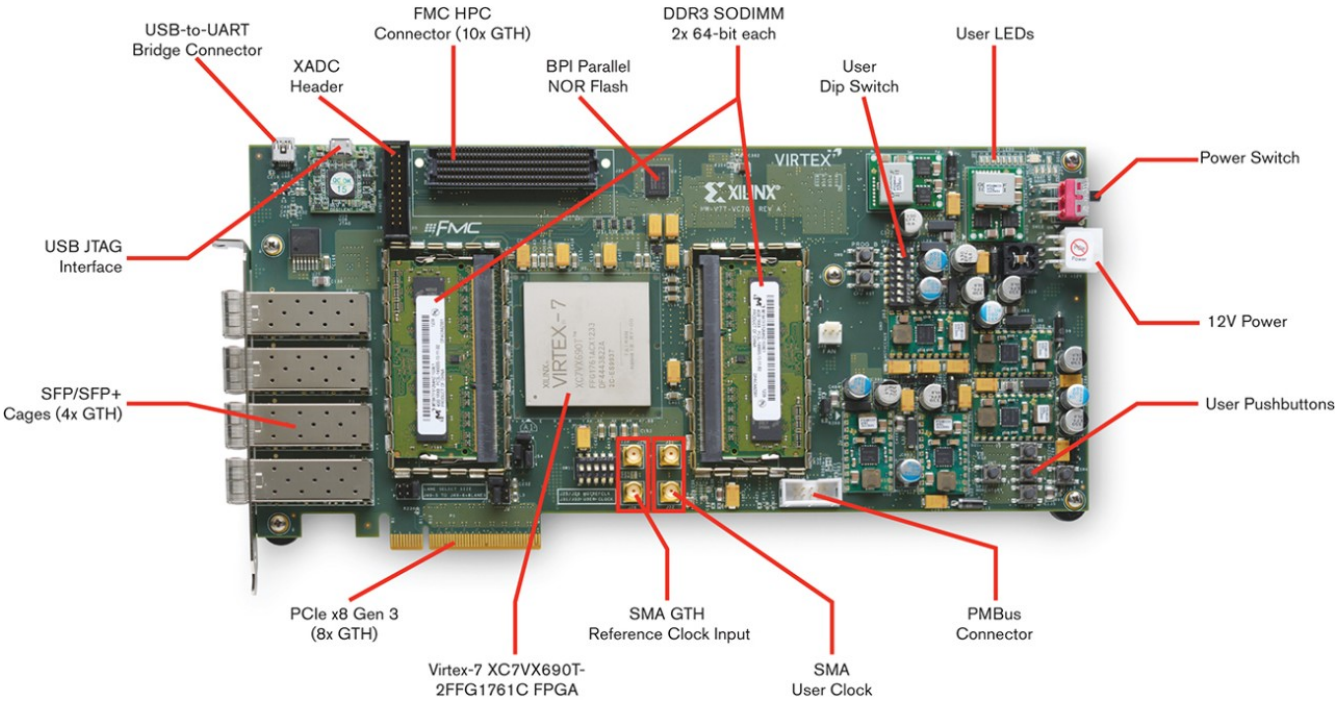

Figure 3.3 The VC-709 development board.

3.4.1 Installation of VC-709

For full details regarding the VC-709 please consult the manual provided with your equipment. In terms of installing the card into a FELIX system please follow the following guidelines. The VC-709 should be installed into an 8-lane or 16-lane Gen 3 PCIe slot on the host motherboard, taking into account the need for clearance on all sides. The board must be connected to power from the system's internal ATA power supply via a custom Molex adapter provided with the board. The power socket on the board is shown on the upper right hand corner of Figure 3.3 , labelled '12V Power'. Ensure that the power switch, just above the socket, is switched to the on position.

The FPGA aboard the VC-709 is configured via an on-board JTAG programmer, which can be connected to a mini-USB cable with the 'USB JTAG Interface' on the top left of Figure 3.3 . A right angled mini-USB connector is recommended to minimise obstruction of the hosts case lid, although a straight cable is provided for free with your kit. Note that this has currently only been tested for USB2, which is the recommended interface. In order to be able to program the card please connect it to a convenient USB port on your host machine, or to another machine which you wish to use as a programming server. Finally, ensure that the link transceivers are safely inserted into the on-board cages.

3.5 FELIX Host PC

The host server must provide at least one PCI-e 6+2 pin power connector (same as used for GPUs) and one x16 PCIe slot connected to the CPU. For FLX-182 and FLX-155 the chassis must offer enough space in the surrounding of the MOLEX power socket: both cards have full-height full-length (FHFL) format and are 312 mm long. Recommended requirements are listed below.

Component	Model
CPU	AMD EPYC 9004 series or 5th Gen Intel Xeon
Memory	at least 32 GB
Network card	Nvidia/Mellanox Connect-X5 or superior

Tested FELIX hosts are described at https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/serverhw.html

3.6 FELIX drivers

The latest recommended version of the FELIX driver is available on the release distribution site.

In order to update the FELIX driver it will first be necessary to remove any existing driver installations from your system. To do this please follow the procedure outlined below. You will require superuser privileges in order to perform the driver de-installation itself and subsequent cleanup.

To check if a driver is already installed issue the following command:

```
rpm -qa | grep tdaq
```

If a driver rpm is installed you'll see a response along the lines of:

```
tdaq_sw_for_Flx-4.7.0-2dkms.noarch
```

To remove the driver do the following (substituting 'filename' for the results of the search in the previous step):

```
rpm -e filename
```

Once this operation is complete you will be in a position to install the latest FELIX driver. To install the FELIX driver RPM, run the following command (superuser privileges required):

```
dnf install tdaq_sw_for_Flx-4.9.0-2dkms.noarch.rpm
```

(this should take 1-2 minutes to complete, due to the need to compile the driver for your kernel as per the DKMS framework)

Once the driver is installed you should start it as follows (as superuser):

```
/etc/init.d/drivers_flx start
```

Once started you can check the status of the card using:

`cat /proc/flx`

You should see output similar to what is shown below (will vary depending on your firmware version(s) and the number of cards in your system; here one card is installed):

```
$ cat /proc/flx
FLX driver (ALMA9 ready) for FELIX release 4.14 (compatible with RM4 and RM5 F/W and
XVC). Based on tag felix-drivers-04-14-00, flx.c revision 01a9f00

Debug                          = 0
Number of devices detected     = 2


Locked resources
      device | global_locks
=============|=============
           0 |   0x00000000
           1 |   0x00000000

Locked resources
device | resource bit |     PID |  tag
=======|==============|=========|=====

Device 0: (BAR0 = 0xf6200000)
Card type                    : FLX-712
Device type                  : 0x0427
FPGA_DNA                     : 0x013a6f281c212245
Reg Map Version              : 5.1
GIT tag                      : rm-5.1
BUILD Date and time          : 16-7-2024 at 17h52
GIT commit number            : 494
GIT hash                     : 0x053ae4d6
Firmware mode                : FULL
Number of descriptors        : 5
Number of interrupts         : 8
Interrupt name  |ToHost 0|ToHost 1|ToHost 2|ToHost 3|reserved|CR Xoff |BUSY    |TH
full |
Interrupt count |    780 |      0 |      0 |      0 |      0 |      0 |      0 |
0 |
Interrupt flag  |      1 |      1 |      1 |      1 |      1 |      1 |      1 |
1 |
Interrupt mask  |      1 |      1 |      1 |      1 |      1 |      1 |      1 |
1 |
MSI-X PBA       00000000

XVC:

Device 1: (BAR0 = 0xf6600000)
Card type                    : FLX-712
```

```
Device type                : 0x0428
FPGA_DNA                   : 0x013a6f281c212245
Reg Map Version            : 5.1
GIT tag                    : rm-5.1
BUILD Date and time        : 16-7-2024 at 17h52
GIT commit number          : 494
GIT hash                   : 0x053ae4d6
Firmware mode              : FULL
Number of descriptors      : 5
Number of interrupts       : 8
Interrupt name  |ToHost 0|ToHost 1|ToHost 2|ToHost 3|reserved|CR Xoff |BUSY    |TH
full |
Interrupt count |    147 |      0 |      0 |      0 |      0 |      0 |      0 |
0 |
Interrupt flag  |      1 |      0 |      0 |      0 |      0 |      0 |      0 |
0 |
Interrupt mask  |      1 |      1 |      1 |      1 |      1 |      1 |      1 |
1 |
MSI-X PBA       00000000

XVC:

The command 'echo <action> > /proc/flx', executed as root,
allows you to interact with the driver. Possible actions are:
debug     -> Enable debugging
nodebug   -> Disable debugging
elog      -> Log errors to /var/log/message
noelog    -> Do not log errors to /var/log/message
swap      -> Enable automatic swapping of 0x7038 / 0x7039 and 0x427 / 0x428
noswap    -> Disable automatic swapping of 0x7038 / 0x7039 and 0x427 / 0x428
clearlock -> Clear all lock bits (Attention: Close processes that hold lock bits
before you do this)
```

Driver Flags

The `/proc/flx` interface makes it possible to toggle certain parameters by issuing the following command:

```
echo <action> > /proc/flx
```

By substituting `<action>` it is possible to do the following (only a selected list below):

- Enable/disable automatic re-ordering of FELIX cards to a more intuitive order w.r.t device type ('swap' or 'noswap').
- Clear all device locks with 'clearlocks'

The FELIX driver makes use of 'Dynamic Kernel Module Support' (DKMS) to automatically track kernel changes once installed. Users should therefore only need to change their installation if a new version of the driver itself is released.

3.7 Network Interfaces

FELIX software makes use of RDMA network technology, more specifically RoCE v2 (RDMA over Converged Ethernet version 2). To benefit of the full software capability it is recommended to install in the FELIX host an RDMA-capable network interface card (NIC).

NICs used by ATLAS in production during Run 3 are listed below. The 25 GbE NIC is the most affordable.

- dual-port 25 GbE NVIDIA ConnectX-5 EN SFP28 3.0 x8 (part number: MCX512A-ACAT)
- dual-port 100 GbE NVIDIA ConnectX-5 EN QSFP28 3.0 x16 (part number: MCX516A-CCAT)

NICs used for Phase-II development of the DAQ system, but rather expensive for users, are

- dual-port 200 GbE NVIDIA ConnectX-6 EN QSFP56 4.0 x16 (part number: MCX653106A-HDAT)
- dual-port 200 GbE (400 GbE) NVIDIA ConnectX-7 EN QSFP112 5.0 x16 OCP3.0 (part number: 900-9X760-0018-MB2/MCX753436MC-HEAB)

For RDMA to be working the NIC ports have to be active. This requires an optical or copper cable to be connected. A dual-port NIC can be connected to itself using short and cheap twinax copper cables such as MCP1600-C001 for 100 Gb/s or MCP2M00-A001 for 25 Gb/s.

3.7.1 Network drivers

It is recommended to install NVIDIA MLNX_OFED drivers. MLNX_OFED recompiled by the FELIX Team for the latest AlmaLinux9 kernels can be downloaded at https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/releases.html

To install the drivers untar the archive and run

```
sudo ./mlnxofedinstall --vma
```

The driver installation will take care of updating the network card firmware as well at need.

3.7.2 Network configuration

If the FELIX/Data Handler node is self-managed (i.e. not managed by ATLAS DAQ SysAdmins) the data network cards have to be assigned static IPs. This can be done as follows.

1. Identify the MAC address of the newly installed cards with `ifconfig -a`. A dual-port NIC has consecutive MAC addresses.
2. Create a device configuration file in `/etc/sysconfig/network-scripts/`. For example, to call the interface `priv0` create a file called `ifcfg-priv0` with content:

```
DEVICE=priv0
HWADDR=<paste MAC here>
TYPE=Ethernet
IPADDR=<your favour IP address e.g. 192.168.100.1>
NETMASK=255.255.255.0
ONBOOT=yes
MTU="1500"
IPV6INIT=no
```

1. A similar file, called e.g. `ifcfg-priv1` can be created to configure the second port of the NIC (if available) as interface `priv1`.

Advanced settings for both the network interface and a switch can be found in the Appendix H

3.8 TTC Systems

3.8.1 ATLAS Local Trigger Interface (LTI)

FLX-155 and FLX-182 can be connected to LTI using the dedicated MTP-12 connector which provides 4 duplex links mapped as shown in Figure 3.4 and Figure 3.5 . Only one duplex link is needed for LTI communication: by default link 0 is used. The LTI interface provides clock, LTI signals and allows FELIX to report BUSY. The LTI data format is described in the LTI Specification Document.

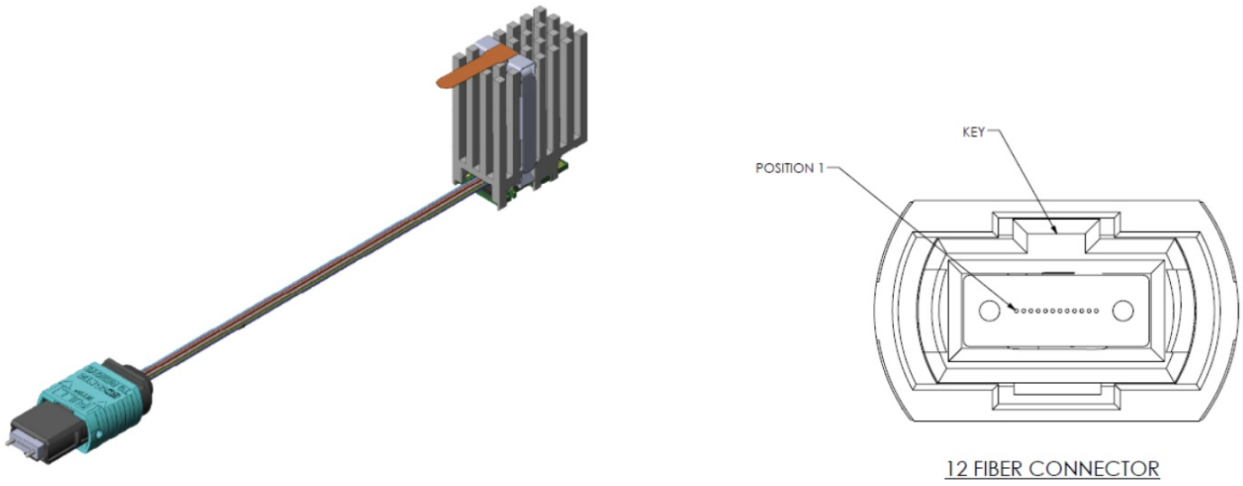

Figure 3.4 Fibre numbering on the MTP-12 connector used by the FELIX optical transceiver.

SIGNAL MAP				
LOAD SEQUENCE	FUNCTION	COLOR		MTP POSITION
1	Rx0	BLUE		1
2	Rx1	ORANGE		2
3	Rx2	GREEN		3
4	Rx3	BROWN		4
5	Dark	SLATE		5
6	Dark	WHITE		6
7	Dark	RED		7
8	Dark	BLACK		8
9	Tx4	YELLOW		9
10	Tx3	VIOLET		10
11	Tx2	ROSE		11
12	Tx1	AQUA		12

Figure 3.5 Fibre mapping of the 4-link transceiver on the MTP-12 connector.

It is possible to interface FLX-712 and FLX-709 to LTI as well using data links and custom firmware. No procedure has been defined so far. In case of interest open an FLXUSERS ticket.

3.8.2 Trigger via Electrical Interfaces

FLX-155 and FLX-182 are equipped with electrical connectors on the front panel that allow to provide a clock and a trigger signal. With reference to Figure 3.2 , for FLX-182, a NIM trigger signal can be connected to the TRG MMCX connector, while a clock signal must be connected to the PPSIN connector.

3.8.3 Legacy TTC System

The TTC system used between Run1 and Run 3 (ALTI, TTCVi) can be connected only to FLX-712 and FLX-709 using a Multi-Mode fibre (using a single-mode fibre from ALTI might require an attenuator). The setup of ALTI and TTCVi is covered in Appendix A.

FLX-712 comes with a dedicated ST connector for TTC and a LEMO connector to propagate the BUSY signal. The BUSY signal is the ATLAS standard open-collector BUSY signal, but with a weak 24 kOhm pull-up to 5 V to allow viewing on an oscilloscope.

VC-709 requires the TTCfx mezzanine card shown in Figure 3.6 . In addition to the ST connector, the mezzanine provides a LEMO for the propagation of BUSY. To install the mezzanine,connect the P and N SMA GTH Reference Clock inputs on the VC-709 (middle bottom of Figure 3.3) to the SMA connectors on the TTCfx v3 (P to P and N to N) via suitable SMA cables, [1]. The right angle side goes on the VC-709, to make the cable bending a bit more gentle. If you have space in your chassis, straight SMAs on both ends will do the job as well. The TTCfx mezzanine card requires no specific firmware programming, and should work out of the box once connected to a TTC peer.

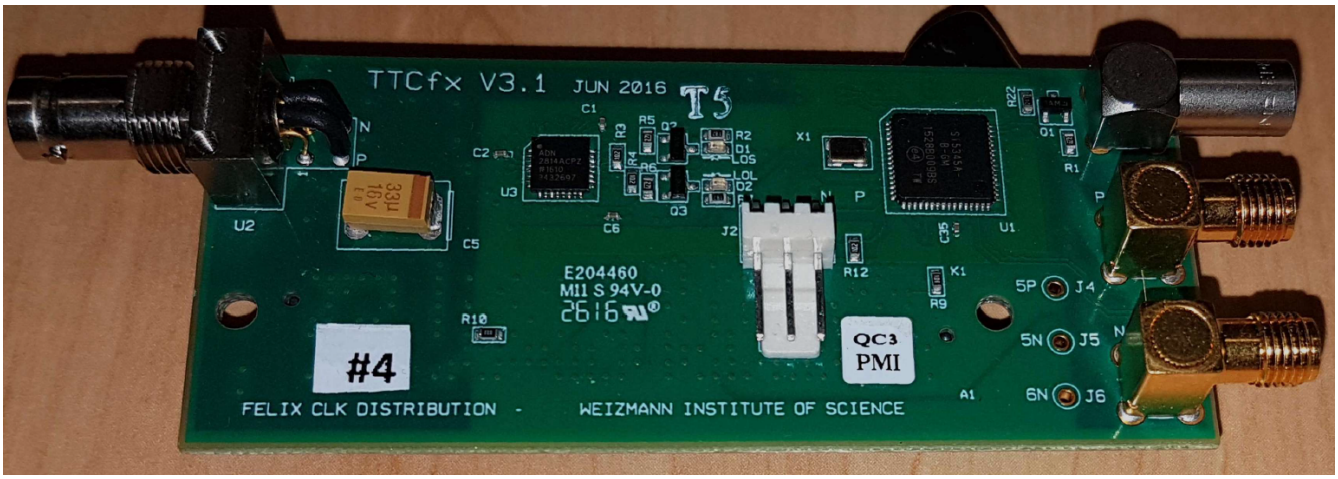

Figure 3.6 Image of a TTCfx v3 card.

0 :!table: 3 :numbering:

[1] An example SMA cable can be found here: https://eu.mouser.com/ProductDetail/Amphenol-RF/135103-01-0600?qs=sGAEpiMZZMv9ULLAfKm5f6h8NuxRC2dJ

4. Software Distribution

A FELIX release contains all that is needed to interact with any (programmed) FELIX card from the host PC. FELIX software is formally supported and built for systems using the AlmaLinux9 operating system.

New releases are announced on the following e-group:

atlas-tdaq-felix-users@cern.ch

The latest recommended version of the FELIX software suite is available on the release distribution site.

https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/releases.html

The FELIX software release is available pre-compiled as a tarball which can be installed anywhere and then set up for use by running a command line script. Each user can download their own version, or the release can be installed centrally and the location of the script shared with users.

To unpack the tarball, run the following command:

```
tar -xvzf <filename>
```

Once unpacked, a setup script gives access to all libraries and binary files. The script is sourced as follows:

```
source felix-05-01-00/x86_64-el9-gcc13-opt/setup.sh
```

This script will need to be run with every new session, or added to the environment setup procedure. Once complete you should have access to all FELIX software. In the next section we will describe how to test your installation to verify full functionality.

4.1 CVMFS

Installation of the latest and nightly versions of the FELIX software are available in CVMFS under:

```
/cvmfs/atlas-online-nightlies.cern.ch/felix/releases
```

```
/cvmfs/atlas-online-nightlies.cern.ch/felix/nightlies
```

to set up run:

```
source felix-05-01-00/x86_64-el9-gcc13-opt/setup.sh
```

:!table: 4 :numbering:

5. Firmware Releases and Programming

FELIX firmware (and software) releases are announced on the following e-group:

atlas-tdaq-felix-users@cern.ch

Releases, complete of changelogs, are advertised at:

https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/releases.html

The reference document for all firmware features and technical details is:

https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/docs/FELIX_Phase2_firmware_specs.pdf

firmware releases are labelled by the register map used. Firmware tagged rm-4.XY is for Phase-I. Phase-II firmware has tag >= rm5.0.

The FLX-709 builds are released for two hardware configurations. If the TTCfx3 mezzanine card is installed, the Si5345 on the mezzanine can be used, and the TTC clock can optionally be selected as an input clock through elinkconfig. Lacking a TTCfx3 mezzanine, another build supports the Si5324 jitter cleaner on the VC709 board. The version with Si5324 support can be recognized with the SI5324 keyword in the archive filename: FLX709_FULLMODE_4CH_CLKSELECT_GIT_master_rm-4.9_278_200619_14_13.tar.gz (TTCfx3 / Si5345 support) or FLX709_FULLMODE_4CH_CLKSELECT_SI5324_GIT_master_rm-4.9_278_200619_05_38.tar.gz (VC709 standalone / Si5324 support).

5.1 Programming a FELIX card

Reprogramming a FELIX card installed in host PC requires rebooting the host computer or executing a PCIe hotplug. The PCIe hotplug procedure is described in Sectio 5.1.1.

During a reboot PCIe devices remain powered. During a power-cycle, PCIe devices loose power. If a FELIX card looses power, the FPGA reprograms itself using the pre-defined persistent memory source (flash memory or the SD card on FLX-182/155).

5.1.1 PCIe hotplug procedure

If you wish to avoid rebooting your machine it is possible to rescan the PCIe bus. This procedure is not stable under all circumstances, and may produce inconsistent results. Having setup a FELIX release run:

```
pcie_hotplug_remove.sh
```

Then, rescan the PCIe devices with:

```
pcie_hotplug_rescan.sh
```

The last step also restarts the drivers.

5.2 Programming an FLX-182

Firmware for FLX-182 and FLX-155 is distributed in the form of files to be loaded on the SD-card provided with the card.

Both cards are equipped with an integrated JTAG programmer accessible via an USB-C connector on the front panel. The JTAG interface can be used to reprogram the programmable logic as described in Section 5.5. Nevertheless, reprogramming the PL might cause the operating system running in the PS to stop responding.

Both FLX-155 and FLX-182 implement an UART interface accessible via USB-C. The UART interface connect to the processing system and gives access to a AlmaLinux9 terminal over serial communication. To access the FLX-182 UART from a Linux PC connected via USB run `sudo screen /dev/ttyUSB1 115200 -cstopb`.

5.3 Programming an FLX-712

FLX-712 can be programmed from the host PC using the FELIX software tools `fflashprog` and `fflash` described in <6_basic_tools#sec:basictools>. The former loads an .mcs flash image file in one of the four memory partitions over the PCIe interface. Then, `fflash` programs the FPGA from the flash memory using I2C commands.

In case of power loss, the FPGA is reprogrammed using partition number 3. The default partition is set by jumpers as described in Appendix B

`fflash` relies on the SMBus to access the PCIe I2C bus. This technology\ must be supported by the motherboard or the SMC. Not all servers support this technology.

Alternatively, both the flash memory and the FPGA can be programmed via JTAG using Vivado as described in Section 5.5. An USB-JTAG programmer such as the Digilent® HS2 is necessary.

5.4 Programming an FLX-709

FLX-709 comes with an on-board JTAG programmer accessible via USB. The JTAG interface is the recommend way to reprogram FLX-709.

5.5 Programming FLX cards via JTAG using Vivado

Programming FELIX cards via JTAG requires Vivado or Vivado Lab. Both can be downloaded from the AMD website and Vivado Lab does not require a license. After the installation you need to install the cable drivers. On Linux:

```
cd ${vivado_install_dir}/data/xicom/cable_drivers/lin64/install_script/
install_drivers/
./install_drivers
```

The USB cable or programmer can then be used to connect a FELIX card to the host where Vivado is run. Vivado is started with

```
<installation path>/xilinx/Vivado/2024.1/bin/vivado
```

A similar path is used for `vivado_lab`. You will then be presented with the Vivado™ splash screen, where you should select 'Open Hardware Manager' as shown in the red box in Figure 5.1 .

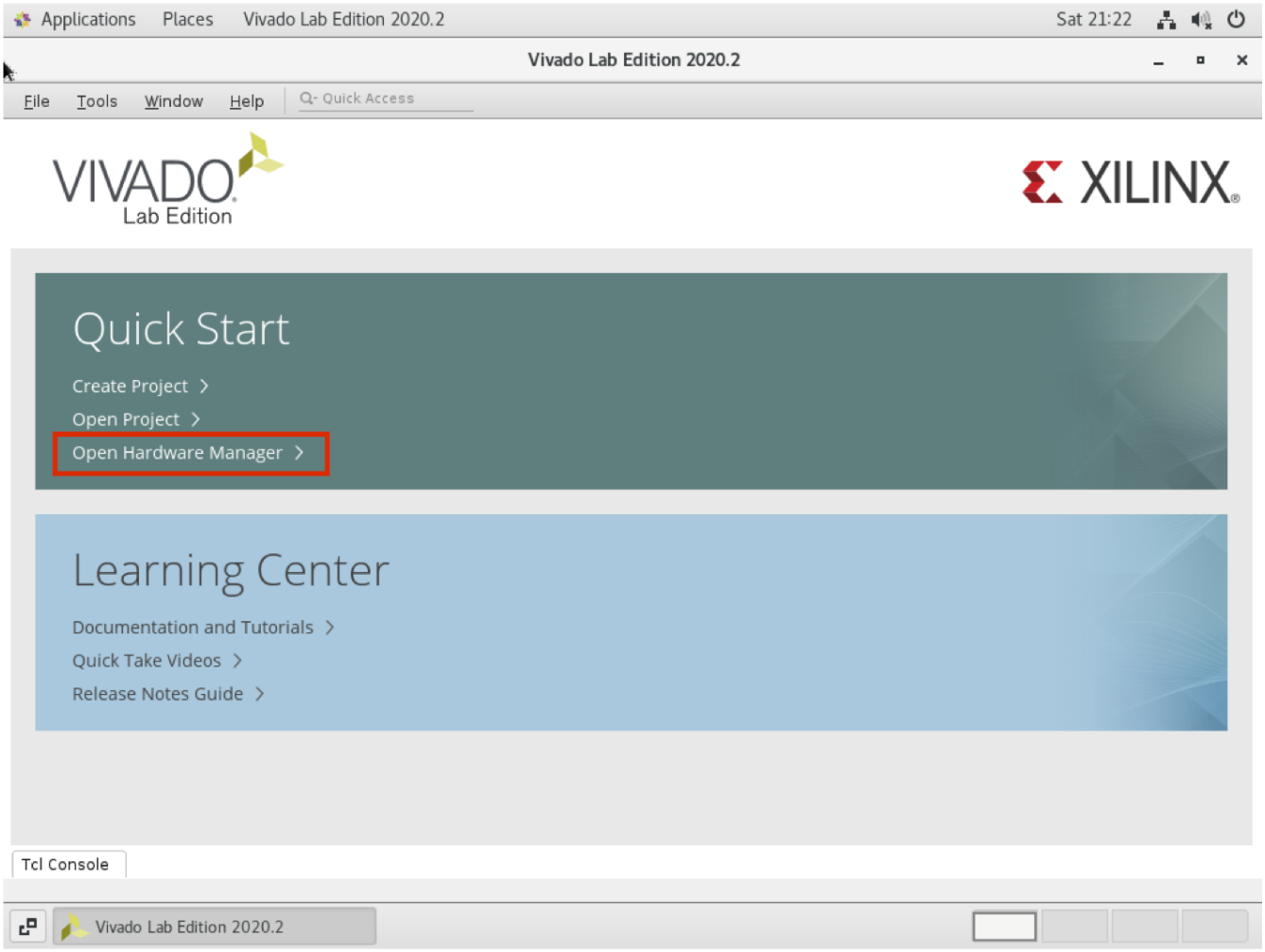

Figure 5.1 Vivado™ Splash Screen.

From the hardware manager select 'Open Target' on the top left as shown in Figure 5.2 and choose 'Open New Target'.

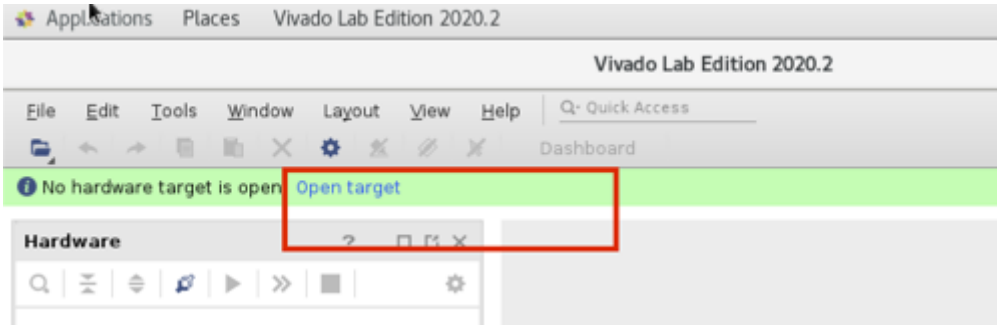

Figure 5.2 Vivado™ Hardware Manager.

From this point, select 'Next' on the following screen and 'Connect to Local Server' after that, once again press 'Next'. This should bring you to the hardware list. On this screen select the FELIX FPGA from the uppermost list (if you have only one board there should be only one entry, if not, find yours in the list by name). The screen you will see is shown in Vivado™ Target Selector. FLX-712's Kintex Ultrascale FPGA appears as *xkcu115_0*.. Once you have found your FPGA and selected it press 'Next' on the bottom right and 'Finish' on the following screen.

Vivado™ Target Selector. FLX-712's Kintex Ultrascale FPGA appears as xkcu115_0.

VC-709's Virtex7 FPGA appears as xc7vx690t_0. image::figures/vivado_selectfpga.png[caption="Figure 5.1 "]

From here, you will be taken to the main programming interface, as shown in Figure 5.3 . You are now ready to program your FPGA or FLASH.

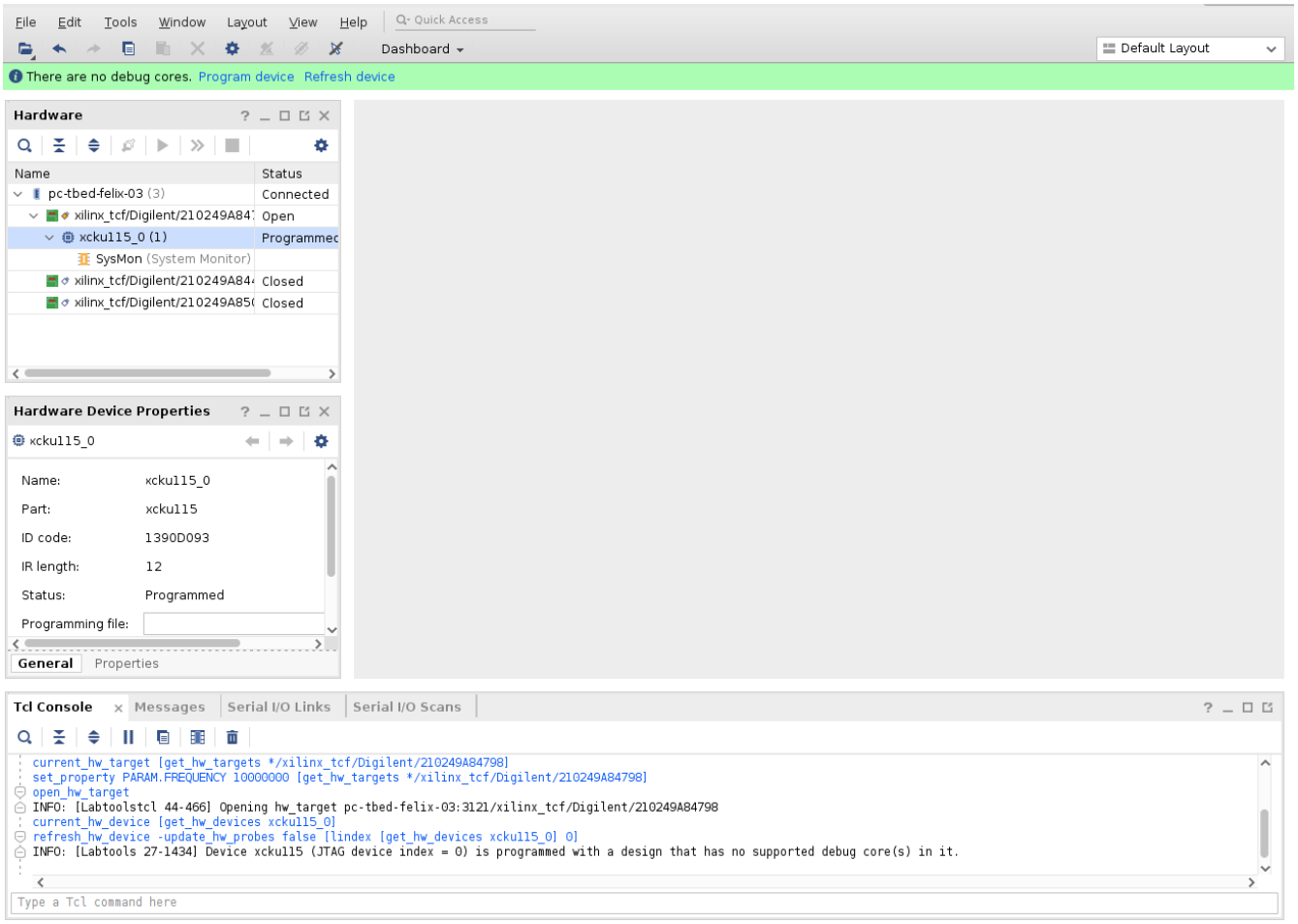

Figure 5.3 Vivado™ Programming Interface.

5.5.1 Programming the FPGA Directly

To program an FPGA directly, select it from the device list on the main programming window (as shown in Figure 5.4 , right click and select 'Program Device'.

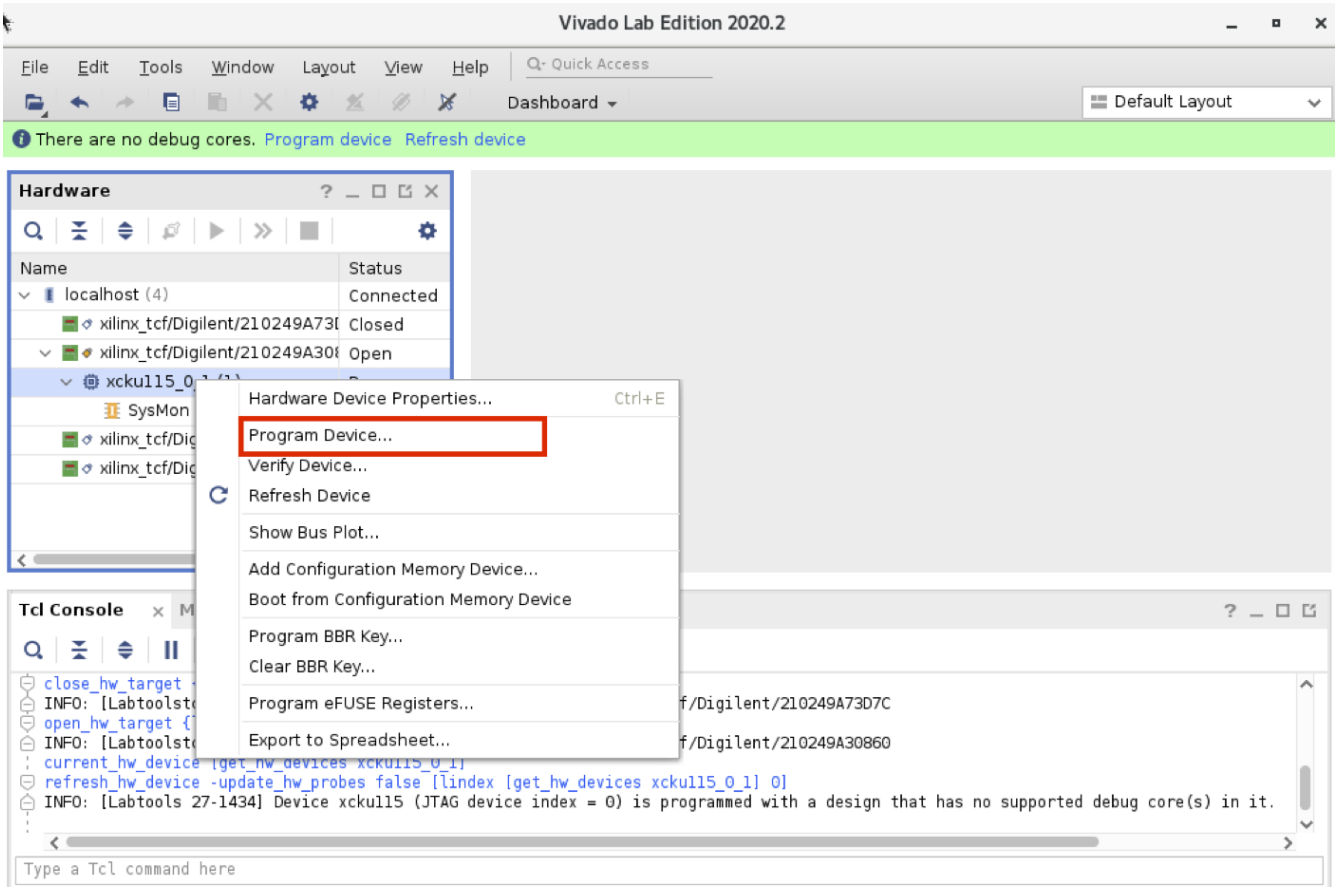

Figure 5.4 Selecting Device to Program.

You will now be asked to select a file as shown in Figure 5.5 . For FLX-712 and FLX-709 the file has extension .bit, for FLX-182 and FLX-185 it has extension .pdi. This file can be found in the firmware tarball as specified at the start of this chapter. You do not need to select a debug probes file. Once a file has been chosen, select 'Program' on the bottom right to write the file to the FPGA. Once complete your FPGA should now be fully reprogrammed.

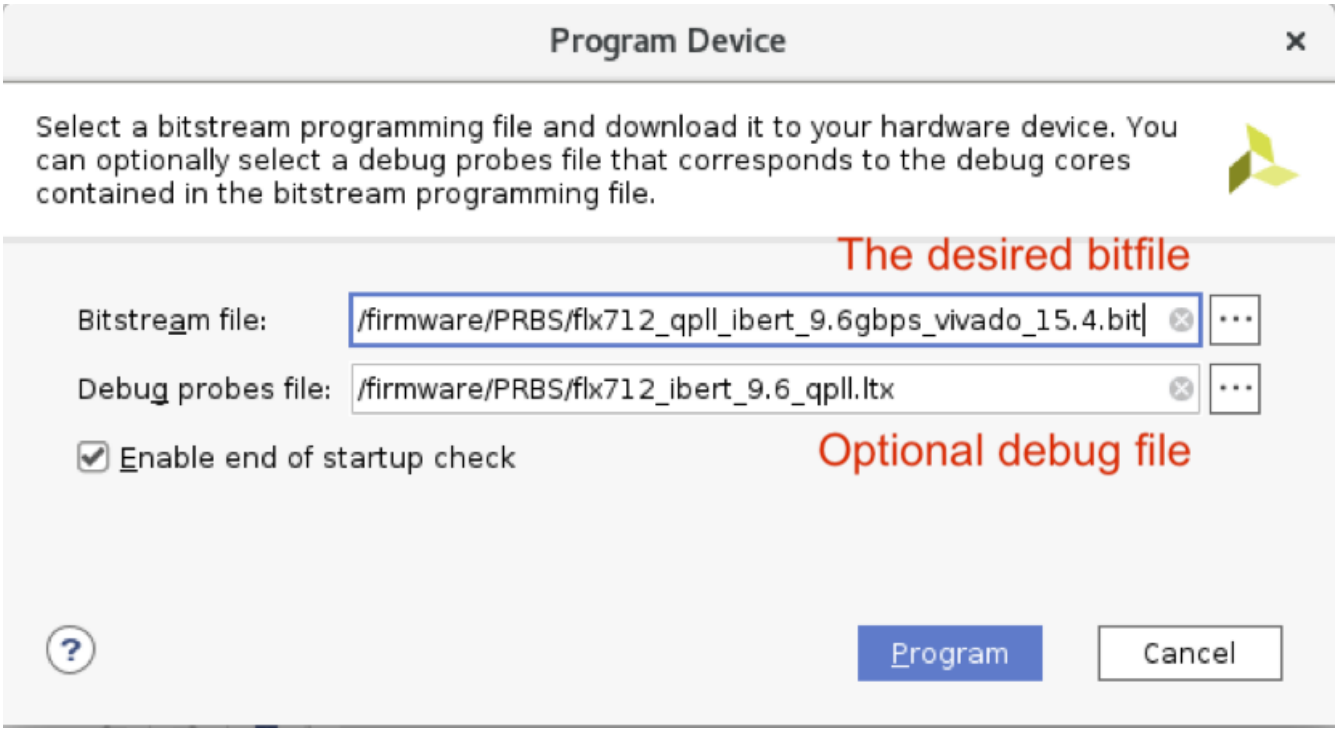

Figure 5.5 Selecting Bit file to Program.

5.5.2 Programming the FLASH ROM (FLX-709/712 only)

To program the FLASH ROM start once again from the main programming window. Find and right click on your FPGA and select 'Add Configuration Memory Device' in the list, as shown in Figure 5.6

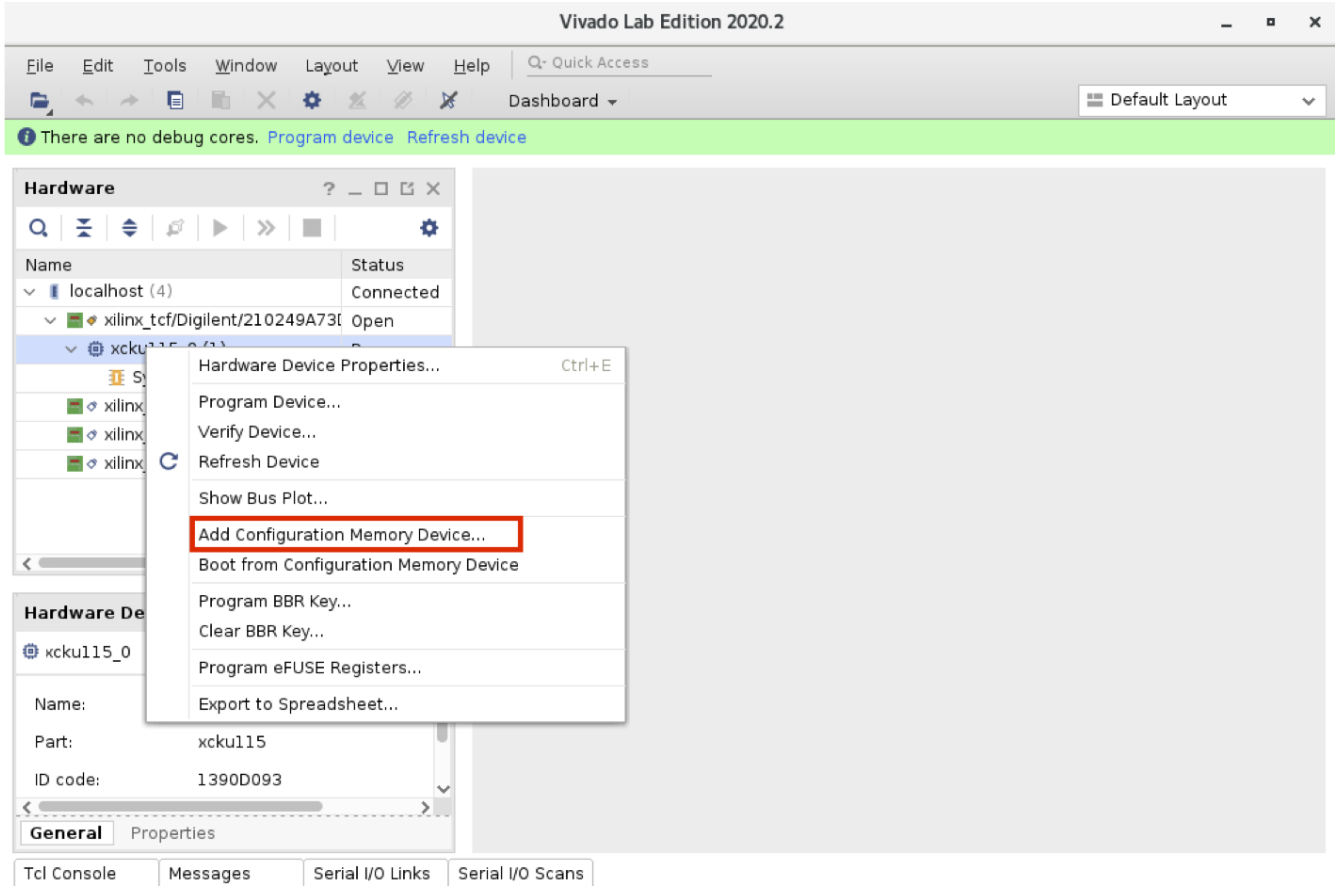

Figure 5.6 Select Vivado™ Flash Programming Dialog.

From here you will be taken to the a dialog requesting that you select the memory device you wish

to program. To find it quickly enter the criteria demonstrated in Figure 5.7 and select the device as shown. Look for the device with alias '28f00ag18f'.

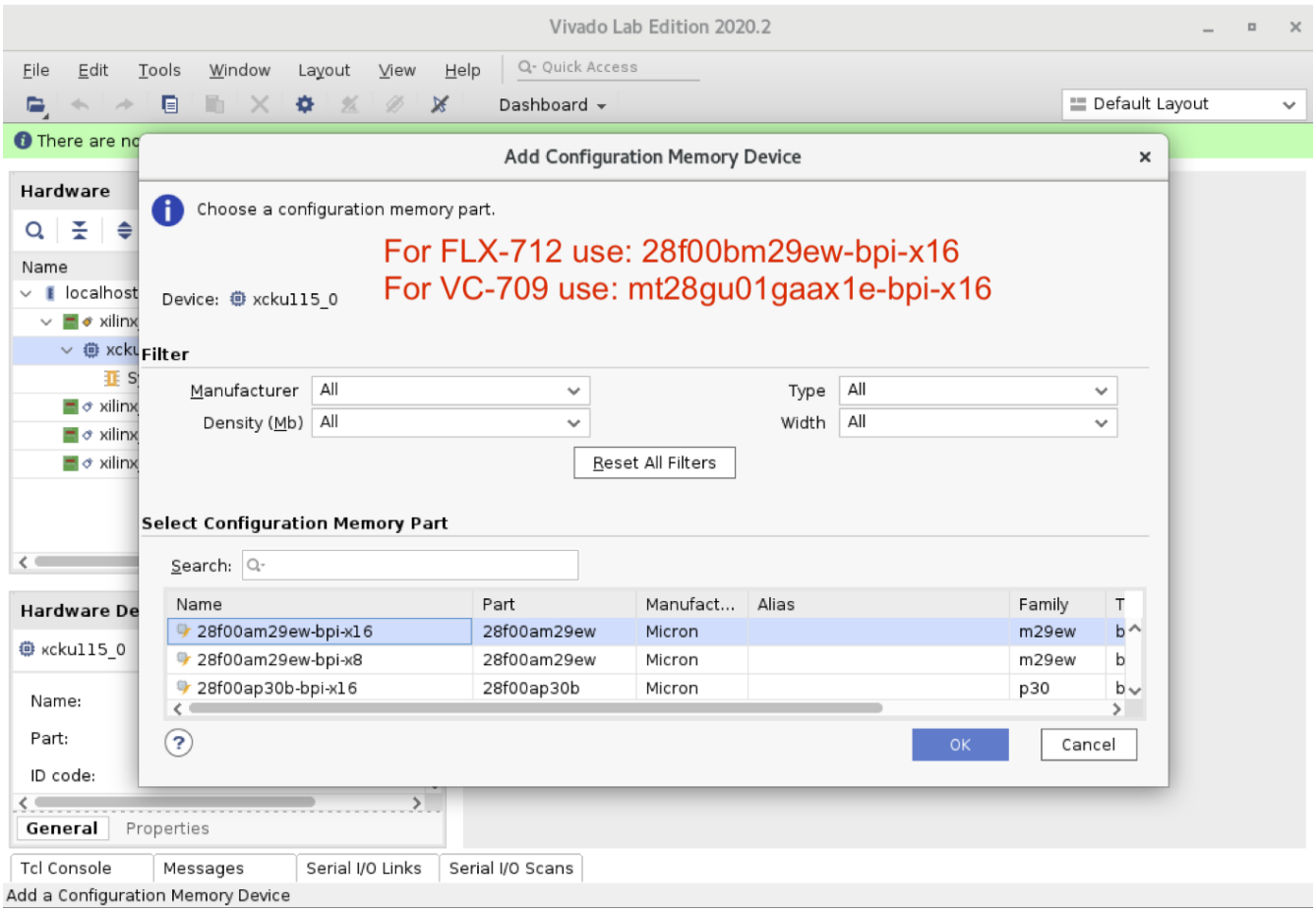

Figure 5.7 Memory Device Selection Interface.

Once selected, press 'Ok' on the bottom right and 'Ok' again on the following window asking 'Do you want to program the configuration memory device now?'. On the subsequent dialog, choose the .mcs file you wish to program (provided with your firmware release) as shown in Figure 5.8 . Select 'Ok' at the bottom to program the FLASH. Once complete your card should be programmed with a non-volatile firmware installation that will survive loss of power to the host.

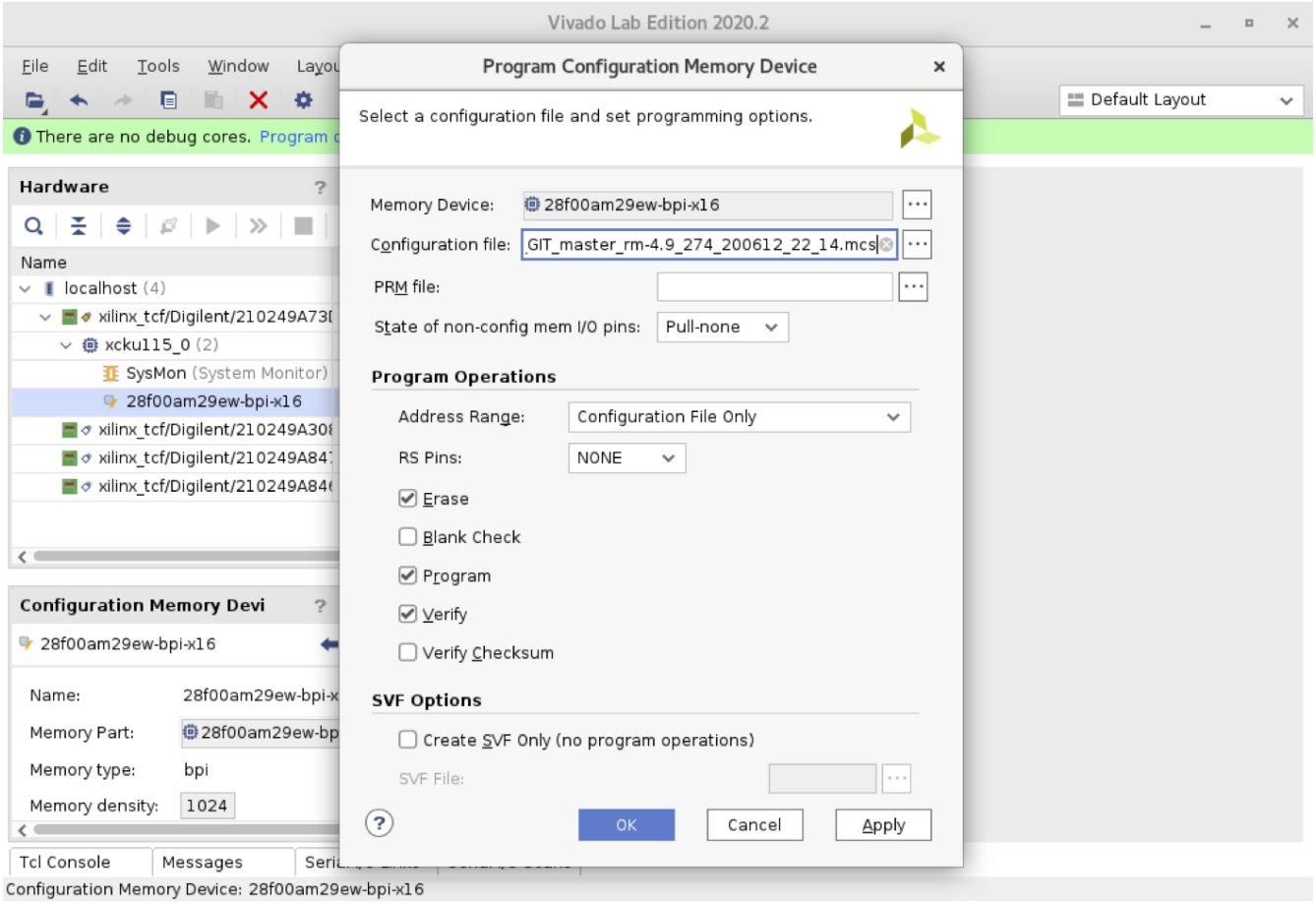

Figure 5.8 Selecting .mcs file to program.

5.6 Firmware debugging over PCIe

5.6.1 XVC (Xilinx Virtual Cable) for FLX-712/709

If a bitfile for FELIX includes an ILA (chipscope / debug probe), and XVC has been enabled at build time, the firmware can be debugged without the need of a USB / JTAG cable.

- A compatible firmware built must be requested.
- The flx driver >= version 4.10 must be installed on the server running the FLX card.
- Execute the program "xvc_pcie", this program can be found in the felix distribution.
- Run Vivado, either on the machine hosting the FELIX card, or on any machine with network access to that machine.
- In Vivado click "Open Hardware Manager", then "Open target", then "Open New Target..."
- In the wizard, click "Next", then select Connect to: Local machine.
- Select "Add Xilinx Virtual Cable (XVC)
- For host name, the host running the felix card, Port: 10200
- When highlighting "debug_bridge_0", select the debug_nets.ltx file from the archive above in the "probes file" section, as shown in the figure below.

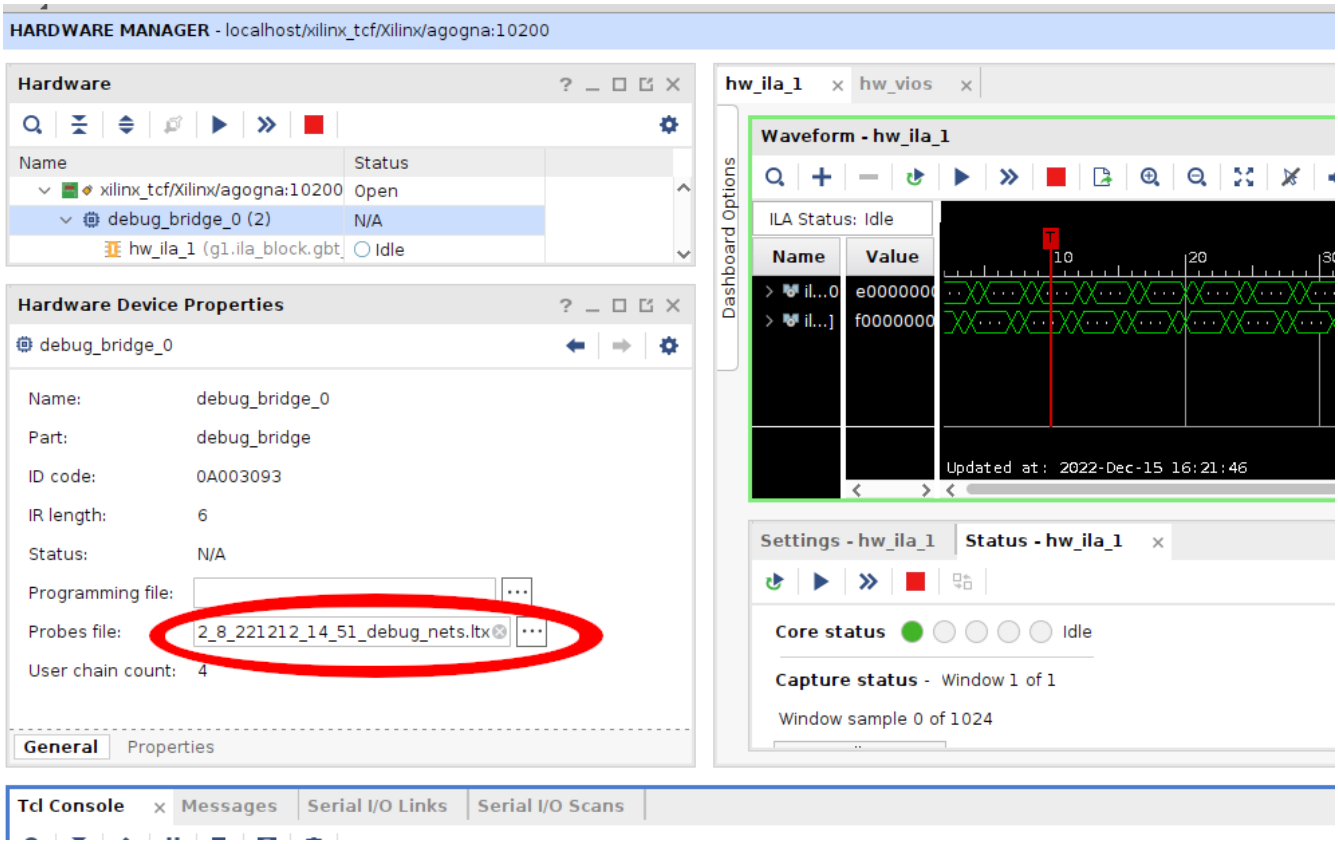

Figure 5.9 Vivado debug interface using Xilinx XVC over PCIe.

5.7 After the Reprogramming

5.7.1 Initialising the Card

This section assumes you have set up the FELIX software infrastructure as in Section 4. If you have not, then please do so before proceeding.

FELIX requires a clock source in order to synchronise propagation of signals both within the FPGA and to external peers. The FELIX firmware supports the use of both a received clock from an external TTC source as well as an internally generated clock for users who don't need or have access to a such a system.

To initiliase the card run `flx-init`. By default the internal clock is taken as source. To use the TTC clock pass the option `-T`

To view the overall clock status, one can run the FELIX info tool, or `flx-info`. This can be run with no command line parameters to dump summary information for your board as follows:

```
flx-info
```

Clock settings can then be viewed in the `Clock resources` section of the output, shown here:

```
Clock resources
---------------------------------
MAIN clock source     : LCLK fixed
Internal PLL Lock     : YES
```

```
GBT Wrapper generated : YES

TTC (ADN2814) status
------------------------
Bit D5=1 (LOS): No light. Check fiber connection to FLX card
(ADN2814 MISC register: 0x29)
```

The TTC clock is successfully configured when the 'Clock resources' section looks like this:

```
Clock resources
---------------------------------
MAIN clock source     : TTC fixed
Internal PLL Lock     : YES
GBT Wrapper generated : YES

TTC (ADN2814) status
------------------------
TTC optical connection is up and working
```

5.7.2 Connecting and Initialising Optical Links

The first step to bringing up your links is to connect your fibres to the transceiver, ensuring not to place excessive strain on them. Once the connectors are properly seated, you can check the physical status of your links.

5.7.3 Physical Link Layer Status: FLX-712

In order to check the status of your physical connections for a FLX-712 (which are MiniPOD-based) run the following:

`flx-info POD`

There will be many lines of output, but you should check the section labelled `MiniPODs` as shown below:

```
MiniPODs
========
Only the 8 active MiniPODs will be shown
NOTE: The MiniPODs of both devices will be shown

               |  1st TX |  1st RX |  2nd TX |  2nd RX |  3rd TX |  3rd RX |  4th TX |
4th RX |

|=========|=========|=========|=========|=========|=========|=========|=========|
Temperature [C]|      49 |      52 |      46 |      46 |      54 |      50 |      49 |
49 |
3.3 VCC     [V]|    3.24 |    3.29 |    3.27 |    3.28 |    3.26 |    3.30 |    3.26 |
3.31 |
```

```
2.5 VCC     [V]|    2.42 |    2.40 |    2.44 |    2.42 |    2.42 |    2.44 |    2.44 |
2.43 |

How to the read the table below:
# = FLX link endpoint OK     (no LOS)
- = FLX link endpoint not OK (LOS)
First letter:  Current channel status
Second letter: Latched channel status
Example: #(-) means channel had lost the signal in the past but the signal is present
now.

Latched / current link status of channel:
        |   0  |   1  |   2  |   3  |   4  |   5  |   6  |   7  |   8  |   9  |   10
|   11  |

|======|======|======|======|======|======|======|======|======|======|=======|=======
|
 1st TX | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) |  -(-)
|  -(-) |
 1st RX | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | #(#) | #(#) | #(#) | #(#) |  #(#)
|  #(#) |
 2nd TX | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) |  -(-)
|  -(-) |
 2nd RX | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) |  -(-)
|  -(-) |
 3rd TX | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) |  -(-)
|  -(-) |
 3rd RX | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) |  -(-)
|  -(-) |
 4th TX | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) |  -(-)
|  -(-) |
 4th RX | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) | -(-) |  -(-)
|  -(-) |
```

If your physical link is working correctly you should see loss of latch status '#' for the relevant MiniPOD RX (receive) or TX (transmit). For a physical map of MiniPOD locations please consult Appendix B.

5.7.4 Physical Link Layer Status: FLX-182

The command `flx-info ffly` replaces `flx-info pod`. However no optical power is reported as FireFly modules do not provide a reading.

5.7.5 Physical Link Layer Status: FLX-709

In order to check the status of your physical connections for a VC-709 (which are SFP-based) run the following:

```
flx-info SFP
```

Look for the line marked 'Link Status' in the output:

```
$ flx-info sfp
This is an FLX-709
               0    1    2    3
---------------------------------
Link Status  | Ok   Ok   Ok   Ok
```

5.7.6 Logical Link Layer Initialisation (All FLX cards)

After `flx-init` links should align. The alignment status can be checked with:

`flx-info link`

Re-alignemnt can be triggered calling `flx-init` again. The results should look like this (this is for a FLX-712 with 24-channel firmware):

```
$ flx-info link
Card type : FLX-712
Firmw type: GBT


Link alignment status
-------------------------

Channel |  0    1    2    3    4    5    6    7    8    9   10   11
        ------------------------------------------------------------
Aligned | YES  YES  YES  YES  YES  YES  YES  YES  YES  YES  YES  YES

Channel | 12   13   14   15   16   17   18   19   20   21   22   23
        ------------------------------------------------------------
Aligned | YES  YES  YES  YES  YES  YES  YES  YES  YES  YES  YES  YES
```

For GBT mode firmware: If this looks correct your GBT links should now be fully operational (configured, trained and locked).

Before attempting to transfer data please ensure you have followed the guide in Section 6.1 for details on how to configure your E-links.

0 :!table: 5

6. Basic Tools

The FELIX software tools suite comprises both high and low level tools. At the highest level, the **felixcore** or **felixstar** application is responsible for communication and bulk dataflow in a full slice system. At a lower level, the suite provides a number of command line based tools and a GUI based configuration tool, to facilitate system configuration and testing. This chapter describes these low level tools such that users will be able to effectively communicate with, configure and test their system.

If you are looking to set up a full system slice with data output to a network please consult Section 7 which describes readout applications. This section assumes that you have set up your FELIX software environment as described in Section 5. None of the tools in this section should require superuser privileges to run. All tools presented below work in both GBT and FULL mode, and for FLX-709 and FLX-711 or FLX-712 cards, unless otherwise stated. Where special parameters are needed to distinguish modes this will be indicated.

A quick reference for all tools to be covered in this section is presented in Table 6.2 .

the FELIX software tools suite contains a number of tools which are considered for developer use only. All tools which are rated for use by front-end users are listed in this document. Use of any other software is not recommended unless asked to do so by a FELIX developer.

Table 6.2 List of all recommended user tools. For more information on each please click the tool name to visit the dedicated section of this chapter.

Low Level Tools	
flx-info	View various FELIX hardware and firmware information and status.
flx-config	View and modify low-level firmware parameters by reading and writing FELIX firmware registers.
flx-init	Initialise FELIX, as well set as low level GBT and clock/jitter cleaning parameters.
flx-reset	Reset FELIX or a specific component.
flx-pod	Display or enable/disable individual MiniPOD channels.
fcap	View FELIX firmware E-link configuration capabilities and some other firmware properties.
felix-cmem-free	Manually deallocate memory in CMEM buffer.
flx-busy-mon	Monitor a FELIX card's BUSY signal.
flx-dma-stat	Display the status of a FELIX device's DMA controllers (expert tool).
flx-irq-counters	Inspect or reset FELIX card interrupt counters (expert tool).
Dataflow Tools	
fdaq	Receive data from a single FELIX device and save to files or perform sanity checks.

Low Level Tools	
fdaqm	Receive data from multiple FELIX devices and save to files or perform sanity checks.
fupload	Upload specific data patterns or data from file to a front-end E-link via a FELIX device.
FELIX Configuration Tools	
elinkconfig	GUI for link and data generator configuration and configuration file creation.
felink	Calculate E-link IDs given inputs with differing formats.
fereverse	Reverse the endianness of data passing through an E-link.
fgpolarity	Switch 0/1 polarity of all data coming or going through a specific GBT link.
feconf	Upload link and/or data generator configuration to FELIX from the command line.
femu	Status and control of the FELIX data generators.
ffmemu	Status and control of the data generator of the FMEMU firmware.
fttcemu	Status and control of the FELIX TTC data generator.
fttcbusy	Status and configuration of FELIX E-link TTC-BUSY settings, as well as other BUSY-related settings.
fexoff	Enable/disable the XOFF feature of FULL mode links.
fexofftx	Generate an XOFF or XON on FULL mode links.
feto	Status and configuration of FELIX timeouts (global, TTC and link data, a.k.a *instant timeout*).
febrc	Configure E-links FromHost broadcast settings.
fflash	Load a firmware image from a FLX-712 card's flash memory into the card's FPGA.
fflashprog	Program or verify firmware images in FLX-712 onboard flash memory.
General Debugging Tools	
fcheck	Run configurable sanity checks on data from a file from **fdaq** or dump selected data chunks or blocks to screen.
fedump	Dump data blocks directly from a FELIX device (or selected E-link) to screen.
GBTX and lpGBT Configuration Tools	
fice	Read or write GBTX or lpGBT chip registers via the (lp)GBT-link IC channel.
flpgbtconf	Read or write lpGBT chip registers or register bit fields by name or address via the IC channel.
fgbtxconf	Read or write GBTX chip registers or register bit fields by name or address via the IC channel.
fscai2cgbtx	Read or write GBTX registers via a GBT-SCA I2C channel, a-la **fice**.
GBT-SCA Tools	

Low Level Tools	
fec	Demo control and communication with a GBT-SCA chip (GPIO, ADC, DAC and/or I2C).
fscaid	Read and display a GBT-SCA chip ID.
fscaio	Read and write GBT-SCA GPIO lines.
fscaadc	Read out GBT-SCA ADC input channels.
fscadac	Read and write GBT-SCA DAC output channels.
fscai2c	Read and write to I2C devices connected to GBT-SCA I2C channels.
fscads24	Read out a 1-Wire 64-bit ID chip connected to a GBT-SCA GPIO pin (demo).
fscajtag	Program a bit-file into a Xilinx FPGA connected to a GBT-SCA JTAG port.
fxvcserver	Interface Vivado to a GBT-SCA chip's JTAG port and connected Xilinx FPGA(s).
fscareply	Parse and display a GBT-SCA reply given as a sequence of raw bytes.
Tools for lpGBT Control and Monitoring Channels	
flpgbtio	Read and write lpGBT GPIO lines via the lpGBT IC channel.
flpgbti2c	Execute I2C operations on an lpGBT I2C Master via the lpGBT IC channel.
flpgbtds24	Read out a 1-Wire 64-bit ID chip connected to an lpGBT GPIO pin (demo).

6.1 FELIX E-link Configuration with elinkconfig

Before FELIX can be used to transfer data its input and output links must be configured. The link configuration for a given FELIX card can be accessed and modified using the *E-link configurator* application called **elinkconfig**. This is a GUI based tool to compile and save an E-link configuration or to inspect and/or edit the E-link configuration read from a given device, and to optionally write the configuration and/or corresponding emulator data contents to a selected device (a FLX-712 card consists of 2 devices). The tool supports GBT, lpGBT and FULL mode types of links.

the link configuration when read from a device must be manually refreshed every time a FELIX FPGA is reprogrammed, including power-cycling of a host, or after having executed flx-reset to reset the registers to their defaults.

To run the **elinkconfig** application issue the following command:

```
elinkconfig&
```

The main configuration panel similar to the one shown in Figure 6.1 will appear (here the panel reflects the configuration from a configuration file that was read in).

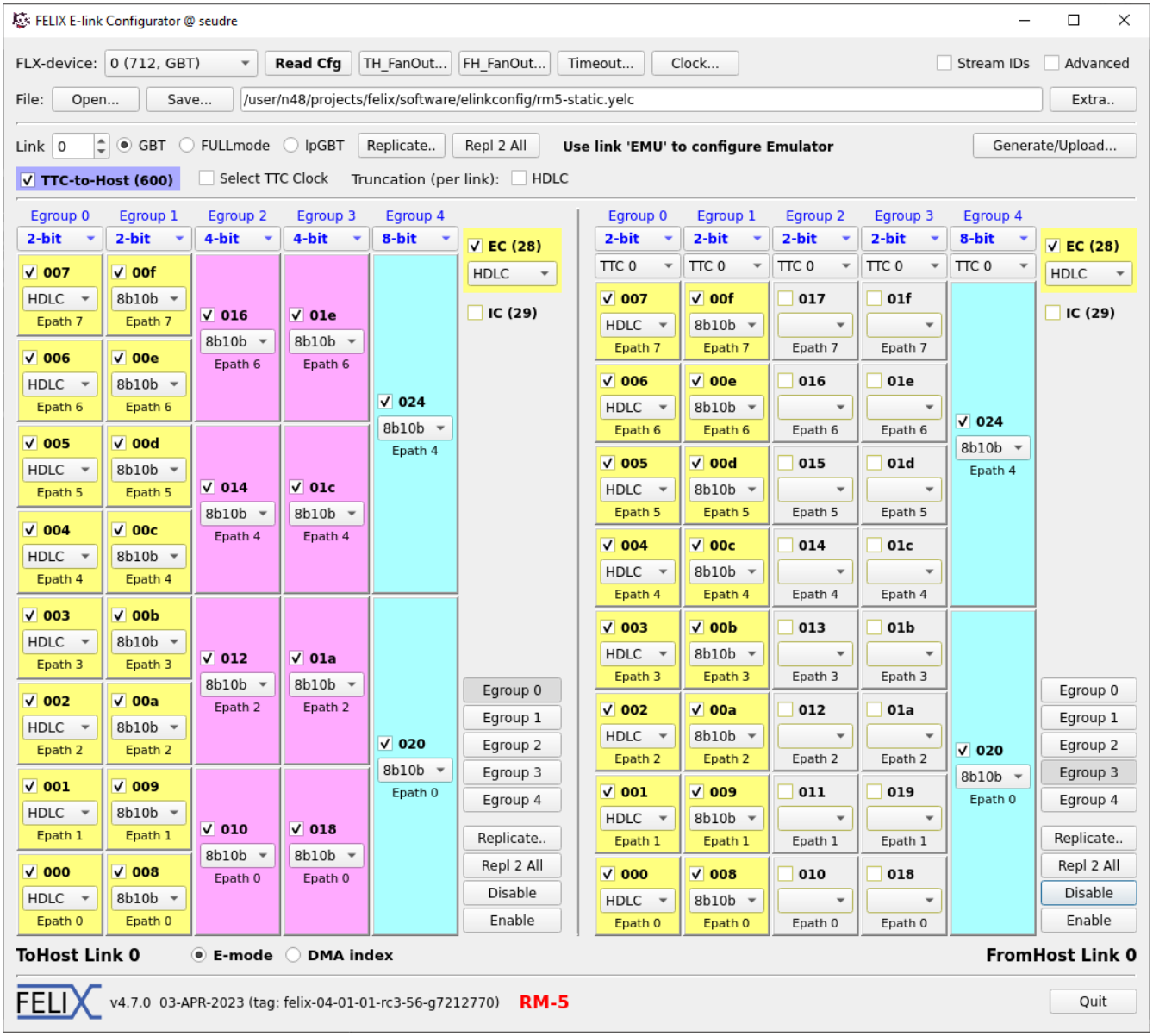

Figure 6.1 Main panel - elinkconfig

The **elinkconfig** interface is split into three main areas. At the top there are two control bars to set FELIX card parameters, open/save configuration files as well as link selectors. The left main panel displays the from front-end to FELIX/host configuration for the selected link.

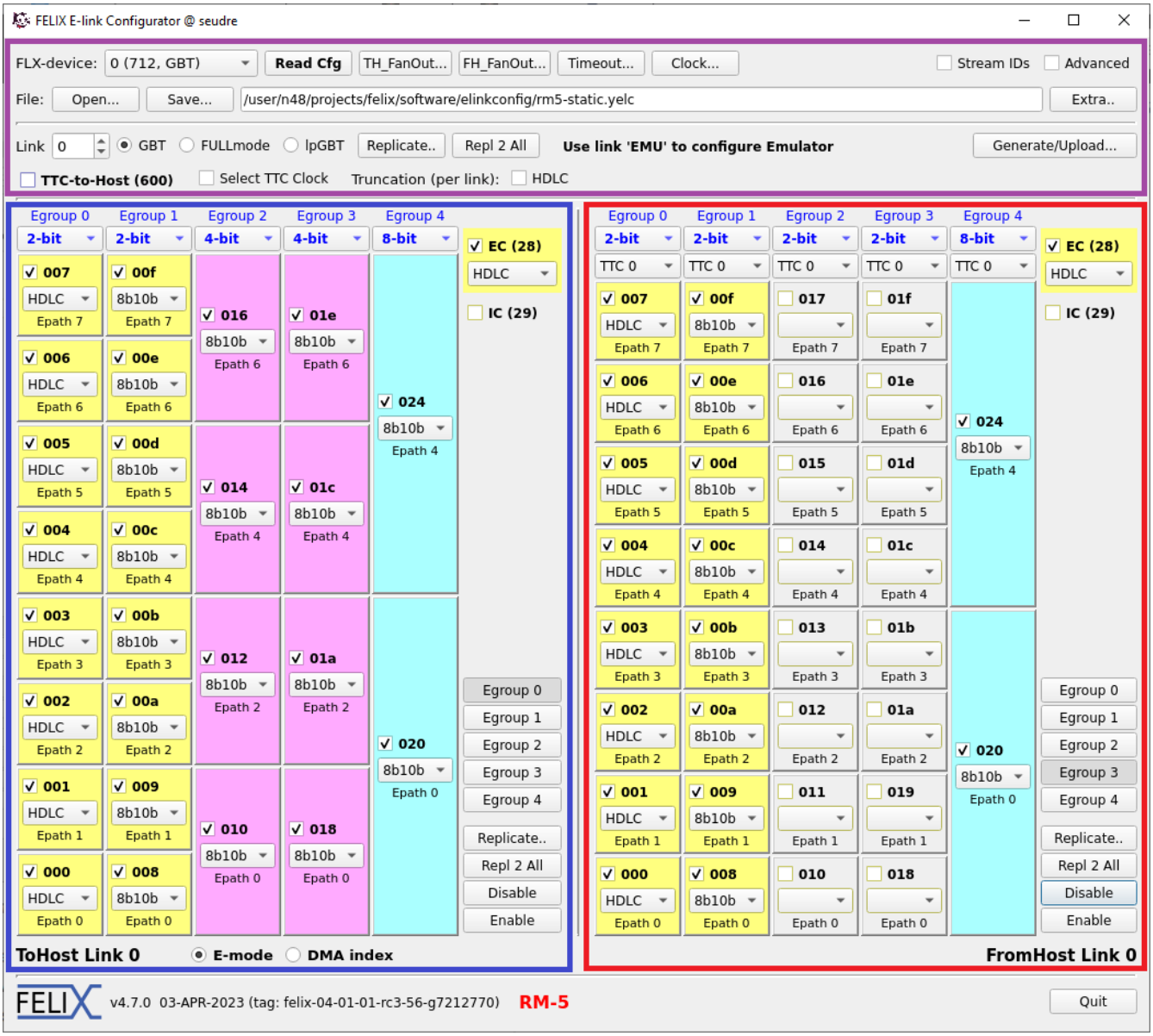

Figure 6.2 elinkconfig panel split. The uppermost panel (purple box) controls global settings and GBT selection. The left main panel contains the E-link configuration for the from front-end to host direction, the left main panel the from host to front-end direction

6.1.1 Global Panel

The **elinkconfig** 'global' panel, shown with explanations in Figure 6.3 provides the top level interface for the tool. Here you select from which FELIX device within your system you wish to read out its (link) configuration, or read and set a number of global settings (using the top row of buttons; note that these settings are then immediately made in the selected device). Note that a FLX-712/FLX-182 card consists of 2 separate FELIX devices (as shown in the FELIX device selection drop-down menu).

Starting from **elinkconfig** version 4.5.0, *reading* a FELIX device's configuration includes its 'firmware configuration' registers, which define, among other things, which E-link widths and modes are supported by the firmware for different E-groups; this is then reflected in the **elinkconfig** user interface in the options available to the user: certain width and mode options for certain E-groups may be disabled in the drop-down menus shown in the sections below. As soon as you select a different FELIX device or link mode all available options become re-enabled, and *Read*

Cfg pushbutton text turns to bold face again to indicate the configuration shown in the panel does not necessarily match the FELIX device's configuration.

In the global panel you select a link number to display and/or to configure in the ToHost and FromHost panels (presented below), as well as an associated link mode (note that the link mode is in fact a *global* card parameter; different links can not have different modes). It is possible to open previously saved configuration files and save new ones. Also there is a button to open the dialog to write the selected or manually configured link configuration to any of the FELIX devices in your system. There is a checkbox to enable *truncation* of data chunks on HDLC E-links, so that chunks with a size larger than can be expected from a GBT-SCA device (which is 12 bytes maximum) from this type of links are suppressed (unconnected links may produce random data).

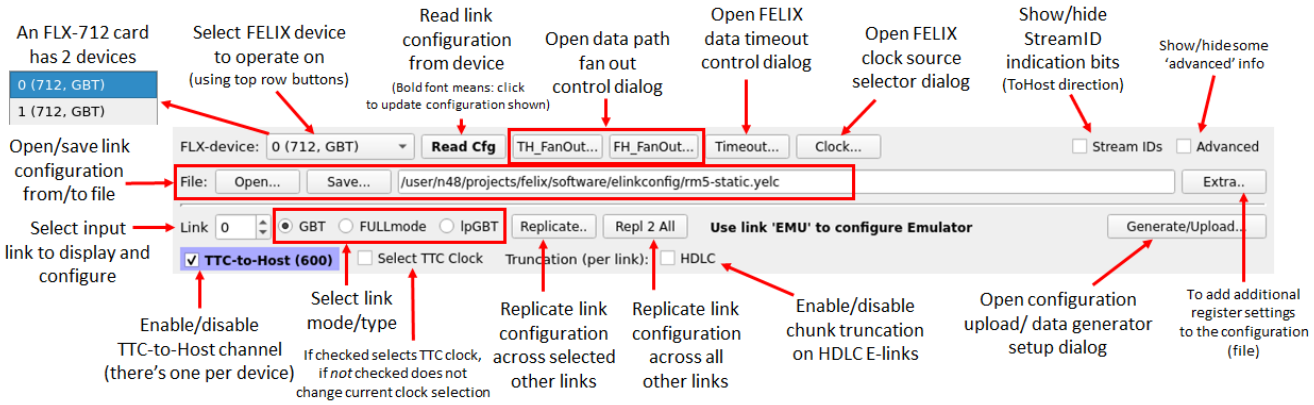

Figure 6.3 **elinkconfig** *global panel.*

From the global panel it is possible to access a number of sub-panels using the buttons at the top of the panel, as indicated in Figure 6.3 . These give access to more advanced global configuration options, details of which are presented below.

The global panel also contains a tickbox to enable the socalled TTC-to-Host channel, a virtual E-link carrying for each TTC Level-1 Accept a packet containing the corresponding TTC information.

6.1.1.1 Data Path Fan Out Selectors: TH_FanOut and FH_FanOut

FELIX operates two separate data generators within its firmware, one attached directly to the data path going to the host, and one attached to the path going towards the front-end. While the generators are attached, they have mutually exclusive access to the data path with regular non-emulated data in both directions. To avoid the two data types colliding only one type may access the path at a time. The fan out selectors control this access by ensuring that only internally emulated data or external data can be configured to pass at any one time. The FELIX applications and tools configure these selectors automatically, but for the purposes of user testing it may be necessary to set these values manually. The selectors are accessed via the TH_FanOut (to host) and FH_FanOut (from host) buttons in the global panel. The resulting dialogs are presented in Figure 6.4 .

Figure 6.4 Fan out control for to-host (top) and from-host (bottom) directions. The setting for each link is displayed separately (in this case for a 12-link FLX-712 device, i.e. for a 24-link FLX-712 card). It is also possible to (soft)lock the settings using the dedicated checkbox, causing a number of tools to not touch these settings when the lock is set.

In order to switch the selector value open the required dialog and click on the link number you wish to toggle. A link displayed with its number alone is set to external data, if a link is displayed with its number plus 'E' it is in emulator mode. You can set/unset all values at once using the *All* and *None* buttons provided.

Changes made here are immediately propagated to the selected FELIX device once you select *OK*.

In some cases a user may wish to prevent other applications from automatically changing these settings. For example, if a specific link is nominated for TTC information transfer it may be convenient to fix this to external data for the duration of a test. In this case it is possible to lock the values by selecting the *locked* check box. Applications will then refrain from changing these settings until the card is reconfigured from this interface or the FPGA is reprogrammed. More information on configuring TTC transfer to the front-end are available in Section 6.1.3 below.

6.1.1.2 Data Timeout Control Dialog

FELIX offers the facility to time out pending incoming data after a configurable window from receipt of the first related packets. This is applicable for both regular and TTC data (in the to-host direction). Should data time out then all available blocks are transferred to the host. The timeout feature is enabled by default, but can me modified or disabled/re-enabled via the control dialog accessible by selecting the *Timeout* button in the global panel. This will open the dialog shown in Figure 6.5 . From here it is possible to disable/enable both regular data and timeouts on the TTC-to-Host channel using the check boxes, as well as modify the timeout window sizes. This should typically only be done under the guidance of a FELIX developer for debugging purposes.

Changes made here are immediately propagated to the selected FELIX device once you select *OK*.

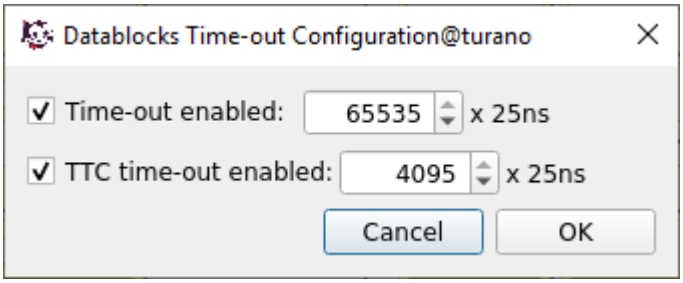

Figure 6.5 elinkconfig data timeout control dialog.

EC and TTC-to-Host data are always subjected to an immediate time-out, independent of the global time-out. In addition it is possible to enable a time-out per E-link independent of the global time-out setting, which may be important for E-links carrying irregular and small data fragments such as those connected to GBT-SCA devices (for this see tool feto).

6.1.1.3 Clock Source Selection Dialog

As mentioned in Section 3.6.1, FELIX supports two different firmware clock sources. It is possible to switch between these sources from **elinkconfig** from the clock source selection dialog, accessible by clicking the *clock* button in the global panel. The selection dialog is shown in Figure 6.6 , and is a simple two button toggle between TTC and local clock. The current status of the PLL lock is indicated as well.

Changes made here are immediately propagated to the selected FELIX device once you select *OK*.

Please also consult Section 3.6.3 before making any clock changes, to ensure you correctly configure your FELIX card's jitter cleaner post-clock change to ensure continued stable operation.

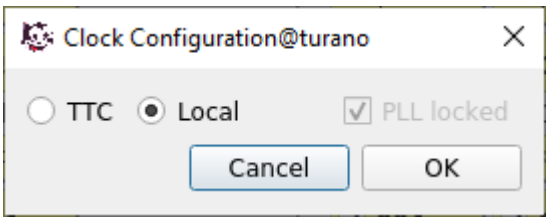

Figure 6.6 **elinkconfig** *clock source selection dialog.*

6.1.1.4 Register Settings Dialog

The *.jelc* configuration file format has an entry to contain extra FELIX device register settings in addition to the settings associated with the link and e-link configuration.

Click the 'Extra' button in the global panel to open the settings dialog, shown in Figure 6.7 . The dialog allows to interactively add settings by entering a register or bitfield name and a value. The name should be one of the defined names as can be seen in the list produced by command **flx-config list** (see flx-config).

If a name entered is not unique, potential name options to choose from will be shown. A value entered is checked to fall within the allowed range of the named item. After entering multiple settings their order can be changed at will (Note: the settings will be applied in the order shown,

after the link configuration has been done), settings can be removed or new settings added, using the buttons in the dialog. Changes are then confirmed by clicking the 'OK' button. The settings are saved in the configuration file.

Note that **elinkconfig** as well as feconf have an option to *exclude* applying these extra settings, when present, when configuring a FELIX device.

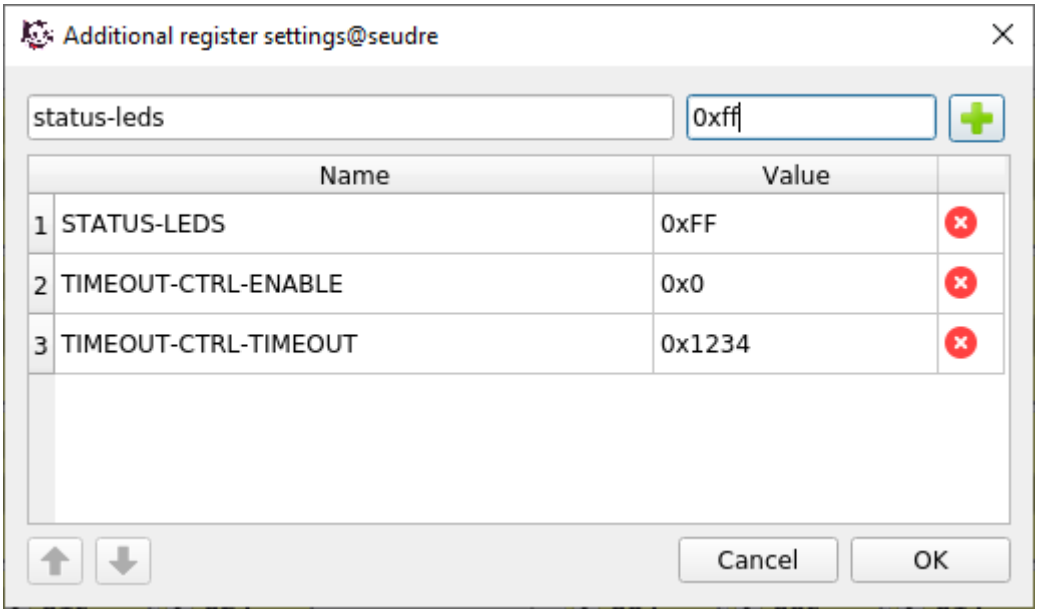

Figure 6.7 **elinkconfig** *register settings dialog.*

6.1.2 ToHost Panel

The *to-host* panel allows for configuration of the currently selected link (GBT or FULL mode) in the *to-host* direction. The type of panel to show can be selected in the global panel as described in Figure 6.3 . In the GBT case it is possible to configure the complete set of E-links associated with this link, split up by E-group, as well as the EC (External Control) and IC (Internal Control) channels. For each E-link two additional parameters can be configured, selected by means of the two radiobuttons at the bottom of this panel: the type of E-link encoding to be used (button 'E-mode'), and the assignment of a DMA (data) stream index to the E-link determining the data stream the E-link's data will be part of (button 'DMA index'); by default the index is DMA #0 and usually the FELIX firmware provides 4 different streams (data endpoints) E-links can be individually assigned to. For example, it could make sense to assign all E-links carrying monitoring, control and configuration traffic only to a separate stream, in order to separate this type of information from 'normal' data as early as possible.

If the *Stream IDs* tickbox in the global panel is ticked the panel shows Stream ID indication tickboxes. It is a bit per E-link on the device that can be set to indicate e.g. to data-acquisition software that the protocol on the corresponding E-link features a *Stream ID*, which can then be taken into account by the software. The setting itself does not have any effect on the workings of the FELIX device. An annotated screenshot of an example of this panel is presented in Figure 6.8 .

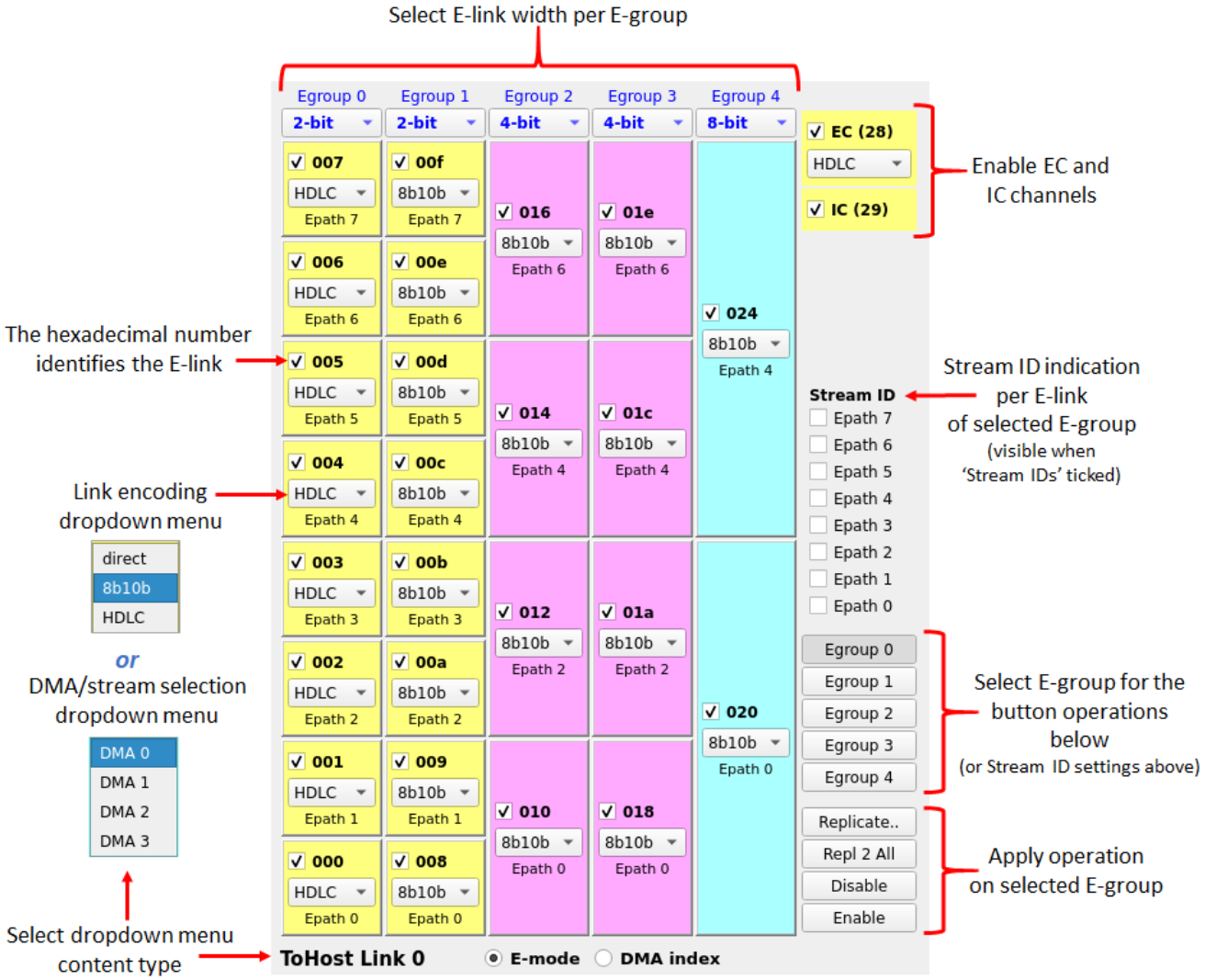

Figure 6.8 **elinkconfig** *configuration panel for to-host direction (GBT mode). Various configuration options and tools are indicated as they appear in the panel.*

In FULL mode this panel provides only a few options, as this link mode does not contain logical E-link subdivisions. This version of the panel is presented in Figure 6.9 . When the 'DMA index' configuration mode of the link is selected a dropdown menu button appears in the FULL link allowing assigment of the link data to one of the data streams of the FELIX device.

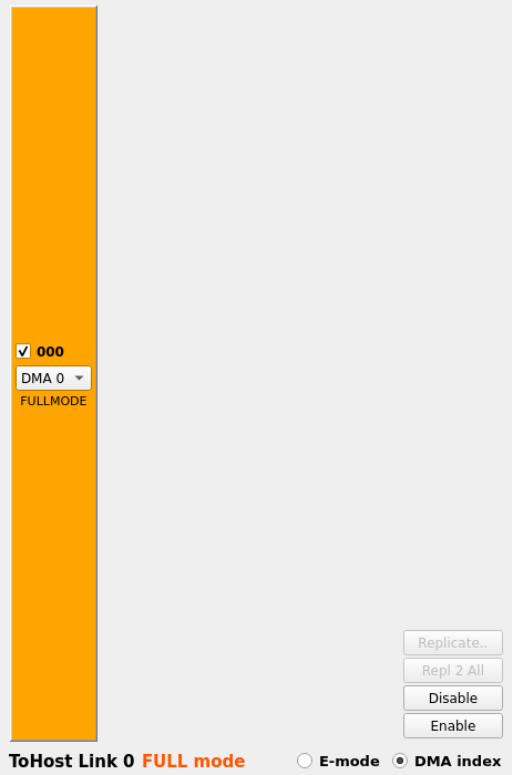

Figure 6.9 **elinkconfig** *to-host panel (FULL mode).*

6.1.3 FromHost Panel

The *from-host* panel allows for configuration of the (GBT) links transporting data from FELIX towards connected front-end electronics. This panel only exists in 'GBT mode' form as FULL mode is only a *to-host* protocol, and any FULL mode firmware will implement *from-host* links as GBT-type links.

An annotated screenshot of an example of this panel is presented in Figure 6.10 . A key difference between this panel and the to-host panel is that the link encoding options include a *TTC* mode; an E-link in this mode will send TTC data to the front-end electronics, with the TTC data format defined by the *TTC option* selected for the E-link's E-group (in the drop-down menu directly underneath the E-group E-link width selection drop-down menu). There are 16 different TTC options to choose from, although not all are currently defined. For the meaning of the various TTC options, i.e. the format of the TTC data transmitted, see elsewhere.

Note that for lpGBT links the from-host E-link mode drop-down menu includes additional modes *Strips*, *Pixel* and *E'vour* (the latter stands for: *Endeavour*), which only apply to matching FELIX card firmware flavours.

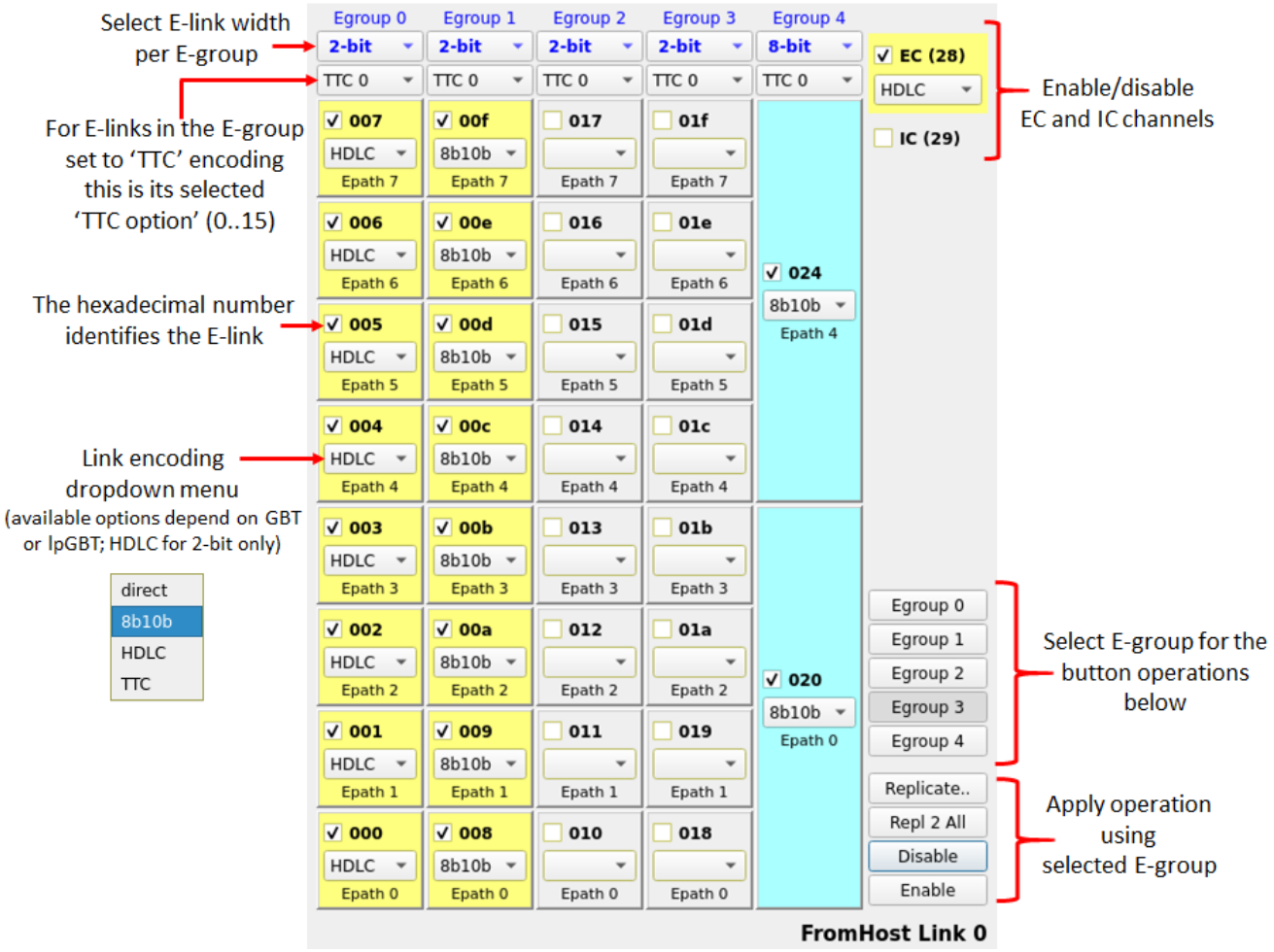

Figure 6.10 **elinkconfig** *configuration panel for from-host direction (GBT mode). Various configuration options and tools are indicated as they appear in the panel.*

6.1.4 Link and Data Generator Configuration Upload Dialog

The to-and-from host panels allow you to put together a complete configuration set for all links handled by a given FELIX card. Once you have prepared your desired configuration, you can upload it to a FELIX device in the current host machine by selecting the *Generate/Upload* button in the upper panel on the right. This will open the upload dialog, as shown in Figure 6.11 . The GBT version is shown, but the FULL mode variant is essentially identical, beyond some disabled developer features. The tickbox concerning register settings is only visible if additional register settings are present in the configuration.

The E-link mapping for the FELIX data generators can be configured by selecting the *EMU* link in **elinkconfig**. If the option to save to a file is used the emulator link configuration is saved separately to the rest of the links.

Once the panel is prepared, select *Upload* from the middle box labeled *E-link Configuration* to write your configuration to the device (Note: you have to upload to both devices of a FLX-712 card if you want to configure the full card). If you also wish to configure the FELIX on-board data generators for tests in emulation mode select the *Upload* button in the lower *Emulator Data* box.

If you are running in emulation mode and wish to change your E-link configuration you must remember to upload to the emulator too, every time you

upload a change.

In FULL mode and lpGBT mode the data generators only produce FULL mode and lpGBT mode data resp. in the to-host direction. In GBT mode GBT data can be produced in both directions.

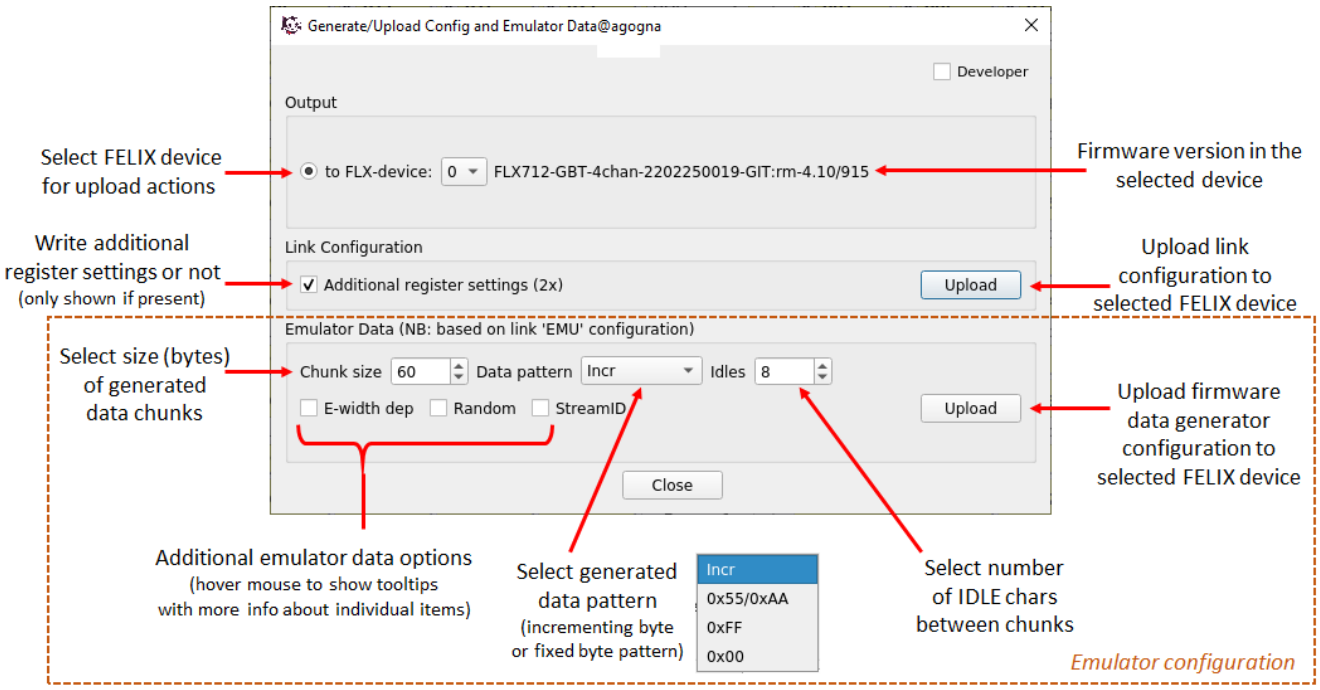

Figure 6.11 **elinkconfig** *upload panel. Note that the specified emulator chunk size does not include an 8 byte header which is automatically added.*

Once your configuration is uploaded you can then proceed to use the FELIX system as normal, the new settings will take effect immediately. To avoid unexpected behaviour please avoid reconfiguring the links while the FELIX device in question is in active use in your system.

6.1.5 Guide to Valid E-link Configurations

The E-link configuration uploaded to a FELIX card is actually a set of instructions to configure a component known as the *Central Router*. It is responsible for sending incoming data (in either direction) to the correct remote end point, as defined by E-link number. For FULL mode there is no such thing as an E-link, and so the Central Router merely propagates a wide stream of bits across the link. In the GBT case, E-links are defined as separate logical links within a given physical GBT link. E-links can have (in the current implementation) three different bit widths, which given the link clock defines the maximum bandwidth they can sustain. The widths are 2, 4 and 8 bits, running at 40 MHz. There is currently no support for 16-bit GBT E-links. A GBT link can therefore be considered as a logical aggregation of low bandwidth links into one high bandwidth transfer. For full details please consult the official documentation [3].

In *Normal* mode, a GBT link is 80 bits wide, and this puts an upper limit on the number of E-links. It is therefore possible to have few wide 8 bit links, a larger number of narrower 2 or 4 bit links, or a mixture of the two (selected on a per-E-group basis). Should a GBT be operated in *Wide* mode (not currently supported) then a further 32 bits are available within the GBT link (i.e. 112 in total), allowing for more E-links. The structure of a normal mode GBT frame is shown in Figure 6.12 . It is

up to the user to decide how much of the GBT width to utilise as per their front-end needs. It is permitted to leave link bandwidth unused by not assigned E-links to that part of the GBT frame.

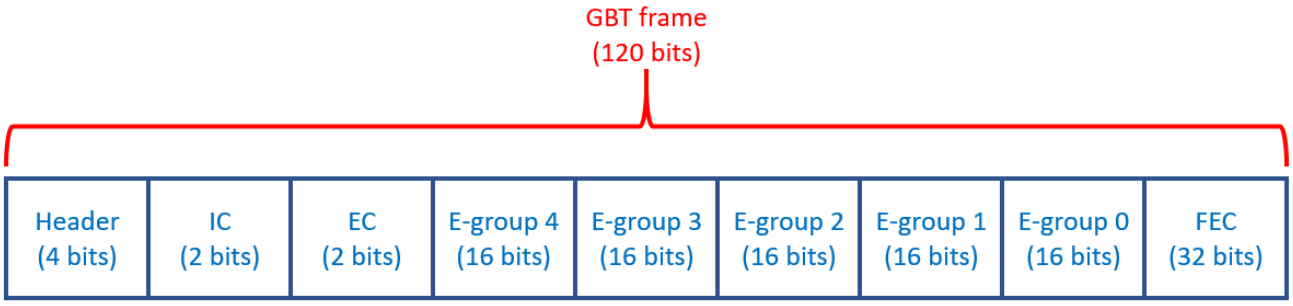

Figure 6.12 Bit structure of a GBT frame, showing E-groups, IC and EC links, as well as GBT header and Forward Error Correction (FEC) bits.

Within a given GBT link, logical links are subdivided for management purposes into 16 bit wide *E-groups*. Each E-group logically contains a combination of E-links up to an aggregate of 16 bits of width, looking at either extreme this means up to 8 of the narrowest 2 bit E-links at one end, or two of the widest 8 bit E-links at the other. The E-group is the unit of connectivity around which the **elinkconfig** interface is built, with the to- and from-host panels designed around the E-group granularity of one complete GBT link.

Note that for lpGBT links E-groups are 32 bits wide in the to-host direction and may be configured as 8-, 16- or 32-bit wide E-links (2- and 4-bit are not available). In the from-host direction the E-groups are 8 bits wide and configured as 2-, 4- or 8-bit wide E-links.

Looking within the E-group, there is one further layer of link identification to consider. Each GBT group supports up to 8 logical *E-paths* (lpGBT up to 4 E-paths). These correspond to the logical connection end-points which the Central Router supports. In the to-host panel in Figure 6.8 and the from-host panel in Figure 6.10 this is reflected in the "Epath" labels shown in each of the E-link selection boxes. Depending on the chosen E-link width in a an E-group certain E-paths are used and some not (for the GBT and lpGBT 4-, 8- and 16-bit widths and for the lpGBT 32-bit width). The *E-path* used for a particular E-link is reflected in the E-link IDs (an 11-bit significant hexadecimal number) in the E-group, so an E-link ID may not be unique for different E-link widths (for example, certain E-link IDs may refer to either a 2-bit or 4-bit wide E-link).

Within a given link map, each E-link can be configured to use different encoding formats as per front-end requirements. This area is still subject to active development, and it is strongly recommended that users work towards basing systems on 8b10b encoding. For FULL mode 8b10b is also the default.

There is a known issue with GBTX chips whereby links disconnected from any front-end source may generate spurious data at random intervals. If using FELIX with a GBTX it is strongly recommended that any links which are disconnected from the front-end be deactivated in the FELIX device using **elinkconfig**. This will prevent spurious data causing confusion in front-end testing.

6.1.5.1 Semi-static Firmware GBT E-link Configuration

In the 24-channel GBT mode build, the FELIX firmware does not support fully configurable E-links due to the need to conserve FPGA resources. The set of configurable E-links in this case is described below.

	To Host	From Host
EC link	2 bit HDLC	2 bit HDLC
E-Group 0	2 bit HDLC, 8 bit 8b/10b	2 bit HDLC
E-Group 1	2 bit HDLC, 2 bit 8b10b, 8 bit 8b/10b	2 bit HDLC, 2 bit 8b10b
E-Group 2	4 bit 8b/10b, 8 bit 8b/10b	2 bit 8b/10b
E-Group 3	4 bit 8b/10b, 8 bit 8b/10b	-
E-Group 4	8 bit 8b/10b	8 bit 8b/10b

6.1.6 Guide to common configuration tasks

6.1.6.1 Working with E-link configurations stored in files

elinkconfig reads and stores configuration sets in YAML format .yelc files. In order to load a previously existing configuration set into the tool, select *Open* from the global panel and choose the file to be loaded. The GUI will be automatically updated to reflect the new configuration. From here you can modify the configuration (if needed) by e.g. using the to and from-host panels to enable/disable E-links. Once your changes are complete you can upload the new configuration to the FELIX card of your choice using the *Generate/Upload* button in the global panel. Make sure to upload both the link and data generator configurations if you wish to use the latter. Finally, you can save your modified configuration to a file by selecting *Save* from the global panel.

6.1.6.2 Modifying the existing E-link configuration on a FELIX card without a file

If you are working without .yelc files and wish you modify the existing configuration on a FELIX device you must first load it into the tool by selecting the device in question via the global panel and then pressing the *Read Cfg* button. This will populate the GUI with the configuration currently active on the device. From here you can modify the configuration as required and upload a new version to the device (or devices) as advised above. You can also save your (modified) configuration to a file for later reuse.

Note that after starting up the tool or after modifying the selected device number the *Read Cfg* button displays the button text in bold face, to indicate the configuration has not been read from the device yet and turns to non-bold as soon as it is read. As mentioned earlier, reading the configuration from a device may alter the E-link width and mode options available in the menus, depending on the capabilities of the firmware of the device.

6.1.6.3 Configure the to-host Level-1 Accept info E-link (*TTC-to-Host* E-link)

The FELIX firmware implements a dedicated 'virtual' E-link (*virtual*, because its source is not an E-link in the electronic definition), called *TTC-to-Host*, for the purpose of forwarding TTC Level-1

Accept information to the host system for transfer to subscribers on the network. Each FELIX endpoint (or device) provides one such 'E-link', which is activated by ticking the `TTC-to-Host` checkbox in the global (top) panel in **elinkconfig**, as shown in Figure 6.3 . The E-link ID (number) on which the data will be transferred is shown there as well and has a fixed value (0x600).

A *TTC-to-Host* E-link produces a 26-byte or 27-byte *L1AInfo* data packet containing information about each Level-1 Accept. The contents of the packet are presented in Table 1 of appendix. More details on this and other FELIX data structures can be found in the appendix.

6.1.6.4 Configure the from-host TTC E-links

this section refers to the Phase-I TTC system.

Users may configure any number of from-host E-links for the purpose of transferring TTC information to their electronics. The TTC data arriving at the FELIX card will be automatically decoded, and subsets made available to users for relay to front-ends in a configurable manner. The subsets which can be sent depend on the width of the E-link chosen for the transfer and the specific 'TTC option' selected for the E-link's E-group.

The currently available options are presented in Table 6.3 , although this can evolve based on user requirements. For example, 2-bit E-links can be configured as TTC option '0' or '6', For TTC option 0 only the L1A and the full, non-decoded B-channel data stream can be sent. Alternatively, 4-bit E-links can be configured to send L1Accepts, Bunch Counter and Event Counter Resets, and a choice of either the non-decoded B-channel data stream or a user defined broadcast bit. Detector groups should communicate to the FELIX group which bits in which locations they need. If their needs are not met by an existing option, an option can be added.

Table 6.3 Currently defined TTC options. Brcst[7:2] are the TTC user defined broadcast command bits. Brcst[1] is ECR, Brcst[0] is BCR. Bit 0 is the first bit transmitted out.

E-group TTC option	**bit 7**	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
0: 2 bits							B-chan	L1A
1: 4 bits					B-chan	ECR	BCR	L1A
2: 8 bits	B-chan	Brcst[5]	Brcst[4]	Brcst[3]	Brcst[2]	ECR	BCR	L1A
3: 8 bits	L1A	Brcst[3]	Brcst[2] *2	ECR	OCR *1	L0A*	Brcst[5]	Brcst[4]
4: 4 bits					BCR	BCR	BCR	BCR
5: 2 bits							BCR	BCRd*
6: 8 bits	L1A	Brcst[3] *3	Brcst[2] *2	ECR	OCR *1	L0A*	Brcst[5]	Brcst[4]
7: 4 bits					ECR	BCR	Xoff	L1A
8: 8 bits	HGTD Fast Command, 6b/8b encoded							

TTC option 3 is as requested by the New Small Wheel and FELIX performs custom functions for

several of the input TTC bits, such as stretching to multiple BC's and copying input bits to more than one output bit. (Needed for compatibility with a two-level trigger)
Notes for option 7:
*1 TTC Broadcast bit 7 is used to request that FELIX send two consecutive BCR's, which are used as OCR (Orbit Count Reset Request).
*2 Brcst[2], used for TestPulse, is held active by FELIX for 16 BC's.
*3 Brcst[3], used for SoftReset, is chosen from the six "toggle" versions of the Brcst[7..2] in TTC option 7. Feature was added in firmware version 4.10
*BCRd is a 1-clock delayed BCR pulse.
*L0A is a copy of L1A when FELIX is driven with the legacy TTC system.

The broadcast bits[7..2] do not behave as they do for the legacy TTCrx or TTCrq ASICs. For FELIX, these bits persist only as long as their transmission from the TTC system is repeated. Whereas for the legacy ASICs, they persist until they are transmitted again, whereupon they are inverted. From firmware version 4.10, the Brcst[3] bit in TTC option 7 has been added providing this feature. For details see FLX-644 From firmware version 4.11, TTC option 8 has been added which is able to distribute Xoff for FULL mode firmware as an alternative to 8b10b encoded XOFF transmission.

If the TTC B-Channel bits do not arrive at the expected time with respect to the L1A bit, a variable delay of 0 to 15 Bunch crossing cycles can be configured for the B-Channel. This delay can be set by writing a value between 0 and 15 in register `TTC_DEC_CTRL_B_CHAN_DELAY`. For details see FLX-1280

```
# Set a delay of 15 BC cycles to the B-Channel bits of the TTC distribution
flx-config TTC_DEC_CTRL_B_CHAN_DELAY 15
```

Configuring an E-link to transfer TTC data is accomplished by setting the E-link mode to *TTC* and then selecting the *TTC option* value for the E-link's E-group. See Figure 6.10 . The value selected must be valid and compatible with the selected E-link width in that E-group.

6.1.6.5 Configure GBT-SCA E-links to/from host

E-links connected to GBT-SCA ASIC devices are 2-bit wide HDLC-encoded links. This type of E-links are designed to carry slow-control information to and from any desired front-end location. Users may set any number of 2-bit E-links (in either direction) into this mode. Note that the "EC" (External Control) E-link is not restricted to be an HDLC E-link but can be configured as a 2-bit 8b/10b E-link as well. Communication with a GBT-SCA ASIC requires both a to-host HDLC E-link and a from-host HDLC E-link enabled and connected to the GBT-SCA ASIC. They are usually, but not required to be, in corresponding positions in the E-groups. FELIX performs the HDLC encoding and decoding.

An OPC-UA server and client for the GBT-SCA ASIC are provided to allow high level communication with the various GBT-SCA's I/O channels by user software.

6.1.6.6 IC channel

The GBT IC channel is a 2-bit wide channel in the GBT protocol dedicated to control of the GBTX chip at the remote end of the link, i.e. to read from or write to its registers. Provided the GBT link is up, this channel is used for additional configuration of GBTX registers. In a FELIX device both IC *to-host* and *from-host* 'E-links' must be enabled for a user to gain access to his front-end GBTX devices.

6.2 Low Level Tools

The following section covers some general tools which allow you to monitor the state of your FELIX system, as well as make configuration changes and reset the system as necessary. All tools provide more detailed descriptions of their functionality through their help output, accessible by running the tool with option -h.

6.2.1 flx-info

The **flx-info** application is a command line tool which can print to the screen a range of monitoring and configuration information for your FELIX card(s). By default you will be presented with some system version and health status, as well as a basic link description. To access this default printout run the following command:

```
flx-info
```

This will give output similar to that what is shown below, including general information such as the firmware revision *GIT tag* and *firmware mode*, e.g. 'GBT' or 'FULL'. To produce more verbose output pass option -v to the tool.

```
> flx-info
General information
------------------------
Card type             : FLX-712
Firmware type         : GBT
Reg Map version       : 4.10
FW version date       : 22/02/24 15:43
GIT tag               : rm-4.10
GIT commit number     : 915
GIT hash              : 0x00000000e4c0ab59
F/W partition jumpers : 3
Number of interrupts  : 8
Number of descriptors : 2
Number of channels    : 12
Number of endpoints   : 2
GBT Wrapper generated : YES
MAIN clock source     : TTC fixed
Internal PLL Lock     : YES

Output of lspci:
------------------------
05:00.0 Communication controller: CERN/ECP/EDU Device 0428
06:00.0 Communication controller: CERN/ECP/EDU Device 0427
```

```
TTC (ADN2814) status
------------------------
TTC optical connection is up and working
```

Apart from this output of general information about the FELIX card and its on-board firmware, **flx-info** has several 'command' options to obtain status information about various parts of the hardware and firmware on the board, for example the MiniPOD™ devices status, various board voltages and temperatures (from onboard LTC2991 devices [15]), the status (aligned or not) of links and E-links and status information of various other devices (note that some information is specific to a certain type of FELIX card; the tool will indicate this).

For a complete list of available options run **flx-info** with option -h. This will produce the output shown below:

Help text of **flx-info***:*

```
Usage: flx-info [OPTIONS] [COMMAND]
Displays information about an FLX card.

Options:
  -c <number>         Use card indicated by <number> (default 0).
  -s <secs>           If necessary wait up to <secs> seconds for free (I2C) lock,
                      accessing the card for actions involving I2C ops (default: 5
secs).
  -h                  Display help.
  -v                  Verbose mode.
  -V                  Display the version number.
  -z                  Clear the TTC/LTI counters (commands TTC or LTI),
                      or clear latched signals and error counters
                      (commands LINK, ELINK or LINKRESET).
  -F                  Show FEC error counter increments during one second,
                      in addition, if any errors (with command LINK).
Commands: (case-insensitive)
  COUNT               Display the number of FELIX devices installed in the host.
  CARDS               Display the number of FELIX cards installed in the host.
  FPGA                Display the status of the FPGA.
  POWER               Display the status of the power monitoring devices
                      (LTC2991 or IN226/TMP435).
  POWERSEQ            As 'POWER', and in addition some FLX-182/155 LTM4700 and ADM1x66
info.
  POD                 Display the overall status of the MiniPODs.
  PODTEMP or PODVCC   Display temperature and voltage (VCC) readings of the MiniPODs.
  PODPOWER            Display optical power readings of the MiniPODs.
  PODPOWERRX          Display optical power readings of the RX MiniPODs only.
  TTC or LTI          Display info from TTC or LTI related registers.
  FREQ                Display the RXUSRCLK frequency.
  LINK (or GBT)       Display the channel alignment status
                      (plus PLL LOL and FEC error counters; see option -F).
  ELINK               Display the E-link alignment status
```

```
                     (plus 'PATH' error counters).
  LINKRESET          Display the channel auto-reset counters.
  ADN2814            Display the ADN2814 register 0x4.
  CXP                Display the temperature and voltage from CXP1 and CXP2.
  SFP                Display info from the Small Form Factor Pluggable transceivers.
  DDR3               Display the values from DDR3 RAM memory.
  SI5324             Display the SI5324 status (on FLX-709 only).
  SI5345             Display the SI5345 status.
  LMK03200           Display the LMK03200 status.
  ICS8N4Q            Display the ICS8N4Q status (on FLX-710 only).
  FFLY or FIREFLY    Display status of the FireFly modules (FLX-182/155 only).
  EGROUP [<ch>] [RAW] Display the values from EGROUP registers:
                     if no channel number <ch> is specified, display all available,
                     using hexadecimal notation if RAW is specified.
  LSPCI              Output of 'lspci | grep -e Xil -e CERN'.
  ALL                Display all.
```

6.2.2 fcap

The **fcap** tool provides some information additional to what can be obtained with flx-config, in particular the E-link configuration capabilities of FELIX GBT firmware. The firmware has a per-Egroup and per-direction setting for E-link configuration. Here's an example of the output of **fcap**:

```
> fcap
Firmware  : FLX712-GBT-12chan-2006041630-GIT:rm-4.9/244
Trailer   : 16-bit
Blocksize : 1024
(FromHost:YES DirectMode:NO Xoff:NO TTCemu:YES)
E-link configurability:
Egroup |    ToHost    |  FromHost
-----------------------------------
   0       8,HDLC        HDLC
   1       2,8,HDLC      2,HDLC
   2       4,8           2
   3       4,8           ---
   4       8             8
```

Per *E-group* is shown what width and type of E-links can be configured, in both the *to-host* and *from-host* directions. For example '8' here means only 8-bit 8b10b-encoded E-links are possible for that group, '2,HDLC' means both 2-bit 8b10b-encoded or HDLC-encoded E-links can be configured for the group, 'HDLC' means the E-group consists of 2-bit HDLC E-links only. Individual E-links always can be disabled or enabled.

6.2.3 flx-config

The flx-config tool allows users to read and modify FELIX control and configuration registers from the command line. This should normally only be done on advice from a member of the FELIX development team. Other features are also available, but these should be considered for experts

only unless advised otherwise by the development team. The two primary features users will use will be the 'list' and 'set' and 'get' features. List mode will dump the values of all known FELIX register items to the screen, including the address of the register containing the item (a register may contain multiple individual items), whether the item is readable and/or writable, the bits in the register the item occupies, its name, its *current* value and a short description of the item. This will be a large amount of output, but can be searched e.g by piping it through *more* or through *grep* with a keyword to get the desired information. To run list mode execute the following command:

```
flx-config list
```

To change a given register bitfield value (or item):

```
flx-config ITEMNAME <val>
```

In this case ITEMNAME corresponds to the register bitfield to be changed and <*val*> to the new value be stored. Once set you can use list mode to confirm the change, or read the item explicitly, as follows:

```
flx-config ITEMNAME
```

If the given ITEMNAME does not correspond exactly to a defined item, a list of items to choose from will be displayed containing the given name as a substring, if any. Item names are case-insentive and '_' characters may be replaced by '-'.

6.2.4 flx-init

The **flx-init** tool resets the GBT wrapper and transceiver, initializes the onboard clock generator device and resets the onboard optical modules, and should be performed every time the FPGA is reprogrammed (including in case of loss of power). The tool should also be run if the GBT fibres are disconnected at any point before attempting to transfer data once again.

To run the basic initialisation issue the following command:

```
flx-init
```

If you wish to use more features (if instructed by a member of the development team), consult the help dialog shown below.

Help text of **flx-init***:*

```
Usage: flx-init [OPTIONS] [<yelc-file1>] [<yelc-file2>]
Initializes an FLX card.
Options:
  -c <number>         Use card indicated by <number> (default: 0).
  -h                  Display help.
  -V                  Display the version number.
  -v                  Display verbose output.
  -E                  Execute the command even if resources are locked.
  -L                  Set clock to 'Local' (default: leave-as-is).
  -T                  Set clock to 'TTC' (default: leave-as-is).
  -s <secs>           If necessary wait up to <secs> seconds for locks to be freed
                      when attempting to open the card (default: 5 secs).
```

```
GBT calibration options:
  -a ONE|CONTINUOUS   Select alignment type (default: ONE).
  -t FEC|WideBus      Select transmission mode (default: FEC).

Clock configuration options:
  -I <input_sel>      The value given is written to register HK_CTRL_FMC_SI5345_INSEL
                      before Si5345 initialisation;
                      valid <input_sel> values are (default: 0):
                        0: FPGA (LA01), 1: FMC OSC, 2: FPGA (LA18)
  -G                  Read and display Si53xx chip registers (does not init anything),
                      i.e. Si5324 (for a 709) and/or Si5345 (2x for a 182 or 155)

E-link configuration with .yelc files <yelc-fileX>:
  Provide a filename per FLX device to configure.
  Example: to initialize and then configure both devices of the *second* FLX-712 card
  in a system (first file for 1st device of the FLX-712, second file for the 2nd
device):
       flx-init -c1 lpGBT-8bit.yelc lpGBT-16bit.yelc
  To configure only the first device, provide a single file name.
```

6.2.5 flx-reset

The **flx-reset** tool makes it possible to selectively reset components of the FELIX firmware, or the complete board, as needed given the situation. This should only be done if advised by a FELIX development team member. To see the list of available parameters please consult the help output shown below:

Help text of **flx-reset***:*

```
Usage: flx-reset COMMAND [OPTIONS]
Tool to reset various resources on the card.

Commands:
  DMA             Resets the DMA part of the Wupper core.
  REGISTERS       Resets the registers to default values.
  SOFT            Global application soft reset.
  GTH             Reset links RX (for FULL mode F/W only).
                  operates on register GBT_SOFT_RX_RESET_RESET_ALL).
    -q <n>        The individual quad (0..11) to reset (default: all).
  LINK            Reset a link or links (RX and/or TX; default all RX;
                  operates on registers GBT_[RX|TX]_RESET).
    -r <n>        Individual RX link (0..47) to reset.
    -t <n>        Individual TX link (0..47) to reset.
  ADN2814         Reset the ADN2814 (not on FLX-182/155 or FLX-128).
  ALL             Do everything. (Note: use -c, not -d:
                  resets the card and the resources of all devices of that card).
Options:
  -d <number>     Use device indicated by <number>
                  (applies to commands DMA/SOFT/REGISTERS; default: 0).
  -c <number>     Use card indicated by <number> (default: 0).
```

```
  -E            Execute the command even if resources are locked
  -s <secs>     If necessary wait up to <secs> seconds for locks to be freed
                when attempting to open the card (default: 5 secs)
  -h            Display help.
  -V            Display the version number.

Note:
Use -c <number> with ADN2814, GTH, LINK and ALL.
Use -d <number> with commands DMA, SOFT and REGISTERS.
If neither -c nor -d are given, card or device number 0 is used as appropriate.
```

To reset a given component pass the name to **flx-reset** on the command line:

```
flx-reset <component_name>
```

6.2.6 flx-pod

Help text of **flx-pod***:*

```
Usage: flx-pod [OPTIONS] [COMMAND]
Displays an FLX card's MiniPOD channel enable status, and allows
a user to enable or disable individual RX and/or TX channels.
Options:
  -c NUMBER     Use FLX card indicated by NUMBER. Default: 0.
  -h            Display help.
  -V            Display the version number.
  -i NUMBER     MiniPOD channel number: [0..11] (default: all).
  -R NUMBER     Receive (RX) MiniPOD number : [1..4] (default: all).
  -T NUMBER     Transmit (TX) MiniPOD number: [1..4] (default: all).
Commands:
 disable | disa | dis   Disable selected MiniPOD channels
 enable  | ena          Enable selected MiniPOD channels
```

6.2.7 felix-cmem-free

The **felix-cmem-free** tool makes it possible for a user to manually deallocate memory from the CMEM buffer. In order to use the tool, the 'handle' of the allocated memory must first be found. To do this issue the following command:

```
cat /proc/cmem_rcc
```

This will dump the current allocation status in a format similar to what is shown below.

```
CMEM RCC driver (FELIX release 4.5.0)

The driver was loaded with these parameters:
gfpbpa_size    = 7500
gfpbpa_quantum = 4
gfpbpa_zone    = 0
numa_zones     = 1
```

```
alloc_pages and alloc_pages_node
   PID | Handle |         Phys. address |               Size | Locked | Order | Type |
Name

GFPBPA (NUMA = 0, size = 7500 MB, base = 0x0000000295c00000)
   PID | Handle |         Phys. address |               Size | Locked | Type | Name
 27549 |      0 |    0x0000000295c00000 | 0x0000000040000000 |     no |    4 |
FlxReceiver0

The command 'echo <action> > /proc/cmem_rcc', executed as root,
allows you to interact with the driver. Possible actions are:
debug    -> enable debugging
nodebug  -> disable debugging
elog     -> Log errors to /var/log/messages
noelog   -> Do not log errors to /var/log/messages
freelock -> release all locked segments
```

The 'handle' is shown in the second column in the 'GFPBGA' table, in the row according to the process whose memory you wish to deallocate. Finally, pass the handle to **felix-cmem-free** as follows:

```
felix-cmem-free <handle>
```

Future FELIX releases (software version 4.2 onwards) should incorporate more advanced automatic deallocation in the case of abnormal program termination, but in the short term **felix-cmem-free** should provide a manual workaround.

6.2.8 flx-busy-mon

The **flx-busy-mon** tool enables a FELIX card's BUSY interrupt (triggered by a BUSY signal change), then starts monitoring the BUSY signal controlled by the BUSY interrupt, displaying a status line every second, indicating the number of triggers seen and the BUSY percentage in the past second, calculated by the software from the time stamps between BUSY active and inactive events.

Here is an example, where the FELIX card's #0 master BUSY gets activated and then deactivated a couple of seconds later:

```
> flx-busy-mon
Firmware: FLX712-GBT-2x2CH-230309-1652-GIT:rm-5.0/3091
Monitoring BUSY interrupt...
 T  #Interrupts   BUSY%/sec     Total BUSY time
--- ------------ ------------ ----------------
  1: cnt=   0     BUSY   0.000% (Total 0.0s 0ns)
  2: cnt=   0     BUSY   0.000% (Total 0.0s 0ns)
  3: cnt=   1     BUSY  32.665% (Total 0.326s 713340ns)
  4: cnt=   1     BUSY 100.000% (Total 1.326s 944741ns)
  5: cnt=   2     BUSY  72.904% (Total 2.56s 77188ns)
  6: cnt=   2     BUSY   0.000% (Total 2.56s 77188ns)
  7: cnt=   2     BUSY   0.000% (Total 2.56s 77188ns)
```

```
  8: cnt=   2     BUSY   0.000% (Total 2.56s 77188ns)
  9: cnt=   2     BUSY   0.000% (Total 2.56s 77188ns)
^C Exiting...
```

Help text of **flx-busy-mon***:*

```
Usage: flx-busy-mon [OPTIONS]
Displays percentage of BUSY (per second), total BUSY time,
as well as number of BUSY interrupts (i.e. changes of the BUSY signal) detected
(in brackets the number of interrupts not matching a detected BUSY change, if any)
in a single line of output, once per second

Options:
  -d NUMBER      Use device indicated by NUMBER. Default: 0.
  -h             Display help.
  -V             Display the version number.
  -i NUMBER      Interrupt number [0..7] of BUSY signal (default: 6).
  -L             Monitor and report latched BUSY sources (and clear when set).
  -B <perc>      Exit when BUSY percentage reaches or exceeds <perc> percent.
```

6.3 Dataflow Tools FELIX from/to Host PC

6.3.1 fdaq(m)

The **fdaq** tool is the primary tool for testing the FELIX data acquisition path (for a data stream from a single FELIX device; **fdaqm** is a version of the same tool supporting multiple data streams, i.e. the same could be achieved by running multiple instances of **fdaq** on different terminal windows at the same time). The tool can run in multiple modes, from waiting for input for FELIX from a front-end source to running with one of the two internal data generators on the card activated, mostly for test purposes. In both modes **fdaq** will measure and report throughput for the duration of the test. Data can be dumped to file or discarded upon receipt. If running in discard mode **fdaq** will check the integrity of the data blocks and chunks it receives (e.g. block headers and chunk sizes; and optionally it will count the chunks per (E-)link). If an error is found the test will, by default, stop and **fdaq** will report on the first detected error. However, if option -D is used the run will continue with a regular report printed on all errors received.

This section assumes that your E-links and data generators are configured properly as specified in Section 6.1. In this section we will cover various scenarios, but a list of all options can be found in the help output, shown below.

When writing data to file, depending on the data rate coming from your FELIX device it might only be possible to run **fdaq** for a couple of seconds when exceeding the maximum write speed of your disc. The host memory buffer which the data from the FELIX device is transferred to, will quickly fill up, and cause **fdaq** to stop the ongoing transfer and exit before the buffer becomes completely full, to avoid any corrupted data gets written to file.

Help text of **fdaq**:

```
fdaq version 24050300
Stream data from FLX-device to file(s). Whenever the set maximum file size
is exceeded a new file is created. Every second a status line
with data rates, data totals and memory buffer status is displayed.
(NB: if no filename is provided all data is consumed while checking the data blocks,
     i.e. blockheader and (sub)chunk trailers;
     chunk truncation and error counts are reported.)

Usage: fdaq -h|V -D -d<devnr> -b<size> -e|E -f<size> -H
            -i<dma> -I -r<runnr> -t<secs> -n -C -R -T -X -o -x<kbyte>
            [<filename-base>]
  -h         : Show this help text.
  -V         : Show version.
  -C         : Do *not* check for presence of data chunk CRC errors
               (when not writing to file).
  -D         : Debug mode on, i.e. output some additional info;
               continue when memory buffer overflows.
  -d <devnr> : FLX-device to use (default: 0).
  -b <size>  : DMA (cmem_rcc) memory buffer size to use, in MB
               (default 1024, max 32768).
  -e|E       : Enable FLX-device data generator, internal (e) or
               external (E) (default: false).
  -f <size>  : Maximum file size, in MB (default 4096, max 16384).
  -H         : Chunks have headers (default: auto-detect).
  -i <dma>   : FLX-device DMA controller to use (default: 0).
  -I         : use interrupt to receive data (default: polling)
  -n         : Display chunk count per e-link
               (when exiting, when not writing to file).
  -o         : Display status output not in columns (but line-by-line).
  -r <runnr> : Run number to use in file names (default: none).
  -R         : Reset DMA at startup (default: 'soft reset' only).
  -t <secs>  : Number of seconds to do acquisition (default: 1).
  -T         : Do NOT add datetime as part of file names.
  -X         : Stream data from individual e-links to separate files
               (default: false).
  -x <kbyte> : Set size of FLX-device unit data block, in KByte
               (forced; normally read from FLX-device itself).
 <filename-base>: Name to be combined with datetime+runnumber+counter
                  of files created (unless option -T is given)
```

6.3.1.1 Running a DAQ Test with External Data Source

The most simple configuration for **fdaq** to run in is to listen for any data coming into FELIX over the GBT/FULL mode link and measure the bandwidth as this arrives at the host. In this mode the data is discarded. The only parameter a user must define is the time in seconds for which **fdaq** should perform the test. The default time is 1 second. The syntax is as follows:

```
fdaq -t <secs>
```

For a three second test the output will resemble this:

```
$ fdaq -t3
Consume FLX-device data while checking the data (blockheader and trailers),
counts errors including chunk truncation, halts when the memory buffer is near
overflowing.
Also counts chunk CRC errors.
Opened FLX-device 0, firmw FLX712-MROD-48chan-2004041603-GIT:RM4.10/1 (cmem
buffersize=1024MB)
**START** using DMA #0 polling
  Secs | Recvd[MB/s] | File[MB/s] | Total[(M)B] | Rec[(M)B] | Buf[%] | Wraps
-------|-------------|------------|-------------|-----------|--------|-------
     1           0.0          0.0             0           0        0       0
     2           0.0          0.0             0           0        0       0
     3           0.0          0.0             0           0        0       0
**STOP**
-> Data checked: Blocks 0, Errors: header=0 trailer=0
Exiting..
```

If you would like to dump your data to a file for analysis specify a filename after the other command line parameters:

`fdaq -t<secs> testfile`

This will run as above and produce a time-stamped .dat file in the directory you are running with a name of the format 'testfile-<timestamp>-<counter>.dat'. You can specify the maximum size for the file with option -f specifying a size in megabytes (default 1024, max 4096); a next file will have an incremented <counter> as part of its name. If you like to split the input from multiple E-links into a separate file per E-link use option -X. In that case the E-link numbers will become part of the filenames too. If you do not want or need the timestamp use option -T. Note that this will overwrite your files when you do a second run and use the same file name in the **fdaq** call.

6.3.1.2 Running a DAQ Test with Internal Data Generation

A facility to use both data generators within the FELIX card for the purposes of testing is provided by **fdaq**. The 'internal' generator is connected directly to the data output path of the card (i.e. after input side of the GBT link interface). Data from this generator therefore passes through the full FELIX firmware data path with the exception of the link layer itself. The 'external' data generator is connected to the output path before the GBT layer. This means it can be configured to send data out of a GBT link. If some loopback fibres are connected it is therefore possible to send data out of one GBT transceiver and into another on the same FELIX and therefore test more of the data path. To access these options in **fdaq** one must use option -e for internal data generation and option -E for external data generation. Note that for external loopback (-E) to work the relevant emulator control register must be set accordingly:

`flx-config GTH_LOOPBACK_CONTROL 0x2`.

The output from **fdaq** will be the same whichever source is used. Here is an example:

```
$ fdaq -t5 -e
Consume FLX-device data while checking the data (blockheader and trailers),
counts errors including chunk truncation, halts when the memory buffer is near
overflowing.
Also counts chunk CRC errors.
Opened FLX-device 0, firmw FLX712-GBT-12chan-1910221102-GIT:rm-4.8/46 (cmem
buffersize=1024MB)
**START(emulator)** using DMA #0 polling
  Secs | Recvd[MB/s] | File[MB/s] | Total[(M)B] | Rec[(M)B] | Buf[%] | Wraps
-------|-------------|------------|-------------|-----------|--------|-------
     1        1449.5          0.0        1449.5           0        2       1
     2        1451.2          0.0        2900.8           0        1       2
     3        1451.0          0.0        4351.8           0        3       4
     4        1451.5          0.0        5803.3           0        2       5
     5        1451.4          0.0        7254.7           0        1       6
**STOP**
-> Data checked: Blocks 7072497, Errors: header=0 trailer=0
Exiting..
```

6.3.2 fupload

The FELIX software tools suite makes it possible to transfer data from the FELIX host PC via the FELIX card to the front-end across any GBT E-link. This is done using the **fupload** tool. With this tool it is possible to transfer data either from a user defined file, or with predefined data chunks with a fixed pattern of a configurable size and data pattern on a specified E-link across a GBT connection. The full range of features of the tool can be seen in the help text shown below.

This tool works with FULL mode firmware versions, where the link to the front-end is a GBT link.

Help text of **fupload***:*

```
fupload version 24050600
Upload data (test data or from file) to the given FLX-device E-link.
The E-link number is provided as a (hex) number directly (-e option)
or as a set of -G/g/p options,
unless option -R is given ('raw' unformatted upload).

Checks whether the E-link is valid and configured on the selected FLX-device,
unless option -c is given.

Usage: fupload [-h|V] [-D] [-d <devnr>] [-b <size>] [-c] (-e <elink>
               | (-G <lnk> (-g <group> -p <path>) [-i <dma>] [-I]
               [-s <bytes>] [-P <patt>] [-f <speed>] [-R] [-t <secs>]
               [-u] [-x <size>] [-X] [-y <tlp>]
               [<filename>]
  -h         : Show this help text.
  -V         : Show version.
  -b <size>  : DMA (cmem_rcc) memory buffer size to use, in MB (default 128, max
```

```
4096).
  -B         : Contents of <filename> is read as binary data (default: ASCII).
  -c         : Do *not* check whether E-link is configured on FLX-device
               (Note: for raw and broadcast uploads no check is done).
  -d <devnr> : FLX-device to use (default: 0).
  -D         : Debug mode on, i.e. display blocks being uploaded.
  -f <speed> : Speed up default upload rate of about 8MB/s by factor <speed>
               (default: 1; only applies when using option -x)
  -i <dma>   : FLX-device DMA controller to use (default: auto).
  -I         : Run a trickle DMA (Trickle DMA index selected by -i).
  -P <patt>  : Test data pattern: 0=incr, 1=0x55/0xAA, 2=0xFF, 3=incr-per-chunk
(default: 0).
  -r <repeat>: Test data repeat count: upload <repeat>*<bytes> bytes of data (default:
30).
  -R         : Upload data unformatted, not as CR from-host data packets with header.
  -s <bytes> : Number of bytes per chunk to upload (default: 32).
  -t <secs>  : Number of seconds for DMA time-out or wait until DMA done when 0
(default: 0).
  -u         : Do not perform the actual upload operation.
  -x <size>  : Size of single-shot DMA transfers, in KByte (default 0: one DMA, all
data).
  -X         : Use continuous-mode DMA for upload (default: single-shot).
  -y <tlp>   : Size of TLP used in FromHost DMA transfers, in bytes (default: 32;
expert use only).

Options to define the E-link to use:
  -e <elink> : E-link number (hex) or use -G/g/p options.
  -E <elink> : an optional 2nd E-link number to upload to
               (alternating with the first given E-link number).
  -G <lnk>   : GBT-link number.
  -g <group> : Group number.
  -p <path>  : E-path number.
  <filename> : Name of file with data to upload (ASCII or binary),
               or test pattern data if no name is given.

In ASCII data files one line represents one data packet (hexadecimal byte values
separated by spaces),
while lines starting with certain characters may be used to:
  #           insert a comment line (ignored)
  + <b> <cnt> insert a packet of the given length 'cnt' containing bytes of the given
(hex) byte value 'b'
  * <n>       repeat the previous chunk 'n' times
  & <d>       insert a configurable delay 'd' in microseconds between two packets
  > <e>       change the E-link number to upload to to 'e' (hex)
```

6.4 FELIX Configuration Tools

6.4.1 felink

The **felink** tool is a link descriptor interpreter which allows you to work out the E-link ID for a given link given GBT/E-group/E-path (or vice versa). This is intended to be used in conjunction with e.g. fupload to allow users to work out which link ID they should target with their data. Some examples of possible uses will be given below, but you can find all possible options in the help text below:

Help text of **felink***:*

```
felink version 23052400
Convert a given E-link number into (lp)GBT, egroup and epath numbers, or viceversa.
The E-link number is provided as a (hex) number directly (-e option)
or as a set of -G/g/p options.
Optionally checks if this E-link is valid and configured on a given FLX-device (option
-d),
in to-host and from-host direction, taking into account GBT or lpGBT.
Use option -l or -L to display a list of valid ((lp)GBT) E-link numbers,
optionally in combination with -G or -g options to restrict the output
to a particular link and/or egroup.
(Note that E-link numbers are also shown by the elinkconfig GUI).

Usage: felink [-h|V] [-d <devnr>] (-e|E <elink>
               | (-G <lnk> (-g <group> -p <path>))
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device to use (default: 0).
  -e <elink> : GBT E-link number (hex) or use -G/g/p options.
  -E <elink> : lpGBT E-link number (hex) or use -G/g/p options.
  -G <lnk>   : (lp)GBT-link number.
  -g <group> : Group number.
  -l|L       : Show a list of valid GBT or lpGBT E-link numbers
               (use options -G/g/p to restrict the list).
  -p <path>  : E-path number.
```

A list of all valid E-links and coordinates can be seen with list mode, available with the following syntax:

```
felink -l
```

6.4.1.1 Finding E-link ID from GBT/E-group/E-path of GBT/Bit address/width

Consider the example where a user wishes to know the E-link ID for a link connected to GBT link 2, within E-group 3 and E-path 4. This can be done as follows:

```
felink -G<GBT ID> -g<egroup ID> -p<epath ID>
```

Filling these in gives results as shown below, from which we can see that the E-link ID is 0x9C. The results also show alternative coordinates for the E-link in terms of GBT bit address and width.

```
$ felink -G2 -g3 -p4
E-link 09C = GBT #2 group #3 path #4, bit#56 width=2|4
```

It is also possible to search for link ID using the GBT ID, bit address of the start of the E-link in the GBT frame and E-link width. The syntax is as follows, noting that the index must correspond to a valid E-link start point.

```
felink -G<GBT ID> -I<bit address> -w<E-link width>
```

If a user then wants to search for GBT 1, bit 4 and width 2 the results will be as shown below. This identifies the E-link in question as 0x42.

```
$ felink -G1 -I4 -w2
E-link 042 = GBT #1 group #0 path #2, bit#4 width=2|4
```

These calculations can also be done in reverse, to yield the coordinates of a given known E-link ID. For this use the following syntax:

```
felink -e<E-link ID in hex>
```

If as user then wants to know the coordinates of e.g. E-link 0x55 the tool can be used to give the results as shown below. From this it can be seen that the GBT ID is 1, the E-group ID 2 and the E-path ID 5. An estimate for the bit address and width is also displayed.

```
$ felink -e55
E-link 055 = GBT #1 group #2 path #5, bit#42 width=2 OR bit#40 width=8
```

6.4.2 fereverse

The **fereverse** tool makes it possible to swap the bit ordering of data transferred through a designated E-link (or set of links: all in one Egroup, or all in a GBT link), including for EC and IC links separately.

See help text below for the list of options.

Help text of **fereverse***:*

```
fereverse version 21011300
Enable, disable or display the bit-order reversal feature for e-links,
a setting per e-path (e-link).
Without keyword '(re)set' the current setting is displayed.

Usage: fereverse [-h|V] [-d <devnr>] [-e <elink>]
                 [-G <lnk> [-g <group>] [-p <path>]] [-E] [-I] [-f] [-t] [set|reset]
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device to use (default: 0).
  -e <elink> : E-link number (hex) or use -G/g/p options.
```

```
  -G <lnk>   : GBT-link number (default: all links).
  -g <group> : Group number (default: all groups).
  -p <path>  : E-path number (default: all paths).
  -E         : Display or enable/disable the EC channel 'bit swap'.
  -I         : Display or enable/disable the IC channel 'bit swap'.
(for options -E/-I use option -G, not -e; options -g/-p are ignored)
  -f         : Configure FromHost (FH) only.
  -t         : Configure ToHost (TH) only.
  set        : Enable e-link bit-reversal.
  reset      : Disable e-link bit-reversal.
```

In order to use the tool to toggle the bits for a given E-link use the following syntax:

```
fereverse -d <FELIX ID> -G <GBT ID> -g <E-group ID> -p 1 <set/reset>)
```

In this case the *set* option indicates the bits should be switched and *reset* indicates deactivation of the switch. It is also possible to pass the E-link ID directly using option -e. If neither set or reset are specified the tool will report back the current status. Examples of both cases are shown below.

```
$ fereverse -d0 -G1 -g1 -p1 set
GBT 1 egroup 1 epath 1 TH: ENABLED
GBT 1 egroup 1 epath 1 FH: ENABLED
```

```
$ fereverse -d0 -e49 reset
GBT 1 egroup 1 epath 1 TH: disabled
GBT 1 egroup 1 epath 1 FH: disabled
```

6.4.3 fgpolarity

The **fgpolarity** tool makes it possible for FELIX to adapt to the bit polarity of data produced by front-end systems and sent via a Versatile Link[16] transceiver. The transceiver, by design, swaps the polarity of incoming and outgoing bits (i.e. 0 becomes 1 and vice-versa). Some front-end systems may already account for the swap in their design, but in order to send and receive packets to and from those who haven't this tool configures FELIX to automatically swap the bits for any designated GBT links (and therefore all E-links within).

See help text below for the list of options.

Help text of **fgpolarity***:*

```
fgpolarity version 21121600
Configure or display the GBT transceivers RX and TX polarity.
NB: settings for the entire FLX-card through device #0 only.

Usage: fgpolarity [-h|V] [-c <cardnr>] [-G <lnk>] [set|reset]
  -h         : Show this help text.
  -V         : Show version.
  -c <cardnr>: FLX-card to use (default: 0).
```

```
  -G <lnk>   : GBT-link number (default: all).
  -r         : Configure RX only.
  -t         : Configure TX only.
  set        : Set reverse polarity for given GBT transceiver(s).
  reset      : Set default polarity for given GBT transceiver(s).
 (without keyword '(re)set' the current setting is displayed)
```

In order to use the tool to toggle the polarity of a particular GBT link use the following syntax:

`fgpolarity -d<FELIX ID> -G<GBT ID> <set/reset>)`

In this case the *set* option indicates the activation of a polarity switch and *reset* indicates deactivation of the switch. It is also possible to modify the Tx and Rx directions separately using options -t and -r accordingly. By default both will be changed. The expected output from the tool is shown below.

```
$ fgpolarity -d0 -G1 set
GBT 1 RX polarity: 1
GBT 1 TX polarity: 1

$ fgpolarity -d0 -G1 reset
GBT 1 RX polarity: 0
GBT 1 TX polarity: 0
```

6.4.4 feconf

The **feconf** tool is a command line tool providing a subset of the functionality of **elinkconfig**. With this tool it is possible to upload a pre-defined E-link configuration file (.jelc or .elc format) to a FELIX card. Alongside the basic configuration, **feconf** also makes it possible to configure the FELIX firmware data generators. For more information on the meaning of each parameter, consult Section Section 6.1 on **elinkconfig**. **feconf** sets the so-called fan-out registers for front-end input (so *not* for emulator input).

See help text below for the list of options.

Help text of **feconf***:*

```
feconf version 22052200
Upload an e-link configuration from file (as generated by elinkconfig) to the given
FLX-device,
including generation and upload of emulator data contents.
Any 'RegisterSettings' present in the file are applied as well
(after the e-link and emulator configuration).
Usage: feconf [-h|V] [-d <devnr>] [-s <chunksz>] [-w] [-R] [-S] [-I <idles>] [-n] [-N]
<filename>
  -h          : Show this help text.
  -V          : Show version.
  -d <devnr>  : FLX-device to use (default: 0).
  -F          : Do not set FLX-card fan-out registers (default: set for DAQ, not
```

```
emulator).
  -L          : 8b10b-words LSB first (GBT only; default: MSB first).
  -n          : Don't write the configuration, just read it in and display some info.
  -N          : Skip the extra register settings included in the configuration file.
  -R          : Generate emulator data chunks with pseudo-random size.
  -S          : Generate emulator data with a StreamID (first byte).
  -s <chunksz>: Emulator data chunksize to generate (default: 32).
  -w          : Generate emulator data chunksize dependent on e-link width (default:
false).
  -I <idles>  : The number of idles between generated emulator data chunks (default:
8).
 <filename>   : Name of .yelc file with FLX-device E-link configuration.
```

6.4.5 femu

The **femu** tool gives users command line control over the FELIX firmware data generators, both in the from and to host directions.

See help text below for the list of options.

Help text of **femu***:*

```
femu version 22030800
Show or configure 'FanOut-Select' registers and start or stop
an emulator on a FELIX device.
Usage: femu -h|V -d<devnr> -e|E|n -l|L [-T|t -s<size> -i<idles>]
  -h         : Show this help text.
  -V         : Show version.
  -d<devnr>  : FLX-device to use (default: 0).
  -e|E|n     : Enable FLX-device data emulator, internal (e) or external (E) or
disable emulator (n).
               When no option is given the current status is displayed.
  -f         : When disabling emulator set TOHOST_FANOUT to emulator (default: to
external).
  -l         : 'Unlock' FanOut-Select registers.
  -L         : 'Lock' FanOut-Select registers.
  -i<idles>  : Number of idles between chunks, in bytes (multiple of 4)
  -s<size>   : L1A-triggered emu chunk size, in bytes (multiple of 4)
               (in combination with option -t or -T).
  -T         : Select the alternative ('logic') L1A-triggered emulator
               (also option -s required; FULL firmware only).
  -t         : Select the alternative ('logic') emulator, untriggered
               (also options -i and -s required; FULL firmware only).
```

6.4.6 ffmemu

The **ffmemu** tool gives users command line control over the data generator of the FMEMU (FULL mode Emulator) FELIX firmware.

See help text below for the list of options.

Help text of **ffmemu***:*

```
ffmemu version 21091300
Configure or show configuration of the FULLMODE emulator firmware.
Starting the emulator is optional.
Enable 2-bit E-link 0 (egroup 0, epath 0, 8b10b) for XOFF.
Enable 8-bit E-link 9 (egroup 1, epath 1, TTC-3) for TTC.
Usage: ffmemu [-h|V] [-d <devnr>] [-c] [-s] [-i <idles> [-t <cnt>] [-T] [-w <words>]
[-X] [-R <seed>]
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device to use (default: 0).
  -c         : Only configure the FMEMU; required for options -i,-R,-t,-T,-w,-X;
               stops the emulator, unless in combination with option -s.
  -E         : Issue an ECR.
  -i <idles> : Set idles-between-chunks count, range [0..0xFFFF].
  -R <seed>  : Set random seed, in combination with random chunk sizes (-w 0), range
[1..0x3FF].
  -s         : Restart the emulator, after any configuration to be done.
  -t <cnt>   : Number of triggers (chunks) to generate (0=unlimited, [1..65534]).
  -T         : Enable TTC-triggered mode (default: free running).
  -w <words> : Set chunk 4-byte word size (0=random, [3..65535]).
  -X         : Enable XON/XOFF.
```

6.4.7 fttcemu

The FELIX TTC emulator can be programmed through the **fttcemu** tool available in the FELIX software. The status of the FELIX TTC emulator is shown running the command **fttcemu**, which displays the values of the various FELIX TTC emulator parameters. This is an example of what is displayed:

```
$ fttcemu
Status:
TTC_EMU_SEL=0, TTC_EMU_ENA=0
TTC_EMU_BCR_PERIOD=3564
TTC_EMU_ECR_PERIOD=0
TTC_EMU_L1A_PERIOD=0
```

See for available options the help text below.

Help text of **fttcemu***:*

```
fttcemu version 23011300
Show or configure TTC emulator registers and enable or disable the TTC emulator.

Usage: fttcemu [-h|V] [-c <cardnr>] [-e|n] [-B <bc>] [-E <period>]
               [-f <freq>] [-L <cnt>] [-t <us>] [-R] [-X <xl1id>]
```

```
  -h         : Show this help text.
  -V         : Show version.
  -c <cardnr>: FLX-card to use (default: 0).
  -e|n       : Enable (-e) or disable (-n) the TTC emulator on the selected card.
  -b <ena>   : Enable (1) or disable (0) TTC emu Busy-In (default: leave untouched)
  -B <bc>    : Set the BCR period, in units of BC (Bunch Count);
               for <bc> equal to 0 a single BCR is generated.
  -E <period>: Set the ECR period, in ms;
               for <period> equal to 0 a single ECR is generated.
  -f <freq>  : Set the TTC emulator L1A freqency, in Hz.
               (any individually generated L1As (option -L) done first)
  -L <cnt>   : Generate <cnt> L1A triggers, using the interval set by -t (default: 0).
               (any single BCR or ECR is generated first).
  -t <us>    : The interval (in microseconds, default: 0)
               between individually generated L1As (option -L).
  -R         : Reset the TTC emulator; also reset the TTC decoder and XL1ID.
  -X <xl1id> : Set value of XL1ID-after-reset and execute XL1ID reset.
Note: options -B, -E, -f and -L also enable the TTC emulator, if necessary.
When no option is given the current TTC emulator status (register contents) is
displayed.
```

The TTC emulator can be enabled and disabled on the fly. TTC_EMU_SEL selects the TTC Source. When set to '0', the TTC data comes from the decoder, when set to '1', the TTC data comes from the TTC emulator. TTC_EMU_ENA starts the emulator. When set to '0' the emulator does not produce any data. When set to '1' the emulator is running. The variables TTC_EMU_SEL and TTC_EMU_ENA are both controlled with the command 'fttcemu -e' (setting both parameters to '1') and 'fttcemu -n' (setting both parameters to '0').

The TTC emulator is able to generate periodic L1A, ECR and BCR signals. TTC_EMU_L1A_PERIOD is the L1A period in units of LHC clock period (25 ns) set by the user as a frequency using option -e. TTC_EMU_BCR_PERIOD is the BCR period in units of LHC clocks and by default has a value 3564 which is the default in the LHC experiments (representing a period of roughly 89.1 microsecond). TTC_EMU_ECR_PERIOD is the ECR period in units LHC clocks, but note that the **fttcemu** tool sets the ECR period in units of milliseconds (e.g. in the ATLAS experiment the ECR period is set to 5 seconds, so that would be achieved by using option '-E 5000')

Set an L1A frequence of 1000 Hz an ECR period of 1 second:

```
fttcemu -f1000 -E1000
```

Generate a single ECR and a BCR:

```
fttcemu -E0 -B0
```

Generate a single ECR, followed by 10 L1A triggers at 10 Hz, then switch to 1000 Hz L1A:

```
fttcemu -E0 -L10 -t100000 -f1000
```

6.4.8 fttcbusy

The **fttcbusy** tool gives an overview of various FELIX firmware BUSY settings, such as the E-link

TTC-BUSY status and enables, as well as the BUSY settings with respect to the main output FIFO and DMA operation and the status and settings of the board's BUSY output. Note that some settings are only effective when made on device 0 (zero) of a dual-device FELIX card, such as the FLX-712.

Here's an example of **fttcbusy** output:

```
> fttcbusy -T
TTC-BUSY timing: Prescale = 15 Width = 15 Limit-time = 15
E-link TTC-BUSY status (latched BUSY requests and enables for BUSY output):
GBT #00: TTC-BUSY=000000000000000  BUSY-ENA=000000000000000
GBT #01: TTC-BUSY=00000000000A020  BUSY-ENA=000000000000000
  E = 045 = 1-0-5 (not enabled)
  E = 04D = 1-1-5 (not enabled)
  E = 04F = 1-1-7 (not enabled)
GBT #02: TTC-BUSY=000000000000000  BUSY-ENA=000000000000000
GBT #03: TTC-BUSY=000000000000000  BUSY-ENA=000000000000000
GBT #04: TTC-BUSY=000000000000000  BUSY-ENA=000000000000000
GBT #05: TTC-BUSY=000000000000000  BUSY-ENA=000000000000000
GBT #06: TTC-BUSY=000000000000000  BUSY-ENA=000000000000000
GBT #07: TTC-BUSY=000000000000000  BUSY-ENA=000000000000000
GBT #08: TTC-BUSY=000000000000000  BUSY-ENA=000000000000000
GBT #09: TTC-BUSY=000000000000000  BUSY-ENA=000000000000000
GBT #10: TTC-BUSY=000000000000000  BUSY-ENA=000000000000000
GBT #11: TTC-BUSY=000000000000000  BUSY-ENA=000000000000000
BUSY-by-DMA  : *enabled=0
               ToHost=0 (BAR0:0) (latched=0)
               buffer free space: assert=200MiB (C800000) deassert=220MiB (DC00000)
BUSY-by-FIFO : enabled=0
               thresh: deassert=3FF assert=4FF
               status: low_crossed=0 high_crossed=0 (latched=0)
BUSY FullMode: busy=000000 (latched=000000)
TTC Bch/TType: *1
BUSY output  : *status=0, *inhibit=0, *master=0
(NB: items marked by * are global, access via card endpoint 0 only!)
```

Settings for the TTC-BUSY signal timing and BUSY-out signal can be configured. See help text below for the list of options.

Help text of **fttcbusy***:*

```
fttcbusy version 23083000
Displays BUSY-related settings and optionally E-link (LTI)TTC BUSY status and enables,
optionally clearing (latched) E-link BUSY bits.
With option -T the 'TTC BUSY accepted' register contents are displayed, as well as
the corresponding E-link numbers, while option -C clears these registers after being
displayed.
Also the tool may be used to configure the TTC-BUSY signal settings (limit, prescale,
width)
and BUSY output settings (master, inhibit, B-channel, DMA and FIFO tresholds).
Option -R resets the TTC decoder.
```

```
NB: some of the settings are only read and written via FLX-card device #0
    (in the output indicated by a '*').

Usage: fttcbusy -h|V -d<devnr> -C -R -G<linknr>|-T -L
                -e|n<elinknr> | -E|N<linknr>
                -l<limit> -p<prescale> -w<width>
                -m<b> -i<b> -b<b> -B<b>
                -X<thresh> -x<diff> -Y<thresh> -y<diff>
  -h           : Show this help text.
  -V           : Show version.
  -d <devnr>   : FLX-device number (default: 0).
  -C           : Clear (latched) TTC-BUSY register bits.
  -R           : Reset TTC decoder.
  -G <linknr>  : GBT-link number, implies option -T.
  -T           : Display per-(E)link TTC-BUSY info (default: all links).
  -L           : Display per-(E)link LTITTC-BUSY info (default: all links).
  -e|n<elinknr>: Set/reset e-link BUSY-enable of selected e-link number (hex).
  -E|N <linknr>: Set/reset e-link BUSY-enable for e-links of this link.
  -l <limit>   : Set TTC BUSY limit time parameter (16-bit).
  -p <prescale>: Set TTC BUSY prescale parameter (20-bit).
  -w <width>   : Set TTC BUSY width parameter (16-bit).
  -m <b>       : Set (1) or clear (0) Master BUSY.
  -i <b>       : Set (1) or clear (0) BUSY Inhibit (= BUSY off).
  -b <b>       : Enable(1) or disable(0) BUSY-by-DMA and BUSY-by-FIFO.
  -D <b>       : Enable(1) or disable(0) BUSY-by-DMA.
  -F <b>       : Enable(1) or disable(0) BUSY-by-FIFO.
  -B <b>       : Enable(1) or disable(0) TTC B-channel/TriggerType.
                 (NB: if set limits TTCtoHost rate to ca. 200KHz max)
  -X <thresh>  : Set BUSY-by-DMA assert threshold (in MiB).
  -x <diff>    : Set BUSY-by-DMA difference assert/deassert thresholds (in MiB).
  -Y <thresh>  : Set BUSY-by-FIFO assert threshold (hex, max=0xFFF).
  -y <diff>    : Set BUSY-by-FIFO difference assert/deassert thresholds (hex).
```

6.4.9 fexoff

The **fexoff** tool is used to display, enable or disable the XON/XOFF feature for FULL mode links.

Help text of **fexoff***:*

```
fexoff version 22012700
Enable, disable and display XOFF feature settings for FLX-device links,for FULL-mode
firmware.
Also displays info about FIFO thresholds crossed (THRESH-X), as well as XON statistics
(STATS).
Without keyword '(re)set' the current settings and status are displayed.

Usage: fexoff [-h|V] [-d <devnr>] [-C] [-L <low>] [-H <high>] [-G <lnk>] [set|reset]
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device to use (default: 0).
```

```
  -C         : Clear (latched) XOFF high-threshold-crossed bits.
  -G <lnk>   : Link number (default: all).
  -L <low>   : Set FIFO XOFF low threshold [1..15], in 1/16ths of size.
  -H <high>  : Set FIFO XOFF high threshold [1..15], in 1/16ths of size.
               (options -L/H can not be combined with a set/reset op)
  set        : Enable link XOFF.
  reset      : Disable link XOFF.
```

6.4.10 fexofftx

The **fexofftx** tool can be used to manually generate an XOFF or XON signal on a selected FULL mode link, for test purposes.

Help text of **fexofftx***:*

```
fexofftx version 21091300
Generate an XOFF or XON (toggles between them) on a to-frontend link, in FULL-mode
firmware.
Usage: fexofftx [-h|V] [-d <devnr>] -G <lnk>
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device to use (default: 0).
  -G <lnk>   : Link number.
```

6.4.11 feto

The **feto** tool gives users command line control over the FELIX block timeout at the level down to individual E-links. If enabled FELIX will time out incoming data blocks taking longer than a designated period to arrive and attach a timeout trailer to the block. The block will then be transferred to the host as normal. The complete list of options of the tool can be seen in the help text below.

Help text of **feto***:*

```
feto version 23030300
Enable, disable or display either the instant time-out setting,
a setting per e-path (e-link), or the so-called global time-out
and associated time-out counter (number of clocks until time-out),
or the TTC time-out and associated counter (RM4 only).
Without keyword '(re)set' the current setting of the requested
(group of) time-outs is displayed (one complete link at most).

Usage: feto [-h|V] [-d <devnr>] [-e <elink>] [-G <lnk> [-g <group>] [-p <path>]]
            [-T] [set|reset] [<globcntr>]
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device to use (default: 0).
  -e <elink> : E-link number (hex) or use -G/g/p options.
  -G <lnk>   : (lp)GBT-link number.
```

```
  -g <group> : Group number (default: all groups).
  -p <path>  : E-path number (default: all paths).
  -T         : Read or configure TTC time-out (RM4 only).
  set        : Enable time-out.
  reset      : Disable time-out.
  <globcntr> : Global or TTC time-out counter value to set.
Examples:
> feto set        (Enable global time-out)
> feto reset 1000 (Disable global time-out, set counter to 1000)
> feto -G1 set    (Enable instant time-out on all E-links of link 1)
> feto            (Show global time-out and counter and instant time-out registers)
```

6.4.12 febrc

The **febrc** tool is used to assign the FromHost broadcast capability to individual E-links, or sets of E-links, e.g. all those belonging to a particular E-group number or all belonging to a link.

Data uploaded to a broadcast E-link number, as chosen by the user, is sent to each E-link enabled and enabled for broadcast matching the broadcast E-link number.

A broadcast can be made to:

- one E-link on all links, provided the E-link is broadcast-enabled on each applicable link,
- all (broadcast-enabled) E-links on one link,
- all (broadcast-enabled) E-links on all links.

See help text below for the list of options.

Help text of **febrc***:*

```
febrc version 23052300
NOTE: for RM5 firmware only.
Enable, disable or display the broadcast enable setting,
a setting per e-path (e-link)
Without keyword '(re)set' the current setting of the requested
(group of) broadcast enables is displayed (one complete link at most).

Usage: febrc [-h|V] [-d <devnr>] [-e <elink>] [-G <lnk> [-g <group>] [-p <path>]]
             [set|reset]
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device to use (default: 0).
  -e <elink> : E-link number (hex) or use -G/g/p options.
  -G <lnk>   : (lp)GBT-link number.
  -g <group> : Group number (default: all groups).
  -p <path>  : E-path number (default: all paths).
  set        : Enable time-out.
  reset      : Disable time-out.
Examples:
> febrc -G4 set    (Enable broadcast on all E-links of link 4;
```

```
                    broadcast e.g. on link 4 using E-link 0x13F)
> febrc -g1 set    (Enable broadcast on E-links of E-group 1 of all links;
                    broadcast e.g. on all links using E-link 0x7ff)
> febrc -e42 set   (Enable broadcast on E-link 0x042 (on Link 1, E-group 0, path 2);
broadcast e.g. on all Links E-group 0, path 2 using E-link 0x7c2)
> febrc            (Show broadcast enable registers content)
```

6.4.13 fflash

The **fflash** tool is designed specifically for loading a selected previously programmed firmware image from FLASH memory aboard a FLX-711 or FLX-712 into the card's FPGA, if the firmware image loaded at power-up is not the required one, or because one wants to switch between versions e.g. for test purposes. At power-up of the FELIX card the firmware image as selected by the setting of onboard switches will become the operational firmware version. Note that after loading a different firmware either a host machine reboot or PCIe hotplug operation is required (see 5.1.1 PCIe hotplug procedure for details) to return the board to normal operation with the new firmware image operational.

See help text below for the list of options plus some examples.

Help text of **fflash***:*

```
fflash version 21020800
Tool for loading a firmware image from one of the partitions
of the onboard flash memory of an FLX-712 into the card's FPGA,
issueing commands to the host system I2C bus to achieve this.
A subsequent hotplug procedure or machine reboot is required.

Usage: fflash [-h|V] [-q] -f<flashnr>
              [[-L|I] [-U|P -d<devslot>] [-S] [-b<busnr>] [-r<chan>] [-R<raddr>]
              [-s<saddr>] [-u<uaddr>] [-T<sec>]]
  -h          : Show this help text.
  -V          : Show version.
  -q          : Be quiet (only errors will be displayed).
  -f <flashnr>: Flash memory segment partition [0..3] selection (no default).
  -I          : Generate an INIT_B pulse on the FLX-card (to reset flash devices).
  -L          : Load firmware from the given flash partition into the card.

The following options are relevant in conjunction with the -L and/or -I option:
  -b <busnr>  : I2C bus number (default=0).
  -r <chan>   : Riser card I2C-switch channel number (default=0)
                (Select I2C-switch address using option -R).
  -R <raddr>  : Riser card I2C-switch I2C address (default 0x70).
  -s <saddr>  : I2C-switch I2C address (hex, default=0x77, expected range: 0x70-0x77).
                NB: 0x70 already taken by the riser card I2C-switch!
  -u <uaddr>  : Embedded microcontroller I2C address
                (hex, default=0x67, expected range: 0x60-0x67).
  -U          : Use USB I2C-dongle instead of system SMBus
                (requires scripts i2cset.py and i2cget.py installed in /opt/flx).
  -P          : Use 'ipmitool' to access system SMBus.
```

```
                !NB: use -d option to select 'device slot': 1 or 2.
  -T <sec>    : Set 'Prog-done' timeout [s] (default: 7)
  -d <devslot>: Device slot (1 or 2), only in combination with -P.
  -S          : Precede calls to i2cget/set or ipmitool with 'sudo'.
                (default: 'sudo' not used; applies to options -L|I|P|U).
Examples:
---------
Load flash memory image partition #2 into the card:
  fflash -f2 -L

Load flash memory image partition #2 into the card, using I2C-bus #1,
riser card I2C-switch channel #0, FLX-card I2C-switch address 0x75 and
FLX-card microcontroller I2C-address 0x65:
  fflash -f2 -L -b1 -r0 -s75 -u65

How to determine the I2C-switch and uC I2C addresses
(options -s and -u respectively):

Note 1: there is an I2C-bus number (option -b) to select as well,
  which is assumed to have the value '1' (following '-y') in the examples below.
Note 2: in the standard FELIX server there is an additional I2C-switch
  on the socalled riser card; its channel is selected using option -r;
  its I2C address (default 0x70) can be selected using option -R;
  it means that the 2 FLX-cards in such a server may have identical
  '-s' and '-u' addresses, i.e. most likely their defaults
  while the riser card setting is: 'top' position = -r 0, 'bottom' = -r 1.

'sudo i2cdetect -y 1' should show you an address in the range 0x70-0x77,
let's say 0x77; this is then the address to use in option -s;
subsequently run 'sudo i2cset -y 1 0x77 1' to set the I2C-switch
causing an additional address in the range 0x60-0x67 to appear
in the output of 'sudo i2cdetect -y 1', so run that command again;
this is the address to use in option -u.

On the FLX-712 dipswitch J14 configures the '-s' and '-u' addresses:
  switch 1-3 to set 3 LSBs of '-s', i.e. 0x70-0x77
  switch 4-6 to set 3 LSBs of '-u', i.e. 0x60-0x67
```

6.4.14 fflashprog

The **fflashprog** tool is designed specifically for programming FLASH memory aboard a FLX-711 or FLX-712 from an .mcs file containing a firmware image in Intel-HEX format. On the FLX-712 card up to 4 separate firmware images may be stored. One of the images, as selected by onboard switches, will be loaded into the card at power-up and become the operational firmware; if another of the stored images is required, the **fflash** tool is used to accomplish that.

In addition the **fflashprog** tool can be used to verify an image against an .mcs file or if necessary, to erase a firmware image from memory.

See help text below for the list of options plus some examples.

Help text of **fflashprog***:*

```
fflashprog version 20040900
Tool for programming, verifying, erasing or dumping firmware images,
stored in a FLX-711/712 card's flash memory.
(to load a selected firmware image into the FLX-card's FPGA use fflash)
Usage: fflashprog [-h|V] [-q] [-c <cardnr>] -f <flashnr> [-D] [-E] [-F]
                  [<filename>] [prog]
  -h          : Show this help text.
  -V          : Show version.
  -q          : Be quiet (only errors will be displayed).
  -c <cardnr> : FLX-card selected (default: 0).
  -d <devnr>  : FLX-device to use (default: 0) OBSOLETE: use -c.
  -D          : Read and display contents of the selected flash partition or flash
file.
  -E          : Erase the selected flash partition.
  -f <flashnr>: Flash memory segment partition [0..3] to dump, to erase,
                to verify or to program (no default).
  -F          : Use the (slow) word-by-word instead of (fast) page programming method.

  <filename>  : Name of MCS file to dump, verify or program.
  prog        : Literal text string to initiate flash programming
                (or else flash verification will take place).
Examples:
---------
Read and dump to screen flash memory image partition #2:
  fflashprog -f2 -D
Erase flash memory partition #2:
  fflashprog -f2 -E
Verify flash memory partition #2 against mcs file <filename>:
  fflashprog -f2 <filename>
Program flash memory partition #2 with the contents of mcs file <filename>:
  fflashprog -f2 <filename> prog
Read flash ID only:
  fflashprog -f0
Extra:
Read and dump to screen the memory image in mcs file <filename>:
  fflashprog -D <filename>
```

6.5 FELIX Data Debugging Tools

6.5.1 fcheck

fcheck is a debugging tool which can analyse the .dat files produced by the **fdaq** tool and check for data integrity issues. The tool will perform checks on a file to a specified degree of severity. As well as running checks, the tool can also be used to dump selected data blocks to screen, either split into data chunks or as raw data, to facilitate closer visual inspection of data or of any of the issues found by **fcheck**. Optionally the tool decodes data chunks as GBT-SCA replies, IC-channel replies or TTC-to-Host messages.

To run a data integrity check, specify the file name and the check detail level as follows:

`fcheck -B<level> testfile.dat`

See help text below for the list of options.

Help text of **fcheck***:*

```
fcheck version 24091300
Usage: fcheck -h|V -A -B<id> -c|C|D -e<elink> -F<blocks> -S<blocks> -H
              -x<kbyte> -X -t|T -w -0|O -2|4|8 <filename>
  -h         : Show this help text.
  -V         : Show version.
  -A         : Decode and display data chunks that are GBT-SCA,
               IC or TTCtoHost frames.
  -B <lvl>   : Do a check on (emulator) data blocks according to <lvl>,
               and display a data summary (default: 2):
               0: Check for proper block headers at block boundaries,
                  for each block 1 line of output is produced.
               1: Same as 0, but only when an error is found a line is output.
               2: Full integrity checking of blocks, starting from
                  the block trailer going through all chunks.
               3: Same as 2, including a check on expected emulator data payload,
                  which must constitute an incrementing byte.
               4: Same as 3, but inconsistent maximum values of L1ID are not reported.
  -c         : Display data 'raw' datablocks (default: chunk data) (with option -F)
  -C         : Display chunk data bytes only, nothing else.
  -D         : Display only whole data chunks, i.e. the user's data frames.
  -e <elink> : E-link number (hex) to filter for block check or block display.
  -F <blocks>: Dump <blocks> FLX data blocks to display (overrules data check option
-B).
               Chunk types: BOTH="<<", FIRST="++", LAST="&&", MIDDLE="==",
               TIMEOUT="]]", NULL="@@", OUTOFBAND="##",
               followed by 'T' for chunk truncation and/or 'E' for error.
  -S <blocks>: Skip <blocks> of data blocks before starting check or display.
  -t         : Do NOT report chunk truncation/error/CRCerror.
  -T         : Do NOT report chunk CRCerror.
  -w         : Instead of displaying, write (binary) chunkdata to file (dataout.dat).
  -0         : Display time-out chunkdata bytes (zeroes) (default:not).
  -O         : Do not display time-out chunks at all.
  -x <kbyte> : Set size of FLX-device unit data block, in KB (default:1)
               (NB: overrules any blocksize derived from data)
  -X         : Trailer size is 32-bit (default:16)
  -2         : Display chunk data as 2-byte words (little-endian).
  -4         : Display chunk data as 4-byte words (little-endian).
  -8         : Display chunk data as 8-byte words (little-endian).
 <filename>  : Name of file containing data to check or display.
```

Here are some examples of the tool's use:

Run a full integrity check of all data blocks in file 'file.dat', reporting data chunk errors as well as

data block corruption, including block number, E-link number and block word index:

```
fcheck -B2 file.dat
```

Display the data chunks from 2 data blocks starting from block number 1000 in file 'file.dat':

```
fcheck -S1000 -F2 file.dat
```

Display the raw data from 2 data blocks originating from E-link 8 starting from block number 1000:

```
fcheck -S1000 -F2 -e8 -c file.dat
```

Display the data from 2 data blocks originating from E-link 8 as 4-byte items starting from block number 1000, and not displaying the zeroes of time-out chunks, just their sizes:

```
fcheck -S1000 -F2 -e8 -4 -0 file.dat
```

Display the data chunks from 2 data blocks and decode the chunks that look like they might be replies from a GBT-SCA device, including control byte, transaction ID, channel number, length, error byte and data (option -A can be used as well to decode chunks originating from a *TTC-to-Host* virtual E-link, displaying clearly the various items and counters contained):

```
fcheck -F2 -A -0 file.dat
```

6.5.2 fedump

The **fedump** tool is designed to make it possible to dump data arriving at FELIX directly to screen for debugging purposes. Users of the tool can filter the data stream by E-link ID and FELIX card number, and also have the option of displaying the data split into its FELIX-format data chunks (and optionally skip the time-out chunk data or time-out chunks all together) or in a raw format, with payload data optionally displayed as either 1-, 2-, 4- or 8-byte values.

See help text below for the list of options.

Help text of **fedump***:*

```
fedump version 24042200
Dump selected E-link chunk data (optionally block-by-block)
received from an FLX-device to screen.
Chunk types are delimited by:
BOTH="<<", FIRST="++", LAST="&&", MIDDLE="==", TIMEOUT="]]", NULL="@@", OUTOFBAND="##"
Usage:
 fedump -h|V -A|B -c|D -d<devnr> -e<elink> -i<dma> -I -0|O -2|4|8 -H
  -h         : Show this help text.
  -V         : Show version.
  -A         : Decode and display data chunks that are GBT-SCA reply,
               IC reply or TTCtoHost frames.
  -B         : Interpret chunks that could be GBT-SCA or IC requests.
  -c         : Display data 'raw', block-by-block (default: chunk data).
  -d <devnr> : FLX-device to use (default: 0).
  -D         : Display only whole data chunks, i.e. the user's data frames,
               merging FIRST, MIDDLE and LAST subchunks.
  -e <elink> : E-link number (hex) to filter out for display (default: no filter).
```

```
  -H         : Chunks have headers (default: auto-detect).
  -i <dma>   : FLX-device DMA controller to use (default: 0).
  -I         : Use interrupt to receive data (default: polling).
  -0         : Do not display time-out chunkdata (zeroes).
  -O         : Do not display time-out chunks at all.
  -2         : Display data as 2-byte words (little-endian).
  -4         : Display data as 4-byte words (little-endian).
  -8         : Display data as 8-byte words (little-endian).
```

6.6 GBTX and lpGBT Configuration Tools

6.6.1 fice

The **fice** tool communicates on the IC channel of a GBT or lpGBT link, to read and write registers on a GBTX or lpGBT chip present on a remote system connected to this link, in order to configure the GBTX or lpGBT as required. Optionally **fice** can configure a **secondary** lpGBT via the EC-link of the *primary* lpGBT, or alternatively even through an I2C Master of the primary lpGBT.

Individual registers may be read or written or a file containing a range of register settings can be read in by the tool and written to the GBTX or lpGBT. In addition the contents of a file with register settings can be *compared* to the current register settings of a GBTX or lpGBT, where any differences are displayed.

See help text below for the list of options.

Help text of **fice***:*

```
fice version 24032200
Tool to read or write GBTX or lpGBT registers via the IC-channel
of an FLX-device (lp)GBT link,
and for lpGBT optionally from/to a secondary lpGBT via the primary lpGBT's EC-channel,
or from/to a secondary lpGBT via an I2C Master of the primary lpGBT.

Read or write a single byte from or to the given GBTX/lpGBT register address
or write to multiple consecutive GBTX/lpGBT registers using the contents
of a file (i.e. ASCII file: 1 (register) byte value (hex) per line of text,
e.g. the 'TXT' file generated by the GBTXProgrammer tool,
or alternatively per line of text an address followed by a byte value,
optionally followed by a comment text.
NB: in the latter case registers are always written one-by-one,
    as the registers may be listed in any order).
Using option -C the file contents is not used to *configure* but *compared*
to the current GBTX or lpGBT register contents.

Provide a file name *or* use option -a with an address and an optional additional byte
value
to read resp. write a single GBTX or lpGBT register or, without option -a,
to read (and display) *all* registers of the GBTX or lpGBT (v0 or v1).
```

```
Without both option -a and file name all registers are read out and displayed
either in one IC read operation or optionally one-by-one (option -o).
Reading via another lpGBT I2C Master (option -s) is either one-by-one or in groups of
16
(the maximum number of bytes that can be read in a single I2C Master operation).

Option -t displays the register values in a format that could be used
as a 'TXT' file for this tool or the I2C-dongle GBTX programmer.
Usage:
 fice -h|V -d<devnr> -G<lnk> -0|1 -e -i<dma> -I<i2c> -Z|R -t|T
      -m<master> -f<khz> -s<i2c> -a<addr> <byte>|<filename>
  -h         : Show this help text.
  -V         : Show version.
  -0         : If lpGBT, assume v0 (default: auto-detect).
  -1         : If lpGBT, assume v1 (default: auto-detect).
  -a <addr>  : GBTX/lpGBT register address (decimal or hex, 0x.. or x..)
               to read or write.
  -C         : In combination with <filename>: compare GBTX/lpGBT register
               contents to file contents and display the differences.
  -d <devnr> : FLX-device to use (default: 0).
  -e         : Use the lpGBT EC-channel to access a secondary lpGBT.
  -f <freq>  : I2C Master bus frequency, in KHz
               (only with -s; 100,200,400 or 1000; default:100).
  -G <lnk>   : GBT-link number.
  -i <dma>   : FLX-device DMA controller for receiving (default: 0).
  -I <i2c>   : GBTX/lpGBT I2C address (hex).
  -m <master>: I2C Master index (only with -s; 0,1 or 2; default:0).
  -o         : When reading or writing all (consecutive) registers, do it one-by-one
               (default: single multi-register read/write operation when possible).
  -R         : Receive replies on any E-link.
  -s <i2c>   : I2C address (hex) of a secondary lpGBT accessed via an lpGBT
               I2C Master (NB: feature available for lpGBT v1 only, for now).
  -t         : Display one register value per line in output
               (i.e. 'TXT'-format like).
  -T         : Display one address + register value per line in output
               (i.e. similar to 'TXT'-format, but different..).
  -Z         : Do NOT receive and process/display replies.
 <byte>      : Byte value (decimal or hex, 0x.. or x..) to write to
               a GBTX/lpGBT register (option -a).
 <filename>  : Name of text file with GBTX/lpGBT (hex) register data to compare
against,
               or to write to consecutive registers (if one value per line;
               also accepts files with address+value (both hex) per line,
               separated by a space, in which case registers are written one-by-one).
=> Examples:
Read all registers of GBTX/lpGBT (I2C address 0x71)
connected to FLX-device GBT link 3:
  fice -G3 -I71
Read GBTX/lpGBT register 32 (0x20):
  fice -G3 -I71 -a 32 (or: fice -G 1 -I 3 -a 0x20)
Write 0xA5 to GBTX/lpGBT register 32 (0x20):
```

```
  fice -G3 -I71 -a 32 0xA5
Write contents of GBT-conf.txt to GBTX/lpGBT registers:
  fice -G3 -I71 GBT-conf.txt
Compare contents of GBT-conf.txt to GBTX/lpGBT registers:
  fice -G3 -I71 -C GBT-conf.txt
Read all registers of a secondary lpGBT (I2C address 0x72)
connected to the EC-link of an lpGBT connected to FLX-device GBT link 3:
  fice -G3 -I72 -e
Read all registers of a secondary lpGBT (I2C address 0x72)
connected to I2C Master 1 (@400KHz) of an lpGBT (I2C address 0x71)
connected to FLX-device GBT link 3:
  fice -G3 -I71 -s72 -m1 -f400
```

Example of **fice** output when reading all registers of a GBTX device connected to the GBT link #3 IC-channel, with I2C address 1 (each line starts with a decimal register address, followed by 16 hexadecimal register byte values):

```
$ fice -G0 -I1
Opened FLX-device 0, firmw FLX712-GBT-2x4CH-220905-2241-GIT:rm-4.11/104
>>> Detected lpGBTv0/GBTX
>>> GBTX@Lnk0 I2C-addr=0x1: READ all registers
Reply (size=444): Parity OK
Reg 0x000 (  0):  f7 dc ef ec 71 bf 7f e9 00 00 00 00 00 00 00 00
Reg 0x010 ( 16):  00 00 00 00 00 00 00 00 00 02 00 28 00 15 15 15
Reg 0x020 ( 32):  66 00 0d 42 00 0f 04 08 00 20 00 00 00 00 15 15
Reg 0x030 ( 48):  15 00 00 00 38 00 00 00 00 00 00 00 00 00 00 15
Reg 0x040 ( 64):  dd 0d 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Reg 0x050 ( 80):  00 ff ff ff 00 00 00 15 dd 0d 00 00 00 00 00 00
Reg 0x060 ( 96):  00 00 00 00 00 00 00 00 00 ff ff ff 00 00 00 15
Reg 0x070 (112):  dd 0d 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Reg 0x080 (128):  00 ff ff ff 00 00 00 15 dd 0d 00 00 00 00 00 00
Reg 0x090 (144):  00 00 00 00 00 00 00 00 00 ff ff ff 00 00 00 15
Reg 0x0a0 (160):  dd 0d 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Reg 0x0b0 (176):  00 ff ff ff 00 00 00 00 00 70 00 00 00 00 00 00
Reg 0x0c0 (192):  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Reg 0x0d0 (208):  00 70 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Reg 0x0e0 (224):  00 00 00 00 00 00 00 dd 0d 70 00 00 00 00 00 00
Reg 0x0f0 (240):  00 00 3f 3f 38 00 00 00 07 00 00 07 00 00 71 ff
Reg 0x100 (256):  ff 01 ff ff 01 ff ff 01 ff ff 01 ff ff 00 00 00
Reg 0x110 (272):  00 20 00 00 00 00 00 00 00 15 00 00 00 00 00 00
Reg 0x120 (288):  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Reg 0x130 (304):  00 00 00 00 00 00 00 00 00 4e 4e 4e aa 0a 07 00
Reg 0x140 (320):  ff ff ff ff ff 00 00 88 88 88 88 88 01 ff ff 01
Reg 0x150 (336):  ff ff 01 ff ff 01 ff ff 01 ff ff 01 ff ff 01 ff
Reg 0x160 (352):  ff 01 ff ff 01 ff ff 01 ff ff f8 00 00 aa 00 d2
Reg 0x170 (368):  67 82 e0 81 81 81 a5 00 00 ff cd cd cd 00 00 00
Reg 0x180 (384):  00 00 00 00 aa bb 9f ff ff ff ff ff ff ff 10 00
Reg 0x190 (400):  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Reg 0x1a0 (416):  00 00 00 00 00 00 00 00 00 00 00 0a 19 ff 00 61
```

```
Reg 0x1b0 (432):  ef fb bd fe
```

Write 0x34 to register 2 of that same GBTX device:

```
$ fice -G0 -I1 -a2 34
Opened FLX-device 0, firmw FLX712-GBT-2x4CH-220905-2241-GIT:rm-4.11/104
>>> Detected lpGBTv0/GBTX
>>> GBTX@Lnk0 I2C-addr=0x1: WRITE 0x22 (34) to reg 0x002 (2)
Reply (size=9): Parity OK  Reg 0x002 (  2):  22
```

Read register 2 of the GBTX device:

```
$ fice -G0 -I1 -a2
Opened FLX-device 0, firmw FLX712-GBT-2x4CH-220905-2241-GIT:rm-4.11/104
>>> Detected lpGBTv0/GBTX
>>> GBTX@Lnk0 I2C-addr=0x1: READ reg 0x002 (2)
Reply (size=9): Parity OK  Reg 0x002 (  2):  22
```

6.6.2 flpgbtconf

The **flpgbtconf** tool communicates on the IC channel of an lpGBT link, to read and write registers on the lpGBT chip at the other end of this link. Optionally **flpgbtconf** can read/write from/to a **secondary** lpGBT via the EC-link of this *primary* lpGBT. The tool allows the user to read and write individual lpGBT registers and bit field items in lpGBT registers, *by name*, as listed and described in the lpGBT manual. A special case is name *list* which will result in all known items being displayed (this may be useful in combination with the 'grep' tool; see example below).

The tool can be considered an addition to fice, which is used to do a 'full' configuration of all registers of an lpGBT using a configuration file.

Example of providing a keyword to **flpgbtconf** to obtain suggestions for names of existing lpGBT items:

```
$ flpgbtconf clk0
->Item not found, suggested options are:
EPCLK0CHNCNTRH
EPCLK0CHNCNTRH_EPCLK0INVERT
EPCLK0CHNCNTRH_EPCLK0DRIVESTRENGTH
EPCLK0CHNCNTRH_EPCLK0FREQ
EPCLK0CHNCNTRL
EPCLK0CHNCNTRL_EPCLK0PREEMPHASISSTRENGTH
EPCLK0CHNCNTRL_EPCLK0PREEMPHASISMODE
EPCLK0CHNCNTRL_EPCLK0PREEMPHASISWIDTH
```

Note that names are case-insensitive, with '-' equivalent to '_', and a name doesn't have to be complete, but has to be unique, so to read one of the items above from an existing lpGBT (the tool

automatically determines if it's dealing with a v0 or a v1), for example:

```
$ flpgbtconf -G0 -I71 epclk0chncntrh-epclk0i
Opened FLX-device 0, firmw FLX712-LPGBT-2x2CH-230805-1227-GIT:rm-5.0/3446
>>> Detected lpGBTv1
EPCLK0CHNCNTRH_EPCLK0INVERT: addr=0x06E (110) size=1bits index=6 RW
"Inverts 0 clock output."
=> Read:
Address 0x06E: 0x00
EPCLK0CHNCNTRH_EPCLK0INVERT: 0x0 (0)
```

Configuring individual items at the same register address:

```
$ flpgbtconf -G0 -I71 epclk0chncntrh-epclk0i 1
Opened FLX-device 0, firmw FLX712-LPGBT-2x2CH-230805-1227-GIT:rm-5.0/3446
>>> Detected lpGBTv1
EPCLK0CHNCNTRH_EPCLK0INVERT: addr=0x06E (110) size=1bits index=6 RW
"Inverts 0 clock output."
=> Read:
Address 0x06E: 0x00
EPCLK0CHNCNTRH_EPCLK0INVERT: 0x0 (0)
=> Write:
Address 0x06E: 0x40
=> Reread:
Address 0x06E: 0x40
EPCLK0CHNCNTRH_EPCLK0INVERT: 0x1 (1)

$ flpgbtconf -G0 -I71 epclk0chncntrh-epclk0f 2
Opened FLX-device 0, firmw FLX712-LPGBT-2x2CH-230805-1227-GIT:rm-5.0/3446
>>> Detected lpGBTv1
EPCLK0CHNCNTRH_EPCLK0FREQ: addr=0x06E (110) size=3bits index=0 RW
"Sets the frequency for 0 clock output."
=> Read:
Address 0x06E: 0x40
EPCLK0CHNCNTRH_EPCLK0FREQ: 0x0 (0)
=> Write:
Address 0x06E: 0x42
=> Reread:
Address 0x06E: 0x42
EPCLK0CHNCNTRH_EPCLK0FREQ: 0x2 (2)
```

Then to see the list of all items containing keyword 'CLK0' and their values, one could do this (note: in this case keyword has to be given in uppercase):

```
$ flpgbtconf -G0 -I71 list | grep CLK0
EPCLK0CHNCNTRH                                0x06E RW   8b [7:0] :   0x42  (66)
EPCLK0CHNCNTRH_EPCLK0LOWRES                   0x06E RW   1b [7]   :   0x0   (0)
EPCLK0CHNCNTRH_EPCLK0INVERT                   0x06E RW   1b [6]   :   0x1   (1)
```

```
EPCLK0CHNCNTRH_EPCLK0DRIVESTRENGTH                0x06E RW   3b [5:3] :   0x0   (0)
EPCLK0CHNCNTRH_EPCLK0FREQ                         0x06E RW   3b [2:0] :   0x2   (2)
EPCLK0CHNCNTRL                                    0x06F RW   8b [7:0] :   0x00  (0)
EPCLK0CHNCNTRL_EPCLK0PREEMPHASISSTRENGTH          0x06F RW   3b [7:5] :   0x0   (0)
EPCLK0CHNCNTRL_EPCLK0PREEMPHASISMODE              0x06F RW   2b [4:3] :   0x0   (0)
EPCLK0CHNCNTRL_EPCLK0PREEMPHASISWIDTH             0x06F RW   3b [2:0] :   0x0   (0)
```

See help text below for the list of options.

Help text of **flpgbtconf***:*

```
flpgbtconf version 22051800
Read or write an lpGBT register or register bitfield, by address or name.
The value of the item is read and displayed, and if requested,
written to and re-read and again displayed.
Requires the IC channel (or EC-channel) of the FLX device to be enabled.
Without options -G and -I some information about the selected register
or bitfield is displayed.
'flpgbtconf list' displays all known 'item' names plus additional info
about each item (for either lpGBTv0 or lpGBTv1: auto-detected or
forced by means of option -0 or -1), and in combination with options -d/-G/-I
displays the current value of each item of the selected lpGBT as well.
(NB: for GBTX devices use the fgbtxconf tool).

Usage: flpgbtconf [-h|V] [-d<devnr>] [-D<dma>] [-G<link> -I<i2c>]
                  [-0|1] [-e] [-X] [-Z]
                  <name> [<value>]
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device number (default: 0).
  -D <dma>   : FLX-device DMA controller for receiving (default: 0).
  -0         : Assume lpGBTv0, also for 'flpgbtconf list' or
               'flpgbtconf <name/addr>' or '-Z' (default: auto-detect).
  -1         : Assume lpGBTv1 (see -0; default: auto-detect).
  -e         : Use the EC-channel (default: IC-channel).
  -G <link>  : lpGBT link number.
  -I <i2c>   : lpGBT I2C address (hex).
  -X         : Debug mode: display bytes of received frames;
               also: continue, even when nothing is received (e.g. with -Z).
  -Z         : Do NOT receive and process/display replies.
 <name>      : Name of register or bitfield, or hex (0x) or decimal address.
               (Name "list" results in output of a list of all known items,
                including their current values in a connected lpGBT chip
                when options -G/I are provided).
 <value>     : Value to write to register or bitfield (hex or decimal).
```

The lpGBT also contains a number of control and monitoring channels. There are a few tools available dedicated to operate some of these; see flpgbtio and flpgbti2c.

6.6.3 fgbtxconf

The **fgbtxconf** tool is for an GBTX device what **flpgbtconf** is for an lpGBT device. It communicates on the IC channel of a GBT link, to read and write registers on the GBTX chip at the other end of the link. The tool allows the user to read and write individual GBTX registers and bit field items in GBTX registers, *by name*, as listed and described in the GBTX manual. A special case is name *list* which will result in all known items being displayed (this may be useful in combination with the 'grep' tool).

See help text below for the list of options.

Help text of **fgbtxconf***:*

```
fgbtxconf version 23072500
Read or write a GBTX register or register bitfield, by address or name.
The value of the item is read and displayed, and if requested,
written to and re-read and again displayed.
Requires the IC channel of the FLX device to be enabled.
Without options -G and -I some information about the selected register
or bitfield is displayed.
'fgbtxconf list' displays all known 'item' names plus additional info
about each item, and in combination with options -d/G/I displays
the current value of each item of the selected GBTX as well.
(NB: for lpGBT devices use the flpgbtconf tool).

Usage: fgbtxconf [-h|V] [-d<devnr>] [-D<dma>] [-G<link> -I<i2c>] [-X] [-Z]
                 <name> [<value>]
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device number (default: 0).
  -D <dma>   : FLX-device DMA controller for receiving (default: 0).
  -G <link>  : GBT link number.
  -I <i2c>   : GBTX I2C address (hex).
  -X         : Debug mode: display bytes of received frames;
               also: continue, even when nothing is received (e.g. with -Z).
  -Z         : Do NOT receive and process/display replies.
 <name>      : Name of register or bitfield, or hex (0x) or decimal address.
               (Name "list" results in output of a list of all known items,
                including their current values in a connected GBTX chip
                when options -G/I are provided).
 <value>     : Value to write to register or bitfield (hex or decimal).
```

6.6.4 fscai2cgbtx

The **fscai2cgbtx** tool allows a user to read and write GBTX registers via its I2C port, accessing it through a selected GBT-SCA chip I2C channel. The functionality of this tool is similar to fice, but for now works for GBTX only.

See help text below for the list of options.

Help text of **fscai2cgbtx***:*

```
fscai2cgbtx version 23080200
Tool to read or write GBTX registers via an I2C-channel of a GBT-SCA chip,
connected to any FLX-device GBT (2-bit HDLC) E-link:
read or write a single byte from or to the given GBTX register address
or write to multiple consecutive GBTX registers using the contents of a file.
(i.e. ASCII file: 1 (register) byte value (hex) per line,
 e.g. the 'TXT' file generated by the GBTXProgrammer tool).
Provide a file name *or* use option -a with an optional additional byte value
to read resp. write a single GBTX register or, without option -a, to read all
registers.
(NB: this tool comparable to fice tool, not fgbtxconf).
Usage:
 fscai2cgbtx [-h|V] [-d<devnr>] [-e<elink>] [-G<lnk> [-g<group> -p<path>]] [-R]
              [-r] [-W] -C<ichan> -I<iaddr> -a<addr>
              [<byte>] | <filename>
  -h         : Show this help text.
  -V         : Show version.
  -c         : In combination with <filename>: compare GBTX/lpGBT register
               contents to file contents and display the differences.
  -d <devnr> : FLX-device to use (default: 0).
  -e <elink> : E-link number (hex) or use -G/g/p options.
  -G <lnk>   : GBT-link number.
  -g <group> : Group number (default: 7=EC).
  -p <path>  : E-path number (default: 7=EC).
  -R         : Reset GBT-SCA.
  -r         : Do not receive and display the GBT-SCA replies.
  -W         : Read writable registers only (default: all).
  -C <ichan> : GBT-SCA I2C channel number.
  -I <iaddr> : GBTX I2C address (hex).
  -a <addr>  : GBTX register address (decimal or hex, 0x.. or x..) to read or write.
 <byte>      : Byte value (decimal or hex, 0x.. or x..)
               to write to GBTX register <addr> (option -a).
 <filename>  : Name of file with GBTX (hex) register data to compare against,
               or to write to consecutive registers (if one value per line;
               also accepts files with address+value (both hex) per line,
               separated by a space.
=> Examples:
Read all registers of GBTX (I2C address 1) connected to GBT-SCA I2C-channel 0,
GBT-SCA connected to FLX-device GBT link 3 EC-link:
  fscai2cgbtx -G3 -I1 -C0 (or: fscai2cgbtx -eff -I1 -C0)
Compare contents of GBT-conf.txt to GBTX registers:
  fscai2cgbtx -G3 -I1 -C0 -c GBT-conf.txt
Read GBTX register 32 (0x20):
  fscai2cgbtx -G3 -I1 -C0 -a32 (or: fscai2cgbtx -G3 -I1 -C0 -a0x20)
Write 0xA5 to GBTX register 32 (0x20):
  fscai2cgbtx -G3 -I1 -C0 -a32 0xA5
Write contents of GBTX-conf.txt to GBTX registers:
  fscai2cgbtx -G3 -I1 -C0 GBTX-conf.txt
```

6.7 GBT-SCA Tools

6.7.1 fec

The **fec** tool is designed for communication with a GBT-SCA chip present on a remote hardware system connected to FELIX via a GBT link. The tool allows to send pre-programmed commands to read out or write to a number of the hardware channels available on a GBT-SCA, such as GPIO, ADC and DAC. The GBT-SCA can be connected to any 2-bit (HDLC-encoded) E-link of the GBT, besides the EC channel.

See help text below for the list of options.

Help text of **fec***:*

```
fec version 21032300
Demo tool for control and read out of various devices on a GBT-SCA
through a GBT link's EC channel or any 2-bit wide, HDLC encoded E-link.
Receives (and displays) GBT-SCA replies, unless option -Z is given
(in that case use e.g. fedump or fdaq to receive the replies).
Usage:
 fec [-h|V] [-d <devnr>] [-i <dma>] [-I] [-N] [-G <lnk>] [-g <group>] [-p <path>]
     [-t <ms>] [-x <par>] [-A] [-C] [-R] [-T] [-P <secs>] [-X] [-Y <seq>] [-Z] [<ops>]
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device to use (default: 0).
  -i <dma>   : FLX-device DMA controller for receiving (default: 0).
  -I         : USE interrupt to receive data (default: polling)
  -N         : Receiver resets DMA at start-up (default: no reset).
  -G <lnk>   : GBT-link number (default: 0).
  -g <group> : Group number (default matches GBT EC 'group' = 7).
  -p <path>  : E-path number (default matches GBT EC 'path' = 7).
  -r <repeat>: Number of GPIO/ADC/DAC operations to perform (default: 1).
  -A         : Use SCA-V1 ADC commands (default: SCA-V2 ADC).
  -C         : Send GBT-SCA connect (HDLC control).
  -R         : Send GBT-SCA reset (HDLC control).
  -T         : Send GBT-SCA test (HDLC control).
  -t <ms>    : Time between some of the ops, in ms (default: 100).
  -P <secs>  : Enable FromHost (circular) DMA then pause for <secs> seconds
               (for DMA check/debug; default: no pause)
  -X         : Use continuous-mode DMA for upload (default: single-shot).
  -x <par>   : Parameter to use in operations, e.g. GPIO number, ADC or DAC channel
(default: 0).
  -Y <seq>   : Use <seq> as first HDLC 'receive sequence number'.
               (to keep receiving side happy in consecutive calls)
  -Z         : Do NOT receive and display the GBT-SCA replies.
  <ops>      : String of chars indicating which operation(s) to perform:
               o=GPIO-out, i=GPIO-in, a=ADC, d=DAC, I=I2C (no-string=default: none).
Examples:
Blink an LED on a VLDB (here connected to GBT link #3, EC-channel)
on GBT-SCA GPIO #18 (the other LED is on GPIO #21) 20 times
```

```
with a rate of 5Hz (100ms ON, 100ms OFF):
  fec -G3 -t100 -r20 -x18 o
Read GPIO inputs (GBT-SCA on GBT-link #3's EC-channel) 20 times
with a rate of 10Hz:
  fec -G3 -t100 -r20 i
```

Example: on the VLDB with GBT-SCA connected to the EC-channel of GBT link #3 blink one of the LEDs (connected to GPIO #18; the other one is connected to GPIO #21) 25 times (-r 50) with an on and off period of 100 ms (the last symbol is an 'oh', which stands for 'GPIO output'):

```
fec -G3 -r50 -t100 -x18 o
```

6.7.2 fscaid

The **fscaid** tool reads out and displays a GBT-SCA chip's ID register.

Help text of **fscaid***:*

```
fscaid version 21110900
Tool to read a GBT-SCA's Chip ID.
Usage:
 fscaid [-h|V] [-d<devnr>] [-e<elink>] [-E<elinkr>] [-G<lnk>] [-g<group>] [-p<path>]
        [-1] [-C] [-R] [-Z]
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device to use (default: 0).
  -e <elink> : E-link number (hex) or use -G/g/p options.
  -E <elinkr>: E-link number (hex) for receiving, if different from
               the E-link number for sending.
  -G <lnk>   : GBT-link number (default: 0).
  -g <group> : Group number (default matches GBT EC 'group' = 7).
  -p <path>  : E-path number (default matches GBT EC 'path' = 7).
  -C         : Send GBT-SCA connect (HDLC control).
  -R         : Send GBT-SCA reset (HDLC control).
  -Z         : Do not receive and display the GBT-SCA replies.
  -1         : Read ID from a GBT-SCA Version 1 (default: V2).
```

6.7.3 fscaio

The **fscaio** tool is used to read and write a GBT-SCA's GPIO lines, either individually or all 32 in one operation. Also the direction register can be configured. In addition **fscaio** can generate a configurable number of output pulses with a configurable width.

See help text below for the list of options.

Help text of **fscaio***:*

```
fscaio version 23060100
Tool to write and/or read the GBT-SCA GPIO bits and direction register.
Usage:
```

```
 fscaio [-h|V] [-d <devnr>] [-e <elink>] [-G <lnk>] [-g <group>] [-p <path>]
        [-i <bit>] [-o <dir>] [-C] [-R] [-D] [-E] [-Z] [-H] [<value>]
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device to use (default: 0).
  -e <elink> : E-link number (hex) or use -G/g/p options.
  -G <lnk>   : GBT-link number (default: 0).
  -g <group> : Group number (default matches GBT EC 'group' = 7).
  -p <path>  : E-path number (default matches GBT EC 'path' = 7).
  -i <bit>   : Read or write GPIO bit number <bit> (default: all).
               NB: if a single I/O pin is written to, its direction bit
                   is set to output (independent of option -o).
  -o <dir>   : Set GPIO direction register to value <dir> (hex).
  -C         : Send GBT-SCA connect (HDLC control).
  -R         : Send GBT-SCA reset (HDLC control).
  -r <rep>   : Generate <rep> pulses up/down or down/up before setting the (single)
output
               to the requested value (for debug/demo purposes).
  -t <us>    : Width of the -r pulses, in microseconds [2..1000] (default: 2).
  -Z         : Do not receive and display the GBT-SCA replies.
  -D         : Disable GBT-SCA GPIO channel after operation (default: leave enabled)
  -E         : Do *not* enable GBT-SCA GPIO channel at start of operation,
               assume it already is.
  -H         : Do not use HDLC delay packet, even if firmware supports it,
               use zero-byte packets (for output pulse width, options -r and -t).
 <value>     : Value to write (0 or 1 for a single GPIO bit,
               or up to 0xFFFFFFFF otherwise, hexadecimal);
               if no value is provided a read operation is performed.
```

6.7.4 fscaadc

The **fscaadc** tool reads out a GBT-SCA's ADC input channels, displaying raw as well as converted (to volts) values. In addition, ADC input channel current sources can be selectively enabled for the read-out.

Example of **fscaadc** output for a single ADC input scan for a GBT-SCA connected to the EC channel of GBT link #3 (here: the GBT-SCA on a VLDB):

```
$ fscaadc -C -G3
Opened FLX-device 0, firmw FLX712-GBT-4chan-2008271931-GIT:rm-4.10/544
GBT-SCA connect
ADC enabled
GBT-SCA ADC readings:
 0: 57B  =  1403  =  0.343 V
 1: 7A7  =  1959  =  0.478 V
 2: 61C  =  1564  =  0.382 V
 3: 6C7  =  1735  =  0.424 V
 4: 64B  =  1611  =  0.393 V
 5: 76E  =  1902  =  0.464 V
```

```
 6: 819  =  2073  =  0.506 V
 7: 836  =  2102  =  0.513 V
 8: 791  =  1937  =  0.473 V
 9: 7DB  =  2011  =  0.491 V
10: 71F  =  1823  =  0.445 V
11: 8DB  =  2267  =  0.554 V
12: 857  =  2135  =  0.521 V
13: 7FF  =  2047  =  0.500 V
14: 6BA  =  1722  =  0.421 V
15: 720  =  1824  =  0.445 V
16: 6DD  =  1757  =  0.429 V
17: 5EC  =  1516  =  0.370 V
18: 7E7  =  2023  =  0.494 V
19: 892  =  2194  =  0.536 V
20: 7FF  =  2047  =  0.500 V
21: 859  =  2137  =  0.522 V
22: 8B5  =  2229  =  0.544 V
23: 6CB  =  1739  =  0.425 V
24: 8A2  =  2210  =  0.540 V
25: 699  =  1689  =  0.412 V
26: CE1  =  3297  =  0.805 V
27: 024  =    36  =  0.009 V
28: 77C  =  1916  =  0.468 V
29: 000  =     0  =  0.000 V
30: FFF  =  4095  =  1.000 V
31: A9D  =  2717  =  0.663 V (T=30.9C approx.)
```

See help text below for the list of options.

Help text of **fscaadc***:*

```
fscaadc version 19031300
Tool to read GBT-SCA ADC input channels and display the readings.
Usage:
 fscaadc [-h|V] [-d <devnr>] [-e <elink>] [-G <lnk>] [-g <group>] [-p <path>] [-c
<msk>] [-A]
         [-i <index>] [-n <kohm>] [-r <cnt>] [-t <us>] [-C] [-D] [-E] [-R] [-X] [-Z]
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device to use (default: 0).
  -e <elink> : E-link number (hex) or use -G/g/p options.
  -G <lnk>   : GBT-link number (default: 0).
  -g <group> : Group number (default matches GBT EC 'group' = 7).
  -p <path>  : E-path number (default matches GBT EC 'path' = 7).
  -A         : Use SCA-V1 ADC commands (default: SCA-V2 ADC).
  -c <mask>  : Enable current sources on the ADC inputs in bitmask <mask>
               (disabled again afterwards).
  -i <index> : Conversion of ADC input <index> only (default: all 32 inputs
consecutively).
  -n <kohm>  : NTC reference resistance value in KOhm;
               for ADC inputs with current source enabled (option -c)
```

```
               a temperature in Celcius is now calculated assuming they have such NTCs
connected.
  -r <cnt>   : Number of times to convert ADC input or inputs (default: 1).
  -t <us>    : Microseconds between ADC conversions (default: 200).
  -C         : Send GBT-SCA connect (HDLC control).
  -R         : Send GBT-SCA reset (HDLC control).
  -D         : Disable GBT-SCA ADC after operation (default: leave enabled)
  -E         : Do *not* enable GBT-SCA ADC at start of operation, assume it already
is.
  -X         : Use continuous-mode DMA for upload (default: single-shot).
  -Z         : Do not receive and display the GBT-SCA replies.
```

6.7.5 fscadac

The **fscadac** tool sets a GBT-SCA's DAC output channels, one at a time, or all four at the same time. In addition it provides an option to sweep through the DAC values for one or all channels within a configurable time period.

See help text below for the list of options.

Help text of **fscadac***:*

```
fscadac version 23060100
Tool to set and/or read back GBT-SCA DAC outputs,
all at the same time or a single one (option -i).
In addition it can do a sweep through the DAC range
for one or all DAC outputs (option -s)
within a configurable time period (option -t).
Usage:
 fscadac [-h|V] [-d <devnr>] [-e <elink>] [-G <lnk>] [-g <group>] [-p <path>]
         [-i <index>] [-s] [-t <ms>] [-C] [-R] [-D] [-E] [-Z] [-H] [<value>]
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device to use (default: 0).
  -e <elink> : E-link number (hex) or use -G/g/p options.
  -G <lnk>   : GBT-link number (default: 0).
  -g <group> : Group number (default matches GBT EC 'group' = 7).
  -p <path>  : E-path number (default matches GBT EC 'path' = 7).
  -i <index> : DAC index (0=DAC_A,1=DAC_B,2=DAC_C,3=DAC_D) to use (default: all).
  -s         : Sweep DAC value for the given DAC output(s).
  -S         : As -s, but send all DAC commands in one DMA operation,
               using delay packets instead of usleep().
  -t <ms>    : Sweep time from DAC value 0 to 255, in milliseconds,
               when option -s given (default: 1000).
  -C         : Send GBT-SCA connect (HDLC control).
  -R         : Send GBT-SCA reset (HDLC control).
  -Z         : Do not receive and display the GBT-SCA replies.
  -H         : Do not use HDLC delay packet, even if firmware supports it,
               use zero-byte packets (in combination with -S).
  -D         : Disable GBT-SCA DAC after operation (default: leave enabled)
```

```
  -E         : Do *not* enable GBT-SCA DAC at start of operation, assume it already
is.
 <value>     : DAC value (decimal or hex, 0x.. or x..) to set.
```

6.7.6 fscai2c

The **fscai2c** tool provides low-level access to I2C-devices connected to GBT-SCA I2C channels, with control over register address size (1 or 2 bytes), register content size and 7-bit or 10-bit addressing.

See help text below for the list of options.

Help text of **fscai2c***:*

```
fscai2c version 22101100
Tool to read or write from an I2C device register
on any I2C port of a GBT-SCA chip connected to any FLX-device E-link
(the latter given by options -G/g/p or option -e)
Usage:
 fscai2c [-h|V] [-d<devnr>] [-e<elink>] [-G<lnk>] [-g<group>] [-p<path>]
         [-f<freq>] -C<ichan> -I<iaddr> [-t] [-a|A<addr>] [-r<nbytes>]
         [-D] [-E] [<value-to-write>]
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device to use (default: 0).
  -e <elink> : E-link number (hex) or use -G/g/p options.
  -G <lnk>   : GBT-link number (default: 0).
  -g <group> : Group number (default matches GBT EC 'group' = 7).
  -p <path>  : E-path number (default matches GBT EC 'path' = 7).
  -C <ichan> : GBT-SCA I2C channel number.
  -f <freq>  : I2C bus frequency, in KHz (100,200,400 or 1000, default: 400).
  -I <iaddr> : I2C device address (decimal or '0x..' for hexadecimal).
  -t         : Use 10-bit I2C addressing mode.
  -a|A <addr>: I2C register address ('a':1-byte, 'A':2-byte).
               (decimal or '0x..' for hexadecimal).
  -r <bytes> : Register content number of bytes (default: 1; max 16).
  -D         : Disable GBT-SCA I2C port after operation (default: leave enabled)
  -E         : Do *not* enable GBT-SCA I2C port at start of operation,
               assume it already is.
 <value-to-write>: hexadecimal value to write, the number of nibbles determining
                   how many bytes to write.
=> Examples:
Read 2-byte register 6 from a device with I2C address 5 on GBT-SCA I2C channel 4
connected to the EC channel of GBT #3:
  fscai2c -G3 -C4 -I5 -a6 -r2
Write 0x1234 to 2-byte register 6 from I2C device address 5 on GBT-SCA I2C channel 4
connected to the EC channel of GBT #3:
  fscai2c -G3 -C4 -I5 -a6 1234
```

6.7.7 fscads24

The **fscads24** tool serves to demonstrate more-or-less strict timing capabilities, as required for the 1-Wire protocol, that FELIX is capable of, operating through a GBT-SCA GPIO device through a GBT link's EC-channel or other 2-bit HDLC E-link.

The **fscads24** tool reads out the unique 64-bit ID from a device from the 1-Wire DS2400-family, connected to a GPIO pin of a GBT-SCA.

Example of **fscads24** output, reading from a DS2411 connected to GPIO pin 3 of a GBT-SCA connected to the EC-channel of FELIX device #0 GBT link #3, including sending an initial 'connect' command:

```
$ fscads24 -C -G3 -i3
Opened FLX-device 0, firmw FLX712-GBT-4chan-2008271931-GIT:rm-4.10/544
GBT-SCA connect
ID:
01 63 41 a0 17 00 00 1a
Replies received: 288
```

See help text below for the list of options.

Help text of **fscads24***:*

```
fscads24 version 23060100
Tool to read out the 64-bit ID from a 1-Wire DS24xx chip
using a GBT-SCA GPIO line.
Usage:
 fscads24 [-h|V] [-d <devnr>] [-e <elink>] [-G <lnk>] [-g <group>] [-p <path>]
          [-r <cnt>] [-C] [-D] [-E] [-R] [-H] [-X] [-Z] -i <pin>
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device to use (default: 0).
  -e <elink> : E-link number (hex) or use -G/g/p options.
  -G <lnk>   : GBT-link number (default: 0).
  -g <group> : Group number (default matches GBT EC 'group' = 7).
  -i <pin>   : Use GPIO line <pin> for the 1-Wire protocol ([0..31]).
  -p <path>  : E-path number (default matches GBT EC 'path' = 7).
  -r <cnt>   : Number of times to read the ID (default: 1).
  -C         : Send GBT-SCA connect (HDLC control).
  -R         : Send GBT-SCA reset (HDLC control).
  -D         : Disable GBT-SCA GPIO after operation (default: leave enabled)
  -E         : Do *not* enable GBT-GPIO at start of operation, assume it already is.
  -H         : Do not use HDLC delay packet, even if firmware supports it,
               use zero-byte packets.
  -X         : Use continuous-mode DMA for upload (default: single-shot).
  -Z         : Do not receive and display the GBT-SCA replies.
```

6.7.8 fscajtag

The **fscajtag** tool is used to program a 'bit' file into a Xilinx FPGA connected to the JTAG port of a GBT-SCA.

Without a file name it reads and displays the connected FPGA's ID and status register contents (optionally the status bits are individually listed and named).

(Note that a version of this tool is available in the ATLAS DCS software for GBT-SCAs, interfacing to the E-link to the GBT-SCA via *felix-star* and *netio*).

See help text below for the list of options.

Help text of **fscajtag***:*

```
fscajtag version 19061800
Tool to program a bit-file into a Xilinx 7-Series FPGA connected to the JTAG port
of a GBT-SCA, connected to any FLX-device GBT (2-bit HDLC) E-link.
The FPGA may be part of a JTAG chain containing multiple devices;
the tool has options to take this into account.
If a bit-file name is not provided the ID-code and Status register
of the FPGA are read out and displayed.
Usage:
 fscajtag [-h|V] [-D] [-d <devnr>] [-e <elink>] [-G <lnk> [-g <group> -p <path>]]
          [-c] [-R <rate>] [-r] [-s]
          [-x <devs> -X <ibits> -y <devs> -Y <ibits> -z <instr>]
          [<filename>]
  -h         : Show this help text.
  -V         : Show version.
  -c         : Use continuous-mode FromHost DMA (default: single-shots).
  -d <devnr> : FLX-device to use (default: 0).
  -D         : Enable debug mode: display all GBT-SCA replies.
  -e <elink> : E-link number (hex) or use -G/g/p options.
  -G <lnk>   : GBT-link number.
  -g <group> : Group number (default: 7=EC).
  -p <path>  : E-path number (default: 7=EC).
  -R <rate>  : JTAG clock rate, in MHz, 1,2,4,5,10 or 20 (default: 20).
  -r         : Do NOT receive and process/display GBT-SCA replies.
  -s         : Display FPGA Configuration register bits.
  -x <devs>  : Number of devices preceding the FPGA in the JTAG chain.
  -X <ibits> : Total number of preceding BYPASS instruction bits, or
               (with option -z) the number of instruction bits per device.
  -y <devs>  : Number of devices trailing the FPGA in the JTAG chain.
  -Y <ibits> : Total number of trailing BYPASS instruction bits, or
               (with option -z) the number of instruction bits per device.
  -z <instr> : The BYPASS instruction value for each of the preceding
               and trailing devices (when unequal to only '1'-bits).
 <filename>  : Name of .bit file containing the FPGA configuration.
```

6.7.9 fxvcserver

The **fxvcserver** tool connects to a selected FELIX device E-link connected to a GBT-SCA and listens for an XVC protocol connection from a Vivado Hardware Manager in order to relay data to and from a Xilinx FPGA connected to the JTAG port of the GBT-SCA device, allowing to run the usual firmware programming and debugging operations from a Vivado session, on the user's remote front-end electronics.

(Note that a version of this tool is available in the ATLAS DCS software for GBT-SCAs, interfacing to the E-link to the GBT-SCA via *felix-star* and *netio*).

See help text below for the list of options.

Help text of **fxvcserver***:*

```
fxvcserver v20082600
Relays Xilinx XVC protocol JTAG bit streams to and from the JTAG port of a GBT-SCA,
through its connection to a FELIX system.
Usage:
 fxvcserver [-h|V] [-v] [-d <devnr>] [[-e <elink>] | [-G <lnk> [-g <group> -p <path>]]
            [-P <portnr>] [-R <rate>]
  -h         : Show this help text.
  -V         : Show version.
  -v         : Be verbose (for debugging only).
  -d <devnr> : FLX-device to use (default: 0).
  -e <elink> : E-link number (hex) or use -G/g/p options.
  -G <lnk>   : GBT-link number.
  -g <group> : Group number (default: 7=EC).
  -p <path>  : E-path number (default: 7=EC).
  -P <portnr>: IP port number to listen on (default: 2542).
  -R <rate>  : JTAG clock rate, in MHz, 1,2,4,5,10 or 20 (default: 10).

In Vivado's Hardware Manager in the TCL Console type "connect_hw_server"
(if necessary), followed by "open_hw_target -xvc_url <address>:<portnr>"
to connect to a Xilinx FPGA connected to the GBT-SCA JTAG port,
with <address> and <portnr> the IP address and port number
of the FELIX host running this fxvcserver.
```

6.7.10 fscareply

The **fscareply** tool takes a sequence of byte values (typically copied from some hex dump of data received from a FELIX device), and parses them as a GBT-SCA reply or request frame and displays information about the various items in the frame.

See help text below for the list of options plus some examples.

Help text of **fscareply***:*

```
fscareply version 22062100
Tool to parse and display a sequence of bytes as a GBT-SCA reply frame,
```

```
optionally as a request (command) frame (option -c)
or as a sequence representing an IC frame (option -I).
It will indicate the CTRL byte, transaction ID, channel, error byte,
data size and data word, if any.
Note that the length (LEN) word in a GBT-SCA reply has little meaning,
the actual number of bytes in the message defines the size of it.
In addition the type of error, if any, will be indicated,
and whether the frame contains an incorrect CRC.
Usage:
 fscareply [-h|V] [-c] [-I] [-r] [<byte0>] [<byte1>] ...
  -h         : Show this help text.
  -V         : Show version.
  -c         : Interpret the sequence as a GBT-SCA (or IC) *request* frame.
  -I         : Interpret the sequence as an *IC* frame.
  -r         : Reverse the byte order before interpretation.
 <byteX>     : Byte X of the reply frame to be parsed (provide it as hex numbers)

Examples:
> fscareply 00 0e 7d 00 00 03 00 fe e6 fb
CTRL=0E(r=0,s=7) TRID=125 Chan=CONF ERR=00 LEN=3 data=0xFE00

> fscareply 00 ec 28 14 00 06 00 00 f3 0b fd 9b
CTRL=EC(r=7,s=6) TRID=40 Chan=ADC ERR=00 LEN=6 data=0x00000BF3

Bytes provided in reversed order:
> fscareply -r 5c 09 06 00 13 36 ec 00
0 ec 36 13 0 6 9 5c : CTRL=EC(r=7,s=6) TRID=54 Chan=JTAG ERR=00 LEN=6

Indication of a CRC error (could be in CRC itself or in the data bytes),
(here CRC has been changed from 095C to 195C):
> fscareply 00 ec 36 13 00 06 19 5c
CTRL=EC(r=7,s=6) TRID=54 Chan=JTAG ERR=00 LEN=6  ###CRC=095C
```

6.8 Tools for lpGBT Control and Monitoring Channels

6.8.1 flpgbtio

The **flpgbtio** tool is used to read and write an lpGBT's GPIO lines, either individually or all 16 in one operation. Also the direction, pull-up/down and drive strength registers can be configured. In addition it can generate a configurable number of output pulses with a configurable width.

See help text below for the list of options.

Help text of **flpgbtio***:*

```
flpgbtio version 22040600
Tool to write and/or read the lpGBT GPIO bits and direction register,
as well as other GPIO-related registers (pull-up/down, drive strength)
Usage:
 flpgbtio [-h|V] [-D<dma>] [-d<devnr>] -G<link> -I<i2c> [-0|1] [-e]
```

```
         [-i<bit>] [-o<dir>] [-E<ena>] [-U<up>] [-S<s>] [-X] [-Z] [<value>]
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device number (default: 0).
  -D <dma>   : FLX-device DMA controller for receiving (default: 0).
  -0         : If lpGBT, assume v0 (default: auto-detect).
  -1         : If lpGBT, assume v1 (default: auto-detect).
  -e         : Use the EC-channel (default: IC-channel).
  -G <link>  : lpGBT link number.
  -I <i2c>   : lpGBT I2C address (hex).
  -i <bit>   : Read or write GPIO bit number <bit> (default: all).
               NB: if a single I/O pin is written to, its direction bit
                   is set to output (independent of option -o).
  -o <dir>   : Set GPIO direction register to value <dir> (16-bit hex).
  -E <ena>   : Set GPIO pull-up/down enable register to value <ena> (16-bit hex).
  -U <up>    : Set GPIO pull-up/down register to value <up> (16-bit hex).
  -S <s>     : Set GPIO drive strength register to value <s> (16-bit hex).
  -r <rep>   : Generate <rep> pulses up/down or down/up before setting
               the (single) output to the requested value (for debug/demo purposes).
  -t <us>    : Width of the '-r' pulses, in microseconds [2..1000] (default: 2).
  -X         : Debug mode: display bytes of received frames.
  -Z         : Do NOT receive and process/display replies.
 <value>     : Value to write (0 or 1 for a single GPIO bit,
               or up to 0xFFFF otherwise, hex);
               if no value is provided a read operation is performed.
```

6.8.2 flpgbti2c

The **flpgbti2c** tool provides low-level access to I2C-devices connected to an lpGBT I2C Master, with control over register address size (1 or 2 bytes), and register content size (up to 4 bytes).

See help text below for the list of options.

Help text of **flpgbti2c***:*

```
flpgbti2c version 22101100
Tool to read or write from and to an I2C slave device (register)
through an lpGBT I2C Master connected to an FLX-device link.
NB: suitable for lpGBT v1 only, for the time being.
Usage:
 flpgbti2c [-h|V] [-d<devnr>] -G<link> -I<i2c> [-f<freq>]
           -m<master> -i<iaddr> [-a|A<addr>] [-r<nbytes>] [-X] [-Z]
           [<value-to-write>]
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device to use (default: 0).
  -G <link>  : lpGBT-link number.
  -I <i2c>   : lpGBT I2C address (hex).
  -m <master>: I2C Master index (0,1 or 2; default:0).
  -f <freq>  : I2C bus frequency, in KHz
```

```
               (100,200,400 or 1000, default: 400).
  -i <iaddr> : I2C device address (hex).
  -a|A <addr>: I2C register address ('a':1-byte, 'A':2-byte).
               (decimal or '0x..' for hexadecimal).
  -L         : In case of 2-byte address (-A) LSByte first (default: MSB first)
  -r <nbytes>: Number of register/data bytes,
               for now limited to 4 (default: 1).
  -X         : Debug mode: display bytes of received frames;
               also: continue, even when nothing is received
               (e.g. in combination with option -Z).
  -Z         : Do NOT receive and process/display replies.
 <value-to-write>: value to write (decimal or '0x..' for hexadecimal).

=> Examples (with lpGBT I2C address 0x70):
Read 2-byte register at address 6 (1-byte address)
from I2C device with I2C address 0x30 using lpGBT I2C Master 1:
  flpgbti2c -G3 -I70 -m1 -i30 -a6 -r2

Read 1-byte register at address 6 (2-byte address)
from I2C device with I2C address 0x30 using lpGBT I2C Master 1:
  flpgbti2c -G3 -I70 -m1 -i30 -A6 -r1

Write 0x1234 to 4-byte register at address 6 (2-byte address)
to I2C device with I2C address 0x30 using lpGBT I2C Master 1:
  flpgbti2c -G3 -I70 -m1 -i30 -A6 -r4 0x1234
```

6.8.3 flpgbtds24

This tool serves to demonstrate more-or-less strict timing capabilities, as required for the 1-Wire protocol, that FELIX is capable of, operating on an lpGBT GPIO device through the lpGBT link's EC-channel.

(For more details about the workings of the tool see fscads24).

See help text below for the list of options.

Help text of **flpgbds24***:*

```
flpgbtds24 version 22020400
Tool to read out the 64-bit ID from a 1-Wire DS24xx chip connected to an lpGBT GPIO.
Usage:
 flpgbtds24 [-h|V] [-d<devnr>] [-D<dma>] -G<link> -I<i2c> [-1] [-e] -i<pin> [-r<cnt>]
[-Z]
  -h         : Show this help text.
  -V         : Show version.
  -d <devnr> : FLX-device number (default: 0).
  -D <dma>   : FLX-device DMA controller for receiving (default: 0).
  -1         : Assume lpGBTv1 (default: lpGBTv0) ###OBSOLETE
  -e         : Use the EC-channel (default: IC-channel).
  -G <link>  : lpGBT link number.
```

```
  -I <i2c>   : lpGBT I2C address (hex).
  -i <pin>   : Use GPIO bit <pin> for the 1-Wire protocol ([0..15]).
  -r <cnt>   : Number of times to read the ID (default: 1).
  -Z         : Do not receive and display the GBT-SCA replies.
```

0 :!table: 6

7. Felix-star

7.1 Introduction

`felix-star` is the name given to the suite of applications responsible for data transfers to and from the FELIX host. In the upstream "to-host" direction, `felix-tohost` publishes data coming from the FELIX card to clients according to a publish/subscribe pattern with E-link granularity. In the downstream "from-host" direction, `felix-toflx` receives messages for the enabled E-links over the network and transfers them to the FELIX card for onward transmission to front-end electronics. Over the network, E-links are identified by a 64-bit number called Felix-ID (fid) that contains both the local e-link number as well as a detector and connector identifiers.

The matching between active E-links and network addresses is provided by `felix-bus`. The `felix-bus` mapping is described by in hierarchical structure of json files.

The directory used for felix-bus needs to be accessible by all peers i.e. it has to reside on a network drive in case the communication happens between different hosts.

Communication with `felix-star` over the network is managed by the `netio-next` library, based on libfabric and capable of exploiting RDMA technology. The network library supports data coalescence to reduce the I/O overhead. Messages are accumulated in network buffers called "netio pages" that are sent either when an occupancy watermark is crossed, or a timeout expires. To minimise the latency it is possible to explicitly flush pages or avoid buffering altogether selecting the zero-copy mode. The zero-copy mode allows to send a message from a pre-defined memory location over the RDMA network without any copy. The main use case of zero-copy mode is the transfer of big messages (> O(10) kB) from the FELIX ToHost DMA buffer onto the network.

Client applications, such as Data Handler or OPCUA-SCA server, should not interface directly with `netio-next` but are instead required to use the `felix-client-thread` C++ interface. This API automatically reads the connection parameters (e.g. connection mode, number and size of netio pages) directly from `felix-bus` to minimise manual configuration. The use of the API is described at:

https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/felix-interface/master/headers.html

7.2 Architecture

The `felix-star` architecture is based on different independent processes responsible for supplying the functions described above, as illustrated in Figure 7.1 . Each device (i.e. PCIe endpoint) of an FLX card supports up to 5 ToHost DMA buffers, 1 FromHost DMA buffer and, for ITk flavours, an extra Trickle Configuration DMA buffer.

One `felix-tohost` instance can read multiple ToHost DMA buffers (also belonging to different devices). By default, one reader thread is spawned for each DMA buffer, but a command line option allows to use an arbitrary number of thread per DMA buffer to ensure performance scalability.

One `felix-toflx` instance can write onto multiple FromHost DMA buffers. For each FromHost DMA

buffer a single-threaded network receiver is created. Trickle configuration DMA buffers are managed by `felix-toflx`, started with a dedicated option. One `felix-toflx` process can either serve FromHost buffers or Trickle buffers.

`felix-register` allows a remote client to read or edit the registers of one or more FLX device. For this purpose, the client API offers dedicated methods to send so-called commands and receive replies.

All the applications described above use the same logging and monitoring systems. Logging is printed on the stdout/stderr. Monitoring information is collected in JSON format and is printed UNIX FIFO. From the FIFO can be read by any user application or by `felix-stats2prometheus` for publishing into the stand-alone monitoring system described in Section 7.4. In the next releases, both Prometheus and the TDAQ Information Service (IS) will be integrated as possible outlets in all felix-star applications.

Details on each application are presented in 7.3 Felix Star executables, while their orchestration is discussed in Section 8.

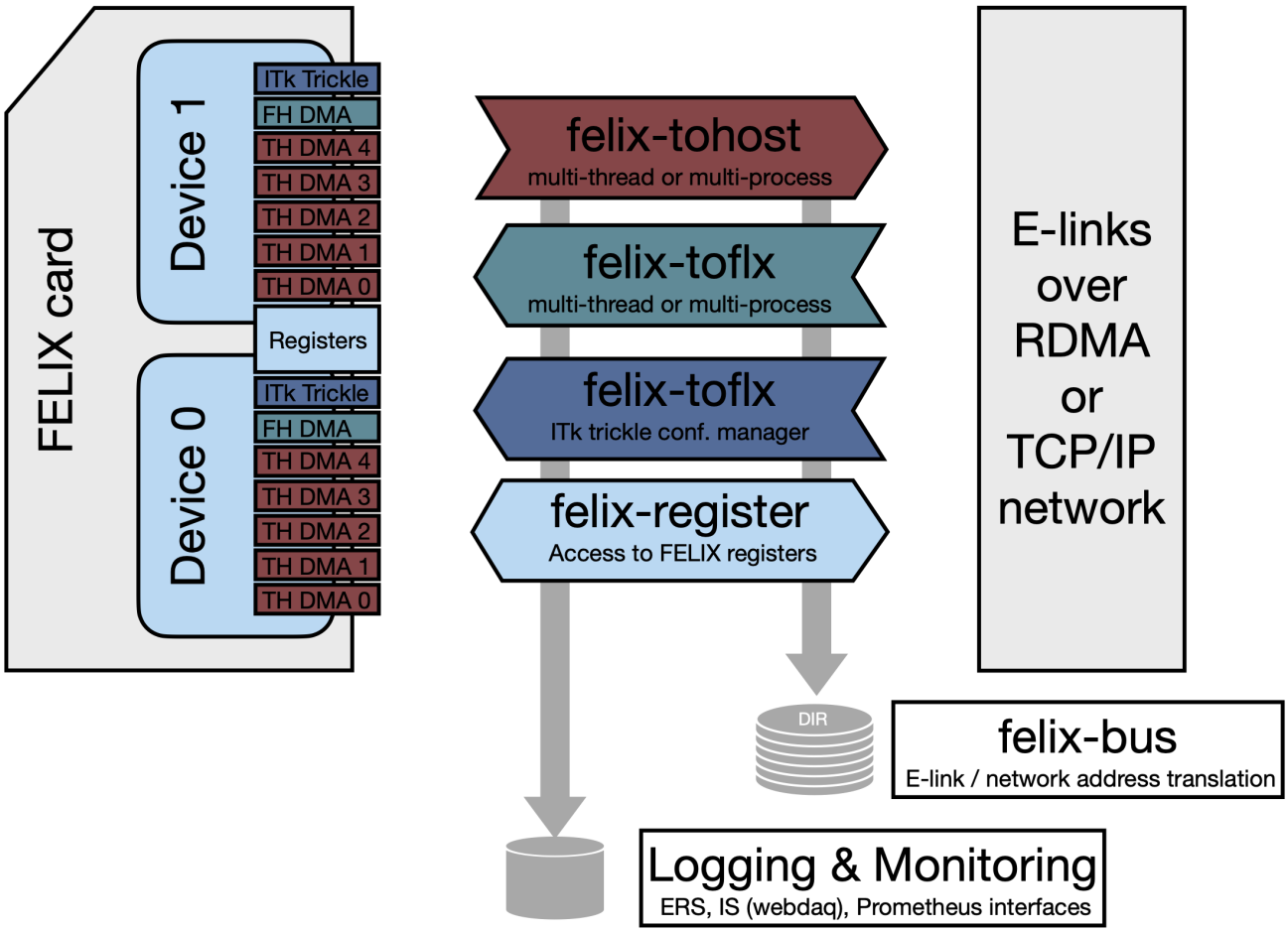

Figure 7.1 The architecture of the FELIX star application.

Netio-next TCP/IP backend emulates the RDMA stack and offers a poor performance. The adoption of netio3, currently under development, will address this issue.

7.3 Felix Star executables

7.3.1 felix-tohost

To run `felix-tohost` it is necessary to specify which devices and DMA buffers to read, the directory for the `felix-bus` and a network interface. To provide a unique identifier to the published E-links, the detector identifier (DID) and connector identifier (CID) can be specified. CIDs map to devices, that on FLX-712 and FLX-182 correspond to to the MTP connectors on the front panel. In the example below, `felix-tohost` is configured to read DMA buffers 0,1, and 2 of devices 0 and 1. The two devices are assigned CID 0x10 and 0x11, respectively. The network interface is `priv0` and the bus directory is created in the user's home.

```
felix-tohost --did 0x12 --cid 0x10 --cid 0x11 -d 0 -d 1 -D 0 -D 1 -D 2 --iface priv0
--bus-dir ~/bus
```

`felix-tohost` distinguishes three types of E-links, DAQ, TTC and DCS, for which different network configurations can be configured. E-link with encoding set to HDLC, as well as IC and EC e-links are identified as DCS E-links. Only the TTC2H E-link (0x600) is of type TTC: in this case TTC2H blocks produced by firmware are matched to network buffers to minimize latency. All remaining E-links are of type DAQ. More details on the network buffer settings are described in Section 7.7.

```
felix-tohost - FELIX central data acquisition application

    Usage:
      felix-tohost [options]

    General Options:
        --bus-dir=DIRECTORY         Write felix-bus information to this directory.
[default: ./bus]
        --bus-groupname=NAME        Use this groupname for the bus. [default: FELIX]
        --cid=N                     CID (Connector Id) to set in FID (Felix ID),
incompatible with --co. [default: device]
        --co=N                      CO (Connector Offset) to offset FID (Felix ID),
incompatible with --did and --cid.
        --did=N                     DID (Detector Id) to set in FID (Felix ID),
incompatible with --co. [default: 0]
    -d, --device=DEVICE             Use FLX device DEVICE. [default: 0]
        --error-out=FIFO            Write error information to a UNIX FIFO
        --free-cmem                 Free previously booked cmem segment by name-
<device>-<dma>
        --iface=IFACE               Send data to the interface. [calculated: use --ip
value]
    -i, --ip=IP                     Publish data on the ip address IP. [default:
libfabric:127.0.0.1]
    -b, --netio-pagesize=SIZE       NetIO page size in Byte. [default: 65536]
    -B, --netio-pages=SIZE          Number of NetIO pages. [default: 256]
        --stats-out=FIFO            Write periodic statistics data to a UNIX FIFO
        --stats-period=ms           Period in milliseconds for statistics dumps.
```

```
[default: 1000]
    -?, --help                      Give this help list
    -v, --verbose                   Produce verbose output
    -V, --version                   Print program version
        --vid=N                     VID (Version Id) to set in FID (Felix ID),
incompatible with --co. [default: 1]
    -w, --netio-watermark=SIZE      NetIO watermark in Byte. [default: 57344]

    ToHost Options:
        --buffered                  Enable buffered mode (FULL-mode only)
    -c, --cmem=SIZE                 CMEM buffer size in MB. [default: 1024]
        --daq-tcp                   Use TCP/IP fo DAQ traffic.
        --dcs-iface=IFACE           Use TCP/IP for DCS on this network interface.
[calculated: use libfabric on main iface]
        --dcs-pages=SIZE            DCS Number of NetIO pages. [default: 512]
        --dcs-pagesize=SIZE         DCS NetIO page size in Byte. [default: 1024]
        --dcs-rate-limit=LIMIT      Drop DCS messages when elink message is above the
given rate in KHz. [default: 0]
        --dcs-size-limit=LIMIT      Drop DCS messages when message size is 0 B or
above this threshold in Byte. [default: 20]
        --dcs-timeout=SIZE          DCS NetIO timeout in ms. [default: 1]
        --dcs-unbuffered            DCS unbuffered mode.
        --dcs-watermark=SIZE        DCS NetIO watermark in Byte. [default: 972]
    -D, --dma=ID                    Use DMA descriptor ID. [default: 0]
        --elink-offset=N            Offset for elinks as they are output and published
from felix-star (compatibility option). [default: 0]
    -l, --l1id-check=FORMAT         Check L1ID sequence. Formats: 1=TTC2H only,
2=LATOME, 3=FMEMU, 4=FELIG, 5=NSWVMM, 6=NSWTP.
                                    Slow for unbuffered DAQ. [default: 0]
        --max-chunk-size-buffered=LIMIT     Maximum chunk size supported for buffered
traffic. Larger chunks are truncated. [calculated: MAX_BUF_CHUNK_SIZE]
    -p, --port=PORT                 Publish data on port PORT. [calculated: 53100 +
10*device + dma]
    -P, --poll-period=us            Polling instead of interrupt-driven readout with
the given poll period in microseconds [default: 0]
    -R, --priority                  Enables round-robin scheduling for main thread
(priority value 2).
    -s, --dcsport=PORT              Publish DCS data on port PORT. [calculated: 53500
+ 10*device + dma]
        --ttc-netio-pages=SIZE      TTC Number of NetIO pages. [default: 512]
        --ttc-netio-pagesize=SIZE   TTC NetIO page size in Byte. [default: 1536]
        --ttc-netio-timeout=SIZE    TTC NetIO timeout in ms. Not necessary as TTC2H
buffers are flushed at end-of-block. [default: 0]
        --ttc-netio-watermark=SIZE  TTC NetIO watermark in Byte. [default: 1248]
    -t, --ttcport=PORT              Publish TTC2H data on port PORT. [calculated:
53300 + 10*device + dma]
    -T, --netio-timeout=SIZE        NetIO timeout in ms. [default: 2]
        --warn-bcr-period           Print warning message when non-periodic BCR is
detected.
```

```
Report bugs to <https://its.cern.ch/jira/projects/FLXUSERS>.
```

it can happen that the bus directory <bus-dir>/<bus-groupname> is owned by another user and cannot be overwritten. To overcome this situation choose a different group name.

on the client side network parameters are retrieved automatically from the felix bus.

7.3.2 felix-toflx

felix-toflx advertises the enabled links on the bus and listens for data. Support for E-link broadcast is supported from software release >= 5.0.1. Received data is encoded in the FromHost data format and written in the FromHost DMA buffer or Trickle DMA buffer. Network options passed to felix-toflx are advertised on the felix bus and are picked up by client applications.

```
felix-toflx - FELIX central data acquisition application

    Usage:
      felix-toflx [options]

    General Options:
        --bus-dir=DIRECTORY         Write felix-bus information to this directory.
[default: ./bus]
        --bus-groupname=NAME        Use this groupname for the bus. [default: FELIX]
        --cid=N                     CID (Connector Id) to set in FID (Felix ID),
incompatible with --co. [default: device]
        --co=N                      CO (Connector Offset) to offset FID (Felix ID),
incompatible with --did and --cid.
        --did=N                     DID (Detector Id) to set in FID (Felix ID),
incompatible with --co. [default: 0]
    -d, --device=DEVICE             Use FLX device DEVICE. [default: 0]
        --error-out=FIFO            Write error information to a UNIX FIFO
        --free-cmem                 Free previously booked cmem segment by name-
<device>-<dma>
        --iface=IFACE               Send data to the interface. [calculated: use --ip
value]
    -i, --ip=IP                     Publish data on the ip address IP. [default:
libfabric:127.0.0.1]
    -b, --netio-pagesize=SIZE       NetIO page size in Byte. [default: 65536]
    -B, --netio-pages=SIZE          Number of NetIO pages. [default: 256]
        --stats-out=FIFO            Write periodic statistics data to a UNIX FIFO
        --stats-period=ms           Period in milliseconds for statistics dumps.
[default: 1000]
    -?, --help                      Give this help list
    -v, --verbose                   Produce verbose output
    -V, --version                   Print program version
        --vid=N                     VID (Version Id) to set in FID (Felix ID),
```

```
incompatible with --co. [default: 1]
    -w, --netio-watermark=SIZE      NetIO watermark in Byte. [default: 57344]

    ToFlx Options:
        --netio-buffersize=SIZE     Obsolete, use --netio-pagesize. NetIO receive
buffer size in byte; maximum size for a single message. [default: netio-pagesize]
        --netio-buffers=SIZE        Obsolete, use --netio-pages. Number of NetIO
receive buffers. [default: netio-pages]
    -c, --cmem=SIZE                 CMEM buffer size in MB. [default: 20]
    -D, --dma=ID                    Use DMA descriptor ID. [calculated: highest dma]
    -m, --cdma                      Use circular DMA buffer instead of one-shot
    -p, --port=PORT                 Send data to port PORT. [calculated: 53200 +
10*device + dma]
    -r, --rawtcp                    Use raw tcp not libfabric (can be used with
--unbuffered)
        --stats-stdout              Prints stats to stdout
    -u, --unbuffered                Use unbuffered mode

Report bugs to <https://its.cern.ch/jira/projects/FLXUSERS>.
```

7.3.3 felix-register

`felix-register` can handle commands (get/set and others) via a command E-link and supply responses via a reply E-link. Both can be accessed easily using the send_cmd function in the `felix-client`. Felix-register also monitors a number of vital registers and publishes their content on a monitoring E-link. Monitored quantities include the FPGA temperature and the alignment status of links. `felix-register` can handle any number of devices in one host.

A single instance of `felix-register` can serve multiple PCI-E endpoints and cards passing as argument multiple triples of the kind `<device> <did> <cid>` e.g. `0 0x12 0x10 1 0x12 0x11 2 0x12 0x13 3 0x12 0x14`.

```
Reading registermap: "/builds/atlas-tdaq-felix/felix-distribution/x86_64-el9-gcc13-
opt/felix-star/share/felix-star/regmap5-symbol.yaml"
felix-register - Receive control data for registers and reply.

    Usage:
      felix-register [options] <local_ip_or_interface> (<device> <did> <cid>)...

    Options:
      -h --help                         Show this screen.
      --version                         Show version.
      --cmd-port=<N>                    Set port number for commands to the devices
[default: 53402 + (10 * first-device)]
      --pub-port=<N>                    Set port number for publications from the
devices [default: 53403 + (10 * first-device)]
      --mon-port=<N>                    Set port number for monitoring of the devices
[default: 53404 + (10 * first-device)]
      --mon-interval=<seconds>          Set monitoring interval [default: 1]
```

```
      --log-level=<loglevel>           Specify level of logging (trace, debug, info,
notice, warning, error, fatal) [default: info]
      --verbose                        Show verbose output
      --verbose-bus                    Show bus information
      --bus-dir=<bus-directory>        Set bus directory [default: bus]
      --bus-groupname=<groupname>      Set groupname for bus to use [default: FELIX]
      --error-out=<fifo>               Write error information to the error FIFO
[default: error]
      --regmap=<hex_version>           Register map version (simulator only).
[default: 0x0400]
      --no-cmd-channel                 Do not instantiate a Cmd channel, for unit
tests only
      --no-reply                       Do not reply, for unit tests only
      --no-mon-channel                 Do not start MON channel
      --no-route                       Do not route device-1 requests to device-0 for
non-existing endpoint-1

    Arguments:
      <local_ip_or_interface>          Local IP or interface
      <device>                         Use FLX device
      <did>                            DID (Detector Id) to use in FID (Felix ID)
      <cid>                            CID (Connector Id) to use in FID (Felix ID)

    Notes:
      <device> <did> <cid> triplets should be declared in endpoint-0, endpoint-1
order.
      Requests for non-existing endpoint-1 will be routed to endpoint-0: device 1 ->
device 0, device 3 -> device 2
```

7.3.4 felix-fid

The `felix-fid` utility translates and decodes FIDs to DID, CID, E-Link, TID and SIDs and vice-versa. Full details of the structure of an FID can be found in:

https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/docs/LinkMappingSpecification.pdf

```
Usage: felix-fid [OPTION...] <felix id>
  or:  felix-fid [OPTION...]
           <detector id> <connector id> <link id> <transport id> [<stream
           id>]
FELIX FID conversion (accepts hex 0x, binary 0b and octal 0)

  -c, --cid                  print Connector ID
  -d, --did                  print Detector ID
  -e, --elink                print E-link
  -h, --hex                  print in hex format
  -i, --vid                  print Version ID
  -l, --lid                  print Link ID
  -o, --co                   print Connector Offset
  -s, --sid                  print Stream ID
```

```
  -t, --tid                  print Transport ID
  -u, --update=VERSION       Update Version ID. Default: 1
  -v, --verbose              Produce verbose output
  -?, --help                 Give this help list
      --usage                Give a short usage message
  -V, --version              Print program version

Mandatory or optional arguments to long options are also mandatory or optional
for any corresponding short options.

Report bugs to <https://its.cern.ch/jira/projects/FLXUSERS>.
```

```
felix-fid --verbose 0x123456789abc3def
```

for example shows:

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|FID                                                            |
|                                          0x123456789abc3def|
|                                         1311768467463749103|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|CO                         |GID                                |
|                  0x1234567|                       0x89abc3def|
|                   19088743|                        36955766255|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|VID|DID    |CID            |LID          |TID          |SID    |
|0x1|   0x23|         0x4567|       0x226a|       0x3c3d|   0xef|
|  1|     35|          17767|         8810|        15421|    239|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|VID|DID    |CID            |V Elink            |D Proto|SID    |
|0x1|   0x23|         0x4567|1           0x09abc|0  0x3d|   0xef|
|  1|     35|          17767|1             39612|0    61|    239|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

FID:   (64 bits) - Felix ID
CO:    (28 bits) - Connector Offset
GID:   (36 bits) - Generic ID
VID:   (4 bits)  - Version ID
DID:   (8 bits)  - Detector ID
CID:   (16 bits) - Connector ID
LID:   (14 bits) - Link ID
TID:   (14 bits) - Transport ID
SID:   (8 bits)  - Stream ID
V:     (1 bit)   - Virtual
Elink: (19 bits) - As used in the FELIX card
D:     (1 bit)   - Direction [0=tohost, 1=toflx]
Proto: (7 bits)  - Protocol
```

The field E-link in the last row of boxes shows the E-link number as reported in elinkconfig.

7.4 Monitoring

Monitoring information produced by felix-star applications can be visualised on the Grafana dashboard provided with all software releases starting from 5.1.0. A description of the quantities displayed can be found clicking on the info button of the corresponding panel. `felix-monitor` is distributed as a tarball containing a docker-compose files to start Prometheus and Grafana containers. The content of the tarball is shown below:

```
felix-monitor/
├──── compose.yml
├──── grafana
│          ├──── dashboards
│          │         ├──── all.yml
│          │         └──── felix-local.json
│          └──── datasources
│              └──── datasource.yml
└──── prometheus
     └──── prometheus.yml
```

Before starting the monitoring edit `prometheus.yml` adding the name of the nodes to monitor. For example, to set `pc-tbed-felix-14.cern.ch` as data source replace the default `localhost:8000` as shown below.

```
static_configs:
  - targets:
    - pc-tbed-felix-14.cern.ch:8000
```

To start felix-monitor on an AlmaLinux node, run from within the felix-monitor folder:

```
sudo podman-compose up -d
```

The Grafana dashboard will be reachable entering in the URL bar of the browser`localhost:3000`. To stop the monitoring run:

```
sudo podman-compose down -v
```

The monitoring information is written on disk in `felix-monitor/prom_data`.

7.5 Enabling streams

In order to make use of the streaming feature, support must be enabled for the links in question using feconf or elinkconfig. This will cause `felix-star` to publish the full set of 256 FIDs corresponding to the available streams on that link. The stream ID is the last byte in the FID, which will therefore have the value 0-255 (rather than the default value of 0).

It is then possible to subscribe to a particular FID as normal by additionally setting the last byte to that StreamID, 0-255. The front-end corresponding to the links in question should then make sure to put the correct stream ID in the first byte of every packet sent over the link. FELIX will then guarantee to route the packet correctly to client applications with the appropriate subscription.

7.6 Quick start

Start felix-tohost

```
mkdir </path/to/bus>
mkfifo /tmp/stats
felix-tohost -d <device> -D <DMA buffer> --bus-dir ~/bus --iface <network interface>
--stats-out /tmp/stats
```

The fids corresponding to the e-links published by felix-tohost are printed on screen. It is possible to see data going through `felix-tohost` by running `cat < stats` To subscribe to one or more e-links you can use the client application

```
felix-test-swrod --bus-dir=<path to bus directory> --fid=<fid1> --fid=<fid2>
--fid=... <local-ip>
```

The remote IP and port for the requested fids are resolved using the content of the bus folder (that neeeds to be readable by `felix-test-swrod`). `felix-test-swrod` will print message rate and counters, it has options to dump the data messages on screen or in a file.

7.7 Network Parameters

Both `felix-tohost` and `felix-toflx` have command-line parameters that configure the number of network buffer (`netio-pages`) and their size (`netio-pagesize`). The number of pages shall not exceed 1024.

Client applications read these settings from the felix-bus and configure themselves accordingly. In case unbuffered mode is selected in felix-tohost, `netio-pages` and `netio-pagesize` determine the network buffer allocation on the receiving side. When unbuffered mode is used in felix-toflx `netio-pages` and `netio-pagesize` determine the allocation of receiving buffers: in this case `netio-pagesize` shall be larger or equal than the largest expected message.

Two additional parameters are relevant on the sender side (be it felix-tohost or a client application) if data coalescence is used: the page watermark and timeout. The watermark (`-w` command line

option in `felix-tohost`) sets an occupancy threshold that causes the buffer to be sent once crossed. The timeout acts similarly using the lifetime of a partially filled buffer rather than its occupancy.

:!table: 7

8. Orchestration of FELIX applications

8.1 Supervisor

As described in the previous section multiple applications have to be run concurrently to run the FELIX DAQ system. To manage all these applications the Supervisord orchestrator is used. Supervisord is a third-party application and is included in the FELIX distribution. The command `supervisord` is used to start the orchestrator while `supervisorctl` is used to control it. The utlity `felix-multivisor` allows to control multiple hosts at once. A configuration file that described what Supervisor has to do is required.

The following subsections describe how Supervisor can be used in the FELIX use case; complete documentation can be found at http://supervisord.org/

8.1.1 Configuration file

Supervisor is steered by a configuration file that describes all the applications to be managed. A simple config file that runs two instances of `felix-tohost` is shown in Listing [sv_example].

The first three blocks are needed to configure Supervisor:

- `inet_http_server` configures Supervisor to run an HTTP interface that can be used to control it. This configuration is safe only for a closed network. For general use prefer a UNIX socket that is protected by read/access rights.
- `supervisord` contains settings about Supervisor itself, including the user who runs it and other parameters.
- `supervisorctl` specifies parameters for the `supervisortctl` command line utility. In particular username and password for the HTTP interface are reported to avoid re-typing them.
- `rpcinterface:supervisor` is necessary and can be left at its default value

The two instances of `felix-tohost` are included as `[program:tohost-d0]` and `[program:tohost-d1]`. Any other application can be included in the same fashion. In detail, `startretries=3` tells Supervisor to restart `felix-tohost` if it fails at startup for at most three times. The meaning of startup is specified by `startsecs=5`. If instead `felix-tohost` crashes after a while, thus returning an error code, Supervisor is instructed to restart it automatically by `autorestart=unexpected` (the restarts after crashes in the running stage are not counted). Finally `stopsignal=INT` tells Supervisor that `felix-tohost` shall be terminated using a SIGINT signal.

For convenience, multiple programs can be grouped together as shown in the `group:40_daq` block. Here the `autostart=true` attribute makes the programs part of the group start as soon as Supervisor is launched.

```
[inet_http_server]
port=*:9001
username=user
password=password
```

```
[supervisord]
logfile=/logs/supervisord.log           ; supervisord log file
logfile_maxbytes=50MB                   ; maximum size of logfile before rotation
logfile_backups=10                      ; number of backed up logfiles
loglevel=error                          ; info, debug, warn, trace
pidfile=/tmp/supervisord.pid        ; pidfile location
nodaemon=false                          ; run supervisord as a daemon
user=%(ENV_USER)s                       ; default user

[supervisorctl]
serverurl=http://localhost:9001
username=user
password=password

[rpcinterface:supervisor]
supervisor.rpcinterface_factory = supervisor.rpcinterface:make_main_rpcinterface

[program:tohost-d0]
command=felix-tohost -d 0 --free-cmem --bus-dir /tmp/bus --ttc-unbuffered --did 0x00
--cid 0x0000 --iface priv0
autorestart=unexpected
startsecs=5
startretries=3
stdout_logfile=/logs/tohost-d0.out
stderr_logfile=/logs/tohost-d0.err
autostart=false
stopsignal=INT

[program:tohost-d1]
command=felix-tohost -d 1 --free-cmem --bus-dir /tmp/bus --ttc-unbuffered --did 0x00
--cid 0x0001 -iface priv0
autorestart=unexpected
startsecs=5
startretries=3
stdout_logfile=/logs/tohost-d1.out
stderr_logfile=/logs/tohost-d1.err
autostart=false
stopsignal=INT

[group:40_daq]
programs=tohost-d0, tohost-d1
autostart=true
```

if are not running in a closed network use `[unix_http_server]` instead of `[inet_http_server]`.

8.1.2 Control

Supervisor is started with

```
supervisord -c <config file>
```

All groups and programs whose `autostart` attribute is set to true will be launched at this point. Otherwise programs can be started with

```
supervisorctl -c <config file> start <group>:<program>
```

To start the whole group of programs use the asterisk as wildcard symbol. To stop replace `start` with `stop`.

The status of applications can be printed on terminal with

```
supervisorctl -c <config file> status
```

or visualized pointing the web browser to the HTTP interface (http://localhost:9090 in the example above). The web interface presents buttons to perform various actions.

To shutdown Supervisor and all associated applications enter

```
supervisorctl -c <config file> shutdown
```

8.1.3 Startup sequence

Any kind of program can be managed by Supervisor, including set up commands like `flx-init` and `feconf`. Yet, no execution sequence can be defined in the configuration file. To implement a startup sequence an Event Listener need to be included. A Supervisor Event Listener can be any application registered as such that communicates with Supervisor via stdin and stdout. The Event Listener is informed about the change of status of managed applications and can issue commands via Supervisor's HTTP Remote Procedure Call (RPC) interface. The Event Listener written for the use-case of ATLAS is called felix-supervisor can be found at

https://gitlab.cern.ch/atlas-tdaq-felix/felix-starter/-/blob/master/supervisor/felix-supervisor

`felix-supervisor` parses the config file and expects groups named accoring to the scheme `<NN_name_blocking>`. The `NN` is a number that determined the order, while the keyword blocking implies that all applications belonging to the group have to have terminated before the next group can be started. `felix-supervisor` is equipped with an interface for the ATLAS ERS system.

`felix-supervisor` is included in the config file with

```
[eventlistener:fsupervisor]
command=felix-supervisor -c /det/tdaq/felix-operation/supervisord/supervisord_pc-tbed-
felix-00.cern.ch.conf
events=PROCESS_STATE
buffer_size=25
```

```
stderr_logfile=/logs/felix-operation/pc-tbed-felix-00.cern.ch-fsupervisor.log
autorestart=false
autostart=true
```

8.1.4 Generation of many config files

Supervisor config files can be long and contain a lot of hardcoded information. For this reason a template-based generation system has been put in place. First, FELIX hosts are described in a file such as

```
# host_name                 direction           device#     mode         clock
interface   detector id     connector id    elinkconfig              DMA number
# fully qualified           tohost/toflx/both   (0-3)       gbt/full     T/L      name
(8 bit)         (16 bit)        filename                 use -1 for Phase-I
pc-tbed-felix-07.cern.ch    both                 0           gbt         T
priv0       0x07            0x1200         LS_MMG.elc                -1
pc-tbed-felix-07.cern.ch    both                 1           gbt         T
priv0       0x07            0x1201         LS_MMG.elc                -1
pc-tbed-felix-07.cern.ch    both                 2           gbt         L
priv0       0x07            0x1a00         LS_MMG.elc                -1
pc-tbed-felix-07.cern.ch    both                 3           gbt         L
priv0       0x07            0x1b00         LS_MMG.elc                -1
pc-tbed-felix-04.cern.ch    tohost               0           full        T
priv0       0x04            0x0000         pc-tbed-felix-04-d0.jelc  -1
pc-tbed-felix-04.cern.ch    tohost               1           full        T
priv0       0x04            0x0001         pc-tbed-felix-04-d1.jelc  -1
pc-tbed-felix-06.cern.ch    none                 0           full        L
priv0       0x06            0x0000         pc-tbed-felix-06-d0.jelc  -1
pc-tbed-felix-06.cern.ch    none                 1           full        L
priv0       0x06            0x0001         pc-tbed-felix-06-d1.jelc  -1
pc-tbed-felix-00.cern.ch    tohost               0           full        L
priv0       0x00            0x0000         pc-tbed-felix-00-d0.jelc  -1
pc-tbed-felix-00.cern.ch    tohost               1           full        L
priv0       0x00            0x0001         pc-tbed-felix-00-d1.jelc  -1
```

that is used by the generator `felix-supervisord-generate`.

```
Generate supervisord configuration files.

Usage:
    felix-supervisord-generate [options] <felix-config-file>

Options:
    -t, --template <supervisord-template>   Supervisord template [default:
supervisord.conf.jinja]
    -d, --destination DIRECTORY             Directory for the generates SUpervisor
config file [default: /det/tdaq/felix-operation/supervisord/]
    -c, --config-dir DIRECTORY              Directory with the configuration files
```

```
(elinkconfig) [default: /det/tdaq/felix-operation/config]
    -b, --bus-dir DIRECTORY                 Directory for the bus [default:
/det/tdaq/felix-operation/bus]
    -m, --tmp-dir DIRECTORY                 Absolute directory for the temporary files
of supervisord [default: /dev/shm]
    -g, --bus-groupname GROUP_NAME          Group name to use for the bus [default:
FELIX]
    -l, --log-dir DIRECTORY                 Directory for the logfiles [default:
/logs/felix-operation]
    -u, --user USER                         User of supervisor HTTP interface
[default: username]
    -p, --password PWD                      Password of supervisor HTTP interface
[default: password]
    -q, --tdaq RELEASE                      TDAQ release for felix2atlas and felix-
supervisor [default: tdaq-11-02-00].
    --tdaq-env                              Generates TDAQ environment for felix-
supervisor
    --error-out FIFO                        FIFO for errors [default: /dev/shm/felix-
errors.json]
    --stats-out FIFO                        FIFO for stats [default: /dev/shm/felix-
statistics.json]
    --error-port PORT                       PORT for errors [default: 53401]
    --stats-port PORT                       PORT for stats [default: 53400]
    --dcs-watermark SIZE                    DCS NetIO watermark in Bytes [default:
972]
    --dcs-unbuffered                        Use unbuffered DCS socket
    -v, --verbose                           Verbose output

Arguments:
    <felix-config-file>         FELIX configuration file
```

the generator produces one config file per FELIX host filling up the template. The template in use by the FELIX team can be found at

https://gitlab.cern.ch/atlas-tdaq-felix/felix-starter/-/blob/master/etc/supervisord.conf.jinja

8.2 Management of multiple FELIX hosts

8.2.1 Autostart via Systemd

To make Supervisor start automatically after a reboot a SystemD unit can be defined. An example of unit can be found below.

```
[Unit]
Description=FELIX Supervisor
Requires=tdaq-driver.service
After=tdaq-driver.service

[Service]
```

```
Type=forking
User=<your favourite user>
LimitMEMLOCK=infinity
LimitNOFILE=65536
ExecStartPre=/bin/echo "Starting Supervisord for FELIX DAQ"
ExecStart=/bin/bash -c "export HOSTNAME=`hostname` && source <path to felix
release>/setup.sh && supervisord -c <config file>"
ExecStartPost=/bin/echo "Supervisord for FELIX DAQ started"
ExecStop=/bin/bash -c "supervisorctl -c <config file> shutdown"
RemainAfterExit=true
Restart=on-failure
StartLimitBurst=3
StartLimitInterval=30s

[Install]
WantedBy=multi-user.target
```

8.2.2 Control multiple hosts

If config files are collected in a single folder and named according to the scheme `supervisor_<fully qualified hostname>.conf` (e.g. `supervisord_pc-tbed-felix-00.cern.ch.conf`) it is possible to use the `felix-multivisor` utility toc control all nodes simultaneously.

---

usage: felix-multivisor [-h] [-p PROC] [-s SVDIR] [-r] action host [host ...]

positional arguments: action Action, e.g. start / stop / status... host Target hostname(s)

optional arguments: -h, --help show this help message and exit -p PROC, --process PROC Target application or group -s SVDIR, --sdir SVDIR Location of SV config files -r, --serial Process hosts serially ---

8.3 Useful scripts

8.3.1 felix-get-ip

The `felix-get-ip` command retrieves the IP for a given interface name. This interface name is retrieved using the `felix-get-config-value` command. The IP is used to start `felix-tohost`.

```
Return (first or fastest) ip4 address for interface or the list of available
interfaces.

Usage:
    felix-get-ip [options] [<interface_name>]

Options:
    --fastest                   Return ip for fastest network
    -v, --verbose               Verbose output
```

```
Arguments:
    <interface_name>            Name of the ethernet interface, see ifconfig
```

8.3.2 felix-get-mode

The `felix-get-mode` command returns the mode for a particular device. It is used to lookup values in the configuration using `felix-get-config-value`.

```
Return mode for device.

Usage:
    felix-get-mode [options] <device>

Options:
    -v, --verbose               Verbose output

Arguments:
    <device>                    Device number
```

:!table: 8

9. Felix-star client applications

9.1 Felix-Client-Thread API

User applications that wish to communicate with `felix-star` shall use the `felix-client-thread` API. The `felix-client-thread` API exposes a small number of methods that allow to subscribe to (and unsubscribe from) E-links, send data to E-links and get notified when data is received, a connection succeeds or fails. The `felix-client-thread` API is only concerned with data and E-links/FIDs: network parameters are read from the felix-bus, connections are managed internally and re-connections are automatically attempted if connection is lost.

The API and its use are documented at

https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/felix-interface/master/

9.2 Data Handler / SW ROD OKS configuration

Data Handler uses the `felix-client-thread` to interface with FELIX. Configuring Data Hanlder to use a given FELIX software release is done with the following OKS snippet.

```
<obj class="TagMapping" id="ftag">
  <attr name="Value" type="string" val="x86_64-centos7-gcc11-opt/lib"/>
  <attr name="SW_Tag" type="enum" val="gcc11-opt"/>
  <attr name="Platform" type="enum" val="x86_64-centos7"/>
</obj>


<obj class="SW_ExternalPackage" id="felix-release">
<attr name="Name" type="string" val="FELIX Software Release"/>
<attr name="InstallationPath" type="string" val="/det/tdaq/felix/felix-05-00-01-stand-
alone"/>
<rel name="SharedLibraries">
 <ref class="TagMapping" id="ftag"/>
</rel>
```

The data network interface and bus directory are set as follows.

```
<obj class="SwRodFelixInput" id="FelixClient">
 <attr name="Type" type="string" val="FelixInput"/>
 <attr name="DataNetwork" type="string" val="vlan413"/>
 <attr name="FelixBusGroupName" type="string" val="FELIX"/>
 <attr name="FelixBusDirectory" type="string" val="/det/tdaq/felix-operation/bus"/>
 <attr name="FelixBusTimeout" type="u32" val="1000"/>
</obj>
```

More information about Data Handler / SW ROD can be found at

https://gitlab.cern.ch/atlas-tdaq-software/swrod

:!table: 9 :numbering:

10. FAQ, Troubleshooting and User Resources

This section is aimed at collecting useful information for front-end developers to aid the design and implementation of front-end firmware/hardware for interaction with FELIX. Useful tips based on experience so far will also be presented, in a section that will grow over time as more feedback is received.

10.1 Frequently Asked Questions

1. *Is GBT wide mode supported?*

   Not currently, can be reviewed on request.

2. *Is GBT 8b/10b mode supported?*

   8b/10b encoded E-links within GBT frames are supported, but this is different from native GBT '8b/10b frame mode', which is not supported.

3. *Is the phase of the eight "utility" clocks fixed with respect to the E-link clocks?*

   Yes, there is a fixed relationship with the E-Link clocks. Note that the eight utility clocks have *worse* jitter than the E-link clocks.

4. *Can the GBT output a 40 MHz E-link clock, use that clock in 40 MHz DDR mode for the to-frontend link, but accept data on the uplink at 160 or 320 Mb/s? (Assuming the FE ASIC multiplies the 40 MHz to 80, 160 or 320 MHz.)*

   Yes, that is possible. Also the to-frontend link can receive at 80, 160 or 320 Mb/s.

5. *Is there a maximum packet length on the E-link in 8b/10b mode?*

   No.

6. *Are direct mode a.k.a. unencoded E-links supported in GBT mode?*

   Not by default in the Phase-I firmware, but this being integrated into Phase-II builds. Please contact the FELIX team for more information.

7. *Can FELIX support additional (or custom) link protocols as they are developed?*

   FELIX will not support additional (or custom) link protocols unless well motivated by a detector requirement. If you think you will need to introduce an additional protocol please contact the FELIX team before making any final implementation decision.

10.2 Troubleshooting

10.2.1 Known Issues with GBTx

- Links disconnected from any front-end source generate spurious data at random intervals. If using FELIX with a GBTx, it is strongly recommended that any links which are disconnected from the front-end be deactivated in elinkconfig. This will prevent spurious data causing confusion in front-end testing. The effect of spurious data from accidentally or temporarily disconnected E-links can be minimized by using the packet truncation options.

- The loading of the configuration from the e-fuses on power-on is not reliable. The GBTx must be explicitly configured on every power-on. For GBTx' used only as transmitters, a way to configure them via I2C must be provided.

10.2.2 IOMMU

Some users have experienced problems with DMA transfers on systems with an active IOMMU. If your FLX card is recognized correctly but the DMA transfers are not working, check the kernel logs. If you find DMAR errors on memory access, it could be that the kernel is running with intel-iommu enabled, and you will need to disable every memory mapping by hand. This can be done by executing the command:

```
dmesg | grep -i -e DMAR -e IOMMU
```

After disabling IOMMU, the card should work without any issue.

10.2.3 File Descriptor (FD) Limit

Depending on the setup of your operating system, it may be necessary to change the FD limit to avoid running out of descriptors and experiencing the following types of error:

For OPC-UA:

```
terminate called after throwing an instance of 'std::runtime_error' what(): could not connect
to endpoint 128.141.177.225:12351 Aborted (core dumped)
```

The current FD limit on can be seen by running:

```
ulimit -aH
```

and looking for the entry marked 'open files'.

The limit can be changed by modifying `/etc/security/limits.conf`. It is recommended to set the limit to the maximum available value, namely 65536.

In general, the recommended values on FELIX and Data Handler are

---

core file size (blocks, -c) 0 data seg size (kbytes, -d) unlimited scheduling priority (-e) 0 file size (blocks, -f) unlimited pending signals (-i) 186944 max locked memory (kbytes, -l) unlimited max memory size (kbytes, -m) unlimited open files (-n) 65536 pipe size (512 bytes, -p) 8 POSIX message queues (bytes, -q) 819200 real-time priority (-r) 0 stack size (kbytes, -s) 8192 cpu time (seconds, -t) unlimited max user processes (-u) 186944 virtual memory (kbytes, -v) unlimited file locks (-x) unlimited ---

10.2.4 Debugging Link Status

When using flx-info to check the status of your links, note that this only corresponds to alignment of incoming (Rx) links, for which it is possible to recover a clock. To check the status of links from FELIX to other electronics it is recommended to implement dedicated monitoring to check alignment there.

In the case of the FLX-712 card, flx-info does provide incoming and outgoing optical power measures, so this should be used to confirm whether a link problem is due to a problem with an optical fibre or light transmission from FELIX.

10.2.5 SMBus Access

SMBus access, as needed for the fflash tool, has been found to not work on Supermicro servers with an X11SPW-TF motherboard when using default factory configuration. However, Supermicro has provided custom firmware, which provides this support by means of "ipmitool raw" commands. This is firmware for the IPMI interface, i.e. not a BIOS, and has to be loaded via the IPMI network connection. The firmware is available here:

https://drive.google.com/open?id=1eVae35mJhdRZam3WlfW0YM2cpXpCz5yB

The file to be loaded is BETA_X11DP_Xilinx_Kintex_UltraScale_FPGA_982_20190628.bin. Follow this description to load it:

https://www.supermicro.com/manuals/other/IPMI_Users_Guide.pdf (chapter 2-99)

The following test with fflash has been done using the latest BIOS and after reloading the BETA firmware. Thanks to Henk Boterenbrood for performing the test.

The help information reported by fflash -h is as follows:

```
fflash version 21020800
Tool for loading a firmware image from one of the partitions
of the onboard flash memory of an FLX-712 into the card's FPGA,
issueing commands to the host system I2C bus to achieve this.
A subsequent hotplug procedure or machine reboot is required.

Usage: fflash [-h|V] [-q] -f<flashnr>
              [[-L|I] [-U|P -d<devslot>] [-S] [-b<busnr>] [-r<chan>] [-R<raddr>]
              [-s<saddr>] [-u<uaddr>] [-T<sec>]]
  -h          : Show this help text.
  -V          : Show version.
  -q          : Be quiet (only errors will be displayed).
  -f <flashnr>: Flash memory segment partition [0..3] selection (no default).
  -I          : Generate an INIT_B pulse on the FLX-card (to reset flash devices).
  -L          : Load firmware from the given flash partition into the card.

The following options are relevant in conjunction with the -L and/or -I option:
  -b <busnr>  : I2C bus number (default=0).
  -r <chan>   : Riser card I2C-switch channel number (default=0)
                (Select I2C-switch address using option -R).
  -R <raddr>  : Riser card I2C-switch I2C address (default 0x70).
  -s <saddr>  : I2C-switch I2C address (hex, default=0x77, expected range: 0x70-0x77).
                NB: 0x70 already taken by the riser card I2C-switch!
  -u <uaddr>  : Embedded microcontroller I2C address
                (hex, default=0x67, expected range: 0x60-0x67).
  -U          : Use USB I2C-dongle instead of system SMBus
```

```
                (requires scripts i2cset.py and i2cget.py installed in /opt/flx).
  -P          : Use 'ipmitool' to access system SMBus.
                !NB: use -d option to select 'device slot': 1 or 2.
  -T <sec>    : Set 'Prog-done' timeout [s] (default: 7)
  -d <devslot>: Device slot (1 or 2), only in combination with -P.
  -S          : Precede calls to i2cget/set or ipmitool with 'sudo'.
                (default: 'sudo' not used; applies to options -L|I|P|U).
Examples:
---------
Load flash memory image partition #2 into the card:
  fflash -f2 -L

Load flash memory image partition #2 into the card, using I2C-bus #1,
riser card I2C-switch channel #0, FLX-card I2C-switch address 0x75 and
FLX-card microcontroller I2C-address 0x65:
  fflash -f2 -L -b1 -r0 -s75 -u65

How to determine the I2C-switch and uC I2C addresses
(options -s and -u respectively):

Note 1: there is an I2C-bus number (option -b) to select as well,
  which is assumed to have the value '1' (following '-y') in the examples below.
Note 2: in the standard FELIX server there is an additional I2C-switch
  on the socalled riser card; its channel is selected using option -r;
  its I2C address (default 0x70) can be selected using option -R;
  it means that the 2 FLX-cards in such a server may have identical
  '-s' and '-u' addresses, i.e. most likely their defaults
  while the riser card setting is: 'top' position = -r 0, 'bottom' = -r 1.

'sudo i2cdetect -y 1' should show you an address in the range 0x70-0x77,
let's say 0x77; this is then the address to use in option -s;
subsequently run 'sudo i2cset -y 1 0x77 1' to set the I2C-switch
causing an additional address in the range 0x60-0x67 to appear
in the output of 'sudo i2cdetect -y 1', so run that command again;
this is the address to use in option -u.

On the FLX-712 dipswitch J14 configures the '-s' and '-u' addresses:
  switch 1-3 to set 3 LSBs of '-s', i.e. 0x70-0x77
  switch 4-6 to set 3 LSBs of '-u', i.e. 0x60-0x67
```

Example of the use of fflash to configure the FPGA from partition 2 of the FLASH memory (the program has to be run using sudo due to the use of "sudo ipmitool" commands):

```
fflash -f 2 -P -L -d 2 -u 63 -s 76
```

```
Parameters:
-f 2: partition 2
-P: use of ipmitool
-L: load firmware from FLASH memory
-d 2: card in slot 2 from riser
```

```
-u 63: i2c address of card is 0x63 (can be set with jumpers on the card)
-s 76: address i2c switch on riser is 0x76

Output program with these command line parameters for our machine:

=> Load firmware partition 2 (I2C via IPMI: switch=0xec, uC=0xc6):
sudo ipmitool raw 0x30 0x70 0xd5 27 1 2 236 0 1

sudo ipmitool raw 0x30 0x70 0xd5 27 1 2 198 0 17 0 16

sudo ipmitool raw 0x30 0x70 0xd5 27 1 2 198 0 16 56

sudo ipmitool raw 0x30 0x70 0xd5 27 1 2 198 0 17 48

sudo ipmitool raw 0x30 0x70 0xd5 27 1 2 198 0 18 0

sudo ipmitool raw 0x30 0x70 0xd5 27 1 2 198 1 12
Prog done (time: 1320 ms)
sudo ipmitool raw 0x30 0x70 0xd5 27 1 2 198 0 16 0

sudo ipmitool raw 0x30 0x70 0xd5 27 1 2 236 0 0

=> Pulse INIT_B (I2C via IPMI: switch=0xec, uC=0xc6):
sudo ipmitool raw 0x30 0x70 0xd5 27 1 2 236 0 1

sudo ipmitool raw 0x30 0x70 0xd5 27 1 2 198 0 19 4

sudo ipmitool raw 0x30 0x70 0xd5 27 1 2 198 0 18 4

sudo ipmitool raw 0x30 0x70 0xd5 27 1 2 198 0 19 0

sudo ipmitool raw 0x30 0x70 0xd5 27 1 2 198 0 19 4

sudo ipmitool raw 0x30 0x70 0xd5 27 1 2 198 0 18 0

sudo ipmitool raw 0x30 0x70 0xd5 27 1 2 236 0 0
```

An attempt to read from an incorrect address looks like this:

```
sudo ipmitool raw 0x30 0x70 0xd5 27 1 2 208 1 12
```

```
Unable to send RAW command (channel=0x0 netfn=0x30 lun=0x0 cmd=0x70 rsp=0x83): Unknown
(0x83)
```

Form of the ipmitool raw command:

```
ipmitool raw 0x30 0x70 0xd5 27 1 slot address readcount writedata
```

```
slot: device slot: 1 or 2
```

```
address: i2 address
readcount: number of bytes to read, if 0: no bytes to read
writedata: bytes to be written
```

NB: The ipmitool raw command requires the addresses to be shifted one bit to the left

10.2.6 Problems with CMEM allocation on boot

Note - if you have a network-booted system (e.g. managed by the ATLAS TDAQ Sysadmins in the context of the ATLAS testbed infrastructure) then you likely won't be able to perform the actions listed below. In this case it might be necessary for you to consider moving to a local boot. Please contact markus.joos@cern.ch for more advice.

If you have a local boot with RPM driver installation, and are having problems configuring CMEM to consistently allocate memory on boot please attempt the following:

1. Contact the FELIX team to confirm the required amount of memory for your setup. The typical recommendation is 4 GB per FELIX card.
    - Depending on your setup and use case, it may be necessary to consider adding additional RAM.
2. If necessary, amend the allocation by opening `/etc/init.d/drivers_flx` and changing `gfpbpa_size=` to the new amount.
3. If this doesn't help, try configuring `systemd` such that `drivers_flx_sd.service` runs as the first service. This can be achieved as follows:
    - Run `systemd-analyze plot drivers_flx_sd.service > p.svg; eog p.svg`. You will see something like what is shown in Figure 10.1 .
    - Open the file `/etc/systemd/system/drivers_flx_sd.service`
    - Look for the `[Unit]`section. Add the statement `BEFORE=` at the end of the `[Unit]` section and list all of the services that are in front of `drivers_flx_sd` (see example below).
    - Reboot the computer and check `systemd-analyze plot drivers_flx_sd.service > p.svg; eog p.svg` again. Keep adding services to the `BEFORE=` list until `drivers_flx_sd` is the first service, as shown in Figure 10.2
    - Try multiple reboots and confirm that this results in stable allocation.
4. If this still does not resolve the issue please contact markus.joos@cern.ch for additional support.

Example modification of [Unit] block:

```
[Unit]

Description=Start the drivers required by a TDAC PC or SBC
DefaultDependencies=no

Before=kmod-static-nodes.service systemd-udevd-kernel.socket dev-hugepages.mount
dev-mqueue.mount nss-user-lookup.target machine.slice dm-event.socket user.slice
```

```
slices.target
```

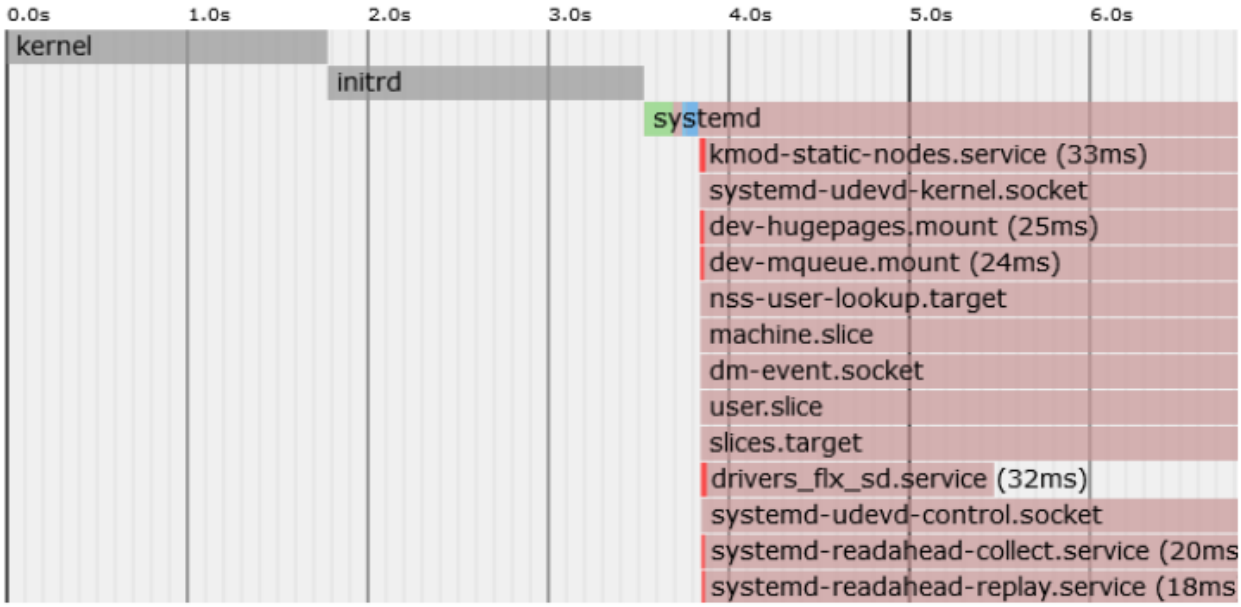

Figure 10.1 Example output of `systemd-analyse`.

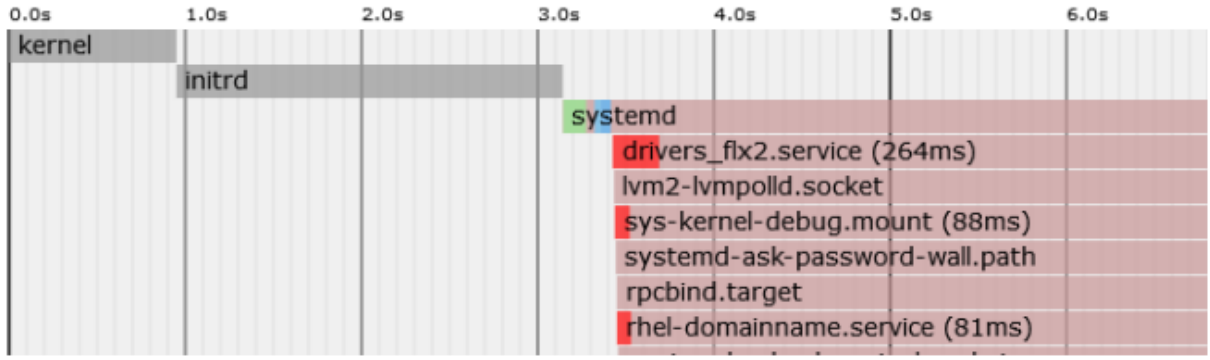

Figure 10.2 Example output of `systemd-analyse` *with drivers_flx first.*

10.3 Guide for System Designers

- For GBT-mode transmission to FELIX, use 8b/10b encoding on the E-links. Avoid the non-encoded fixed length or variable length formats, because no resynchronization is possible if bits are lost or repeated on the E-link. Comma symbols are used to align to *10-bit symbols* in the bit stream. They are considered idles and can be inserted in the data stream anywhere. Transmit "frequent" pairs of commas. This will minimize data loss when FELIX tries to resynchronize when the symbol boundary is lost due to a missed or repeated bit on the E-link. The out-of-band SoP (Start-of-Packet) and EoP (End-of-Packet) "K-Characters" are used to delimit packets in 8b/10b E-links.
- The E-link clock, input, and output data rates are independent. The only restriction is that within a GBT E-link group, all the clocks must have the same frequency, all the data inputs the same data rate and all the outputs the same data rate. However, groups can be setup independently from each other. Read the GBTx manual carefully to understand the GBTx group restrictions and bit order. Note, however, that a clock output is only available if its corresponding Tx is enabled. This means, for example, that a bank running with 320 Mb/s E-

links can supply only **two** (identical) clocks, but they can be 40, 80, 160 or 320 MHz.

- E-link "chunks" or packets are even multiples of bytes or 8b/10b symbols. If an odd number of bytes are received from the front end, FELIX will add an extra padding byte. In the to-front end direction, the length must be an even number of bytes.
- Synchronization of 8b/10b encoding requires *two* consecutive comma characters.
- In 8b/10b encoded E-links, FELIX can be asked to assert BUSY by sending BUSY-ON and BUSY-OFF symbols (i.e. out-of-band symbols that can be sent any time, even within data packets). This should be done only in exceptional cases or at start of run. It should not be the normal mode of protecting against buffer overflow. Instead, complex dead time should be defined to prevent most buffer overflows.
- The event data sent to FELIX are not expected to be ATLAS-standard event fragments. FELIX just transports the data to the Software ROD where detector specific software may transform the data as required and format it into ATLAS-standard event fragments for the ATLAS Read out system (ROS).
- The use of a CRC or the IP checksum is recommended to detect any transmission errors for E-links run over cables.
- In addition to sending all events to the SW ROD, FELIX can send all, or a sample of, events to other network end points for monitoring. Extra monitoring data may be included as packets separate from event data packets in the E-link data stream by using FELIX's stream IDs at the start of the packet.
- Even if there is no hit data associated with a Level-1 Accept, it will still be necessary to send a packet to the SW ROD that contains at least the L1ID and BCID. Without this is will be almost impossible to properly recover from error conditions that may arise.
- DCS information may be included as packets separate from event data packets in an E-link data stream by using FELIX's stream IDs at the start of the packet.
- Any 80 Mb/s E-link can be used to connect to a GBT-SCA ASIC. The E-link clock must be configured to use 40 MHz, i.e. the data is sent in DDR mode.
- The "EC" link can be used as an ordinary E-link at 80 Mb/s; its E-link clock may be either 40 or 80 MHz.
- TTC: FELIX can send TTC Level-1 Accept information on any E-link declared as a 'TTC' E-link. 'TTC' E-links can be 80, 160 or 320 Mb/s E-links, to transfer, respectively, 2, 4 or 8 TTC bits on every BC clock. The contents of the TTC word is defined by the FELIX configuration and can be chosen from the ten bits in Table 10.4 . Note: In all three cases, the E-link clock can be 40 MHz, i.e. the BC clock. The data is sent with FIXED latency.

Table 10.4 List of bits decoded from the TTC system that can be chosen to be sent on an E-link defined as a TTC E-link.

Brcst[7]	Brcst[6]	Brcst[5]	Brcst[4]	Brcst[3]	Brcst[2]	ECR	BCR	B-chan	L1A

- Network end-points, such as the SW ROD, can receive Level-1 Accept information (L1ID, BCID, Trigger Type, etc.) by subscribing to the Level-1 Info (virtual) E-link (also known as the TTC-to-host E-link). See 6.1.6.3 Configure the to-host Level-1 Accept info E-link (*TTC-to-Host* E-link).
- The FULL mode firmware has the ability to send XOFF / XON characters on 8b10b encoded

80Mb/s E-Links towards the FULL Mode FrontEnd. For details on the XOFF mechanism and also the FELIX BUSY system, see BusyXonSpecs. The duration and count that an XOFF was issued on a particular link can be measured by FELIX. This feature was added in firmware 4.10, for details see FLX-1171. The measurements can be read using the following commands:

```
#Read the XOFF duration and count for FULL mode channel 0, divide TOTAL_DURATION by
COUNT for an avarage duration.
flx-config XOFF_PEAK_DURATION00
flx-config XOFF_TOTAL_DURATION00
flx-config XOFF_COUNT00
```

10.4 FELIX Firmware Modules for Front-end Users

The FELIX team have produced a number of self-contained firmware modules which are intended for integration into front-end firmware both for testing and production purposes. These make it possible to test data transfer functionality from the output layer of the front-end firmware to FELIX and beyond, before integrating more of the front-end logic. Modules exist for both GBT and FULL mode use cases.

10.4.1 Downloading Firmware Source

A full description (including diagrams) of the modules discussed below, as well as the relevant firmware source, is available on the FELIX project distribution site:

https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/dist/examples/

The site contains multiple revisions, for compatibility with different FELIX firmware versions. GBT-compatible packages are labeled 'ElinkInterfaceSources' and FULL mode-compatible packages are labeled 'FullmodeInterfaceSources'. Please consult the documentation within the files for compatibility information.

10.4.2 GBT Test Modules

From a GBT perspective the modules provided depend on whether the GBT implementation is in an FPGA or with a GBTX chip. Common to both is a simple data generator module, which generates an incrementing counter and can be attached to the input port of the GBT module to provide a basic data source for link testing.

10.4.2.1 GBT-FPGA

For FPGA-based GBT a module will be provided to wrap and drive the GBT link in communication with FELIX (in both directions). All modules will be fully compatible with the official GBT-FPGA core[17].

10.4.2.2 GBTx

For GBTx chips all that is needed is to connect the provided data generator to a chip e-port, thus providing data on one E-link across the GBT.

10.4.3 FULL Mode Test Modules

10.4.3.1 Link Layer Tests

For FULL mode implementations the FELIX developers provide a link layer test package, making it possible to verify functionality at transceiver level (e.g. clock jitter stability, cleaning and configuration). Users should be able to integrate this into their front-end design for basic tests before implementing higher level link protocols.

10.4.3.2 Protocol Tests

Once the link layer is verified, users can integrate the 'stream controller' module, also provided by the FELIX developers, which manages the FULL mode link protocol and adds e.g. start and end of packet markers. This is recommended for use not just in testing but also final implementation. Alongside this module a simple data generator is also provided which can be used for testing data transfer across the link.

10.4.4 E-link Wrapper

The E-link Wrapper is a standalone FPGA module that implements the E-link interface. It instantiates the Elink2FIFO and FIFO2Elink components alongside a reset sequence logic. The Elink2FIFO and FIFO2Elink contain FELIX's Central Router modules, which manage the TX/RX datarate (80, 160 or 320 Mbps) and the line code that is being used (8b10b or HDLC). These components provide a simple FIFO interface to the user that can write the data-to-be-sent into the FIFO2Elink buffer, and read the received data from the Elink2FIFO buffer. The wrapper can therefore be used as an interface between an FPGA and FELIX, via a GBTx or a GBT-FPGA instantiation [17], or as a means of communication between two FPGAs. More information on how to use the component can be found on [18]. The source files themselves can be found in the repository which is included in the aforementioned user guide's References section.

10.5 External Software Resources and Tools

10.5.1 SCA eXtension — FPGA emulation of the SCA ASIC

A standard way to configure a front-end device is via the GBT Slow Control Adapter (GBT-SCA) ASIC [1]. The SCA ASIC implements several interfaces to communicate with other on-board devices, in order to configure them and monitor their status. The SCA eXtension (SCAX) FPGA firmware module [19] [20] emulates the SCA's communication protocol, in order to transparently provide access via FELIX to the registers of the FPGA logic in which it is instantiated. The use of SCAX enables DAQ and DCS to take advantage of the whole back-end OPC-UA software ecosystem to access FPGA registers. (See Appendix D.) SCAX also supports reading and writing FPGA memories and FIFOs. More information can be found in Appendix E.

10.5.2 IC-over-NetIO

The GBTx ASIC can be configured via a dedicated E-link, the IC E-link, provided that a bi-directional optical connection between that GBTx and FELIX exists. FELIX offers the ability to configure the GBTx via the low-level *fice* command, which accepts a file containing the values of the entire GBTx

address space, and forwards them to the ASIC accordingly. The tool may also write values into specific registers, or read the values from some of the GBTx' registers.

The IC-over-NetIO application [21] is based on *fice's* logic, with the main difference being that it allows the user to perform the same operations via the NetIO interface, i.e. while the FELIX software is running, something that the *fice* application cannot do, being a low-level tool.

At the time of writing the application is in its testing phase, with some minor issues remaining to be addressed.

:!table: 10 :numbering:

Appendix A: Setting up a TTC System for use with FELIX

This section is meant to help users of FELIX systems with the set-up of a TTC system. Both the new ALTI and legacy TTCvx/TTCvi systems are described below.

When connecting a TTC system to the FLX-712 card, note that the hardware is sensitive to signals down to -36 dBm and up to -3 dBm. Incoming optical power should be within this range.

A.1 The ALTI System

The ATLAS Local Trigger Interface (ALTI) is an upgrade to the former TTC system, and replaces the functionality of the previous LTPi, LTP, TTCVi and TTCvx. The ALTI board provides functionalities such as rate counters for all TTC signals, per-bcid counters, busy monitoring, a pattern generation, and as monitoring/synchronization of input signals and programable phase shift of output signals. For a full list of functionalities and detailed ALTI instructions please see:

https://twiki.cern.ch/twiki/bin/viewauth/Atlas/LevelOneCentralTriggerALTI

The forward signals are ones sent by the CTP to the sub-systems (BC, ORB, L1A, TTR, BGO, TTYP), while the backward signals are sent by the subsystems to the CTP (BUSY, calibration). The front panel of the ALTI also has six SFP connectors for -mode fibre optic transmitter/receiver modules. The first five SFPs are used for dual-transmitters, while the last one can be used for TTC signal monitoring.

To test the FELIX with the ALTI card, connect system as shown in Figure A.1 . The cable from the LEMO connector of the FELIX timing card is plugged into the lower left connector labelled as BUSY OUT. In the future, it will be possible to configure the BUSY connector on the ATLI for either a NIM or TTL signal via software. Until this software feature is available in the ALTI, a NIM to TTL adapter can be used since the FELIX sends a signal compatible with a TTL signal, while the ALTI accepts a NIM signal by default.

To send TTC signals (L1A for example) from the ALTI to the FELIX, connect top SFP connector (using a single LC connector), to the ST connector on the FELIX timing card as shown in Figure A.1 . An LC to ST adapter will be needed for this.

The bottom SFP connector with the orange connections is connected in loopback mode for ALTI debugging and can be ignored for the purposes of FELIX commissioning.

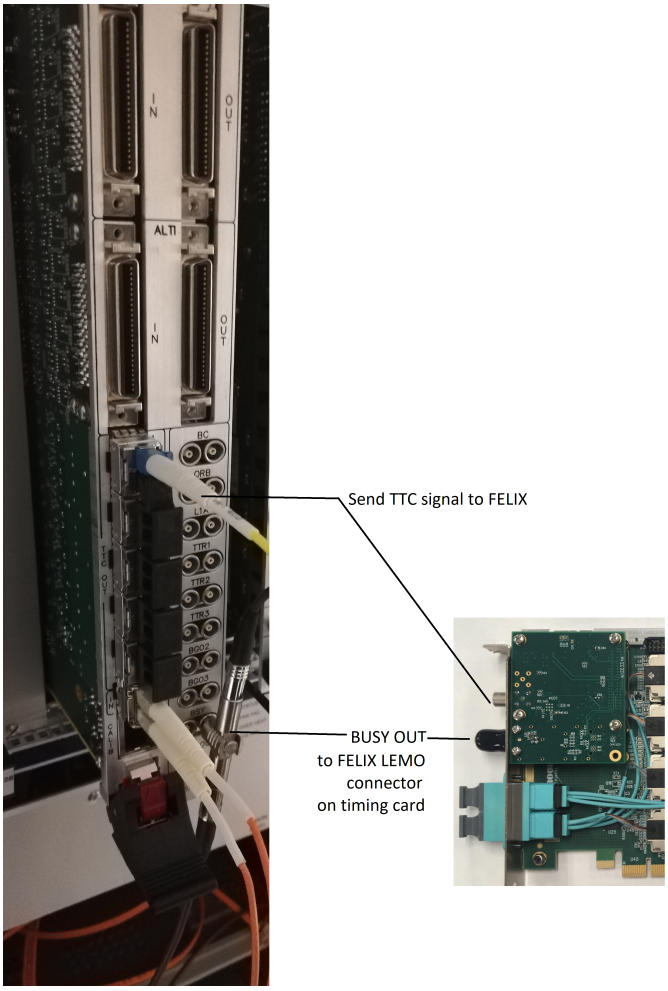

Figure A.1 Image of cabled ALTI

A.1.1 Software Setup

Once the hardware is setup log into the SBC hosting the crate and setup the TDAQ and ALTI software either locally, in testbed, or at Point 1 following instructions at:

https://twiki.cern.ch/twiki/bin/viewauth/Atlas/LevelOneCentralTriggerALTI#Software

Once the TDAQ and ALTI environment is setup, configure the correct slot for the ALTI using the command:

```
testAltiInitial -s 6 -B -b 0x07000000 -R -S -c
```

where -s 6 is the ALTI slot number, -B -b 0x07000000 is to change the base address to 0x07000000 and -R -S -c is to reset and check the ALTI board.

A.1.2 Sending TTC Signals with ALTI

First make sure the FELIX is set to TTC clock following instructions in [subsubsec:clocksel] and [subsubsec:clockRecovery]. After the software is setup you are ready to test sending L1A with the ATLI. You can use a configuration located here to send the L1A from the ATLI: /afs/cern.ch/atlas/project/tdaq/level1/ctp/l1ct-08-03-05/ALTI/data/cfg_1MHz.dat.

The file sends Level-1 Accepts (L1A) at a 1 MHz rate. You can either copy the file locally or read it directly from afs if you have access to it from your TTC crate. You can then start sending the L1A by

executing the command below from the SBC:

```
testAltiInitial -s 4 -R -S -C -f /afs/cern.ch/atlas/project/tdaq/level1/ctp/l1ct-08-03-05/ALTI/data/cfg_1MHz.dat
```

At this point the FELIX should forward the L1A to the front end, your front end should respond accordingly and send data through the FELIX, which you can monitor with FELIXcore.

If one would like to stop/start the pattern sending you can follow the commands below from the SBC, and selecting 7, followed by 3 (to enable L1A sending) or 4 (to disable L1A sending):

```
menuAltiModule
```

` 7 [PAT menu] Pattern generation memory`

` 3 enable pattern generation`

` 4 disable pattern generation`

If a different patter generation or a different frequency of pattern is required, it is possible to configure one following instructions on

https://twiki.cern.ch/twiki/bin/viewauth/Atlas/LevelOneCentralTriggerALTI#Scripts_to_generate_of_Pattern_G

A.1.3 Testing BUSY signal with ALTI

In order to test whether the ALTI is corresponding correctly to a BUSY from the FELIX, it is possible to force a BUSY in the FELIX using the commands below on the FELIX server:

```
flx-config set TTC_DEC_CTRL_BUSY_OUTPUT_INHIBIT=0 -d 2
```

```
flx-config set TTC_DEC_CTRL_MASTER_BUSY=1 -d 2
```

To remove the BUSY execute the commands below on the FELIX card:

```
flx-config set TTC_DEC_CTRL_BUSY_OUTPUT_INHIBIT=0 -d 2
```

```
flxcard $ ./flx-config set TTC_DEC_CTRL_MASTER_BUSY=0 -d 2
```

The ALTI should respond to the FELIX BUSY by stopping to send the L1A (or other generated) patterns. To test if the ALTI has stopped sending the patterns, the following can be executed on the SBC hosting the ALTI:

```
menuAltiModule
```

And select 11 CNT menu counters, and then select 1 to read counters. If selecting 1 to read counters several times does not make the counters go up, it means the ALTI has stopped sending the L1A (or other pattern), and thus correctly responded to the FELIX BUSY signal.

A.2 The TTCvi/TTCvx (A)

Figure A.2 shows the final cabling of TTCvi and TTCvx modules for a TTC setup with B-channel. The

A-channel carries the Level-1 Accept; the B-channel carries BCR and the other TTC commands. The TTCvi-TTCvx pair should have already been tuned. If not, see A.2.1 Tuning a TTC system below. Note: For a TTCex this may look different. A list of all the materials you will require to set up a TTC system is presented in Table A.5 .

Figure A.2 Image of cabled TTC system with B-channel connections

Table A.5 Materials needed to set up a TTC system

Item	Source	Remarks

VMEbus crate	Can be rented from the CERN Electronics Pool	Other crates may do as well
VMEbus master	We recommend a SBC from Concurrent Technologies (ATLAS standard). Support can be given for VP717, VP917 and VP-E24 (64 bit, compatible with TDAQ software release tdaq-05-05-00 and above).	TTCvi VMEbus card
Can be rented from the CERN Electronics Pool (but the Pool may be out of stock)	The TTCvi is no longer in production. Make sure the VME base address switches are set to match your software.	TTCvx VMEbus card or TTCex VMEbus card
TTCvx and TTCex can be rented from the CERN Electronics Pool (but the Pool may be out of stock)	The TTCvx/ex is no longer in production. The TTCvx has a LED driver, the TTCex has a laser driver	3 LEMO cables (1 or 0.5 ns)
		1 optical multi-mode (TTCvx) or single-mode (TTCex) fiber with ST connectors on both ends
	Max length of the fibre: TTCvx: 20 m; TTCex: 100 m	TTCoc & ATLAS (not clear who to ask; Maybe P. Farthouat) & TTC fan-out; needed if you have several FELIX
optical attenuator		Needed only for use with a TTCex without TTCoc. The optical attenuator has to be a single-mode attenuator of 3-20 dB and has to be connected directly to the TTCex output. The FTPDA-R155 should work with a TTCvx without attenuator. In case of a TTCex an attenuator of 3 dB is recommended for the FTPDA-R155. The FTPDA-R155 has a sensitivity of -31 dBm and saturates at +1 dBm.
If you need to tune the TTCvi-TTCvx pair, you need in addition: • 2 LEMO cables (5-10 ns) • 2 LEMO Y-adapters • 2 LEMO-BNC adapters • 2 50 Ohm terminators (Only required if your oscilloscope has no internal termination.)		

A.2.1 Tuning a TTC system

If your TTCvi-TTCvx pair has not been tuned, follow the instructions in this section. Cable the TTC system as shown in Figure A.3 . Note: for a TTCex this may look different. For more information please consult the section "Tuning procedure 2" of the TTCvi manual (http://www.cern.ch/TTC/TTCviSpec.pdf).

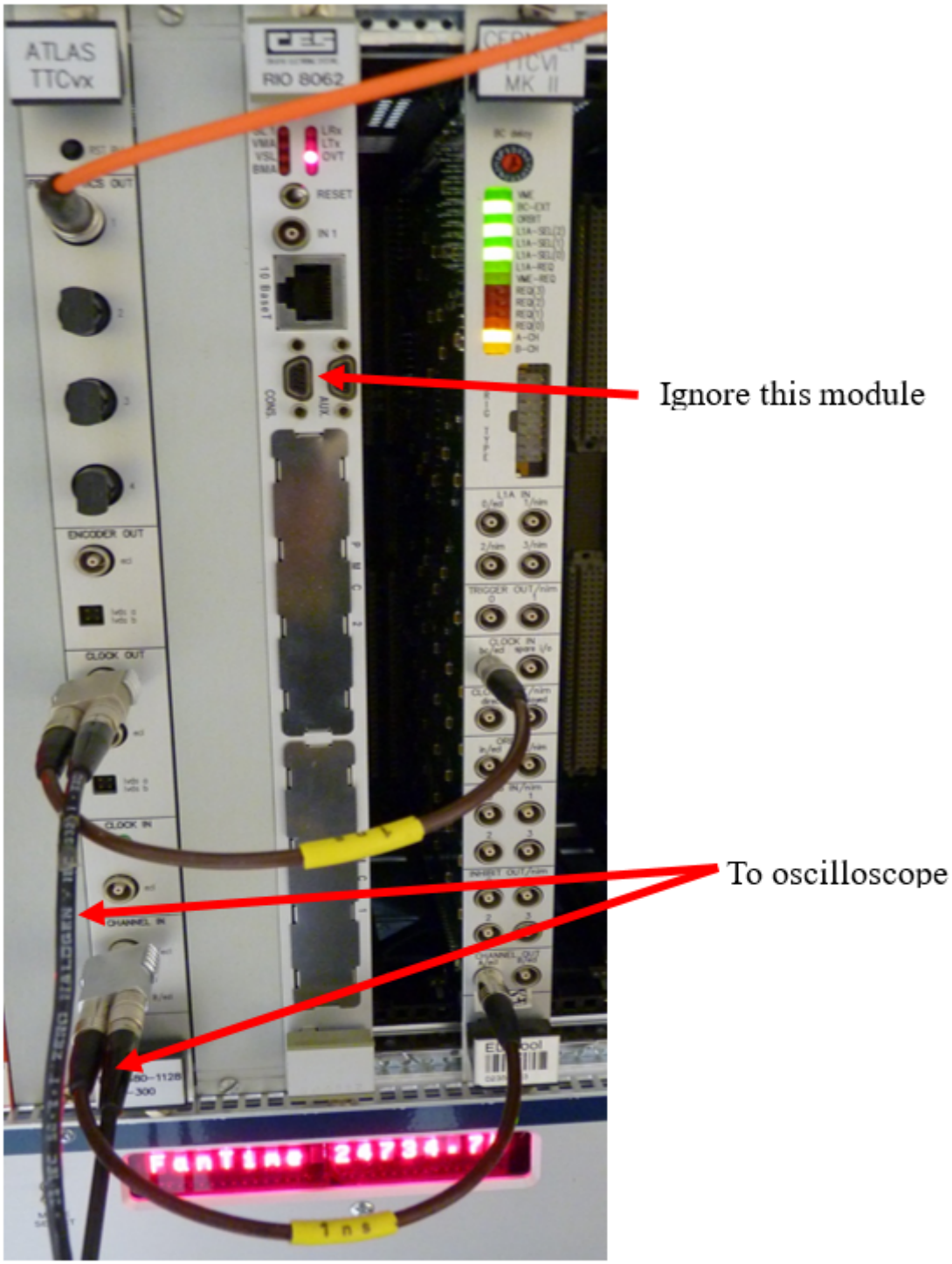

Figure A.3 Image of cabling for tuning a TTC system

Note: The question has come up if channel A and channel B are correctly cabled in the picture above. Here is a reply from the TTC expert (Sophie Baron):

A "good" configuration when channel B is not used is indeed to have it tied to "1". And it is right that having Channel B connected to OUTPUT B gives a static "1" on channel B. However, the termination

scheme at the TTCex inputs keeps as well unconnected channel inputs (both B and A) to "1" by default (it is negative ECL logic, and the Vin is at -2.08V by default). Therefore, both schemes could be used identically. One additional remark: of course, if you leave both A and B unconnected at the input of the TTCex, you will have both channels A and B to "1", and this is not good as the TTCrx needs to see two different behaviours on A and B to be able to differentiate them (the rule is that the A-channel must not have more than 11 consecutive "1" whereas B can have any type of sequence).

This description can be broken down into the following points:

- Connect the TTCvi **A/ecl CHANNEL OUT** output to the TTCvx **A/ecl CHANNEL IN** input via a Y-adapter.

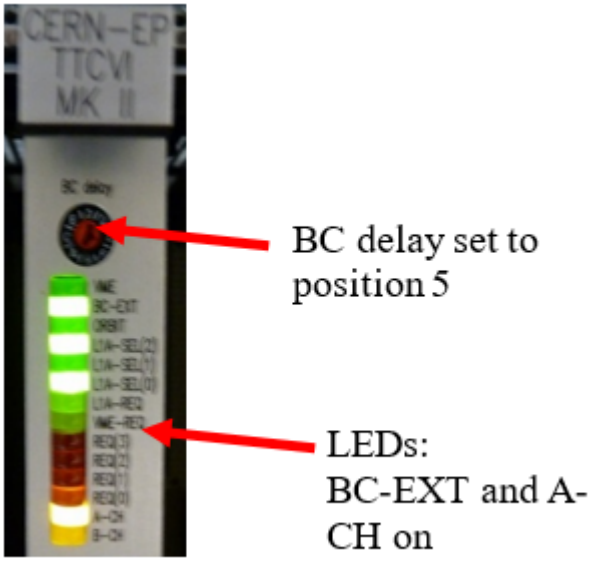

Figure A.4 Image of cabling for tuning a TTC system

- Connect, via a Y-adapter, one of the **TTCvx CLOCK OUT/ecl** outputs to the **TTCvi CLOCK IN bc/ecl** input. Check that the BC_EXT indicator is lit on the TTCvi as shown below. The TTCvx internal clock may be used.
- Set the TTCvi trigger mode (= 5) to random at the highest rate (100 kHz) and disable the event/orbit/trigger-type transfers. In order to do this write 0x7005 to the D16 VMEbus CSR1 register at offset 0x80. This can be done easily for vme_rcc_test. Note: The A24 base address of the TTCvi in the CERN reference system in TBED is 0x555500. This should light up the TTCvi A-Ch yellow indicator and the A/ecl CHANNEL OUT output should now carry 25 ns long trigger pulses.
- With an oscilloscope look at the TTCvx Channel-A input in respect to the clock output, as shown below.

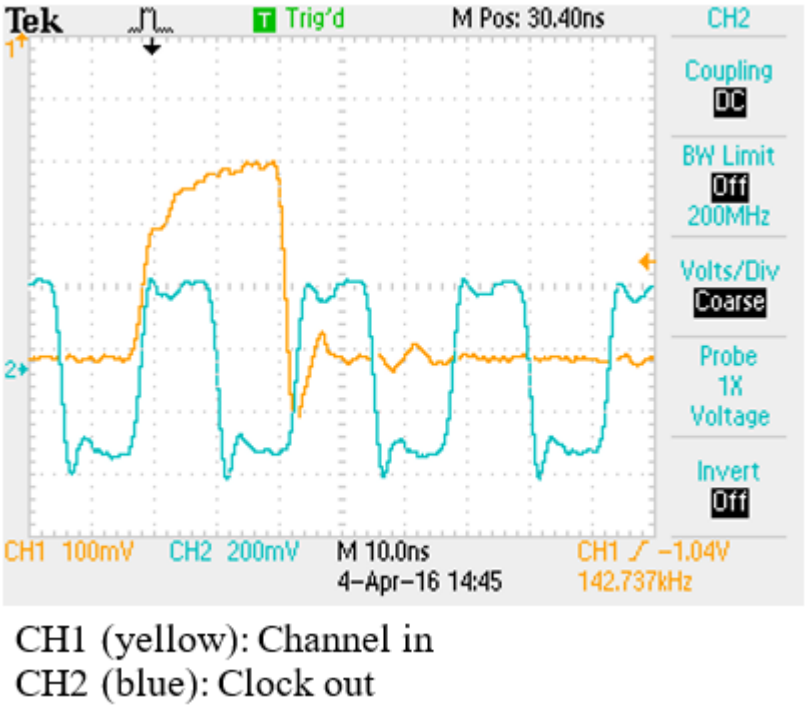

Figure A.5 Image of cabling for tuning a TTC system

- Adjust the TTCvi BC delay switch such that the rising edges of the Channel-A pulses occur within 4 ns before to 2 ns after the rising edges of the clock signal.
- Setting the delay switch in position 2 and using 1 ns long interconnecting cables for the clock and the A and B channels corresponds to the above mentioned timing criteria. Note from Markus Joos: Even though I used 1 ns cables, I had to set the switch to position 5 (see picture above) in order to meet the requirement of step 5.

A.2.2 Guide to TTC Channel B

The following section describes the structure of the TTC 'B channel' data stream, and how it may be decoded and operated by users. The information in this section is provided courtesy of Alessandra Camplani and the LAr group.

The data stream arriving through TTC B channel can be of two types: short broadcast commands or long individually-addressed commands/data.

Short broadcast commands are used to deliver messages to all TTC destinations in the system, while long individually-addressed commands/data are used to transmit user-defined data and instructions over the network to specific addresses and sub-addresses. These two types of command have different dedicated frame formats, as shown in Figure A.6 :

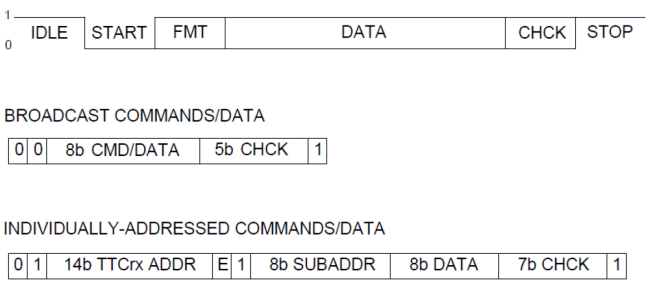

Figure A.6 Image of cabling for tuning a TTC system

The difference between the two command types can be illustrated with the example below. When not in use the B channel IDLE state is set to 1. When a sequence of commands is sent, the data transmission state changes from 1 to 0. After the first zero received it is possible to distinguish between short broadcast and long address commands: if the second bit in the stream is a 0 then the command is a short broadcast, if it is a 1 then the command is of long address type.

```
IDLE=111111111111
    Short Broadcast, 15 bits:
    00TTDDDDEBHHHHH:
        T= test command, 2 bits
        D= Command/Data, 4 bits
        E= Event Counter Reset, 1 bit
        B= Bunch Counter Reset, 1 bit
        H= Hamming Code, 5 bits
    Long Addressed, 41 bits
    01AAAAAAAAAAAAAAE1SSSSSSSSDDDDDDDDHHHHHHH:
        A= TTCrx address, 14 bits
        E= internal(0)/External(1), 1 bit
        S= SubAddress, 8 bits
        D= Data, 8 bits
        H= Hamming Code, 7 bits
```

The short broadcast command type is used to send two important values: the Bunch Counter Reset (BCR) and the Event Counter Reset (ECR).

The BCR is used to reset the bunch crossing counter, which is increased every clock cycle on the 40 MHz clock. This is a 12-bit counter, also called BCID. A BCR command is sent roughly every 89 μs, corresponding to the time that a bunch needs to do an entire circuit of the LHC. During this time the BCID counter reach its maximum value, 3564 counts.

The ECR is used to increase the event reset counter. The periodicity of this reset is decided by each experiment, with ATLAS having it set to 5 seconds. The event reset counter combined with the L1A counter gives the Extended L1ID (EVID). This is a 32-bit value consisting the L1A counter in the lower 24 bits, and the event reset counter in the upper 8. Every time that an ECR is received the upper counter is increased by 1 and the lower part is reset to zero. Every time that a L1A is received the lower part is increased by 1.

BCID and EVID values are used as a label for the data accepted by the trigger.

The long address command type is used to transport another important value: the Trigger Type (TType). Each L1A transmission is followed, with variable latency, by an 8-bit TType word. This word is generated inside the LVL1 Central Trigger Processor (CTP) and distributed from the CTP to the TTCvi modules for each of the TTC zones in the experiment via the corresponding LTP modules.

The presence of a Trigger Type within long address commands is announced by a sub-address (8 bits) set to 0.

Table A.6 Trigger type 8-bit word: Each bit represent the sub-detector which fired the trigger or the data type.

Sub-Trigger	physics	ALFA	FTK	LAr demonstrator	Muons	Calorimeter	ZeroBias	Random
Bit	7	6	5	4	3	2	1	0

As shown in Table A.6 , each bit has a specific role. In calibration mode, bits 0 to 2 can be used to distinguish between up to eight different possible types of calibration trigger within each sub-detector. Bits 3 to 6 are used to indicate which sub-detector or subsystem fired the trigger. Bit 7 represents physics trigger-mode when set to 1, and calibration mode when set to 0.

A.2.3 B channel decoding firmware

An effort is under way to provide a centrally maintained firmware module to decode TTC B-channel data. In the short term, users are advised to refer to a version produced for LAr front-ends by Alessandra Camplani. The module code can be found in gitlab:

https://gitlab.cern.ch/atlas-lar-ldpb-firmware/LATOME-ttc

The code itself is in the folder *code_ttc* and the files dedicated to TType decoding are: *Bchan_top.vhd*, *SMdecoding_cnt.vhd* and *TType_decoding*. The simulations for this specific part can be found in the *simulation* folder. Here there is a testbench for the *Bchan_top* entity and another one for *TType_decoding* entity.

Development of this module is ongoing, with the *latome_ttc* branch being actively maintained and kept up-to-date.

A.2.4 Channel B decoding software

In order to test channel decoding, it is recommended that users employ the menuRCDTtcvi application, provided as part of the ATLAS TDAQ software release. Within the application select 'BGO menu' and then option 13 'send asynchronous command'. From here it should be possible to select either a short of long command. In the case of a short command simply enter the data word to be sent. For a long command enter an address 0 (for broadcast), 0 for internal registers, subaddress 0 for trigger type, and the data word to be sent.

A.2.5 Useful documents

You may find additional useful information in this document from the ATLAS LAr group:

https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/community/CPPM_MiniFELIX_tests_results_and_TTC_system_experience.pdf

0 :!table: 11

Appendix B: FLX-712 Technical Information

This appendix will collect technical information for the FLX-712 board.

B.1 Overall Design

The FELIX card hosts 4 MiniPOD transmitters and 4 MiniPOD receivers. Each MiniPOD has 12 channels. The TTC clock from the ADN2814 is cleaned by an Si5345 or LMK03200. The clean 240 MHz is used as a reference clock for the GTH transceivers. Two of the PCIe hardcore EndPoints within the FPGA are used, with the PEX8732 PCIe switch used to connect them to a 16-lane slot.

An on-board 2 Gb FLASH memory can store 4 different firmware bit files. An on-board microcontroller (which the host can communicate with either via SMBus or through the FPGA and PCIe interface) can be used to select one of the four FLASH memory partitions and trigger FPGA programming from the image stored in the selected partition.

As shown in Figure B.1 and Figure B.2 , the FPGA has two Super Logic Regions (SLRs), referred to as devices in the software. To balance resource usage and 54 minimize the number of traces crossing the boundary each SLR has one 8-lane PCIe endpoint. For the 24-ch GBT firmware flavour, banks 126-128 and 131-133 are used.

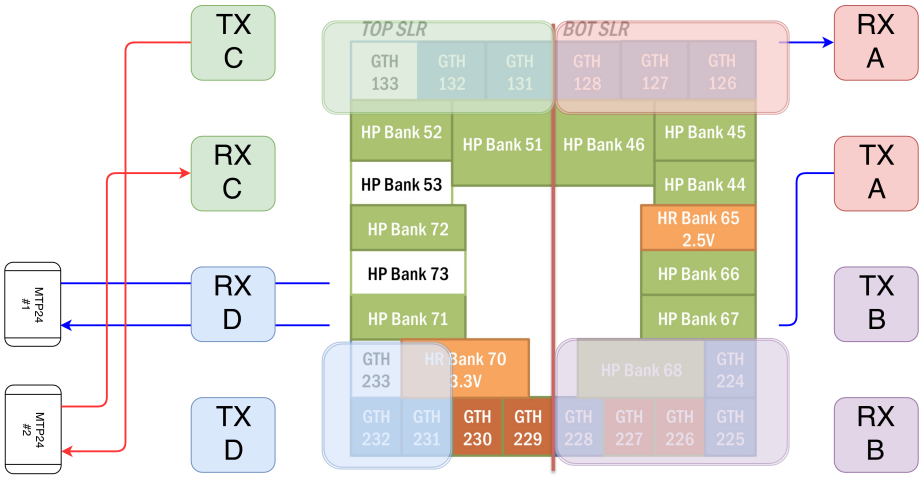

Figure B.1 24ch configuration.

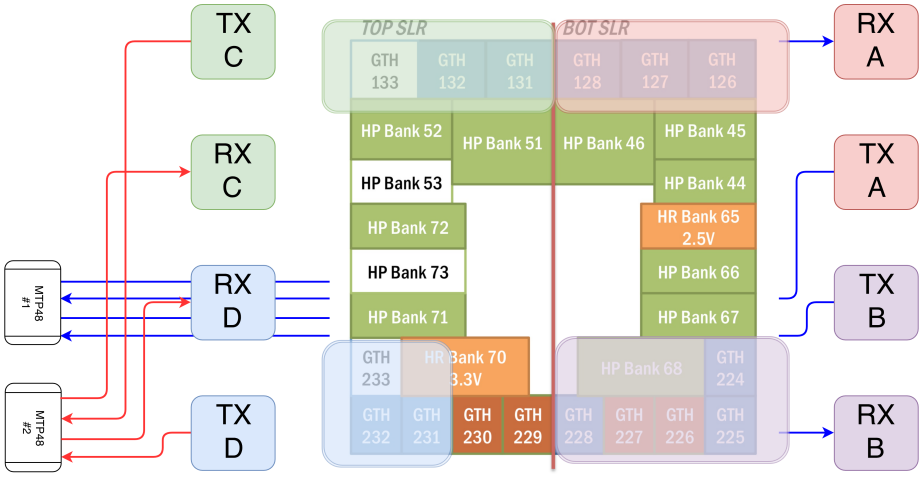

Figure B.2 48ch configuration.

B.2 Fibre Mapping and Connectivity

Every FLX-712 comes with PRIZM patches pre-installed, connecting the MTP inputs to the MiniPODs. These patches are custom made for the card and mean that the users should only need to connect to their data source to the MTP cable. No internal cabling work on the FLX-712 is required. The Fibre mapping and pin assignment for channels in each MiniPODs are shown described in this section. Two configurations are shown: 48 channel and 24 channel.

B.2.1 24 Channel Version

For 24 channel version, the channel number is 12 for each MTP coupler. Figure B.3 / Figure B.4 show how the banks are connected to the MiniPODs in this case.

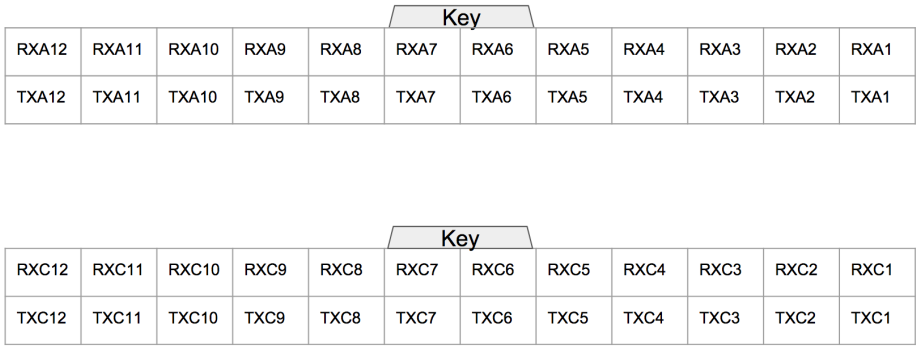

Figure B.3 24ch fibre mapping.

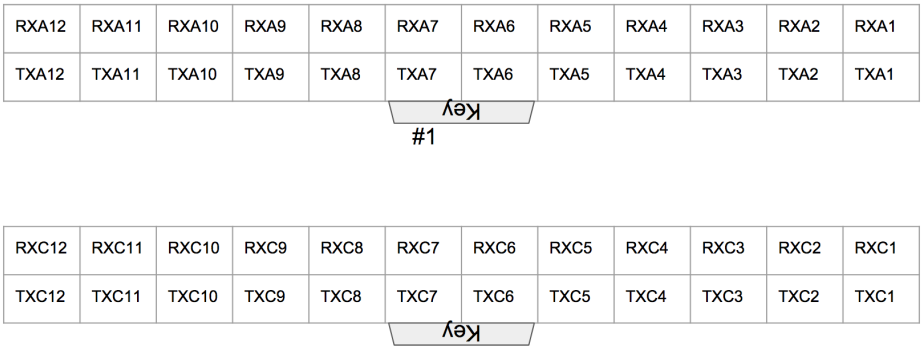

Figure B.4 24ch fibre mapping looking from MTP coupler.

B.2.2 48 Channel Version

For 48 channel version, there are 24-TX and 24-RX in fibre connected to each MTP coupler. Figure B.5 / Figure B.6 show how the banks are connected to the MiniPODs in this case.

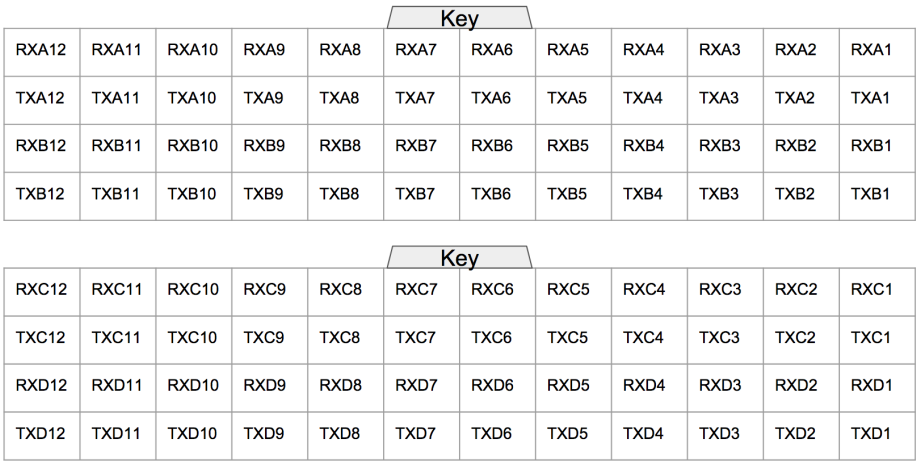

Figure B.5 48ch fiber mappping.

Figure B.6 48ch fibre mapping looking from MTP coupler.

0 :!table: 12

Appendix C: Guide to FELIX Data Structures

This appendix introduces conceptually the FELIX data formats. The description in terms of byte fields can be found in Appendix B of The Firmware Specification Document.

C.1 ToHost blocks

In the uplink direction (ToHost), FELIX firmware produces writes E-link data in the ToHost DMA buffers in blocks of 1 kB. Each block contains packets, called chunks, of a single E-link. If a chunk is larger than a block, or does not fit in the remaining space in a block, it is split in sub-chunks. If a block is partially filled, it is padded with zeros and output after a timeout. A sub-chunk can be of type:

- first: the first part of a chunk;
- middle: not the first nor the last part of a chunk;
- last: the last part of a chunk;
- timeout: filling to complete a block upon timeout;
- null: padding.

Headers are preceded by blocks. Sub-chunks are either preceded by headers or trailers depending on the firmware version.

C.2 TTC2H messages

The TTC2H is a virtual e-link whose data is generated by FELIX firmware upon reception of LTI messages.

C.3 FromHost blocks

Downlink messages are split by software in blocks whose format is parsed by firmware. Each block has size 32, 64 or 128 depending on the PCIe generation (3, 4 or 5). A header contains the E-link identifier and the payload size.

0 :!table: 13

Appendix D: Guide to Using FELIX with the GBT-SCA

This appendix is included thanks to Paris Moschovakos and the DCS team.

D.1 Introduction

The Slow Control Adapter ASIC (GBT-SCA, or SCA for short) is part of the GBT chip-set and is dedicated to the slow control of the front-end boards. It features several sub-devices that facilitate both front-end configuration and monitoring of environmental variables (voltages, temperatures, etc.) on and around the detector. The SCA contains an ADC, DACs, general purpose IO, and controllers for I2C, SPI and JTAG. An SCA is connected to a GBTX via any 2-bit E-link in 40 MHz DDR mode (80Mb/s) with HDLC encoding. Up to 41 SCAs can be potentially connected to a single GBTX with the corresponding link on FELIX configured accordingly.

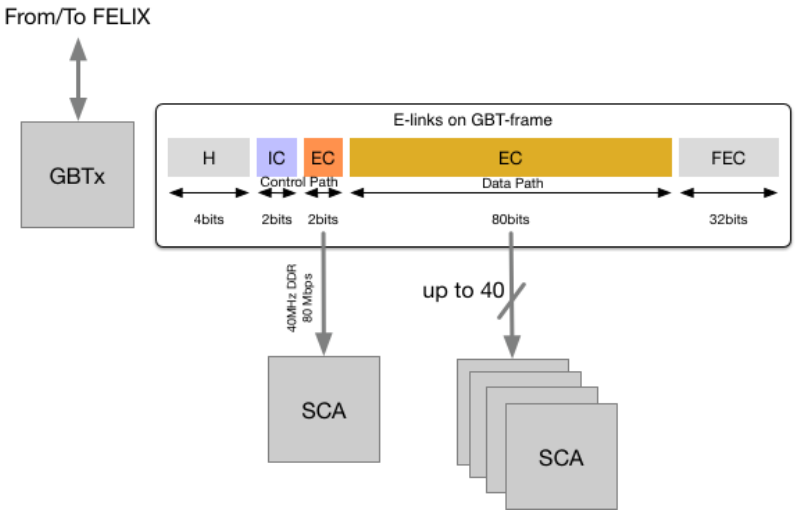

Figure D.1 GBT frame paths and E-links.

D.2 Typical test setup

A typical test setup consists of a board with a GBTX that is connected to FELIX via an optical fibre and to an SCA via one of the GBTX's E-links.

The Versatile Link Demo Board (VLDB) (https://espace.cern.ch/GBT-Project/VLDB/default.aspx) contains both a GBTX and an SCA. It can be directly connected to a FELIX card. (The VLDB demo board can be procured from the GBT group). A schematic of such a setup is shown in Figure D.2 .

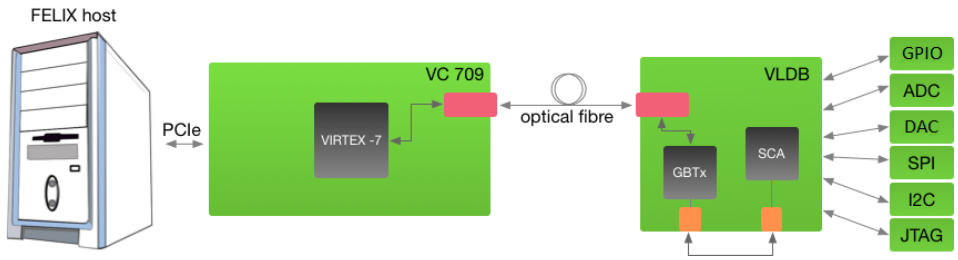

Figure D.2 Evaluation setup with a GBT-SCA on a VLDB. SCA and GBTX are interconnected on the VLDB externally via a pair of mini-HDMI connectors (J32 (PRIMARY) and J33 (SCA PORT), in that case using the GBT EC E-link).

To simplify the evaluation of the setup, the VLDB possesses two LEDs connected to two general-purpose digital outputs of the SCA. This can be used to quickly validate visually the communication path and functionality all the way from the FELIX host to the SCA itself. In order to do that, the 'fec' tool, of the *ftools* family as mentioned in Section 6, can be used.

The following command instructs the SCA connected to the VLDB with the GBT link <*gbt_link_number*> EC link, to blink one of its LEDs 50 times at a rate of 5Hz (*-t 100*: 100ms on, 100ms off; last character is an 'o' for 'output'; *-x* selects the digital output: on the VLDB there's an LED on 18 and one on 21):

```
fec -G <gbt_link_number> -r 50 -t 100 -x 18 o
```

D.3 Procedure to set up an E-link to a GBT-SCA

A configuration procedure is needed both for FELIX and the GBTX itself. The configuration is mostly a description of the setup at hand and the mapping of the e-links that are connected. There are also some setup specific parameters to be configured.

In the case the SCA is connected to the dedicated EC E-link, one should check that this E-link is enabled using the *elinkconfig* GUI. That specific E-link is pre-configured with the appropriate HDLC SCA encoding and corresponding bit endianness and can be used directly for an SCA. Figure D.3 shows an *elinkconfig* screenshot with the enabled EC channel in both the to-host and from-host direction, indicated by the checked tick boxes with yellow background labeled 'EC'; the hexadecimal number in brackets is the FELIX E-link ID associated with this E-link. Note that we happen to have selected GBT link #2, which is also reflected in the hexadecimal numbers next to the various E-link 'enable' tickboxes.

In the case that one wants to use an E-link from one of the GBT E-groups instead of or in addition to the EC E-link, one has to configure FELIX accordingly via *elinkconfig*. The SCA uses HDLC encoding instead of the typical 8b/10b which is the default for the data E-links, so this needs to be configured for the E-link. Moreover, the bit orientation is different from the 'normal' data E-links. By selecting the HDLC format in the drop-down menu, elinkconfig takes care both of the orientation and the encoding, indicating that an SCA is connected to that specific GBT group and path. As an example, Figure D.3 shows that the 8th 2-bit E-link of E-group 0 of GBT link 2 has been enabled in the FELIX configuration in both the to-host and from-host direction. The FELIX E-link ID is shown here to have a value of 0x087. Using *ftools* tool felink one can confirm this is indeed the requested E-link:

```
> felink -e 87
E-link 087 = GBT #2 group #0 path #7, bit#14 width=2
```

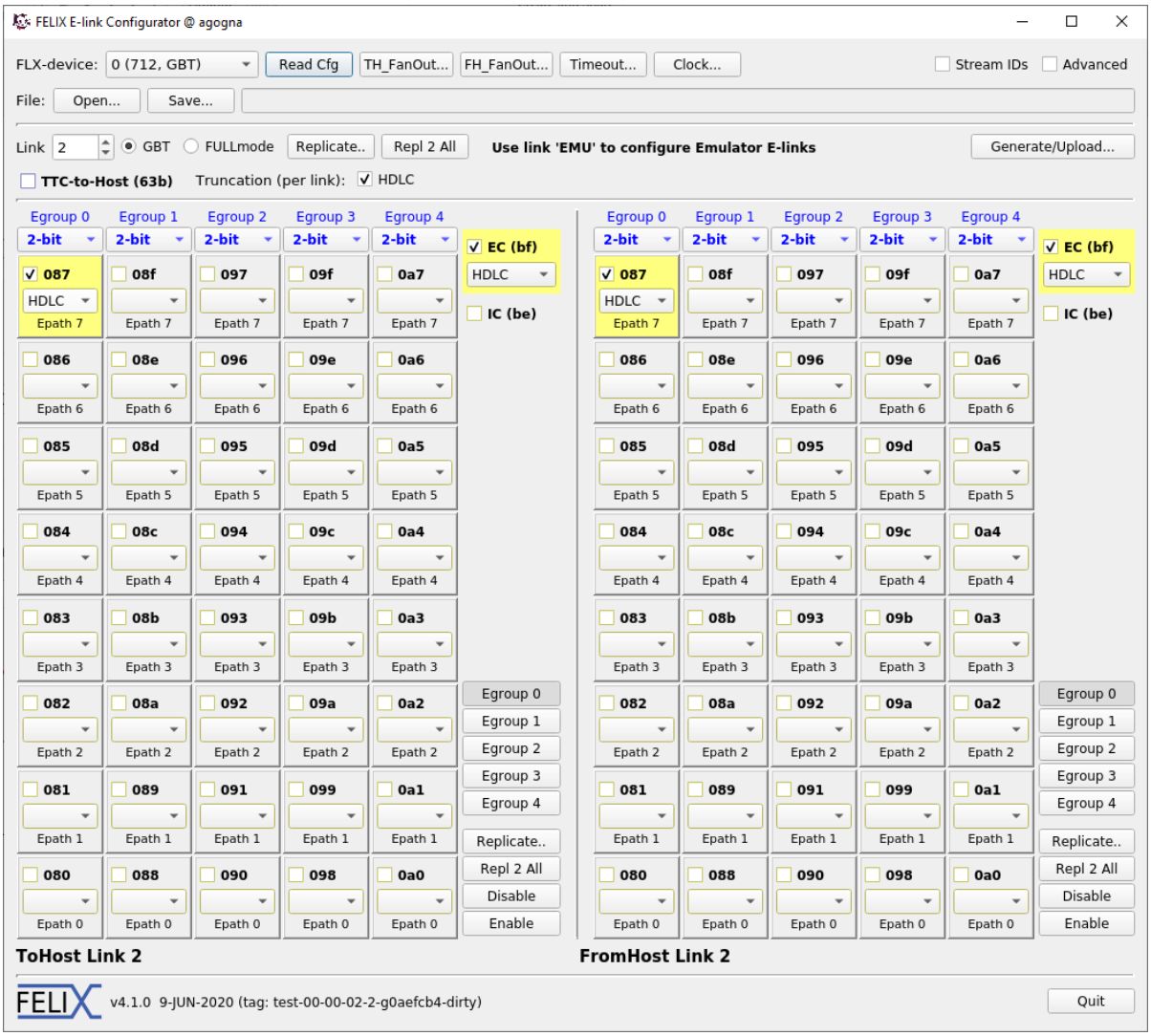

Figure D.3 Enabling E-links connected to GBT-SCAs.

D.4 Low level operations with the *fec* tool

The fec tool, is a dedicated tool from the *ftools* suite of tools to demo the handling of a number of I/O channels available on the SCA. Please check the full list of possible operations in Section 6.

D.5 A Software Suite for the Radiation Tolerant GBT-SCA - The Production system

The on-detector DCS system that handles the slow control traffic and the configuration of the front-end electronics, based on the GBT-SCA. The global scheme is presented in the following figure.

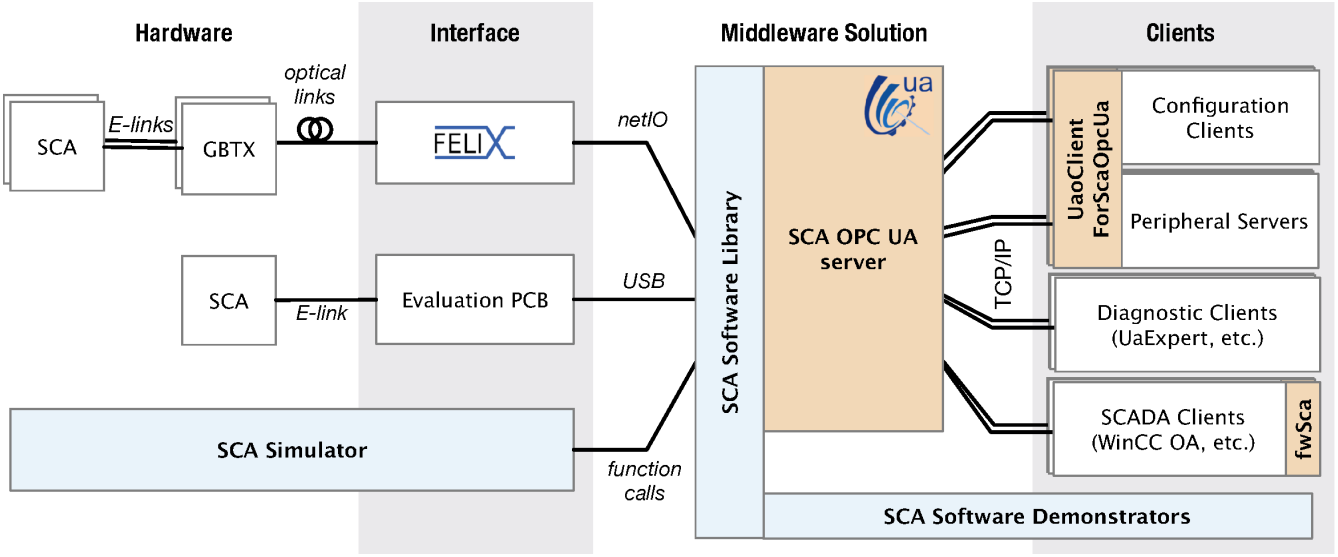

Figure D.4 Global picture of the software suite. The SCA Software package, in light blue, comprises the SCA Software API to communicate with the SCA via different back-ends, the SCA Simulator to emulate SCA traffic for testing and development, and the Demonstrator tools which are used for standalone operations. The SCA OPC UA server and its ecosystem, in orange, is the middleware of choice to exchange data with the front-ends. UaoClientForScaOpcUa is a library that clients use to communicate with the SCA server. Finally, the fwSca module automatizes the integration of the server data into SCADA systems.

The slow control and configuration traffic, unlike physics data, has different requirements in terms of throughput, latency, availability and reliability. SCA DCS is the software that handles the SCA traffic arriving on the FELIX card. Towards the FELIX clients it is based on the middleware Open Platform Communications Unified Architecture (OPC UA, of the OPC foundation, (https://opcfoundation.org) which is an industry standard for secure and reliable exchange of data in industrial automation and other controls-related areas.

The server/client architecture that the platform uses, allows for different purpose clients to be served by a single server per FELIX host. The data flow to/from the ATLAS control room, not only serves the control and monitoring data of the detectors' conditions but also implements the configuration path of the on-detectors electronics and their initialization for data taking or calibration. In addition, system experts can monitor the status of the employed technology and get statistics and other information in order to diagnose the various system layers.

All those requirements potentially imply many different OPC UA clients that would like to receive SCA data from the setup at the same time. The chosen OPC UA architecture ensures the reliable and seamless data delivery and the compatible integration into the current DCS systems. This means that OPC UA clients in both DCS and a detector configuration server can communicate with the same SCA and the OPC UA server arbitrates their access.

D.5.1 OpcUaSca server

The provided OPC UA server implementation for the SCA is based on the ScaSoftware (explained below) intending to profit from all features of the ScaSoftware library and providing a high-level and user-friendly OPC UA address space to OPC UA clients.

The OpcUaSca server has been designed and implemented using the quasar framework (see https://github.com/quasar-team/quasar). Its design is presented in the following figure.

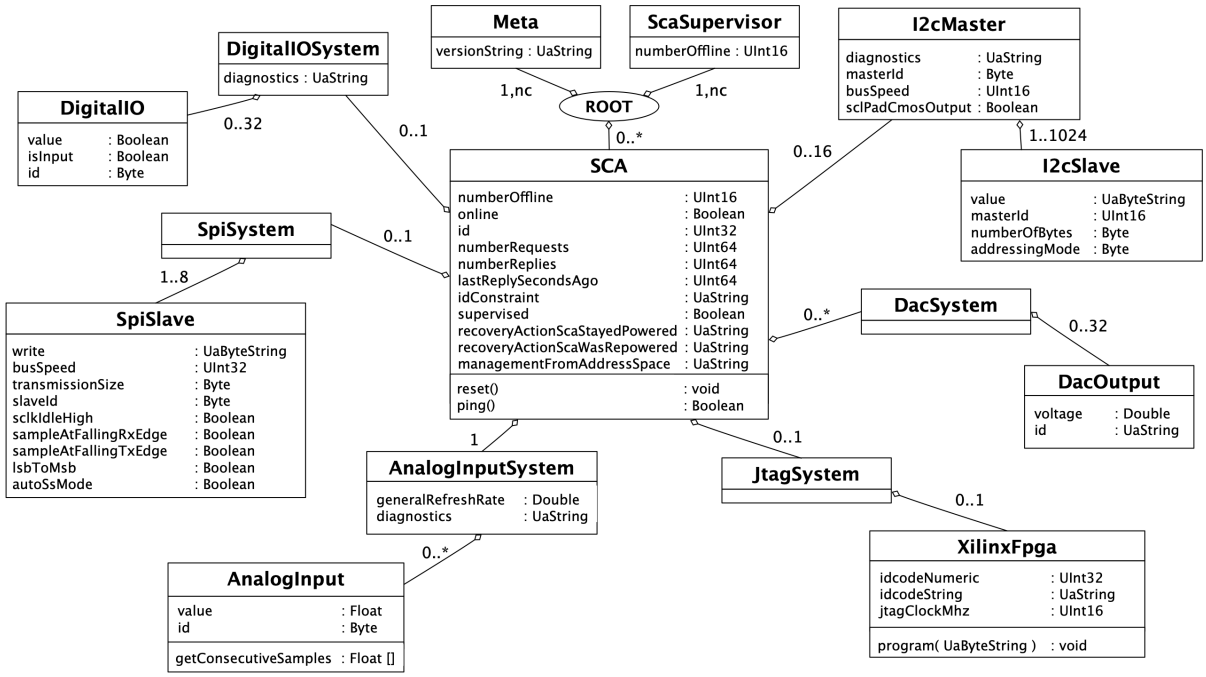

Figure D.6 The quasar design diagram of the OPC UA server for the SCA.

As of May 2020, the server supports the following functionality:

- Communication with any number of SCAs, through NetIO or any other HDLC backend. Each SCA is identified in its address-space by a name, and its unique 24-bit "SCA identifier" which is written by the chip manufacturer in the SCA silicon. The identifier is read using the ScaSoftware library when a connection to given SCA is opened.
- Up to 32 ADC channels per SCA which are polled with the configured conversion frequency. Note that channel 32 has no external connection; it is connected to the on-chip temperature sensor that monitors the SCA temperature.
- Up to 32 General Purpose I/O pins per SCA. Each pin can be configured as an input or output through the server config file.
- Up to 4 DACs per SCA; the DACs take the desired voltage as a float (0..1V).
- Up to 8 SPI slaves per SCA. The SPI configuration (like speed, phase, mode . . .) can be configured in the server config file.
- Up to 16 independent configurable I2C master controllers
- Up to 1024 I2C slaves per I2C master controller
- A single Xilinx FPGA over JTAG interface

D.5.2 ScaSoftware Package

In the SCA Software package core there is a library that is structured in modules that implement the required functionality in various layers. The library was designed to be flexible and easily adaptable to the diverse systems intended to use it by its polymorphic HDLC back-end. A block diagram of the software architecture of this library is shown in Figure D.7

Moreover, the SCA Software package contains the Demonstrators which are tools that directly use the library and are used for testing and for low level diagnostics. Finally, as part of the package, an SCA Simulator was developed that is able to generate SCA traffic, simulating realistic SCA behaviour, in order to allow for development and testing without real hardware.

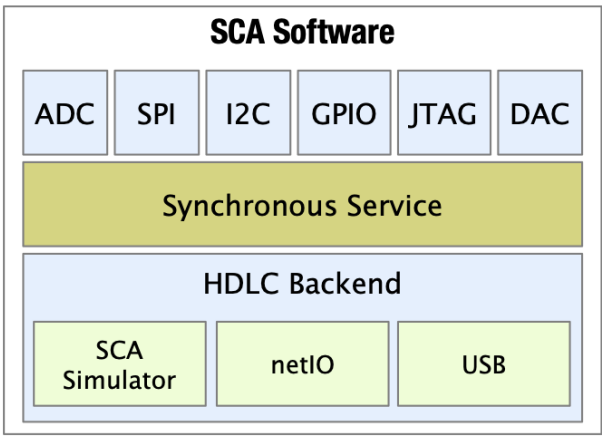

Figure D.7 SCA Software Library stack.

- The library is a modular piece of software supporting SCA chips no matter how it is physically connected to the host system. Therefore the core part of the library operates on protocol data units of the SCA chip, which normally would be encapsulated in the HDLC protocol. There are a number of predefined "HDLC backends" which are services to send such encapsulated SCA requests and receive replies.
- The library scales from the simplest use cases up to scenarios of thousands of SCAs.
- The library is able to profit from concurrency features of the host system, including multi-core and multi-threaded operation.
- The library is written in a chosen version of the standard C++ dialect.
- The library is designed with reliability and robustness as a key design choice because it would serve critical, 24/7 communication.

D.6 SCA References

1. P. Moschovakos, P. P. Nikiel, et al., "A Software Suite for the Radiation Tolerant Giga-bit Transceiver - Slow Control Adapter", presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper WEPHA102.
2. OpcUaSca repository, https://gitlab.cern.ch/atlas-dcs-opcua-servers/ScaOpcUa
3. ScaSoftware repository, https://gitlab.cern.ch/atlas-dcs-common-software/ScaSoftware
4. Uao Client for OpcUaSca, https://gitlab.cern.ch/atlas-dcs-opcua-servers/UaoClientForOpcUaSca

0 :!table: 14

Appendix E: Guide to Using FELIX with the SCA eXtension

The Slow Control Adapter eXtension (SCAX) is an FPGA module that emulates the GBT-SCA's communication with the back-end (i.e. FELIX and the OPC UA server) in order to provide access to registers in the FPGA. The OPC-UA server uses the protocol for the SCA's I2C interface. SCAX's connection to the registers is, however, parallel and not I2C.

E.1 Introduction

The SCA ASIC is typically installed on front-end boards in order to configure and monitor other front-end devices (usually other ASICs) on the board. Being radiation-tolerant, it can be deployed on front-end boards that are subject to high radiation doses. However, there may be several parts of the DAQ system that are FPGA-based, and are either situated in parts of the detector which are not exposed to a disruptive particle flux, or are even outside the experimental cavern, in the counting room (or USA15).

The New Small Wheel (NSW) DAQ system for instance, uses the SCA to configure and monitor six other front-end ASICs. The NSW electronics includes also FPGAs deployed on the rim of the wheel (i.e. Pad Trigger) and in USA15 (NSW Trigger Processor). The board of the latter FPGA-based system, does not feature an SCA, but like the ASICs, its parameters must be tuned and its status monitored. Even though the SCA is not present in the Trigger Processor, it was desirable to include it in a unified configuration and status monitoring scheme.

The solution came in the form of the SCAX, which makes use of the FPGA's direct interface with FELIX (via optical fiber and by deploying the GBT-FPGA [17] in its logic) , to communicate with the OPC UA server (see Appendix D). The SCAX's logic, has been designed in such as a way as to be completely transparent to the OPC server, which was initially designed to interface only with the SCA. By emulating the command-response protocol dictated by the SCA's specifications, the SCAX can establish a connection with a server as an SCA FELIX, and use the server's features to access the registers of the FPGA in which it is implemented; this is being achieved by mimicing the SCA's I2C device and to write into and read from the FPGA fabric registers. Full access to registers in the FPGA, using the already existing and well-established OPC software ecosystem is thus provided. The general scheme can be examined in Figure E.1 . SCAX has been deployed successfully in the NSW Trigger Processor FPGA, but it can be used by any FPGA device that features a direct connection with FELIX.

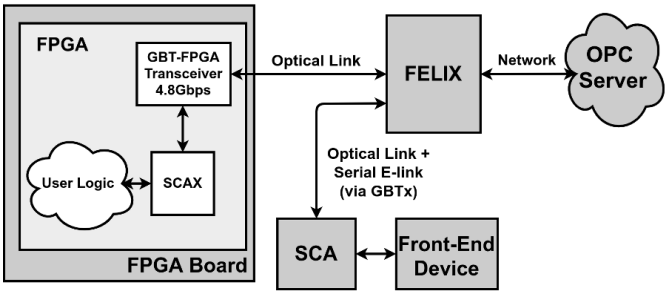

Figure E.1 Connectivity of the SCA and the SCAX with the OPC Server.

E.2 Establishing a Connection between the SCAX and FELIX

This Appendix will focus on the procedure it must be followed by the user of the SCAX to connect the instance of the logic with FELIX. The reader may refer to [19] for a detailed user guide on how to deploy the SCAX in an FPGA, and to [20] for a detailed write-up of the module's architecture.

If a proper configuration is found, then going through all the steps mentioned below will not be necessary, but it is strongly recommended to follow them after the first attempt to establish a connection between all nodes.

Note also that it is not needed to connect the SCAX with any of the user logic registers when going through the connectivity validation procedure.

E.2.1 General Steps

In principle, the user must follow the procedure below to establish a system that allows access to the registers of their FPGA logic using the SCAX ←→ FELIX ←→ OPC Server arrangement:

1. Deploy the SCAX in their FPGA netlist
2. Connect the SCAX either with a GBT-FPGA instantiation, or with a GBTx device, either of which must feature a direct optical bidirectional connection with FELIX
3. Configure FELIX accordingly
4. Validate the SCAX's RX path
5. Validate the SCAX's TX path
6. Connect the OPC UA server with the SCAX instance in question

In the following subsections, each step will be addressed in more detail.

E.2.1.1 Deploying the SCAX in a pre-existing FPGA design

The procedure that must be followed by the user/designer to deploy the SCAX into their firmware is covered in greater detail at the associated user guide [19]. However, some recommendations will be mentioned in this subsection.

First of all, the clocking scheme must be chosen carefully, in order to avoid data corruption on both directions of the communication. For a GBT-FPGA-driven implementation, the SCAX's E-link clocks (40, 80, 160 and 320 MHz) must be related with the transceiver's reference clock. For a GBTx-driven use-case, the SCAX's E-link clocks must be derived from the E-link clock, as delivered by the GBTx to the user FPGA.

Also, for the first implementation iterations where the user is attempting to establish a connection with FELIX, the following pins must be tied to a Xilinx Virtual Input/Output (VIO) IP core: *rx_swap*, *tx_swap*, *dbg_fifo_rd*, and *ena_flx_test*.

Finally, it is strongly recommended to set the SCAX in 8b10b, 80 Mbps, and in *debug mode* via the associated generics.

E.2.1.2 Connecting the SCAX to a GBT-FPGA or a GBTx

There are two use-cases that must be studied; they are depicted in Figure E.2 .

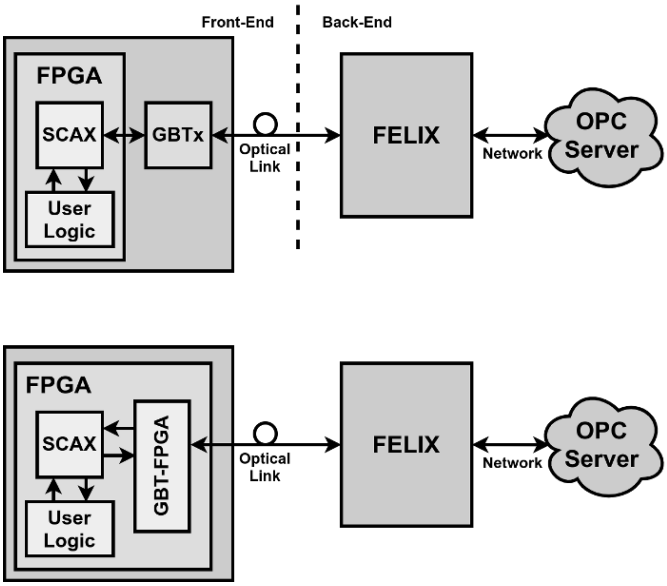

Figure E.2 Two possible ways to connect the SCAX with the OPC Server.

In the case of connecting the SCAX with the FELIX/OPC Server via a GBT-FPGA, the user should connect the *parallel* ports of the SCAX (e.g. *rx/ tx_elink2bit*) with the 84-bit TX/RX bus of the GBT-FPGA instance. Depending on the bits of the bus that are chosen, the SCAX will belong to a different E-link. If the recommended SCAX interfacing configuration is chosen (i.e. 8b10b 80 Mbps), then the user should use the 2-bit TX/RX ports of the SCAX (2-bit is for 80 Mbps, 4-bit is for 160 Mbps etc.), and connect it to Egroup0 or Egroup1 of FELIX. Egroup0 EPATH0 corresponds to bits [1:0] of the GBT-FPGA's TX/RX bus. Egroup0 EPATH1 corresponds to bits [2:3] of the bus. Egroup1 EPATH0 corresponds to bits [16:17] of the bus, etc. If the same part of the bus is chosen for both directions, then SCAX will reside on the same E-link for both the To-Host and the From-Host paths, which eases the procedure.

In the case of connecting the SCAX with the FELIX/OPC Server via a GBTx, the user should connect the *serial* ports of the SCAX (i.e. *rx/ tx_elink*) with the desired GBTx pins. Testing so far has shown that by choosing a low data rate (i.e. 8b10b 80 Mbps), the communication of the SCAX with FELIX via the GBTx is easier, as it is probably not needed to train the GBTx on the datastream coming from the SCAX (see the GBTx manual [3] for more details on how to perform training on the GBTx ports). If a faster data rate has to be chosen due to system restrictions, it is recommended to perform training on the GBTx bank that the SCAX's TX path corresponds to, in order to avoid data corruption on that direction. Finally, the user must deduce the E-link ID to which the SCAX corresponds, by going through the PCB's schematic and taking note to which GBTx package pins the SCAX's serial I/Os are connected.

Note that different Egroups of FELIX support specific protocols and data rates for semi-static builds.

E.2.1.3 Configuring FELIX Prior to Connectivity Testing

If the To-Host (SCAX-TX) and From-Host (SCAX-RX) E-links to which the SCAX corresponds are known, then these must be activated on the FELIX side via the *elinkconfig* tool, as per the data rate and protocol to which the SCAX is configured via its generics. *felix-star* must be running throughout

the connectivity testing.

E.2.1.4 E.2.1.4 Validating the SCAX's RX Path

In order to validate the SCAX's RX path (or *From-Host* direction, in FELIX jargon), then after configuring the FELIX E-links accordingly, the embedded SCAX's ILA must be used to probe the RX path. The ILA is activated if the SCAX is deployed in *debug mode*, as recommended for the first implementation attempt.

There are two ports that must be examined: these are the *din_dbg* and *drdy_dbg*. The 10-bit *din_dbg* bus yields the decoded data originating from the FELIX E-link. In the case of 8b10b encoding at 80Mbps, pairs of standard commas, K28.5, *must* appear periodically on the bus. This is the *0xbc* byte, accompanied by "11" in the first two bits of the bus. Hence, if the communication is sound, the *0x3bc* word will appear every five cycles, and the *drdy_dbg* will go high for the cycle the word appears. If any of these is not true, then the user should check if the optical link is aligned via the *$ flx-info GBT* command in FELIX, then check if the E-link configuration on the FELIX side is correct. If these appear to be OK, then the user should either attempt to probe the aforementioned ILA ports after attempting a different *rx_swap* state, or after attemtping a different *fereverse* configuration on the FELIX side.

E.2.1.5 Validating the SCAX's TX Path

In order to validate the SCAX's TX path (or *To-Host* direction, in FELIX jargon), then after configuring the FELIX E-links accordingly, the VIO connected to the SCAX's critical ports must be used to send a test packet to FELIX. First of all, the user should have *felix-star* running in FELIX, and *felix-stest-swrod* subscribed to the To-Host E-link of the SCAX, in order to ensure the test packet is indeed being received by FELIX. Then, the user should toggle the VIO's port *ena_flx_test* from low to high and then back to low (note that it *must* be kept low when not used). By doing this, the SCAX sends the following message to FELIX: *0xff 0x63 0xe5 0x5e*. If the message is reported by *felix-stest-swrod*, then the communication on that direction has been established. If not, then the user should check if the optical link is aligned via the *$ flx-info GBT* command in FELIX, then check if the E-link configuration on the FELIX side is correct. If these appear to be OK, then the user should attempt to send the message again after attempting a different *rx_swap* state, or after attempting a different *fereverse* configuration on the FELIX side. If a GBTx is used in the communication chain, then training the GBTx should also be considered.

E.2.1.6 Connecting the OPC Server

If both TX/RX paths are validated, then the user should attempt to connect the OPC server with the SCAX. Note that it is advised to check the system's state by trying to connect the server with already existing *SCA's*, prior to any SCAX connection attempts. If the server and FELIX seem to be working as they should, then the SCAX instance can be added to the OPC's configuration .xml. Choosing the correct E-link as a parameter in the .xml is crucial.

If the OPC server can connect to the SCAX (There is no reason not to, if Steps 4 and 5 have been validated by the user.), then the user may implement the SCAX again, in non-debug mode and by removing the VIO (note that the *rx_swap* and *tx_swap* values that work must be retained). If the SCAX still connects, as it should, then the user may proceed with interfacing the SCAX with the rest of their logic, as per the SCAX user guide. Note that if a GBTx is used, it is recommended that the

user always train the GBTx after configuring their FPGA.

If the server fails to connect, then Steps 4 and 5 should be revisited. Note that testing so far has shown that if the OPC server fails to connect, felix-star must be restarted prior to another connection attempt.

0 :!table: 15

Appendix F: External emulators

Dedicated firmwares allow to turn a FELIX card into a data generator for testing and performance assessment purposes. A FELIX in such configuration is called external emulator, where the adjective *external* is meant to avoid confusion with the internal emulator present in the GBT and FULL mode firmware flavours. Two kinds of external emulators exist: FELIG for the GBT mode, and FMEmu for FULL mode. Instructions on how to operate FELIG and FMEmu are listed in the following.

F.1 FELIG

The FELIG firmware is available for the FLX-712 card (previously only the HTG-710). Full details are available in the dedicated user manual on CDS:

https://cds.cern.ch/record/2752360/

F.2 FMEmu

The FMEmu firmware is available for the FLX-712 and FLX-711 card models. It can be loaded on a 48-channel card but it will use 24 channels only. The FMEmu can be connected to a FELIX via a patch panel or using MTP-24 loopback fibres (in the latter case FELIX and FMEmu have to be hosted on FLX cards with the same number of channels). The FMemu can be set to receive the clock from the TTC system as a long as the TTC ST fibre is connected to it. The FMEmu can send data continuously or in triggered mode (upon reception of a L1A from FELIX). The FMEmu supports the XOFF traffic control system.

F.2.1 Quick start guide

Instructions on how to setup a FMEmu+FELIX system are listed in the following. Commands have to be entered in both the FELIX and FMEmu hosts. To distinguish between the two hosts the commands are preceded by the labels [FELIX] and [FMEmu]. Comments that are not commands are written within parenthesis.

Felix configuration: On elinkconfig enable all the desired ToHost links. In addition for each GBT link, enable the following FromHost e-links:

- Egroup 0, Epath 0 (2-bit wide, 000), 8b10b encoding (for the XOFF)
- Egroup 1, Epath 1 (8-bit wide, 009), TTC-3 encoding (8-bit wide, for the TTC)

Alignment procedure:

```
[FELIX] # (configure clock selection and links with elinkconfig)
[FMEmu] flx-config MMCM_MAIN_LCLK_SEL 0 (0 is TTC clock | 1 for local clock)
[FELIX] flx-init
[FMEmu] flx-init
[FELIX] flx-info link (check for alignment)
[FELIX] flx-info freq (check if all rxoutclk frequencies match (+/- 1 Hz)
```

```
[FMEmu] flx-info link (check for alignment)
```

Once the links are aligned the FMEmu can be started in either continuous mode or triggered mode. In the latter case the L1A received by FELIX from a TTC system are forwarded to the FMEmu.

Continuous mode:

```
[FMEmu] # if you want to enable XOFF, else skip
[FMEmu] flx-config FMEMU_CONTROL_XONXOFF 1
[FMEmu] # Use the ffmemu utility. For example to generate 128-byte messages upon the
reception of L0As
[FMEmu] ffmemu -c -T -w 32
```

F.2.2 FMEmu data format and payload

Each FMEmu message consists of a 32-bit field containing the extended L1ID, followed by an incremental sequence of bytes. The payload size can be set to constant using

```
flx-config FMEMU_RANDOM_CONTROL_SELECT_RANDOM 0
flx-config FMEMU_COUNTERS_WORD_CNT <value>
```

Otherwise, the payload size is randomly drawn from a distribution. The payload size distribution can be generated with a script such as fragsizegen and loaded onto the FMEmu with

```
femuran <file.coe>
```

The FMEmu is capable of generating chunks of size up to 4-5 kB at 100 kHz.

0 :!table: 16

Appendix G: XOFF Connection

G.1 Introduction

In the FULLMODE firmware of FELIX, backpressure can be applied to the Frontend links by means of the XOFF mechanism. The XOFF mechanism was designed to prevent data loss in case of a short burst of high bandwidth, especially if the total (sum) bandwidth of the links exceeds the available PCIe bandwidth.

In the case of XOFF, it is assumed that the Frontend electronics host enough buffer memory in order to store the data while XOFF is active.

G.2 Operation

Every FULLMODE channel (referenced as FM0..FM23) in FELIX has a 16 kB channel FIFO. A (High/Low) watermark level can be set for each fifo in steps of 1 kB, the default value of both high and low are set to 11kB. If the channel FIFO is filled beyond the high watermark, XOFF will be issued for the corresponding E-Link. If the fifo is emptied beyond the low watermark, XON will be issued.

To enable the XOFF mechanism which is disabled by default, two actions must be taken:

- The link XOFF setting must be enabled for the given link. This can be done through fexoff.
- The FromHost / Downlink GBT E-Link must be Enabled as 2-bit / 8b10b to be able to transmit XON and XOFF K-Characters
    - The E-Links that are capable of transmitting XOFF for a related FULL mode link (FM0..FM23) are listed in XOFF E-Links
    - Elinks can be configured as 2-bit/8b10b using elinkconfig or feconf

For testing purposes, a soft XON or XOFF can be transmitted using the fexofftx tool. This tool can also be used to manually set the frontend in XON state while it is in XOFF.

G.3 XOFF capable E-Links

Table G.7 XOFF E-Links: E-Links on the GBT Downlink in FULL Mode firmware, capable of transmitting XOFF

GBT Downlink	Egoup 0	Egroup 1	Egroup 2	Egroup 3	Egroup 4	
0	FM0..7	FM8..15	FM16..23 [1]	N.A.	N.A.	
1	FM1	N.A.	N.A.	N.A.	N.A.	
2	FM2	N.A.	N.A.	N.A.	N.A.	
3	FM3	N.A.	N.A.	N.A.	N.A.	
4	FM4	N.A.	N.A.	N.A.	N.A.	

GBT Downlink	Egoup 0	Egroup 1	Egroup 2	Egroup 3	Egroup 4	
5	FM5	N.A.	N.A.	N.A.	N.A.	
6	FM6	N.A.	N.A.	N.A.	N.A.	
7	FM7	N.A.	N.A.	N.A.	N.A.	
8	FM8	N.A.	N.A.	N.A.	N.A.	
9	FM9	N.A.	N.A.	N.A.	N.A.	
10	FM10	N.A.	N.A.	N.A.	N.A.	
11	FM11	N.A.	N.A.	N.A.	N.A.	
12	FM12..19	FM20..23	N.A.	N.A.	N.A.	
13	FM13	N.A.	N.A.	N.A.	N.A.	
14	FM14	N.A.	N.A.	N.A.	N.A.	
15	FM15	N.A.	N.A.	N.A.	N.A.	
16	FM16	N.A.	N.A.	N.A.	N.A.	
17	FM17	N.A.	N.A.	N.A.	N.A.	
18	FM18	N.A.	N.A.	N.A.	N.A.	
19	FM19	N.A.	N.A.	N.A.	N.A.	
20	FM20	N.A.	N.A.	N.A.	N.A.	
21	FM21	N.A.	N.A.	N.A.	N.A.	
22	FM22	N.A.	N.A.	N.A.	N.A.	
23	FM23	N.A.	N.A.	N.A.	N.A.	

G.4 Retransmission

Since Version 4.11 of the firmware the XON and XOFF K-Characters will not only be sent on a change of the XOFF state of the FULL mode channel FIFO, but the transmission of the state will also be repeated with an interval of 4 us (160 Bunch Crossing cycles). Additionally, a new TTC option; TTC-8 has been added in version 4.11 which continuously transmits the XOFF state on bit 1, and L1A on bit 0 of a 2-bit (80 Mb/s) E-Link.

G.5 Data format

The XOFF and XON commands are sent over a regular 8b10b (2-bit/80 Mb/s) E-Link. This E-Link can still be used to transmit 8b10b encoded messages to the FrontEnd. In case of an XOFF or XON event, the command will be sent as a special K-Character, shortly interrupting the data transmission towards the frontend.

The 8b10b encoding for the E-Links is encoded MSB first, that means the Most significant bit is serialized first.

Table G.8 Data format of XOFF capable E-Link

Command	K-Char	10b+	10b-	8b	Remark	
XOFF	K28.2	30A	0F5	5C	Set Front-End in XOFF state and start internal buffering	
XON	K28.3	30C	0F3	7C	Resume normal operation	
IDLE	K28.5	305	0FA	BC	E-Link is idle, also used for E-Link alignment	
SOC	K28.1	306	0F9	3C	Start-Of-Chunk	
EOC	K28.6	309	0F6	DC	End-Of-Chunk	
Data	Dxx.x				Any decimal character (non-K) will be treated as regular payload, but has to be incapsulated within SOC and EOC K-chars	

G.6 XOFF Statistics

The FELIX firmware includes a mechanism to measure some statistics about XOFF. This mechanism can measure:

- The peak duration of XOFF
- The average duration of XOFF
- The number of XOFF events that occurred

To get the peak value of XOFF for XOFF channel 5 (05 can be replaced by any value between 00 and 11 for each PCIe endpoint) the following command returns the duration as a 64-bit hexadecimal number, in 25ns (Bunch-crossing clock) bins.

```
flx-config XOFF_PEAK_DURATION05
```

To obtain the number of XOFF events for a certain channel, the following command can be issued.

```
flx-config XOFF_COUNT05
```

To obtain the average duration, the following number must be divided by XOFF_COUNTnn:

```
flx-config XOFF_TOTAL_DURATION05
```

The tool fexofftx can also display the statistics shown above:

```
$ fexoff
FIFO threshold Low=11 High=11 (4 MSBs, i.e. in 1/16ths of the FIFO size)
Link  0 XOFF: disabled  THRESH-X: L=0 H=0 (latched=0)  STATS: count=0 duration[25ns]
total=0 peak=0
Link  1 XOFF: disabled  THRESH-X: L=0 H=0 (latched=0)  STATS: count=0 duration[25ns]
total=0 peak=0
Link  2 XOFF: disabled  THRESH-X: L=0 H=0 (latched=0)  STATS: count=0 duration[25ns]
total=0 peak=0
Link  3 XOFF: disabled  THRESH-X: L=0 H=0 (latched=0)  STATS: count=0 duration[25ns]
total=0 peak=0
Link  4 XOFF: disabled  THRESH-X: L=0 H=0 (latched=0)  STATS: count=0 duration[25ns]
total=0 peak=0
Link  5 XOFF: disabled  THRESH-X: L=0 H=0 (latched=0)  STATS: count=0 duration[25ns]
total=0 peak=0
Link  6 XOFF: disabled  THRESH-X: L=0 H=0 (latched=0)  STATS: count=0 duration[25ns]
total=0 peak=0
Link  7 XOFF: disabled  THRESH-X: L=0 H=0 (latched=0)  STATS: count=0 duration[25ns]
total=0 peak=0
Link  8 XOFF: disabled  THRESH-X: L=0 H=0 (latched=0)  STATS: count=0 duration[25ns]
total=0 peak=0
Link  9 XOFF: disabled  THRESH-X: L=0 H=0 (latched=0)  STATS: count=0 duration[25ns]
total=0 peak=0
Link 10 XOFF: disabled  THRESH-X: L=0 H=0 (latched=0)  STATS: count=0 duration[25ns]
total=0 peak=0
Link 11 XOFF: disabled  THRESH-X: L=0 H=0 (latched=0)  STATS: count=0 duration[25ns]
total=0 peak=0
```

[1] Transmission of XOFF from FULL mode links 12..19 through GBT link 0 has been added since firmware version 4.11

Advanced interface and switch configuration

Specific configurations of the host networking as well as the network switches have to be applied and both have to co-operate in order to work properly. We make use of DSCP (Differentiated Services Code Point), an IPv4 QoS mechanism, and Explicit Congestion Notification (ECN), an extension to IP that allows end-to-end notification of network congestion, to minimize the impact of network issues inherent to Ethernet.

For the configuration of the hosts:

- Enable ECN for all priorities:

```
for i in `seq 0 7`; do
  VAL=`cat /sys/class/net/$IFACE/ecn/roce_np/enable/$i`
  if [ "x$VAL" != "x1" ]; then
    echo 1 > /sys/class/net/$IFACE/ecn/roce_np/enable/$i
  fi

  VAL=`cat /sys/class/net/$IFACE/ecn/roce_rp/enable/$i`
  if [ "x$VAL" != "x1" ]; then
    echo 1 > /sys/class/net/$IFACE/ecn/roce_rp/enable/$i
  fi
done
```

These parameters are not persistent and need to be set after each boot. The openibd service (Mellanox drivers) provides a post-start hook that can be used for this. Create this file with permissions as follows and add the previous commands in it (for each Mellanox cards):

```
$ ls -l /etc/infiniband/post-start-hook.sh
-rwxr-xr-x 1 root root 406 Dec  8 13:47 /etc/infiniband/post-start-hook.sh
```

The recommended network switches are Juniper QFX 5120 or 5200. Please see this Juniper document for minimum software release. However, based on our tests, we recommend Junos OS 20.2R1 for these two platforms. Please refer to the device documentation for generic configuration instructions.

The recommended switch configuration consists of:

- creating a dedicated drop profile:
  We aim for "continous" congestion notifications as this seemed to work better with shared-buffer configuration.

```
drop-profiles {
    dp1 {
        interpolate {
```

```
            fill-level [ 1 90 ];
            drop-probability [ 0 5 ];
        }
    }
}
```

- partitioning the shared ingress buffer as follows:
  40% is assigned to lossless (RoCE) traffic
  55% provides extra space for lossless traffic when congestion are signaled
  5% is left to other, lossy traffic

```
shared-buffer {
    ingress {
        buffer-partition lossless {
            percent 40;
        }
        buffer-partition lossy {
            percent 5;
        }
        buffer-partition lossless-headroom {
            percent 55;
        }
    }
}
```

- defining a custom traffic class:
  lossless traffic class called "roce" assigned to queue 2

```
forwarding-classes {
    class roce queue-num 2 no-loss;
}
```

- configuring congestion signalling:
  DSCP is configured for Mellanox NICs' default code-points, 110000.
  DSCP and PFC are mutually exclusive (per port) on this platform so PFC is left inactive but visible for the sake of making it clear where to enable it if needed.

```
congestion-notification-profile {
    cnp1 {
        input {
            inactive: ieee-802.1 {
                code-point 110 {
                    pfc;
                }
            }
            dscp {
                code-point 110000 {
```

```
                    pfc;
                }
            }
        }
    }
}
```

- binding everything together:
  Create a scheduler assigning the drop profile to all traffic
  Enable ECN
  Assign this scheduler to roce traffic class
  Apply all of this to all interfaces

```
schedulers {
    s1 {
        drop-profile-map loss-priority any protocol any drop-profile dp1;
        explicit-congestion-notification;
    }
}
scheduler-maps {
    sm1 {
        forwarding-class roce scheduler s1;
    }
}
interfaces {
    et-0/0/* {
        congestion-notification-profile cnp1;
        scheduler-map sm1;
        unit 0 {
            forwarding-class roce;
        }
    }
}
```

References

- [4] GBT Module for the FELIX Project, url: https://twiki.cern.ch/twiki/pub/Atlas/GBT2LAN/FELIX_GBT_MANUAL.pdf.
- [5] lpGBT user manual, url: https://cds.cern.ch/record/2809058/files/lpGBT_manual.pdf.
- [6] ATLAS Felix Group, Specifications for the FELIX FULL mode link, url: https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/docs/FullMode.pdf.
- [7] Xilinx, Xilinx VC709 Development Kit, url: http://www.xilinx.com/products/boards-and-kits/dk-v7-vc709-g.html.
- [8] ATLAS FELIX Group, BNL-711 v2 Manual, url: https://atlas-project-felix.web.cern.ch/atlas-project-felix/user/docs/BNL-711_V2P0_manual.pdf.
- [9] Supermicro, Supermicro X10SRA-F Motherboard Model Specification, 2016, url: http://www.supermicro.nl/products/motherboard/Xeon/C600/X10SRA-F.cfm.
- [10] CERN TTC FMC project, url: http://www.ohwr.org/projects/optical-cdr-fmc/wiki.
- [11] TTC group, CERN TTC homepage, url: http://ttc.web.cern.ch/TTC.
- [12] Analog Devices Inc., ADN2814: Continuous Rate 10Mb/s to 675Mb/s Clock and Data Recovery IC with Integrated Limiting Amp, url: http://www.analog.com/static/imported-files/data_sheets/ADN2814.pdf.
- [13] Silicon Labs Inc., Si5345/44/42 Rev D Data Sheet - 10-Channel, Any-Frequency, Any-Output Jitter Attenuator/ Clock Multiplier, url: http://www.silabs.com/SupportDocuments/TechnicalDocs/Si5345-44-42-D-DataSheet.pdf.
- [14] Silicon Labs Inc., Si5324 Data Sheet - Any-Frequency, Any-Output Precision Clock Multiplier / Jitter Attenuator, url: https://www.silabs.com/documents/public/data-sheets/Si5324.pdf.
- [15] Xilinx, Xilinx Vivado Design Suite, 2016, url: https://www.xilinx.com/products/design-tools/vivado.html.
- [16] Linear Technology, LTC2991 Data Sheet - Octal I2C Voltage, Current, and Temperature Monitor, url: http://cds.linear.com/docs/en/datasheet/2991ff.pdf.
- [17] The Versatile Link Developers, The Versatile Link Common Project, 2008, url: https://espace.cern.ch/project-versatile-link/public/default.aspx.
- [18] CERN GBT-FPGA project, url: https://espace.cern.ch/GBT-Project/GBT-FPGA/default.aspx.
- [19] E-link Wrapper Deployment User Guide, url: https://espace.cern.ch/ATLAS-NSW-ELX/Shared%20Documents/Overview%20and%20General/elink_wrapper_userGuide.pdf.
- [20] SCA eXtension User Guide, url: https://espace.cern.ch/ATLAS-NSW-ELX/Shared%20Documents/NSW%20Trigger%20Processor/scax_userGuide.pdf.
- [21] SCA eXtension: a Design for FPGA Parameter Configuration within the ATLAS DAQ Scheme - IEEE/NSS Proceeding, url: https://ieeexplore.ieee.org/document/9059894.
- [22] IC-over-NetIO Gitlab Repository, url: https://gitlab.cern.ch/cbakalis/ic-over-netio