pALPIDEfs software - Installation and command line interface

Markus Keil

rev. 1, July 3, 2014

This manual is intended as a quick start guide to get the pALPIDE software installed and perform the most important tests via the command line interface. It does not (yet) contain a detailed overview of the software structure or instructions for more specialised tests that require code changes. Nonetheless the tests described here should be sufficient to perform a complete test of the chip functionality.

1 Installation

The software is available on a git repository:

`https://git.cern.ch/web/pALPIDEfs-software.git`

Versions are updated regularly, make sure you have checked out the latest version. The repository contains four subdirectories:

- pALPIDEfs-software: The main software package with the low-level driver as well as test routines.
- crystalball: a lean standalone tool to configure the chip and read events.
- FPGA: a script to download the firmware to the FPGA.
- FX3: the configuration file for the FX3 chip as well as a tool to download it to the chip.

In order to compile the software you need to have libusb installed. If not yet installed, download and install version 1.0 from `http://www.libusb.org`. Once this is done you should be able to compile both the software (execute `make` in the directory `pALPIDEfs-software`) and the tool to configure the FX3 chip (execute `build_mac.sh` or `build_linux.sh`, resp., in the directory `fx3`).

If compilation fails this is most likely due to the installation path of the libusb package. In that case you need to locate the path of the header and library file on your system and modify the makefile / scripts accordingly.

2 Getting Started

After installation of the software you should be able to start testing with two simple steps:

1. Configure FX3 chip: This needs to be done after each power-cycle of the board and each reconfiguration of the FPGA. In order to download the configuration file to the FX3 chip you should go to the directory `fx3` and execute

   ```
   ./download_fx3 -t RAM -i SlaveFifoSync.img
   ```

   Upon successful execution you should see the message

   ```
   FX3 firmware programming to RAM completed
   ```

2. Start the program: The test program is started by executing `runTest` in the directory `pALPIDEfs-software`. The executed test is determined by the command line parameters passed to the program (see below).

3 Tests

The set of tests described in the following are a very first attempt to provide a comprehensive set of tests to qualify the pALPIDEfs-chip without having to dig deep into the software. The set of tests and even input parameters and output data format are therefore bound to change in the following weeks. Therefore please check for a newest version of the manual and the software frequently.

A list of the available tests and their syntax can be obtained from the command line by executing the program without any parameters (`./runTest`).

Each test is preceded by a powering-on and configuration of the chip. After each of the two steps the current consumptions (and the NTC temperature) are measured and printed on screen. After each test the chip is powered down (Note that this is not the case if the program crashes / is interrupted).

For tests that write output data you have to create a subdirectory `Data` under the working directory. All data files will be written there.

3.1 FIFO Test

The FIFO test is a quick test to check the JTAG communication with the chip. It writes three different bit patterns (`0x0000, 0xffff` and `0x5555`) into each cell of the end-of-column FIFOs, reads them back and checks the correctness of the readback value. The test is started by passing the parameter `FIFO` to the program:

```
./runTest FIFO
```

3.2 On-chip DAC Test

The output of the on-chip DACs can be connected to monitoring pins of the pALPIDE chip and measured by ADCs on the DAQ board. The `READDAC` test loops over all chip DACs, measures their output once and prints the measured values to screen:

```
./runTest READDACS
```

In order to measure the full DAC characteristics the `SCANDACS` test can be used:

```
./runTest SCANDACS
```

For each DAC it loops over the values from 0 to 255 and measures the output values. The measured values are written into a file for each DAC.

3.3 Digital Scan

The digital scan generates a digital pulse in a number of pixels and reads the hits out. It is started with two parameters

```
./runTest SCANDIGITAL PAR1 PAR2
```

where `PAR1` is the number of injections per pixel, `PAR2` the number of mask stages. E.g. `./runTest DIGITAL 50 160` will test 1% of the pixels (cf. box), doing 50 digital injections into each.

The output data is written into a file `DigitalScan.dat`, each line has the format

```
Doublecol Address NHits
```

with `Doublecol` ranging from 0 to 511, `Address` from 0 to 1023 (Address is the address as described in the pALPIDEfs manual, not the row number).

General remarks on scans:

- Mask stages: All injection-based scans (digital, analogue and threshold) work on a certain number of pixels at a time. In the current implementation this is one pixel in each of the 32 regions, starting from address 0 in the first double column of each region. After the required number of injections has been done the scan moves to the next set of pixels. In order to scan the entire chip the mask has to be staged 1024 * 16 times, 164 mask stages correspond to approximately 1% of the chip.
- Output files: due to the large amount of data in particular for threshold scans, output data is written only for pixels with > 0 hits.

3.4 Analogue Scan

The analogue scan works similar to the digital scan, however instead of generating a digital pulse after the discriminator, a programmable charge is injected into the preamplifier. The scan therefore requires an additional parameter:

```
./runTest SCANANALOGUE PAR1 PAR2 PAR3
```

with `PAR1` being the charge in DAC units[1], `PAR2` the number of injections per pixel and `PAR3` the number of mask stages. The output file format is identical to the one of the digital scan (filename `AnalogueScan.dat`).

3.5 Threshold Scan

The threshold scan performs analogue injections, looping over the charge. For each charge point 50 injections are performed. The command is

```
./runTest THRESHOLD PAR1 PAR2 PAR3
```

with the number of mask stages `PAR1` and the charge loop ranging from `PAR2` to `PAR3` (both in DAC units). The output file `ThresholdScan.dat` contains the raw data, i.e. the number of hits for each charge point, in the format

```
Doublecol Address Charge NHits
```

3.6 Noise Occupancy

The scan gives a selectable number of random triggers and returns the number of hits. The command is

```
./runTest NOISEOCC PAR1
```

with the only parameter being the number of triggers. The hitmap is written into a file `NoiseOccupancy.dat` in a format identical to digital and analogue scan.

3.7 Source Scan

The source scan option does the same as the noise occupancy measurement, but with a longer STROBEB to increase the probability to see source hits with random trigger. The command is

```
./runTest SOURCE PAR1 [PAR2]
```

The first parameter is the number of events, the second an optional mask file name. Hits are written into a file `SourceScan.dat` (same format as above).

[1] Preliminary calibration: 7 electrons / DAC unit

3.8 Noise Mask

This scan prepares a noise mask that can be used in the source scan. The scan is started by the command

```
./runTest NOISEMASK PAR1 PAR2
```

The scan will issue `PAR1` triggers and write all pixels that had one or more hits into an output file with name `PAR2`. This file can directly be used as `PAR2` of the source scan.