apiGedek

Contents

1 Introduction

This page give the API to talk to the GEDEK card using UDP sockets :

- port **1024**: DAQ or input from the GEDEK card
- port **1040**: SLC or order to the GEDEK card
- port **1056**: internal GEDEK register bank
- port **1072**: NeCTAr slow control chip

remarque:

- Convertir un hexadécimal en décimal en ligne de commande

```
$ printf "%i\n" 0x0400
$ echo "ibase=16; obase=A; 400" | bc
1024
```

- Convertir un décimal en hexadécimal en ligne de commande

```
$ printf "%x\n" 1040
$ echo "ibase=10; obase=16; 1040" | bc
410
```

2 Serializer

7 blocks are used to configure the slow control of the board. All the following words are 32 bits.

2.1 CntrlSlc (ok)

CntrlSlc is used to program the threshold for the pixel, the pixel sum and the common mode voltage. It is 24 bytes long.

```
CntrlSlc    = AAAA-AAAA
              0000-7E0C
              Data
              AAAA-AAAA

Data        = Thrshld1
              Thrshld2
              VMC

Thrshld1    = 0000-00XX (8 bits)
Thrshld2    = 0000-00YY (8 bits)
VMC         = 0000-0ZZZ (12 bits)
```

Here is a test file: cntrlSlc.bin.

- Thrshld1 = 170
- Thrshld2 = 85
- VMC = 4095

2.2 CntrlNectarReg

CntrlNectarReg is used to program the internal SAM registers. It is 32 bytes long.

```
CntrlNectarReg = AAAA-AAAA
                 0000-7E3E
                 Data
                 AAAA-AAAA

Data           = CDR1
                 CDR2
                 TestFCR
                 CDR6
                 CDR7

CDR1           = XXXX-XXXX
CDR2           = XXXX-XXXX
TestFCR        = XXXX-XXXX
CDR6           = XXXX-XXXX
CDR7           = XXXX-XXXX
```

Here is a test file: cntrlNectarReg.bin.

- CDR1 = 1111-1111
- CDR2 = 2222-2222
- TestFCR = FFFF-FFFF
- CDR6 = 6666-6666
- CDR7 = 7777-7777

2.3 CntrlNectarDac

CntrlNectarDac is used to program the 32 internal SAM DAC's. It is 80 bytes long.

```
CntrlNectarDac = AAAA-AAAA
                 0000-7E3A
                 Data
                 AAAA-AAAA

Data           = MemNum
                 DACL{16}

MemNum         = 0000-000X [0:15]
DACLx          = XXXX-XXXX
```

Here is a test file: cntrlNectarDac.bin.

- MemNum = 2
- DACLx = xxxx-xxxx

2.4 CntrlNectarNd

CntrlNectarNd is used to define the delay pointer **Nd**. It is 16 bytes long.

```
CntrlNectarNd = AAAA-AAAA
                0000-7E3C
                Nd
                AAAA-AAAA

Nd            = 0000-YXXX
Y             = NeCTAr number (F=broadcast)
 XXX          = NdValue [0:1023]
```

Here is a test file: cntrlNectarNd.bin.

- Y = F
- Nd = 9

2.5 CntrlReadBack

CntrlReadBack is 16 bytes long.

```
CNTRLRealBack = 0000-AAAA
                0000-7E40
                0000-0Y00
                0000-AAAA

Y           = 1: DAC, 2: Nd, 3: NeCTAr chip reg
```

Here is a test file: cntrlReadBack.bin.

- Y = 1

2.6 CntrlIntReg

CntrlIntReg is used to program an read back the internal GEDEK core registers. In case of read back, the message does not include the data. It is 16 or 36 bytes long.

```
CNTRLIntReg = AAAA-AAAA
              0000-7E50
              0000-000Y
              Data{0,1}
              AAAA-AAAA

Y           = 1: write the following registers, 0: read back
Data        = XXXX-XXXX  FPGA Board MAC Address (32 LSB)
            = XXXX-XXXX  FPGA Board IP Address
            = XXXX-XXXX  Destination MAC Address (32 LSB)
            = 0000-XXXX  Destination MAC Address (16 MSB)
            = XXXX-XXXX  Destination IP Address
```

Here are 2 test files for test:

- cntrlIntReg0.bin :
  - Y = 0
- cntrlIntReg1.bin :
  - Y = 1
  - FPGA MAC 32LSB = 75:d6:34:3f
  - FPGA IP = 127.0.0.1
  - Destination MAC 32 LSB = 75:d6:34:3f
  - Destination MAC 16 MSB = 00:04
  - Destination IP = 127.0.0.1

2.7 CntrlDAQ

CntrlDAQ is used to define the DAQ parameters. It is 16 bytes long.

```
CntrlDAQ    = AAAA-AAAA
              0000-7E30
              Data
              AAAA-AAAA

Data        = 0000-XXXY
XXX         = Nf
   Y        = unused QSampls T0 ToT

Nf          = "valeure Nf codée sur 10 bits" [0:1023]
unused      = 1 bit (inutilisé)
QSampls     = 1 bit (0:Sample 1:Charge)
T0          = 1 bit
ToT         = 1 bit
```

T0 et Tot: demande d'ajouter l'info T0 aux blocs DAQ. Ie:

- une fenêtre est composée de Nf samples
- T0 donne le numéreau de sample du maximum du signal dans la fenêtre

- ToT donne le nombre de samples au dessus du seuil (threshold) dans la fenêtre

Here is a test file: cntrlDaq.bin.

- Nf = 1023
- Qsampls = 1
- T0 = 1
- ToT = 1

3 Parser

The following blocks are used to receive data in either charge mode or sampling mode. In order to get a correct *IPNum* word, it is mandatory to launch at least one time the **CntrlIntReg Y=0** block to read the internal registers.

3.1 DAQ IntReg

DAQIntReg is used to read back the internal GEDEK core registers. It is 32 bytes long.

```
DAQIntReg   = BBBB-BBBB
              XXXX-XXXX  FPGA Board MAC Address (32 LSB)
              XXXX-XXXX  FPGA Board IP Address
              XXXX-XXXX  Destination MAC Address (32 LSB)
              0000-XXXX  Destination MAC Address (16 MSB)
              XXXX-XXXX  Destination IP Address
              0000-000X  Host detected status [0:1]
              BBBB-BBBB
```

Here is a test file: daqIntReg.bin.

- FPGA MAC 32LSB = 75:d6:34:3f
- FPGA IP = 127.0.0.1
- Destination MAC 32 LSB = 75:d6:34:3f
- Destination MAC 16 MSB = 00:04
- Destination IP = 127.0.0.1
- Status = 1

3.2 DAQ charge

Data in charge mode is 48 bytes long:

```
DAQCharge   = AAAA-AAAA
              0000-EEE0
              FPGA's IPNum
              EvtCounter
              DataBlock
              AAAA-AAAA

FPGA IPNum  = XXXX-XXXX
EvtCounter  = 0000-XXXX
DataBlock   = PM{7}
PM          = Ch_Charges
```

```
Ch_Charges  = YYYY-ZZZZ

YYYY        = Data2
     ZZZZ   = Data1
```

Here is a test file: daqCharge.bin.

- FPGA IPNum = 127.0.0.1
- EvtCounter = 1
- DataBlocks = 0X0Y-0X0Y ({PM}{Channel}):
  `0000-0001 0100-0101 0200-0201...`

3.3 DAQ sample

Data in sampling mode is (5 + 14*Nf)*4 bytes long:

```
DAQSample   = 0000-AAAA
              0000-EEE3
              FPGA's IPNum
              EvtCounter
              DataBlock
              0000-AAAA

FPGA IPNum  = XXXX-XXXX
EvtCounter  = 0000-XXXX
DataBlocks  = PM{7}
PM          = Ch_Sample{2}
Ch_Sample   = Data{Nf}
Data        = 0000-XXXX
```

Here is a test file: daqSample.bin.

- FPGA IPNum = 127.0.0.1
- EvtCounter = 1
- NF = 2
- Data = 0000-XYZZ ({PM}{Channel}{NF})