Slow Control for EMC Proto192 with EPICS

Florian Feldbauer

Experimentelle Hadronenphysik
Ruhr-Universität Bochum

XXXII. $\overline{\text{P}}$ANDA Collaboration Meeting
March 9th, 2010

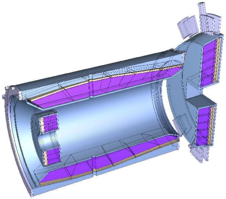
RUHR
UNIVERSITÄT
BOCHUM
RUB

Outline

1. Introduction
    - $\overline{\text{P}}$ANDA Electromagnetic Calorimeter and Proto192
2. Devices
    - Slow Control for Proto192
    - I-7565 USB/CAN Converter
    - THM$\overline{\text{P}}$
    - Further Devices
3. EPICS Applications
    - Overview
    - PV Naming
    - StreamDevice Module and Asyn
    - Channel Archiver
4. First Experiences using EPICS
    - Irradiation Tests of Electronics
5. Conlusion and Outlook

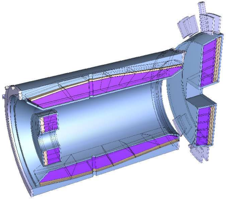
$\overline{\text{P}}$ANDA Electromagnetic Calorimeter and Proto192

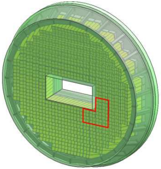
- Electromagnetic calorimeter (EMC) of the $\overline{\text{P}}$ANDA target spectrometer consists of ~ 16000 PWO crystals
- Designed as barrel with 2 endcaps
- Cooled down to $-25\,^\circ\text{C}$ to increase light yield of PWO by factor 4



$\overline{\mathrm{P}}$ANDA Electromagnetic Calorimeter and Proto192

- Electromagnetic calorimeter (EMC) of the $\overline{\mathrm{P}}$ANDA target spectrometer consists of $\sim$ 16000 PWO crystals
- Designed as barrel with 2 endcaps
- Cooled down to $-25\,^\circ$C to increase light yield of PWO by factor 4
- Proto192:
    - Prototype of the forward endcap of the EMC consisting of 192 PWO crystals
    - Allows tests of mounting, cooling, read-out electronics and slow control





Slow Control for Proto192

- Monitoring temperature and humidity
  Temperature and Humidity Monitoring Board for $\overline{\text{P}}$ANDA (THM$\overline{\text{P}}$)
  custom hardware with CAN interface

Slow Control for Proto192

- Monitoring temperature and humidity
  Temperature and Humidity Monitoring Board for $\overline{\text{P}}$ANDA (THM$\overline{\text{P}}$)
  custom hardware with CAN interface
- Controlling LED pulser for monitoring radiation damages and transmission of the PWO crystals
  custom hardware with CAN interface (not yet implemented)

Slow Control for Proto192

- Monitoring temperature and humidity
  Temperature and Humidity Monitoring Board for $\overline{\text{P}}$ANDA (THM$\overline{\text{P}}$)
  custom hardware with CAN interface
- Controlling LED pulser for monitoring radiation damages and transmission of the PWO crystals
  custom hardware with CAN interface (not yet implemented)
- Monitoring and setting power supplies
  - Photodetectors: power supply by ISEG with CAN interface
  - LED pulser: ISEG NHQ202M with RS232C interface
- Monitoring e.g. temperature and fan speed of VME crates by Wiener via CAN bus

Slow Control for Proto192

- Monitoring temperature and humidity
  Temperature and Humidity Monitoring Board for $\overline{\text{P}}$ANDA (THM$\overline{\text{P}}$)
  custom hardware with CAN interface
- Controlling LED pulser for monitoring radiation damages and transmission of the PWO crystals
  custom hardware with CAN interface (not yet implemented)
- Monitoring and setting power supplies
  - Photodetectors: power supply by ISEG with CAN interface
  - LED pulser: ISEG NHQ202M with RS232C interface
- Monitoring e.g. temperature and fan speed of VME crates by Wiener via CAN bus
- Using the I-7565 USB/CAN Converter

I-7565 USB/CAN Converter

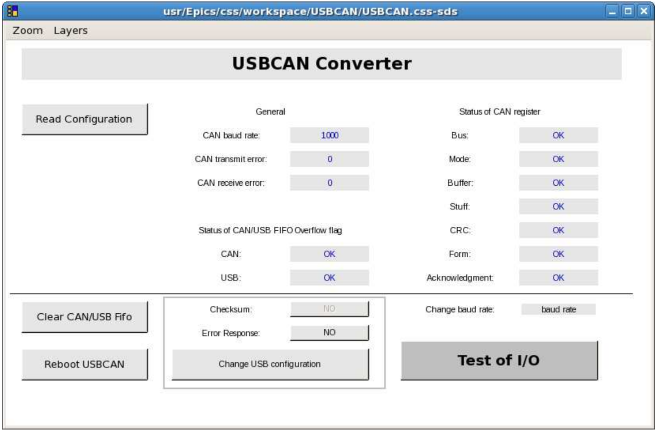
- Allows testing, readout, setting and controlling of all devices connected to CAN bus
- Supports CAN 2.0A and 2.0B
- Transfer rate of up to 1 Mbps for CAN and 921.6 kbps for USB (USB baudrate is fixed)

I-7565 USB/CAN Converter

- Allows testing, readout, setting and controlling of all devices connected to CAN bus
- Supports CAN 2.0A and 2.0B
- Transfer rate of up to 1 Mbps for CAN and 921.6 kbps for USB (USB baudrate is fixed)
- Syntax for sending and receiving a standard CAN data frame: tIIILDD. . .
  - t → Represent a standard data frame
  - III → 11 bit identifier (000 - 7FF)
  - L → Data length (0 - 8)
  - DD. . . → Input data frame value
- Uses ttyUSB as interface

I-7565 USB/CAN Converter

GUI build in CSS (DESY version)



USBCAN Converter

General

CAN baud rate:	1000
CAN transmit error:	0
CAN receive error:	0

Status of CAN/USB FIFO Overflow flag

CAN:	OK
USB:	OK

Status of CAN register

Bus:	OK
Mode:	OK
Buffer:	OK
Stuff:	OK
CRC:	OK
Form:	OK
Acknowledgment:	OK

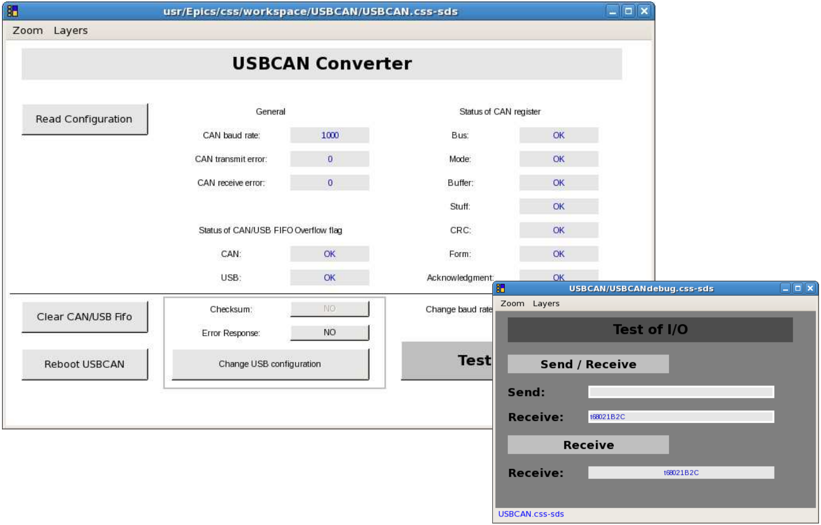
Checksum:	NO
Error Response:	NO

Change baud rate: baud rate

I-7565 USB/CAN Converter

RUB

GUI build in CSS (DESY version)



General

CAN baud rate: 1000

CAN transmit error: 0

CAN receive error: 0

Status of CAN/USB FIFO Overflow flag

CAN: OK

USB: OK

Status of CAN register

Bus: OK

Mode: OK

Buffer: OK

Stuff: OK

CRC: OK

Form: OK

Acknowledgment: OK

Checksum: NO

Error Response: NO

Send:

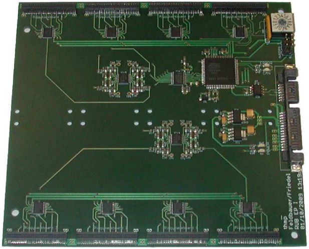
Receive: t68021B2C

Receive: t68021B2C

THM$\overline{\text{P}}$

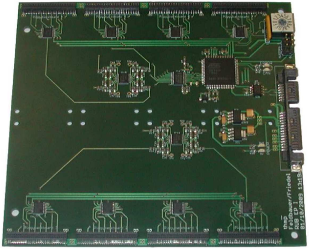
RUB

- THM$\overline{\text{P}}$: Temperature and Humidity Monitoring for $\overline{\text{P}}$ANDA
  Consists of a mainboard and 8 piggyback boards for humidity sensors and temperature sensors, respectively



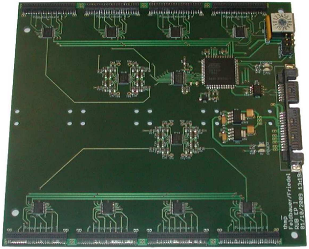
THM$\overline{\text{P}}$

- THM$\overline{\text{P}}$: Temperature and Humidity Monitoring for $\overline{\text{P}}$ANDA
  Consists of a mainboard and 8 piggyback boards for humidity sensors and temperature sensors, respectively
- Read-out via CAN-Interface: CAN ID can be changed with HEX-Code switch (0x680 – 0x68F)



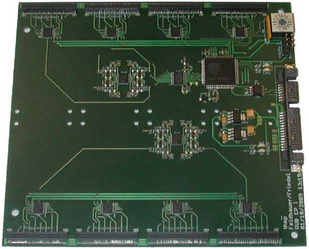
THM$\overline{\text{P}}$

- THM$\overline{\text{P}}$: Temperature and Humidity Monitoring for $\overline{\text{P}}$ANDA
  Consists of a mainboard and 8 piggyback boards for humidity sensors and temperature sensors, respectively
- Read-out via CAN-Interface: CAN ID can be changed with HEX-Code switch (0x680 – 0x68F)
- Send 1 byte data frame to THM$\overline{\text{P}}$ with channel number to read-out the appropriate channel
  THM$\overline{\text{P}}$ answer is 2 byte long (14-bit ADC) – will be extended (e.g. checksum)

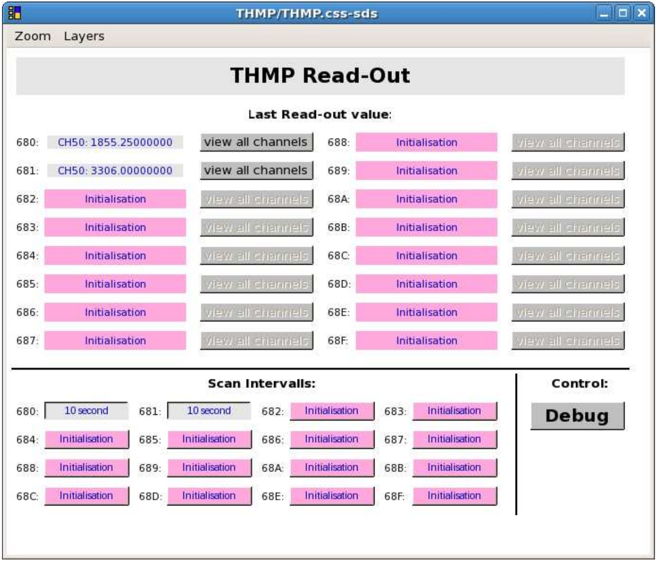


THM$\overline{\text{P}}$

- THM$\overline{\text{P}}$: Temperature and Humidity Monitoring for $\overline{\text{P}}$ANDA
  Consists of a mainboard and 8 piggyback boards for humidity sensors and temperature sensors, respectively
- Read-out via CAN-Interface: CAN ID can be changed with HEX-Code switch (0x680 – 0x68F)
- Send 1 byte data frame to THM$\overline{\text{P}}$ with channel number to read-out the appropriate channel
  THM$\overline{\text{P}}$ answer is 2 byte long (14-bit ADC) – will be extended (e.g. checksum)
- Every channel is read-out by its own record



THMP̄



THMP Read-Out

Last Read-out value:

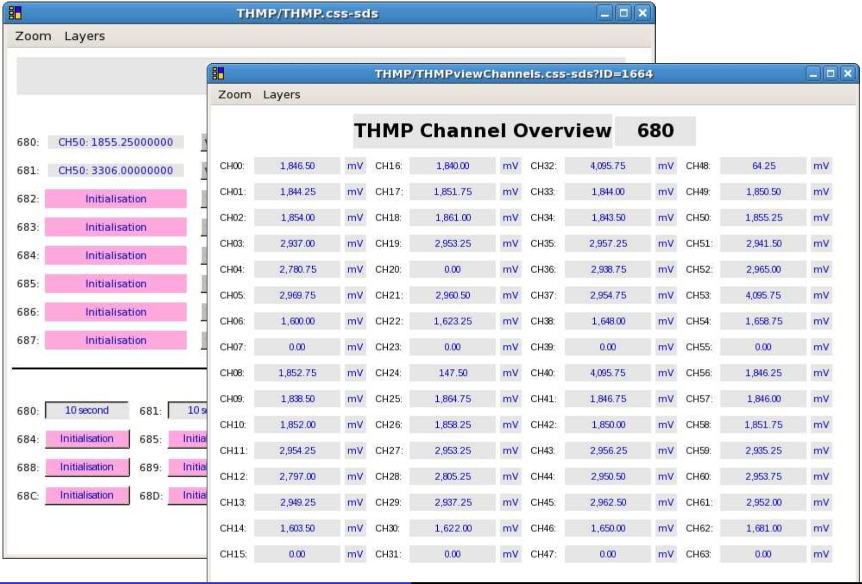
680:	CH50: 1855.25000000	view all channels	688:	Initialisation	view all channels
681:	CH50: 3306.00000000	view all channels	689:	Initialisation	view all channels
682:	Initialisation	view all channels	68A:	Initialisation	view all channels
683:	Initialisation	view all channels	68B:	Initialisation	view all channels
684:	Initialisation	view all channels	68C:	Initialisation	view all channels
685:	Initialisation	view all channels	68D:	Initialisation	view all channels
686:	Initialisation	view all channels	68E:	Initialisation	view all channels
687:	Initialisation	view all channels	68F:	Initialisation	view all channels

Scan Intervalls:

680:	10 second	681:	10 second	682:	Initialisation	683:	Initialisation
684:	Initialisation	685:	Initialisation	686:	Initialisation	687:	Initialisation
688:	Initialisation	689:	Initialisation	68A:	Initialisation	68B:	Initialisation
68C:	Initialisation	68D:	Initialisation	68E:	Initialisation	68F:	Initialisation

Control:

THM$\overline{P}$



680:	CH50: 1855.25000000
681:	CH50: 3306.00000000
682:	Initialisation
683:	Initialisation
684:	Initialisation
685:	Initialisation
686:	Initialisation
687:	Initialisation

680:	10 second	681:	10 s
684:	Initialisation	685:	Initia
688:	Initialisation	689:	Initia
68C:	Initialisation	68D:	Initia

THMP Channel Overview 680

CH00:	1,846.50	mV	CH16:	1,840.00	mV	CH32:	4,095.75	mV	CH48:	64.25	mV
CH01:	1,844.25	mV	CH17:	1,851.75	mV	CH33:	1,844.00	mV	CH49:	1,850.50	mV
CH02:	1,854.00	mV	CH18:	1,861.00	mV	CH34:	1,843.50	mV	CH50:	1,855.25	mV
CH03:	2,937.00	mV	CH19:	2,953.25	mV	CH35:	2,957.25	mV	CH51:	2,941.50	mV
CH04:	2,780.75	mV	CH20:	0.00	mV	CH36:	2,938.75	mV	CH52:	2,965.00	mV
CH05:	2,969.75	mV	CH21:	2,960.50	mV	CH37:	2,954.75	mV	CH53:	4,095.75	mV
CH06:	1,600.00	mV	CH22:	1,623.25	mV	CH38:	1,648.00	mV	CH54:	1,658.75	mV
CH07:	0.00	mV	CH23:	0.00	mV	CH39:	0.00	mV	CH55:	0.00	mV
CH08:	1,852.75	mV	CH24:	147.50	mV	CH40:	4,095.75	mV	CH56:	1,846.25	mV
CH09:	1,838.50	mV	CH25:	1,864.75	mV	CH41:	1,846.75	mV	CH57:	1,846.00	mV
CH10:	1,852.00	mV	CH26:	1,858.25	mV	CH42:	1,850.00	mV	CH58:	1,851.75	mV
CH11:	2,954.25	mV	CH27:	2,953.25	mV	CH43:	2,956.25	mV	CH59:	2,935.25	mV
CH12:	2,797.00	mV	CH28:	2,805.25	mV	CH44:	2,950.50	mV	CH60:	2,953.75	mV
CH13:	2,949.25	mV	CH29:	2,937.25	mV	CH45:	2,962.50	mV	CH61:	2,952.00	mV
CH14:	1,603.50	mV	CH30:	1,622.00	mV	CH46:	1,650.00	mV	CH62:	1,681.00	mV
CH15:	0.00	mV	CH31:	0.00	mV	CH47:	0.00	mV	CH63:	0.00	mV

Further Devices

ISEG NHQ202M Dual HV Power supply

- Dual HV supply in single NIM package
- Regulated 0 to 2 kV DC output, 0 to 6 mA
- Programmable via RS232C interface
- Readout and change of all parameters

Further Devices

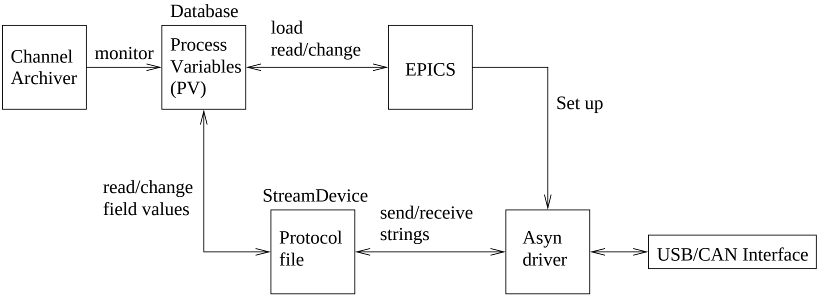
ISEG NHQ202M Dual HV Power supply

- Dual HV supply in single NIM package
- Regulated 0 to 2 kV DC output, 0 to 6 mA
- Programmable via RS232C interface
- Readout and change of all parameters

VME Crate by Wiener

- Monitoring temperatures, voltages, fan speed, status and control of VME Crate via CAN, RS232C or Ethernet is possible
- Currently most important functions controlled and readout with EPICS via CAN

Overview


Channel Archiver
monitor
Database
Process Variables (PV)
load
read/change
EPICS
Set up
read/change field values
StreamDevice
Protocol file
send/receive strings
Asyn driver
USB/CAN Interface

PV Naming

- In PV names variables are used for subsystem, sector, device ID and optionally channel number

PV Naming

- In PV names variables are used for subsystem, sector, device ID and optionally channel number
- Example: THM$\overline{\text{P}}$ readout record
  `PANDA:$(subsys):$(sector):THMP$(ID):SendMsg$(no)`

PV Naming

- In PV names variables are used for subsystem, sector, device ID and optionally channel number
- Example: THM$\overline{\mathrm{P}}$ readout record

  ```
  PANDA:$(subsys):$(sector):THMP$(ID):SendMsg$(no)
  ```
- Assigning values to the variables
  - when loading the database

    ```
    dbLoadRecords("db/dbTHMP.db","subsys=EMC,...")
    ```

PV Naming

- In PV names variables are used for subsystem, sector, device ID and optionally channel number
- Example: THMP̅ readout record
  `PANDA:$(subsys):$(sector):THMP$(ID):SendMsg$(no)`
- Assigning values to the variables
  - when loading the database
    `dbLoadRecords("db/dbTHMP.db","subsys=EMC,...")`
  - or using a database template
    `dbLoadTemplate "db/thmp.substitutions"`

PV Naming

- In PV names variables are used for subsystem, sector, device ID and optionally channel number
- Example: THM$\overline{\text{P}}$ readout record
  `PANDA:$(subsys):$(sector):THMP$(ID):SendMsg$(no)`
- Assigning values to the variables
  - when loading the database
    `dbLoadRecords("db/dbTHMP.db","subsys=EMC,...")`
  - or using a database template
    `dbLoadTemplate "db/thmp.substitutions"`
- ⇒ `PANDA:EMC:PROTO192:THMP1664:SendMsg00`

PV Naming

- Structure of a substitution file:

```
file "db/dbTHMP.db" {
  pattern { subsys, sector, ID, no }
      { "EMC", "PROTO192", 1664, 00 }
      { "EMC", "PROTO192", 1664, 01 }
      ...
  }
```

PV Naming

- Structure of a substitution file:

```
file "db/dbTHMP.db" {
  pattern { subsys, sector, ID, no }
     { "EMC", "PROTO192", 1664, 00 }
     { "EMC", "PROTO192", 1664, 01 }
     ...
  }
```

- Compatible with MSI (Macro Substitution and Include Tool)

PV Naming

- Structure of a substitution file:

```
file "db/dbTHMP.db" {
  pattern { subsys, sector, ID, no }
     { "EMC", "PROTO192", 1664, 00 }
     { "EMC", "PROTO192", 1664, 01 }
     ...
  }
```

- Compatible with MSI (Macro Substitution and Include Tool)
- Using the variables allows you to write one record and load it as often as it is needed.

StreamDevice Module and Asyn

- Generic EPICS module, supports devices controlled by sending and receiving strings via serial port or ethernet
- Strings are defined in protocol files

StreamDevice Module and Asyn

- Generic EPICS module, supports devices controlled by sending and receiving strings via serial port or ethernet
- Strings are defined in protocol files
- Example: THM$\overline{\text{P}}$ readout
- Set Epics environment variable:
  `epicsEnvSet("STREAM_PROTOCOL_PATH","$(TOP)/protocols")`

StreamDevice Module and Asyn

- Generic EPICS module, supports devices controlled by sending and receiving strings via serial port or ethernet
- Strings are defined in protocol files
- Example: THM$\overline{\text{P}}$ readout
- Set Epics environment variable:

```
epicsEnvSet("STREAM_PROTOCOL_PATH","$(TOP)/protocols")
```

- and set up interface (serial port):

```
drvAsynSerialPortConfigure("USBCAN1","/dev/ttyUSB0")
asynSetOption ("USBCAN1", 0, "baud", "921600")
asynSetOption ("USBCAN1", 0, "bits", "8")
asynSetOption ("USBCAN1", 0, "parity", "none")
asynSetOption ("USBCAN1", 0, "stop", "1")
asynSetOption ("USBCAN1", 0, "clocal", "N")
asynSetOption ("USBCAN1", 0, "crtscts", "N")
```

StreamDevice Module

Record functions as output and input:

```
record(scalcout,"PANDA:EMC:$(sector):THMP$(ID):SendMsg$(no)")
{
  field (DTYP, "stream")
  field (INPA, "$(no)")
  field (INPB, "$(ID)")
  field (OUT, "@THMP.proto SendMsg USBCAN1")
}
```

StreamDevice Module

Record functions as output and input:

```
record(scalcout,"PANDA:EMC:$(sector):THMP$(ID):SendMsg$(no)")
{
  field (DTYP, "stream")
  field (INPA, "$(no)")
  field (INPB, "$(ID)")
  field (OUT, "@THMP.proto SendMsg USBCAN1")
}
```

Protocol controls the named interface:

```
SendMsg {
  out "t%(B)3X1%(A)2X";
  in "%*5c%(LL)X";
}
```

Channel Archiver

- Channel Archiver used to store data to hard disk
- Configuration with xml file (Which PV's should be saved, period, etc.)

Channel Archiver

- Channel Archiver used to store data to hard disk
- Configuration with xml file (Which PV's should be saved, period, etc.)
- Channel Archiver can run a data server
  ⇒ Connect to CSS Data Browser

Channel Archiver

RUB

- Channel Archiver used to store data to hard disk
- Configuration with xml file (Which PV's should be saved, period, etc.)
- Channel Archiver can run a data server
  ⇒ Connect to CSS Data Browser
- Using ArchiveDaemon to create daily sub-archives

Channel Archiver

- Channel Archiver used to store data to hard disk
- Configuration with xml file (Which PV's should be saved, period, etc.)
- Channel Archiver can run a data server
  ⇒ Connect to CSS Data Browser
- Using ArchiveDaemon to create daily sub-archives
- In future storing the data as Sql database

Irradiation Tests of Electronics

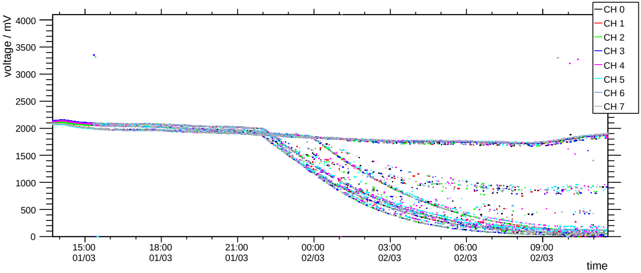
- Tested the radiation hardness of humidity sensors and voltage regulators at the Gießen Irradiation Facility

Irradiation Tests of Electronics

- Tested the radiation hardness of humidity sensors and voltage regulators at the Gießen Irradiation Facility
- Parts were irradiated for 19 hours with a ^{60}Co γ-source at dose rates of 215 Gy/h and 54 Gy/h, respectively
- Used THM$\overline{\text{P}}$ and EPICS for monitoring output voltages

Irradiation Tests of Electronics

- Tested the radiation hardness of humidity sensors and voltage regulators at the Gießen Irradiation Facility
- Parts were irradiated for 19 hours with a ^{60}Co γ-source at dose rates of 215 Gy/h and 54 Gy/h, respectively
- Used THM$\overline{\text{P}}$ and EPICS for monitoring output voltages



See talk "Radiation hardness tests of electronics for the EMC Slow Control" by Patrick Friedel (after coffee break at the EMC session)

Conlusion and Outlook

Conclusion

- Slow Control for Proto192 nearly complete. (HV Power supply, temperature and humidity monitoring)
- Using StreamDevice for easy access to hardware interfaces
- Using ChannelArchiver to store data to hard disk

Conlusion and Outlook

RUB

Conclusion

- Slow Control for Proto192 nearly complete. (HV Power supply, temperature and humidity monitoring)
- Using StreamDevice for easy access to hardware interfaces
- Using ChannelArchiver to store data to hard disk

Outlook

- Writing application for the LED Pulser
- Implement extension for applying calibration data of sensors (e.g. BURT)
- Implementation of database interface