Implementation of ALFRED to control the setup in the FIT lab

Ignacy Mermer

22.08.2024

Project overview

- ALFRED (ALICE Low Level Front End Device) consists of FRED (Front End Device) and ALF (ALICE Low Level Frontend);
- WinCC O.A. - supervisory software for the control centreormachine operation on the user's device;
- DIM (Distributed Information Management) topics for connection between WinCC-FRED and FRED-ALF;
- Dim servers needs DNS, to communicate on the same host;

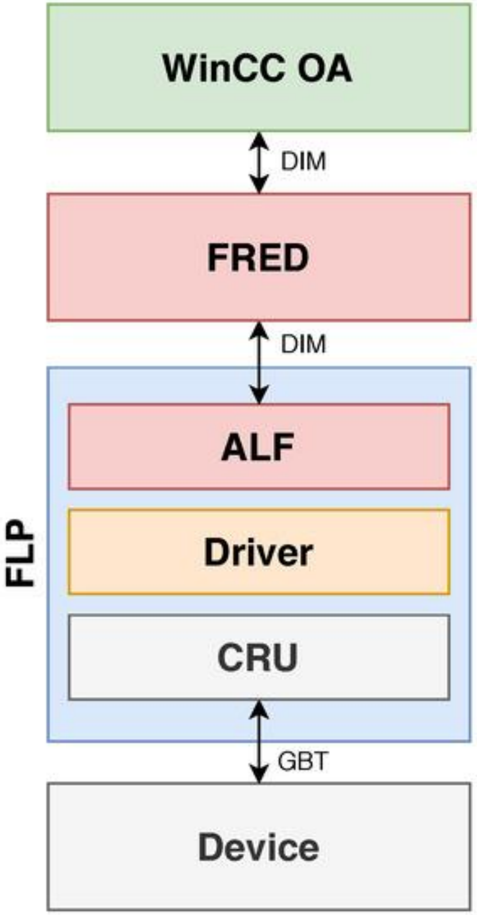


Fig.1 System schema [1]

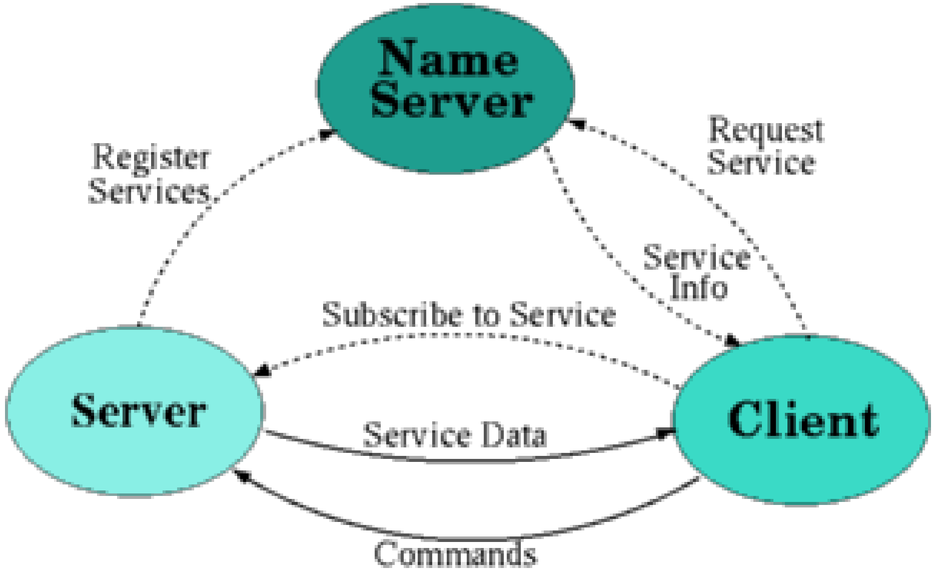


Fig.2 DIM schema [2]

Project establishment

- “Control Server” is communicating by IPbus and run on localhost with WinCC;
- IPbus - Software and firmware that provide control for electronics, by implementing control protocol for reading and modifying resources within FPGA-based hardware devices;
- The final goal is to communicate between ALF-WinCC using DIM and between CRU-FEE using GBT;
- Our actual goal is to connect by IPbus between ALFIPbus and Frontend Electronics (FEE);

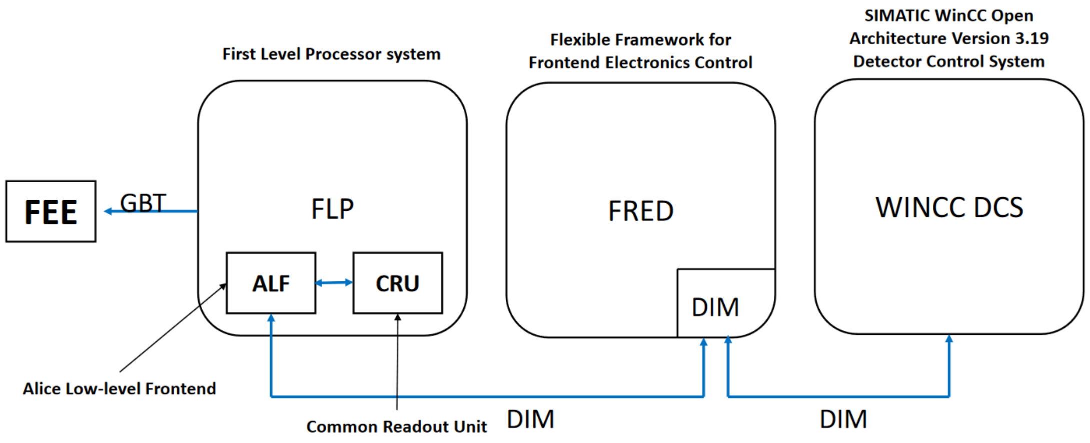

First Level Processor system
Flexible Framework for Frontend Electronics Control
SIMATIC WinCC Open Architecture Version 3.19 Detector Control System
FEE
GBT
FLP
ALF
CRU
FRED
DIM
WINCC DCS
Alice Low-level Frontend
Common Readout Unit
DIM
DIM

Fig.4 Final assumption of the system

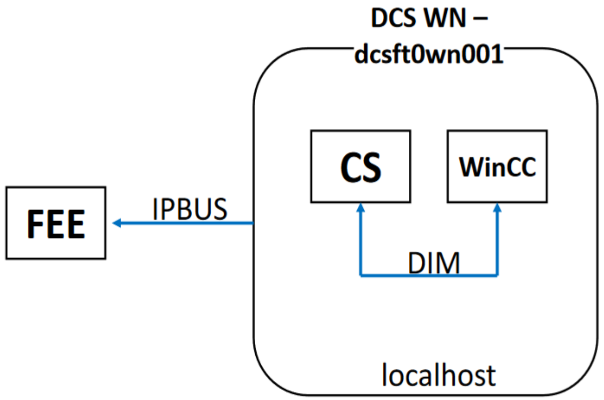


Fig.3 Current system structure

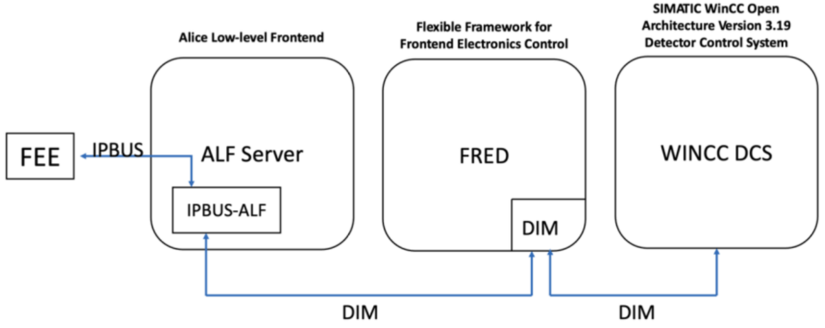
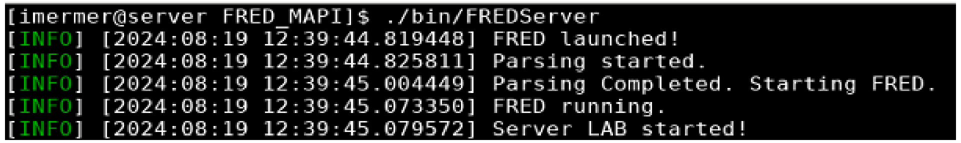

Alice Low-level Frontend
Flexible Framework for Frontend Electronics Control
SIMATIC WinCC Open Architecture Version 3.19 Detector Control System
FEE
IPBUS
ALF Server
IPBUS-ALF
FRED
DIM
WINCC DCS
DIM
DIM

Fig.5 System architecture currently being created

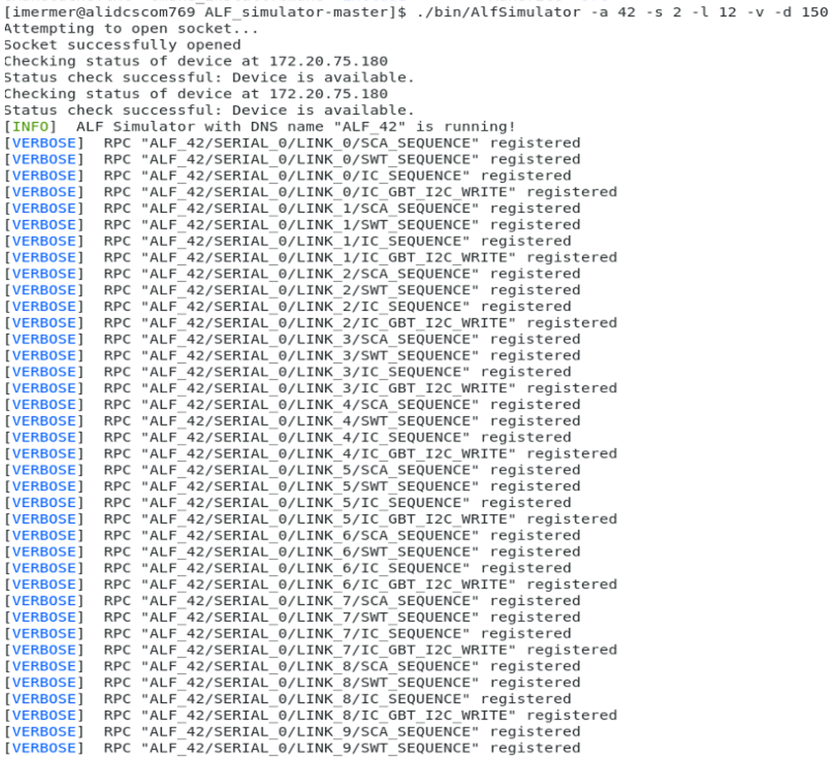
Connection between WinCC and ALFsim

- ALFsim (simulator emulating ALF) connecting to provided IP address based on “Control Server” code;
- Two different computers for FRED and ALFsim and communication between them using DIM RPC;
- Different types of provided protocols to pass data in text between FRED and ALF;



```
[imermer@server FRED_MAPI]$ ./bin/FREDServer
[INFO] [2024:08:19 12:39:44.819448] FRED launched!
[INFO] [2024:08:19 12:39:44.825811] Parsing started.
[INFO] [2024:08:19 12:39:45.004449] Parsing Completed. Starting FRED.
[INFO] [2024:08:19 12:39:45.073350] FRED running.
[INFO] [2024:08:19 12:39:45.079572] Server LAB started!
```

FRED



```
[imermer@alidcscom769 ALF_simulator-master]$ ./bin/AlfSimulator -a 42 -s 2 -l 12 -v -d 150
Attempting to open socket...
Socket successfully opened
Checking status of device at 172.20.75.180
Status check successful: Device is available.
Checking status of device at 172.20.75.180
Status check successful: Device is available.
[INFO]  ALF Simulator with DNS name "ALF_42" is running!
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_0/SCA_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_0/SWT_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_0/IC_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_0/IC_GBT_I2C_WRITE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_1/SCA_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_1/SWT_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_1/IC_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_1/IC_GBT_I2C_WRITE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_2/SCA_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_2/SWT_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_2/IC_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_2/IC_GBT_I2C_WRITE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_3/SCA_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_3/SWT_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_3/IC_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_3/IC_GBT_I2C_WRITE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_4/SCA_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_4/SWT_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_4/IC_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_4/IC_GBT_I2C_WRITE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_5/SCA_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_5/SWT_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_5/IC_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_5/IC_GBT_I2C_WRITE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_6/SCA_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_6/SWT_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_6/IC_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_6/IC_GBT_I2C_WRITE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_7/SCA_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_7/SWT_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_7/IC_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_7/IC_GBT_I2C_WRITE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_8/SCA_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_8/SWT_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_8/IC_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_8/IC_GBT_I2C_WRITE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_9/SCA_SEQUENCE" registered
[VERBOSE]  RPC "ALF_42/SERIAL_0/LINK_9/SWT_SEQUENCE" registered
```

Fig.6 ALFsim with IPbus connecting with electronics

Communication FRED-ALF SWT

- 3 bites for type of operation;
- 32 bites for address;
- 32 bites for data;
- one service on ALF and based on address parameter;

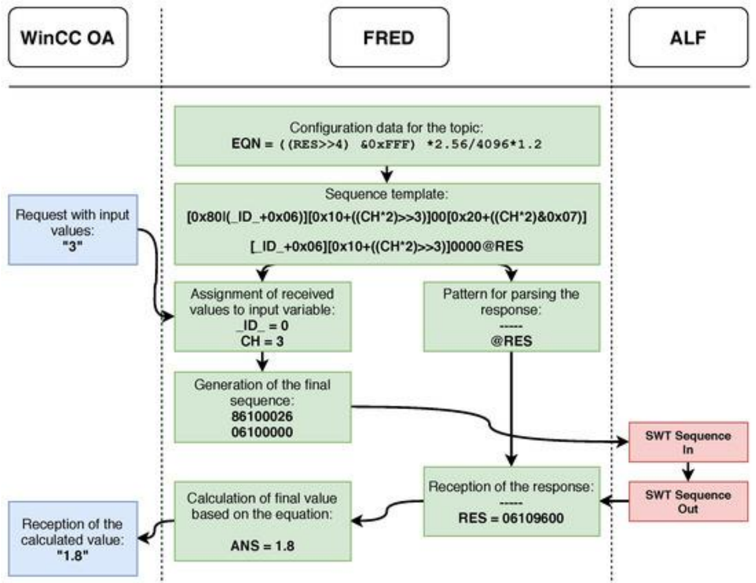

WinCC OA
FRED
ALF
Configuration data for the topic:
EQN = ((RES>>4) &0xFFF) *2.56/4096*1.2
Sequence template:
[0x80I(_ID_+0x06)][0x10+((CH*2)>>3)]00[0x20+((CH*2)&0x07)]
[_ID_+0x06][0x10+((CH*2)>>3)]0000@RES
Request with input values: "3"
Assignment of received values to input variable: _ID_ = 0 CH = 3
Pattern for parsing the response: ----- @RES
Generation of the final sequence: 86100026 06100000
SWT Sequence In
SWT Sequence Out
Reception of the response: ----- RES = 06109600
Calculation of final value based on the equation: ANS = 1.8
Reception of the calculated value: "1.8"

Fig.9 Request pattern in DIM topics [3]

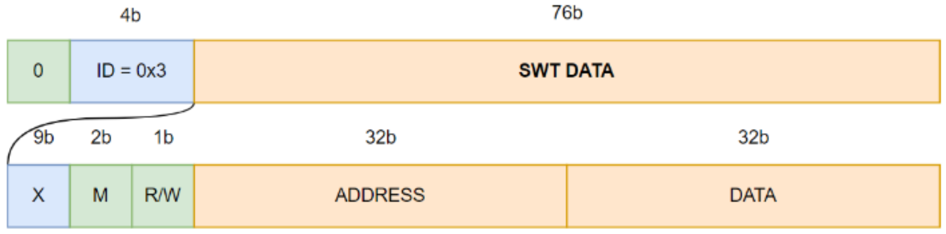
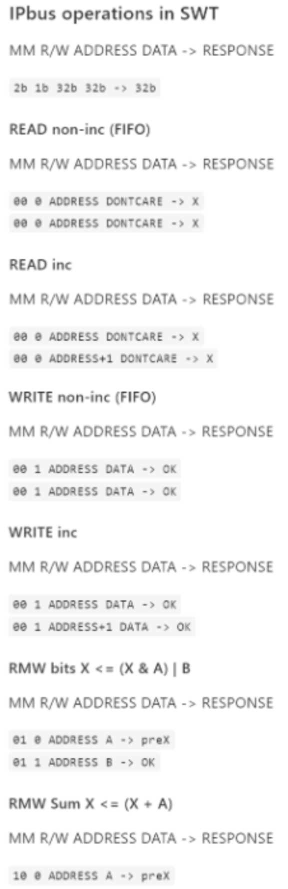


Fig.8 SWT frame structure


IPbus operations in SWT

MM R/W ADDRESS DATA -> RESPONSE

```
2b 1b 32b 32b -> 32b
```

READ non-inc (FIFO)

MM R/W ADDRESS DATA -> RESPONSE

```
00 0 ADDRESS DONTCARE -> X
00 0 ADDRESS DONTCARE -> X
```

READ inc

MM R/W ADDRESS DATA -> RESPONSE

```
00 0 ADDRESS DONTCARE -> X
00 0 ADDRESS+1 DONTCARE -> X
```

WRITE non-inc (FIFO)

MM R/W ADDRESS DATA -> RESPONSE

```
00 1 ADDRESS DATA -> OK
00 1 ADDRESS DATA -> OK
```

WRITE inc

MM R/W ADDRESS DATA -> RESPONSE

```
00 1 ADDRESS DATA -> OK
00 1 ADDRESS+1 DATA -> OK
```

RMW bits X <= (X & A) | B

MM R/W ADDRESS DATA -> RESPONSE

```
01 0 ADDRESS A -> preX
01 1 ADDRESS B -> OK
```

RMW Sum X <= (X + A)

MM R/W ADDRESS DATA -> RESPONSE

```
10 0 ADDRESS A -> preX
```

Fig.7 Types of operations in ALF

FRED MAPI for TCM

- MAPI – Message Application Interface;
- TCM - Trigger and Clock Module (master module in FIT detector's electronics);
- Using SWT (Single Word Transaction) frames in declaration of DIM topics;
- Using only MAPI topics in FRED;
- 4 types of operations: READ, WRITE, RMW AND/OR;

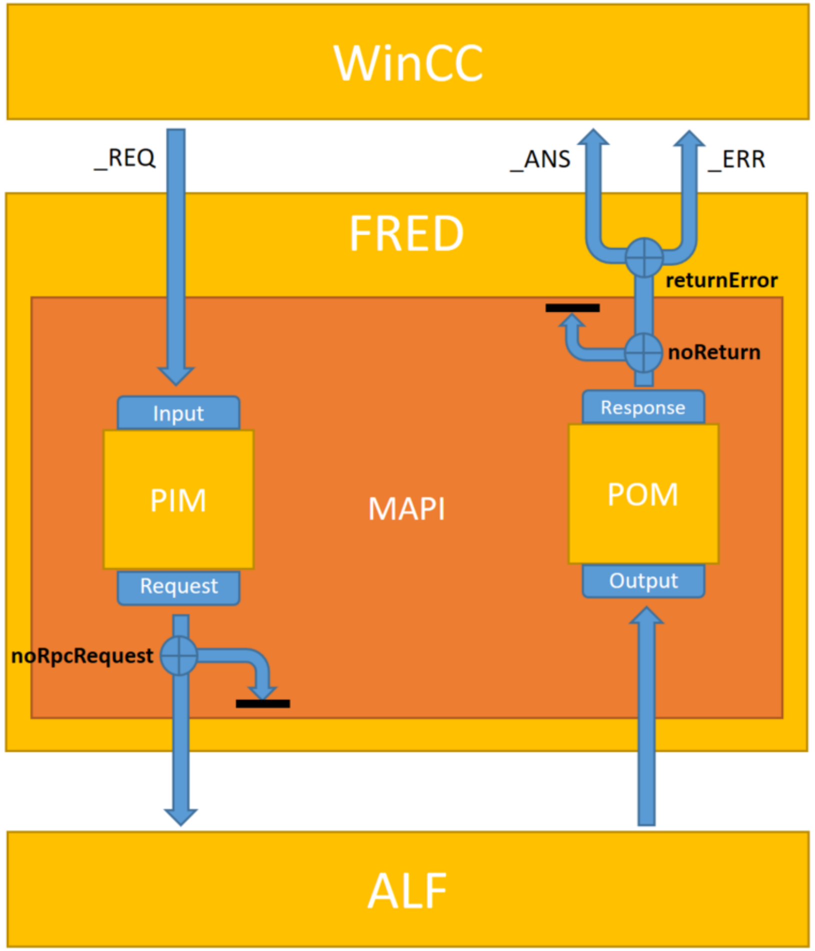
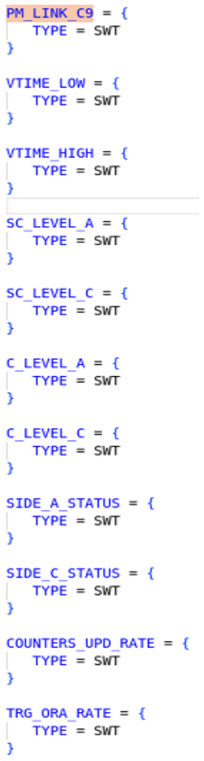


Fig.10 MAPI schema [4]



```
PM_LINK_C9 = {
    TYPE = SWT
}

VTIME_LOW = {
    TYPE = SWT
}

VTIME_HIGH = {
    TYPE = SWT
}

SC_LEVEL_A = {
    TYPE = SWT
}

SC_LEVEL_C = {
    TYPE = SWT
}

C_LEVEL_A = {
    TYPE = SWT
}

C_LEVEL_C = {
    TYPE = SWT
}

SIDE_A_STATUS = {
    TYPE = SWT
}

SIDE_C_STATUS = {
    TYPE = SWT
}

COUNTERS_UPD_RATE = {
    TYPE = SWT
}

TRG_ORA_RATE = {
    TYPE = SWT
}
```

Fig.11 Declaration of DIM topics

Refresh data in WinCC

- One SWT sequence consists of all parameters addresses send to ALF;
- Update only this variables which has changed, calculate output and update service;

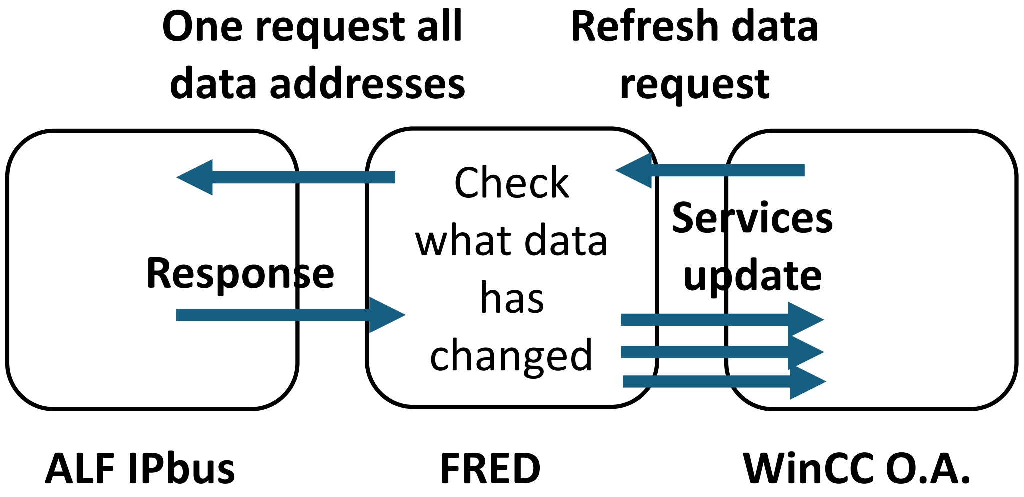
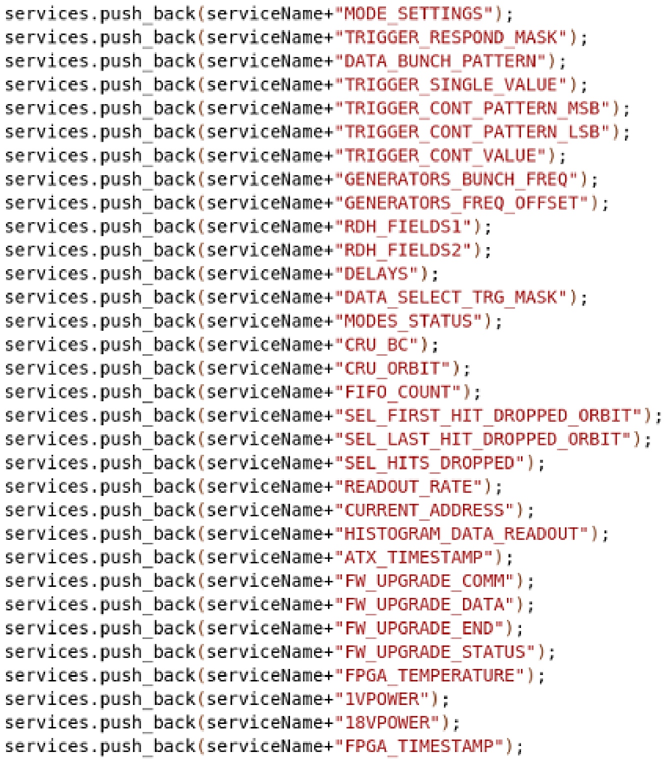


Fig.12 Refresh data schema

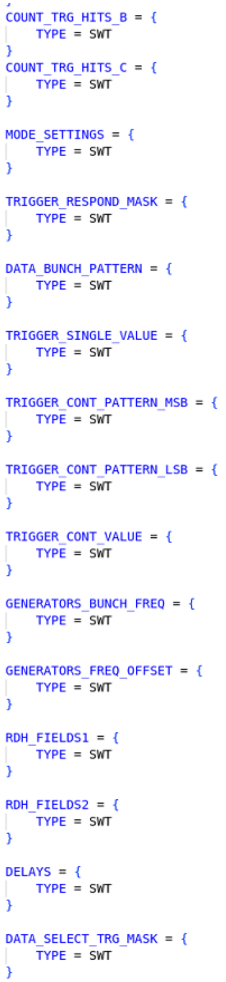


```
services.push_back(serviceName+"MODE_SETTINGS");
services.push_back(serviceName+"TRIGGER_RESPOND_MASK");
services.push_back(serviceName+"DATA_BUNCH_PATTERN");
services.push_back(serviceName+"TRIGGER_SINGLE_VALUE");
services.push_back(serviceName+"TRIGGER_CONT_PATTERN_MSB");
services.push_back(serviceName+"TRIGGER_CONT_PATTERN_LSB");
services.push_back(serviceName+"TRIGGER_CONT_VALUE");
services.push_back(serviceName+"GENERATORS_BUNCH_FREQ");
services.push_back(serviceName+"GENERATORS_FREQ_OFFSET");
services.push_back(serviceName+"RDH_FIELDS1");
services.push_back(serviceName+"RDH_FIELDS2");
services.push_back(serviceName+"DELAYS");
services.push_back(serviceName+"DATA_SELECT_TRG_MASK");
services.push_back(serviceName+"MODES_STATUS");
services.push_back(serviceName+"CRU_BC");
services.push_back(serviceName+"CRU_ORBIT");
services.push_back(serviceName+"FIFO_COUNT");
services.push_back(serviceName+"SEL_FIRST_HIT_DROPPED_ORBIT");
services.push_back(serviceName+"SEL_LAST_HIT_DROPPED_ORBIT");
services.push_back(serviceName+"SEL_HITS_DROPPED");
services.push_back(serviceName+"READOUT_RATE");
services.push_back(serviceName+"CURRENT_ADDRESS");
services.push_back(serviceName+"HISTOGRAM_DATA_READOUT");
services.push_back(serviceName+"ATX_TIMESTAMP");
services.push_back(serviceName+"FW_UPGRADE_COMM");
services.push_back(serviceName+"FW_UPGRADE_DATA");
services.push_back(serviceName+"FW_UPGRADE_END");
services.push_back(serviceName+"FW_UPGRADE_STATUS");
services.push_back(serviceName+"FPGA_TEMPERATURE");
services.push_back(serviceName+"1VPOWER");
services.push_back(serviceName+"18VPOWER");
services.push_back(serviceName+"FPGA_TIMESTAMP");
```

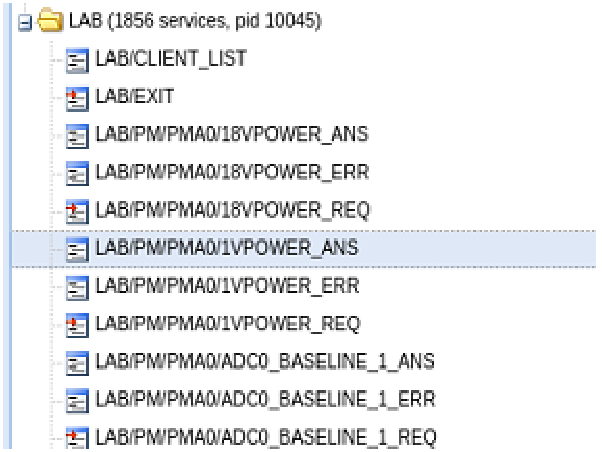
Fig.13 Declaration of services

PMs implementation

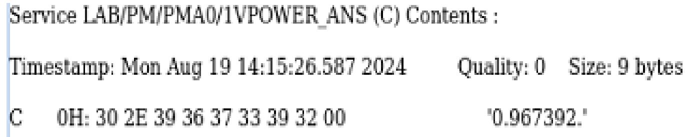
- PM – Processing Module, PM provides 12 independent inputs and each PM is connected to TCM via an HDMI cable;
- Services configured automatically depending on configuration from config file (FRED independent for detector type and Lab);



```
,
COUNT_TRG_HITS_B = {
    TYPE = SWT
}
COUNT_TRG_HITS_C = {
    TYPE = SWT
}

MODE_SETTINGS = {
    TYPE = SWT
}

TRIGGER_RESPOND_MASK = {
    TYPE = SWT
}

DATA_BUNCH_PATTERN = {
    TYPE = SWT
}

TRIGGER_SINGLE_VALUE = {
    TYPE = SWT
}

TRIGGER_CONT_PATTERN_MSB = {
    TYPE = SWT
}

TRIGGER_CONT_PATTERN_LSB = {
    TYPE = SWT
}

TRIGGER_CONT_VALUE = {
    TYPE = SWT
}

GENERATORS_BUNCH_FREQ = {
    TYPE = SWT
}

GENERATORS_FREQ_OFFSET = {
    TYPE = SWT
}

RDH_FIELDS1 = {
    TYPE = SWT
}

RDH_FIELDS2 = {
    TYPE = SWT
}

DELAYS = {
    TYPE = SWT
}

DATA_SELECT_TRG_MASK = {
    TYPE = SWT
}
```



```
LAB (1856 services, pid 10045)
    LAB/CLIENT_LIST
    LAB/EXIT
    LAB/PM/PMA0/18VPOWER_ANS
    LAB/PM/PMA0/18VPOWER_ERR
    LAB/PM/PMA0/18VPOWER_REQ
    LAB/PM/PMA0/1VPOWER_ANS
    LAB/PM/PMA0/1VPOWER_ERR
    LAB/PM/PMA0/1VPOWER_REQ
    LAB/PM/PMA0/ADC0_BASELINE_1_ANS
    LAB/PM/PMA0/ADC0_BASELINE_1_ERR
    LAB/PM/PMA0/ADC0_BASELINE_1_REQ
```



```
Service LAB/PM/PMA0/1VPOWER_ANS (C) Contents :

Timestamp: Mon Aug 19 14:15:26.587 2024      Quality: 0    Size: 9 bytes

C     0H: 30 2E 39 36 37 33 39 32 00                    '0.967392.'
```

Fig.14 Services add to MAPI

Coming days

- Testing FRED and ALFIPbus with unit testing and performance testing for more requests and all PMs running;
- Histogram’s services implementation in FRED;

Thank you for your attention!

Ignacy Mermer

ignacy.mermer.stud@pw.edu.pl

Contact Person:

Krystian Roslon

krystian.roslon@cern.ch

Bibliography

- [1] – “DIM - A distributed information management system for the DELPHI experiment at CERN” - https://inspirehep.net/files/a5d11cd059531d5d3bb91dc93a7cad51 - dim.web.cern.ch

- [2], [3], [4] – “FRED—Flexible Framework for Frontend Electronics Control in ALICE Experiment at CERN” - https://www.mdpi.com/2227-9717/8/5/565