Integration of Programmable Logic Controllers into the FAIR Control System using FESA

R. Haseitl, C. Andre, H. Bräuning, T. Hoffmann, R. Lonsing
GSI Helmholtzzentrum für Schwerionenforschung GmbH, Darmstadt, Germany

PCaPAC 2018 WEPL009

Abstract

For the upcoming 'Facility for Antiproton and Ion Research' (FAIR) at GSI, the Front End Software Architecture (FESA) framework built by CERN has been chosen to serve as front-end level of the future FAIR control system. All beam diagnostic devices of FAIR will be controlled by FESA classes that are addressable by the new control system. The connectivity to the old control system is retained, since both control systems will be in operation contemporaneously for several years. Commercially available Programmable Logic Controllers (PLCs) have been installed as part of Beam Induced Fluorescence (BIF) monitors to replace outdated network attached devices and to improve the reliability of the BIF systems. The new PLC devices are controlled by FESA classes which are addressed from the existing C++ software via Remote Data Access (RDA) calls. This contribution describes the system setup and the involved software components to access the PLC hardware.

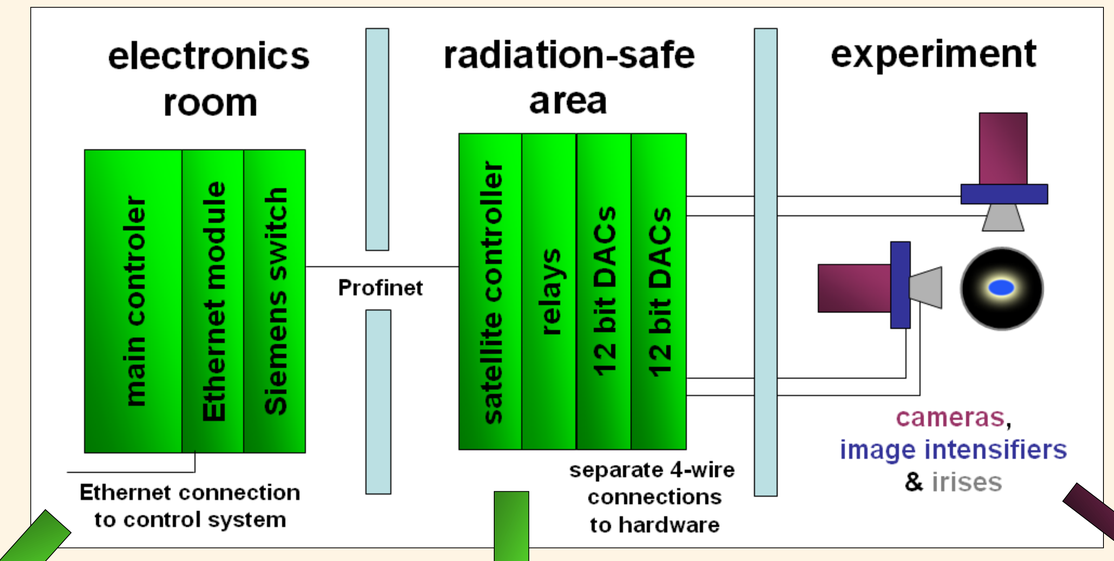
Motivation: Previous Hardware

- based on embedded controller chip
- controllable via Ethernet and web browser
- four outlets for iris and image intensifier control
  0 – 5 V, controlled by 8 bit DACs
- four outlets for calibration LEDs (on/off)

- unstable during long operation
- undefined voltages after crash

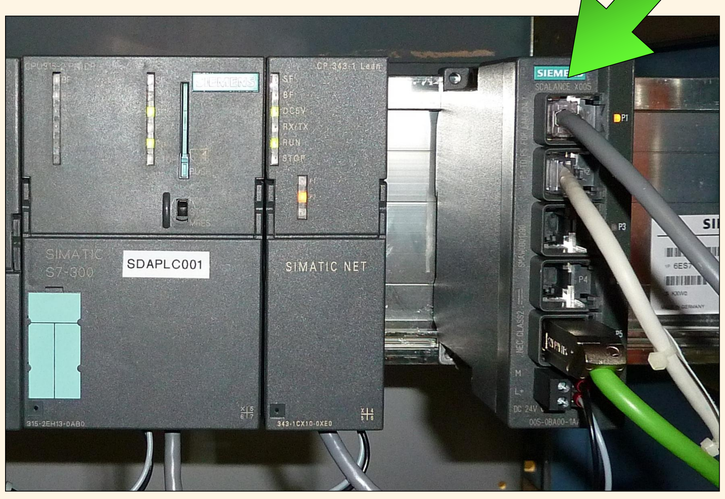
PLC Hardware Setup

- PLC hardware is distributed along the accelerator
- one main controller for multiple distributed subsystems ('satellites')
- connection between main controller and satellites via Profinet
- satellites consist of local controller and relays/DACs
- satellites are located close to experiment to reduce cable length
- experiment hardware connected via 4-wire technique to sense and eliminate conduction losses



Siemens SIMATIC PLC components:
S7-300 main controller
CP343-1 Lean communication module
ET 200M satellite controller
SM322 relay module with eight outlets
SM332 12-bit DAC with four outlets

optical system:
Basler A311f FireWire CCD camera
Proxitronic image intensifier
Pentax B2514ER with remote controllable iris



electronics room installation: main controller, communication module and switch for satellite connection

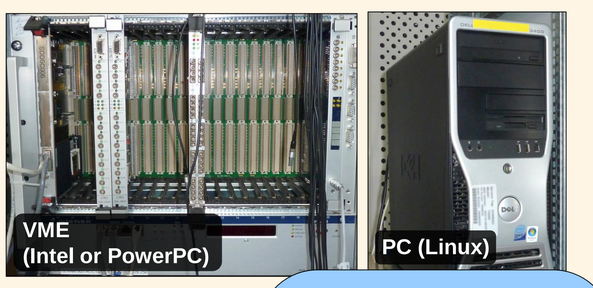
satellite installation in radiation-safe are near experiment to control two BIF monitors: contains two sets of: satellite controller, relay module, DAC module (2x)

at the experiment: FireWire camera with image intensifier and iris

Communication with the PLC

ProfileView

- controls up to three BIF monitors at the same time
- written in C++, running on Linux
- linked with CERN Middleware Library for device access
- sets new values over device handle
- subscribes to data changes of the FESA class → callback function is notified whenever a value is changed from any connected application



FESA class

running on VME CPU or Linux PC

data transfer to/from PLC Ethernet module

notifies all subscribed client applications on data change

set new values

notify on value change

ProfileView Subscription Callback

gets notified on data changes in the FESA class

separate thread to avoid blocking the application

new data is processed and passed to the GUI

Communication via the FESA class

1. Get the device handle from the directory server via RDA:
   ```
   rdaDeviceHandle* pDevice = RdaService->
                              getDeviceHandle("name");
   ```
   "name" is the instance name of the class e.g. 'BIF1.Control.sddsc004':
   Get instance named 'BIF1.Control' running on frontend 'sddsc004'.
2. Subscribe to data changes and get current state of all values:
   ```
   pDevice->monitorOn("propertyName", pReplyHandler,....);
   ```
   monitorOn() takes the name of the FESA property to be monitored and a pointer to a handler class.
3. Process new data in handler class:
   ```
   handleReply(rdaData& newData, ....);
   ```
   In the rdaData object, all values of the monitored property are enclosed:
   ```
   float value = newData.extractFloat("IRIS1");
   ```

When a value is changed by the user:

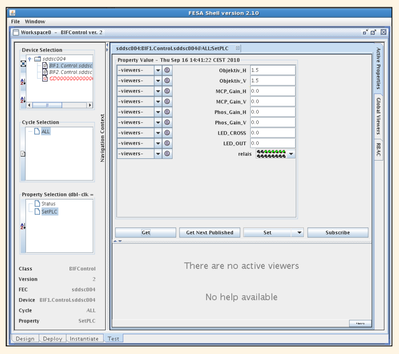
4. Set a value:
   Pack one or more values into an rdaData package and send it:
   ```
   rdaData* data = new rdaData();
   data->insert("IRIS1", 1.9);
   pDevice->set("propertyName", data, ....);
   ```

Please note: For reasons of clarity and readability some function parameters are omitted in the code fragments shown.

get/set/subscribe

Other applications / expert GUIs

- multiple applications have read/write access to the same class
- critical components (LEDs) are only accessible in expert applications



r.haseitl@gsi.de, c.andre@gsi.de, h.braeuning@gsi.de, t.hoffmann@gsi.de, r.lonsing@gsi.de