The LLRF control server at ELBE.

Martin Hierholzer

24th November 2015
ARD-ST3 Mini-Workshop: SRF controls and CW operation, Dresden-Rossendorf
LLRF software has two parts:

- Firmware running on FPGA
  - handles control loop and all fast reactions

- LLRF control server running on CPU
  - sets controls loop parameters,
  - acquires data for diagnostics and controls,
  - handles automation, and
  - deals with communication to the rest of the control system
Overview: Firmware and Hardware

LLRF control server

Martin Hierholzer (DESY) LLRF server at ELBE

(Ch. Schmidt, I. Rutkowski)
Current status

- The LLRF server used at ELBE is based on servers used at DESY for pulsed machines
- Server is based on “pulses” even in CW operation, but loop in firmware runs continuously
- Currently based on the DOOCS control system framework
- Can be used in stand-alone installation
- Server recently used to test MTCA-based LLRF system at ELBE (will replace the analogue system)
**Features**

- **Low-level interface to firmware:**
  - Convert I/Q measurements into amplitude and phase
  - Compute I/Q control tables based on amplitude/phase values from operator’s panel (set-points, feed-forward, gain)
  - Compute rotation matrices for input channels and vector modulator output

- **Safety features and calibration:**
  - Set limiters on forward, reflected and probe signals (fast shutdown in FPGA)
  - Configure attenuators and filters
  - Calibration to physical units (gradient in MV/m)

- **High-level algorithms:**
  - Output vector correction: automatically compensate for slow drifts
  - Automated ramp-up procedure with error checking

- **Additional features unused/deactivated for ELBE:**
  - Vector sum for multiple cavities
  - Learning feed-forward
  - Piezo driver control
Features

- **Low-level interface to firmware:**
  - Convert I/Q measurements into amplitude and phase
  - Compute I/Q control tables based on amplitude/phase values from operator's panel (set-points, feed-forward, gain)
  - Compute rotation matrices for input channels and vector modulator output

- **Safety features and calibration:**
  - Set limiters on forward, reflected and probe signals (fast shutdown in FPGA)
  - Configure attenuators and filters
  - Calibration to physical units (gradient in MV/m)
Features

- **Low-level interface to firmware:**
  - Convert I/Q measurements into amplitude and phase
  - Compute I/Q control tables based on amplitude/phase values from operator’s panel (set-points, feed-forward, gain)
  - Compute rotation matrices for input channels and vector modulator output

- **Safety features and calibration:**
  - Set limiters on forward, reflected and probe signals (fast shutdown in FPGA)
  - Configure attenuators and filters
  - Calibration to physical units (gradient in MV/m)

- **High-level algorithms:**
  - Output vector correction: automatically compensate for slow drifts
  - Automated ramp-up procedure with error checking
Features

- Low-level interface to firmware:
  - Convert I/Q measurements into amplitude and phase
  - Compute I/Q control tables based on amplitude/phase values from operator’s panel (set-points, feed-forward, gain)
  - Compute rotation matrices for input channels and vector modulator output

- Safety features and calibration:
  - Set limiters on forward, reflected and probe signals (fast shutdown in FPGA)
  - Configure attenuators and filters
  - Calibration to physical units (gradient in MV/m)

- High-level algorithms:
  - Output vector correction: automatically compensate for slow drifts
  - Automated ramp-up procedure with error checking

- Additional features unused/deactivated for ELBE:
  - Vector sum for multiple cavities
  - Learning feed-forward
  - Piezo driver control
Automated ramp-up procedure for ELBE

- Switch RF on with low gradient, pulsed mode and in open loop
- Adjust feed-forward and phase to match setpoint
- Close loop with low gain, check if error is below threshold
Automated ramp-up procedure for ELBE

- Switch RF on with low gradient, pulsed mode and in open loop
- Adjust feed-forward and phase to match setpoint
- Close loop with low gain, check if error is below threshold
- Switch to CW operation
- Ramp up gradient, correct amplitude and phase after each step
Automated ramp-up procedure for ELBE

- Switch RF on with low gradient, pulsed mode and in open loop
- Adjust feed-forward and phase to match setpoint
- Close loop with low gain, check if error is below threshold
- Switch to CW operation
- Ramp up gradient, correct amplitude and phase after each step
- Switch integral controller on
- Ramp up proportional gain while checking error
- Enable output vector correction

Implemented in BOOST meta state machine (eUML syntax)

Martin Hierholzer (DESY)
Automated ramp-up procedure for ELBE

- Switch RF on with low gradient, pulsed mode and in open loop
- Adjust feed-forward and phase to match setpoint
- Close loop with low gain, check if error is below threshold
- Switch to CW operation
- Ramp up gradient, correct amplitude and phase after each step
- Switch integral controller on
- Ramp up proportional gain while checking error
- Enable output vector correction
- Implemented in BOOST meta state machine (eUML syntax)
Current structure strongly tied to DOOCS:
- Divided into “locations” to achieve some modularity
- DOOCS variables are meant to be inherited to act e.g. on change: some important code (e.g. computation of rotation matrix) is inside DOOCS-inheriting classes
- Timer interrupts are handled by DOOCS and passed on to all locations

Unfortunately multiple copies of the code tree exist for different accelerators (needs to be merged)
Restructuring for full integration in ELBE

- Current structure strongly tied to DOOCS:
  - Divided into “locations” to achieve some modularity
  - DOOCS variables are meant to be inherited to act e.g. on change: some important code (e.g. computation of rotation matrix) is inside DOOCS-inheriting classes
  - Timer interrupts are handled by DOOCS and passed on to all locations
- Unfortunately multiple copies of the code tree exist for different accelerators (needs to be merged)

To do
- Modularisation needs to be changed for better maintainability
- Decouple logic and algorithmic code from control system interface
- Interface to timing system
- Take into account requirements for other (also pulsed) machines
Current structure of the server

- DOOCS + Panels
- IlrfCtrl
  - control loop parameters
  - automation
- cavity
  - set limits for forward, reflected and probe signals
- vm
  - calibrate vector modulator
  - control worker thread calculating control tables
- ctrlTableParams
  - RF pulse shape (delay, filling, flattop etc.)
- reference
  - calibrate reference
- algOVC
  - control OVC worker thread
- globalParameters and controlTables
- workerCalcCtrlTables
- workerDmaData
Planned structure with control system adapter

- Keep control system interface modular for multi-cavity systems
- Still just an idea, might change slightly in actual implementation!
Automated tests using the VirtualLab framework

- Automated tests are an important tool to ensure software quality.
- So far, the server can only be tested with the real hardware.
Automated tests using the VirtualLab framework

- Automated tests are an important tool to ensure software quality.
- So far, the server can only be tested with the real hardware.
- For automated tests, the server should run with simulated hardware.
- This allows testing specific conditions and procedures, including failures.
Automated tests using the VirtualLab framework

- Automated tests are an important tool to ensure software quality.
- So far, the server can only be tested with the real hardware.
- For automated tests, the server should run with simulated hardware.
- This allows testing specific conditions and procedures, including failures.
- The VirtualLab framework is developed as part of MTCA4U.
- A virtual device can be implemented and inserted in place of the actual device (via DMAP file).
Automated tests using the VirtualLab framework

- Automated tests are an important tool to ensure software quality.
- So far, the server can only be tested with the real hardware.
- For automated tests, the server should run with simulated hardware.
- This allows testing specific conditions and procedures, including failures.
- The VirtualLab framework is developed as part of MTCA4U.
- A virtual device can be implemented and inserted in place of the actual device (via DMAP file).
- Simple cavity model can be connected for realistic tests.
- Important parts of the firmware (control loop) have to be implemented (in some approximation).
Automated tests using the VirtualLab framework

- Automated tests are an important tool to ensure software quality
- So far, the server can only be tested with the real hardware
- For automated tests, the server should run with simulated hardware
- This allows testing specific conditions and procedures, including failures
- The VirtualLab framework is developed as part of MTCA4U
- A virtual device can be implemented and inserted in place of the actual device (via DMAP file)
- Simple cavity model can be connected for realistic tests
- Important parts of the firmware (control loop) have to be implemented (in some approximation)
- Currently under development, framework and skeleton ready
VirtualLab - framework for simulating hardware in MTCA4U

Virtual Device: SIS8300L

Firmware State Machine
Simulates control loop etc.

Dummy Register Set

IlrfCtrl server

Cavity Model

Signal Sink: I/Q

Timer

Signal Source: I/Q

Calculations

Signal Sink: I/Q

Signal Source: I/Q

Test Routines: Simulate operator commands, check server and cavity behavior
Summary

- LLRF control server sets control loop parameters
- FPGA firmware implements control loop
- Server for ELBE also implements automation for ramp-up etc.
- Server will use control system adapter to work with other control systems
- Automated tests in preparation to ensure quality while restructuring