MANUAL: DEV4 FOR LLRF FOR AWAKE2

KRISTIAAN PELCKMANS, APRIL 2024

Abstract. A manual for the of the Low Level Radio Frequency (LLRF) development for AWAKE2 in Uppsala University.

Contents

1. Introduction

This document describes the development of a low level RF (LLRF) control system for exploratory use with a micro Telecommunication Computing Architecture (MicroTCA.4) platform in the AWAKE2 (run C) experiment.

The present assignment is to make a working LLRF embedded system in the MicroTCA crate, using traditional tools as PID control and IQ sampling. There are plenty of opportunities for further research, summarised in conclusions. Key challenge in this assignment was not so much conceptual, but to get acquainted and constructive with the nitty gritty details of the embedded system implementation.

1.1. **MicroTCA4.0.**

MicroTCA.4 is a platform for embedded system programming, specially designed to address the needs in large physics experiments. MicroTCA is based on advanced TCA for implementing active embedded systems, often used in the telecommunication or military industry. Presently, implementations of LLRF systems based on the outdated VME or PXI standard are upgraded to MicroTCA, with new developments increasingly often realised natively in MicroTCA4.0.

1.2. **AWAKE 2C.**

AWAKE is a collaboration project [1] at CERN aiming to proof feasibility of wakefield acceleration. In AWAKE, a plasma wakefield is driven by a high-energy proton bunch (400 GeV, length 6-8 cm, or 19kJ per bunch, Adli et al., 2018). This wakefield is then used to accelerate electrons to 2 GeV, aiming for levels of up to 200 GeV. Wakefield acceleration may provide a 100 fold increase of gradient (i.e. 10GV/m rather than 0.1 GV/m) as compared to standard RF acceleration techniques. The proton bunch is delivered by the Super Proton Synchrotron (SPS) at CERN, delivering beam in 'only' a few percent of the cycles. The plasmas operate in a constant temperature of 200C, and an ambient atmosphere of Xenon. Figure (1) displays the energy regime where the AWAKE technology is of relevance.

The first iteration of AWAKE has shown promising result based on a 10 meter single plasma (Acc'or) field for realising a wakefield (Adli et al., 2018). Run 2C (see Figure 2) introduces a 2e plasma field (tube of 10cm $\times$10m on the beam line) preceding (by about 10cm) the accelerating plasma field order to increase efficiency further. This (self-modulating) plasma field is used to pre-seed the wakefield.

1.3. **A second e- LINAC.**

Figure 4 displays the new electron (e-) LINAC needed in AWAKE 2C for driving the self modulating (SM) plasma field. Such LINAC is equipped with an electron gun, a buncher and two RF accelerator structures. Each of these is properly controlled by an RF line. Figure (5) displays a typical electron gun as used in AWAKE. Operation of this 2e plasma requires a new electron line to be developed.

[1] See `https://home.cern/science/accelerators/awake`

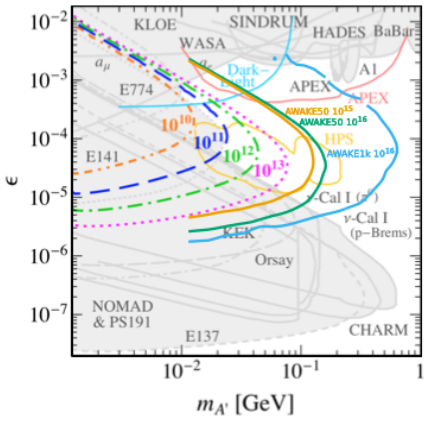


Figure 7. Limits on dark photon production decaying to an e^+e^- pair in terms of the mixing strength, ϵ, and dark photon mass, $m_{A'}$, from previous measurements (light grey shading). The expected sensitivity for the NA64 experiment is shown for a range of electrons on target, 10^{10}–10^{13}. Expectations from other potential experiments are shown as coloured lines. Expected limits are also shown for 10^{15} (orange line) or 10^{16} (green line) electrons of 50 GeV ("AWAKE50") on target and 10^{16} (blue line) electrons of 1 TeV ("AWAKE1k") on target provided to an experiment using the future AWAKE accelerator scheme. From Ref. [84].

FIGURE 1. Regimes (energy levels) that AWAKE can prompt, as compared to other machines. This graph displays particularly the regime of mixing strength versus dark photon mass. [84] Alemany, R.; Alemany, R.; Burrage, C.; Bartosik, H.; Bernhard, J.; Boyd, J.; Brugger, M. Summary Report of Physics Beyond Colliders at CERN. arXiv 2019, arXiv:1902.00260.

Both e-guns operate in S band. The first electron injector line needs an added S band RF. The second electron injector line needs 2 added X band controllers. The x-band technology is an up-converted version of the S-band developments. This totals 5 RF structures that need to be controlled RF supply and measurement points. A schematic overview of a control loop is shown in Figure (6). The layout of the first control loop is displayed in Figure (7).

The components are described as follows:

SSPA: The Solid State Power Amplifier has as input signal the result of the VM, and amplifies this to 200W (roughly 55dB).

Klystron: The Pulse Modulator and Klystron amplifies this 200W signal to peak powers of 7.5MW.

PC: The BOC Pulse Compressor (PC) compresses the pulse with a factor 6, resulting in a peak of 15MW.

Acc'or: The RF structure accelerating the electrons before injecting in the main beam.

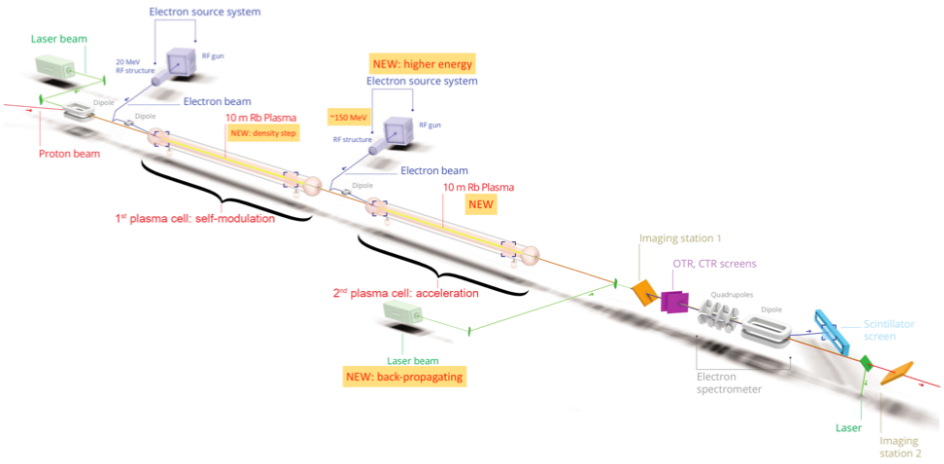


FIGURE 2. *Overall layout of the AWAKE2 (Run C) experiment with both plasma fields.*

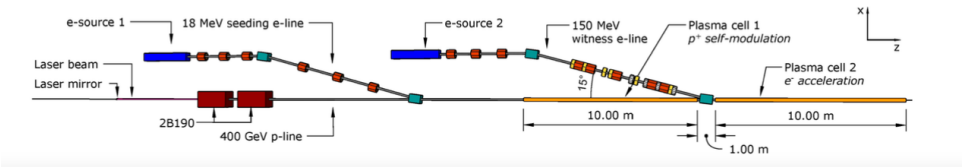


FIGURE 3. *Schematic overview of the 2 electron injector lines to be controlled.*

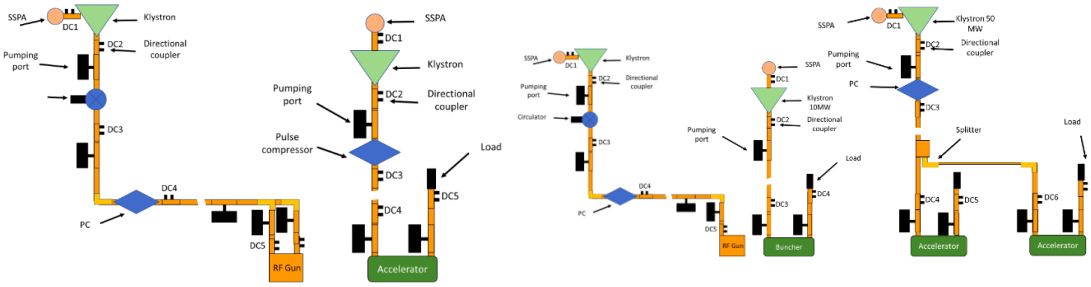
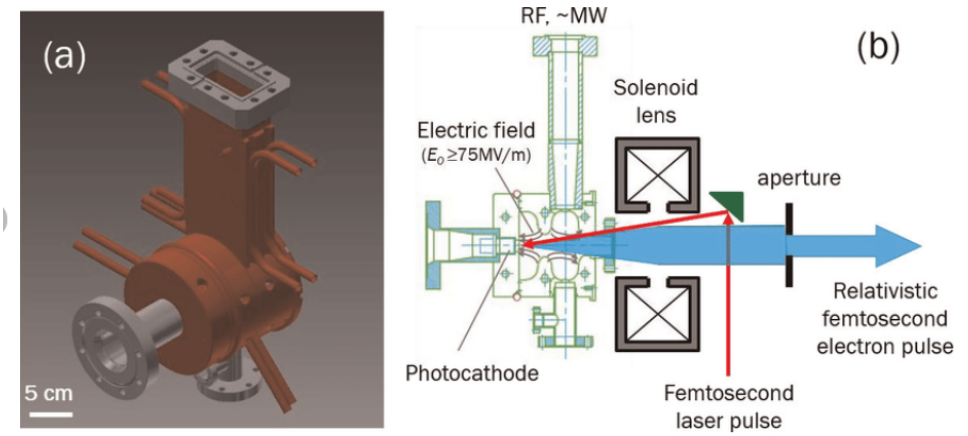


FIGURE 4. *The two electron injector lines (needed for AWAKE run 2C. Left panel : 1e injector line with S-band e-gun and a single S-band RF accelerating structure, Right panel : 2e injector line with S-band e-gun, an X-band buncher and two X-band RF accelerating structures.*



(a) Photocathode RF gun and (b) schematic for the generation of femtosecond electron pulses in the RF gun.

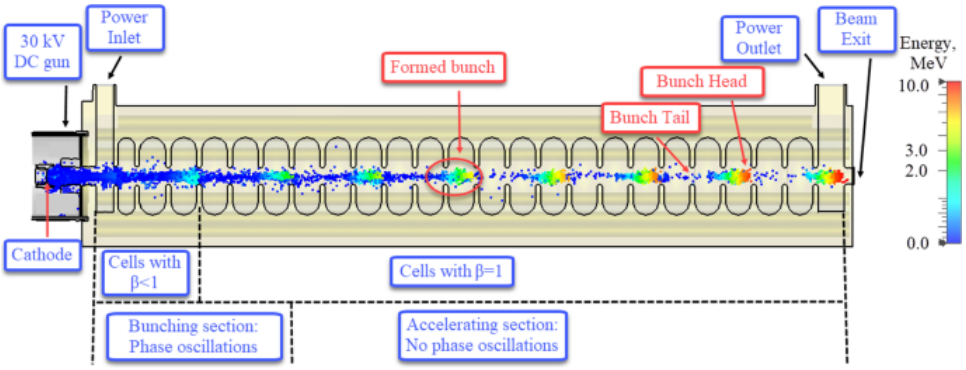


FIGURE 5. *(top panel) a schematic overview of an electron gun. (bottom panel) a schematic overview of a bunching and acceleration structure.*

1.4. **Requirements.** The desired stability properties (i.e. phase and amplitude coherence) are summarised in Table (1). Process stability of the closed control loop is a given. Stability in this table refers to pulse-to-pulse variation, and is more commonly referred as amplitude/phase coherence. The ultimate aim is to synchronise the electron bunches exactly with the wakefield, requiring tight (actively controlled) phase coherence of the feeding RF fields.

1.5. **Disturbances.** This subsection gives an overview of the disturbances on the present signals.

MICROPHONICS: The effect of external mechanical vibrations account for the bulk of measured disturbances.

DROOP The performance of the active hardware as the klystron declines over time.

RIPPLE External effects tend to induce ripple in the desired signals: external effects will be compensated for when consistent.

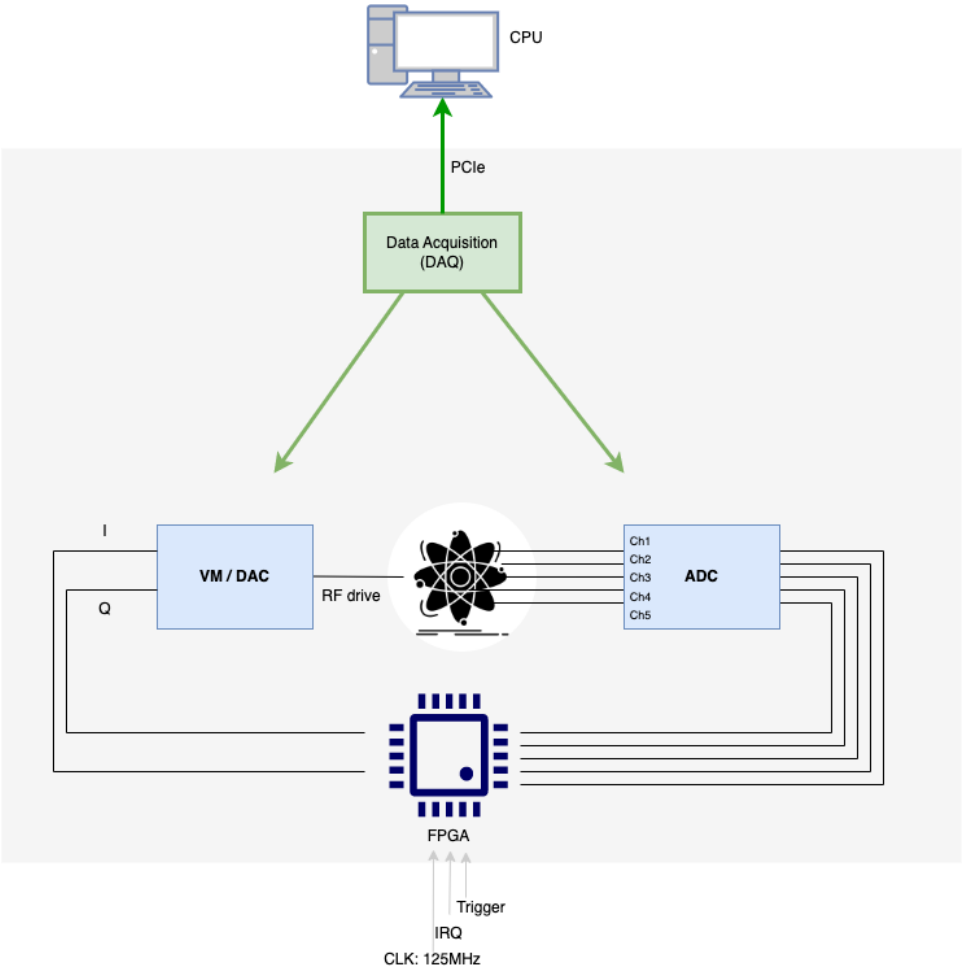


FIGURE 6. *Schematic overview of a feedback loop in this context. Note that the data acquisition 'only' acts as an observer.*

TABLE 1. Desired stability properties.

Property	**S-band**	**X-band**
Operating freq.	2.997899068GHz	11.991596272GHz
Pulse length	6 us	100-1500 ns
Bandwidth	>20 MHz	>20 MHz
Pulse repetition rate	9.97 Hz	9.97 Hz
Jitter (difference timing-pulse start)	< 30 fs	<30 fs
Amplitude stability	10^{-4}	10^{-4}

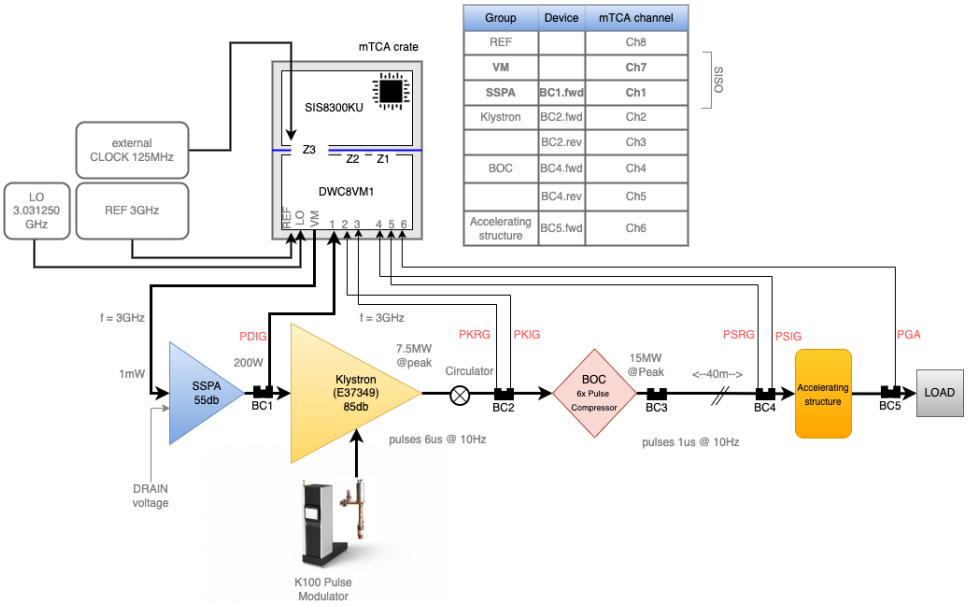


FIGURE 7. *Layout of the present control loop, with BCs as connected to the MicroTCA crate (names of BC as used in the above figure marked in red). The PID SISO loop focusses on the VM (as input signal) and BC1.fwd (Channel 1, as output signal).*

BEAM The loading of the beam induces a disturbance of the field. This effect is insignificant in the setup of AWAKE.

POWER The supply of power induces significant variations.

THERMAL Variations in the outside temperature imply a significant disturbance effect.

DUTY Variations into the duty cycle induce a significant disturbance effect.

JITTER Clock jitter in the ADC clocks is impacting directly the measurement accuracy. The manual of the AD9510 PLL chips indicate a 40 fs jitter.

X-talk One has considerable cross-talk between the different channels, especially if the MicroTCA powers more SIS+RTM pairs.

AE Ambient (environmental) electronic radiation is a source of noise in the ADC.

1.6. **Pulse compressors.** Figure (10) illustrates the effect of a SLED Pulse Compressor (PC). The result of a BOC pulse compressor is similar, only needing a single cavity (see `3GHzBarrelOpenCavity(BOC)RFpulsecompressorforCTF3`). CERN has developed a SLED-I pulse compressor for use in X-band (see Woolley, Syratchev, Dexter, 2017), and figured a PI pulse-to-pulse approach to give the pulse from the compressor a flat top. The S-band will use a BOC, and the X-band setup a SLED-I.

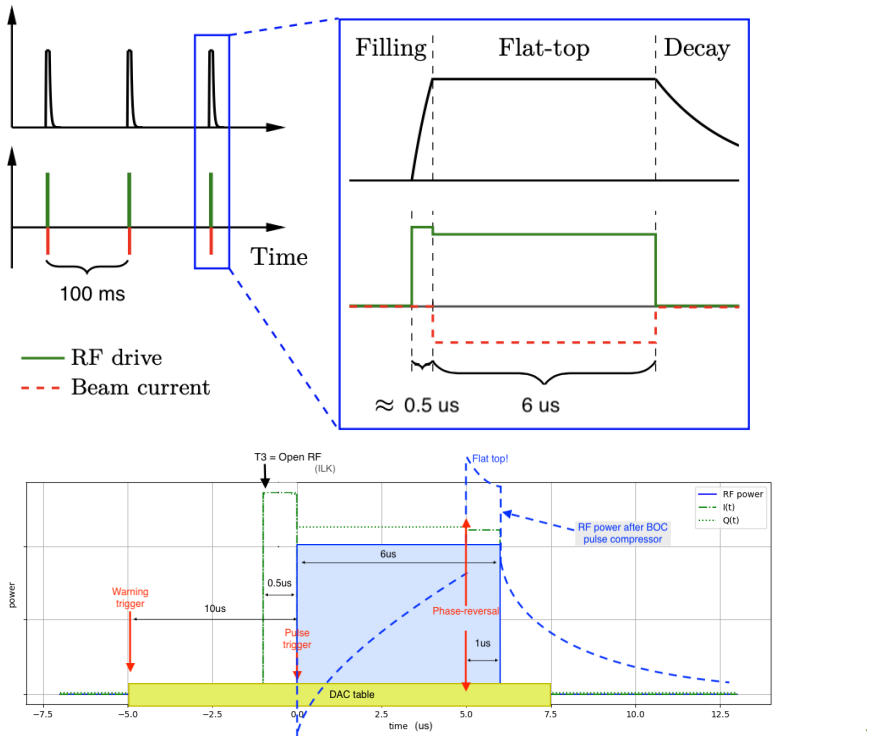
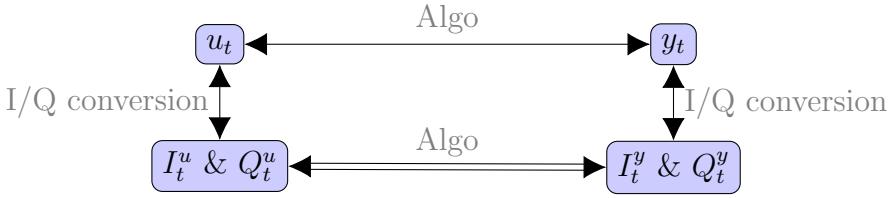


FIGURE 8. *Schematic of a pulse, occuring at 10Hz: (a) (from O.Troeng, 2017) with adjusted time lengths. The Y-axes denote power (Watt). (b) the layout of a pulse (blue square) in terms of the I and Q signals, with triggers from the interlock.*



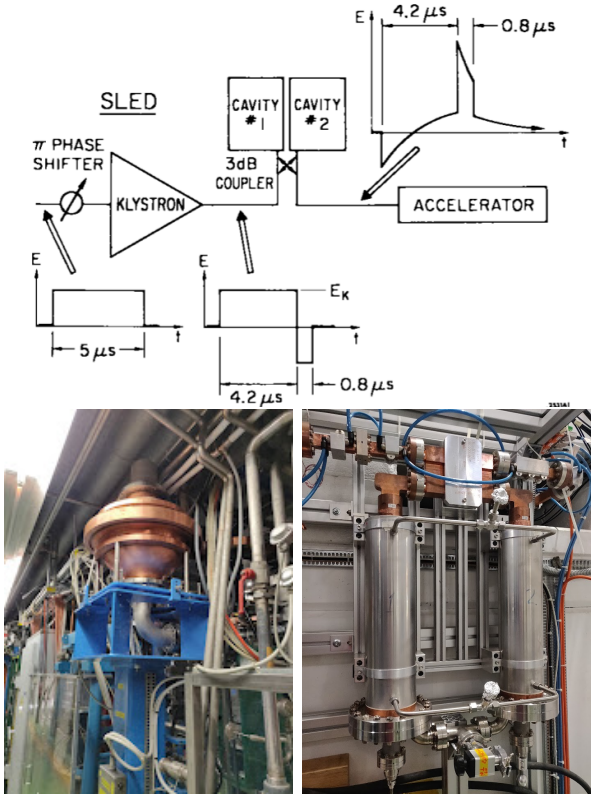


FIGURE 9. *(a) The behavior of the SLED pulse compressor (Z.D. Farkas et al., 1974). (b) Photo of the BOC pulse compressor to be used in the first electron injector line, (c) Photo of the SLED-I pulse compressor to be used in the second electron injector line.*

2. LLRF Hardware

2.1. **Setup.** The hardware consists of:

- Workstation (running centOS), with a licensed Vivado 2020.1 running.
- MicroTCA (micro telecommunication Computing Architecture) crate.
- R&S signal generator devices, serving LO, reference and clock signal.
- An internet router, connecting all hardware modules.

The setup of the workstation/crate/router/signal generators is displayed in (10). The MicroTCA crate has the following components on board:

- Crate (Schroff 19", 19U crate, with vertical cooling).
- N.A.T.-MCH (-PHYS80) (with CLK module and PCIe switch).
- N.A.T.-MCH-RTM (-BM) (with E3 COMex based on a Xeon processor, running Ubuntu 20.04 TLS).
- 2 N.A.T. power units (600 W).
- Backplane (TCLKA, TCLKB).

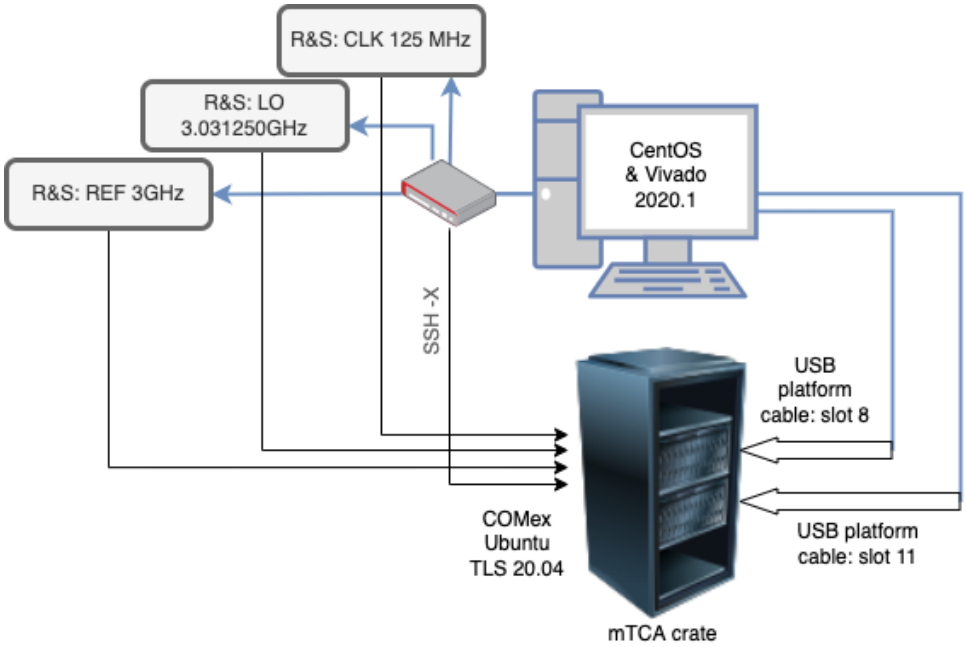


FIGURE 10. *Hardware setup.*

- RF-backplane, distributing power for the RF-backplane, CLK, REF and LO to all the RTMs connected to the RF-backplane.
- 2 pairs of SIS8300KU+DWC8VM1 AMC cards (on slot 8 and slot 11 respectively)
- A NAMC - PMC module (NAMC-PMC-T261-20R).

The workstation runs vivado 2020.1, and is connected to the NAT-MCH via telnet

```
> telnet 192.168.30.50
```

and to the COMex via a secure shell (SSH):

```
> ssh -X mch@192.168.30.105
```

The RF-backplane provides functionality to distribute LO, REF and clock signals to all the RTM cards connected to the RF-backplane. Fotos from the MicroTCA crate (as of 20 Aug. 2022) are displayed in figure (11). The signals to be provided at the frontplate of the SIS, and the RTM are specified in Figures (12) and Figure (13).

2.2. **ICs on the AMCs.** The following ICs are implemented on the SIS8300ku:

AD9510 (PLL) The two Phase Lock Loop ICs proceed the dual ADCs, and feed them with a cleaned clock signal.

AD9628 (ADC) The dual 125s/s ADC digitise the IF RF signals coming in from zone 3.

FIGURE 11. *Foto's of the MicroTCA crate (a) front side of the MicroTCA crate (with the 2 SIS 8300KU digitiser AMCs), (b) rear side of the MicroTCA crate (with the 2 RTMs (DWC8VM1) cards).*

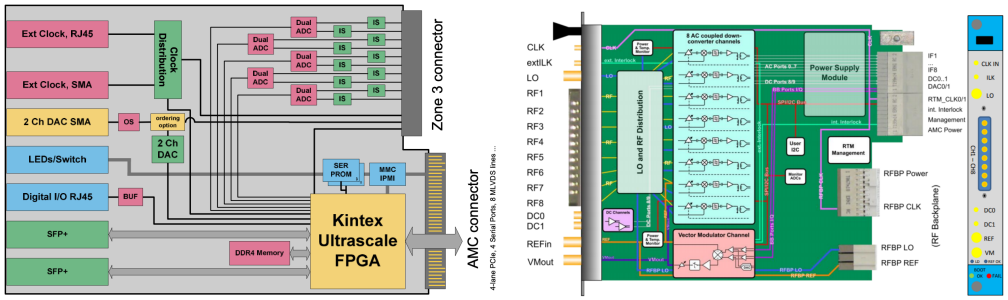


FIGURE 12. *Schematic of (a) the SIS8300KU AMC and (b) the DWC8VM1 RTM card.*

MAX5878 (DAC) The 250 s/s converter of the dual I and Q signals to their analog counterparts. The resulting 2 RF channels are fed to the Vector Modulator IC (TRF370417) on the RTM.

LTC2493 24 bit ADC for power measurement.

PCA9535 I/O expander.

SI5338A Programmable quad output PLL.

AD8363 LO power and reference power detector.

On the DWC8VM1, the following ICs are implemented:

TRF370417 Vector Modular (IQ modulator) IC.

HMC624LP4 Variable gain RF attenuator.

for additional information, see the respective manuals.

2.3 Signal Specification

2.3.1 Front Panel Signals

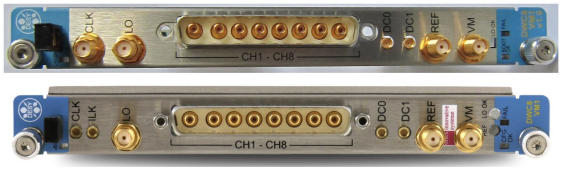


Figure 2: Front panel of DRTM-DWC8VM1 (R11, top and R12 bottom)

Table 1: front panel signals

Signal	Parameter	Value
ADC clock input (CLK)	Connector type	SMA (R11), MMCX (R12), single-ended
	Impedance / Coupling	50 Ω / AC
	Frequency range	10..130 MHz
	Power level	10..13 dBm
Interlock input (ILK, R12 only)	Connector type	MMCX (R12), single-ended
	Impedance / Coupling	50 Ω / DC
	Frequency range	DC like
	Input Voltage	-0,5…6,0 V
Local oscillator input (LO)	Connector type	SMA, single-ended
	Impedance / Coupling	50 Ω / AC
	Frequency range	700..4000 MHz
	Nominal power level	+12 dBm (max. +18 dBm)
	Return Loss	< -14 dB
Down converter input (CH1..CH8)	Connector type	FBM multi-coax, single-ended (FCT – FM8W8, FBM022154MR)
	Impedance / Coupling	50 Ω / AC
	Frequency range	700..4000 MHz
	Max. input power level	+15 dBm (for min. attenuation)
	Input 1 dB compression	+9 dBm (for min. attenuation)
	Return loss	< -14 dB
	Max. adjustable attenuation	> 30 dB
DC input (DC0, DC1)	Connector type	MMCX, single-ended
	Impedance / Coupling	50 Ω / DC
	Frequency range	DC..650 MHz
	Max. voltage level	2 V (peak-to-peak)
Reference input (REF)	Connector type	SMA, single-ended
	Impedance / Coupling	50 Ω / AC
	Frequency range	700..4000 MHz
	Nominal power level	12 dBm (max. +18 dBm)
	Return Loss	< -14 dB
Vector modulator output (VM)	Connector type	SMA, single-ended
	Impedance / Coupling	50 Ω / AC
	Frequency range	700..4000 MHz
	Output power level	+10 dBm (for min. attenuation)
	Return Loss	< -14 dB
	Max. adjustable attenuation	> 15 dB

FIGURE 13. *Signal specifications as to be provided on the frontplate of the RTM (courtesy Struck manual, 2023).*

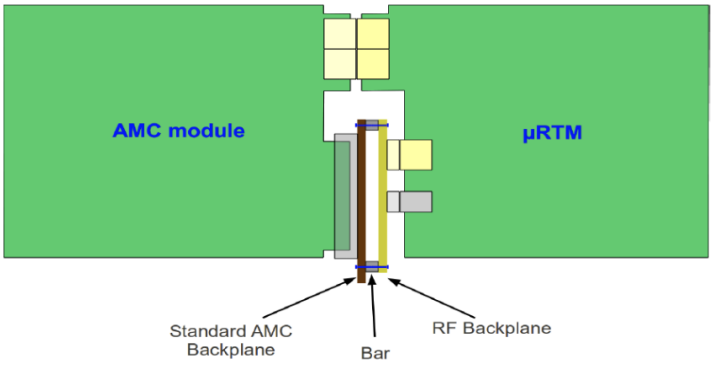


FIGURE 14. *Backplane interconnecting ACMs, and RF-backplane for distributing LO, REF RF and clock signals to the RTMs.*

2.3. **Installation.** The first step is to have the COMex run a full linux OS. We installed UBUNTU 20.04 TLS via the USB ports on the MCH-RTM (in person), and installed the SSH server so we could connect via the workstation. The COMex serves as the root to the PCIe tree, meaning changes in the setup need a reboot

of the COMex (in turn re-enumerating the PCIe tree). The (signals of the) DAQ is accessible here via DMA.

2.4. **Management.** Management is done via the N.A.T. CLI interface to the MCH. Alternatively, one can use the web interface (HTTP) or NATview (Java tool).

2.5. **Monitoring.** Operations of the crate (e.g. temperatures of the hardware components) can be monitored via the I2C and IPMI protocols. However, the MCH hardware provider (N.A.T.) supports a web-interface to many of those features.

2.6. **Interlock and safety.** Figure (15) displays the interlocks (ILKs) of the design.

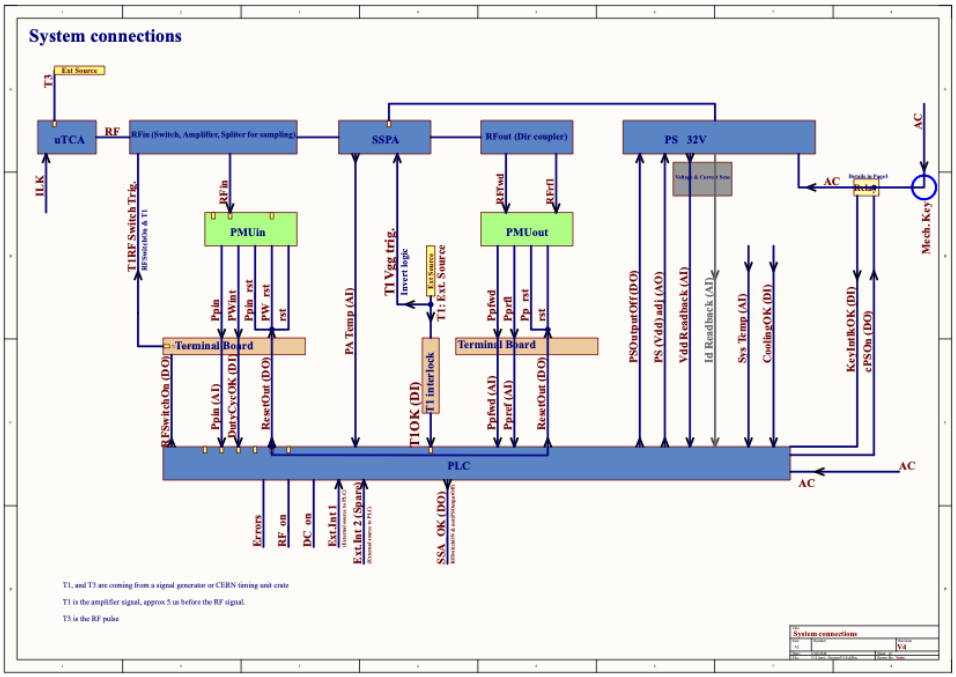


FIGURE 15. The interlocks (ILKs) of the design.

3. Firmware

3.1. **Overview.** An overview of the functionality of the firmware implemented on the FPGA is given in Fig. (16).

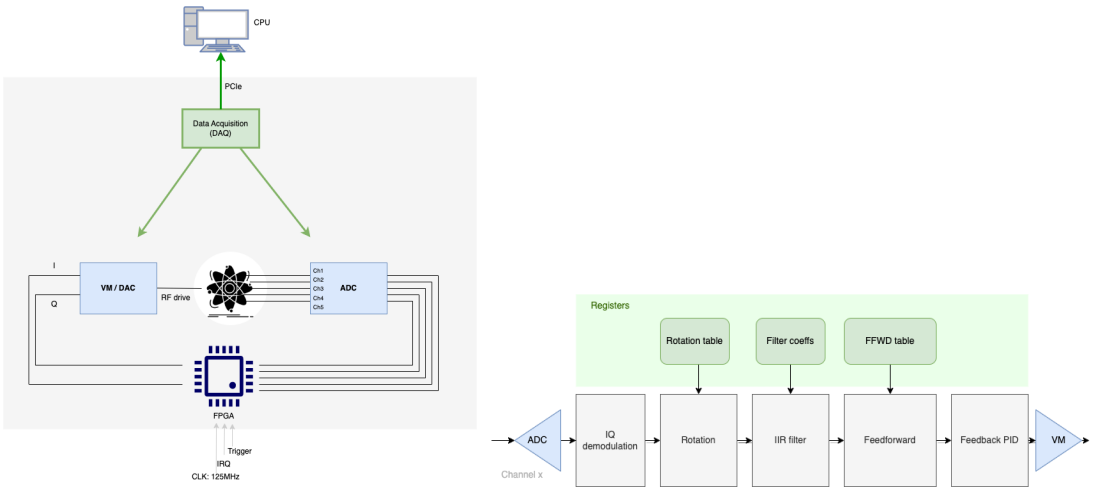


Figure 16. *Schematic overview of a feedback loop in this context. Note that the data acquisition 'only' acts as an observer.*

3.2. **Installation.** The firmware is developed in Vivado 2020.1, running on our workstation. Vivado communicates with the 2 FPGAs via two USB platform cables, connected to the SIS8300KU cards. The file `devo.py` in the `sw` development sets the configuration options, implemented in hardware using the `MicroTCA4u` Python library.

Initial installation from scratch follows those steps:

- Setup the MicroTCA crate, COMex, external workstation running Vivado 2020.1, R&S signal generators and connections as described above.
- Clone devo from gitlab, at

  `https://gitlab.desy.de/kpelckma/devo`

  and clone all submodules as

  `git submodule update --init --recursive`
- Run the installation scripts:

  `make install`

  and

  `make project cfg=dwc8vm1`
- Open the generated project in Vivado 2020.1, and generate the bitstream. Then program the device (we use a JTAG USB platform cable), and reboot the COMex (root) to re-enumerate the PCIe tree.

If one wants to develop the project further, do

- Clone the devo as above, and clone all submodules recursively.
- Open the development in your favourite editor, update and resynthesize at will.
- Push the update to gitlab as

3.3. **DESY development.** This implementation is based on DESYs development in the MSK group. This development provides the following functionality

RDL: The Register Description Language (RDL) provides a standardised way to describe registers. DesyRDL is a compiler for such RDL files, compiling the registers for access (called *view*) through (1) VHDL firmware, and (2) Python-based software. The interface between either view is implemented via PCIe and the direct memory access (DMA) implemented in the MicroTCA4u (ChimeraTK) package, in turn building on the Xilinx IP core for (XDMA).

ChimeraTK: ChimeraTK is a library providing 3 functions:
(1) Libraries supporting PCI-express (PCIe).
(2) API for the applications running on the FPGA (i.e. read and write registers using DeviceAccess).
(3) Interface to the higher-level control software (as EPICS, see `https://github.com/aquenos/ChimeraTK-ControlSystemAdapter-EPICS`).
The Python library `MicroTCA4u` (`https://github.com/ChimeraTK/DeviceAccess-PythonBind blob/master/MicroTCA4u.py`) provides Python bindings to this library (for use, see `devo.py`).

DeviceAccess: The ChimeraTK DeviceAccess library provides an interface for register-based devices.

QtHardMon: A GUI to readout values of registers created/managed by desyRDL, based on the ChimeraTK-DeviceAcess library.

XDMA: A DESY wrapper to the Xilinx IP core implementing Direct Memory Access (DMA) over PCIe.

PKG: DESY provides a number of library packages (common, math, peripeh).

BSP: DESY developed a Board Support Package for the SISS8300ku card.

FPGA: Configuration of the FPGA (Ultrascale Kintex) is served by DESY's FPGA configuration manager.

RTM: Functionality of the RTM is provided by DESY's RTM module.

Earlier on, we considered alternative developments: (1) the STRUCK cern ROOT demo, and (2) the CERN internal developments for the update of the Super Proton Synchrotron (SPS). The STRUCK demo is based on outdated IP cores (as the MicroBlaze soft core). The CERN developments use extended features from a.o. the LM32 soft core and the white rabbit clock network. Since we do not use those features, but latency requirements make this development not

applicable, we decided to start working with DESYs MSK standardisation effort. Developments are in general not compatible.

3.4. **DESY's Register Description Language.** A tool is provided (`QtHardMon`) (see `https://github.com/ChimeraTK/QtHardMon`) to readout current values of the registers. The registers are divided in 2 types:

- FR The Firmware registers (FR) describe (in `fw > src > hw > devo > app.rdl`) the smaller registers used in the firmware development.
- DR Larger blocks of memory are described in the DMA registers (DDR4) (in `fw > src > hw > devo > app_dma.rdl`).

3.5. **Numerical data formats.** Conversion is done though ChimeraTK DeviceAccess, based on the `.mapp` files detailing generated by desyRDL. DeviceAccess can also provide *raw* readings.

3.6. **PCI express.** The DAQ forwards readings of the ADCs into DDR4 memory, which are read out with the Direct Memory Access (DMA) modules using PCIe (port 4-8). This functionality is covered by the DeviceAccess libraries as developed by DESY, and as based on the xDMA module as provided by Xilinx.

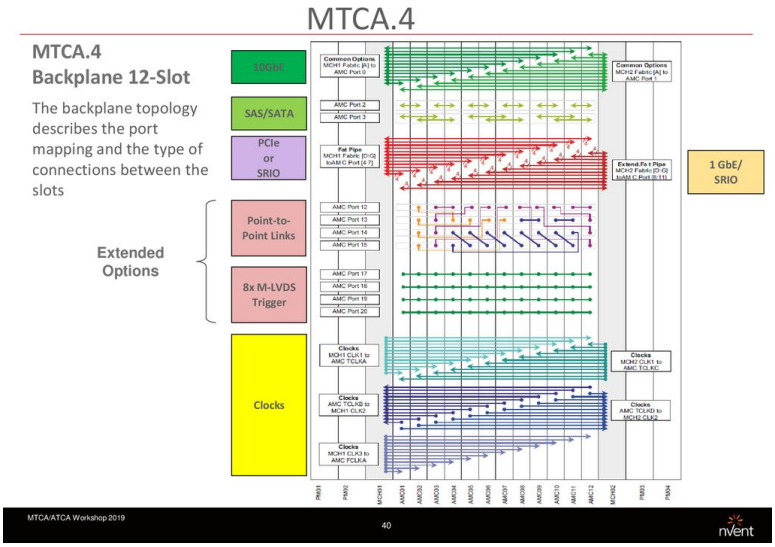


FIGURE 17. The layout of the AMC backplane of the MicroTCA crate, implementing the PCI express, clock signals and MLVDS trigger lines.

3.7. **Timing and clocks.** Clock signals provide the backbone to any firmware implementation. By default, the firmware developments is driven by the internal quartz clock, running on a 125MHz clock signal provided by the internal quartz in the SIS8300KU card. Figure (18) displays the paths of the different clocks on the SIS8300KU card: if one likes to use MicroTCA crate-wide clocks (from either backplane, or RF backplane), one needs to re-route the clock signal and adjust the MCH configurations accordingly.

The clock network is designed as follows in the firmware.

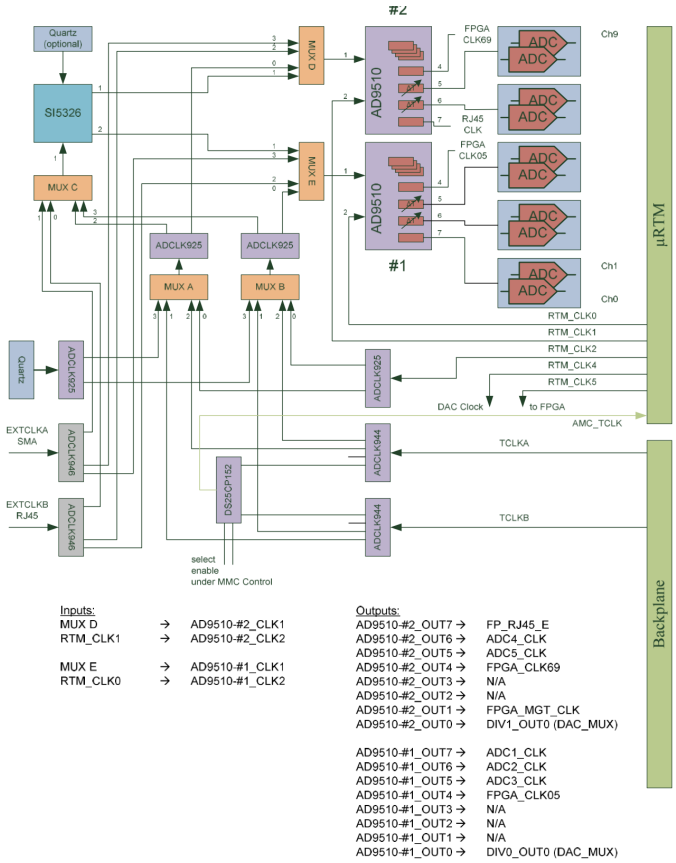


FIGURE 18. *The paths of the CLOCK signal in the SIS8300KU card (see manual SIS8300KU, p16). Route the desired CLKs through the appropriate MUXes.*

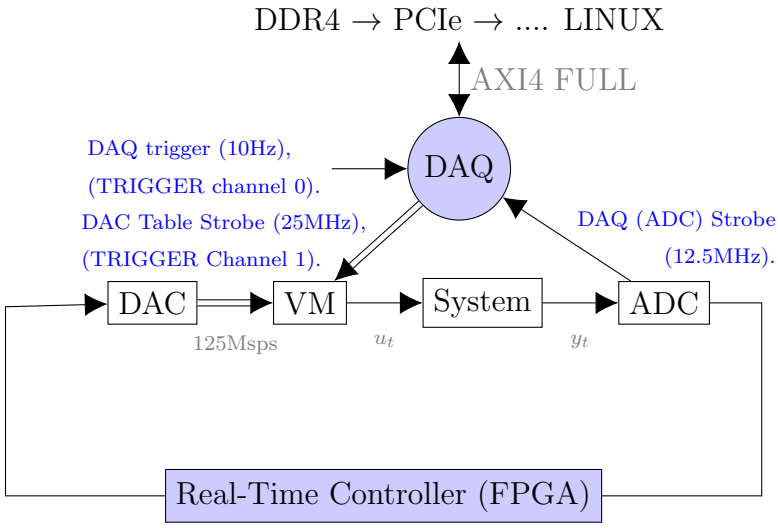
3.8. **Modules.** The DAQ ('Data AcQuisition' module) *observes* the realtime working of the ADC-DAC every 10 samples (WORD-DIVIDER=10). The aim of the signals acquired by the DAQ is to

- Record the signals for post processing.
- Monitor the signals for drift detection.

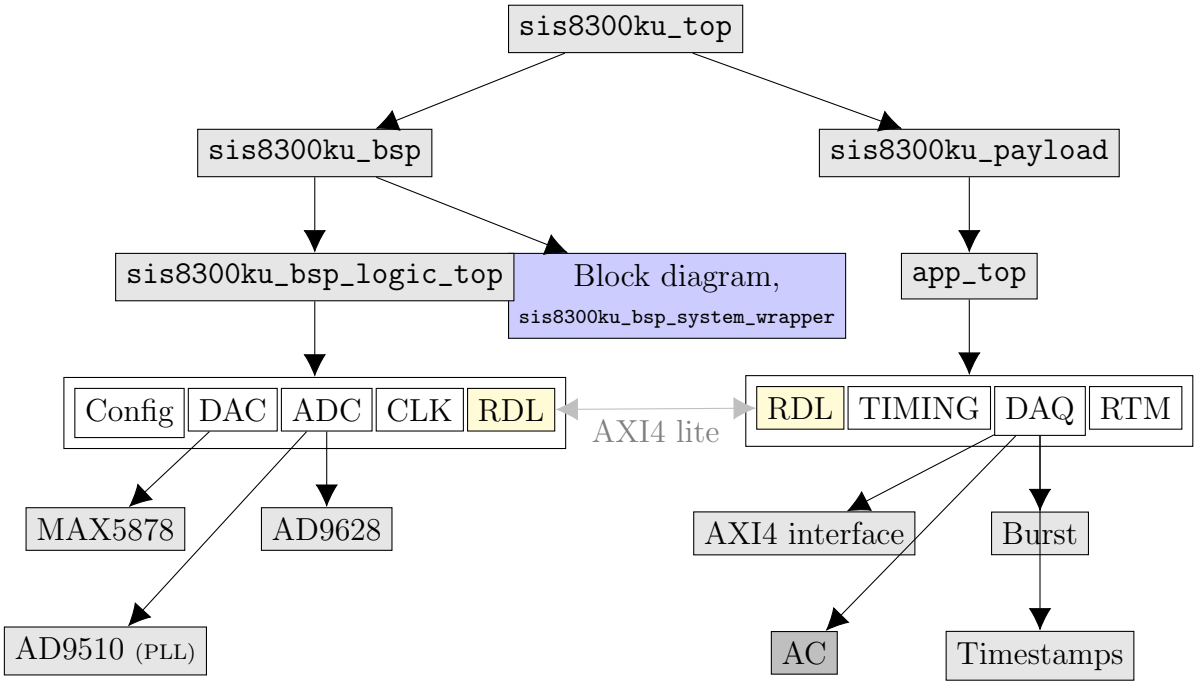
- Testing and development of the firmware.

The DAQ is implemented by a *strobe* signal, observing the ADC at 12.5M and the DAC at 12.5M (interleaving I and Q) samples per second, every time the trigger comes (at 10Hz). That means that if the DAC has to output 1024 samples ('words'), it would take 40.96us for all entries in this buffer to be realised. This is repeated every time a trigger comes (at 10Hz).

This sampling architecture is represented as follows:



The overall architecture of the `fw` VHDL code is as follows:



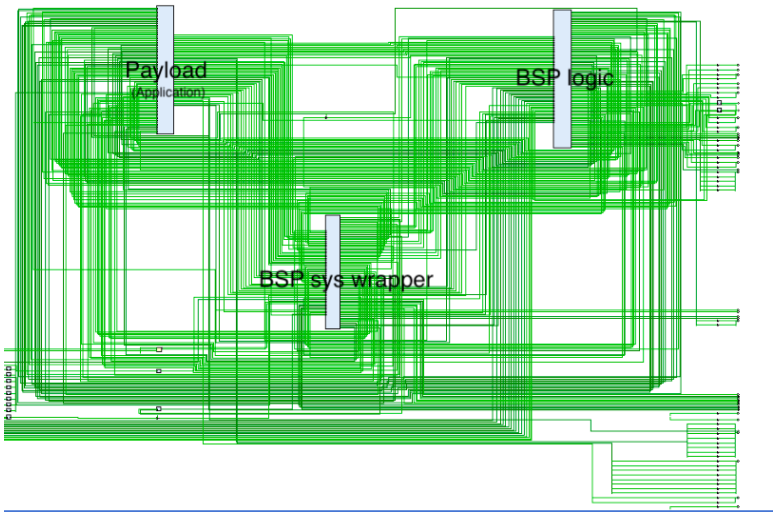


FIGURE 19. *Schematic of the complete development. The application-specific logic resides in the* payload.

The `sis8300ku_top` is tied (*'constrained'*) to the pins of the FPGA in the `.xdc` files. Both AXI4 full and AXI4 light protocols are used to interconnect the involved modules, see also figure (20) and DESYs FWFWK documentation. This design separates the development into:

- A hardware-specific arm (front ended by the 'board support package'), more or less tied to the hardware.
- An application-specific arm (called the 'payload'), to be tailored to the intended application.

Key top modules are:

BSP: The Board Support Package of the SIS8300KU (Struck) card.
APP: The payload where the desired application-specific logic resides.

Key modules of the payload are:

DAQ: The Data AcQuisition module grabbing data from the ADCs and sending those into the DDR4 using AXI4L.
TIMING: The TIMING module converts a general-purpose clock signal to trigger and strobe signals (of the DAQ).
RTM: Functionality for the RTM - in our case the DWC8VM1.
RDL: The Registers (with the RDL module describing the interface) provide the *glue* between the 2 arms.

The three timing signals are as follows:

DAQ trg.: The 10Hz trigger of the DAQ module triggers

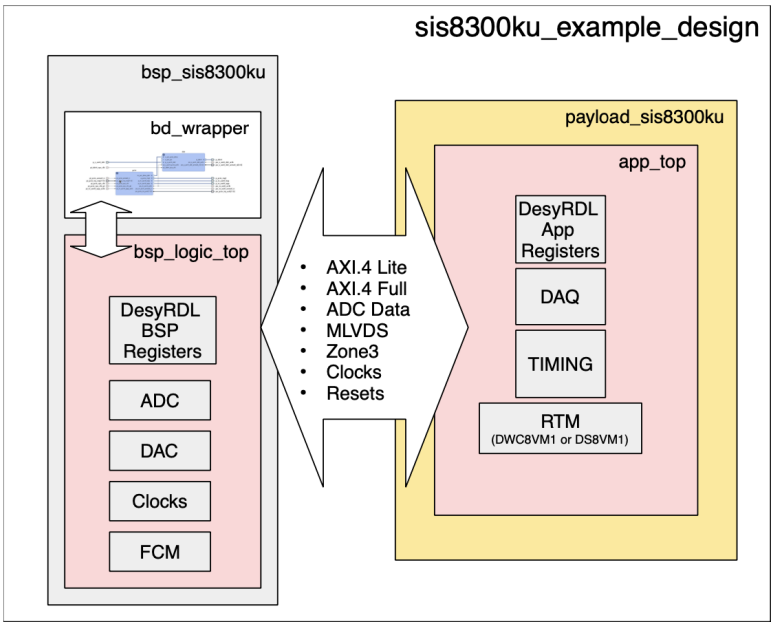


FIGURE 20. *Overall design abstracting BSP layer from the payload (application specific) of the FWK design from DESY on which this development is based. See FWK documentation.*

DAC stb.: The DAQ strobe times how fast the entries in the DAC table are fed to the DAC. For example, if the DAC strobe operates at 25MHz, it takes 40.96us to realize 1024 samples ('words') in the DAC/VM

ADC stb.: The ADC strobe (named DAQ strobe in the `fw`) times the observations made from the ADCs and sent to the DAQ. The DAQ in turn packages those and sends the messages over AXI4 Full into the DDR4 memory.

The ADC logic works at a sampling rate of 125MS/S as follows. The 5 dual ADC ICs (AD9268) are driven by the clock signals generated by the 2 AD9510 PLL ICs.

The DAC logic works at a sampling rate of 250 MS/S. The two digital signals (I and Q) are converted into analogue signals by DAC IC (a dual MAX 5878). This signal in turn feeds a Vector Modulator (VM) IC on the RTM (the DWC8VM1 card paired with the SIS8300KU). The resulting signal is up-converted using the LO and output at the VM output port of the RTM. (see figure 12 and the manuals from the SIS8300KU and DWC8VM1).

3.9. **Memory map.** The memory map is generated and managed by desyRDL. The description is performed in systemRDL. The resulting map is given in .mapp files.

CERN has developed similar tools here:
`https://gitlab.cern.ch/be-cem-edl/common/cheby`

TABLE 2. *Estimated latency in terms of 125 MHz (spacing 8ns) clock cycles.*

Module	**clock cycles**
ADC sampling	3

which interface both to the higher level control units (header files, FESA classes) and generate source code (VHDL, ...).

3.10. **Development.** The added development in `devo` is mainly invoked in the APP module. A convenient way to understand the development is by reading:

`fw > src > hw > devo > hdl > app_top.hdl`

In particular, signals `pi_adc` (input signal) and `po_dac_data_i,po_dac_data_q` (output signals) contain the 16 bit samples from the ADCs and DAC respectively. This file also describes instantiation of desyRDL, RTM, TIMING, DPM (dual port memory) and the DAQ module.

3.11. **Intermediate Frequency.** The intermediate frequency (IF) on which the firmware operates is set to 31.25MHz. This means that the 125MS/S (M Samples per Second) sampler provides exactly 4 samples per cycle of the IF. The RTM (DWC8VM1) supports a range of IF between 5MHz and 100Mhz, and we require a bandwidth of at least 20MHz.

This choice makes also a good trade-off between different effects:

(1) We need sufficient bandwidth (and hence the IF larger than 20Mhz).
(2) We need to satisfy Nyquist: for 125MSPS, we need at most an IF=62.5MHz as we need 2 samples per cycle.
(3) Too high IF frequencies will yield considerable aliasing effects.
(4) Too low IF will result in slower artefacts because of rise time of the carrier. For example, if one has an IF of 1kHz, it would take about 31 250 samples before the carrier rises to the desired maximum.
(5) Individual samples are noisy, so one likes to stay away from the bare maximal IF.

3.12. **Latency.** The total latency of the LLRF is estimated as in table (2).

Latency (delay) has in general a detrimental impact on the performance (stability) of a control system, often resulting in severe overshoot (ripple) effects. A standard way to compensate for delay is to use a Smith control estimator.

4. Digital Signal Processing

4.1. **Setup.** Signals in continuous time are denoted as $y(t)$ and $u(t)$. They are generally sampled in discrete (Zero-Order Hold, ZOH) with time constant τ and index k, or

$$\begin{cases} u_k = u(k\tau) = u(t) \\ y_k = y(k\tau) = y(t). \end{cases} \tag{1}$$

In case of a 125 MS/S ADC sampling device, $\tau = 8ns$.

4.2. **I/Q Sampling.** Consider a continuous signal $s(t)$, sampled in discrete time s_k at regular instances with $\tau > 0$ spacing, or $s_k = s(k\tau)$. This signal looks as

$$s(t) = A(t)\sin\big(2\pi f_c t + \varphi(t)\big), \tag{2}$$

with $A(t)$ denoting amplitude, and $\varphi(t)$ denoting phase. This signal is untangled in its I-phase (I) and In-quadrature (Q) part, denoted as s_k^I and s_k^Q respectively. Both are combined into the original continuous signal s as

$$s(t) = I(t)\cos(2\pi f_c t) + Q(t)\sin(2\pi f_c t), \tag{3}$$

or in discrete time as

$$s_k = I_k \cos(2\pi f_c \tau k) + Q_k \sin(2\pi f_c \tau k). \tag{4}$$

Figure (21) displays an example of a signal with a fixed IQ value. It follows that amplitude and phase are a nonlinear map from I and Q so that

$$\begin{cases} A(t) = \sqrt{I^2(t) + Q^2(t)} \\ \varphi(t) = \tan^{-1}\left(\frac{I(t)}{Q(t)}\right) \end{cases} \tag{5}$$

and similarly in discrete time.

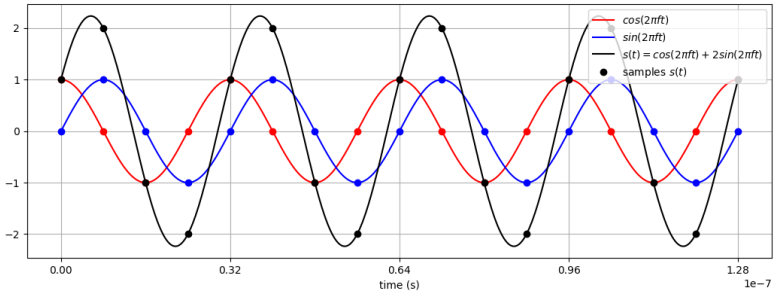


Figure 21. Example of a signal with a fixed I/Q value where $I = 1$ and $Q = 2$. In this case, we have 4 samples per cycle, and we display 4 cycles of the signal. If the signal has $f = 125$Mhz, this would represent 128 nano seconds (i.e. $1.28e-7s$).

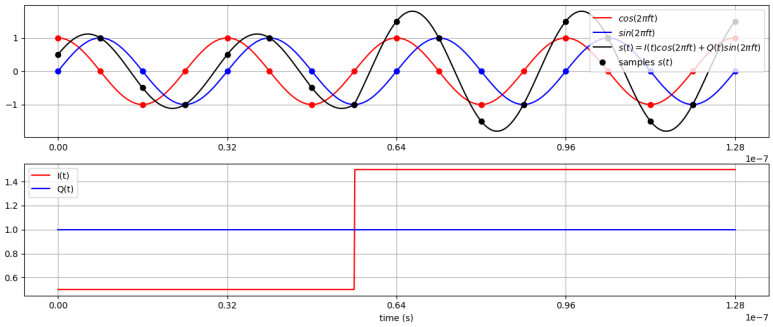


FIGURE 22. Example of a signal with a varying I/Q value.

Fig. (22) gives an example of a signal with a varying IQ value:

In general, I/Q can be recovered from s as (see T. Schilcher, 2008, RF applications in digital signal processing, p. 257)

$$\begin{cases} Q_k = \frac{2}{N}\sum_{l=0}^{N-1} s_{k-l}\sin\left(2\pi f_c \tau(k-l)\right) \\ I_k = \frac{2}{N}\sum_{l=0}^{N-1} s_{k-l}\cos\left(2\pi f_c \tau(k-l)\right), \end{cases} \tag{6}$$

where we use a least squares argument, and the quadrature property of the sin and cos parts. This simplification relies on the assumption that the N considered samples span (a number of) complete cycles of the carrier: i.e. $N \geq 4$. One can deviate from this restriction, but one needs then to pre-compute the (inverse of the) *cos*/*sin* conversion matrices. Here N is tuned properly. This approach is sometimes referred to as near I/Q sampling. In case the carrier frequency runs at 1/4 part the sampling rate (i.e. one has 4 samples per cycle of the carrier, and $N = 4$), one can simplify considerably.

4.3. **Two-sample I/Q reconstruction.** We consider an approach based on only 2 successive signal values (L. Doolittle, 2008). This approach implies less latency, is more transparent and less reliant on the in-quadrature property, at the price of more signal perturbation. Assume that $I_k \approx I_{k-1}$ and $Q_k \approx Q_{k-1}$ for any k. Consider that samples s_k and s_{k-1} correspond with angles $t_k = k\tau$ and $t_{k-1} = (k-1)\tau$ respectively. Then

$$\begin{bmatrix} s_{k-1} \\ s_k \end{bmatrix} = \begin{bmatrix} \cos\left(2\pi f t_{k-1}\right) & \sin\left(2\pi f t_{k-1}\right) \\ \cos\left(2\pi f t_k\right) & \sin\left(2\pi f t_k\right) \end{bmatrix} \begin{bmatrix} I_{k-1} \\ Q_{k-1} \end{bmatrix}, \tag{7}$$

or

$$\begin{bmatrix} I_{k-1} \\ Q_{k-1} \end{bmatrix} = \begin{bmatrix} \cos\left(2\pi f t_{k-1}\right) & \sin\left(2\pi f t_{k-1}\right) \\ \cos\left(2\pi f t_k\right) & \sin\left(2\pi f t_k\right) \end{bmatrix}^{-1} \begin{bmatrix} s_{k-1} \\ s_k \end{bmatrix}. \tag{8}$$

The inverse of a 2×2 matrix has a closed form solution so that

$$\begin{bmatrix} \cos(2\pi f t_{k-1}) & \sin(2\pi f t_{k-1}) \\ \cos(2\pi f t_k) & \sin(2\pi f t_k) \end{bmatrix}^{-1} = c_k \begin{bmatrix} \sin(2\pi f t_k) & -\sin(2\pi f t_{k-1}) \\ -\cos(2\pi f t_k) & \cos(2\pi f t_{k-1}) \end{bmatrix}, \tag{9}$$

with

$$c_k = \left(\frac{1}{\cos(2\pi f t_{k-1}) \sin(2\pi f t_k) - \sin(2\pi f t_{k-1}) \cos(2\pi f t_k)} \right) = \left(\frac{1}{\sin(2\pi f \tau)} \right) = c, \tag{10}$$

which is a constant independent from sampling index k. This follows from a trigonometric identity. In summary:

$$\begin{cases} I_{k-1} = c\left(\sin(2\pi f t_k)\, s_{k-1} - \sin(2\pi f t_{k-1})\, s_k\right) \\ Q_{k-1} = c\left(\cos(2\pi f t_{k-1})\, s_k - \cos(2\pi f t_k)\, s_{k-1}\right), \end{cases} \tag{11}$$

where c is a constant. In case $f\tau = \frac{1}{4}$, traditional I/Q sampling is recovered and $c = 1$.

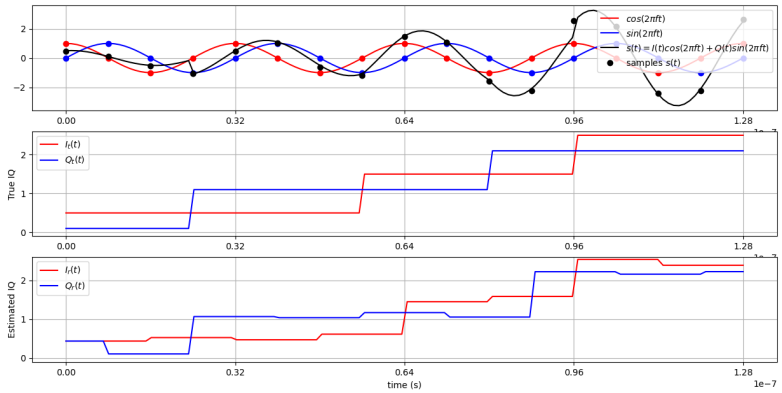


FIGURE 23. *Example of a signal with a varying I/Q value where I/Q is recovered via above argument. Here white noise with a SNR of 10 is added to the samples.*

4.4. **Synchronization.** In our particular case, the pulses will be fed with a carrier having a various initial phase offset. Figure (24) displays an example of two sampled pulses with 4s between, as sampled using exactly the same hardware/firmware settings. Note that the initial phase offset is adrift. This means e.g. that full IQ sampling (hitting the cycle at phase *exactly* equal to $0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}$) is not going to work.

However, the REF signal provides a reliable signal absolutely stable across pulses. That means that when a pulse trigger comes, the value of the REF signal

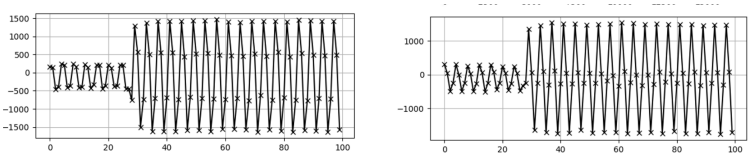


FIGURE 24. *Two readings of a pulse 4s apart. Note that the 'small signal' before the actual pulse (a delay of 28 samples) is due to path mismatching.*

indicates the phase of the carrier at that instance. This value can then be used to compensate the phase offset by rotating the carrier accordingly. We describe below how this works.

Rather than performing this explicitly, we integrate this phase offset compensation within our 2sample reconstruction scheme. It is convenient to use complex notation with $\Re(\cdot)$ representing the real part, and $\Im(\cdot)$ the imaginary part of its argument. In this case, the signal is represented as

$$s(t) = \Re\left(e^{j\omega t + j\phi}\right) = \Re\left(e^{j\phi}\ e^{j\omega t}\right). \tag{12}$$

Comparison to the IQ representation of equation (3) gives

$$\begin{cases} Q(t) = \Re\left(e^{j\phi}\right) \\ I(t) = -\Im\left(e^{j\phi}\right), \end{cases} \tag{13}$$

as $\Re\left(e^{j\omega t}\right) = \cos(\omega t)$ and the in-quadrature $\Im\left(e^{j\omega t}\right) = \sin(\omega t)$, and from the rules of multiplication of complex numbers $(a+jb)(c+jd) = (ac-bd)+j(ad+bc)$. Now, let ϕ_i denote the random phase offset of pulse i. Since $\sin\left(\theta - \frac{\pi}{2}\right) = -\cos(\theta)$ or $\Im(e^{j\theta}) = -\Re\left(e^{j\left(\theta-\frac{\pi}{2}\right)}\right)$, it follows that the I and Q values according to the random phase offset are given as

$$I(\phi_i) = -\Im\left(e^{j\phi_i}\right) = \Re\left(e^{j\phi_i - j\frac{\pi}{2}}\right) = \Re\left(e^{-j\frac{\pi}{2}}\ e^{j\phi_i}\right), \tag{14}$$

and assuming a sampling rate of 4ω gives $I_k = Q_{k-1}$. Assume that we have a reference signal r with constant I and Q values. This then implies that a signal with a random phase offset of every pulse can be tied to this r by compensating the observed IQ values of the observed signal with r_k and r_{k-1} respectively.

In practice, we can observe the reference signal

$$r(t) = \Re\left(e^{j\omega t}\right), \tag{15}$$

with unspecified initial condition $t = 0$. We also observe the relevant signal as

$$s(t) = \Re\left(e^{j\omega t + jm(t)}\right). \tag{16}$$

with m denoting the relevant information as modulated on the carrier. The question now reads as how to find the compensated signal s^c defined as

$$s^c(t) = \Re\left(e^{j\omega(t-t_i)+jm(t)}\right) = \Re\left(e^{-j\omega t_i}\ e^{j\omega t + jm(t)}\right), \tag{17}$$

with t_i indicating the time the ith pulse starts. Expanding this product gives

$$\begin{aligned} s^c(t) &= \Re\left(e^{j\omega t + jm(t)}\right)\ \Re\left(e^{-j\omega t_i}\right) - \Im\left(e^{j\omega t + jm(t)}\right)\ \Im\left(e^{-j\omega t_i}\right) \\ &= \Re\left(e^{j\omega t + jm(t)}\right)\ \Re\left(e^{-j\omega t_i}\right) + \Im\left(e^{j\omega t + jm(t)}\right)\ \Im\left(e^{j\omega t_i}\right) \\ &= r(t_i)s(t)\ +\ r\left(t_i - \frac{\pi\omega}{2}\right) s\left(t - \frac{\pi\omega}{2}\right). \end{aligned} \tag{18}$$

Note that this expression relies *not* on any particular frequency or sampling rate. On the other hand, if the sampling rate were exactly $4\frac{\omega}{2\pi}$, the discrete time version becomes

$$s^c_k = r(t_i)s_k\ +\ r\left(t_i - \frac{\pi\omega}{2}\right) s_{k-1}, \tag{19}$$

as before.

5. Mathematical models and descriptions

5.1. **Pulse compressor.** Any cavity of a pulse compressor is adequately described as an inductor, a capacitor and a resistor (LCR) circuit (Woolley and Syratchev, 2017). The voltage equals the real part of the complex-valued function V, which solves

$$\frac{1}{L}\int V(t)dt + C\frac{dV(t)}{dt} + \left(\frac{1}{R_s} + \frac{1}{Z_{wg}}\right)V(t) = \frac{2F}{Z_{wg}}e^{j\omega t}, \tag{20}$$

where Z_{wg} is the impedance of the transmission line into the cavity, R_s is the cavity shunt impedance, and F is the phasor of the forward wave into the cavity. The phasor F is typically given by the application. The phasor R of the outgoing wave at the port of the cavity is determined as the solution to

$$V(t) = (F(t) + R(t))\, e^{j\omega t} \tag{21}$$

The outgoing wave is the sum of reflected and emitted wave.

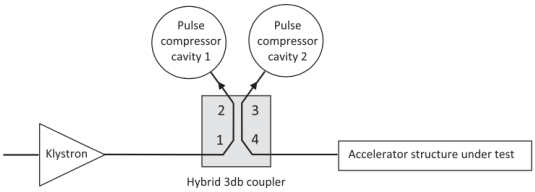


FIG. 1. Pulse compressor layout.

FIGURE 25. *Schematic layout of a SLED-I pulse compressor.*

Using reparameterisations

$$\begin{cases} \omega_0 = \frac{1}{\sqrt{LC}} \\ Q_0 = \omega_0 R_s C \\ \frac{1}{Q_L} = \frac{1}{Q_0} + \frac{1}{Q_e}, \end{cases} \tag{22}$$

one gets the more familiar 2e order system description

$$\frac{d^2V(t)}{dt^2} + \left(\frac{\omega_0}{Q_L}\right)\frac{dV(t)}{dt} + \omega_0^2 V(t) = \left(\frac{2\omega_0}{Q_e}\right)\frac{d}{dt}\left(F(t)e^{j\omega t}\right). \tag{23}$$

with F the phasor describing the input signal.

5.2. **Waveguide.** A waveguide guides an electromagnetic wave over a distance in a material. Description starts with the wave equation for electric field $E(t)$ given as

$$c_0^{-2}\frac{\partial^2 E(t)}{\partial t^2} - \nabla^2 E(t) = 0, \tag{24}$$

where $c_0 = 2.998 \times 10^8$ for vacuum (or $c_0 = 1/\sqrt{\mu\epsilon}$ otherwise, with permeability μ and permittivity ϵ of the material inside the waveguide).

5.3. **SSPA.** Figure (26) displays a first characterisation of the S-band power amplifier.

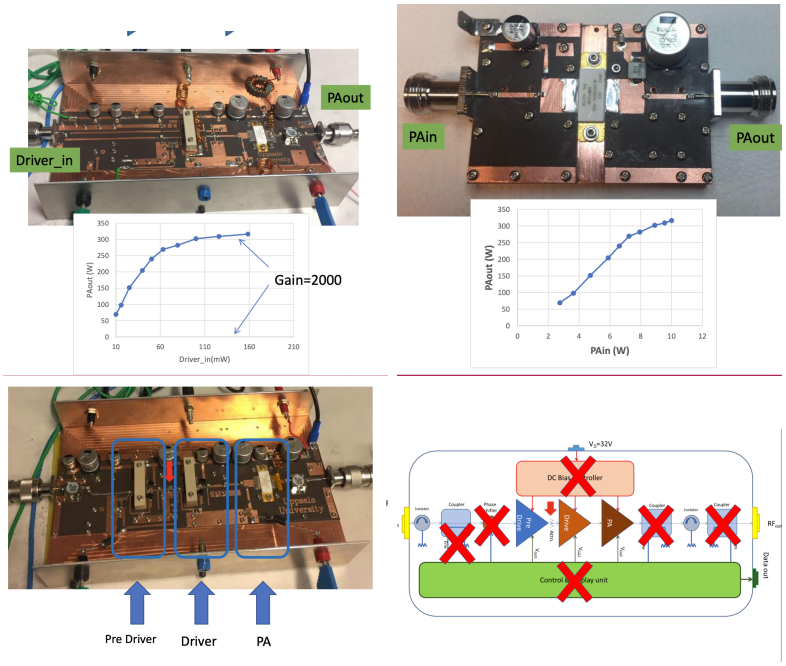


FIGURE 26. *Characterisation of the solid state power amplifier (SSPA) for the S-band (3GHz). (a) $P_{in} - P_{out}$ characterisation of the SSPA (without pre-driver). (b) PA element. (c) Layout (including the pre-driver embedding). (d) Schematic.*

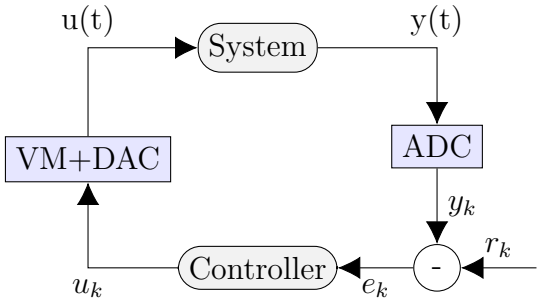
5.4. **Klystron+K100.** (ScandiNova) Klystron based RF unit. Designed to provide RF peak power in the 3 MW to 10 MW range, it provides unparalleled performance and ease of integration and use by customers. Can be provided as a pure pulse modulator all the way to turn-key RF System. Below are specifications and data sheets for the most common configurations based on frequency and type of klystron.

RF peak power	3 – 10 MW
RF average power	10 kW (max)
Pulse voltage	115 – 190 kV
Pulse current	90 – 140 A

TABLE 3. Specifications of the K100 pulse modulator (Scandinova), powered by a Klystron vacuum tube.

6. Automatic control algorithms

The general control loop looks as follows:



From the above discussion, we see that there are 6 output signals $y(t)$, while there is only a single input signal $u(t)$. For the single input single output (SISO) control loops, we work only with the output signal on BC2.fwd (see Fig. (7)).

There are two main effects:

- FF Repeatable pulse-to-pulse disturbances are handled by feedforward (FF) control.
- FB Non-repeating external disturbances (on the flat-top of the pulse) are handled by feedBack (FB) control.

In the present context, it is convenient to represent the reference signal r_k as a trigger signal. Three events are announced in this trigger signal:

- Start filling the RF field - a new pulse is coming in 15us!
- Realise the pulse now!
- Reverse phase for the pulse compressor!

see also figure (8.b).

6.1. **PID Control Loops.** The de facto standard approach is to implement two independent PI(D) loops on the I and the Q branch of the signal. The PID control loop is a SISO loop, meaning it can only handle a single input and single output signal. In this case, we use as input the VM driving the SSPA, and as output signal the reading from BC4.fwd. This SISO loop is shown in figure (27).

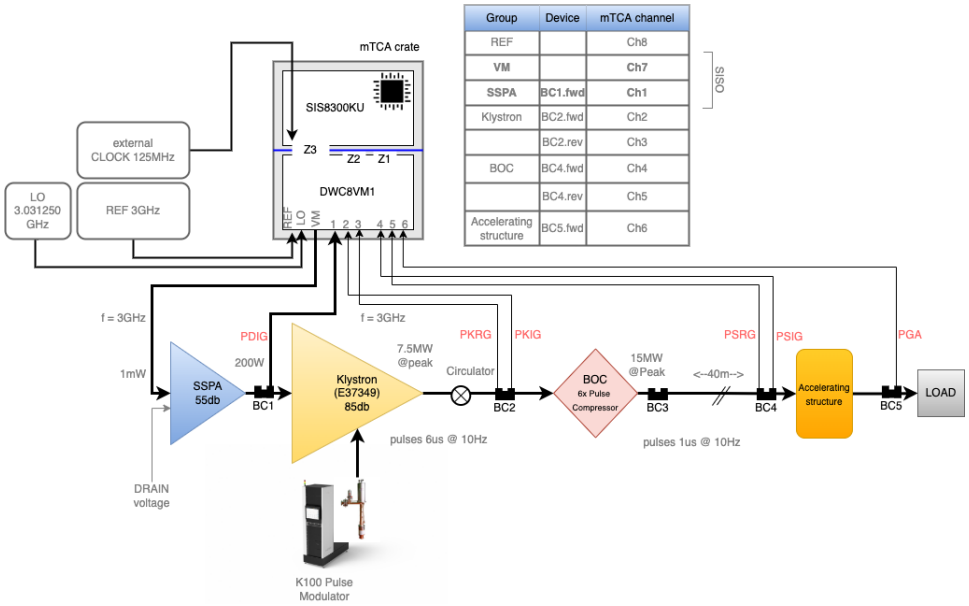
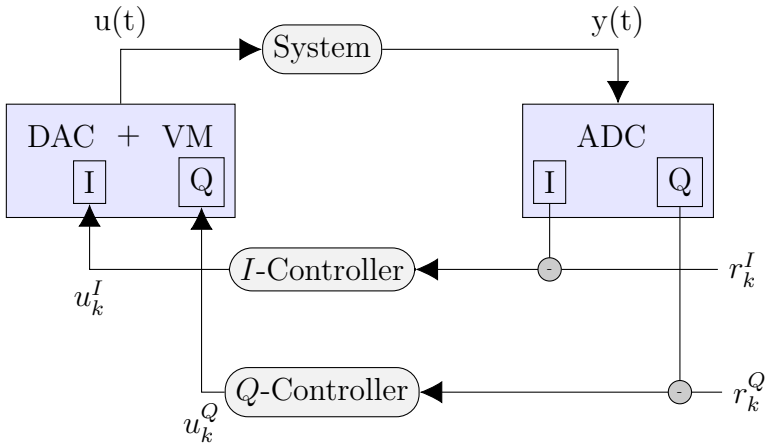


FIGURE 27. *Layout of the SISO control loops implemented by a PID controller on both I and Q arms.*

The common design is to have independent I and Q arms, denoted respectively as C^I and C^Q: For the control of a single resonating cavity, this design is close to optimal (O. Troeng 2019).



or

$$u_k = C^I \left(r_k^I - I_k\right) \cos\left(2\pi f_c \frac{k}{f_S}\right) + C^Q \left(r_k^Q - Q_k\right) \sin\left(2\pi f_c \frac{k}{f_S}\right). \tag{25}$$

Since the control blocks can implement phase shift (delay), independence of the I and Q paths is not necessary preserved. That is, the same output behaviour can

be achieved using different controller settings, resulting in ill conditioning of the problem.

Let's focus now on a single arm (say, the I-arm). The PID loop is described by the following equations. A continuous version of a PID loop is given as

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + P_d \left(\frac{de(t)}{dt} \right), \tag{26}$$

where

$$e(t) = r(t) - y(t), \ \forall t, \tag{27}$$

and $T_p, T_i, T_d \geq 0$ are parameters to be tuned. A discretised version of a PID loop is given as

$$u_k = u_{k-1} + P_0 e_{k-1} + P_2 e_{k-2} + P_2 e_{k-3}, \tag{28}$$

where

$$e_k = r_k - y_k, \ \forall k, \tag{29}$$

and P_0, P_1, P_2 are equivalent parameters to be tuned (i.e. there exists a one-to-one mapping from P_p, P_i, P_d to P_0, P_1, P_2 depending on the used discretisation scheme).

6.2. **Anti-windup filter.** Anti-windup filters provide a standard way to handle saturation effects for avoiding overly large integrator terms. An implementation of the familiar Smith anti-windup approach is included in the firmware.

6.3. **Feedforward Tables.** A considerable control effort is implemented in a feedforward way - i.e. in a pulse-to-pulse way. Especially, the EM field is build up by implementing pre-recorded actions when needed. That is, if the bunch pre-warning messages arrives, this table will be implemented at u. FeedBack serves as an add-on to handle non-repetitive (stochastic) noise/disruptions.

A pulse of 6 us spans 750 samples (at 125 MS/S). A filling of 600 ns spans an extra 75 samples. So a pulse is spanned well by a sequence of 1024 samples, starting roughly 1us in advance.

The setup of a feedforward (FF) controller is displayed below. Here, the real-time controller is denoted as C. The system to be controlled is denoted as P ('Plant'), mapping input u_k to output y_k. There is no *direct* feedBack in this case.

However, in the present case it is more convenient to represent FF control as a table of values to be injected when a new pulse will be made (at 10Hz).

6.4. **Iterative Learning Control.** The feedforward table are adjusted by an adaptive pulse-to-pulse scheme. This scheme corresponds roughly to iterative learning control (ILC). Let $[k]$ denote the pulse iteration of sample k. That is, if the iteration takes $n > 0$ samples, then $[k] = \lfloor \frac{k}{n} \rfloor$ (also referred as the quotient). The remainder is given as $(k) = k \mod n$, indicating the index in the present iteration to which the kth sample relates. For example, let an iteration be spanned by $n = 10$ samples. Then sample $k = 25$ would fall in the $[k] = 2$e iteration, on the $(k) = 5$e place within this 2e iteration.

Then the main ILC approach can be denoted as

$$u_{[k],(k)} = u_{[k]-1,(k)} + \boxed{P_0 e_{[k]-1,(k)} + P_1 e_{[k]-2,(k)} + P_2 e_{[k]-3,(k)}}. \tag{30}$$

Here, the content of the framed box indicates the *update* of the feedforward table from the previous pulse. In case of a fixed number of n samples in an iteration, one has

$$\begin{cases} u_{s,i} = u_{ns+i} \\ e_{s,i} = r_{ns+i} - y_{ns+i}, \end{cases} \tag{31}$$

where s indexes *iteration*, and i the *sample* in this iteration. Again, P_0, P_1, P_2 are to be tuned properly.

7. Control firmware

7.1. **Control loop.** As indicated before, the control loop implements the stages as displayed in Figure (28): The key idea is to separate within-pulse and pulse-to-

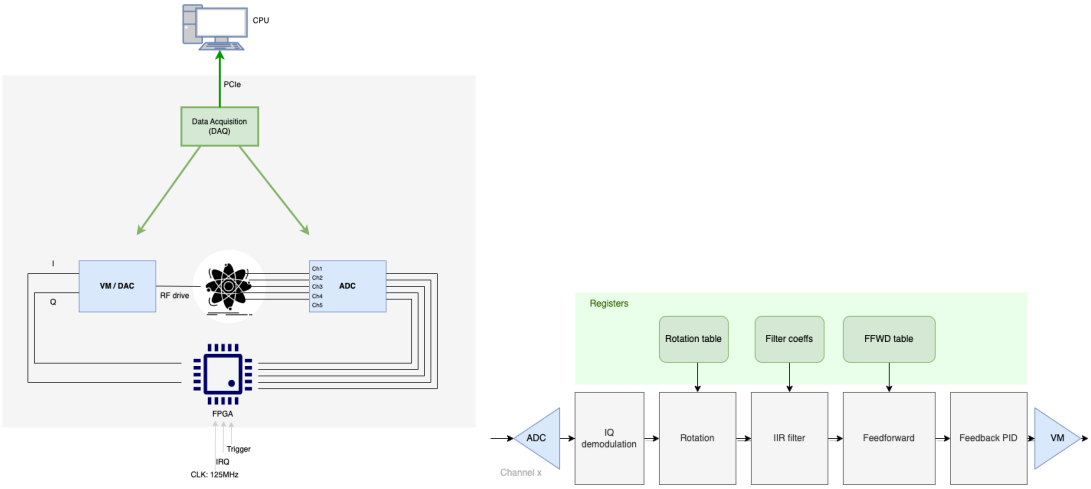


Figure 28. *Schematic overview of the implemented feedback loop.*

pulse components out of the signal s. Then, either component is addressed with their appropriate feedback loop.

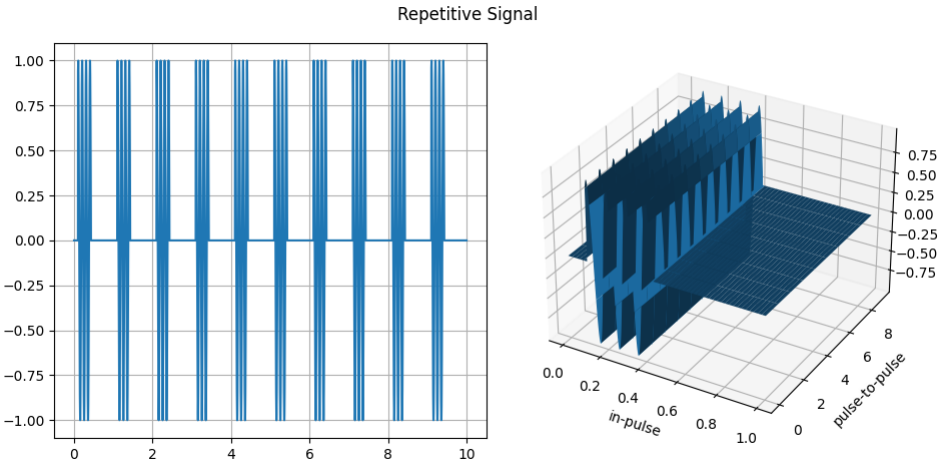


Figure 29. *A regularly pulsed signal (left panel) can be folded in a multivariate fashion in Pulse-to-Pulse and within-pulse dimensions (right panel).*

7.1.1. *Within-Pulse feedback.*

7.1.2. *Pulse-to-Pulse feedback.*

8. Numerical representation

We use 64 bit fixed point numerical representations, with the final result truncated to 16 (15+1) bit integers. The fixed point representation is $s32.32$, or 1 sign bit (2-complement), 32 integer bits, and 32 fractional bits. There are at least three good reasons for this decision:

- Future DACs will have a resolution exceeding the current 16 bit, but we like the development to handle those cases seamlessly.
- Since the loops are fairly complex, numerical errors will accumulate quickly.
- We like to be fairly robust towards changing dynamical range.

The firmware developments on top of the BSP consists of the following modules ('IP cores').

9. IP cores

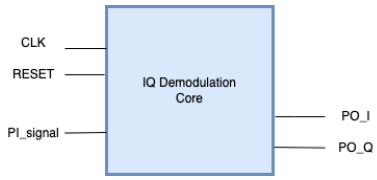


Figure 30. Overview of the I/Q demodulation core.

9.1. **IQ demodulation core.**

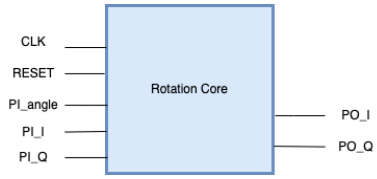


Figure 31. Overview of the core implementing a rotation table.

9.2. **Rotation core.**

9.3. **IIR filter core.**

9.4. **ILC Control core.**

9.5. **PID core.**

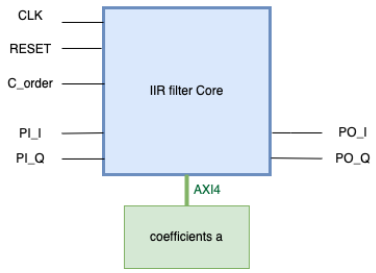


FIGURE 32. Overview of the core realising an Infinite Impulse Response (IIR) filter.

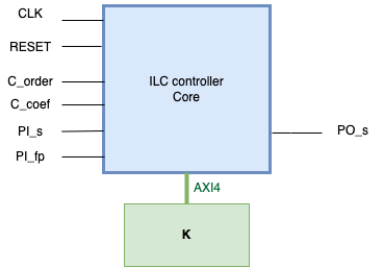


FIGURE 33. Overview of the core implementing the Iterative Learning Control (ILC) pulse-to-pulse control loop.

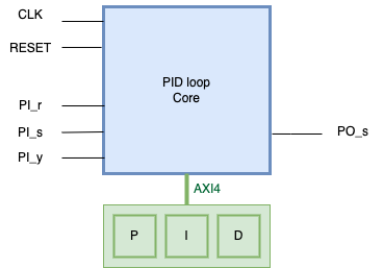


FIGURE 34. Overview of the core implementing a PID feedback control loop

10. Graphical User Interface

The central way to interact with this firmware is through a graphical user interface (GUI) connected to the COMex and the firmware via PCI express. This interface allows to get all samples from the DAQ (via Direct Memory Access, or DMA), and to change settings in the memory map. A key design decision is to wait for explicit events (click on 'Update!') for acquiring new signals to be fed to the GUI: there is no continuous event system as to keep the GUI lightweight.

Figure (35) displays an example of a 5MHz carrier signal, where 25 samples are made per cycle. The GUI shows the 8 channels for a fixed pair SIS8300ku+DWC8VM1

in a specified slot of the crate. Since the software is run on the COMex (MCH-RTM), the selection of card need only be made on the software side (in the according .dmap file).

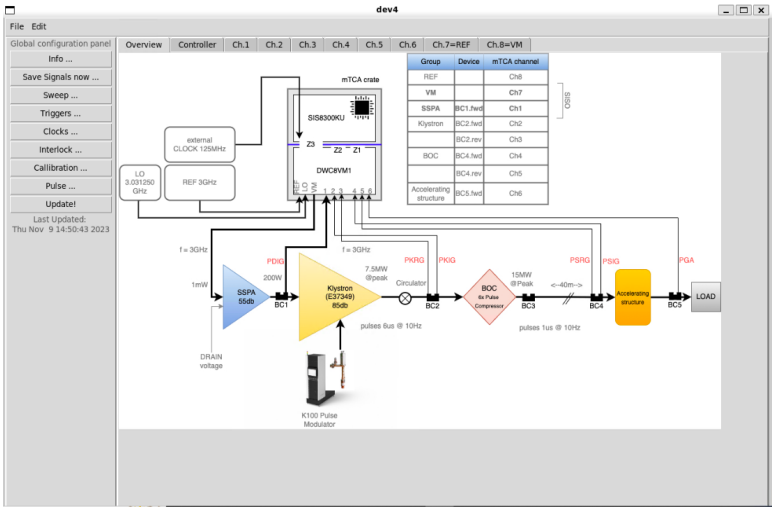


FIGURE 35. *The GUI developed to monitor and setup the firmware project.*

10.1. **Software interface to firmware.** The high-level (but slow) software is run on the COMex (Python3.8.10 on Ubuntu 20.04 TLS, running on Xeon on the MCH-RTM), accepting information (of the DAQ) via PCI express. The interface is served by the package mtca4u.py, which is a Python wrapper of the DESY-developed framework.

11. Conclusion

This development implements a low level (LLRF) control system for use in a plasma-driven particle accelerator (AWAKE) project in CERN. The development uses the MicroTCA hardware, and builds further on the framework developed by DESY. Practical operation of the development is enabled via a GUI (written in Python), running on the linux system integrated in the MicroTCA crate on the rear (MCH-RTM). The aim is to integrate this LLRF system in the overall supervisory control (SCADA) framework as used in CERN (i.e. the socalled FESA system) or as used elsewhere (e.g EPICS).

12. Research challenges

- Pulsed Control is a fascinating, and underdeveloped control paradigm. The approach taken here - integrating feedback schemes with pulse-to-pulse (learning) control, provides a practical solution but is riddled by lacking theoretical understanding. Especially questions regarding long-term control stability of the integrated system are wide open. However, since the system at hand is fast but simple (second order), those questions do not imply any practical challenges as yet.
- Fast ADC: it is possible to achieve effectively high sampling rates (>1GSPS) by using a suitable combination of slow, cheap ADC sampling hardware. This is know as the emerging paradigm of 'sub-Nyquist' sampling. The most direct way is to use interleaving ADC samplers, providing "only" a linear increase. For example, interleaving 4 ADCs of 125MSPS will give one effective fast ADC of 500 MSPS. This enables direct sampling (DS) mechanisms, rather than the need to down convert the signals to lower frequencies (IF) with implied constraint bandwidths.