TESLA Report 2005-20

8-Channel, FPGA based, DSP Integrated Cavity Simulator & Controller for VUV-FEL

SIMCON 3.0

Ver. 3.0. rev. 1, 06.2005

HARDWARE MANUAL

Krzysztof T. Pozniak, Tomasz Czarski, Waldemar Koprek, Wojciech Giergusiewicz, Ryszard S. Romaniuk

ELHEP Group, http://www.desy.de/~elhep
Institute of Electronic Systems, Warsaw University of Technology
Nowowiejska 15/19, 00-665 Warsaw, Poland

ABSTRACT

The note describes integrated, eight channel system of hardware controller and simulator of the resonant superconducting, narrowband niobium cavity, originally considered for the TTF and TESLA in DESY, Hamburg (now tested for the VUV FEL and developed for X-Ray FEL). The controller bases on a programmable circuit Xilinx VirtexII V4000. The solution uses *DSP EMBEDDED BOARD* module positioned on a Modular LLRF Control Platform. The algorithm and FPGA circuit configuration was done in the VHDL language. The internal hardware multiplication components, present in Virtex II chips, were used, to improve the floating point calculation efficiency. The implementation was achieved of a device working in the real time, according to the demands of the LLRF control system for the TESLA Test Facility (now associated with the VUV FEL machine). The device under consideration will be referred to as superconducting cavity (SCCav) SIMCON throughout this work. The manual describes hardware features of SIMCON, ver. 3.0 in modular solution.

The following components are described here in detail: functional layer, parameter programming, foundations of control of particular blocks and monitoring of the real time processes. This note is accompanied by the one describing the multichannel DOOCS interface for the described hardware system. The interface was prepared in DOOCS for Solaris and in Windows. The hardware and software of 8-channel SIMCON was tested in CHECIA and ACC1 module of VUV FEL linac. The measurements results are presented.

While giving all necessary technical details required to understand the work of the integrated hardware controller and simulator and to enable its practical copying, this document is a unity with other TESLA technical notes published by the same team on the subject. Thus, some modeling and other subjects were omitted, as they were addressed in detail in the quoted references.

Keywords: Super conducting cavity, cavity simulator, CAVITIES CONTROLLER, SIMCON system, linear accelerators, FPGA, FPGA-DSP enhanced, VHDL, FEL, TESLA, TTF, UV-FEL, Xilinx, FPGA based systems, LLRF control system of third generation, electronics for UV-FEL, X-Ray FEL and TESLA.

MAJOR CHANGES FROM

SIMCON

VERSION 1.0, REV. 1, 04.2004

PUBLISHED AS TESLA FEL REPORT 2004-04

HTTP://TESLA.DESY.DE/NEW_PAGES/FEL_REPORTS/2004/FEL2004-04.PDF

- Automatic switching of parameters and tables was added. The switching enables control parameter changes without stopping, resetting or switching off the SIMCON.
  This change was presented in chapter 8.2.3.,

- An input compensation matrix was added to the algorithm of the CAVITIES CONTROLLER. This change was described in chapter 8.2.5.,

- The access rules to the tables have been changed.
  This change was described in chapter 4.2.

- Two options for the voltage range of the cavity Simulator were introduced.
  This change was described in chapter 8.2.4.

- The dimensions in bits of the tables for amplification values for CAVITIES CONTROLLER and beam current values for cavity simulator were equalized. The equalization enables realization of automatic switching mechanism.
  This change was described in chapter 8.2.3.

- An additional VMEbus interface was added.

- A number of representative exemplary results from SIMCON tests with CHECHIA were addend.
  The results are presented in Appendix D.

- Many tables and figures were updated and renumbered throughout the whole text.

MAJOR CHANGES FROM

SIMCON

VERSION 2.1, REV. 1, 02.2005

PUBLISHED AS TESLA REPORT 2005-06

HTTP://TESLA.DESY.DE/NEW_PAGES/TESLA_REPORTS/2005/PDF_FILES/TESLA2005-06.PDF

- A universal modular LLRF platform was used (PCB Backbone Motherboard, BMB). The platform is equipped in VME controller, embedded ETRAX processor (*100 MHz*) with *100Mb ETHERNET* link and proprietary bus INTERNAL INTERFACE. This module is described in chapter 3.1, and in appendix A.
- The superconducting cavity control algorithm was extended from a single channel to eight channels. 8 channel algorithm is described in chapter 2.2.
- 8-channel DSP module was implemented. This module is described in chapter 3.1, and in appendix B.
- The system implementation was realized for the chip FPGA *VIRTEXII-V4000-4*. The implementation was described in chapter 3.2.
- There was implemented a nondependent service of 8 input channels. The implementation is described in chapter 6.
- The calibration was added for input channels. This feature was described in chapter 7.
- There were realized nondependent I/Q demodulators for each channel. This feature was described in chapter 8.2.2.
- The block of memory switching was modified. This modification is described in chapter 8.2.3.
- *There is not implemented the service and processing of demodulated I and Q signals.*
- There was implemented the control algorithm of control for 8 cavities. This feature is described in chapter 10.
- There was introduced an individual readout of I and Q signals after detection. This feature is described in chapters 10.2, 13.2.
- The readout of the vector sum for the I and Q was added. The vector sum is described in chapters 10.2, 13.2.
- The readout of all input analog channels is made accessible. This feature is described in chapter 13.2.
- There was modified the functional structure of the input multiplexer. This feature was described in chapter 12.
- Exemplary control results of the ACC1 module of S.C. linac for the VUV FEL were added. They were presented in appendix D.

CONTENTS

1 INTRODUCTION

The TESLA – VUV-FEL/X-FEL project bases on the nine-cell super conducting niobium resonators to accelerate electrons and positrons. The acceleration structure is operated in standing p-mode wave at the frequency of 1,3 GHz. The RF oscillating field is synchronized with the motion of a particle moving at the velocity of light across the cavity.

The LLRF – Low Level Radio Frequency control system has been developed to stabilize the pulsed accelerating fields of the resonators (see fig. 1). The prospective FPGA technology solution is projected for the digital implementation of the cavity control system algorithm.

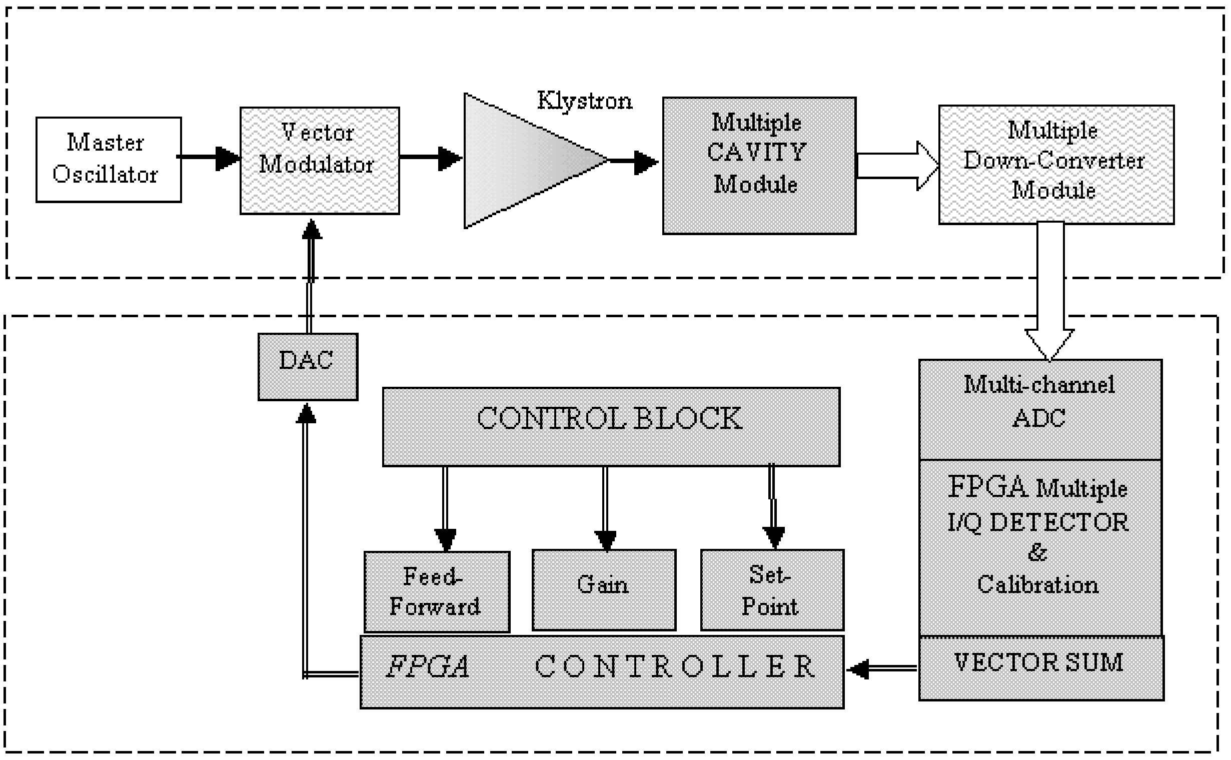


Fig. 1. Functional block diagram of LLRF Cavity Control System

The control section, powered by one klystron, may consists of many cavities (see fig. 2). One klystron supplies the RF power to the cavities through the coupled wave-guide with a circulator. The fast amplitude and phase control of the cavity field is accomplished by modulation of the signal driving the klystron from the vector modulator. The cavities are driven with the pulses of 1.3 ms duration and the average accelerating gradients of 25 MV/m. The RF signal of each cavity is down-converted to an intermediate frequency of 250 KHz preserving the amplitude and phase information. The ADC and DAC converters link the analog and digital parts of the system.

Input data integration idea bases on 8-channel ADC board with FPGA concentrator and fast fiber optic data transmission in future. So, the careful multi-channel solution is required for the OPTO and PCB design. Alternatively, the first FPGA module may include also the initial digital signal processing applied for the field vector detection (I/Q detector) and calibration of each cavity signal. Furthermore, the vector sum of multiple signals is considered for the actual control processing. In case of many cavities the several 8-channel boards are applied for the initial data processing. So, the reduced data of low frequency are transmitted to the main DSP controller board which completes the vector sum of received partial data.

The control feedback system regulates the total vector sum of the pulsed accelerating fields in multiple cavities. According to the desired set point, the digital controller stabilizes the average value of the envelope detected as real (in-phase) and imaginary (quadrature) components of the incident wave. Additionally, the adaptive feed-forward is applied to improve compensation of repetitive perturbations induced by the beam loading and by the dynamic Lorentz force detuning. The control block generates the required data of Set-Point, Fed-forward and Gain for the internal memory of the FPGA based controller (see fig. 2).

The comprehensive digital system modeling has been developed for the investigation of the optimal control method for the super conducting cavity. The design of a fast and efficient digital controller is a challenging task and it is an important contribution to the optimization of the TESLA – VUV-FEL/X-FEL accelerator.

2 CAVITY SIMULATOR AND CONTROLLER ALGORITHM

The cavity resonator modeling has been developed for the efficient testing of the control system and for the investigation of the optimal control method. The FPGA hardware implementation of the cavity model is intended for the real time operation.

2.1 Cavity simulator algorithm

The cavity electromechanical model including Lorentz force detuning and the beam loading is applied for analyzing the basic features of the plant. The cavity control system proceeds within the low-level frequency range of the *complex envelope* for the input current and output voltage of the cavity. The *complex envelope* signal is represented by real (I – in-phase) and imaginary (Q – quadrature) components. The discrete processing of the cavity behavior has been developed for the digital implementation of the cavity model. The functional diagram of the cavity simulator algorithm is presented in fig. 2.

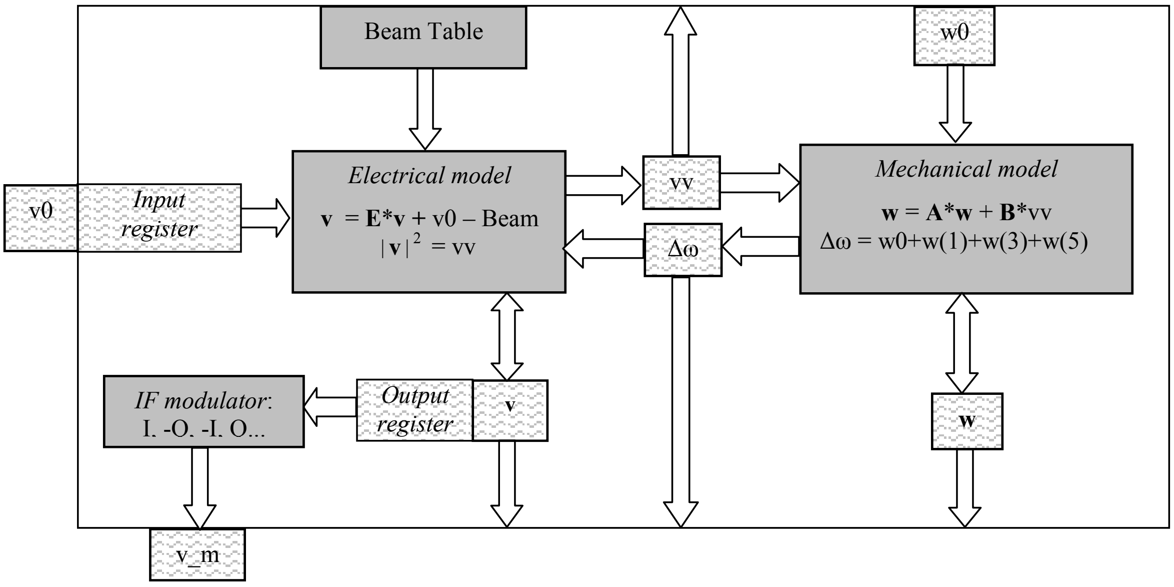


Fig. 2. The functional diagram of the cavity simulator algorithm

The electrical part of the cavity simulator consists of the DSP function block. The DSP procedure is realized according to the *state space* relation with the state vector **v** representing (I, Q) components of the cavity output *envelope*. The system matrix **E** depends on the cavity detuning Δω and the cavity bandwidth only. The normalized current generator as the input signal $\mathbf{v_0}$ and the beam from the table drives the DSP unit. Additionally, the non-stationary detuning Δω modulates the object feature by the matrix **E**. The square of the cavity field gradient $|\mathbf{v}|^2$ = vv drives the mechanical part of the model. The input and output registers correspond to the time delay of the cavity environment (waveguide). The intermediate frequency modulator converts the cavity output vector to the signal v_m of frequency 250 kHz. Therefore, the data samples, like from the down-converter, can be conveyed to the outer digital controller.

The mechanical model of the super-conductive cavity consists of the DSP unit according to the *state space* relation with the state vector **w**. The time-varying detuning Δω and its time derivative are two state-variables for each mechanical mode. The system matrix **A** and the input matrix **B** depend on the cavity parameters: resonance frequency, quality factor and Lorentz force-detuning constant for each mechanical mode. Each of the mechanical modes is driven by the square of the cavity field gradient vv generated from the electrical part of the model. Three dominating resonance frequencies are considered in the cavity model and the superposition of all modes, together with the initial *predetuning* w_0, yield the resultant detuning $\Delta\omega$.

2.2 Cavity controller algorithm

The comprehensive model of the control system has been developed to investigate different operational conditions of the cavity. The functional diagram of the controller algorithm is presented in fig. 3.

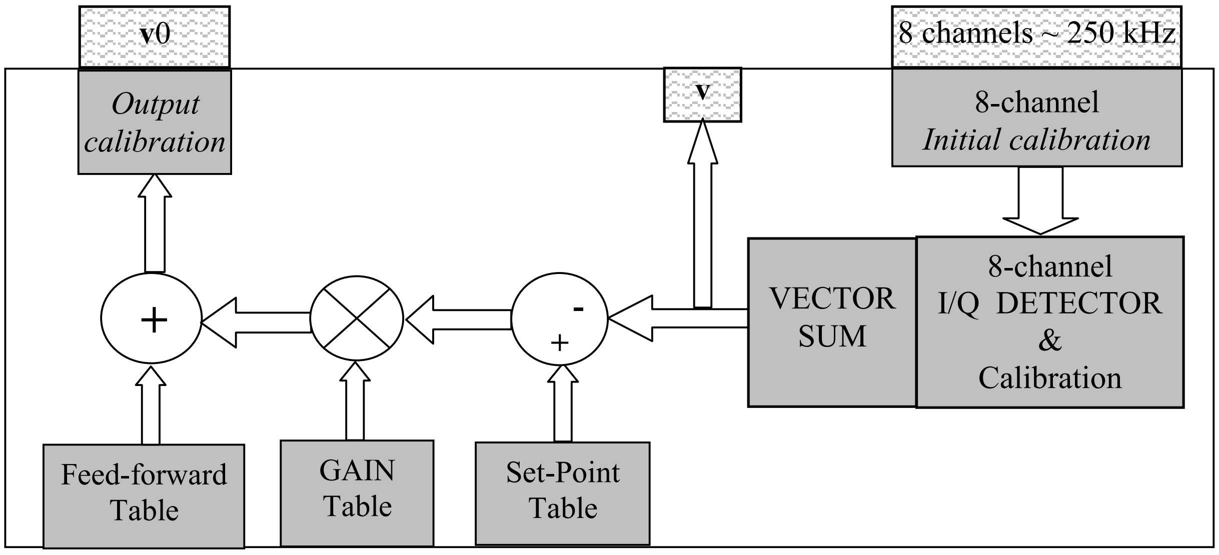


Fig. 3. The functional diagram of the controller algorithm

After initial calibration (scaling and leveling), the digital processing is performed in I/Q detector applying the signal v_m of intermediate frequency 250 kHz for 8 signal channels. The resultant cavity voltage *envelope* (I, Q) is calibrated, so to compensate the phase shifting for an individual measurement channel. The vector sum of 8 signals is considered for the actual control processing. The Set-Point table delivers the required signal level, which is compared to the actual average value of the cavities voltage envelope. The multiplier as the proportional controller amplifies the signal error according to data from the GAIN table and closes the feedback loop. Additionally the Feed-Forward Table is applied to improve compensation of the repetitive perturbations induced by the beam loading and by the dynamic Lorentz force detuning. The resultant output signal $\mathbf{v}_0$ can drive the cavity simulator.

2.3 Simulation procedure

The FPGA cavity simulator and controller are coupled to the MATLAB system via communication interface. The real time tests are carried out according to the schematic block diagram in fig. 4. The MATLAB system initiates the simulation process for the given primary parameters. The list of parameters for user utility is combining in the table below. The secondary, internal parameters required for the FPGA system are calculated in the beginning. Additionally the optimal data for Set Point and Feed Forward tables are generated according to the cavity model. Finally, the MATLAB simulation process is verified by plot. The resulting example, for the real operational condition, is presented in fig 5. The cavity is driven in the pulse mode forced by the control feedback supported by the feed forward.

Subsequently, resultant parameters and data are loaded to the FPGA memory tables. The cavity simulator and controller can be driven independently via the external connection applying the analog-to-digital converters (ADC– 14-bit resolution). Alternatively, the FPGA controller can drive the FPGA cavity simulator via internal digital connection (18-bit data resolution). Then, the FPGA system can run itself cyclically according to the given data tables (see below). The digital-to-analog converter (DAC) conveys data from the FPGA cavity simulator or from the FPGA controller outside the system.

Tables of the primary parameters for user utility.

CAVITY SIMULATOR parameters	CONTROLLER parameters
f_0 = 1300 [MHz]…………………resonance frequency	G …………………………………………….... gain
ρ =520 [Ω] ………………….. characteristic resistance	cal …….……...initial calibration vector for 8 channels
$Q_L = 3{\cdot}10^6$ ……………………..…. loaded quality factor	c_in …..……….input calibration vector for 8 channels
Δf = Δω/2π = 390 [Hz]…………..…...….pre-detuning	c_out…………………….…. output calibration vector
d1 = 0, d2 = 1 …….……………….… input, output delay	CONTROL parameters
f = [235, 290, 450] [Hz].. resonance frequencies vector	F = 1, (0) ……..……. Feed-forward enable, (disable)
Q = [100,100,100] …………….. quality factors vector	a = 25 [MV] …………...Set point for cavity amplitude
K = [0.4, 0.3, 0.2] [Hz/(MV)^2]… LFD constants vector	ph = 0 [rad] ……………….Set point for cavity phase
Ib = 8 [mA] ……………………….…average beam	D = 509 ………………………..…….filling time
D1 = 509, D2 = 1300 ……..………..…start, stop beam	L = 800 ……………….......…………flattop time

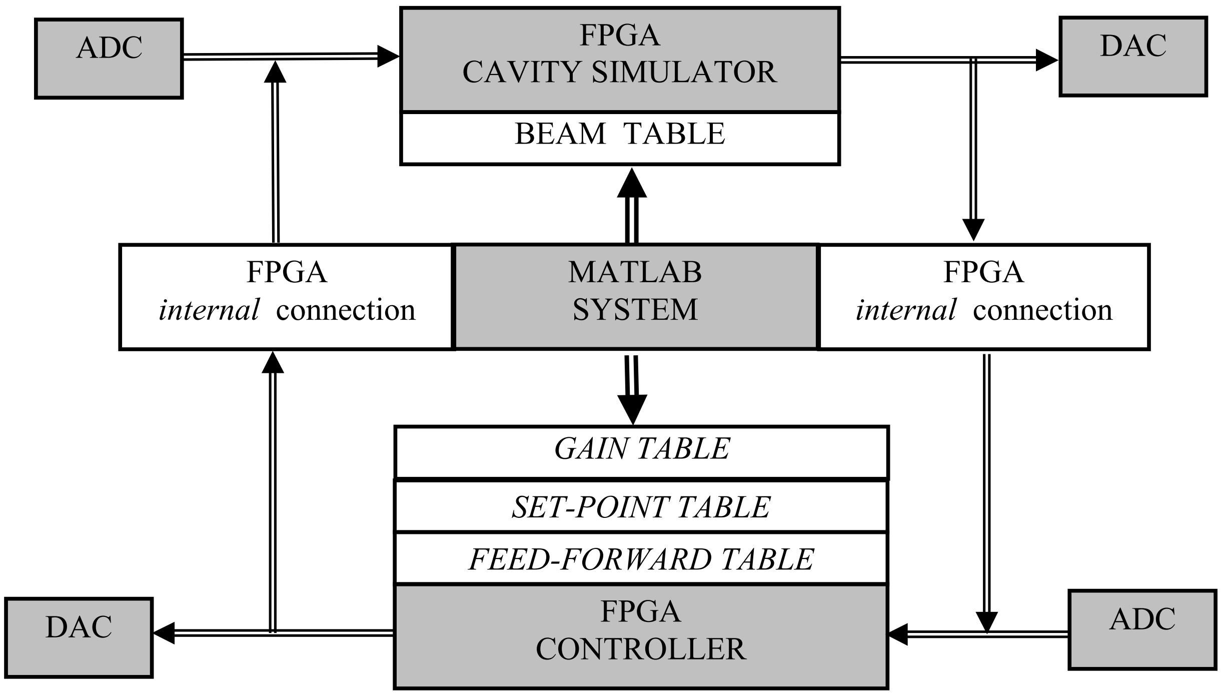


Fig. 4. Functional diagram for one chip FPGA system

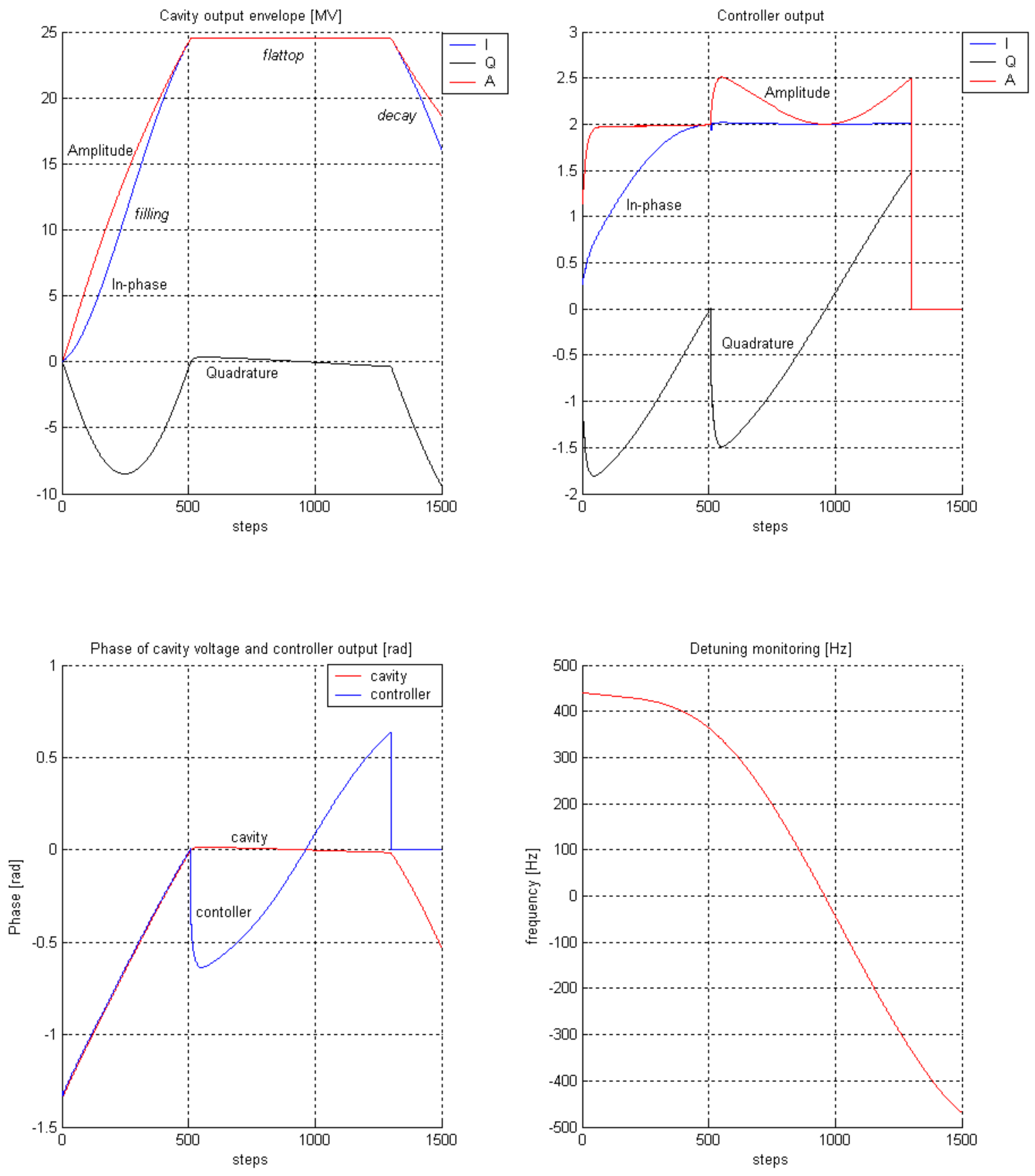


Fig. 5. The MATLAB results of simulation for real operation condition

3 GENERAL DESCRIPTION OF SIMCON SYSTEM

The integrated and parameterized controller and simulator system for the resonant, superconducting, narrowband cavity of the UV-FEL (SIMCON) was implemented in a programmable FPGA chip Virtex II V4000. The chip has inbuilt hardware DSP components [7]. This chapter presents in a general way the functional and hardware structure of the device.

3.1 Hardware structure

The hardware layer was realized with modular *DSP EMBEDDED BOARD* [11] positioned on a Modular LLRF Control Platform [8].

DSP EMBEDDED BOARD module was designed as integrated board for amplitude and phase control of the EM field for a single section of the VUV FEL accelerator. Its construction is presented in fig. 6. The *DSP EMBEDDED BOARD* module possesses:

- Eight input analog channels:
  - Each channel was equipped in nondependent ADC, *AD6645* by Analog Devices [14] of 14-bit resolution.
  - High analog input bandwidth *270 MHz* enables the work with high intermediate frequency signals *IF=81 MHz* or bigger.
  - The sampling frequency of the converter is from 50 to 100 MSPS. There is the ability to realize a hardware based, fast averaging of the I/Q vector.
  - The output range of voltage is ±*1V/50Ω*,
  - There was applied an input isolating amplifier with symmetric input to the converter, to minimize the SNR,

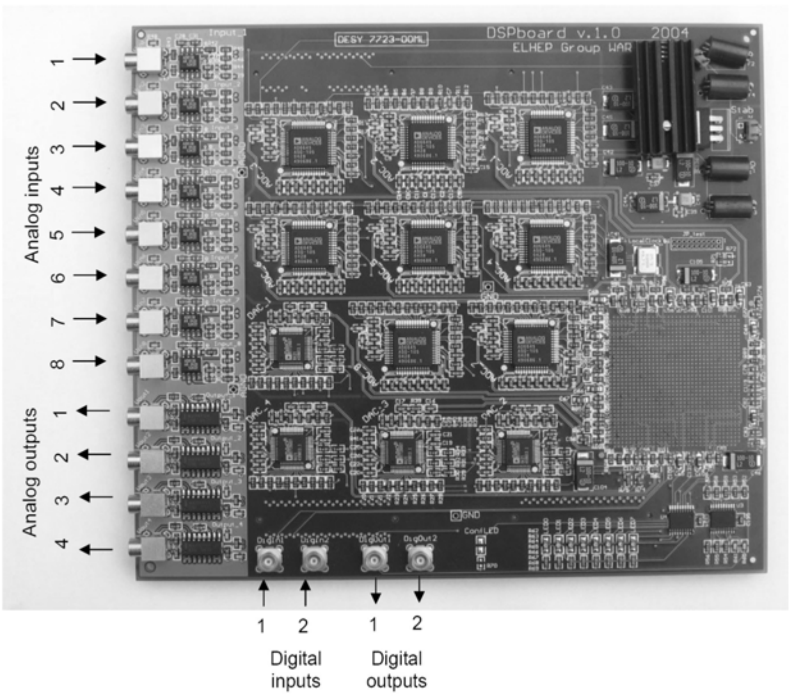


Fig. 6. DSP Embedded Board PCB

- Four analog input channels:
  - Each channel is controlled by the DAC, *AD9772A* by Analog Devices [15] of 14-bit resolution with symmetric output to minimize the SNR.
  - The frequency range of DAC is from 40 to 160 MSPS (the converter realizes additionally the hardware based signal approximation),
  - The output voltage range is *1V/50Ω*,
  - The output isolating amplifier of the following voltage range ±*1V/50Ω.*
- Two input and two output buffered digital channels in the LVTTL standard predicted to input and output the synchronization channels.

The *DSP EMBEDDED BOARD* module is realized on a single PCB board (see appendix B) embedded in two slots of the Modular LLRF Control Platform [8]. This solution makes the LLRF Platform a universal, modular and reconfigurable test ground for specialized functional modules basing on FPGA matrices. This solution enables extraction of common components of LLRF system and use them in multiple ways instead of multiplication of the same modules in separate functional block. This leads to considerable simplification of hardware layer of these repeated modules and to unification of the control from the software level. The design process embraces less variations of the functionalities and, thus, is less time consuming and cheaper. The general structure of the LLRF Modular Platform was presented in Appendix A. This solution provided:

- Standardized communication system with VME-bus [12] and via *Ethernet 100T* [10],
- Remote configuration of FPGA via the *JTAG standard* [13],
- Fast, electrical data transmission buses for connections with other embedded modules, for example with the OPTO module [9] which provided optical data transmission with the rate 1.6Gb/s,
- Distribution of common clock and synchronization signals, to enable synchronous work of several embedded modules on a single LLRF System Modular Platform,
- Necessary supply voltages for FPGA chips, converters and digital buffers.

The module *DSP EMBEDDED BOARD* positioned on the Modular Control Platform occupies two clots in the VME-6U crate. The power supply is provided via the VMEbus. A complete system is presented in fig. 7.

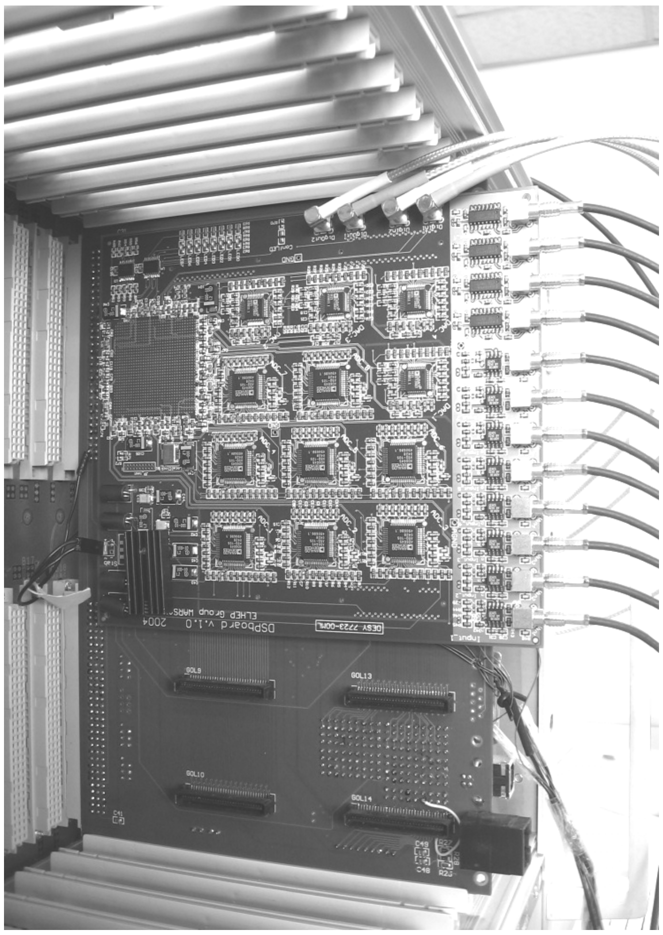


Fig. 7. Installation of the LLRF modular platform in the VME crate with the DSP EMBEDDED BOARD module.

Application of the VME bus [12] stems from the requirements imposed by the DOOCS system [1]. The DOOCS is obligatory GUI to control the accelerator of the FEL. The access was realized in the *A24D32* mode. There are 24 lines of address bus and 32 lines of data bus. The access is *SLAVE* for *AM=39H* and *MASTER* for *AM=3DH* [12]. The base address of the PCB in the address space of the VME bus is defined by the 4 oldest address bits (*A23-A20*). The VME-BUS controller was realized in the FPGA *ACEX 100K* matrix chip by Altera.

The communication inside the LLRF Platform is realized in the proprietary INTERNAL INTERFACE [7] standard, for the following configuration - 32-bits of data and 32-bits of address.

3.2 Functional structure

Integrated SIMCON system was realized in the form of parameterized structure of functional blocks in the VHDL language (Very_High_Speed_Integrated_ Circuit_Hardware_Description_Language). The implemented code was loaded in the Xilinx VirexII V4000-4 chip on the *DSP EMBEDDED BOARD* [11]. There were used the AD and DA converters situated on this board. The optional connection of the external control to the simulator or controller of the FEL cavity is possible. The digital TTL inputs present on the base DSP EMBEDDED BOARD were used for synchronization with the *1 MHz* clock and *5 Hz* trigger. These signals are distributed in the whole control system of the FEL. The digital TTL outputs provide monitoring of internal signals. An overall functional structure of the SIMCON, implemented in ver.3.0 was presented in fig. 8.

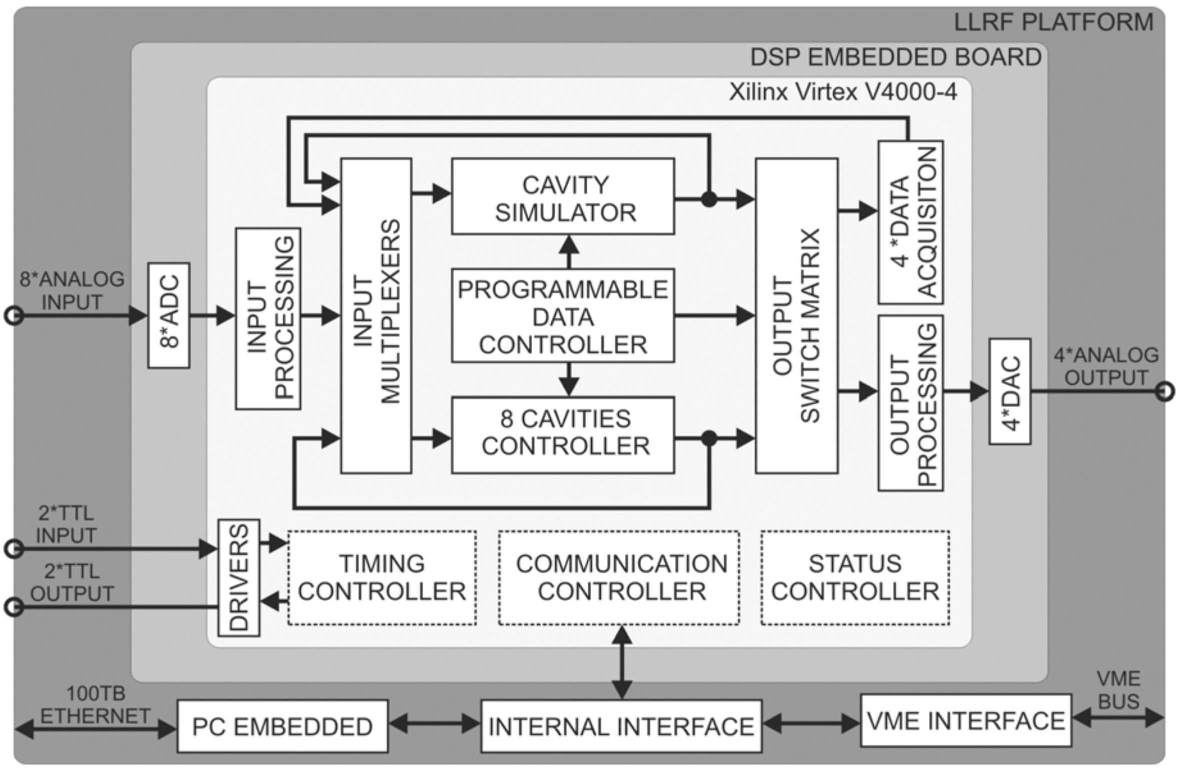


Fig. 8. Multi-Layered hardware and functional structure of the SIMCON, ver. 3.0.

The solution applied in the SIMCON system bases on the backbone of parameterized and programmable blocks of parallel processing.

The core is constructed of two nondependent modules `CAVITY SIMULTOR` and 8-`CAVITIES CONTROLLER`. They were programmed inside the FPGA *Virtex II 4000-4* chip as hardware DSP algorithms. The algorithms use fast internal multiplication components. The blocks work in parallel in the real time. They are controlled by programmable parameters provided by the `PROGRAMMABLE DATA CONTROLLER` block. The parameters are scalars (like parameters of the cavity and controller) and vectors (like the feed-forward for CAVITIES CONTROLLER, beam of cavity simulator). The set parameters stem from the algorithms described in detail in the following papers [4,5,6].

The block of `INPUT MULTIPLEXERS` serves for programmable choice of the control signals of the controller and simulator blocks. The realization of the following functions is possible through this functionality: internal digital feedback loops, connection of external analog signals from the AD converters, set test vectors initially programmed in the `DAQ` block. The task for the OUTPUT SWITCH MATRIX block is a programmable choice of the signals outputs for the DA converters or signal registration in one of the four memories in the DATA ACQUISITION block. A suitable configuration of the switching matrices gives appropriate analog feedback between the modules of CAVITIES CONTROLLER and simulator

The block TIMING & STATUS CONTROLLER provides internal synchronization of the all processes of SIMCON system. It is possible to choose the external clock signals provided by the accelerator control system or from the external generators. The latter case enables autonomous work of the system. Switching of the work states of the system is possible, i.e. performing of processes in real time or in step simulation regime with reference vectors.

The programming layer of all the blocks of SIMCON system is realized by the supervisory control computer system with the aid of the COMMUNICATION CONTROLLER block. The VME-BUS [12] hardware transmission protocol was used via the block VME INTERFACE [8] or network communication via 100Mb ETHERNET with the control realized via the block EMBEDDED PC (ETRAX processor [10]). The information distribution bases on the *Internal Interface* standard, described in detail in [2,7]. The bus is controlled by INTERNAL INTERFACE block.

4 STATUS CONTROLLER BLOCK DESCRIPTION

The SIMCON system may work in several work states (called operation modes). It provides possibility to realize various functionalities in a single integrated system. The work states are: autonomous, cooperation with external timing systems, functional tests state, diagnostic state and system programming state. Inside each operation mode two work states are available SETUP and RUN.

4.1 Functional description

The block STATUS CONTROLLER manages the work states of the SIMCON system. From the operation point of view, setting of a particular work state has a superior character. There are distinguished four system work states in the SIMCON:

- *INTERNAL* – the work in the real time mode is possible with the usage of internal timing signals (see chapter 5.2, 5.3.1)
- *EXTERNAL* – the work in the real time mode is possible with the usage of external timing signals (see chapter 5.2)
- *VECTOR* – the work is possible in the real time mode with internal timing signals and set exciting vectors (see chapter 5.2, 11.2.4)
- *STEP* – the work is possible in the step operation mode with the usage of internal timing and programmable set input exciting data, separately for each step (see chapter 5.2, 5.3.2)

Each of the above modes has *SETUP* stage. The *SETUP* enables full programmers approach to the spaces of registers and memory through COMMUNICATION CONTROLLER. Each of the above modes has *RUN* mode. The *RUN* realizes functionalities of the system mode.

4.2 Programming description

The choice of the system work mode is set with the aid of the register: MODE_OPER_SEL. The register has the following values:

- value 0 – *INTERNAL* work mode is chosen (compare chapter 4.2.1),
- value 1 – *EXTERNAL* work mode is chosen (compare chapter 4.2.2),
- value 2 – *VECTOR* work mode is chosen (compare chapter 4.2.3),
- value 3 – *STEP* work mode is chosen (compare chapter 4.2.4),

The flag CTRL_PROC_REQ enables, for the block CAVITIES_CONTROLLER, enables the choice between activation or programming of tables for the CAVITIES CONTROLLER inside the block of PROGRAMMABLE_DATA_CONTROLLER. It also allows activation of static values (compare chapter 8.2.5). The acknowledgement of setting the required work state is done through reading the identical logical state of the flag CTRL_PROC_ACK:

- CTRL_PROC_REQ=0 and CTRL_PROC_ACK=0 causes switching of tables TSETPOINT_I, TSETPOINT_Q, TFEEDFORWARD_I, TFEEDFORWARD_Q, TGAIN_I and TGAIN_Q to the access of memory programming by the user via the block COMMUNICATION CONTROLLER. It also causes automatic activation of respective static registers: SSETPOINT_I, SSETPOINT_Q, SFEEDFORWARD_I, SFEEDFORWARD_Q, SGAIN_I and SGAIN_Q.
- CTRL_PROC_REQ=1 and CTRL_PROC_ACK=1 causes connection of tables TSETPOINT_I, TSETPOINT_Q, TFEEDFORWARD_I, TFEEDFORWARD_Q, TGAIN_I and TGAIN_Q to the CAVITIES CONTROLLER. The values of the static registers SSETPOINT_I, SSETPOINT_Q,

SFEEDFORWARD_I, SFEEDFORWARD_Q, SGAIN_I and SGAIN_Q are ignored.

The flag SIM_PROC_REQ enables, for the block CAVITY SIMULATOR, a choice of activation or programming of tables in the block PROGRAMMABLE DATA CONTROLLER and respectively activation of the static values (compare chapter 8.2.3). The acknowledgement of the setting of required state is done through reading of identical logical state of the flag SIM_PROC_ACK:

- SIM_PROC_REQ=0 and SIM_PROC_ACK=0 causes table switching TBEAM_I and TBEAM_Q to the access of memory programming by the user via the block COMMUNICATION CONTROLLER. It also causes automatic activation of respective static register: SBEAM_I and SBEAM_Q.
- SIM_PROC_REQ=1 and SIM_PROC_ACK=1 causes connection of the tables TBEAM_I and TBEAM_Q to the CAVITIES CONTROLLER. The values of the static registers SBEAM_I and SBEAM_Q are ignored.

The request to change the flag state CTRL_PROC_REQ or SIM_PROC_REQ should result in respective change in the state of CTRL_PROC_ACK or SIM_PROC_ACK during the **time period of 100 ns**.

The programming conditions for particular work states are described below in the successive sub-chapters.

4.2.1 INTERNAL mode operation

In the *INTERNAL* work state the system is fully real-time and totally autonomous with the internal triggering signals and control tables (compare chapters 8.2.3 and 8.2.5).

4.2.2 EXTERNAL mode operation

In the *EXTERNAL* mode of operation the outside timing signals are used in the TTL standard (compare chapter 5.1). The signals are respectively connected to the LEMO sockets (compare chapter 3.1):

- *EXTERNAL CAVITY STROBE* connected to *DIGITAL INPUT 2*,
- *EXTERNAL CAVITY TRIGGER* connected to *DIGITAL INPUT 3*.

From the programming steering side, the *EXTERNAL* operation mode is considerably identical with the *INTERNAL* operation mode (compare chapter 4.2.1). Additional functionality is the possibility to adjust external clock signals via the modules *CAVITY STROBE DELAY* and *CAVITY TRIGGER DEL* in block TIMING CONTROLLER (compare chapter 5.3.3).

4.2.3 VECTOR mode operation

The *VECTOR* mode operation uses internal memories DAQ1.. DAQ3 implemented in the block DATA ACQUISITION as programmable input signal generators (compare chapter 11.2.4). By the choice of the channels in the block INPUT MULTIPLEXERS they are respectively connected to the input of the CAVITIES CONTROLLER and input of the cavity simulator. (compare chapter 12.2).

From the programming steering side, the *EXTERNAL* operation mode is considerably identical with the *INTERNAL* operation mode (compare chapter 4.2.1). Only in the case of the DAQ memory choice as an input generator, it requires programming with a set of signals (compare chapter 11.2.4). The memory module working as a generator **may not be used simultaneously** for data acquisition.

4.2.4 STEP mode operation

In the *STEP* operation mode there are used the internal registers to control and read the results of the DSP processing from the block `CAVITY SIMULATOR` (see chapter 8.2.3) and the block `CAVITY CONTRLLLER` (see chapter 8.2.5). A single step is realized in the module *CIVITY STROBE STEP TIMER* in block `TIMING CONTROLLER` (see chapter 5.3.2).

The STEP operation mode is used for service purposes and tests, like emulation of vector content TSETPOINT_I, TSETPOINT_Q, TFEEDFORWARD_I, TFEEDFORWARD_Q and other ones. **Due to this reason, the STEP operation mode may not be used in the real time.**

For the servicing purposes, the access to the read registers of signals from the DSP processing of the cavity simulator and controller via the block `COMMUNICATION CONTROLLER` may be done in an arbitrary moment of time during the SIMCON system activity. **It is recommended for the users to read from these registers after the operation step was completely done.**

5 TIMING CONTROLLER BLOCK DESCRIPTION

The block `TIMING CONTROLLER` processes and controls the timing signals distributed in the whole *SIMCON* system. It generates internal timing signals of the parameters set by program. The system has three basic clock signals. The time dependence between these signals were presented in fig. 8):

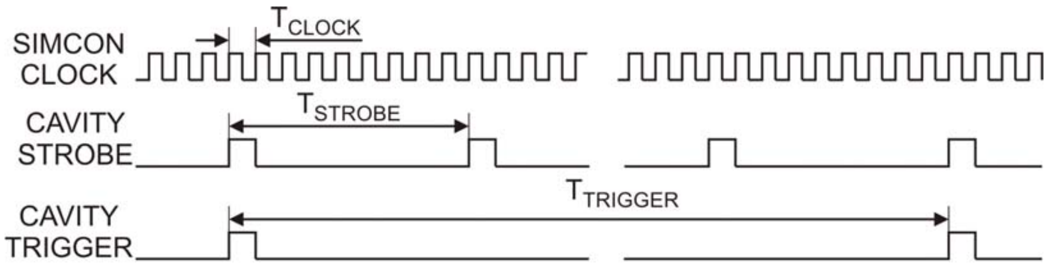


Fig. 8. Time dependencies between clock signals in the block `TIMING CONTROLLER`

- *SIMCON CLOCK* – internal timing signal with the period T_{CLOCK}=*25 ns* (*40 MHz*),
- *CAVITY STROBE* – internal or external synchronizing signal for processing of the analog signals in the AD and DA converters with the period T_{STROBE}=*1 μs* (*1 MHz*),
- *CAVITY TRIGGER* – internal or external signal initializing the process of cavity control, now the period of this signal for FEL is $T_{TRIGGER}$=*200 ms* (*5 Hz*) but may be changed on demand.

5.1 Functional structure

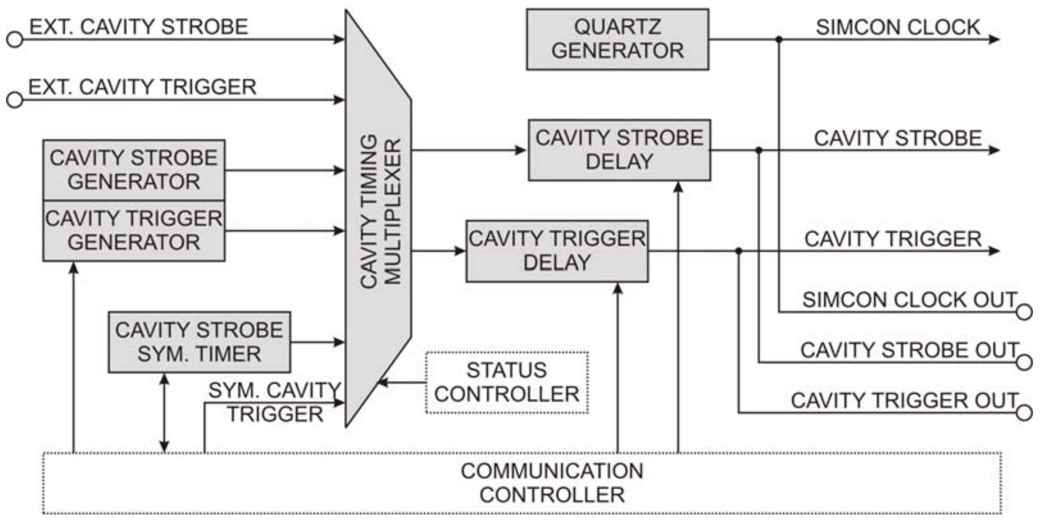


Fig. 9. Functional structure of the block `TIMING CONTROLLER`

The functional structure of the block `TIMING CONTROLLER` was presented in fig. 9. The are three processing layers in this structure:

- Choice of the clock signals, which are realized in the module *CAVITY TIMING MULTIPLEXER*,
- Generators of internal clock signals; the following modules create this structure: *CAVITY STROBE GENERATOR*, *CAVITY TRIGGER GENERATOR*, *CIVITY STROBE STEP TIMER*, *QUARTZ GENERATOR*,

- Timing adjustment consists of the following modules: *CAVITY STROBE DELAY*, *CAVITY TRIGGER DELAY*,

External signals *EXTERNAL CAVITY TRIGGER* and *EXTERNAL CAVITY STROBE* are output to the digital LEMO connectors. For the diagnostic and synchronization purposes with the external devices, the timing signals *SIMCON CLOCK OUT*, *CAVITY TRIGGER OUT* and *CAVITY STROBE OUT* were output to the digital LEMO connectors.

5.2 Cavity timing multiplexer description

Choice of the source for clock signals is done automatically in accordance with the state of the register MODE_OPER_SEL. The register is situated in block `STATUS CONTROLLER`:

- For the operation modes of the system *MODE_OPER_INTERNAL* and *MODE_OPER_VECTOR* the clock signals are taken from the internal generators *CAVITY STROBE GENERATOR*, *CAVITY TRIGGER GENERATOR*,
- For the operation mode of the system *MODE_OPER_EXTERNAL,* there are taken external clock signals *EXTERNAL CAVITY TRIGGER* and *EXTERNAL CAVITY TRIGGER*. They are automatically synchronized with the signal *SIMCON CLOCK*,
- For the operation mode *MODE_OPER_STEP,* the clock signal is taken from internal generator *CIVITY STROBE SIMULATOR TIMER* and signal *SIMULATOR CIVITY TRIGGER,* which is programmed in block `COMMUNICATION CONTROLLER`.

5.3 Programming description

The extent to program the block `TIMING CONTROL` includes setting the parameters of internal generators of clock signals and values of delays.

5.3.1 Internal timing generation

The usage of internal clock signals requires a priori programming of the generator parameters *CAVITY STROBE GENERATOR* and *CAVITY TRIGGER GENERATOR*. To set the operation mode the following registers are used:

- For the *CAVITY STROBE GENERATOR* the signal period *CAVITY STROBE* is defined as a number of the periods of the signal *SIMCON CLOCK* (*25 ns*). The value of the rate diminished by *1* is stored in the signal register GENER_STROBE_RANGE. The period may be calculated using the following expression, where *x* is given parameter:

$$T_{STROBE} = T_{CLOCK} * (x+1) \Rightarrow x = \frac{T_{STROBE}}{T_{CLOCK}} - 1 \,,$$

The nominal range of register values is confined to 0 - 63. To obtain the period equal to *1 µs* from the signal *SIMCON CLOCK* (*25 ns*) it is necessary to set the value 39.

The implemented DSP algorithms allow to set the minimum value equal to 7. The sampling period is then *200 ns*, or the modulated signal reaches *1.25 MHz*.

- For the *CAVITY TRIGGER GENERATOR* the signal period *CAVITY TRIGGER* is defined as a number of the signal periods *CAVITY STROBE*. The rate value diminished by *1* is stored in the signal register GENER_TRIGGER_RANGE. The period may be calculated using the following expression, where *y* is set parameter:

$$T_{TRIGGER} = T_{STROBE} * (y+1) = T_{CLOCK} * (x+1) * (y+1) \Rightarrow y = \frac{T_{TRIGGER}}{T_{STROBE}} - 1 = \frac{T_{TRIGGER}}{T_{CLOCK} * (x+1)} - 1 \,,$$

The nominal range of the values for the register is from 0 to 1048575 (0xFFFFF). To obtain the period *200 µs* from the signal *CAVITY STROBE* (*1 µs*) it is to input the value 199999, and the maximal period of the trigger signal is *1 s*.

5.3.2 Step operation process

The operation mode *STEP OPERATION PROCESS* is a dedicated method of a computer aided DSP processes testing. The foundation of this operation mode is that the SIMCON system works in the real time during a strictly defined period of time. The time period is set as *REAL-TIME STEP PERIOD*. During the breaks in the processing, it is possible to do computer based reading of the DSP processing results and to set new input data for next DSP processes.

The step operation method is active when the state register MODE_OPER_SEL of the operation mode is set for *MODE_OPER_STEP*. The period *REAL-TIME STEP PERIOD* is generated in the module *CAIVITY STROBE STEP TIMER* according to the prior setting of the parameters. The module is triggered with the signal *SIMCON CLOCK*. The timing diagram *STEP OPERATON PROCESS* is presented in fig 22:

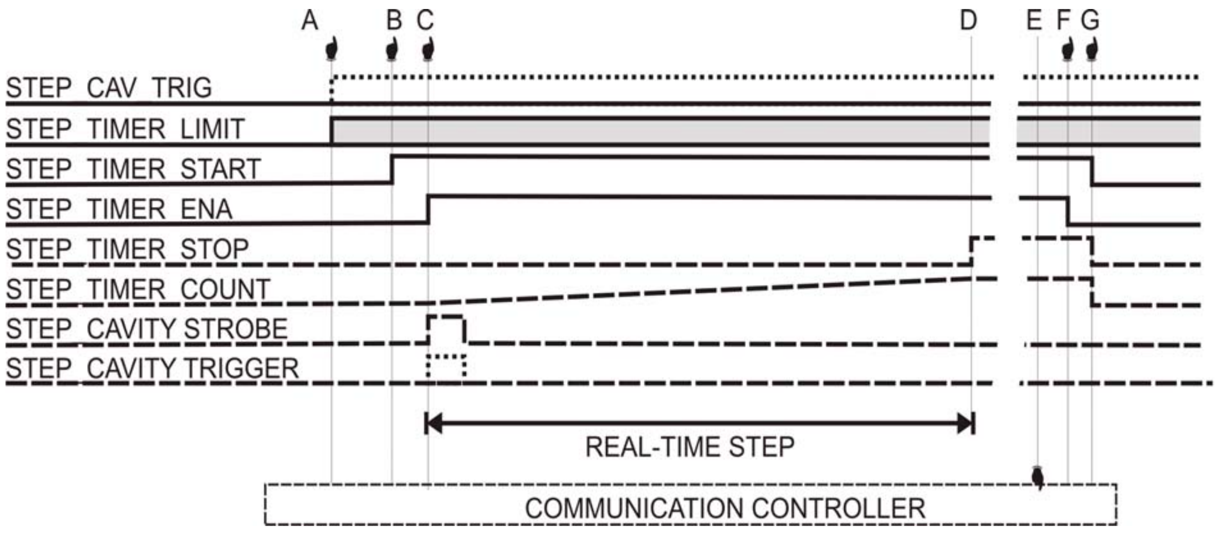


Fig. 10. Time diagram for the process *STEP OPERATION*

- $\boxed{\text{A}}$ initialization of the global conditions of the process *STEP OPERATION* embraces setting of the following:
  - register STEP_TIMER_LIMIT to the value equal to number of signal periods *SIMCON CLOCK* in the range from *0* to *63*. The given data are diminished by *1*, i.e. for the value *0* a single signal period for *SIMCON CLOCK* will be registered.
  - register STEP_CAV_TRIG should be set to the appropriate value:
    0: in the current step will not be generated *STEP CAVITY TRIGGER*,
    1: in the current step will be generated *STEP CAVITY TRIGGER*.
- $\boxed{\text{B}}$ initialization of the module *STEP TIMER* through setting STEP_TIMER_START=1.
- $\boxed{\text{C}}$ activation of the module *STEP TIMER* through setting STEP_TIMER_ENA=1. From this very moment, the *STEP OPERATON PROCESS* is automatically triggered. The counter starts STEP_TIMER_COUNT which measures the time of the process.
- $\boxed{\text{D}}$ automatic stop of the DAQ process after the counter STEP_TIMER_COUNT reaches a value set in the register STEP_TIMER_LIMIT. The following flag is set STEP_TIMER_STOP=1.
- $\boxed{\text{E}}$ checking flag reading STEP_TIMER_STOP. Reading of value *0* means that *STEP*

OPERATON PROCESS continues. Reading the value *1* means that the process is finished. The flag reading may be done many times, waiting for the process to be finished.

- $\boxed{\text{F}}$ stopping the work of the module *STEP TIMER* through setting STEP_TIMER_ENA=0.
- $\boxed{\text{G}}$ introducing the module in the blocked state *STEP TIMER* through setting STEP_TIMER_START=0. The flag is deleted STEP_TIMER_STOP=0 and zeroing of STEP_TIMER_COUNT.

If the global acquisition conditions remain not changed, the next initialization of the DAQ process may disregard the stage $\boxed{\text{A}}$.

Temporary change in the flag state STEP_DSP_RESET from the value *0* to value *1* causes asynchronous resetting of the DSP processes in the CAVITIES CONTROLLER and simulator. For the servicing purposes of the flag state through the block COMMUNICATION CONTROLLER may be done in an arbitrary moment of the SIMCON system work. **The SIMCON system users are strongly advised to reset the DSP processes only in the *STEP MODE OPERATION* just before doing the stage [A].**

The flag state STEP_DSP_STOP=1 which means finishing of the calculation period for both DSP processes. For the servicing purposes of the flag state through the block COMMUNICATION CONTROLLER may be done in an arbitrary moment of the work state of SIMCON system. **The users are strongly advised to reset the DSP processes only in the *STEP MODE OPERATION* just after doing the stages [E], [F] or [G].**

5.3.3 Time adjustment of the trigger signals

Time adjustments of the triggering signals is done by two modules:

- Module *CAVITY STROBE DELAY* delays the signal *CAVITY STROBE* of set number of signal periods *SIMCON CLOCK* (*25 ns*) in the range from 0 to 63. The value of delay is set in the register CAV_STROBE_DELAY. The range of delay embraces approximately *1.5 μs*, or exceeds a single period of signal *CAVITY STROBE*. Taking the value *0* means no additional delay of the signal *CAVITY STROBE*.
- Module *CAVITY TRIGGER DELAY* delays the signal *CAVITY TRIGGER* of set number of signal periods *CAVITY STROBE* (*1 μs*) in the range from 0 to 2047. The value of delay is set in the register CAV_TRIGGER_DELAY. The range of delay embraces above *2 ms*, or exceeds the longest control time of the cavity. Taking the value *0* means no additional delay of the signal *CAVITY TRIGGER*.

The signals *CAVITY STROBE* and *CAVITY TRIGGER* considered in the next part of this document are referenced **only** to the signals after the delay modules.

6 INPUT PROCESSING BLOCK DESCRIPTION

The block `INPUT PROCESSING` provides proper conversion of values between a physical 14-bit resolution of the ADC converters and 18-bit resolution of the internal DSP processing, input signal calibration including amplification and regulated shift of constant voltage value, as well as initial smoothing of the input channels using a method of averaging of a set value of samples.

6.1 Functional structure

The block `INPUT PROCESSING` consists of an input module for resolution conversion *INPUT RESOLUTION CONVERTER, INPUT CALIBRATION* module and an averaging module *INPUT SIGNAL AVERAGING* for the input signal. Its functional structure is presented in fig. 11.

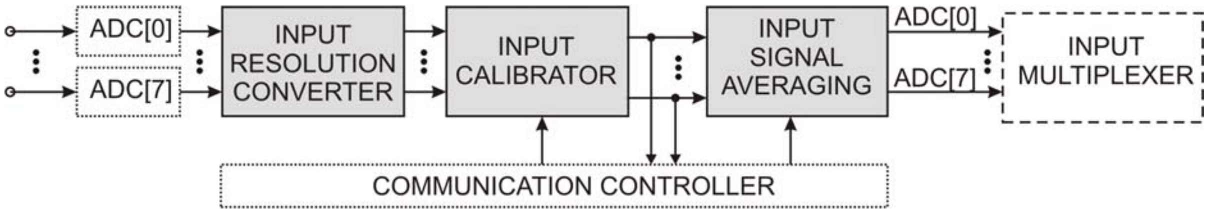


Fig. 11. Functional structure of the block `INPUT PROCESSING`

The module *INPUT RESOLUTION CONVERTER* realizes nondependently for each input channel the change from a *14*-bit U2 code obtained from particular ADC converter to *18*-bit representation of U2 code for the DSP processes. The conversion process relies on multiplication of the input signal value by the correction coefficient equal to *16*, what in such a case is equivalent to a logical shift of the input value to four places to the left.

The module of *INPUT CALIBRATOR* allows for fitting of the real input signal to the set levels of signals required in the algorithms of cavity simulator and controller. The module realizes, nondependently for each input channel a correction of the input signal. The performed process relies on the following DSP operation for all ADC channels in the 18-bit range:

$$y = x * G + O$$

where, the *G* parameter is the gain, the *O* parameter is constant voltage shift added to the signal.

The module *INPUT SIGNAL AVERAGING* realizes nondependently for each input channel the following averaging functional operation:

$$Y_{AV}[K] = \frac{\sum_{t=0}^{K-1} x_{-t}}{K}$$

where: $Y_{AV}[K]$ expresses the averaging value of the last *K* samples, or the current sample (time moment *t=0*), and the preceding samples, from *t=(-1..-K+1)* moments of time. The timing of the samples is defined by the signal *CAVITY STROBE*. The averaging coefficient *K* is set as: $K=2^N$, or for the range *N=0..3*, there are obtained the following values *K=1,2,4,8*.

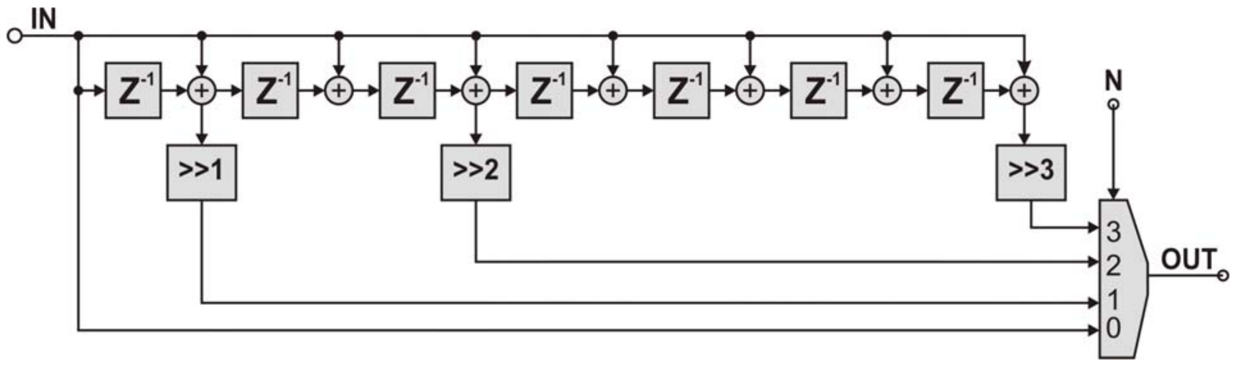


Fig. 12. Time dependencies for cyclical data

For the value *N=0* (*K=1*) work of the averaging circuit is confined for transmission of the input value to the output: $Y_{AV}[1]=x_0$. The structure of the averaging module is presented in fig.12.

6.2 Programming description

Programming of the block `INPUT PROCESSING` relies on setting of individual calibration coefficients for all 8 ADC channels and the choice of a common averaging coefficient for all channels.

The calibration parameters are determined, respectively for *N* channels (in the range of N from 0 to 7) registers ADC_GAIN[N] and ADC_OFFSET[N].

The choice the common averaging coefficient for all *ADC* channels is performed by writing to the register ADC_AVER. The value of the averaging coefficient is in the range from *0* to *3*.

In order to dynamically change the parameters of the calibration one has to use the exchange registers ADC_GAIN_BUF[N] and ADC_OFFSET_BUF[N] respectively for *N* channels in the range from 0 to 7 (chapter 8.2.3).

- .

For the servicing purposes, the choice of the value for the averaging coefficient through the block `COMMUNICATION CONTROLLER` may be done during the arbitrary moment of the SIMCON system work time. **The users of the SIMCON system are strongly advised to set the averaging coefficient choice register only during the *SETUP MODE OPERATION.***

The current state of the *ADC* converter from the N-th channel, in the range from 0 to 7, may be done through reading of the register ADC_DATA[N].

The following reading of the *ADC* analog channels are only for service purposes. It is to remember, that the read values may possess instable character, because they stem from the analog character of the input signals. The sampling period of the A/D converters results from the signal period *SIMCON CLOCK* and equals *25 ns*.

7 OUTPUT PROCESSING BLOCK DESCRIPTION

The block `OUTPUT PROCESSING` provides proper value conversion between the physical 18-bit resolution of the internal DSP processing and 14-bit resolution of the DAC converters.

7.1 Functional structure

The block `OUTPUT PROCESSING` consists only from the output module of resolution bits conversion *OUTPUT RESOLUTION CONVERTER*. Its functional structure was presented in figure 13.

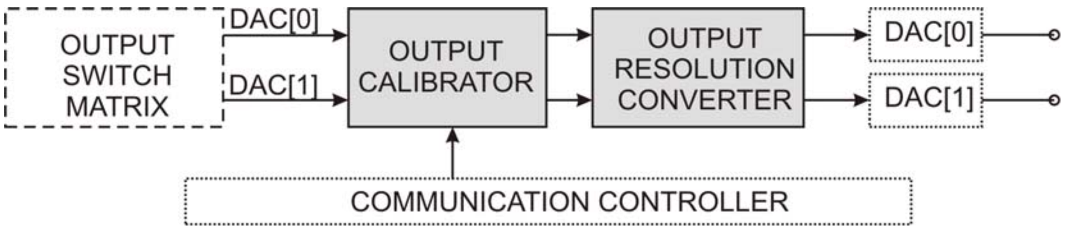


Fig. 13. Functional structure of the block `INPUT PROCESSING`

The module *OUTPUT CALIBRATOR* enables fitting of the real output signal to assumed signal levels required respectively in the algorithms of the CAVITIES CONTROLLER and simulator or required by the vector modulator or klystron. The module realizes, for each output channel, a nondependent correction of the input signal. For all DAC channels, there is performed a nondependent DSP process with 18-bit range:

$y=x+O$, where O is a constant shift of the output signal to the DACs.

The module *OUTPUT RESOLUTION CONVERTER* realizes, for each input channel, no dependently, the change of *18*-bit U2 code, used in the DSP processes, to *14*-bit representation NB required by the DA converters. The performed process relies on the dividing of the DSP signal value by the correction number equal to *16*, what in this case is equivalent to logical shift to the value to the left of 4 bits, and changing of the notation from U2 to NB.

7.2 Programming description

The programming of block `OUTPUT PROCESSING` relies on setting the calibration coefficients for both DAC channels. The calibration parameters are determined by the respective registers DAC_OFFSET[0] and DAC_OFFSET[1].

8 PROGRAMMABLE DATA CONTROLLER

The block `PROGRAMMABLE DATA CONTROLLER` provides programming facility and data input to both DSP processes (for both cavity SIMULATOR and CONTROLLER) as well as data enabling control of the DSP processes. Three kinds of data are distinguished by the system:

1. *static data* – they are input in the form of constant values (cavity parameters, controller parameters, like amplification coefficient of the CAVITIES CONTROLLER SGAIN_I and SGAIN_Q, see chapters 2.2, 10.2),
2. *dynamic data* – signal tables which are input in a form of a priori preprogrammed time dependent shape. Triggering of the beginning of the function is done by the signal *CAVITY TRIGGER*, and the next changes of these values are timed by the signal *CAVITY STROBE*. The example may be the values of tables TBEAM_I and TBEAM_Q of the cavity simulator (compare chapters 2.1, 9.2),
3. *control data* – they are automatically generated in accordance with a priori set parameters and given in a form of periodic functions. The example may be values of I/Q modulator controller, which is described in the next chapter.

8.1 Functional structure

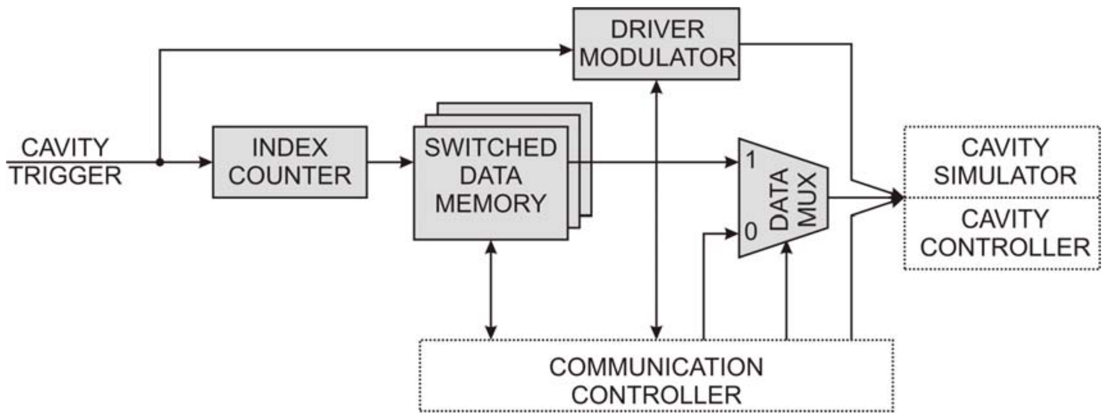


Fig. 14. Functional structure of the block `PROGRAMMABLE DATA CONTROLLER`

The functional structure of the block `PROGRAMMABLE DATA CONTROLLER` was presented in fig. 14. The block provides separate mechanisms of data input to the both DSP blocks, depending on the data type:

- Only the static type data are provided directly from the block registers of `COMMUNICATION CONTROLLER`. Each register is 18-bit. During the real time work it is possible to change the values of respective control register of the DSP process. A priori, the alternative registers are programmed and the request for change is performer. The change is done automatically before the cavity process starts.
- For the dynamic data, the module *INDEX COUNTER* calculates the current address of the cells *SWITCHED DATA MEMORY*. From the moment of signal trigger *CAVITY TRIGGER,* the successive cells in the memory table are input to the particular DSP process. The change of index has a periodic nature from *0* to *2047*, till the next value of *0*. The signal *CAVITY STROBE* means next steps of the process. The time dependencies of this process were shown in fig. 16.

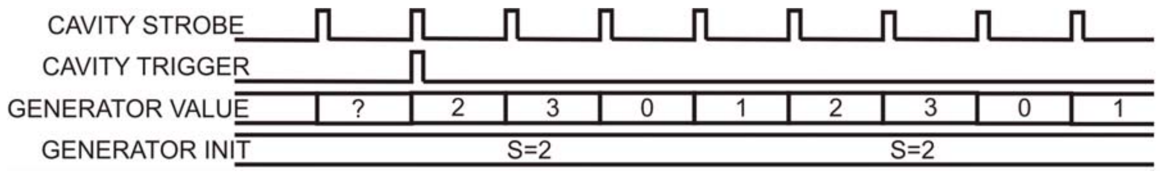


Fig. 15. Time dependencies for cyclic data.

The dynamic data are remembered in a form of tables of the dimensions 18-bits for 2048 cells. In this way, the change dynamics is provided for *1 μs* for the period of time *2048 μs*, which embraces the whole period of cavity control by the controller.

During the real time work it is possible to change the values of tables by earlier programming of alternative tables and setting the request for change. The change is done automatically before the cavity control process begins (compare chapter 8.2.3).

The choice of data of dynamic or static type (variant advice only in the *STEP MODE OPERATION)* is done through the control of the module *DATA MUX*.

- Control data for the modulation are generated with the aid of a cyclic generator working in the range from *0* to *3* in the module *DRIVER MODULATOR*. The signal *CAVITY TRIGGER* initializes cyclic generator to the *initial value (S)*, but the change of his value is triggered by the signal *CAVITY STROBE.* The time dependencies were shown in fig. 15 for the initializing value S=2.

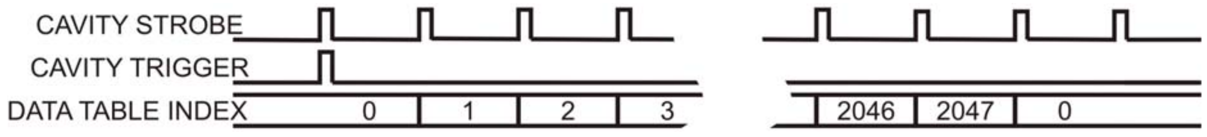


Fig. 16. Time dependencies for dynamic data

8.2 Programming description

The programming of the block PROGRAMMABLE DATA CONTROLLER relies on the input of static data (writing of 18-bit register) and dynamic data (filling the memory areas of 2048 cells 18-bit each) and on the choice of the channel in the module *DATA MUX*. The details of programming of particular components are described in the chapters below.

8.2.1 Dynamic data multiplexer

The multiplexer *DATA MUX* controls the choice of data kind, in dependence on the values of flags CTRL_PROC_REQ and SIM_PROC_REQ (compare chapter 4.2):

- CTRL_PROC_REQ=0 chooses channel *0* for all static data of block CAVITIES CONTROLLER (compare chapter 8.2.5),
- SIM_PROC_REQ=0 chooses channel *0* for all static data block CAVITY SIMULATOR (compare chapter 8.2.3),
- CTRL_PROC_REQ=1 chooses channel *1* for all dynamic data block CAVITIES CONTROLLER (compare chapter 8.2.5),
- SIM_PROC_REQ=1 chooses channel *1* for all dynamic data block CAVITY SIMULATOR (compare chapter 8.2.3),

8.2.2 Modulator driver

The module *MODULATOR DRIVER* requires programming of the initial value in the register VM_DRV_START[N] in the range from *0* to *3,* for each channel N=0…7.

Physical writing of the value VM_DRV_START[0...7] to the module *MODULATOR DRIVER* is done through performing write operation of an arbitrary value to the register VM_DRV_COUNT[0...7].

For the servicing purposes, there is a possibility to read the current value of the module *MODULATOR DRIVER* via the register VM_DRV_COUNT[0..7]. This value has, however, a nonstable character, because it changes periodically in the range from *0* to *3* every *1 μs*. The stable value is obtained in the *STEP MODE OPERATON* or *SETUP MODE OPERATON*.

8.2.3 Data switching

The request to change the static and dynamic data relies on the negation of the actual state of appropriate flags: : SWITCH_ADC_GAIN, SWITCH_ADC_OFFSET, SWITCH_CAL, SWITCH_COMP, SWITCH_TFEEDFORWARD, SWITCH_TSETPOINT, SWITCH_TGAIN, i.e. changing its value from 0 to 1 or vice versa. The acknowledgement of the change causes stetting of the flag TAB_SWITCH_ACK to 1 till the request is done. Then the value of the flag TAB_SWITCH_ACK returns to 0, what means the acknowledgment of the change

The change of registers and memory is started by signal *CAVITY TRIGGER* in an automatic way, simultaneously for all of the following components, when there was recently the change of the switching flag:

• SWITCH_ADC_GAIN[0..7]	- ADC_GAIN[0..7] with ADC_GAIN_BUF[0..7],
• SWITCH_ADC_OFFSET[0..7]	- ADC_OFFSET[0..7] with ADC_OFFSET_BUF[0..7]
• SWITCH_CAL	- ROT1 with ROT1_BUF and ROT2 with ROT2_BUF,
• SWITCH_COMP	- COMP1 with COMP1_BUF and COMP2 with COMP2_BUF
• SWITCH_TSETPOINT	- TSETPOINT_I with DAQ1 and TSETPOINT_Q with DAQ2,
• SWITCH_TFEEDFORWARD	- TFEEDFORWARD_I with DAQ3 and TFEEDFORWARD_Q with DAQ4,
• SWITCH_TGAIN	- TGAIN_I with TBEAM_I and TGAIN_Q with TBEAM_Q.

Till the moment to obtain the acknowledgment TAB_SWITCH_ACK it is necessary not to change the value of any SWITCH flag.

8.2.4 Cavity simulator programmable data packet

The block CAVITY SIMULATOR, in accordance with the algorithm described in chapter 2.1 requires setting of the following data:

- *BEAM* (dynamic data): is represented by a table TBEAM_I and TBEAM_Q or alternatively by the registers SBEAM_I and SBEAM_Q,
- *ROTATION MATRIX [R]* (static data): $R = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$ are expressed in succession by the parameters ROT1 and ROT2,
- *COMPENSATION MATRIX [C]* (static data): $C = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$ are expressed in succession by the parameters COMP1 and COMP2,
- MATRIX_A1_21 and MATRIX_A1_22 – individual coefficients of the matrix [A_1],
- *MATRIXES [A_1], [A_2], [A_3]* (static data): $A_{n=1,2,3} = \begin{bmatrix} 1 & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ are expressed by the following

parameters:

- MATRIX_A12 – common coefficient a_{12} for all three matrixes,
- MATRIX_A1_21 and MATRIX_A1_22 – individual coefficients of the matrix $[A_1]$,
- MATRIX_A2_21 and MATRIX_A2_22 – individual coefficients of the matrix $[A_2]$,
- MATRIX_A3_21 and MATRIX_A3_22 – individual coefficients of the matrix $[A_3]$,

- *MATRIXES $[B_1]$, $[B_2]$, $[B_3]$* (static data): $B_{n=1,2,3} = [b_1 \quad 0]$ the are expressed by the following parameters b_1 appropriately for the successive matrixes: MATRIX_B1_1, MATRIX_B2_1 and MATRIX_B3_1,
- *COEFFICIENT „H”* (static data): is expressed by the PARAM_H,
- *COEFFICIENT „P”* (static data): is expressed by the PARAM_P,
- flag SIM_MODE (static data) sets the dynamic range of the cavity voltage simulation:
  - the value *0*: *16 MV/m*,
  - the value *1*: *32 MV/m*.
- Activation of the static register requires setting SIM_PROC_REQ=0, alternatively switching on the set point tables SIM_PROC_REQ=1 (compare chapter 4.2).

8.2.5 Cavities controller programmable data packet

The block CAVITIES CONTROLLER, in agreement with the algorithm described in the chapter 2.2 requires setting the following data CTRL_PROC_REQ=0:

- *SET POINT* (dynamic data): is represented by the tables TSETPOINT_I and TSETPOINT_Q or alternatively by the registers SSETPOINT_I and SSETPOINT_Q,
- *FEED FORWARD* (dynamic data): is represented by the tables TFEEDFORWARD_I and TFEEDFORWARD_Q or alternatively by the registers SFEEDFORWARD_I and SFEEDFORWARD_Q,
- *GAIN* (dynamic data): is represented by the tables TGAIN_I and TGAIN_Q or alternatively by the registers SGAIN_I and SGAIN_Q,
- *ROTATION MATRIX* (static data): is represented by registers ROT1 and ROT2, (swap registers: ROT1_BUF and ROT2_BUF)
- *COMENSATION MATRIX* (static data): is represented by the registers COMP1 and COMP2. (swap registers: COMP1_BUF and COMP2_BUF)

Activation of the static registers requires setting MODE_OPER_SEL=3. The following work mode is set: *STEP OPERATON PROCESS* (chapter 4.2.4).

9 CAVITY SIMULATOR BLOCK DESCRIPTION

The block `CAVITY SIMULATOR` performs, in the real time, the algorithm of the superconducting cavity behavior, in agreement with the requirements of the LLRF system (see chapter 2.1). An 18-bit fixed point algorithm was implemented with the use of the DSP components present in the FPGA chip Xilinx VirtexII-V4000.

9.1 Functional structure

The block `CAVITY SIMULATOR` consists of the synchronous numerical processing module *DSP CAVITY ALGORITHM*, from the modules of signal delays *INPUT DELAY* and *OUTPUT DELAY* and from the modulator module of I/Q *MODULATOR*. Its functional structure was presented in figure 17.

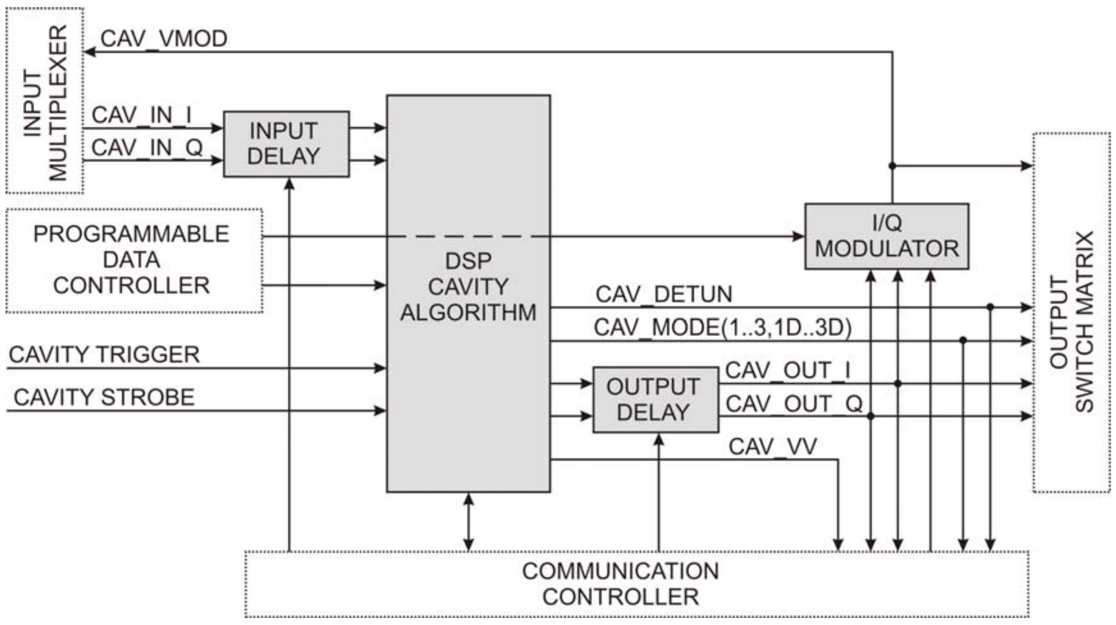


Fig. 17. Functional structure of the block `CAVITY SIMULATOR`

The module *DSP CAVITY ALGORITHM* processes the signal vector for cavity control *CAV_IN_I* and *CAV_IN_Q*, which is provided from the block `INPUT MULTIPLEXER` (see chapter 12) in accordance with the parameters provided from the block `PROGRAMMABLE DATA CONTROLLER` (see chapter 6). There are the following signals obtained at the output of this block:

- Basic signals from the cavity (*CAV_OUT_I* and *CAV_OUT_Q*),
- Modulated signal I/Q (*CAV_VMOD*),
- Detuning signal from the cavity mechanical model (*CAV_DETUN*),
- Six signals of the state vector [W] of the mechanical model (*CAV_MODE(1..3,1D..3D)*).
- Signal of the square value for the high power EM field gradient v^2 (*CAV_VV*),

The module *I/Q MODULATOR* realizes modulation process for the signals I and Q from the cavity. The modulator control is provided by the module *MODULATOR DRIVER* situated in the block `PROGRAMMABLE DATA CONTROLLER` (see chapter 8.1).

The modules of delay of the input and output DSP data (*INPUT DELAY* and *OUTPUT DELAY*) allow to simulate the physical delays introduced by the transmission lines (waveguides). A single step of the delay defines the timing of the signal *CAVITY STROBE*, which is currently equal to *1 μs.*

9.2 Programming description

The programming of the work of the block CAVITIES CONTROLLER relies on:

- Setting of proper parameters in the block PROGRAMMABLE DATA CONTROLLER (see chapter 8.2.5). These are the following parameters: *BEAM*, matrixes *[A_1], [A_2], [A_3]*, *[B_1], [B_2], [B_3],* coefficients *„H”* and *„P”*,
- Setting of the modulation phase realized in the module *I/Q MODULATOR* in reference to the demodulation realized in the CAVITIES CONTROLLER (compare chapter 8.1 and 8.2.2). The phase change is determined by the register VM_DRV_OFFSET[N] in the value range of 0 - 3, for channels N=0…7;
- Setting of the delays for the input and output signals, respectively via the programming of the registers CAV_DELAY_IN and CAV_DELAY_OUT. Each of the registers allows to write the values from *0* to *15*. In this way, the range of delays is provided up to *15 μs* with a step of *1 μs*. Setting the value of *0* means no additional delay introduced.

In the operation mode *STEP OPERATON PROCESS* the following registers are made available for computer based writing via the block COMMUNICATION CONTROLLER. The registers have the same eigen-names with the source signals for the cavity simulator DSP processing:

- registers CAV_IN_I and CAV_IN_Q controlling directly the signals *CAV_IN_I* and *CAV_IN_Q*.

In the operation mode *STEP OPERATON PROCESS*, for the computer based reading, via the block COMMUNICATION CONTROLLER, there are made available the following current values of the cavity simulator DSP processing results, via the registers with the eigen-names equal to the relevant signals:

- signals *CAV_OUT_I* and *CAV_OUT_Q* respectively through the registers CAV_OUT_I and CAV_OUT_Q,
- signal *CAV_VMOD* via the register CAV_VMOD,
- signal *CAV_DETUN* via the register CAV_DETUN,
- six signals of mechanical modes *CAV_MODE(1..3,1D..3D)* via the registers:
  - o *CAV_MODE1* – the first mechanical mode is accessible in the register CAV_MODE1,
  - o *CAV_MODE1D* – derivative of the first mechanical mode is accessible via the register CAV_MODE1D,
  - o *CAV_MODE2* – the second mechanical mode is accessible in the register CAV_MODE2,
  - o *CAV_MODE2D* – derivative of the second mechanical mode is accessible via the register CAV_MODE2D,
  - o *CAV_MODE3* – the third mechanical mode is accessible via the register CAV_MODE3,
  - o *CAV_MODE3D* – derivative of the third mechanical mode is accessible via the register CAV_MODE3D.
- The signal *CAV_VV* via the register CAV_VV,

For the servicing purposes, the access to the read registers of the signals from the cavity simulator DSP process, via the COMMUNICATION CONTROLLER, may be done in the arbitrary moment during the work time of the SIMCON system. **The users are strongly recommended to read these registers only during the *SETUP MODE OPERATION.***

10 CAVITIES CONTROLLER BLOCK DESCRIPTION

The block `CAVITIES CONTROLLER` performs, in the real time, a control algorithm for the superconductive cavity, in agreement with the requirements of the LLRF system design parameters (compare chapter 2.2). There was implemented an 18-bit fixed point algorithm, with the usage of the DSP components integrated into the FPGA Xilinx VirtexII-V4000 chip.

10.1 Functional structure

The block `CAVITIES CONTROLLER` consists firm the synchronous module of numerical processing (*DSP CONTROLLER ALGORITHM*) and from synchronization module of I/Q detection (*DRIVER MODULATOR*). Its functional structure was presented in figure 18.

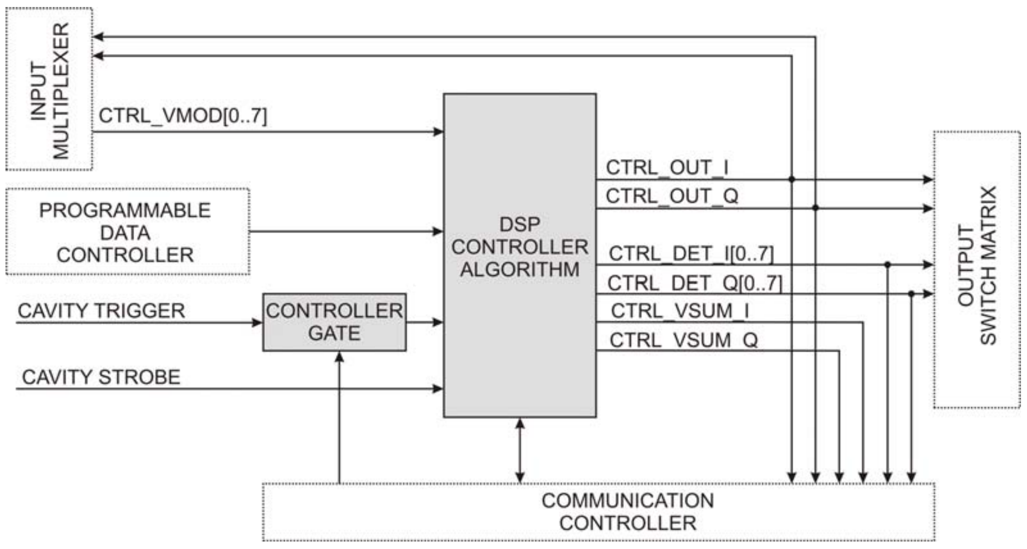


Fig. 18. Functional structure of the block `CAVITIES CONTROLLER`

The module *DSP CONTROLLER ALGORITHM* processes the appropriate modulated input signals *CTRL_VMOD[0..7]* provided from the block `INPUT MULTIPLEXER` (see chapter 12) in accordance with the parameters provided by the block `PROGRAMMABLE DATA CONTROLLER` (see chapter 6). The output of the module gives two output vectors:

- Basic control signal for vector modulator of the klystron (*CTRL_I*, *CTRL_Q*),
- Vector sum (*SUM_I*, *SUM_Q*) for monitoring purposes,
- Auxiliary signal after the detection (CTRL_DET_I[0...7], CTRL_DET_Q[0...7]) for monitoring purposes.

The module *CONTROLLER GATE* allows to activate the block `CAVITIES CONTROLLER` only during the active state of the time gate, and during the rest of time the output data from the block have *0* value. The signal *CAVITY TRIGGER* initializes the gate for a set period of time by the *time range (R).* The gate is timed with the signal *CAVITY STROBE.* The time dependencies of these processes are presented in fig. 19.

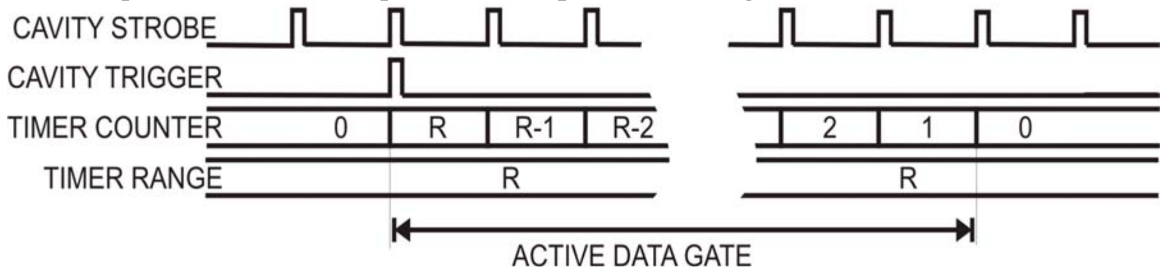


Fig. 19. Time dependencies of signals for the time gate of the CAVITIES CONTROLLER

10.2 Programming description

The programming of the block CAVITIES CONTROLLER relies on:

- Setting of appropriate parameters in the block PROGRAMMABLE DATA CONTROLLER (see chapter 8.2.3). These are the following parameters: *SET POINT*, *FEED FORWARD* and *GAIN*.
- setting of activity time for the time gate in the register CTRL_ACTIVE in the range from *1* to *2047* periods of the signal *CAVITY STROBE* (or nominally every *1 μs*).

Setting the value to *0* in the register CTRL_ACTIVE is reserved only to the servicing purposes – it keeps the gate active all the time, or the CAVITIES CONTROLLER DSP process is all the time zeroed.

In the operation mode *STEP OPERATON PROCESS*, for the computer reading, via the block COMMUNICATION CONTROLLER, the following register is made available with the name identical as the CAVITIES CONTROLLER DSP processing source signal:

- register CTRL_VMOD controlling directly all the signals *CTRL_VMOD[0…7]*.

In the operation mode *STEP OPERATON PROCESS*, for the computer reading, via the block COMMUNICATION CONTROLLER, the following current values of the DSP processing are made accessible, via the registers of the names identical as the CAVITIES CONTROLLER DSP signals:

- signals *CTRL_I[N]* and *CTRL_Q[N]* respectively through the registers CTRL_OUT_I[N] and CTRL_OUT_Q[N] for input channels N=1…7,
- signals *CTRL_ DET_I* and *CTRL_ DET_Q* respectively through the registers CTRL_DET_I and CTRL_DET_Q,

For the servicing purposes, the access to the reading registers of the CAVITIES CONTROLLER DSP process signals is available, via the block COMMUNICATION CONTROLLER. The access may be done during the arbitrary moment of the SIMCON system activity. **The SIMCON users are strongly recommended to read these data from these registers only during the *SETUP MODE OPERATION.***

11 DATA ACQUISITION (DAQ) BLOCK DESCRIPTION

The block `DATA ACQUISITION (DAQ)` allows for current monitoring of the most important signals in the system. These may be input signals, as well as output, internal results from the DSP processing in the algorithms of the cavity simulator and controller. It may additionally fulfill the function of a programmable signal generator for tests of input and output signals.

11.1 Functional structure

The block `DAQ` realizes parallel, synchronized registration of four data streams. Its functional structure is presented in figure 20.

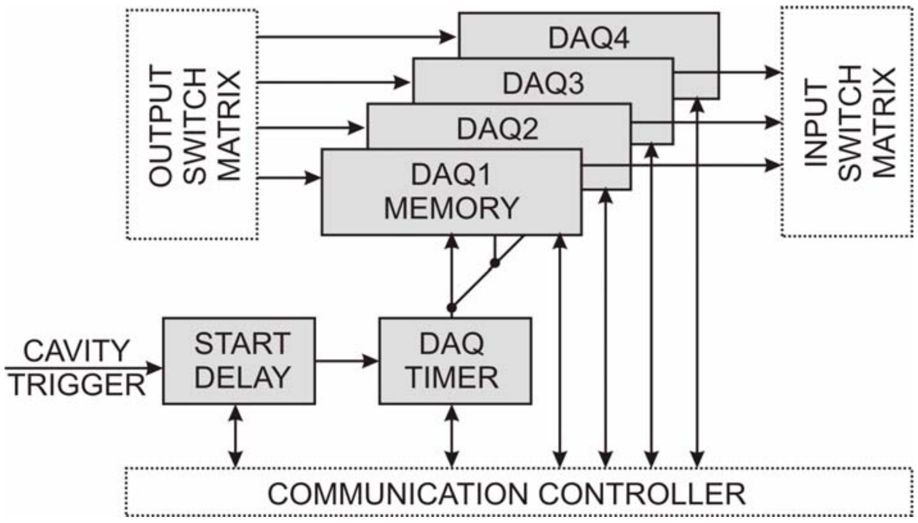


Fig. 20. Functional structure of the block `DATA ACQUISITION`

The choice of the source data streams is done in the block `OUTPUT SWITCH MATRIX`. The block `DAQ` bases on four memory modules. Each memory (DAQ1 .. DAQ4) has 2048 words per 18-bits each. The acquisition process is controlled by the *DAQ TIMER,* in agreement with the a priori set parameters. Triggering of the acquisition process is done by the signal *CAVITY TRIGGER*. This signal may be delayed in the module *START DELAY* of a preset number of clock signals *CAVITY STROBE* (*1 MHz*). In this way, one obtains the possibility to shift the reading time window in relation to the trigger signal, with the step of *1 μs*.

Additionally, the block `DAQ` realizes the functions of programmable input test vectors in the operation mode *VECTOR_OPERATION*. The data from the memory DAQ1 .. DAQ3 are transmitted via the block `INPUT MULTIPLEXERS` respectively to a single input of the CAVITIES CONTROLLER and to two inputs of the cavity simulator.

11.2 Programming description

The programming of the block `DATA ACQUISITION` allows to set operation modes, for direct access to the memory areas, programming the conditions of the acquisition in the real time and control of the status of the data acquisition process.

11.2.1 DAQ modes control

The basic operation modes of the `DAQ` block are set with the flag DAQ_PROC_REQ. Acknowledgement of the required operation mode is obtained by reading of the identical

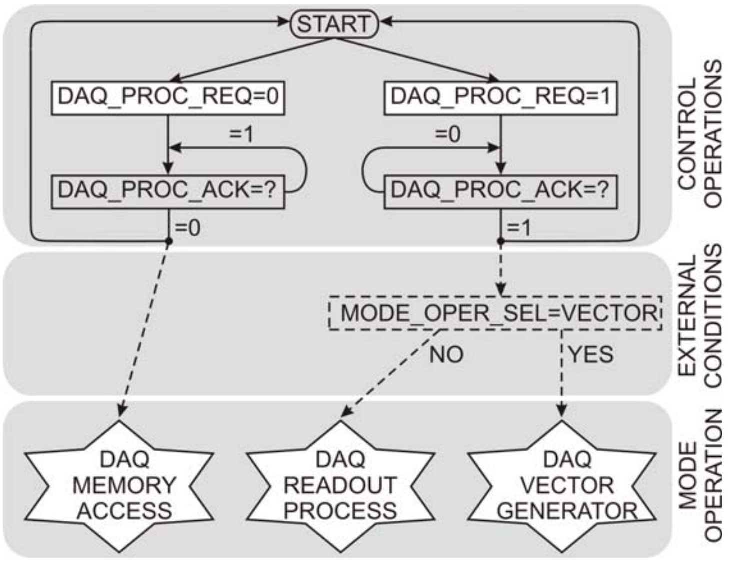


Fig. 21. Flow diagram for the choice of the operation mode of the DAQ block.

logical state of the flag DAQ_PROC_ACK. Till the time, both flags have the same states, the DAQ block is in the state of switching and has no defined state. The block DAQ may be set in one of three different operation modes, what was presented in fig. 21. The choice of the operation mode in the real time is forced by the current state of the register MODE_OPER_SEL. The particular operation modes are described in details in the next sub-chapters.

11.2.2 DAQ memory access

For the flag value DAQ_PROC_REQ=0 (and acknowledgement via setting of the value of flag DAQ_PROC_ACK=0) the direct access to the memory DAQ1..*DAQ4* is obtained in the write or read mode. The base addresses are determined by the parameters DAQ1..DAQ4_MEM. Each memory represents a continuous area of 2048 relative address positions counted from the value of 0 till 2047 and including 18-bit words.

11.2.3 DAQ readout process

The operation mode *DAQ_READOUT_PROCESS* is obtained after programming the value of the flag DAQ_PROC_REQ=1 (confirmed by setting the value of the flag DAQ_PROC_ACK=1) and after fulfilling the condition, that no operation mode *OPER_MODE_VECTOR* was programmed for the system in the register MODE_OPER_SEL. The debated operation mode allows for parallel data acquisition of four data streams. It is

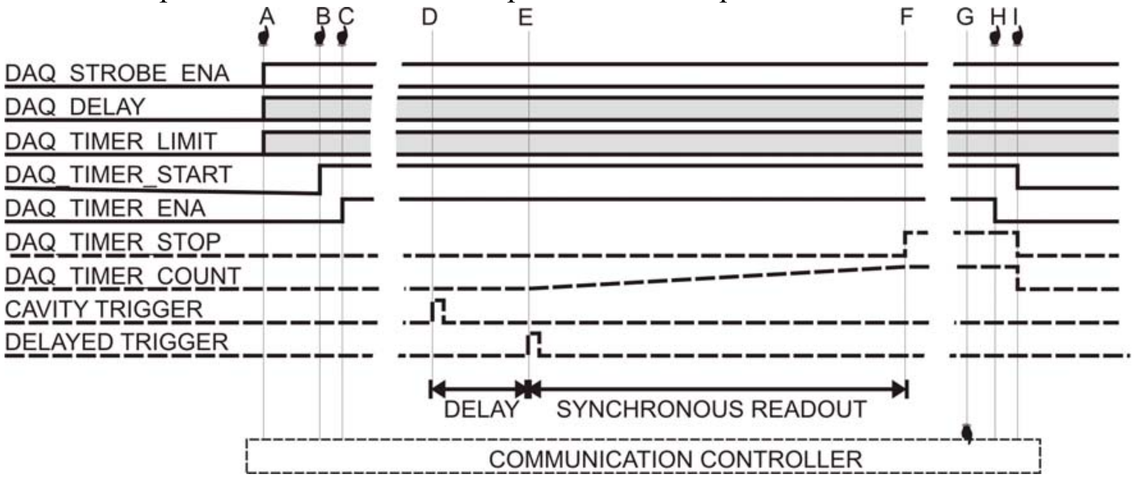


Fig. 22. Time diagram of the data acquisition process in the block DAQ

performed automatically, according to the a priori preset parameters. The key stages of the control process for the data acquisition are presented in figure 22:

- $\boxed{\text{A}}$ initialization of the global parameters for the acquisition process embraces setting of the following parameters:
  - flag DAQ_STROBE_ENA respectively to the value:
    0: data will be registered with the speed of the system clock *40 MHz* (every *25 ns*),
    1:data will be registered with the speed of the FEL clock *1 MHz* (every *1 μs*)
  - register DAQ_TIMER_LIMIT to the value of successive number of data registered in the DAQ memories in the range of from *0* to *2047*. The set values are diminished by *1*, i.e. for the value *0* the registration of a single data is done.
  - register DAQ_ DELAY to the value of signal delay *CAVITY TRIGGER*. The delay means the number of clock signals of *1 MHz,* or a single step is *1 μs*. Assuming the value of 0 means no additional delay for the signal *CAVITY TRIGGER*.
- $\boxed{\text{B}}$ module initialization *DAQ TIMER* via setting DAQ_TIMER_START=1.
- $\boxed{\text{C}}$ module activation *DAQ TIMER* via setting DAQ_TIMER_ENA=1. From this moment on, the block waits for the signal *CAVITY TRIGGER*, which synchronously triggers the data acquisition process.
- $\boxed{\text{D}}$ automatic triggering of the delay process for the signal *CAVITY TRIGGER*. The delay value is determined by the value of the register DAQ_DELAY
- $\boxed{\text{E}}$ automatic triggering of the data acquisition process delayed by the signal *CAVITY TRIGGER*. The counter starts DAQ_TIMER_COUNT which counts the amount of the registered data.
- $\boxed{\text{F}}$ automatic ending of the data acquisition process, after the counter DAQ_TIMER_COUNT reaches the value set in the register DAQ_TIMER_LIMIT. The flag is set DAQ_TIMER_STOP=1.
- $\boxed{\text{G}}$ checking reading of the flag value DAQ_TIMER_STOP. Reading of the value *0* means, that the data reading process still lasts. Reading the value *1* means , that the DAQ process was finished. Reading of the flag state may be done many times, waiting for the end moment of the DAQ process.
- $\boxed{\text{H}}$ stopping of the work of the module *DAQ TIMER* via setting DAQ_TIMER_ENA=0.
- $\boxed{\text{I}}$ introduction of the module in the blocked state *DAQ TIMER* via setting DAQ_TIMER_START=0. The flag is deleted DAQ_TIMER_STOP=0. In this operation mode of the *DAQ TIMER* it is possible to switch the operation modes of the block DAQ, for example to read the contents of the memories DAQ1..*DAQ4* during operation mode *DAQ_MEMORY_ACCESS* (see chapter 11.2.2).

If the global DAQ conditions remain unchanged, at the next initialization of the DAQ process, the stage $\boxed{\text{A}}$ may be omitted.

11.2.4 DAQ vector generator

The work in operation mode *DAQ_VECTOR_GENERATOR* relies on periodic generation of the of the memory contents DAQ1.. DAQ3 synchronously with the signal *CAVITY TRIGGER*. In the operational sense, the generators are acting identically as *CONTROL_DATA_TABLES*. It requires only previous data loading in the operation mode *DAQ_MEMORY_ACCESS* (see chapter 11.2.2).

The operation mode *DAQ_VECTOR_GENERATOR* is now in the testing period.

12 INPUT MULTIPLEXERS BLOCK DESCRIPTION

The input multiplexers allow for nondependent programmable choice of the input control signals for the blocks `CAVITY SIMULATOR` and `CAVITIES CONTROLLER`. Choice of the multiplexer input signals provides the realization of the feedbacks (external analog, internal digital) between the DSP blocks and nondependent control of the particular DSP blocks (external analog, internal testing digital).

12.1 Functional structure

The block `INPUT MULTIPLEXERS` provides simultaneous choice of the two input signals for the block `CAVITY SIMULTOR` and a single signal for the block `CAVITIES CONTROLLER`. Both multiplexers have 4 variants for the choice of the inputs. . The functional structure of the block `INPUT MULTIPLEXERS` was presented in figure 23.

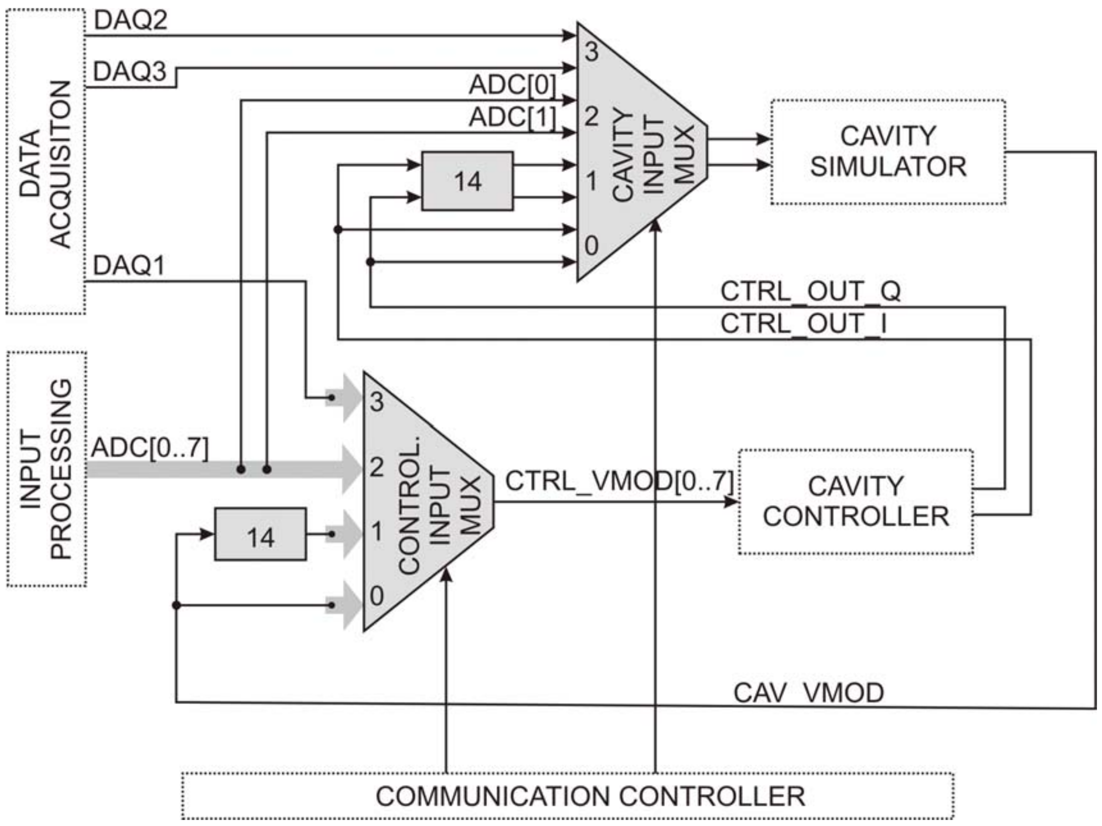


Fig. 23. Functional structure of the block `INPUT MULTIPLEXERS`

For each of the multiplexers, the following types of the input signals are distinguished:

- *channel 0*: realization of full resolution (18-bit) of the digital feedback between the DSP blocks of the CAVITIES CONTROLLER and simulator. The input signal is transmitted to all channels in parallel for the block `CAVITIES CONTROLLER` (see chapter 10.1),
- *channel 1*: digital simulation of the analog (14-bit) feedback, respectively between the DSP blocks of CAVITIES CONTROLLER and simulator, simulating the resolution of the AD and DA converters. The input signal is transmitted to all channels in parallel for the block `CAVITIES CONTROLLER` (see chapter. 10.1),
- *channel 2*: connection of respective signals from the AD converters from blocks `ADC[0..7],` to particular input channels of block `CAVITIES CONTROLLER`. For the

block CAVITY SIMULATOR, the signal ADC[0] is connected to input CAV_IN_I and respectively signal ADC[1] is connected to input CAV_IN_Q (see chapter 9.1),

- *channel 3*: connection of blocks DAQ1..DAQ3 with the choice of the VECTOR operation mode (see chapter 4.2.3) or zeroing the inputs for the rest of the operation modes. For the block CAVITIES CONTROLLER the input signal DAQ1 is output to all channels in parallel. For the block CAVITY SIMULATOR the signal DAQ1 is connected to the input DAQ3 and respectively the signal ADC[1] is connected to input CAV_IN_Q (see chapter 9.1),

12.2 Programming description

The choice of the source channel (compare figure 20) is done for each multiplexer nondependently. The number of the chosen channel in the range of from 0 to 3 is programmed in the block COMMUNICATION CONTROLLER:

- For the module *CAVITY INPUT MUX* in the register MUX_IN_CAVITY
- For the module *CONTROLLER INPUT MUX* in the register MUX_IN_CONTRL

For the servicing purposes the choice of the active multiplexer channel, via the block COMMUNICATION CONTROLLER may be done during an arbitrary moment of the SIMCON activity. **The users are strongly recommended to set the registers only during the *SETUP MODE OPERATION.***

13 OUTPUT SWITCH MATRIX BLOCK DESCRIPTION

The switching matrix realized in the block `OUTPUT SWITCH MATRIX` allows for nondependent programmable choice of the output signals from the blocks `PROGRAMMABLE DATA CONTROLLER`, `CAVITY SIMULTOR`, `CAVITIES CONTROLLER`, `ADC[0…7]` and from the module *TEST GENERATOR* to the inputs of the blocks `DAQ[0..1]` and `DAC1..2`. The choice possibility for input signals of multiplexers provides realization of monitoring of particular signals and choice of signals output in the analog form from the SIMCON system to the outer world, via the DAC converters.

13.1 Functional structure

The block `OUTPUT SWITCH MATRIX` is a switching matrix of 23 inputs to 6 outputs. It enables a simultaneous choice of the output signals for four `DAQ` blocks and for two `DAC` channels. All 23 input signals may be nondependently connected to each output. The functional structure of the block `OUTPUT SWITCH MATRIX` is presented in figure 24.

The module *TEST GENERATOR* generates a rising saw-like signal initialized by the

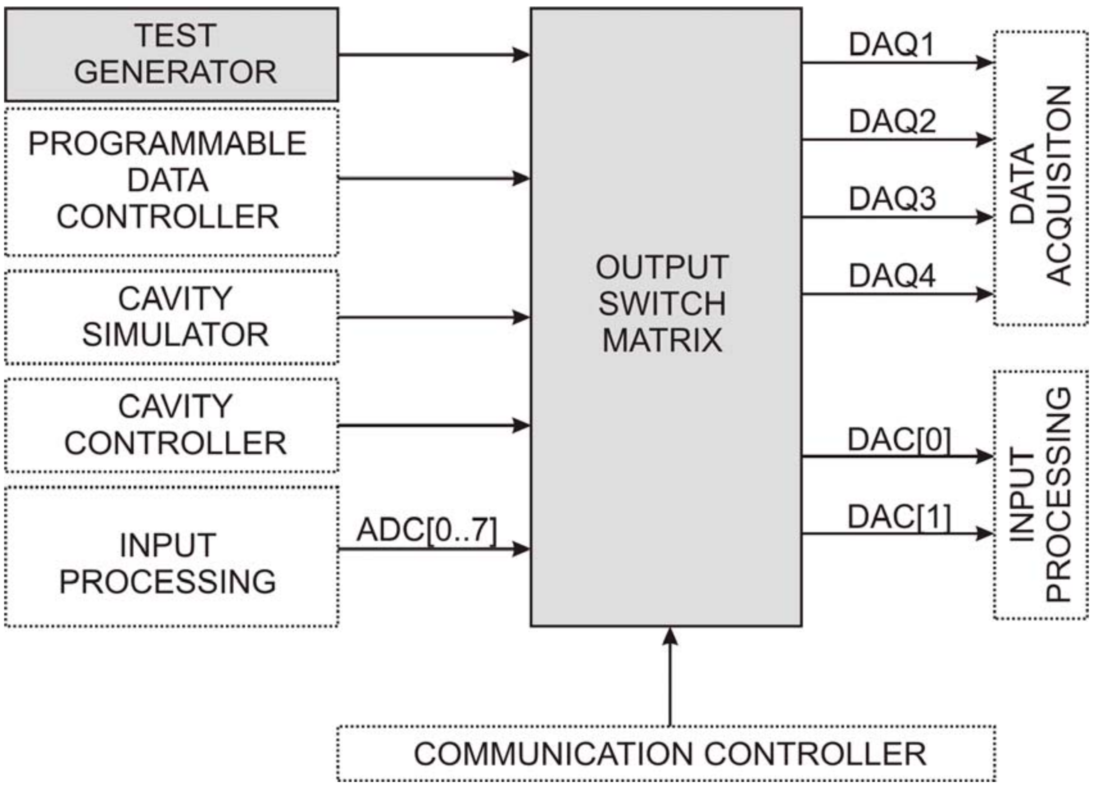


Fig. 24. Functional structure of the block `INPUT MULTIPLEXERS`

signal *CAVITY TRIGGER*. The initialization of the generator causes its setting to 0 value and next each clock of the signal *SIMCON CLOCK* increases this value by 1. The generator gives 18-bit values in a periodic way.

For each input channel there are distinguished the following kinds of the input signals of 18-bit in resolution:

- *channel 0*: test signal from module *TEST GENERATOR*,
- *channel 1*: external signal *CAV_ OUT_I* (compare chapter 9.1),
- *channel 2*: internal signal *CAV_ OUT_Q* (compare chapter 9.1),
- *channel 3*: internal signal *CAV_ DETUN* (compare chapter 9.1),

- *channel 4*: internal signal *CAV_VMOD* (see chapter 9.1),
- *channels 5-12*: internal signal *CTRL_DET_I[0..7]* (compare chapter10.1),
- *channels 13-20*: internal signal *CTRL_DET_Q[0..7]* (compare chapter10.1),
- *channel 21*: internal signal *CTRL_I* (compare chapter10.1),
- *channel 22*: internal signal *CTRL_Q* (compare chapter10.1),
- *channel 23*: internal signal *TGAIN_I* (compare chapter8.1),
- *channel 24:* internal signal *TGAIN_Q* (compare chapter8.1),
- *channel 25*: internal signal *TSETPOINT_I* (compare chapter8.1),
- *channel 26*: internal signal *TSETPOINT_Q* (compare chapter8.1),
- *channel 27* : internal signal *TFEEDFORWARD _I* (compare chapter8.1),
- *channel 28*: internal signal *TFEEDFORWARD _Q* (compare chapter8.1),
- *channel 29*: internal signal *TBEAM _I* (compare chapter8.1),
- *channel 30*: internal signal *TBEAM _Q* (compare chapter8.1),
- *channel 31*: internal signal *CAV_MODE1* (compare chapter 9.1),
- *channel 32*: internal signal *CAV_MODE1D* (compare chapter 9.1),
- *channel 33*: internal signal *CAV_MODE2* (compare chapter 9.1),
- *channel 34*: internal signal *CAV_MODE2D* (compare chapter 9.1),
- *channel 35*: internal signal *CAV_MODE3* (compare chapter 9.1),
- *channel 36*: internal signal *CAV_MODE3D* (compare chapter 9.1),
- *channels 37-44*: input signal *ADC[0..7]* (compare chapter6.1),
- *channel 45*: internal signal *SUMV_I* (compare chapter.10.1),
- *channel 46*: internal signal *SUMV_Q* (compare chapter 10.1),

13.2 Programming description

The choice of the source channel (compare figure 20) is done for each output of the switching matrix nondependently. The number of the input channel in the range of from 0 to 22 is programmed in the block `COMMUNICATION CONTROLLER`:

- for the block `DAQ1` for the register MUX_OUT_DAQ1,
- for the block `DAQ2` for the register MUX_OUT_DAQ2,
- for the block `DAQ3` for the register MUX_OUT_DAQ3,
- for the block `DAQ4` for the register MUX_OUT_DAQ4,
- for the block `DAC[0]` for the register MUX_OUT_DAC0,
- for the block `DAC[1]` for the register MUX_OUT_DAC1.

The choice of improper channels numbers from the range 23-31 automatically switches the channel 0. **The users are strongly advised not to set the improper channel values.**

For the servicing purposes, the choice of the active channels via the block `COMMUNICATION CONTROLLER` may be done during an arbitrary moment of the SIMCON activity. **The users are strongly advised to set the choice for the multiplexer channel numbers only during the *SETUP MODE OPERATION.***

14 PROGRAMMABLE I/O SPECIFICATION

This chapter presents the specification for the I/O space of the SIMCON system, which is made accessible for the priority computer control via the block COMMUNICATION CONTROLLER.

14.1 I/O specification list by addresses

CHECKSUM (0000H)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHECKSUM																															

CHECKSUM (RO) - contains constant hexadecimal control value: 0029633BH.

CREATOR (0001H)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CREATOR																															

CREATOR (RO) - contains constant ASCII symbol which identifies the constructor (group „*E*L*H*EP-*WA*RSAW”): „EHWA”, what in the hexadecimal reading means the value: 45485741H.

IDENTIFIER (0002H)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDENTIFIER																															

IDENTIFIER (RO) - contains constant ASCII symbol identifying the system: „SIMC”, what in the case of hexadecimal reading means: 4D435452H.

VERSION (0003H)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAIN_VERSION								SUB_VERSION								REV_VERSION															

Packet VERSION - contains constant identifier of the version expressed hexadecimally:

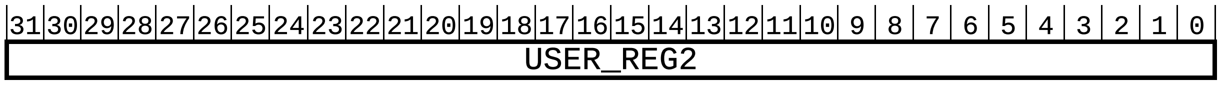
- MAIN_VERSION (RO): contains value 03H,
- SUB_VERSION (RO): contains value 00H,
- REV_VERSION (RO): contains value 0001H,

USER_REG1 (0004H)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USER_REG1																															

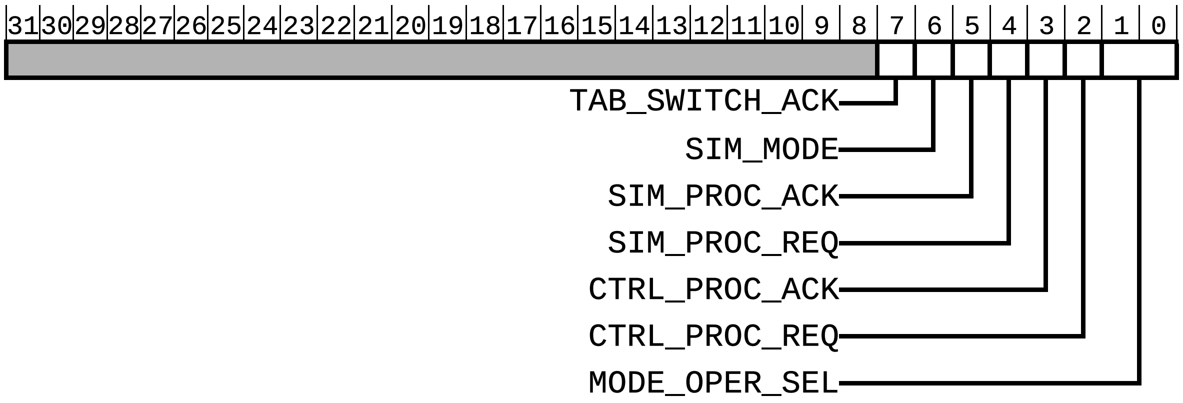
USER_REG1 (RW) – control-test register designed solely for the user.

USER_REG2 (0005H)



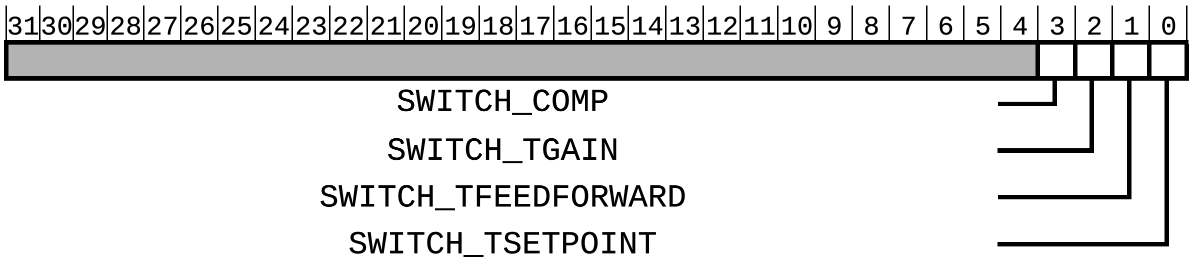
USER_REG2 (RW) – control-test register designed solely for the user.

STATUS (0006H)



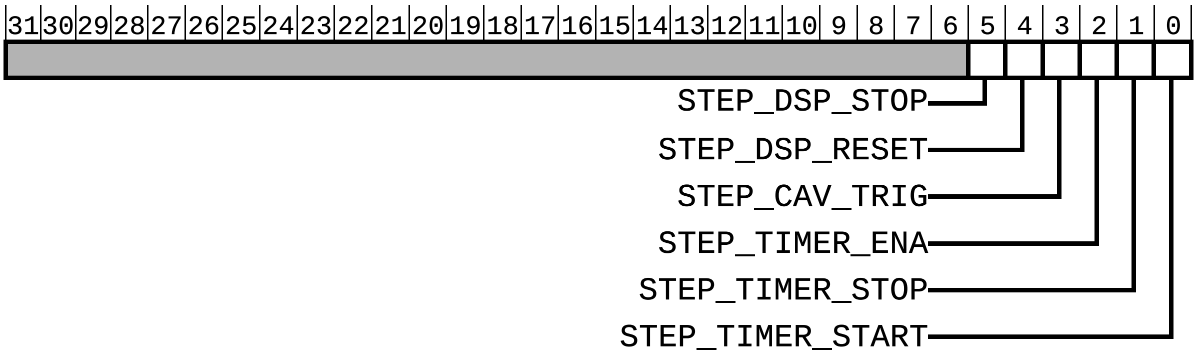
Packet STATUS contains global components of the control of SIMCON system:
- MODE_OPER_SEL (RW): see chapter 4.2,
- CTRL_PROC_REQ (RW): see chapter 4.2,
- CTRL_PROC_ACK (R): see chapter 4.2,
- SIM_PROC_REQ (RW): see chapter 4.2,
- SIM_PROC_ACK (R): see chapter 4.2,
- SIM_MODE (RW): see chapter 4.2,
- TAB_SWITCH_ACK (R): see chapter 4.2,

SWITCH_TAB (0007H)



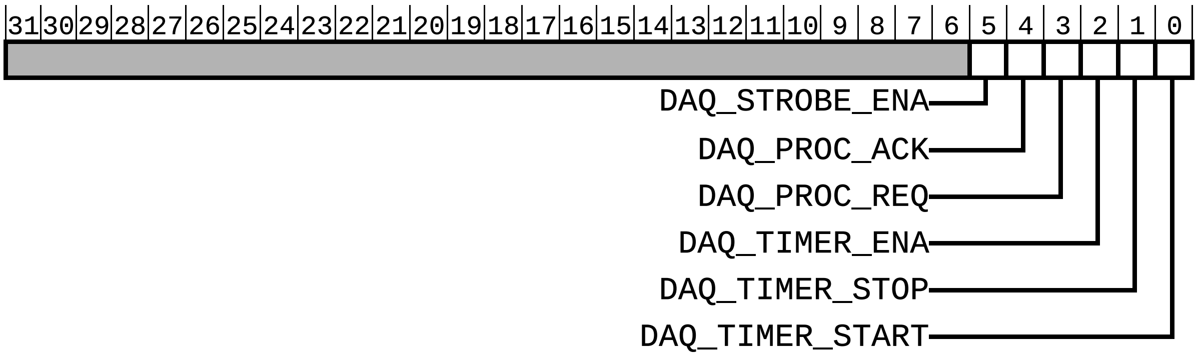
Packet SWITCH_TAB contains control components for step mode of SIMCON system:
- SWITCH_COMP (RW): see chapter 8.2.3,
- SWITCH_TGAIN (RW): see chapter 8.2.3,
- SWITCH_TFEEDFORWARD (RW): see chapter 8.2.3,
- SWITCH_TSETPOINT (RW): see chapter 8.2.3,

STEP (0008H)



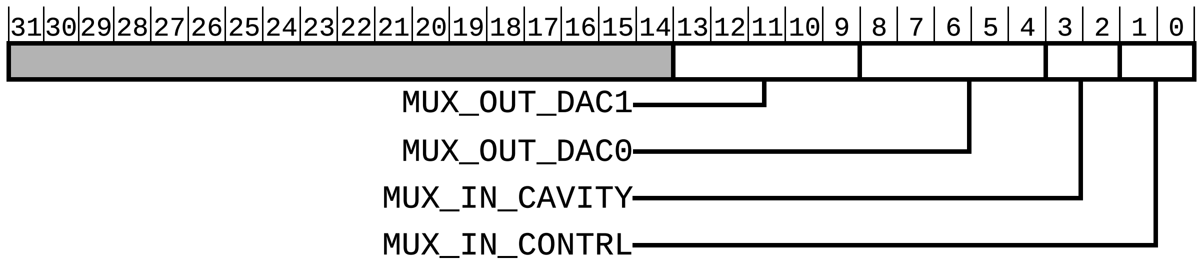
Packet STEP contains components for the step control of the SIMCON system:
- STEP_TIMER_START (RW): see chapter 5.3.2,
- STEP_TIMER_STOP (RO): see chapter 5.3.2,
- STEP_TIMER_ENA (RW): see chapter 5.3.2,
- STEP_CAV_TRIG (RW): see chapter 5.3.2,
- STEP_DSP_RESET (RW): see chapter 5.3.2,
- STEP_DSP_STOP (RO): see chapter 5.3.2.

DAQ (0009H)



Packet DAQ contains control components for the DAQ process of the SIMCON system:
- DAQ_TIMER_START (RW): see chapter 11.2.3,
- DAQ_TIMER_STOP (RO): see chapter 11.2.3,
- DAQ_TIMER_ENA (RW): see chapter 11.2.3,
- DAQ_PROC_REQ (RW): see chapter 11.2.1,
- DAQ_PROC_ACK (RO): see chapter 11.2.1,
- DAQ_STROBE_ENA (RW): see chapter 11.2.3.

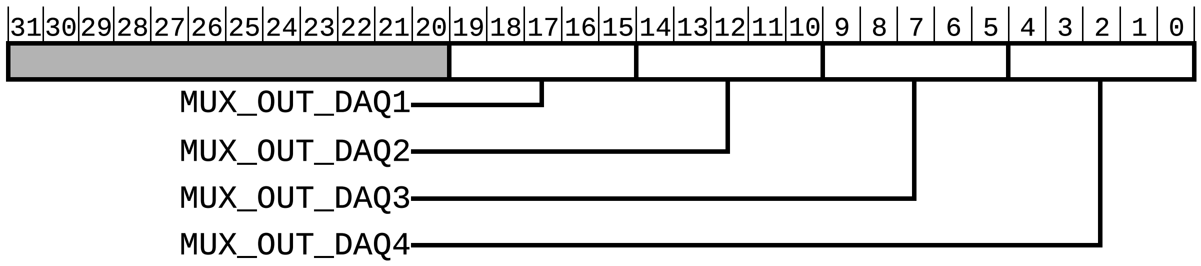
SIGNAL_MUX (000AH)



Packet SIGNAL_MUX contains control components for input DSPs and DACs signal:

- MUX_IN_CONTRL (RW): see chapter 12.2,
- MUX_IN_CAVITY (RW): see chapter 12.2,
- MUX_OUT_DAC0 (RW): see chapter 13.2,
- MUX_OUT_DAC1 (RW): see chapter 13.2.

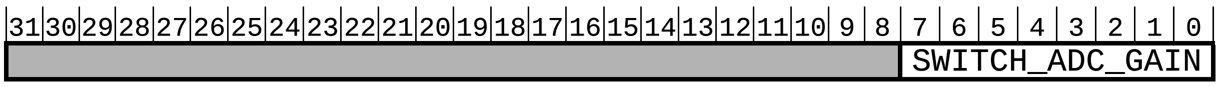
DAQ_MUX (000BH)



Packet DAQ_MUX contains control components for multiplexers for DAQ blocks:

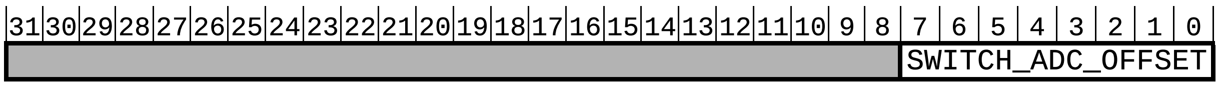
- MUX_OUT_DAQ1 (RW): see chapter 13.2,
- MUX_OUT_DAQ2 (RW): see chapter 13.2,
- MUX_OUT_DAQ3 (RW): see chapter 13.2,
- MUX_OUT_DAQ4 (RW): see chapter 13.2.

SWITCH_ADC_GAIN (000CH)



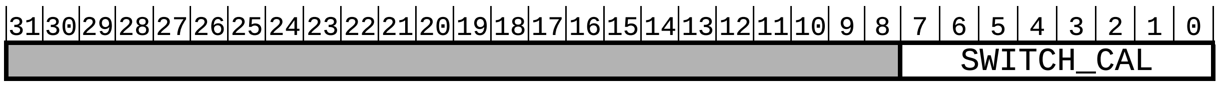
Register SWITCH_ADC_GAIN (RW) – see chapter 8.2.3.

SWITCH_ADC_OFFSET (000DH)



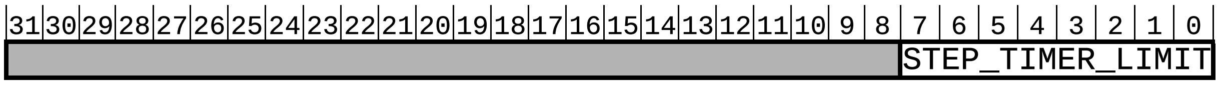
Register SWITCH_ADC_OFFSET (RW) – see chapter 8.2.3.

SWITCH_CAL (000EH)



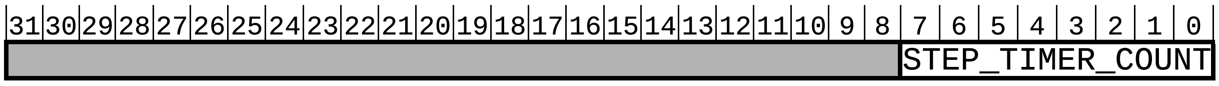
Register SWITCH_CAL (RW) – see chapter 8.2.3.

STEP_TIMER_LIMIT (000FH)



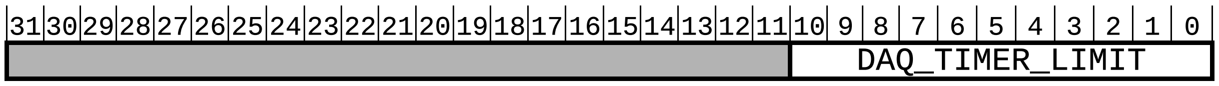
Register STEP_TIMER_LIMIT (RW) – see chapter 5.3.2.

STEP_TIMER_COUNT (0010H)



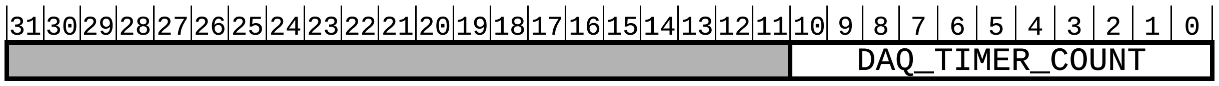
Register STEP_TIMER_COUNT (RO) – see chapter 5.3.2.

DAQ_TIMER_LIMIT (0011H)



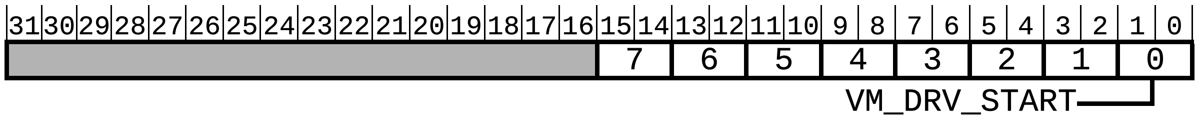
Register DAQ_TIMER_LIMIT (RW) – see chapter 11.2.3.

DAQ_TIMER_COUNT (0012H)



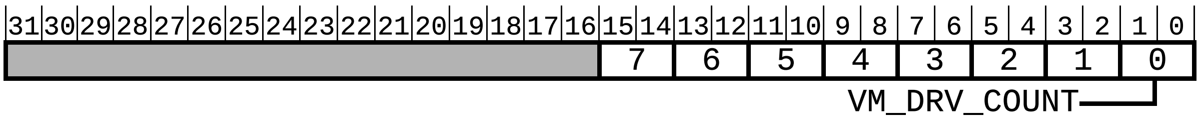
Register DAQ_TIMER_COUNT (RW) – see chapter 11.2.3.

VM_DRV_START (0013H)



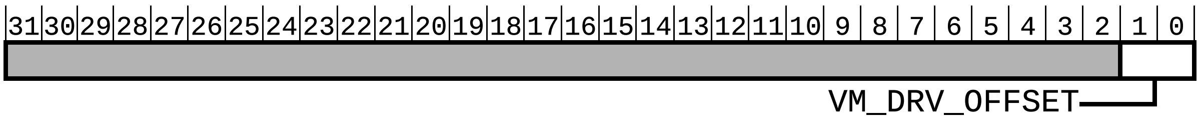
Register VM_DRV_START[0..7] (RW) – see chapter 8.2.

VM_DRV_COUNT (0014H)



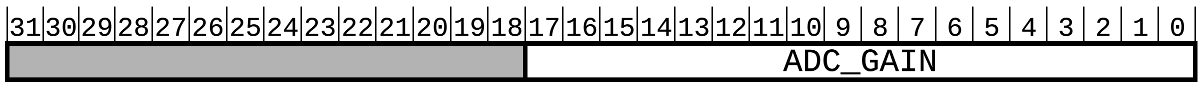
Register VM_DRV_COUNT[0..7] (RW) – see chapter 8.2.

VM_DRV_OFFSET (0015H)



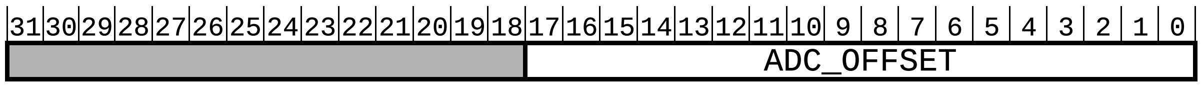
Register VM_DRV_OFFSET (RW) – see chapter 9.2.

ADC_GAIN (0016H-002DH)



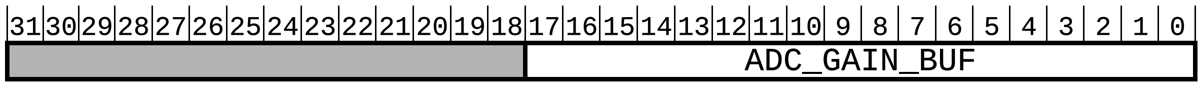
Register ADC_GAIN[0..7] (RW) – see chapter 6.2.

ADC_OFFSET (002EH-0025H)



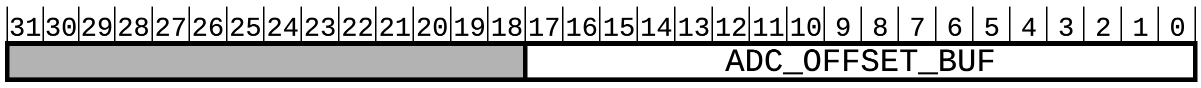
Register ADC_OFFSET[0..7] (RW) – see chapter 6.2.

ADC_GAIN_BUF (0026H-002DH)



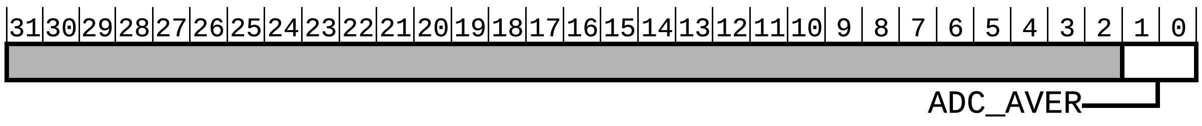
Register ADC_GAIN_BUF[0..7] (RW) – see chapter 6.2.

ADC_OFFSET_BUF (002EH-0035H)



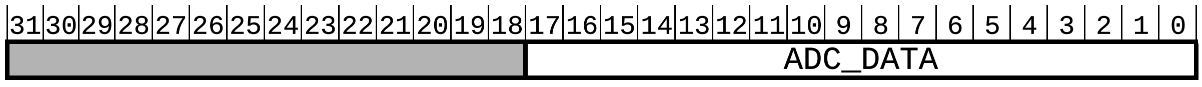
Register ADC_OFFSET_BUF[0..7] (RW) – see chapter 6.2.

ADC_AVER (0036H)



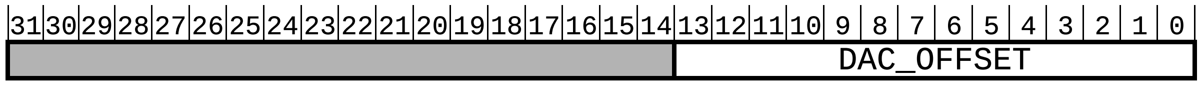
Register ADC_AVER (RW) – see chapter 6.2.

ADC_DATA (0037H-003EH)



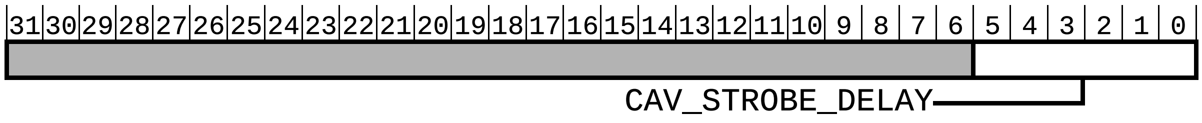
Register ADC_DATA[0..7] (RO) – see chapter 6.2.

DAC_OFFSET (003FH-0040H)



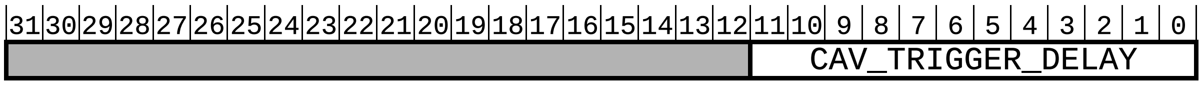
Register DAC_OFFSET[0..1] (RO) – see chapter 6.2.

CAV_STROBE_DELAY (0041H)



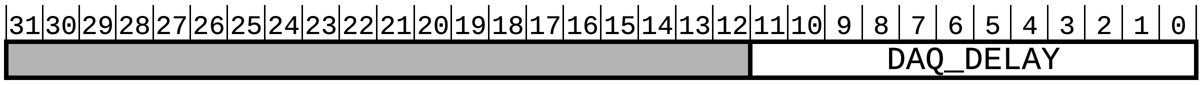
Register CAV_STROBE_DELAY (RW) – see chapter 5.3.3.

CAV_TRIGER_DELAY (0042H)



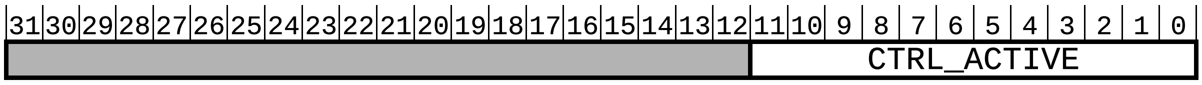
Register CAV_TRIGGER_DELAY (RW) – see chapter 5.3.3.

DAQ_DELAY (0043H)



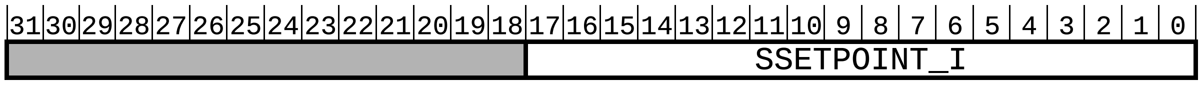
Register DAQ_DELAY (RW) – see chapter 5.3.3.

CTRL_ACTIVE (0044H)



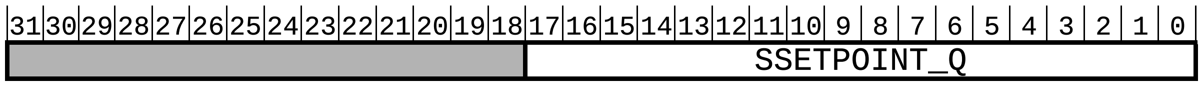
Register CTRL_ACTIVE (RW) – see chapter 10.2.

SSETPOINT_I (0045H)



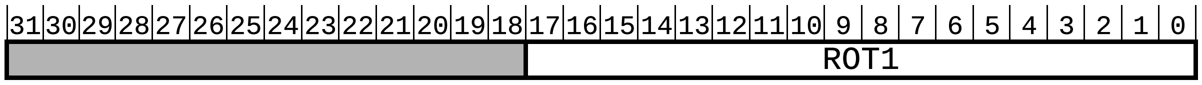
Register SSETPOINT_I (RW) – see chapter 8.2.5.

SSETPOINT_Q (0046H)



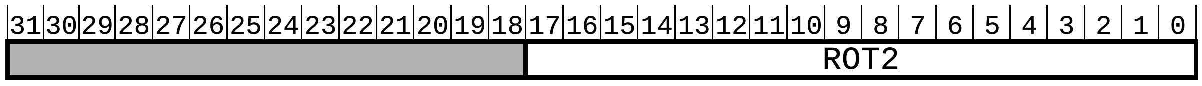
Register SSETPOINT_Q (RW) – see chapter 8.2.5.

ROT1 (0047H-004EH)



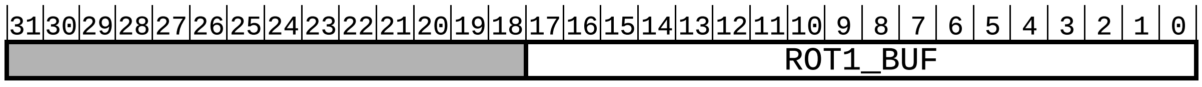
Register ROT1[0..7] (RW) – see chapter 8.2.4.

ROT2 (004FH-0056H)



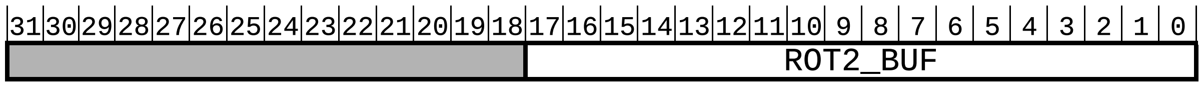
Register ROT2[0..7] (RW) – see chapter 8.2.4.

ROT1_BUF (0057H-005EH)



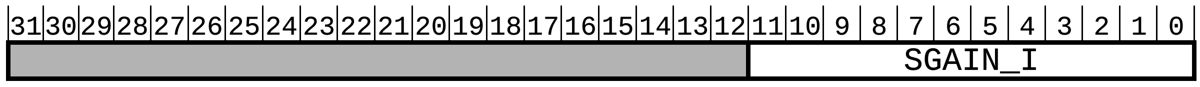
Register ROT1_BUF[0..7] (RW) – see chapter 8.2.4.

ROT2_BUF (005FH-0066H)



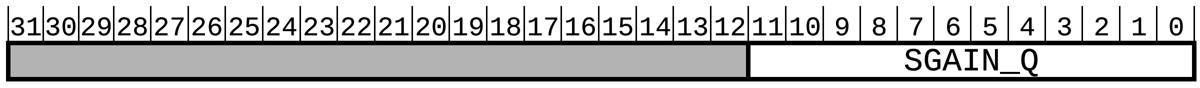
Register ROT2_BUF[0..7] (RW) – see chapter 8.2.4.

SGAIN_I (0067H)



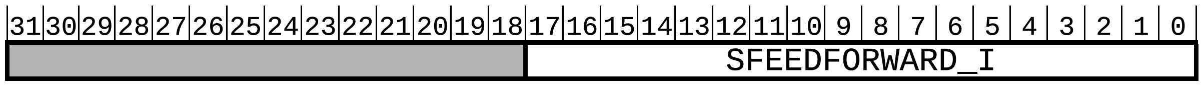
Register SGAIN_I (RW) – see chapter 8.2.5.

SGAIN_Q (0068H)



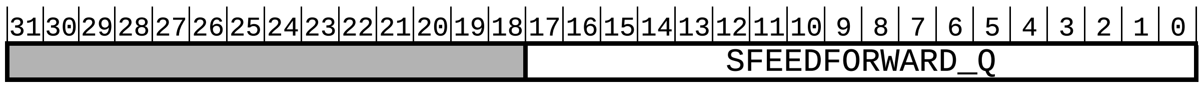
Register SGAIN_Q (RW) – see chapter 8.2.5.

SFEEDFORWARD_I (0069H)



Register (RW) SFEEDFORWARD_I– see chapter 8.2.5.

SFEEDFORWARD_Q (006AH)



Register SFEEDFORWARD_Q (RW) – see chapter 8.2.5.

COMP1 (006BH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	COMP1																	

Register COMP1 (RW) –see chapter 8.2.5.

COMP2 (006CH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	COMP2																	

Register COMP2 (RW) – see chapter 8.2.5.

COMP1_BUF (006DH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	COMP1_BUF																	

Register COMP1_BUF (RW) –see chapter 8.2.5.

COMP2_BUF (006EH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	COMP2_BUF																	

Register COMP2_BUF (RW) – see chapter 8.2.5.

CTRL_DET_I (006FH-0076H)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	CTRL_DET_I																	

Register CTRL_DET_I[0..7] (RW) – see chapter 10.2.

CTRL_DET_Q (0077H-007EH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	CTRL_DET_Q																	

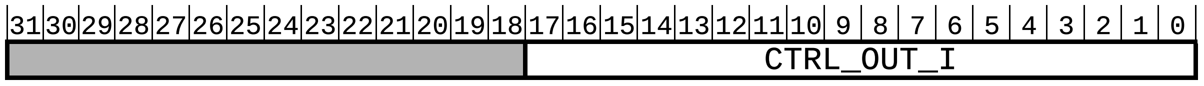
Register CTRL_DET_Q[0..7] (RW) – see chapter 10.2.

CTRL_VMOD (007FH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	CTRL_VMOD																	

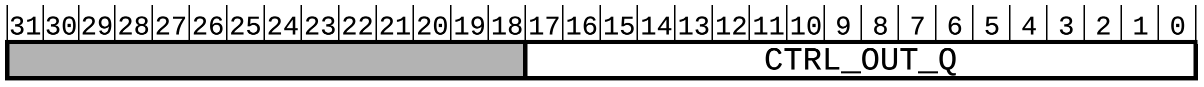
Register CTRL_VMOD (RW) – see chapter 10.2.

CTRL_OUT_I (0080H)



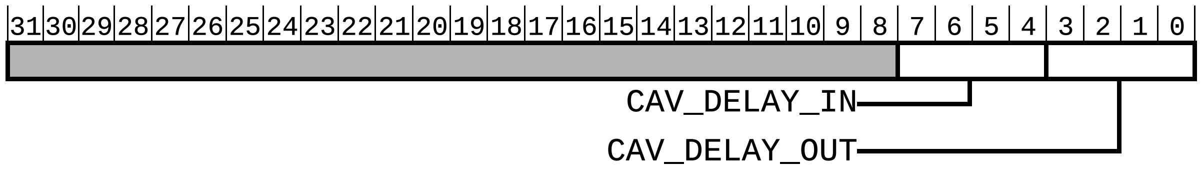
Register CTRL_OUT_I (RW) – see chapter 10.2.

CTRL_OUT_Q (0081H)



Register CTRL_OUT_Q (RW) – see chapter 10.2.

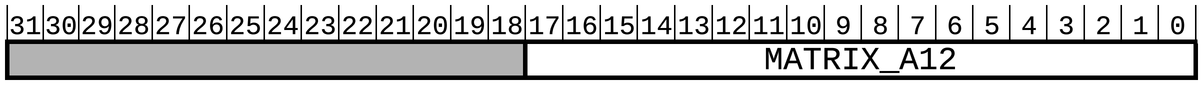
CAV_DELAY (0082H)



Packet CAV_DELAY contains control components for the delays of input and output signals of the DSP process of the cavity simulator:

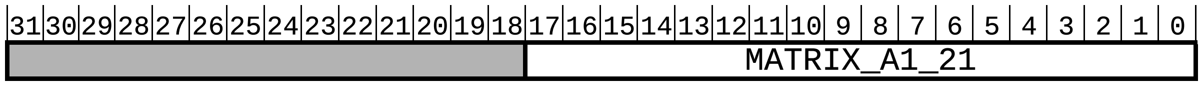
- CAV_DELAY_IN (RW): see chapter 9.2,
- CAV_DELAY_OUT (RW): see chapter 9.2,

MATRIX_A12 (0083H)



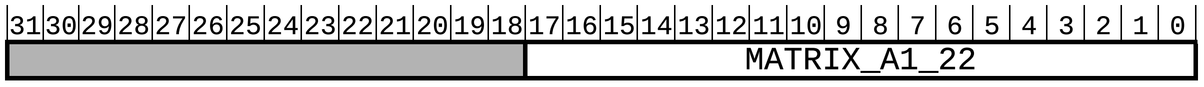
Register MATRIX_A12 (RW) – see chapter 8.2.4.

MATRIX_A1_21 (0084H)



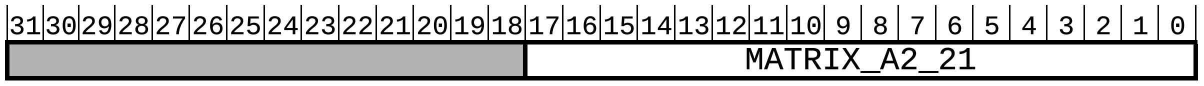
Register MATRIX_A1_21 (RW) – see chapter 8.2.4.

MATRIX_A1_22(0085H)



Register MATRIX_A1_22 (RW) – see chapter 8.2.3.

MATRIX_A2_21 (0086H)



Register MATRIX_A2_21 (RW) – see chapter 8.2.3.

MATRIX_A2_22 (0087H)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	MATRIX_A2_22																	

Register MATRIX_A2_22 (RW) – see chapter 8.2.3.

MATRIX_A3_21 (0088H)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	MATRIX_A3_21																	

Register MATRIX_A3_21 (RW) – see chapter 8.2.3.

MATRIX_A3_22 (0089H)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	MATRIX_A3_22																	

Register MATRIX_A3_22 (RW) – see chapter 8.2.3.

MATRIX_B1_1 (008AH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	MATRIX_B1_1																	

Register MATRIX_B1_1 (RW) – see chapter 8.2.3.

MATRIX_B2_1 (008BH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	MATRIX_B2_1																	

Register MATRIX_B2_1 (RW) – see chapter 8.2.3.

MATRIX_B3_1 (008CH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	MATRIX_B3_1																	

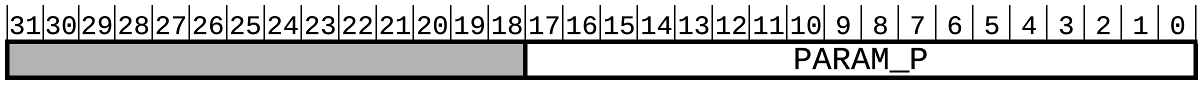
Register MATRIX_B3_1 (RW) – see chapter 8.2.3.

PARAM_H (008DH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	PARAM_H																	

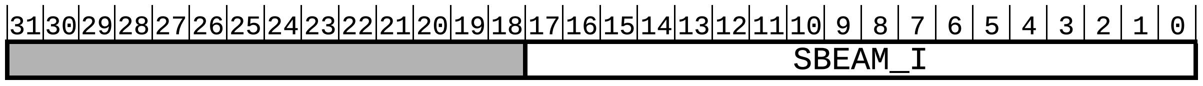
Register PARAM_H (RW) – see chapter 8.2.3.

PARAM_P (008EH)



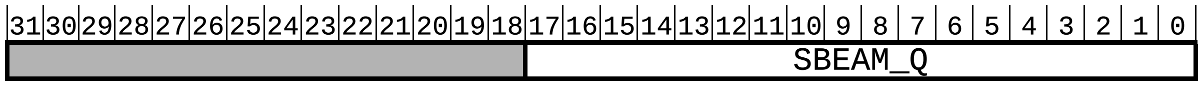
Register PARAM_P (RW) – see chapter 8.2.3.

SBEAM_I (008FH)



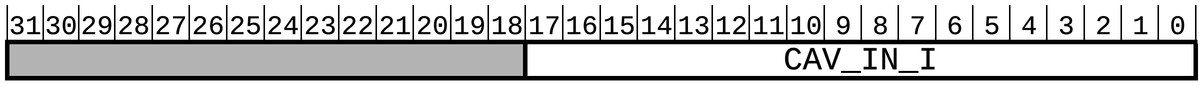
Register SBEAM_I (RW) – see chapter 8.2.3.

SBEAM_Q (0090H)



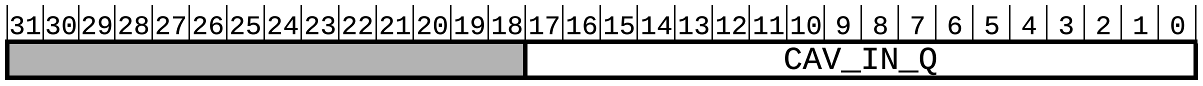
Register SBEAM_Q (RW) – see chapter 8.2.3.

CAV_IN_I (0091H)



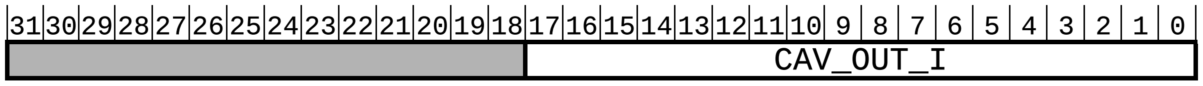
Register CAV_IN_I (RW) – see chapter 9.2.

CAV_IN_Q (0092H)



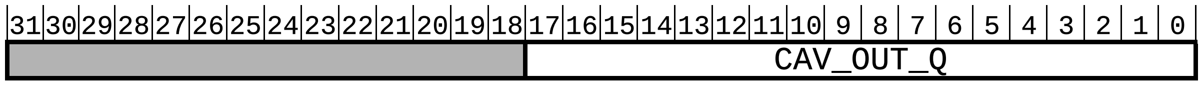
Register CAV_IN_Q (RW) – see chapter 9.2.

CAV_OUT_I (0093H)



Register CAV_OUT_I (RO) – see chapter 9.2.

CAV_OUT_Q (0094H)



Register CAV_OUT_Q (RO) – see chapter 9.2.

CAV_VMOD (0095H)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	CAV_VMOD																	

Register CAV_VMOD (RO) – see chapter 9.2.

CAV_DETUN (0096H)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	CAV_DETUN																	

Register CAV_DETUN (RO) – see chapter 9.2.

CAV_MODE1 (0097H)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	CAV_MODE1																	

Register CAV_MODE1 (RO) – see chapter 9.2.

CAV_MODE1D (0098H)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	CAV_MODE1D																	

Register CAV_MODE1D (RO) – see chapter 9.2.

CAV_MODE2 (0099H)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	CAV_MODE2																	

Register CAV_MODE2 (RO) – see chapter 9.2.

CAV_MODE2D (009AH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	CAV_MODE2D																	

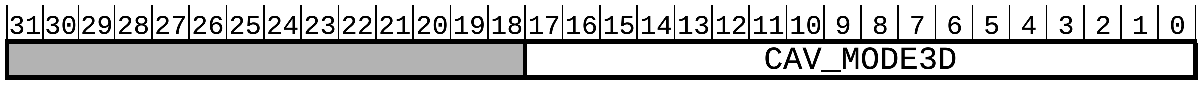
Register CAV_MODE2D (RO) – see chapter 9.2.

CAV_MODE3 (009BH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	CAV_MODE3																	

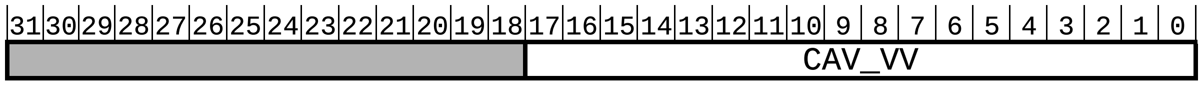
Register CAV_MODE3 (RO) – see chapter 9.2.

CAV_MODE3D (009CH)



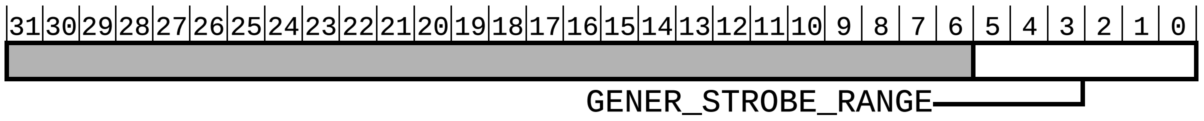
Register CAV_MODE3D (RO) – see chapter 9.2.

CAV_VV (009DH)



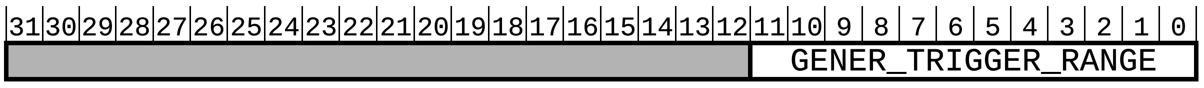
Register CAV_VV (RO) – see chapter 9.2.

GENER_STROBE_RANGE (009EH)



Register GENER_STROBE_RANGE (RW) – see chapter 5.3.1.

GENER_TRIGER_RANGE (009FH)



Register GENER_TRIGGER_RANGE (RW) – see chapter 5.3.1.

TSETPOINT_I (0800H-0FFFH)

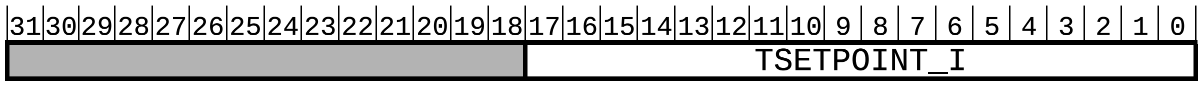


Table TSETPOINT_I (RW) – see chapter 8.2.5.

TSETPOINT_Q (1000H-17FFH)

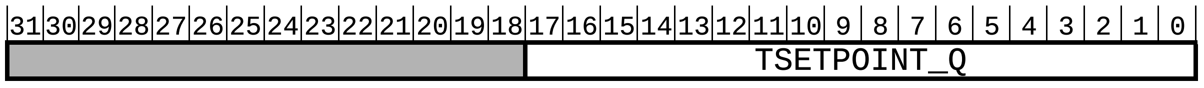


Table TSETPOINT_Q (RW) – see chapter 8.2.5.

TFEEDFORWARD_I (1800H-1FFFH)

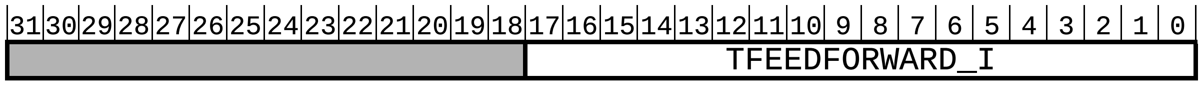


Table TFEEDFORWARD_I (RW) – see chapter 8.2.5.

TFEEDFORWARD_Q (2000H-27FFH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	TFEEDFORWARD_Q																	

Table TFEEDFORWARD_Q (RW) – see chapter 8.2.5.

TGAIN_I (2800H-2FFFH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	TGAIN_I											

Table TGAIN_I (RW) – see chapter 8.2.5.

TGAIN_Q (3000H-37FFH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	TGAIN_Q											

Table TGAIN_Q (RW) – see chapter 8.2.5.

TBEAM_I (3800H-3FFFH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	TBEAM_I											

Table TBEAM_I (RW) – see chapter 8.2.3.

TBEAM_Q (4000H-47FFH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	TBEAM_Q											

Table TBEAM_Q (RW) – see chapter 8.2.3.

DAQ1 (4800H-4FFFH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	DAQ1																	

Table DAQ1 (RW) – see chapter 11.2.

DAQ2 (5000H-57FFH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	DAQ2																	

Table DAQ2 (RW) – see chapter 11.2.

DAQ3 (5800H-5FFFH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	DAQ3																		

Table DAQ3 (RW) – see chapter 11.2.

DAQ4 (6000H-67FFH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	DAQ4																		

Table DAQ4 (RW) – see chapter 11.2.

14.2 I/O specification list by names

A LLRF PLATFORM DESCRIPTION

The LLRF Control Platform [2] was adapted toward the key requirements of the TESLA experiment [5,6]. The control of the system was provided via the VMB bus [12] and the DOOCS control interface [1]. The construction provides distribution of clock signals and the work of distributed modules in a synchronous way. General functional diagram of the Modular Control Platform is presented in fig. 25.

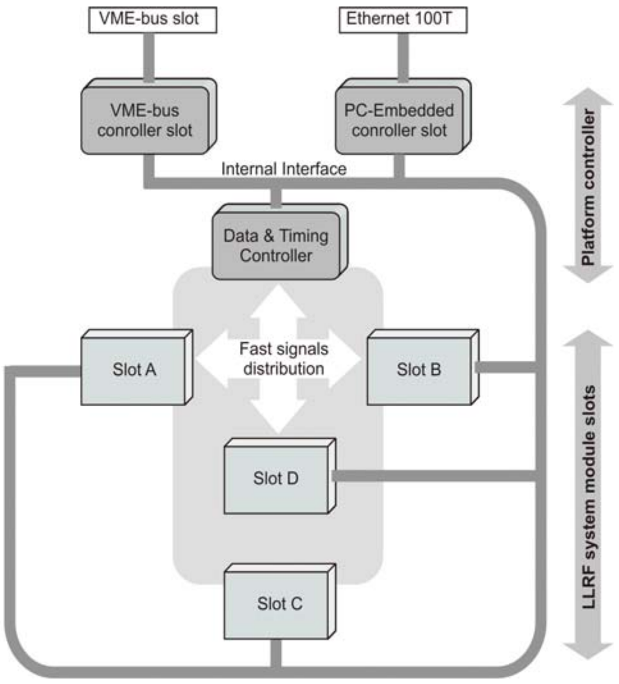


Fig. 25. Functional structure of LLRF modular platform

The block PLATFORM CONTROLLER consists of three closely co-working modules:

- The VME bus communication is realized by the module VME BUS CONTROLLER. It is a variant of a passive control, in which full control over the LLRF is provided by outside controller. This solution enables integration of the LLRF platform together with executive (actuating) modules with the DOOCS programming environment [1]. The control is done via SUN-SPARC computer which has the VME controller.
- The active control is provided by an alternative communication module PC EMBEDDED CONTROLLER. Implementation of a PC allows to realize inside the board of complex control, data acquisition, data processing, and monitoring operations for the LLRF system. Standard Ethernet connection was used for communications with the outside control systems, and with particular blocks of the LLRF system [10].
- The module DATA & TIMING CONTROLLER provides global distribution of fast synchronization signals and data to all slots. The distribution system resides on a programmable matrix FPGA. Its functionality may be modified depending on the current needs of the LLRF control system. In particular, the configuration of the executive hardware blocks, positioned in different platform slots, may be changed.

The communications inside the platform modules is realized using the proprietary standard INTERNAL INTERFACE [7]. The modules VME BUS CONTROLLER and PC EMBEDDED CONTROLLER play the role of communication bus controller. The rest of modules work in the „slave" work mode. The communication layer of the platform with upper supervisory computer system provides all the mechanisms to configure programmable the FPGA chips which are present on the platform and which are on the functional blocks inserted to the platform slots A - D.

The platform has four universal user slots predicted for insertion of functional modules. The slots A, B and C are configured in parallel, this is peer to peer, and the slot D was configured as a central one. Direct signal connections are provided between slots. They assure transmission of fast data and synchronization signals between slots.

The modular platform for LLRF control system was realized as a multilayer PCB in mechanical standard EURO-6HE. It is designed to work with the communication VME-BUS [12]. Fig. 26 presents the board from the front side, while fig. 27 presents the same board from the rear side.

The front side has the following features. (fig. 26):

- slots for inserted communication modules VME BUS CONTROLLER and PC EMBEDDED CONTROLLER,
- module DATA & TIMING CONTROLLER implemented fully in the FPGA Cyclone matrix by Altera,
- block POWER SUPPLY consisting of a few nondependent resistive voltage stabilizers, providing the following voltages: *1.5V*, *1.8V*, *2.5V*, *3.3V*
- links of VME-BUS (J1 and J2), connector RS 232C (to the PC-EMBEDDED terminal), socket ETHERNET 100Mb and USB 1.0.
- slot for the D module.

At the rear side of the Platform PCB (fig. 27) there are situated slots for module A, B and C.

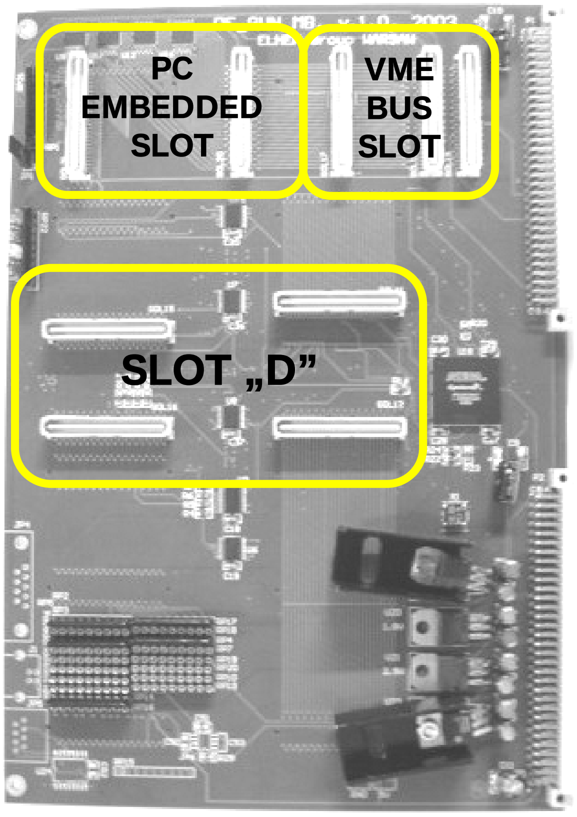


Fig. 26. Front side of the LLRF Modular Platform PCB

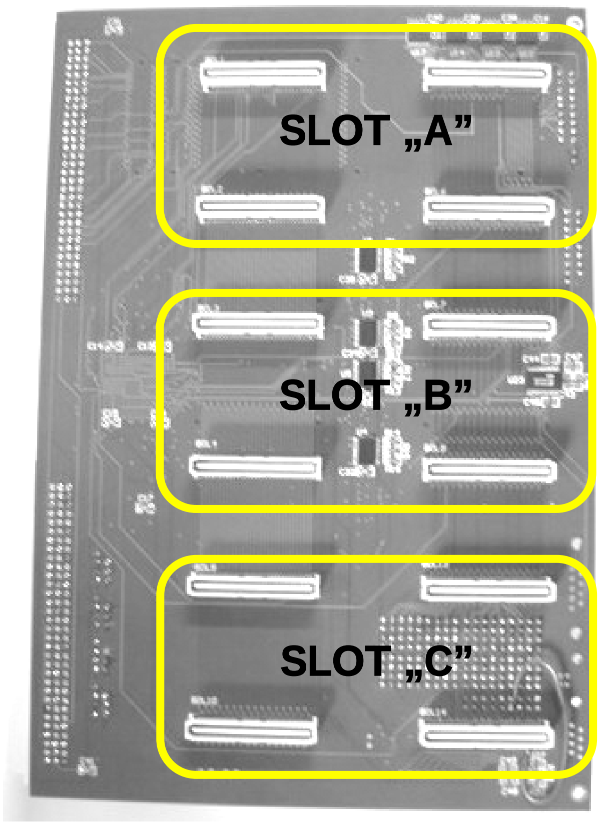


Fig. 27. Rear side of the LLRF Modular Platform PCB

B DSP EMBEDDED BOARD SPECIFICATION

The PCB of the *DSP EMBEDDED BOARD* module [11] was done using standard laminate FR4. The board consists of 16 layers: four negative power supply layers, and twelve signal layers. The power supply layers have three values of voltages for analog parts: (*-5V*; *+5V*; *+3.3V*), three voltages for the digital part of the board: (*+1.5V*; *3.3V*; *+5V*) and separately the analog chassis and digital chassis. The analog part was separated from the digital with the aid of separating coils of the inductance *45 μH*.

The discrete components, sockets, converters, amplifiers were positioned on the front side of the PCB, for the easy access to the signals with the measurement probe. There were used the high quality, high frequency (up to several GHz) subminiature sockets of MCX type. The small dimensions allow to position all necessary 12 connections on a single side of the board. The PCB possesses 4 additional digital sockets to distribute the synchronization signals of the LLRF control system. The board possesses 8 LEDs and an 8-bit port controlled directly from the FPGA chip, to provide monitoring of the FPGA chip itself.

The FPGA chip was positioned on the rear side of the board. The package is of the BGA fine pitch type. The pins in fine-pitch package are placed all over the chip surface creating a matrix. The package footprint is composed by small soldering pads connected with a matrix of vias. Thus, there is a direct access to all of the pins at the opposite side of the PCB.

The performed tests showed the ability of the debated system to control the resonant cavity at the total latency in the control, feed back loop below 1 μs. With this latency, the possible amplification in the control loop is over 100. Table 1 gathers technical parameters of the *DSP EMBEDDED BOARD module*. Table 2 has the signal description which are connected to the *Xilinx Virtex II 4000* chip [13].

Table 1.
Fundamental technical data of the 8-channel LLRF control module

Parameter		Value
Dimensions		162 mm x 142 mm; thickness 2,3 mm
Analog inputs	Number of channels	8
	Range	± 1V
	Maximum range of input voltage	± 3.8V
	Range of sampling	Minimum 30 MHz Maximum 105 MHz
	Analog bandwidth	270 MHz
Analog outputs	Number of channels	4
	Range of sampling	Depending on the work mode: from 3 MHz to maximum 160 MHz
	Range of output voltage	± 1V
Digital inputs	Number of channels	2
	Standard	LVTTL
	Input impedance	50 Ω
Digital outputs	Number of channels	2
	standard	LVTTL

Parameter		Value
Current consumption	Only the module DSP EMBEDDED BOARD	+5V ≈2.04A (+5V ≈4A together for DSP-board + Mother-board) +3.3V ≈0.32A +1.5V ≈0.02A - 12V ≈0.4A
Inbuilt clock circuit	Frequency	60 MHz

Table 2.
Description of signals connected to the Xilinx Virtex II 4000 chip

part	Signal description	Signal name	Xilinx Virtex II Pin number	remarks
ADC 1	Data bits	adc_data(1)(0)	U2	
		adc_data(1)(10)	F2	
		adc_data(1)(11)	F3	
		adc_data(1)(12)	G3	
		adc_data(1)(13)	H3	
		adc_data(1)(1)	T2	
		adc_data(1)(2)	P2	
		adc_data(1)(3)	N2	
		adc_data(1)(4)	M2	
		adc_data(1)(5)	L2	
		adc_data(1)(6)	K2	
		adc_data(1)(7)	J2	
		adc_data(1)(8)	H2	
		adc_data(1)(9)	G2	
	differential clock input LVPECL standard (**negative** part)	adc_clk_n(1)	P1	
	differential clock input LVPECL standard (**positive** part)	adc_clk_p(1)	N1	
	Data ready bit	adc_dry(1)	E2	
	Over flow bit	adc_ovr(1)	D2	
ADC 2	Data bits	adc_data(2)(0)	U3	
		adc_data(2)(10)	J4	
		adc_data(2)(11)	K4	
		adc_data(2)(12)	L4	
		adc_data(2)(13)	M4	
		adc_data(2)(1)	T3	
		adc_data(2)(2)	R3	
		adc_data(2)(3)	P3	
		adc_data(2)(4)	N3	
		adc_data(2)(5)	M3	
		adc_data(2)(6)	L3	
		adc_data(2)(7)	J3	
		adc_data(2)(8)	F4	
		adc_data(2)(9)	H4	
	differential clock input LVPECL standard (**negative** part)	adc_clk_n(2)	A4	
	differential clock input LVPECL standard (**positive** part)	adc_clk_p(2)	A5	
	Data ready bit	adc_dry(2)	B5	
	Over flow bit	adc_ovr(2)	C8	
ADC 3	Data bits	adc_data(3)(0)	B8	
		adc_data(3)(10)	B16	

part	Signal description	Signal name	Xilinx Virtex II Pin number	remarks
		adc_data(3)(11)	A17	
		adc_data(3)(12)	B17	
		adc_data(3)(13)	C16	
		adc_data(3)(1)	B9	
		adc_data(3)(2)	B10	
		adc_data(3)(3)	B11	
		adc_data(3)(4)	B12	
		adc_data(3)(5)	B13	
		adc_data(3)(6)	A13	
		adc_data(3)(7)	B14	
		adc_data(3)(8)	A14	
		adc_data(3)(9)	A15	
	differential clock input LVPECL standard (**negative** part)	adc_clk_n(3)	A11	
	differential clock input LVPECL standard (**positive** part)	adc_clk_p(3)	A12	
	Data ready bit	adc_dry(3)	C14	
	Over flow bit	adc_ovr(3)	D10	
ADC 4	Data bits	adc_data(4)(0)	C6	
		adc_data(4)(10)	D16	
		adc_data(4)(11)	D15	
		adc_data(4)(12)	D14	
		adc_data(4)(13)	D13	
		adc_data(4)(1)	C11	
		adc_data(4)(2)	C12	
		adc_data(4)(3)	C13	
		adc_data(4)(4)	C15	
		adc_data(4)(5)	A18	
		adc_data(4)(6)	B18	
		adc_data(4)(7)	C18	
		adc_data(4)(8)	D18	
		adc_data(4)(9)	D17	
	differential clock input LVPECL standard (**negative** part)	adc_clk_n(4)	A21	
	differential clock input LVPECL standard (**positive** part)	adc_clk_p(4)	A22	
	Data ready bit	adc_dry(4)	C27	
	Over flow bit	adc_ovr(4)	D27	
ADC 5	Data bits	adc_data(5)(0)	D19	
		adc_data(5)(10)	C22	
		adc_data(5)(11)	B22	
		adc_data(5)(12)	D23	
		adc_data(5)(13)	C23	
		adc_data(5)(1)	C19	
		adc_data(5)(2)	B19	
		adc_data(5)(3)	D20	
		adc_data(5)(4)	C20	
		adc_data(5)(5)	A20	
		adc_data(5)(6)	D21	
		adc_data(5)(7)	C21	
		adc_data(5)(8)	B21	
		adc_data(5)(9)	D22	
	differential clock input LVPECL standard (**negative** part)	adc_clk_n(5)	A30	
	differential clock input LVPECL standard (**positive** part)	adc_clk_p(5)	A31	

part	Signal description	Signal name	Xilinx Virtex II Pin number	remarks
	Data ready bit	adc_dry(5)	C33	
	Over flow bit	adc_ovr(5)	E33	
ADC 6	Data bits	adc_data(6)(0)	B23	
		adc_data(6)(10)	B26	
		adc_data(6)(11)	B27	
		adc_data(6)(12)	B28	
		adc_data(6)(13)	A28	
		adc_data(6)(1)	A23	
		adc_data(6)(2)	D24	
		adc_data(6)(3)	C24	
		adc_data(6)(4)	B24	
		adc_data(6)(5)	A24	
		adc_data(6)(6)	D25	
		adc_data(6)(7)	B25	
		adc_data(6)(8)	D26	
		adc_data(6)(9)	C26	
	differential clock input LVPECL standard (**negative** part)	adc_clk_n(6)	E34	
	differential clock input LVPECL standard (**positive** part)	adc_clk_p(6)	D34	
	Data ready bit	adc_dry(6)	C28	
	Over flow bit	adc_ovr(6)	B32	
ADC 7	Data bits	adc_data(7)(0)	A29	
		adc_data(7)(10)	G32	
		adc_data(7)(11)	F33	
		adc_data(7)(12)	H32	
		adc_data(7)(13)	G33	
		adc_data(7)(1)	B29	
		adc_data(7)(2)	C29	
		adc_data(7)(3)	D29	
		adc_data(7)(4)	B30	
		adc_data(7)(5)	B31	
		adc_data(7)(6)	F32	
		adc_data(7)(7)	D32	
		adc_data(7)(8)	D33	
		adc_data(7)(9)	E32	
	differential clock input LVPECL standard (**negative** part)	adc_clk_n(7)	G34	
	differential clock input LVPECL standard (**positive** part)	adc_clk_p(7)	F34	
	Data ready bit	adc_dry(7)	U34	
	Over flow bit	adc_ovr(7)	L32	
ADC 8	Data bits	adc_data(8)(0)	H33	
		adc_data(8)(10)	T33	
		adc_data(8)(11)	U33	
		adc_data(8)(12)	R32	
		adc_data(8)(13)	P32	
		adc_data(8)(1)	J33	
		adc_data(8)(2)	K33	
		adc_data(8)(3)	L33	
		adc_data(8)(4)	M33	
		adc_data(8)(5)	N33	
		adc_data(8)(6)	P33	
		adc_data(8)(7)	N34	
		adc_data(8)(8)	P34	
		adc_data(8)(9)	R34	

part	Signal description	Signal name	Xilinx Virtex II Pin number	remarks
	differential clock input LVPECL standard (**negative** part)	adc_clk_n(8)	M34	
	differential clock input LVPECL standard (**positive** part)	adc_clk_p(8)	L34	
	Data ready bit	adc_dry(8)	J34	
	Over flow bit	adc_ovr(8)	N32	
DAC 1	Data bits	dac_data(1)(0)	N31	
		dac_data(1)(10)	W32	
		dac_data(1)(11)	K31	
		dac_data(1)(12)	J31	
		dac_data(1)(13)	H31	
		dac_data(1)(1)	M31	
		dac_data(1)(2)	L31	
		dac_data(1)(3)	AB31	
		dac_data(1)(4)	AA31	
		dac_data(1)(5)	Y31	
		dac_data(1)(6)	W31	
		dac_data(1)(7)	V31	
		dac_data(1)(8)	U31	
		dac_data(1)(9)	T31	
	differential clock input LVPECL standard (**negative** part)	dac_clk_n(1)	AC34	
	differential clock input LVPECL standard (**positive** part)	dac_clk_p(1)	AD34	
	DIV0	dac_div(1)(0)	R31	See datasheet for description
	DIV1	dac_div(1)(1)	P31	
	MOD0	dac_mode(1)(0)	V32	
	MOD1	dac_mode(1)(1)	V33	
	PLLLOCK	dac_pll_lock(1)	V34	
DAC 2	Data bits	dac_data(2)(0)	AM4	
		dac_data(2)(10)	AM15	
		dac_data(2)(11)	AP17	
		dac_data(2)(12)	AN17	
		dac_data(2)(13)	AM16	
		dac_data(2)(1)	AL5	
		dac_data(2)(2)	AM6	
		dac_data(2)(3)	AM7	
		dac_data(2)(4)	AM8	
		dac_data(2)(5)	AM9	
		dac_data(2)(6)	AM11	
		dac_data(2)(7)	AM12	
		dac_data(2)(8)	AM13	
		dac_data(2)(9)	AM14	
	differential clock input LVPECL standard (**negative** part)	dac_clk_n(2)	AP31	
	differential clock input LVPECL standard (**positive** part)	dac_clk_p(2)	AP30	
	DIV0	dac_div(2)(0)	AM17	See datasheet for description
	DIV1	dac_div(2)(1)	AL17	
	MOD0	dac_mode(2)(0)	AL6	
	MOD1	dac_mode(2)(1)	AL8	
	PLLLOCK	dac_pll_lock(2)	AL16	
DAC 3	Data bits	dac_data(3)(0)	AA1	
		dac_data(3)(10)	AC3	
		dac_data(3)(11)	AD3	
		dac_data(3)(12)	AF3	

part	Signal description	Signal name	Xilinx Virtex II Pin number	remarks
		dac_data(3)(13)	AG3	
		dac_data(3)(1)	V2	
		dac_data(3)(2)	W2	
		dac_data(3)(3)	AA2	
		dac_data(3)(4)	AC2	
		dac_data(3)(5)	AH1	
		dac_data(3)(6)	W3	
		dac_data(3)(7)	Y3	
		dac_data(3)(8)	AA3	
		dac_data(3)(9)	AB3	
	differential clock input LVPECL standard (**negative** part)	dac_clk_n(3)	AK34	
	differential clock input LVPECL standard (**positive** part)	dac_clk_p(3)	AL34	
	DIV0	dac_div(3)(0)	AL3	See datasheet for description
	DIV1	dac_div(3)(1)	AK3	
	MOD0	dac_mode(3)(0)	AJ3	
	MOD1	dac_mode(3)(1)	AH3	
	PLLLOCK	dac_pll_lock(3)	AJ4	
DAC 4	Data bits	dac_data(4)(0)	AL18	
		dac_data(4)(10)	AL27	
		dac_data(4)(11)	AL29	
		dac_data(4)(12)	AL30	
		dac_data(4)(13)	AD31	
		dac_data(4)(1)	AL19	
		dac_data(4)(2)	AL20	
		dac_data(4)(3)	AL21	
		dac_data(4)(4)	AL22	
		dac_data(4)(5)	AL23	
		dac_data(4)(6)	AL24	
		dac_data(4)(7)	AN26	
		dac_data(4)(8)	AL25	
		dac_data(4)(9)	AL26	
	differential clock input LVPECL standard (**negative** part)	dac_clk_n(4)	AP24	
	differential clock input LVPECL standard (**positive** part)	dac_clk_p(4)	AP23	
	DIV0	dac_div(4)(0)	AG31	See datasheet for description
	DIV1	dac_div(4)(1)	AF31	
	MOD0	dac_mode(4)(0)	F31	
	MOD1	dac_mode(4)(1)	E31	
	PLLLOCK	dac_pll_lock(4)	AE31	
		ii_addr(0)	AP18	
		ii_addr(10)	AN25	
		ii_addr(11)	AP26	
		ii_addr(12)	AN27	
		ii_addr(13)	AP28	
		ii_addr(14)	AN28	
		ii_addr(15)	AP29	
		ii_addr(16)	AN29	
		ii_addr(17)	AN30	
		ii_addr(18)	AN31	
		ii_addr(19)	AN32	
		ii_addr(1)	AN18	
		ii_addr(20)	AM19	
		ii_addr(21)	AM20	

part	Signal description	Signal name	Xilinx Virtex II Pin number	remarks
		ii_addr(22)	AM21	
		ii_addr(23)	AM22	
		ii_addr(24)	AM23	
		ii_addr(25)	AM24	
		ii_addr(26)	AM26	
		ii_addr(27)	AM27	
		ii_addr(28)	AM28	
		ii_addr(29)	AM29	
		ii_addr(2)	AN19	
		ii_addr(30)	AM31	
		ii_addr(31)	AM33	
		ii_addr(3)	AP20	
		ii_addr(4)	AN21	
		ii_addr(5)	AP21	
		ii_addr(6)	AP22	
		ii_addr(7)	AN22	
		ii_addr(8)	AN23	
		ii_addr(9)	AN24	
		ii_data(0)	AL32	
		ii_data(10)	AG32	
		ii_data(11)	AG33	
		ii_data(12)	AF32	
		ii_data(13)	AF33	
		ii_data(14)	AF34	
		ii_data(15)	AE33	
		ii_data(16)	AD32	
		ii_data(17)	AD33	
		ii_data(18)	AC32	
		ii_data(19)	AC33	
		ii_data(1)	AL33	
		ii_data(20)	AB32	
		ii_data(21)	AB33	
		ii_data(22)	AB34	
		ii_data(23)	AA32	
		ii_data(24)	AA33	
		ii_data(25)	AA34	
		ii_data(26)	Y32	
		ii_data(27)	Y34	
		ii_data(28)	W33	
		ii_data(29)	AK31	
		ii_data(2)	AK32	
		ii_data(30)	AJ31	
		ii_data(31)	AC31	
		ii_data(3)	AK33	
		ii_data(4)	AJ32	
		ii_data(5)	AJ33	
		ii_data(6)	AJ34	
		ii_data(7)	AH32	
		ii_data(8)	AH33	
		ii_data(9)	AH34	
		ii_irqN(0)	M1	
		ii_irqN(1)	L1	
		ii_irqN(2)	J1	
		ii_irqN(3)	G1	
		ii_irqN(4)	F1	
		ii_irqN(5)	C2	

part	Signal description	Signal name	Xilinx Virtex II Pin number	remarks
		ii_irqN(6)	B4	
		ii_irqN(7)	A6	
		ii_operN	M32	
		ii_resetN	A26	
		ii_strobeN	E17	
		ii_writeN	D9	
		bus_d(0)	AN16	
		bus_d(10)	AN10	
		bus_d(11)	AN9	
		bus_d(12)	AP9	
		bus_d(13)	AN8	
		bus_d(14)	AN7	
		bus_d(15)	AP7	
		bus_d(16)	AN6	
		bus_d(17)	AP6	
		bus_d(18)	AN5	
		bus_d(19)	AP5	
		bus_d(1)	AP15	
		bus_d(20)	AN4	
		bus_d(21)	AP4	
		bus_d(22)	AN3	
		bus_d(23)	AM2	
		bus_d(24)	AL2	
		bus_d(25)	AL1	
		bus_d(26)	AK2	
		bus_d(27)	AK1	
		bus_d(28)	AJ2	
		bus_d(29)	AJ1	
		bus_d(2)	AN14	
		bus_d(30)	AH2	
		bus_d(31)	AG2	
		bus_d(3)	AP14	
		bus_d(4)	AN13	
		bus_d(5)	AP13	
		bus_d(6)	AN12	
		bus_d(7)	AP12	
		bus_d(8)	AN11	
		bus_d(9)	AP11	
		clk	D12	
		clkout	D11	
		trgout	B3	
		mclk(0)	B6	
		mclk(1)	T32	
		mclk(2)	J32	
		mclk(3)	B7	
		mtrg(0)	A7	
		mtrg(1)	D8	
		mtrg(2)	D6	
		mtrg(3)	A9	
		ii_ackN	C9	
Local clock		lclk	E19	
Auxiliary LED diodes		led(0)	AB1	
		led(1)	AB2	
		led(2)	AC1	
		led(3)	AD1	

part	Signal description	Signal name	Xilinx Virtex II Pin number	remarks
		led(4)	AD2	
		led(5)	AE2	
		led(6)	AF1	
		led(7)	AF2	
		conf_led	AL9	Not mounted
Digital In/out	inputs	sigin(0)	Y1	
		sigin(1)	V1	
	outputs	sigout(0)	U1	
		sigout(1)	R1	
8 bits digital auxiliary port		sigtest(0)	C7	
		sigtest(1)	D3	
		sigtest(2)	E3	
		sigtest(3)	E4	
		sigtest(4)	D1	
		sigtest(5)	E1	
		sigtest(6)	N4	
		sigtest(7)	P4	

C Exemplary scope pictures of SIMCON system outputs

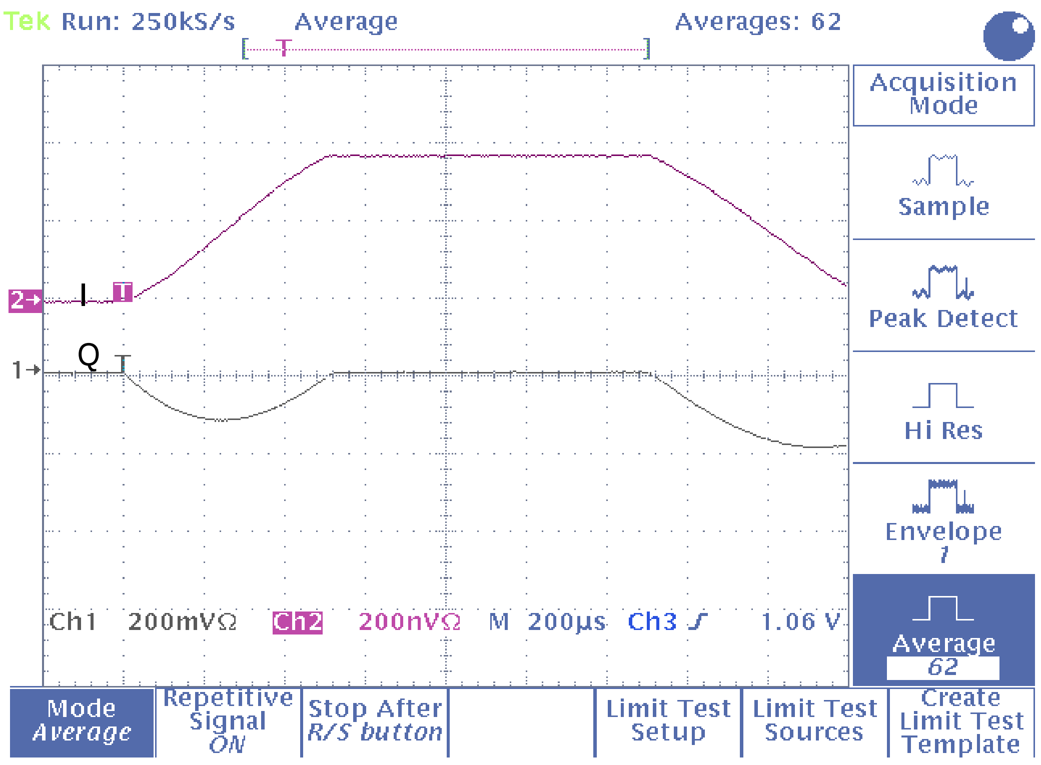


Fig. 28. I and Q outputs of cavity simulator driven by feedback and supported by feedforward

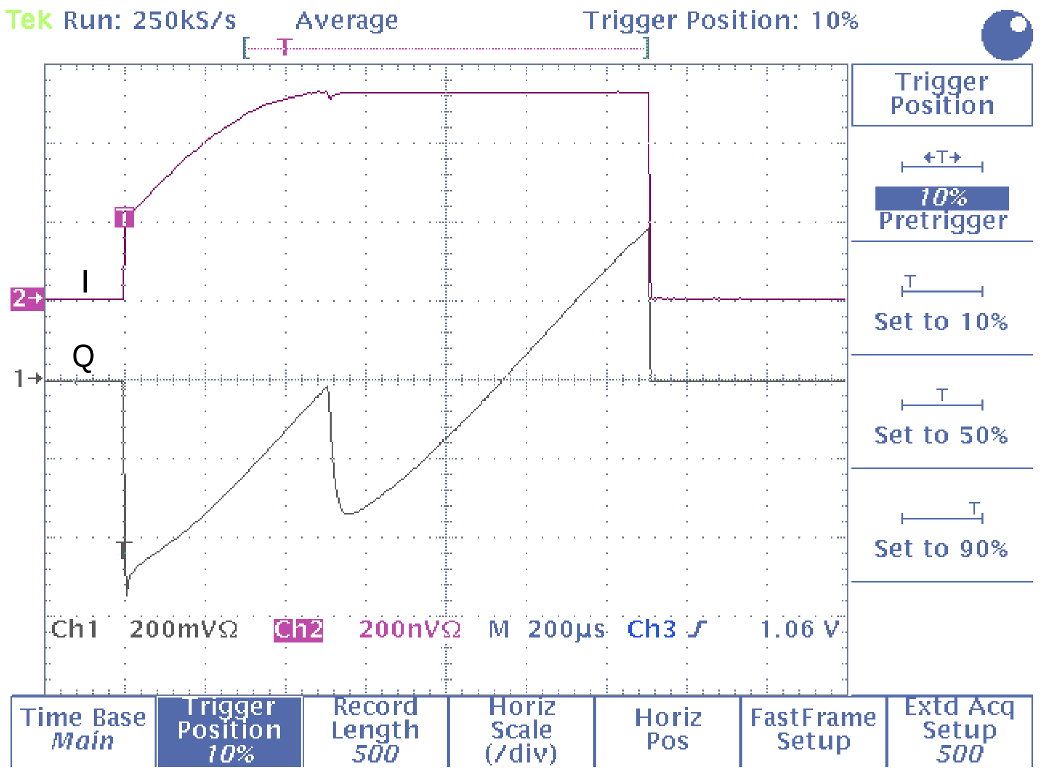


Fig. 29. I and Q outputs of CAVITIES CONTROLLER operated in feedback and feedforward mode

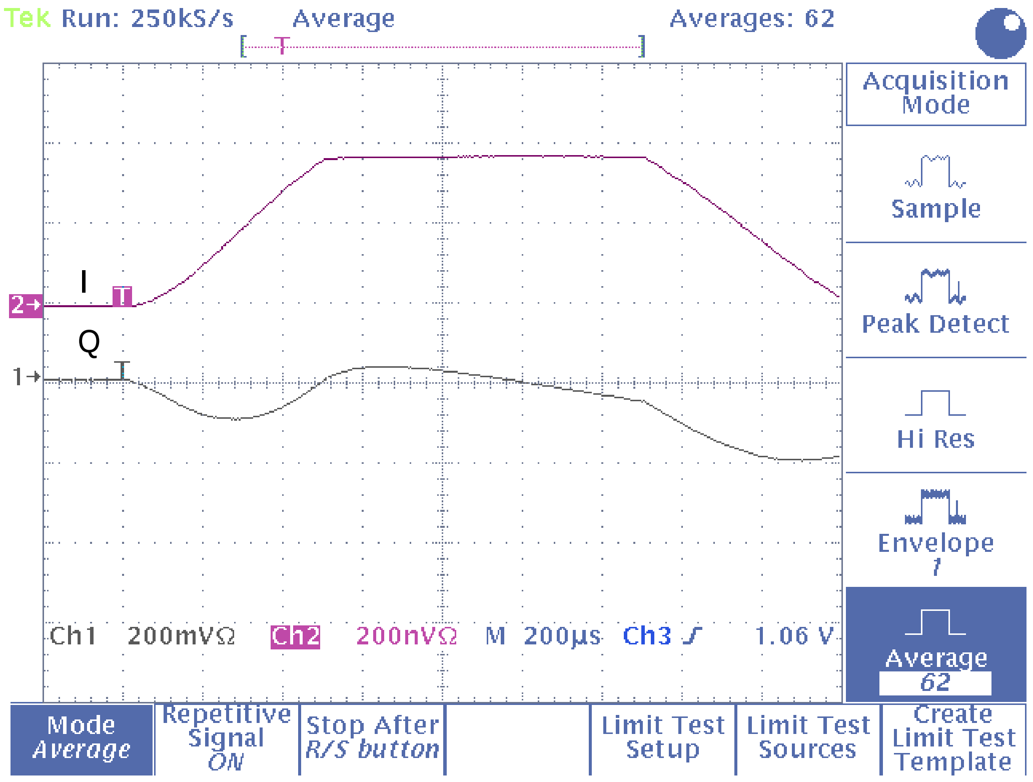


Fig. 30. I and Q outputs of cavity simulator driven by low gain feedback.

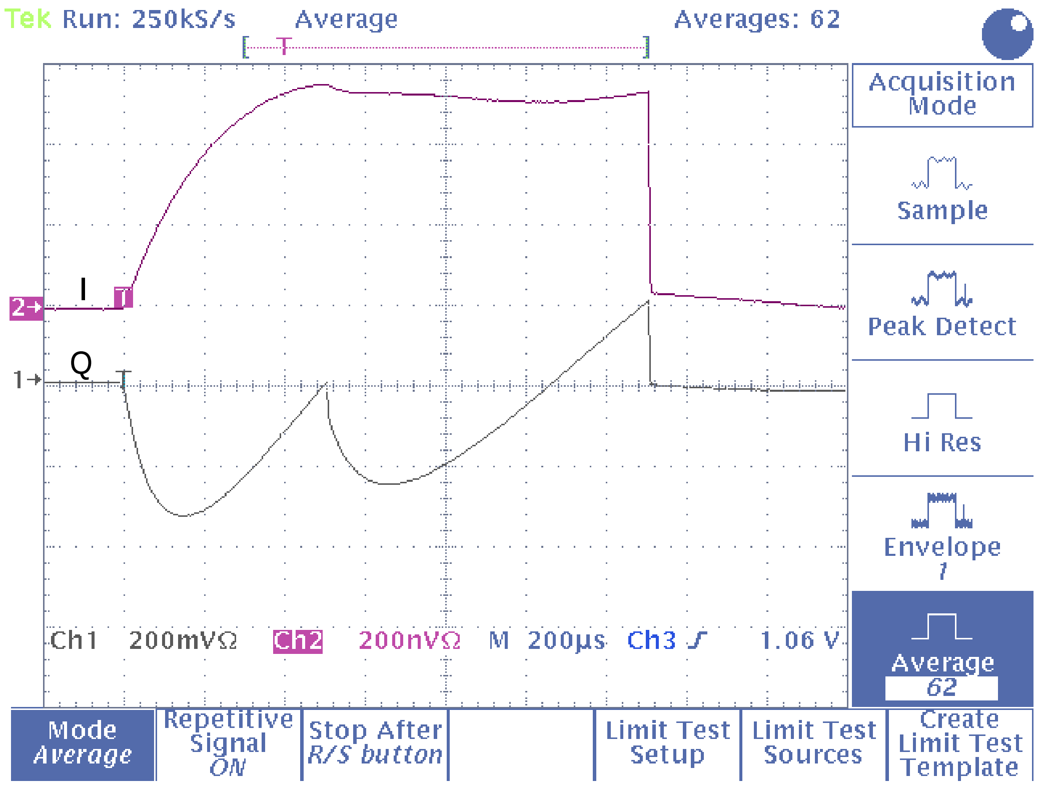


Fig. 31. I and Q outputs of CAVITIES CONTROLLER operated in low gain feedback mode.

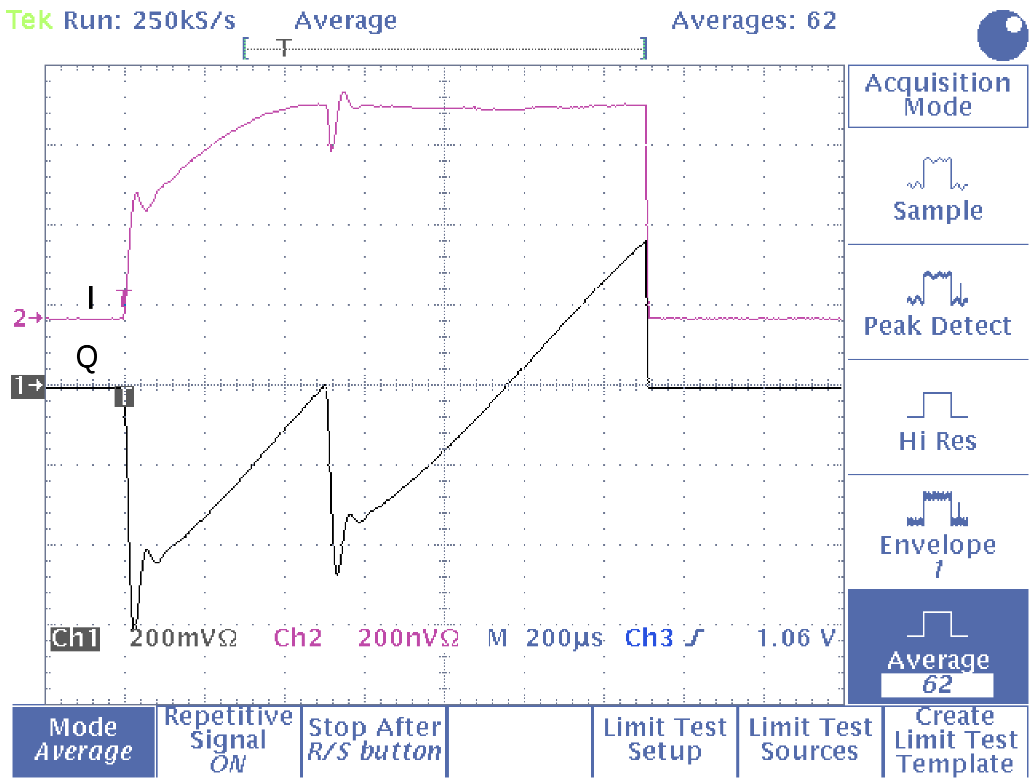


Fig. 32. I and Q outputs of CAVITIES CONTROLLER operated in delay loop condition.

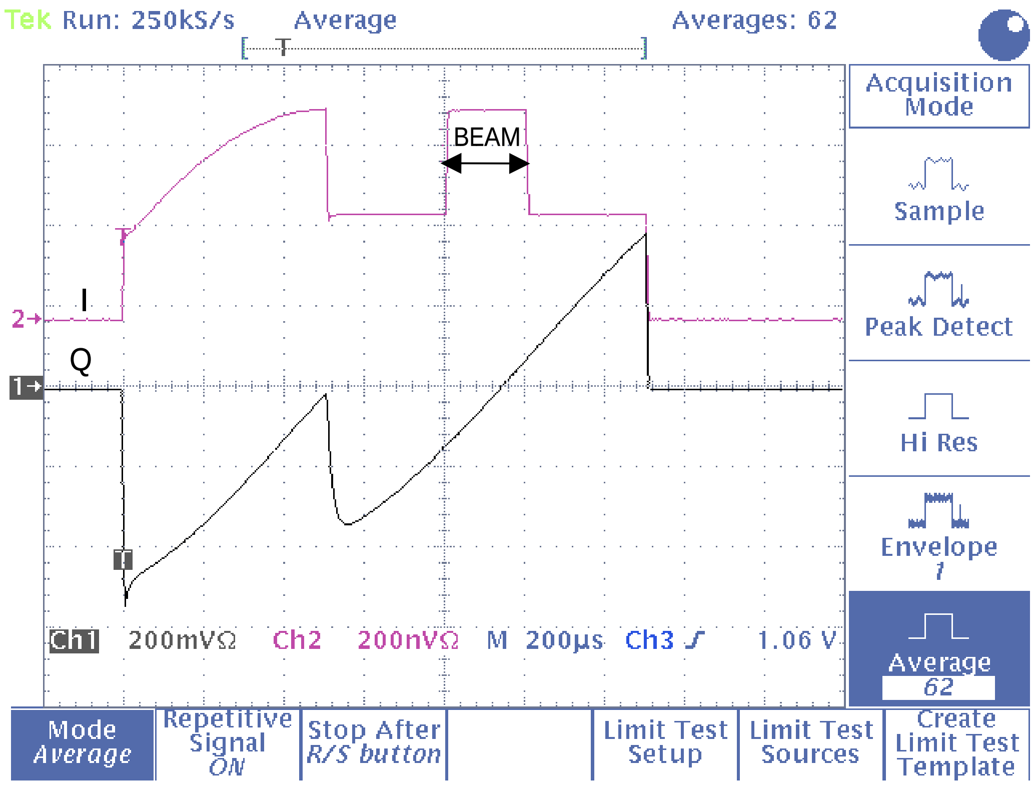


Fig. 33. I and Q outputs of CAVITIES CONTROLLER with the beam switched on during the flattop.

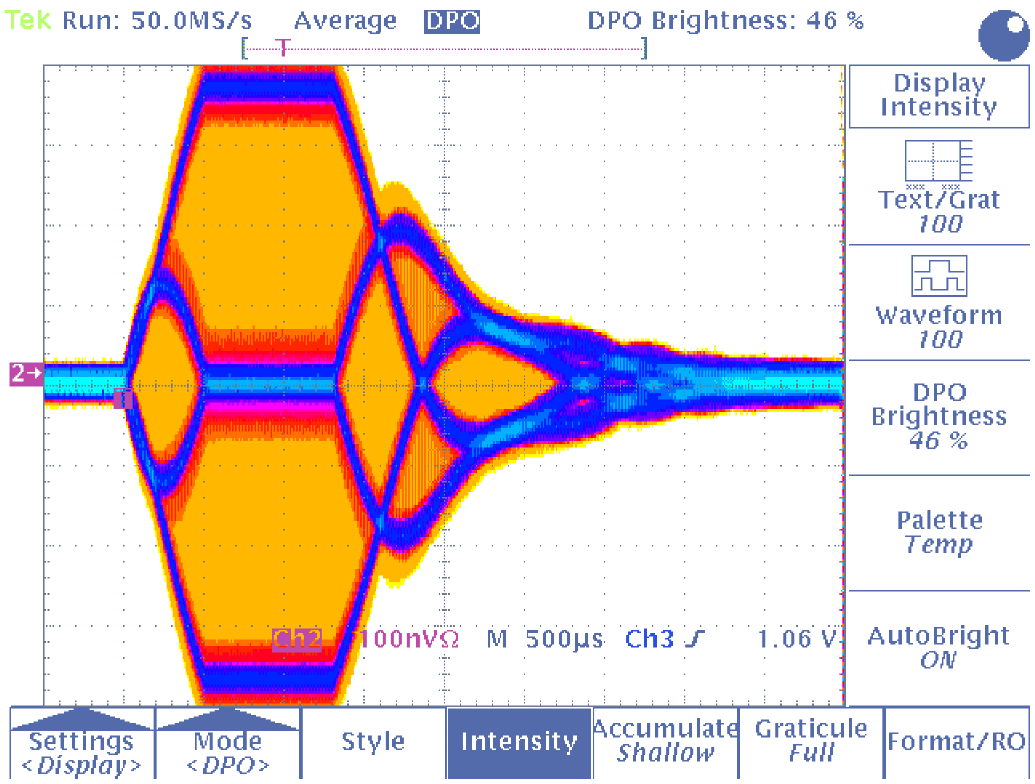


Fig. 34. Cavity simulator output for IF modulated signal (presented in DPO scope mode).

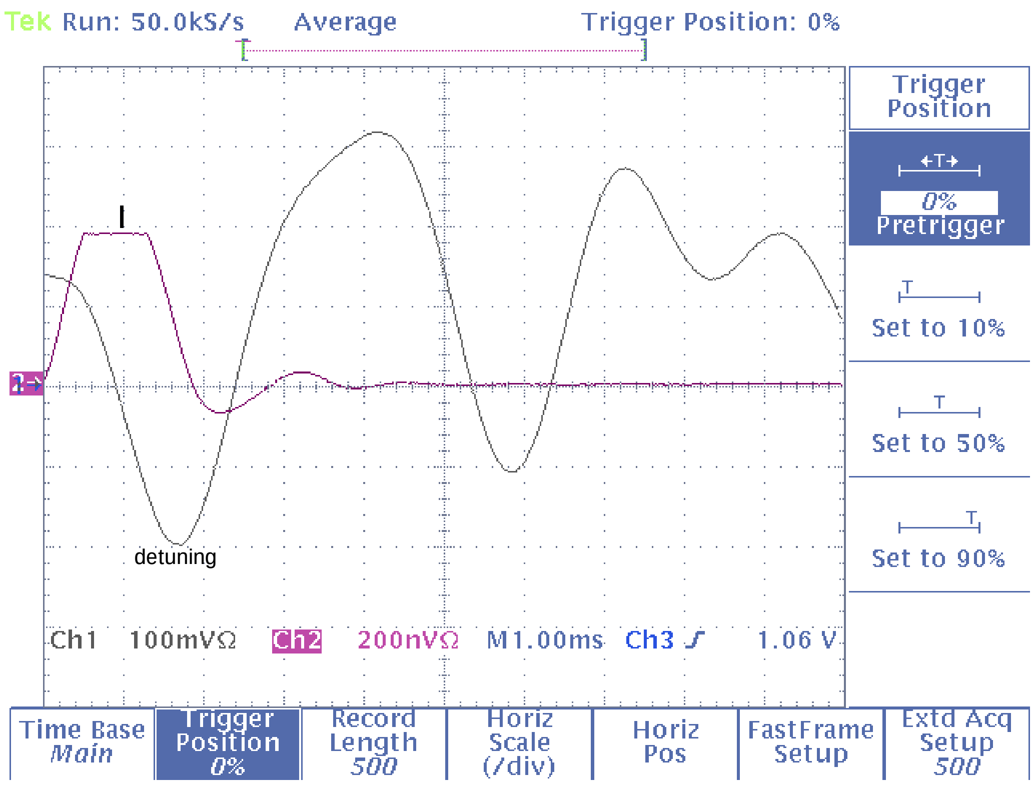


Fig. 35. Cavity simulator output for detuning signal related to I component of envelope.

D Exemplary results of CHECHIA real-time control

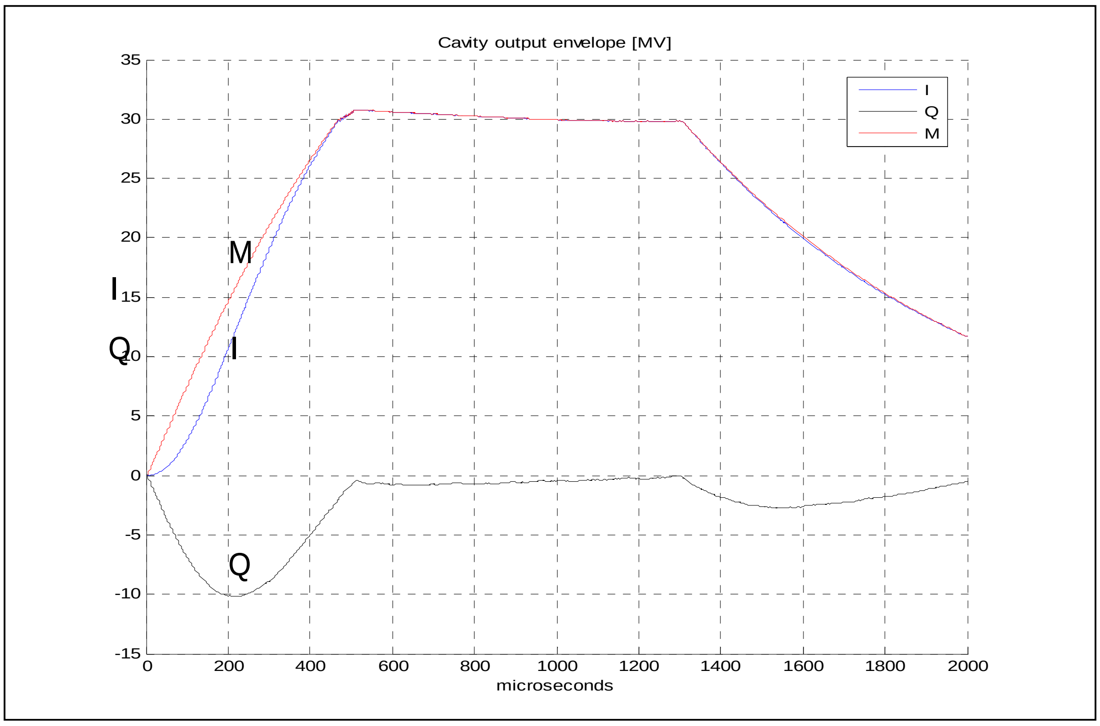


Fig. 36. Feed-forward cavity driving: selected readout of output envelope for 30 MV flattop level.

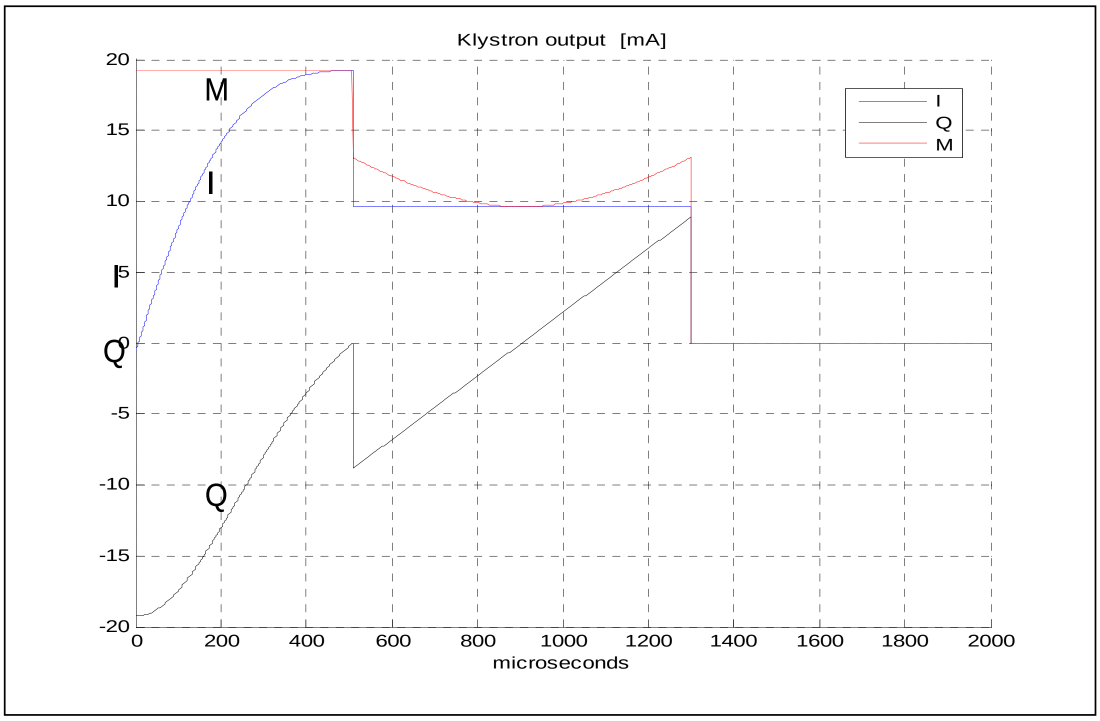


Fig. 37. Feed-forward cavity driving: selected readout of input envelope for 30 MV flattop level.

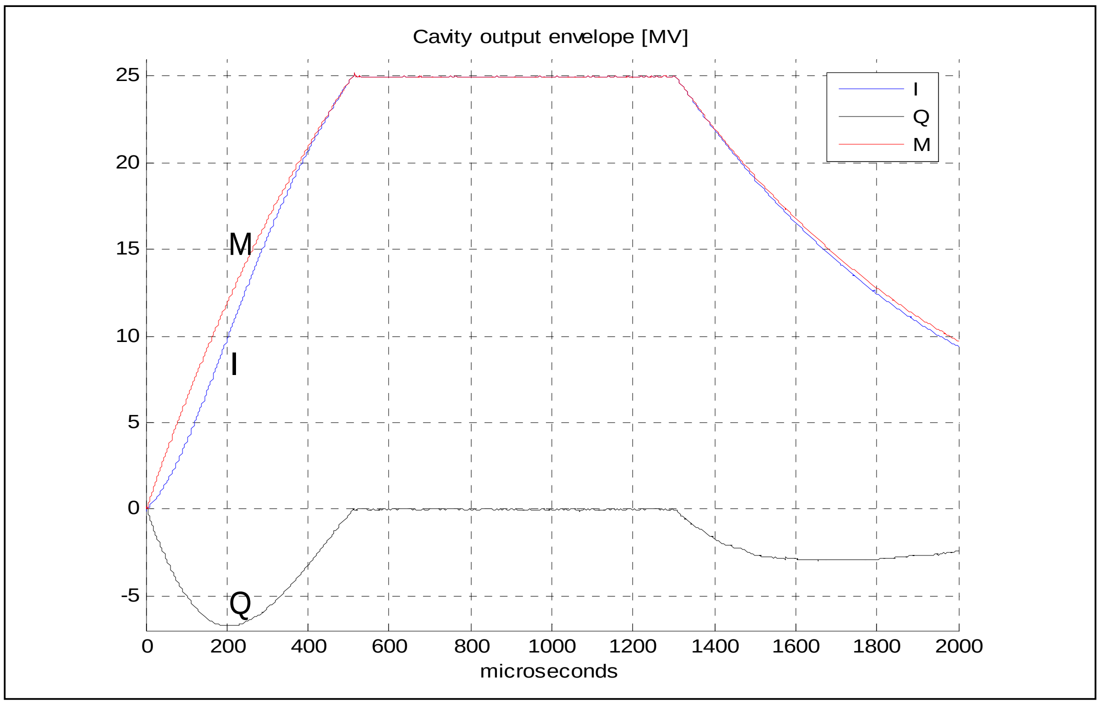


Fig. 38. Feed-forward with feedback cavity driving (gain = 100): selected readout of output envelope for 25 MV flattop level.

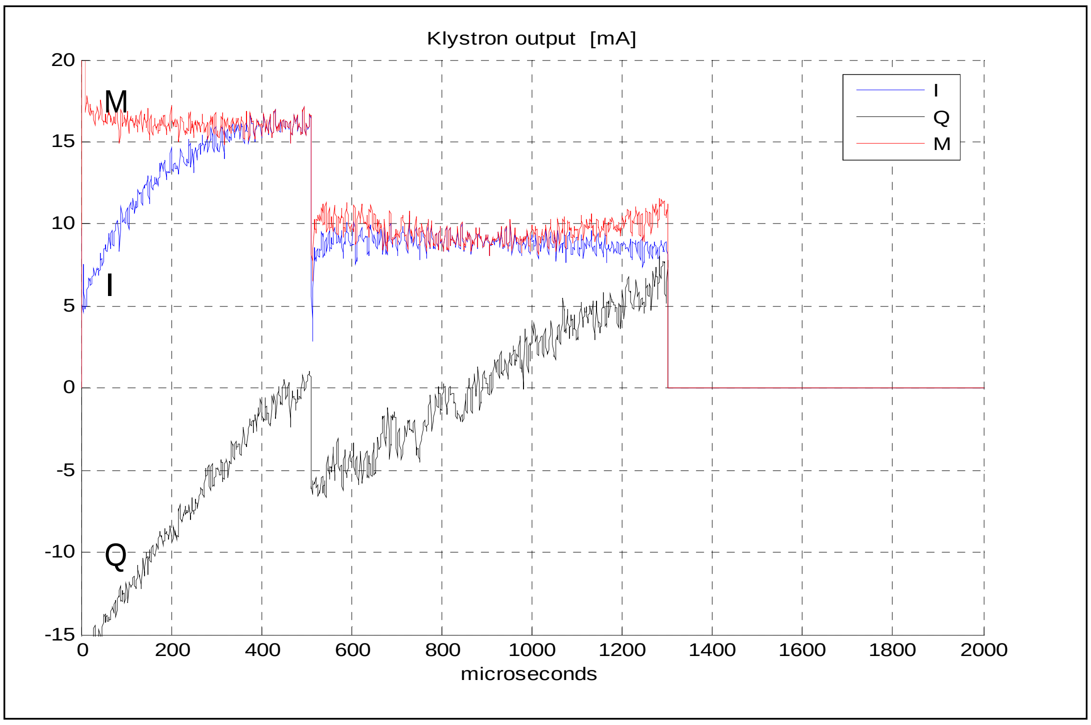


Fig. 39. Feed-forward with feedback cavity driving (gain = 100): selected readout of input envelope for 25 MV

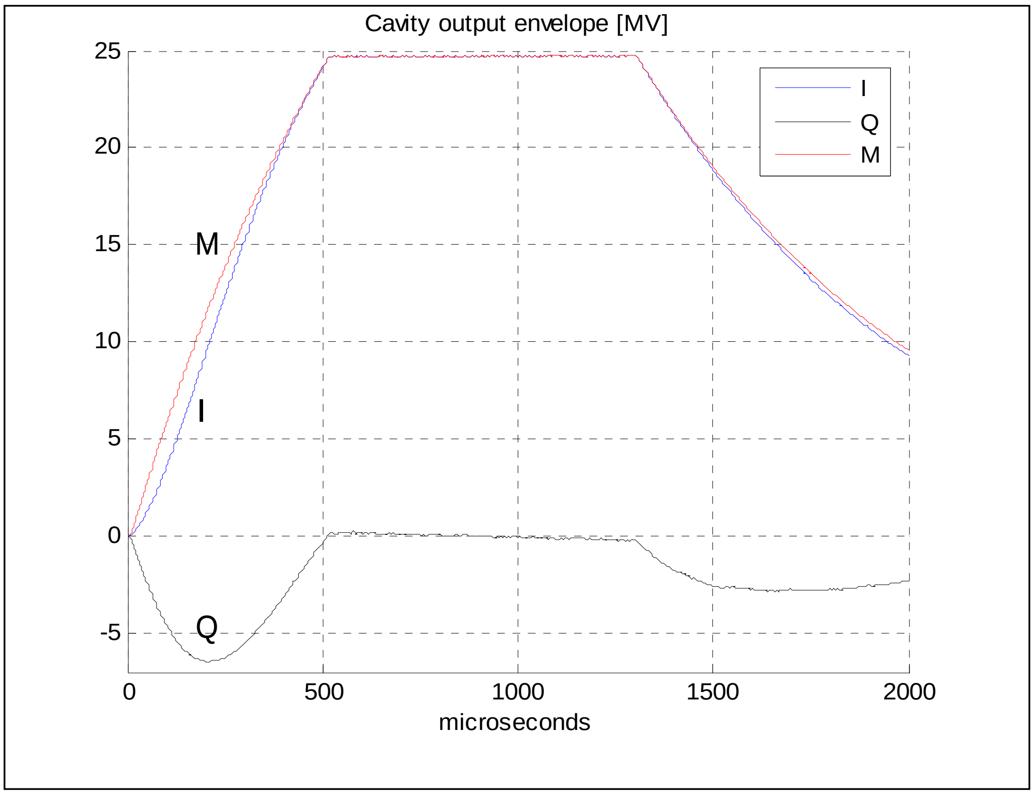


Fig. 40. Feedback cavity driving (gain = 100): selected readout of output envelope for 25 MV flattop level.

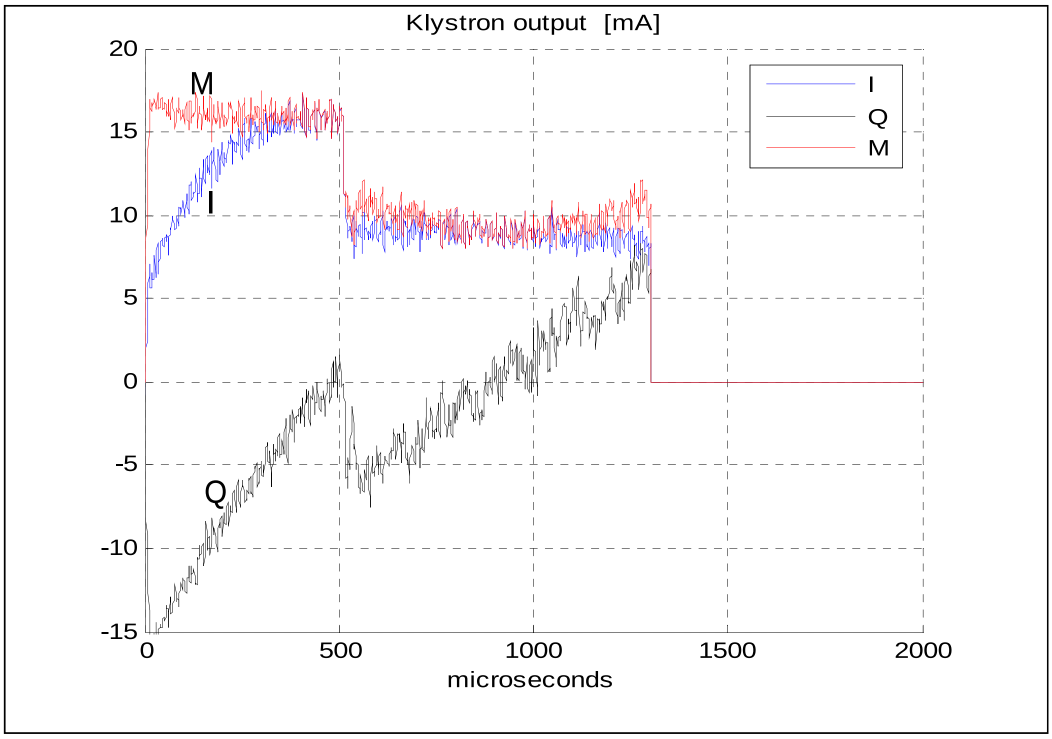


Fig. 41. Feedback cavity driving (gain = 100): selected readout of input envelope for 25 MV flattop level.

E Exemplary pictures of ACC1 module real-time control

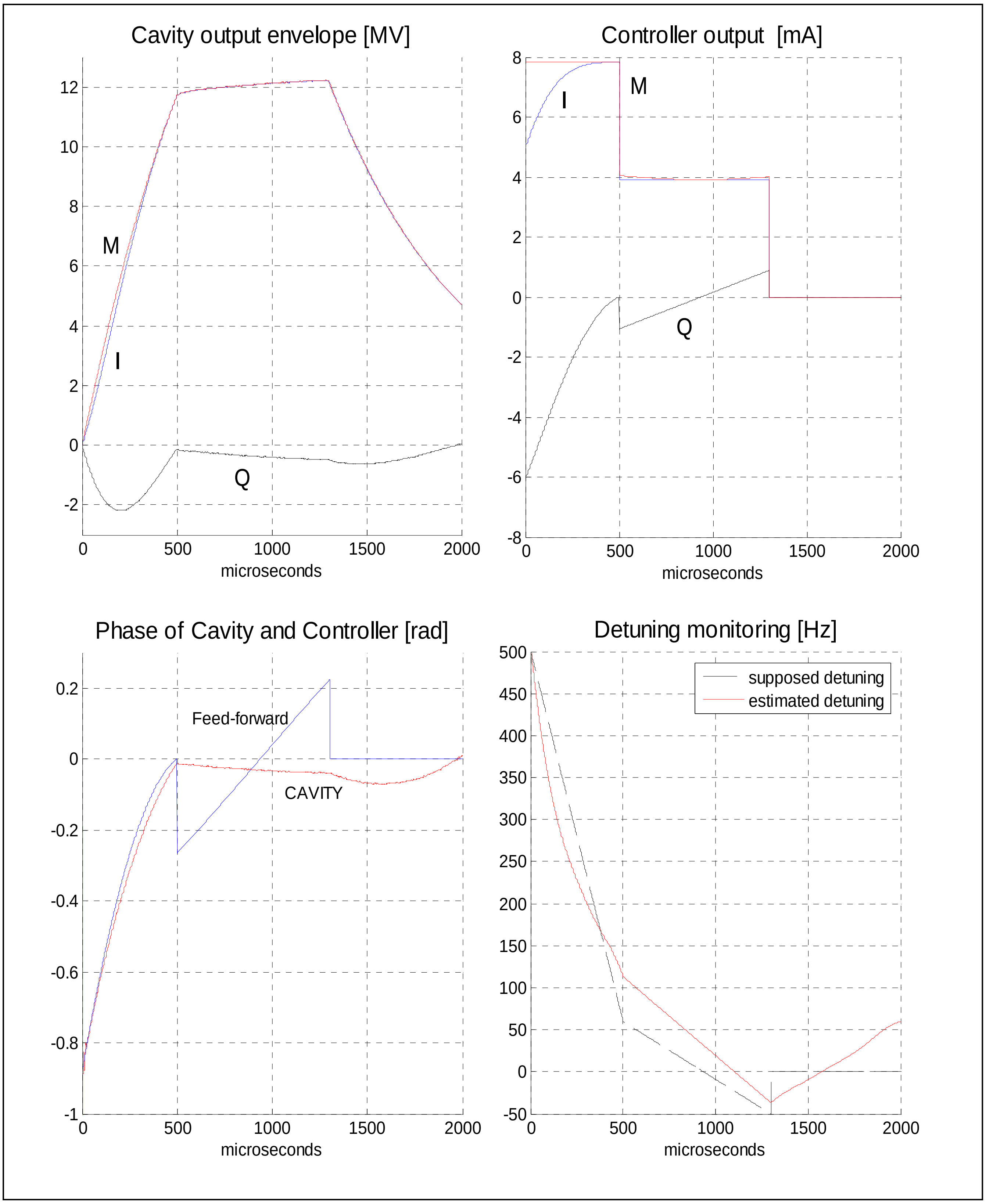


Fig. 42. Feed-forward single cavity driving for ACC1 module

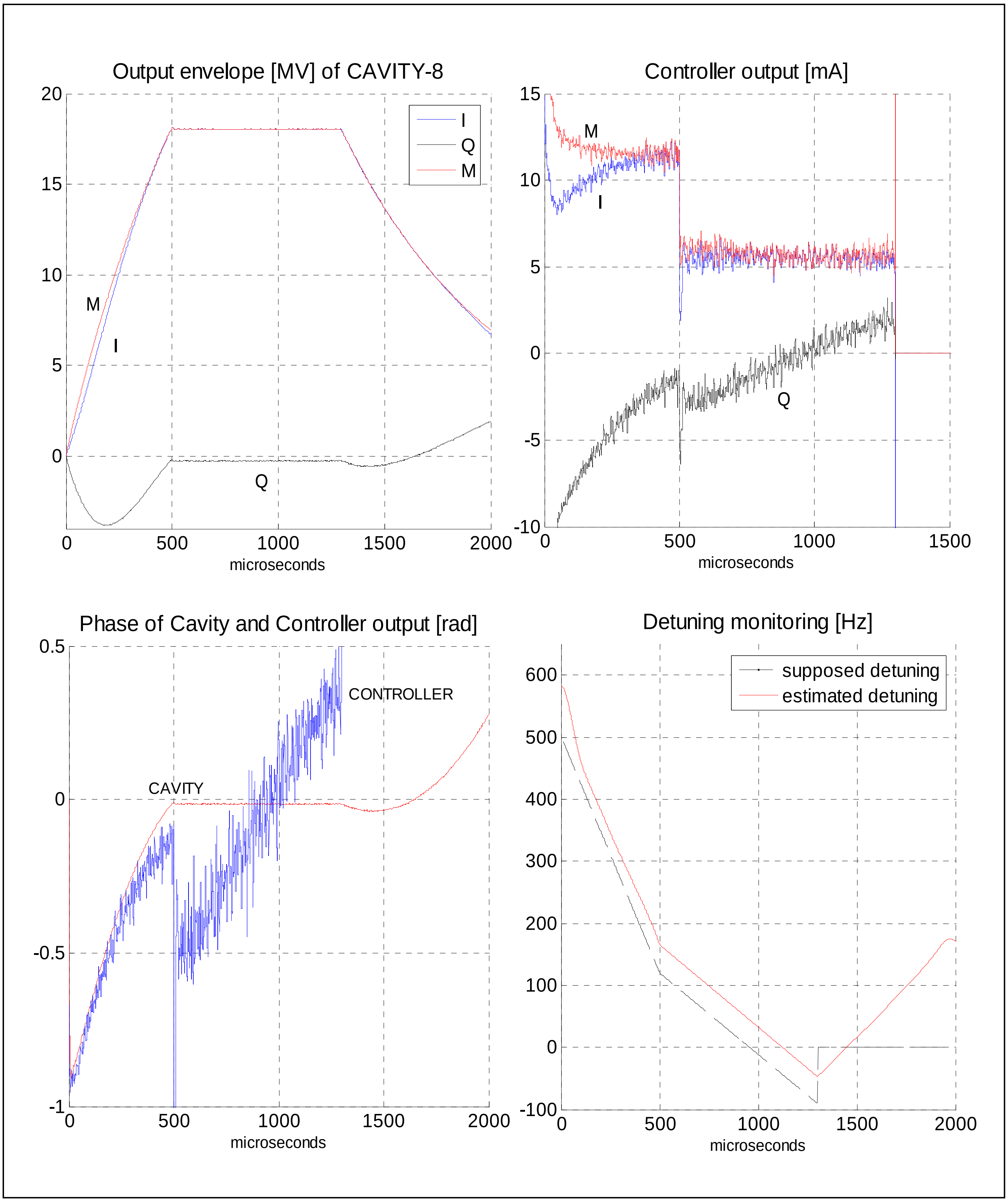


Fig. 43. Feed-back and feed-forward single cavity driving for ACC1 module

References

1. P.Rutkowski, R.Romaniuk, K.T.Pozniak, T.Jezynski, P.Pucyk, M.Pietrusinski, S.Simrock: “FPGA Based TESLA Cavity SIMCON DOOCS Server Design, Implementation and Application”, TESLA Technical Note, 2003-32
2. K.T.Pozniak, R.Romaniuk, K.Kierzkowski: “Parameterized Control Layer of FPGA Based CAVITIES CONTROLLER and Simulator for TESLA Test Facility”, TESLA Technical Note, 2003-30
3. K.T.Pozniak, T.Czarski, R.Romaniuk: “Functional Analysis of DSP Blocks in FPGA Chips for Application in TESLA LLRF System”, TESLA Technical Note, 2003-29
4. T.Czarski, K.T.Pozniak, R.Romaniuk, S.Simrock: “TESLA Cavity Modeling and Digital Implementation with FPGA Technology Solution For Control System Purpose”, TESLA Technical Note, 2003-28
5. T.Czarski, R.S.Romaniuk, K.T.Pozniak S.Simrock “Cavity Control System Essential Modeling For TESLA Linear Accelerator”, TESLA Technical Note, 2003-08
6. T.Czarski, R.S.Romaniuk, K.T. Pozniak “Cavity Control System, Models Simulations For TESLA Linear Accelerator”, TESLA Technical Note, 2003-09
7. K.T.Poźniak, M.Bartoszek M.Pietrusiński: “Internal Interface for RPC Muon Trigger electronics at CMS experiment”, Proceedings of SPIE, Photonics Applications II In Astronomy, Communications, Industry and High Energy Physics Experiments, Vol. 5484, 2004
8. K.T.Pozniak, R.S.Romaniuk, K.Kierzkowski, “Modular & Reconfigurable Common PCB-Platform of FPGA Based LLRF Control System for TESLA Test Facility”, TESLA Technical Note, 2005-4
9. K.T.Pozniak, R.S.Romaniuk, W.Jalmuzna, K.Olowski, K.Perkuszewski, J.Zielinski, K.Kierzkowski: “FPGA Based, Full-Duplex, Multi-Channel, Multi-Gigabit, Optical, Synchronous Data Transceiver for TESLA Technology LLRF Control System”, TESLA Technical Note, 2004-07
10. P.Roszkowski, W.M.Zabolotny, K.Pozniak, R.S.Romaniuk, K.Kierzkowski, S.Simrock: “Prototype Implementation of the Embedded PC Based Control and DAQ Module for TESLA Cavity SIMCON”, TESLA Technical Note, 2004-11
11. W.Giergusiewicz, W.Koprek, W.Jalmuzna, K.T.Pozniak, R.S.Romaniuk: “FPGA Based, DSP Integrated, 8-Channel SIMCON, ver. 3.0. Initial Results for 8-Channel Algorithm”, TESLA Technical Note, 2005-14
12. W. Petersen “The VMEbus Handbook”
13. http://www.xilinx.com/ [Xilinx Homepage]
14. http://www.analog.com/en/prod/0,2877,AD6645,00.html [AD6645 datasheet]
15. http://www.analog.com/en/prod/0%2C2877%2CAD9772A%2C00.html [AD9772 datasheet]