DSP Integrated, Parameterized, FPGA Based Cavity Simulator & Controller for VUV-FEL

SC Cavity SIMCON

version 2.1. rev. 1, 02.2005

USER'S MANUAL

Krzysztof T. Pozniak, Tomasz Czarski, Waldemar Koprek, Ryszard S. Romaniuk
pozniak@mail.desy.de; tczarski@ntmail.desy.de; wkoprek@ntmail.desy.de, rrom@mail.desy.de

Institute of Electronic Systems, Warsaw University of Technology, ELHEP Group
Nowowiejska 15/19, 00-665 Warsaw, Poland

ABSTRACT

The note describes integrated system of hardware controller and simulator of the resonant superconducting, narrowband niobium cavity, originally considered for the TTF and TESLA in DESY, Hamburg (now predicted for the VUV and X-Ray FEL). The controller bases on a programmable circuit Xilinx VirtexII V3000 embedded on a PCB XtremeDSP Development Kit by Nallatech. The 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. The device under consideration will be referred to as superconducting cavity (SCCav) SIMCON throughout this work..

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 DOOCS interface for the described hardware system. The interface was prepared in DOOCS and in Windows. The hardware and software of SIMCON was tested in CHECIA. The results were 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, cold option, cavity simulator, cavity controller, 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.

Acknowledgement: We acknowledge the support of the European Community Research Infrastructure Activity under the FP6 "Structuring the European Research Area" program (CARE, contract number RII3-CT-2003-506395)

MAJOR CHANGES FROM

SIMCON

VERSION 1.0, REV. 1, 04.2004

Published as TESLA FEL Report 2004-04

- 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 paragraph 7.2.3.,

- An input compensation matrix was added to the algorithm of the cavity controller. This change was described in paragraph 7.2.5.,

- The access rules to the tables have been changed.
  This change was described in paragraph 3.2.

- Two options for the voltage range of the cavity Simulator were introduced.
  This change was described in paragraph 7.2.4.

- The dimensions in bits of the tables for amplification values for cavity controller and beam current values for cavity simulator were equalized. The equalization enables realization of automatic switching mechanism.
  This change was described in paragraph 7.2.3.

- An additional VMEbus interface was added.
  This change was described in Annex 13.

- A number of representative exemplary results from SIMCON tests with CHECHIA were added.
  The results are presented in Annex F.

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

CONTENTS

1 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 software models allow for testing of the hardware controller by the step operation mode. The FPGA hardware implementation of the cavity model is intended for the real time operation.

1.1 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. 1.

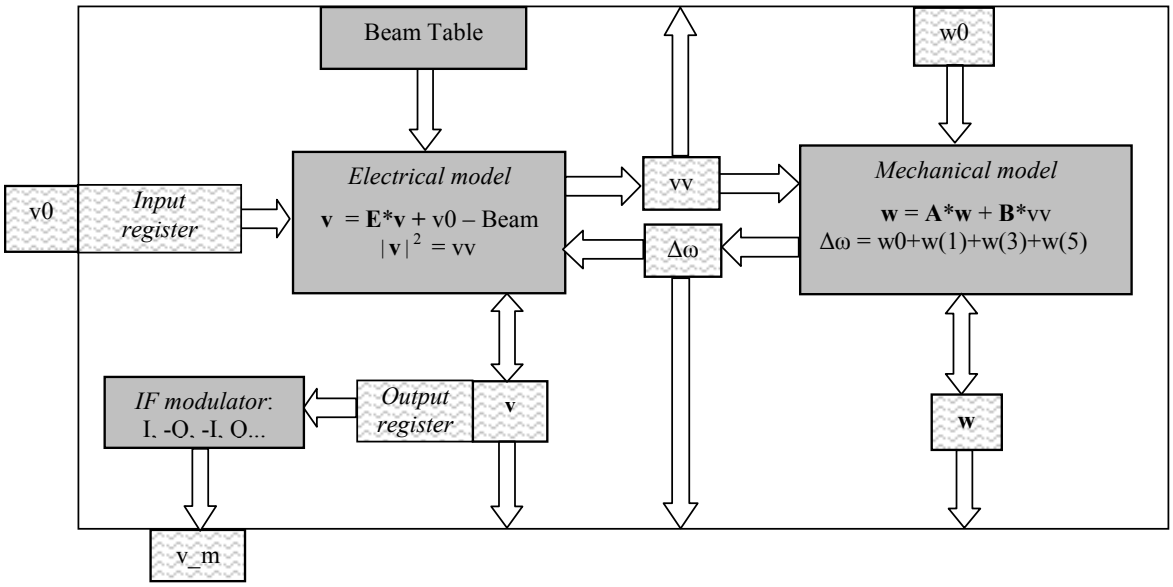


Fig. 1. The functional diagram of the cavity simulator algorithm.

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 $\Delta\omega$.

1.2 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. 2.

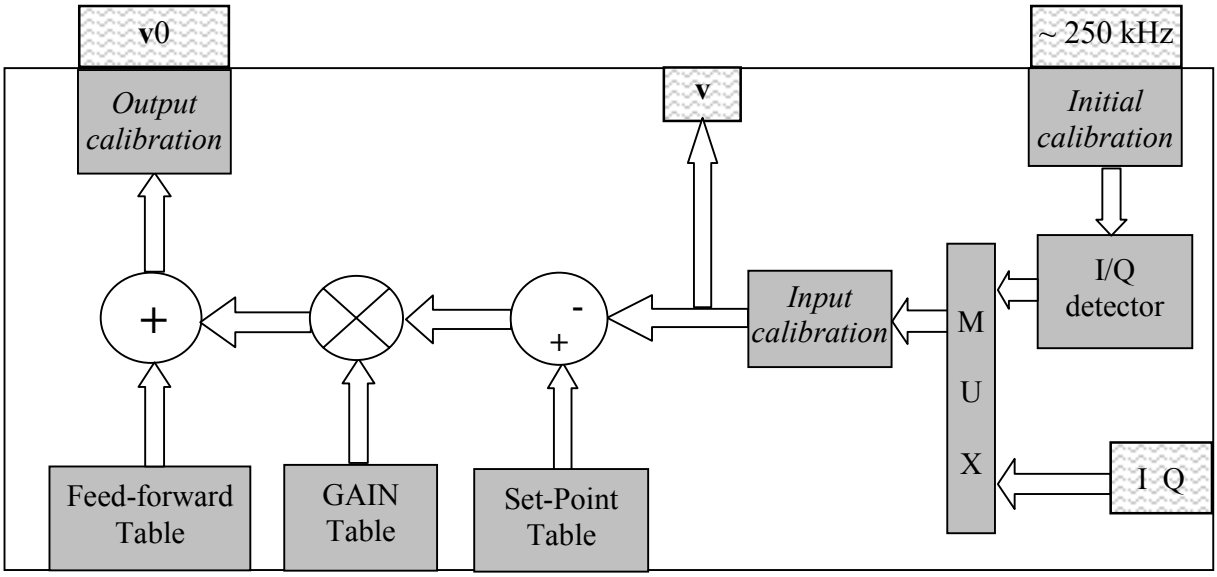


Fig. 2. The functional diagram of the controller algorithm

The digital processing is performed in I/Q detector applying the signal v_m of intermediate frequency 250 kHz from the cavity simulator. Additionally the (I, Q) vector can be directly accepted by MUX switch. The resultant cavity voltage *envelope* (I, Q) is calibrated in the next unit, so to compensate the measurement channel attenuation and phase shifting for an individual cavity. The Set-Point table delivers the required signal level, which is compared to the actual cavity voltage. 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.

1.3 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. 3. 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 during the auto calibration process. Finally, the MATLAB simulation process is verified by plot. The resulting example, for the real operational condition, is presented in fig 4. 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). On the other hand, 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 the user utility.

CAVITY SIMULATOR parameters	CONTROLLER parameters
f_0 = 1300 [MHz]…….resonance frequency	G = [50;50]…….……..... gain
ρ =520 [Ω] …….. characteristic resistance	cal = [1,0] ……...……. initial calibration vector
$Q_L = 3 \cdot 10^6$ ……..…… loaded quality factor	c_in = [1,0] …….. input calibration vector
Δf = Δω/2π = 390 [Hz]……..…...….pre-detuning	c_out = [1,0] …… output calibration vector
d1 = 0, d2 = 1 …….……..… input, output delay	CONTROL parameters
f = [235, 290, 450] [Hz].. resonance frequencies vector	
Q = [100,100,100] …….. quality factors vector	F = 1, (0) ……..……. Feed-forward enable, (disable)
K = [0.4, 0.3, 0.2] [Hz/(MV)^2]… LFD constants vector	a = 25 [MV] ……..…….. cavity amplitude
Ib = 8 [mA] ……..…average beam	ph = 0 [rad] ……..……..… cavity phase
D1 = 509, D2 = 1300 ……..……..…start, stop beam	D = 509 …….filling time
	L = 800 ……...……....……flattop time

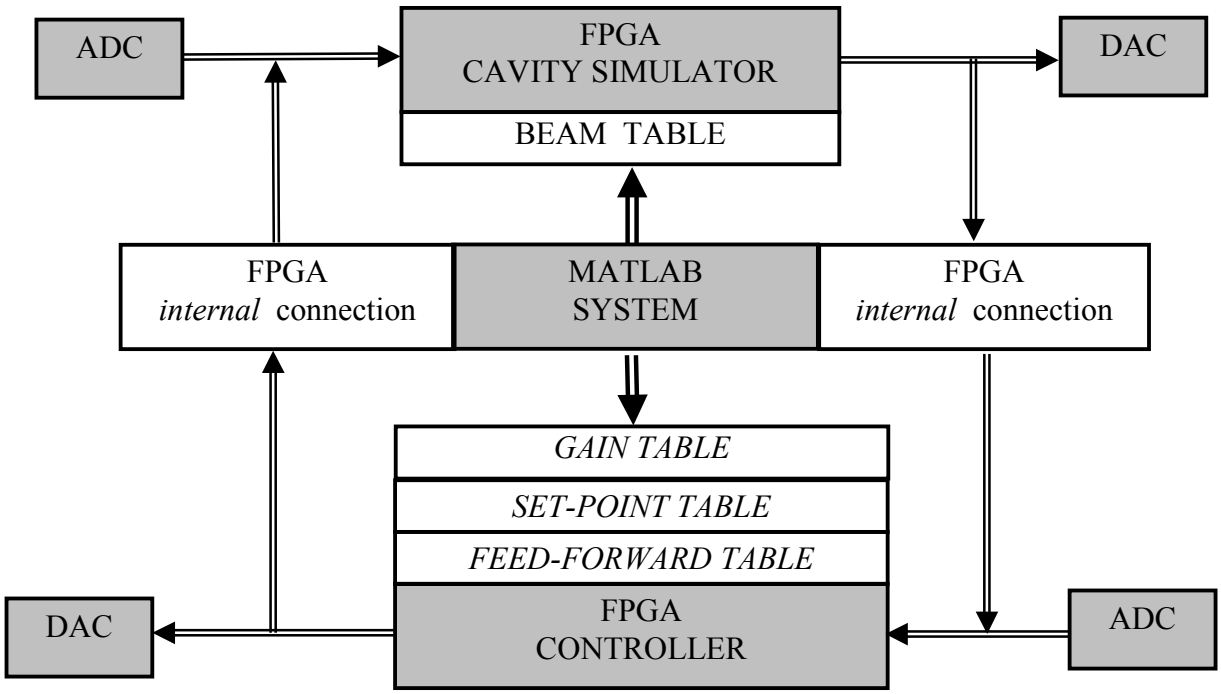


Fig. 3. Functional diagram for one chip FPGA system

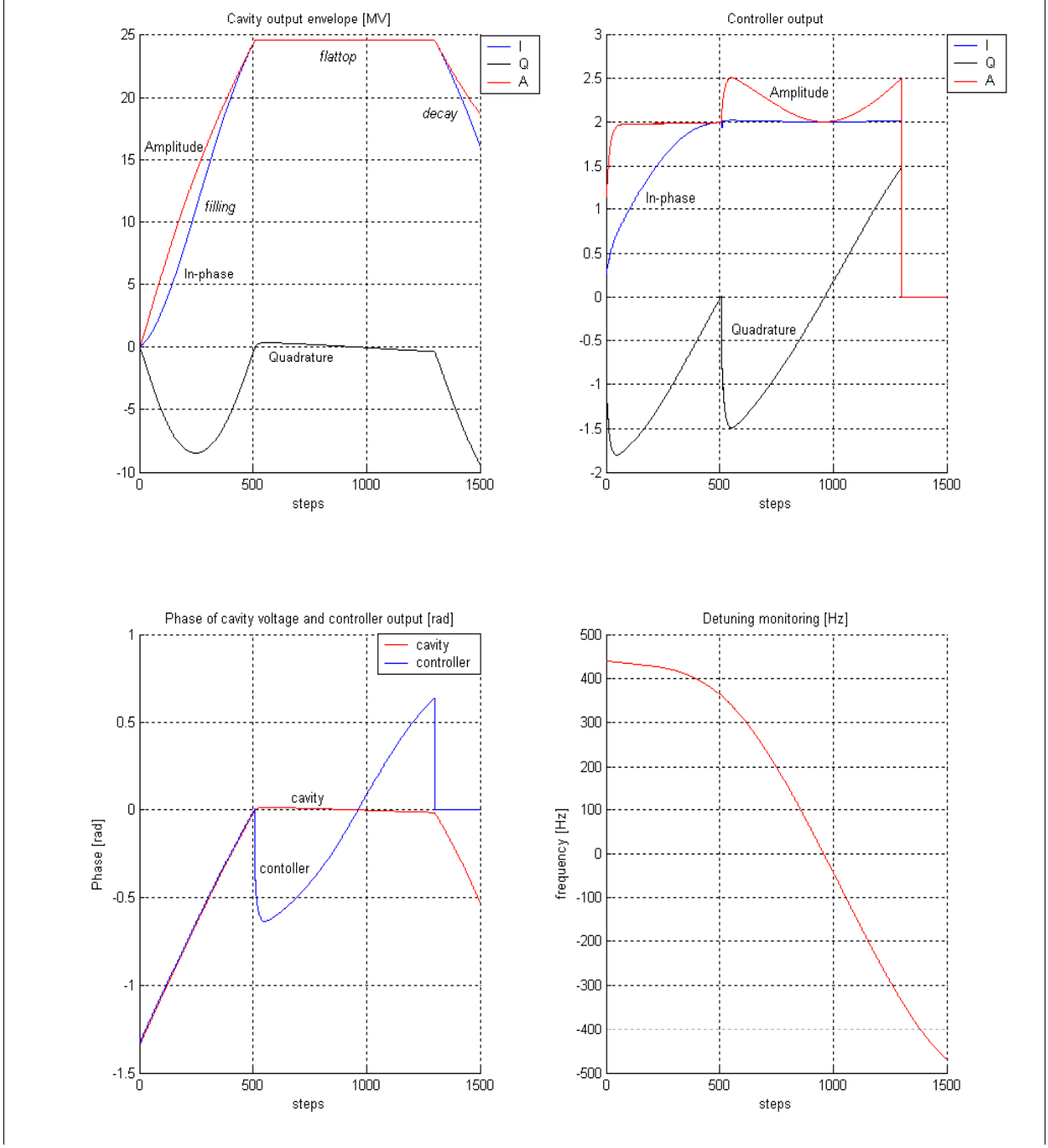


Fig. 4. The MATLAB results of simulation for real operation condition

2 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 V3000. The chip has inbuilt hardware DSP components [8]. This chapter presents in a general way the functional and hardware structure of the device.

2.1 Hardware structure

The hardware layer was realized with *XtremeDSP Development Kit* by Nallatech [12]. The version of the main board (MB) was *BenONE* integrated with daughter board (DB) *BenADDA* (fig.5) The DB is realizing hardware DSP algorithms. The DB possesses two fast 14-bit ADC and DAC and a programmable FPGA Xilinx VirtexII V3000-4 chip equipped in 18x18 bit multiplication circuits.

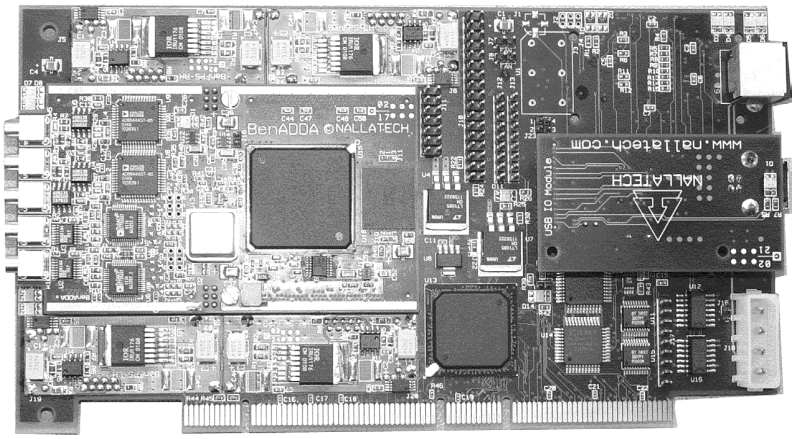


Fig. 5. XtremeDSP Development Kit by Nallatech

Adaptation of the *XtremeDSP Development Kit* to the needs of the user in the EURO 6HE standard was realized by embedding on a dedicated base carrier board *6HE-EURO-VME/EPP*, what was presented in fig. 6. The front-side of the board possesses:

- Two analog inputs of the signals, with the signal range ±1V. Each channel signal is processed nondependently by a 14-bit ADC converter, working with 40MHz clock;
- Input of an external clock, for TTL standard;
- Two analog signal outputs with the signal range ±1V. Each channel signal is processed nondependently by a 14-bit DAC converter, working with 40MHz clock.
- Three digital inputs and three digital outputs in TTL standard, connected to the FPGA Virtex II via a suitable buffer;
- Two information LEDs;
- Socket for a parallel interface in EPP standard for communication of FPGA Virtex II with a control PC;

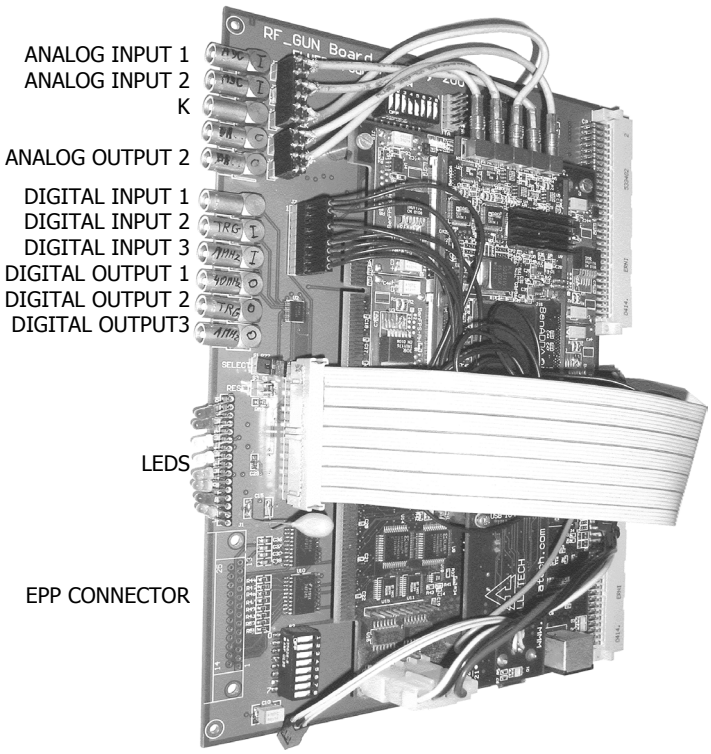


Fig. 6. XtremeDSP Development Kit embedded on a carrier board EURO-6HE.

The SIMCON card occupies two slots in the VME 6U crate. Power supply is provided either from the VME bus or from the power unit provided with the Nallatech board.

Application of the VME bus [9] 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* [9]. 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 appendix 13 describes the implemented VME bus interface.

The application of the EPP protocol to communicate with the computer is excused by the low throughput of the proprietary USB. Ver.1. offered by the board manufacturer. The choice of the EPP protocol is justified by its comparably high transmission speed, simplicity and popularity in the PC computers. The EPP protocol has a simple implementation in the FPGA chip. The hardware realization of the interface was described in detail in [2, 12]. The appendix B describes the technical realization of the EPP interface.

2.2 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 V3000-4 chip on the *XtremeDSP Development Kit* board. There were used the AD and DA converters situated on the *BenADDA* daughter 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 board *6HE-EURO-VME/EPP* were used for synchronization with the 1MHz clock and 5 Hz trigger. These signals are distributed in the whole control system of the FEL. An overall functional structure of the SIMCON, implemented in ver.1.0 was presented in fig. 7.

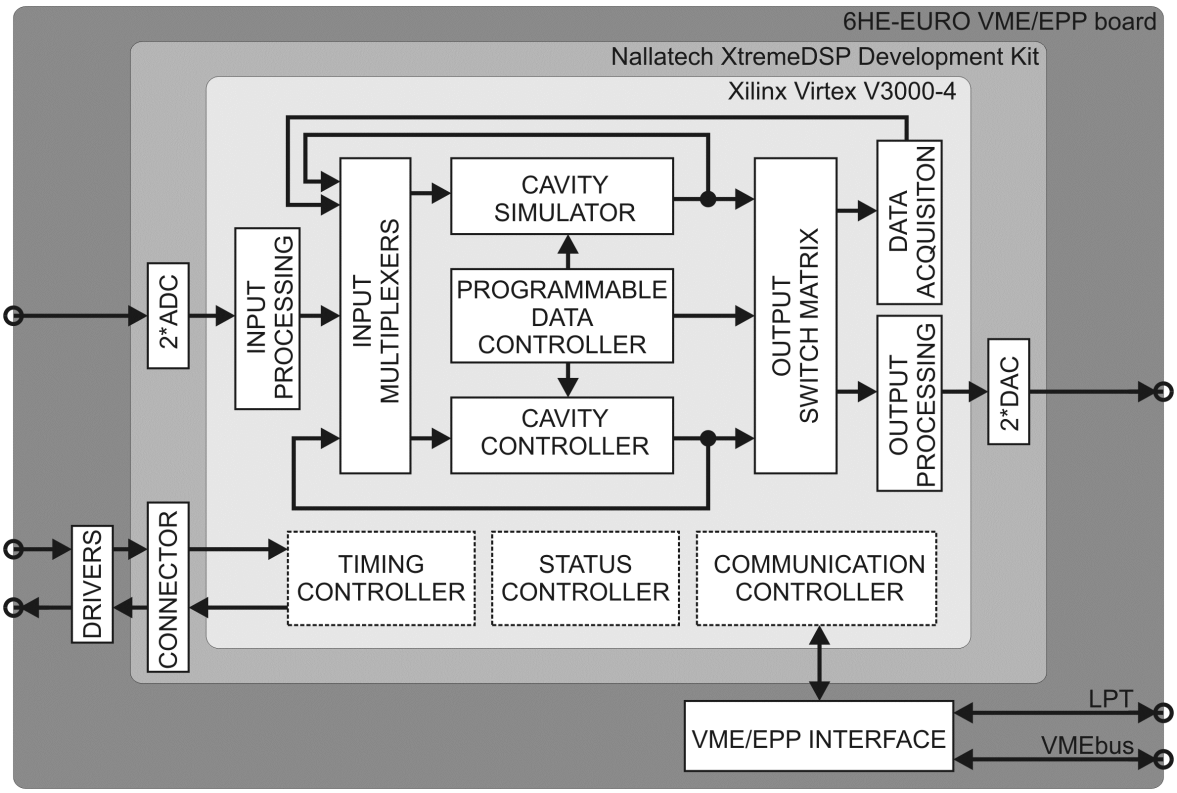


Fig. 7. Multi-Layered hardware and functional structure of the SIMCON

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 CAVITY CONTROLLER. They were programmed inside the FPGA Virtex II 3000-4 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 cavity controller, beam of cavity simulator). The set parameters stem from the algorithms described in detail in the following papers [4,6,7].

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 the DAQ block. A suitable configuration of the switching matrices gives appropriate analog feedback between the modules of cavity 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 control computer system with the aid of COMMUNICATION CONTROLLER block. The EPP hardware transmission protocol was used. The information distribution bases on the *Internal Interface* standard, described in detail in [2,8].

3 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.

3.1 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 4.2, 4.3.1)
- *EXTERNAL* – the work in the real time mode is possible with the usage of external timing signals (see chapter 4.2)
- *VECTOR* – the work is possible in the real time mode with internal timing signals and set exciting vectors (see chapter 4.2, 10.2.4)
- *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 4.2, 4.3.2)

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.

3.2 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 paragraph 3.2.1),
- value 1 – *EXTERNAL* work mode is chosen (compare paragraph 3.2.2),
- value 2 – *VECTOR* work mode is chosen (compare paragraph 0),
- value 3 – *STEP* work mode is chosen (compare paragraph 3.2.4),

The flag CTRL_PROC_REQ enables, for the block CAVITY_CONTROLLER, enables the choice between activation or programming of tables for the cavity controller inside the block of PROGRAMMABLE_DATA_CONTROLLER. It also allows activation of static values (compare paragraph 7.2.5). 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 cavity 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 7.2.4). 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 cavity 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 100ns**.

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

3.2.1 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 paragraphs 7.2.3 and 7.2.5).

3.2.2 EXTERNAL mode operation

In the *EXTERNAL* mode of operation the outside timing signals are used in the TTL standard (compare chapter 4.1). The signals are respectively connected to the LEMO sockets (compare chapter 2.1):

- *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 3.2.1). 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 4.3.3).

3.2.3 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 10.2.4). By the choice of the channels in the block INPUT MULTIPLEXERS they are

respectively connected to the input of the cavity controller and input of the cavity simulator. (compare chapter 11.2).

From the programming steering side, the *EXTERNAL* operation mode is considerably identical with the *INTERNAL* operation mode (compare chapter 3.2.1). Only in the case of the DAQ memory choice as an input generator, it requires programming with a set of signals (compare chapter 10.2.4). The memory module working as a generator **may not be used simultaneously** for data acquisition.

3.2.4 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 7.2.4) and the block `CAVITY CONTRLLLER` (see chapter 7.2.5). A single step is realized in the module *CIVITY STROBE STEP TIMER* in block `TIMING CONTROLLER` (see chapter 4.3.2).

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.**

4 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):

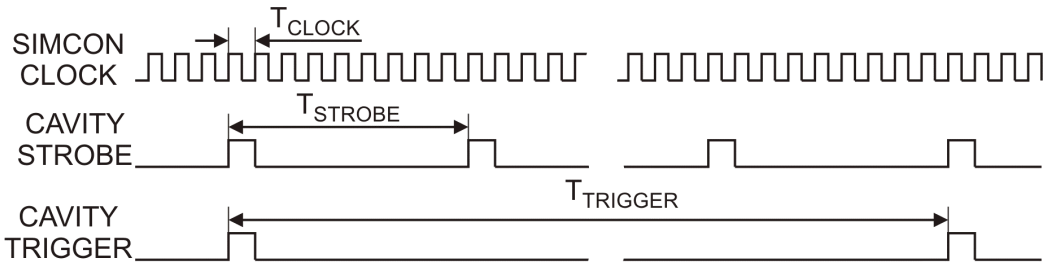


Fig. 8. Time dependencies between clock signals in the block `TIMING CONTROLLER`

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

4.1 Functional structure

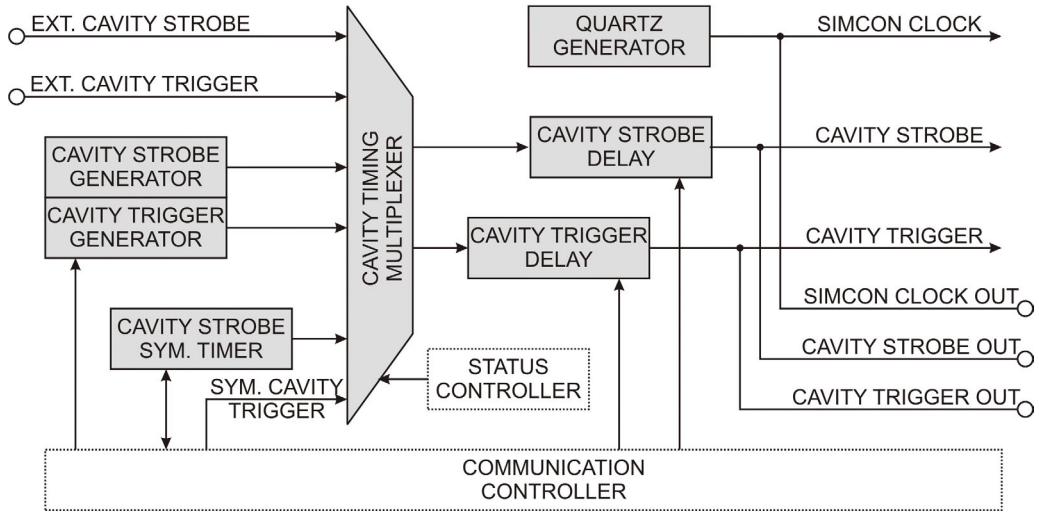


Fig. 8. Functional structure of the block `TIMING CONTROLLER`

The functional structure of the block `TIMING CONTROLLER` was presented in fig. 8. 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.

4.2 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`.

4.3 Programming description

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

4.3.1 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 *200ns*, 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 1s.

4.3.2 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 21:

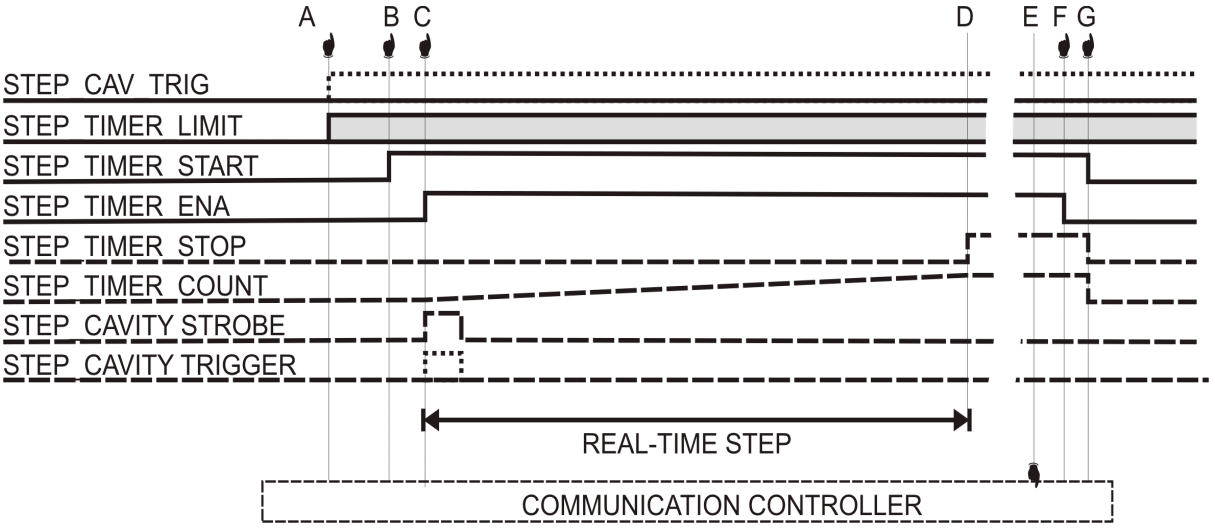


Fig. 9. Time diagram for the process *STEP OPERATION*

- 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*.
- B initialization of the module *STEP TIMER* through setting STEP_TIMER_START=1.
- 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.
- 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.
- 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.

- F stopping the work of the module *STEP TIMER* through setting STEP_TIMER_ENA=0.
- 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 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 cavity 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].**

4.3.3 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.

5 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.

5.1 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. 10.

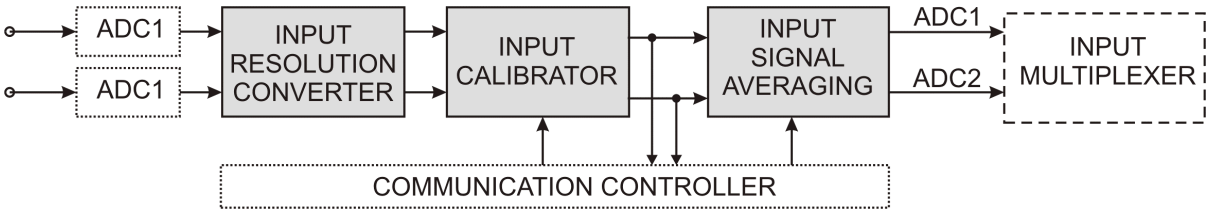


Fig. 10. Functional structure of the block `INPUT PROCESSING`

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 each ADC channel 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 $K=1,2,4,8$.

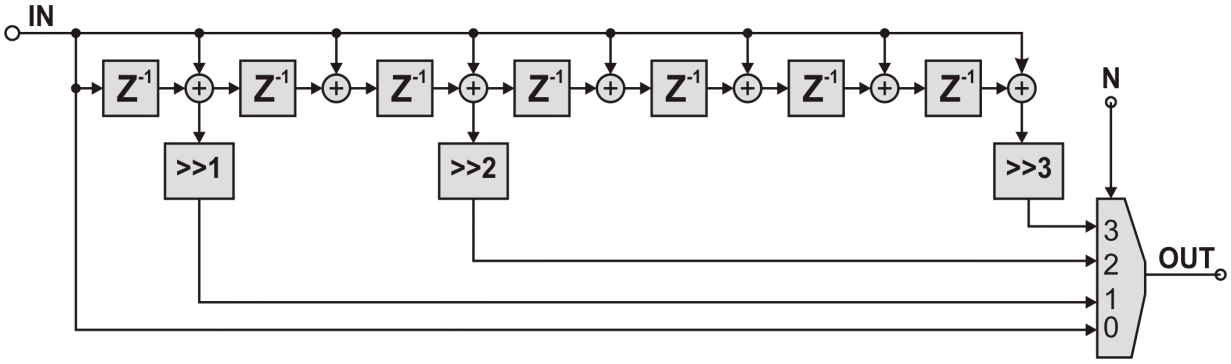


Fig. 11. Time dependencies for cyclical data

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.11.

5.2 Programming description

Programming of the block `INPUT PROCESSING` relies on setting of calibration coefficients, the choice of a common, for both ADC channels, averaging coefficient.

The calibration parameters are determined by:

- For the channel *ADC1*, registers ADC1_GAIN and ADC1_OFFSET,
- For the channel *ADC2*, registers ADC2_GAIN and ADC2_OFFSET.

The choice the common averaging coefficient for both *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 (compare paragraph 7.2.3):

- for the channel ADC1 registers ADC1_GAIN_BUF and ADC1_OFFSET_BUF,
- for the channel ADC2 registers ADC2_GAIN_BUF and ADC2_OFFSET_BUF.

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 both *ADC* converters may be done through reading:

- For the channel *ADC1*, the register ADC1_DATA,
- For the channel *ADC2*, the register ADC2_DATA.

The following reading of the *ADC* 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 *25ns*.

6 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.

6.1 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 12.

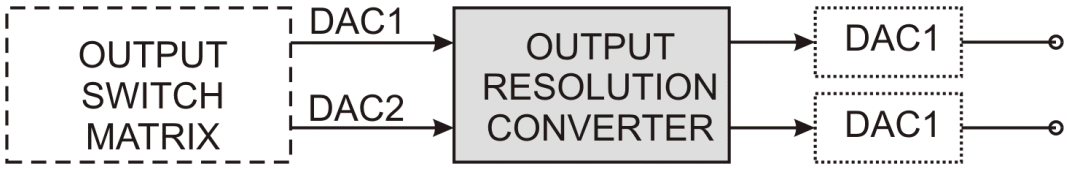


Fig. 12. Functional structure of the block `INPUT PROCESSING`

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.

6.2 Programming description

The current version of the SIMCON system block `OUTPUT PROCESSING` does not require any programming and does not forward any results of its actions to the block `COMMUNICATION CONTROLLER.`

7 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 cavity controller SGAIN_I and SGAIN_Q, see chapter s 1.2, 9.2),
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 1.1, 8.2),
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.

7.1 Functional structure

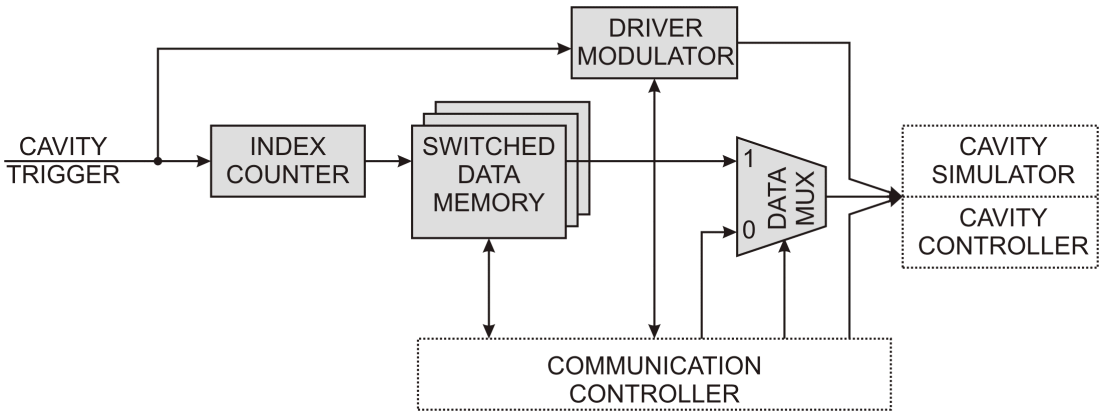


Fig. 13. Functional structure of the block PROGRAMMABLE DATA CONTROLLER

The functional structure of the block PROGRAMMABLE DATA CONTROLLER was presented in fig. 13. 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. 15.

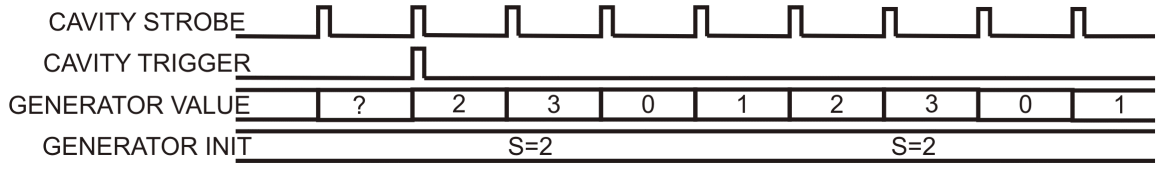


Fig. 14. Time dependencies for cyclic data.

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 *1μs* for the period of time *2048μs*, 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 7.2.3).

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. 14 for the initializing value S=2.

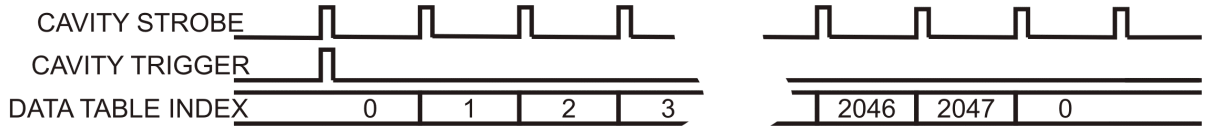


Fig. 15. Time dependencies for dynamic data

7.2 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.

7.2.1 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 paragraph 3.2):

- CTRL_PROC_REQ=0 chooses channel *0* for all static data of block CAVITY CONTROLLER (compare paragraph 7.2.5),
- SIM_PROC_REQ=0 chooses channel *0* for all static data block CAVITY SIMULATOR (compare paragraph 7.2.4),
- CTRL_PROC_REQ=1 chooses channel *1* for all dynamic data block CAVITY CONTROLLER (compare paragraph 7.2.5),
- SIM_PROC_REQ=1 chooses channel *1* for all dynamic data block CAVITY SIMULATOR (compare paragraph 7.2.4),

7.2.2 Modulator driver

The module *MODULATOR DRIVER* requires programming of the initial value in the register VM_DRV_START in the range from *0* to *3*.

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

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

7.2.3 Data switching

The request to change the static and dynamic data relies on the negation of the actual flag state TAB_SWITCH_REQ, i.e. changing its value from 0 to 1 or vice versa. The acknowledgement of the change causes copying of the new value to TAB_SWITCH_ACK.

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

- ADC1_GAIN with ADC1_GAIN_BUF and ADC1_OFFSET with ADC1_OFFSET_BUF,
- ADC2_GAIN with ADC2_GAIN_BUF and ADC2_OFFSET with ADC2_OFFSET_BUF,
- CAL1 with CAL1_BUF and CAL2 with CAL2_BUF,
- TSETPOINT_I with DAQ1 and TSETPOINT_Q with DAQ2,
- TFEEDFORWARD_I with DAQ3 and TFEEDFORWARD_Q with DAQ4,
- 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 TAB_SWITCH_REQ.

7.2.4 Cavity simulator programmable data packet

The block CAVITY SIMULATOR, in accordance with the algorithm described in chapter 1.1 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 [C]* (static data): $C = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$ are expressed in succession by the parameters CAL1 and CAL2 (exchange registers: CAL1_BUF, CAL2_BUF),.
- MATRIX_A1_21 and MATRIX_A1_22 – individual coefficients of the matrix $[A_1]$,
- *MATRIXES [A1], [A2], [A3]* (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*: *16MV/m*,
  - the value *1*: *32MV/m*.
- Activation of the static register requires setting SIM_PROC_REQ=0, alternatively switching on the set point tables SIM_PROC_REQ=1 (compare paragraph 3.2).

7.2.5 Cavity controller programmable data packet

The block CAVITY CONTROLLER, in agreement with the algorithm described in the chapter 1.2 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,\
- *COMP1 .. COMP4* (static data): is represented by the registers COMP1 .. COMP4. (swap registers: COMP1_BUF .. COMP4_BUF)

Activation of the static registers requires setting CTRL_PROC_REQ=0 or alternatively switching on the set point tables CTRL_PROC_REQ=1 (compare paragraphs 3.2).

8 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 1.1). An 18-bit fixed point algorithm was implemented with the use of the DSP components present in the FPGA chip Xilinx VirtexII-V3000.

8.1 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 16.

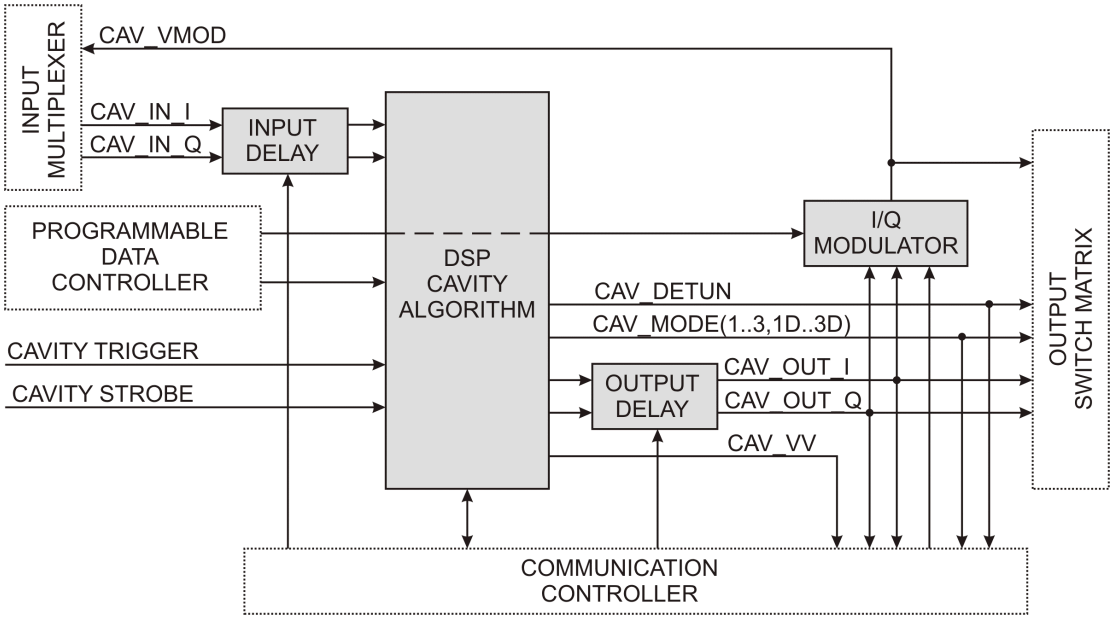


Fig. 16. Functional structure of the block CAVITY SIMULATOR

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 11) in accordance with the parameters provided from the block PROGRAMMABLE DATA CONTROLLER (see chapter 5). 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 7.1).

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 *1μs*.

8.2 Programming description

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

- Setting of proper parameters in the block PROGRAMMABLE DATA CONTROLLER (see chapter 7.2.5). 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 cavity controller (compare chapter 7.1 and 7.2.2). The phase change is determined by the register VM_DRV_OFFSET in the value range of 0 - 3,
- 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 *15μs* with a step of *1μs*. 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:
  - *CAV_MODE1* – the first mechanical mode is accessible in the register CAV_MODE1,
  - *CAV_MODE1D* – derivative of the first mechanical mode is accessible via the register CAV_MODE1D,
  - *CAV_MODE2* – the second mechanical mode is accessible in the register CAV_MODE2,
  - *CAV_MODE2D* – derivative of the second mechanical mode is accessible via the register CAV_MODE2D,
  - *CAV_MODE3* – the third mechanical mode is accessible via the register CAV_MODE3,
  - *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.***

9 CAVITY CONTROLLER BLOCK DESCRIPTION

The block CAVITY 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 1.2). There was implemented an 18-bit fixed point algorithm, with the usage of the DSP components integrated into the FPGA Xilinx VirtexII-V3000 chip.

9.1 Functional structure

The block CAVITY 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 17.

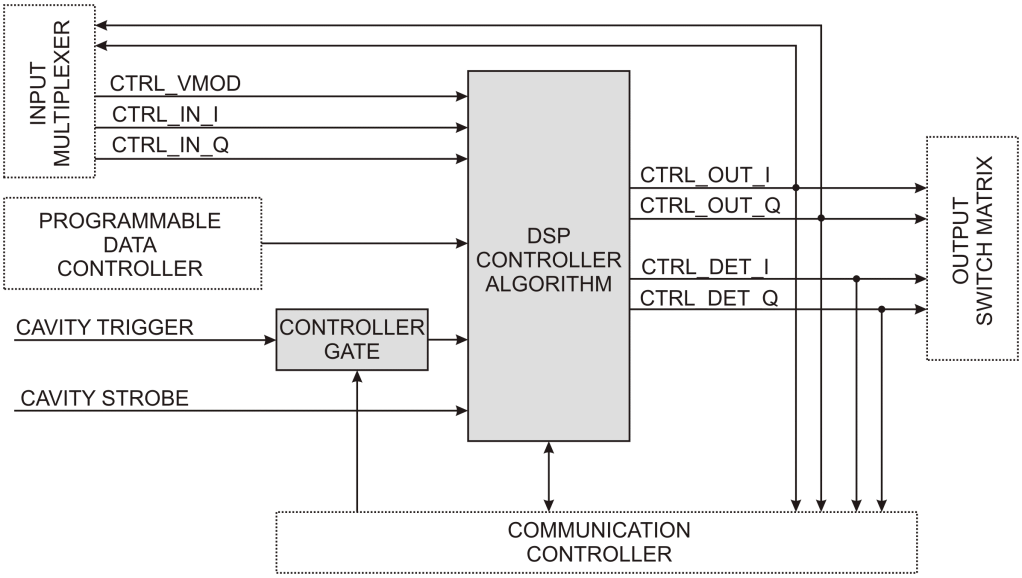


Fig. 17. Functional structure of the block CAVITY CONTROLLER

The module *DSP CONTROLLER ALGORITHM* processes the appropriate modulated input signals provided from the block INPUT MULTIPLEXER (see chapter 11) in accordance with the parameters provided by the block PROGRAMMABLE DATA CONTROLLER (see chapter 5). The output of the module gives two output vectors:

- Basic control signal for vector modulator of the klystron (*CTRL_I*, *CTRL_Q*),
- Auxiliary signal after the detection (CTRL_DET_I, CTRL_DET_Q)

The module *CONTROLLER GATE* allows to activate the block CAVITY 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. 18.

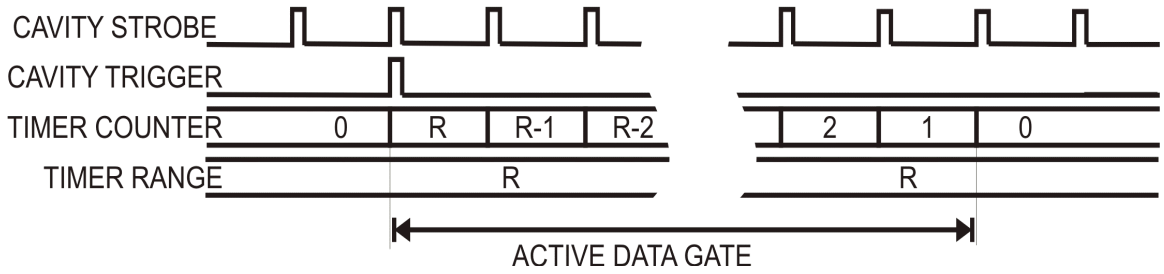


Fig. 18. Time dependencies of signals for the time gate of the cavity controller

9.2 Programming description

The programming of the block CAVITY CONTROLLER relies on:

- Setting of appropriate parameters in the block PROGRAMMABLE DATA CONTROLLER (see chapter 7.2.3). These are the following parameters: *SET POINT*, *FEED FORWARD* and *GAIN*.
- Setting of the choice variant for control signals through the value CTRL_IQ_MODE:
  - *0:* choice of modulated signal *CTRL_VMOD*,
  - *1:* choice of pair of signals CTRL_IN_I and CTRL_IN_Q
- 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 *1μs*).

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 cavity 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 cavity controller DSP processing source signal:

- register CTRL_VMOD controlling directly the signal *CTRL_VMOD*.

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 cavity controller DSP signals:

- signals *CTRL_I* and *CTRL_Q* respectively through the registers CTRL_OUT_I and CTRL_OUT_Q,
- 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 cavity 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.***

10 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.

10.1 Functional structure

The block DAQ realizes parallel, synchronized registration of four data streams. Its functional structure is presented in figure 19.

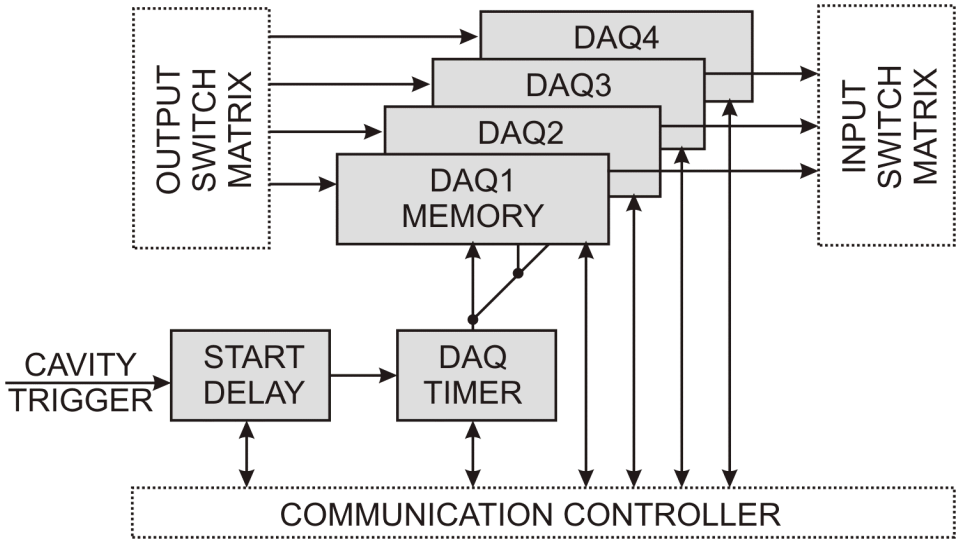


Fig. 19. Functional structure of the block DATA ACQUISITION

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 $1\mu s$.

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 cavity controller and to two inputs of the cavity simulator.

10.2 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.

10.2.1 DAQ modes control

The basic operation modes of the DAQ block are set with the flag DAQ_PROC_REQ.

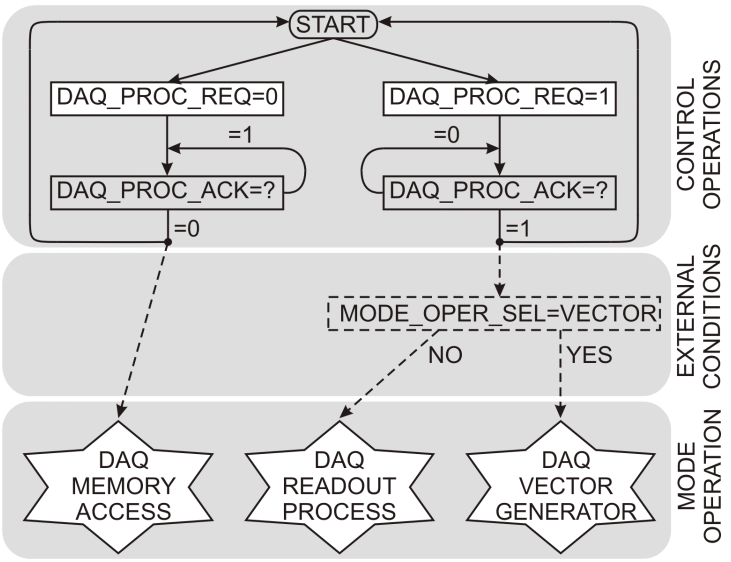


Fig. 20. Flow diagram for the choice of the operation mode of the DAQ block.

Acknowledgement of the required operation mode is obtained by reading of the identical 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. 20. 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.

10.2.2 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.

10.2.3 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.

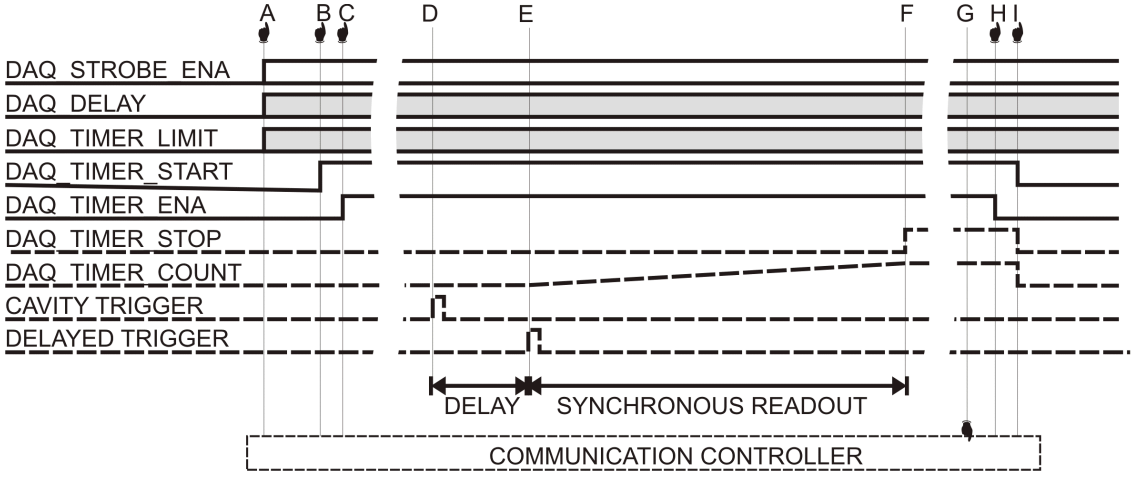


Fig. 21. Time diagram of the data acquisition process in the block DAQ

The debated operation mode allows for parallel data acquisition of four data streams. It is performed automatically, according to the a priori preset parameters. The key stages of the control process for the data acquisition are presented in figure 21:

- 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 *40MHz* (every *25ns*),
    1:data will be registered with the speed of the FEL clock *1MHz* (every *1μs*)
  - 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 *1MHz,* or a single step is *1μs*. Assuming the value of 0 means no additional delay for the signal *CAVITY TRIGGER*.
- B module initialization *DAQ TIMER* via setting DAQ_TIMER_START=1.
- 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.
- 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
- 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.
- 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.
- 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.
- H stopping of the work of the module *DAQ TIMER* via setting DAQ_TIMER_ENA=0.
- 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 10.2.2).

If the global DAQ conditions remain unchanged, at the next initialization of the DAQ process, the stage A may be omitted.

10.2.4 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 10.2.2).

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

11 INPUT MULTIPLEXERS BLOCK DESCRIPTION

The input multiplexers allow for nondependent programmable choice of the input control signals for the blocks `CAVITY SIMULATOR` and `CAVITY 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).

11.1 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 `CAVITY CONTROLLER`. Both multiplexers have 4 variants for the choice of the inputs. . The functional structure of the block `INPUT MULTIPLEXERS` was presented in figure 22.

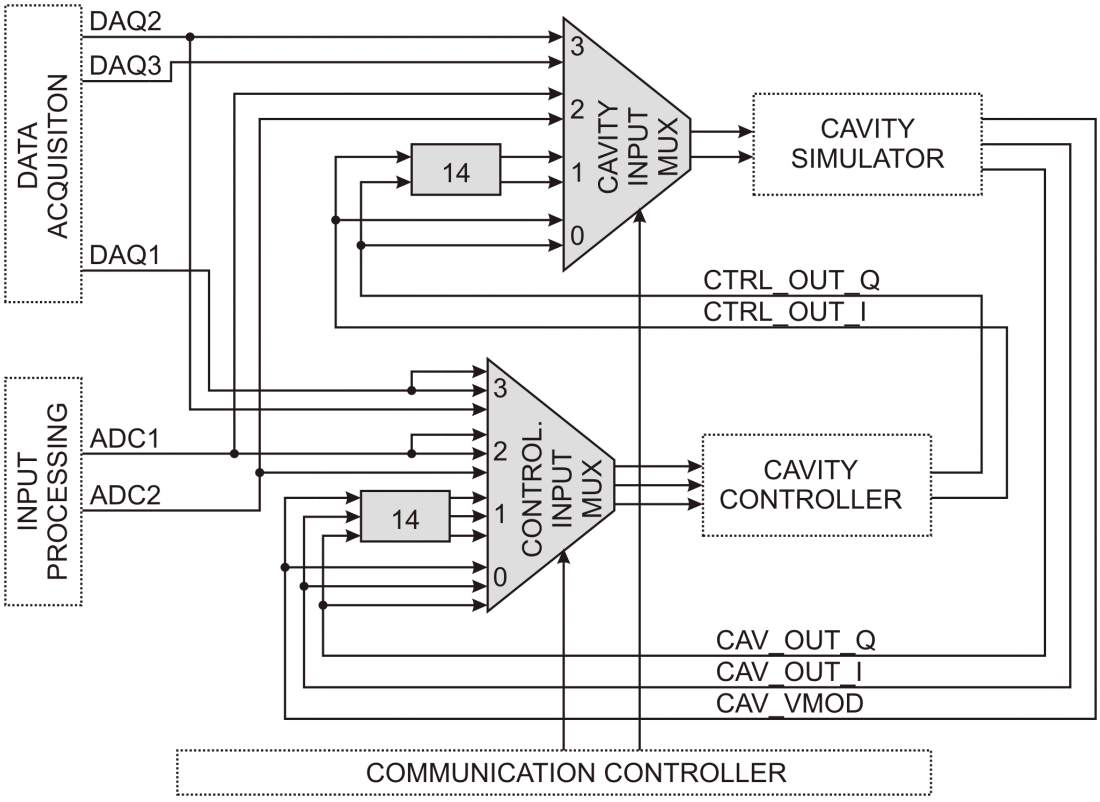


Fig. 22. Functional structure of the block `INPUT MULTIPLEXERS`

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 cavity controller and simulator,
- *channel 1*: digital simulation of the analog (14-bit) feedback, respectively between the DSP blocks of cavity controller and simulator, simulating the resolution of the AD and DA converters,
- *channel 2*: connection of respective signals from the AD converters from blocks `DAC1` and `DAC2`,
- *channel 3*: connection of blocks `DAQ1..DAQ3` with the choice of the VECTOR operation mode (see chapter 0) or zeroing the inputs for the rest of the operation modes.

11.2 Programming description

The choice of the source channel (compare figure 19) 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.***

12 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`, `CAVITY CONTROLLER`, `ADC1` and `ADC 2` and from the module *TEST GENERATOR* to the inputs of the blocks `DAQ1..4` 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.

12.1 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 23.

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

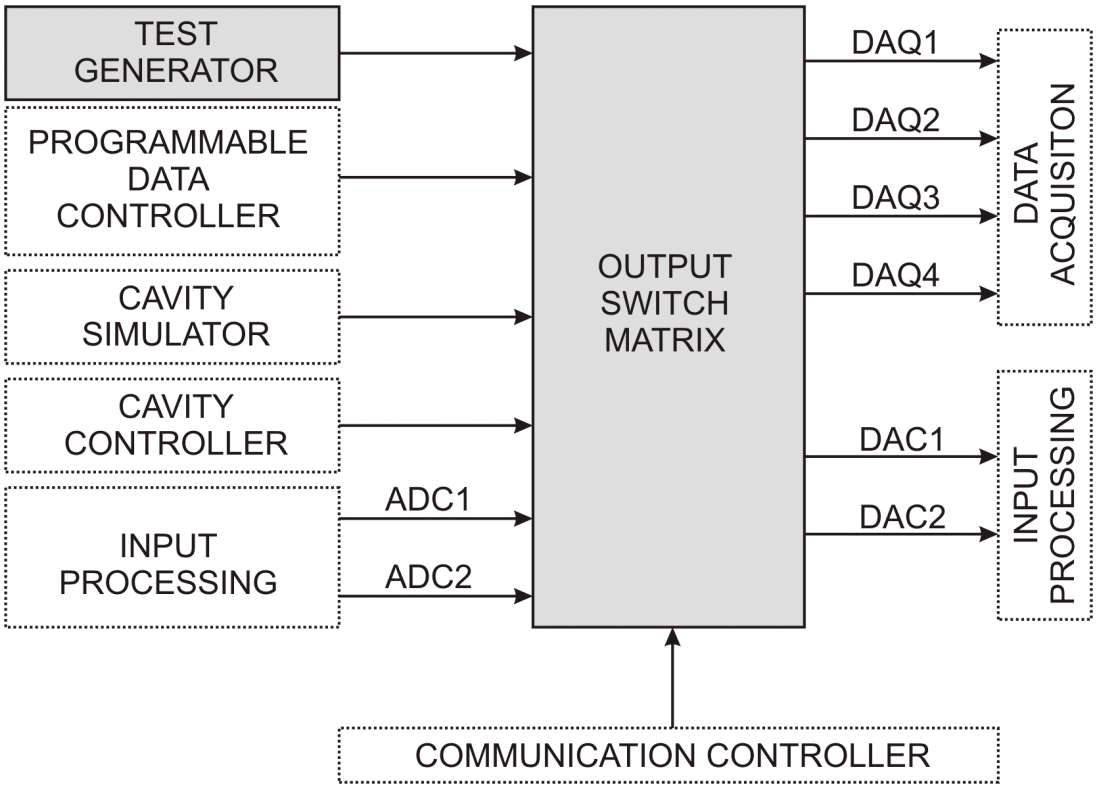


Fig. 23. Functional structure of the block `INPUT MULTIPLEXERS`

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 8.1),
- *channel 2*: internal signal *CAV_ OUT_Q* (compare chapter 8.1),
- *channel 3*: internal signal *CAV_ DETUN* (compare chapter 8.1),

- *channel 4*: internal signal *CAV_VMOD* (see chapter 8.1),
- *channel 5*: internal signal *CTRL_DET_I* (compare chapter9.1),
- *channel 6*: internal signal *CTRL_DET_Q* (compare chapter9.1),
- *channel 7*: internal signal *CTRL_I* (compare chapter9.1),
- *channel 8*: internal signal *CTRL_Q* (compare chapter9.1),
- *channel 9*: internal signal *TGAIN_I* (compare chapter7.1),
- *channel 10:* internal signal *TGAIN_Q* (compare chapter7.1),
- *channel 11*: internal signal *TSETPOINT_I* (compare chapter7.1),
- *channel 12*: internal signal *TSETPOINT_Q* (compare chapter7.1),
- *channel13* : internal signal *TFEEDFORWARD _I* (compare chapter7.1),
- *channel 14*: internal signal *TFEEDFORWARD _Q* (compare chapter7.1),
- *channel 15*: internal signal *TBEAM _I* (compare chapter7.1),
- *channel 16*: internal signal *TBEAM _Q* (compare chapter7.1),
- *channel 17*: internal signal *CAV_MODE1* (compare chapter 8.1),
- *channel 18*: internal signal *CAV_MODE1D* (compare chapter 8.1),
- *channel 19*: internal signal *CAV_MODE2* (compare chapter 8.1),
- *channel 20*: internal signal *CAV_MODE2D* (compare chapter 8.1),
- *channel 21*: internal signal *CAV_MODE3* (compare chapter 8.1),
- *channel 22*: internal signal *CAV_MODE3D* (compare chapter 8.1),
- *channel 23*: input signal *ADC1* (compare chapter5.1),
- *channel 24*: input signal *ADC2* (compare chapter5.1),

12.2 Programming description

The choice of the source channel (compare figure 19) 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 DAC1 for the register MUX_OUT_DAC1,
- for the block DAC2 for the register MUX_OUT_DAC2.

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.***

13 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.

13.1 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: 0028BD94H.

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: 53494D43H.

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:

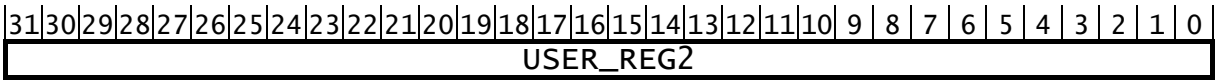
- MAIN_VERSION (RO): contains value 02H,
- SUB_VERSION (RO): contains value 01H,
- REV_VERSION (RO): contains value 0001H.
-

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																															

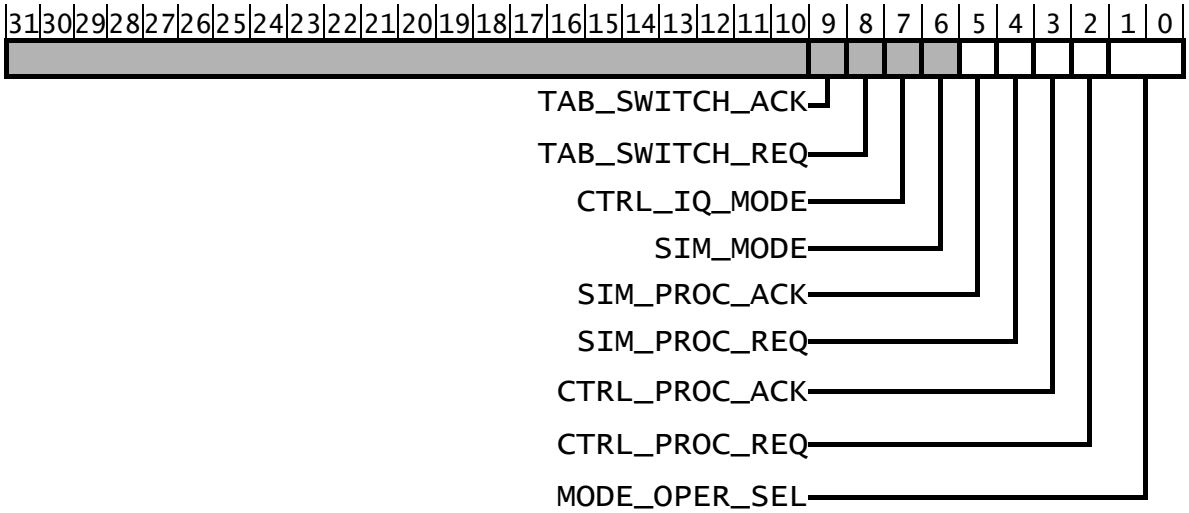
USER_REG1 (RW) – control-test register designed solely for the user.

USER_REG2 (0005H)



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

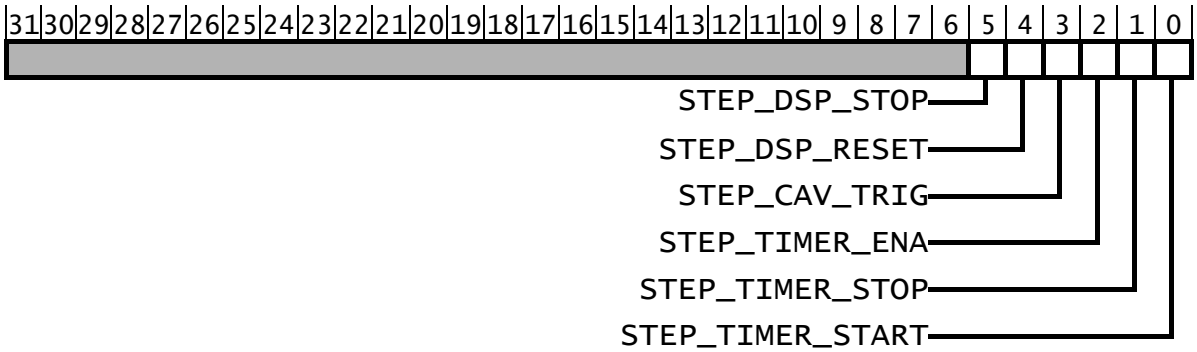
STATUS (0006H)



Packet STATUS contains global components of the control of SIMCON system:

- MODE_OPER_SEL (RW): see chapter 3.2,
- CTRL_PROC_REQ (RW): see chapter 3.2,
- CTRL_PROC_ACK (R): see chapter 3.2,
- SIM_PROC_REQ (RW): see chapter 3.2,
- SIM_PROC_ACK (R): see chapter 3.2,
- SIM_MODE (RW): see chapter 3.2,
- CTRL_IQ_MODE (RW): see chapter 3.2,
- TAB_SWITCH_REQ (RW): see chapter 3.2,
- TAB_SWITCH_ACK (R): see chapter 3.2.

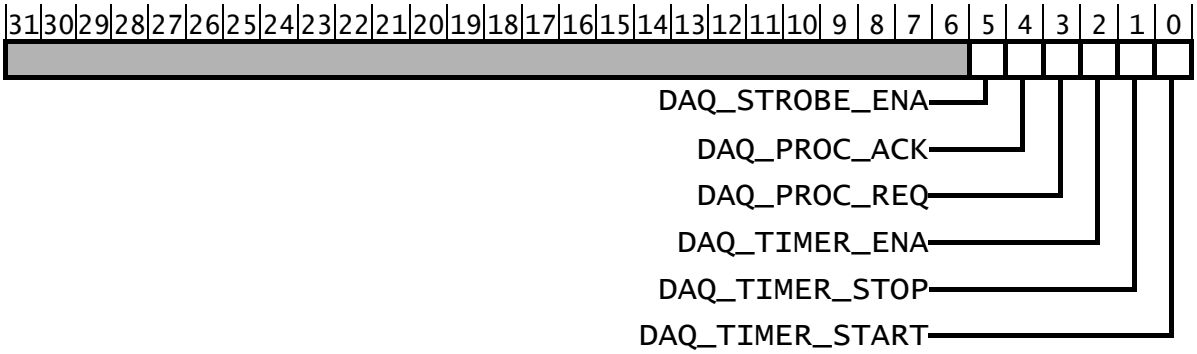
STEP (0007H)



Packet STEP contains components for the step control of the SIMCON system:

- STEP_TIMER_START (RW): see chapter 4.3.2,
- STEP_TIMER_STOP (RO): see chapter 4.3.2,
- STEP_TIMER_ENA (RW): see chapter 4.3.2,
- STEP_CAV_TRIG (RW): see chapter 4.3.2,
- STEP_DSP_RESET (RW): see chapter 4.3.2,
- STEP_DSP_STOP (RO): see chapter 4.3.2.

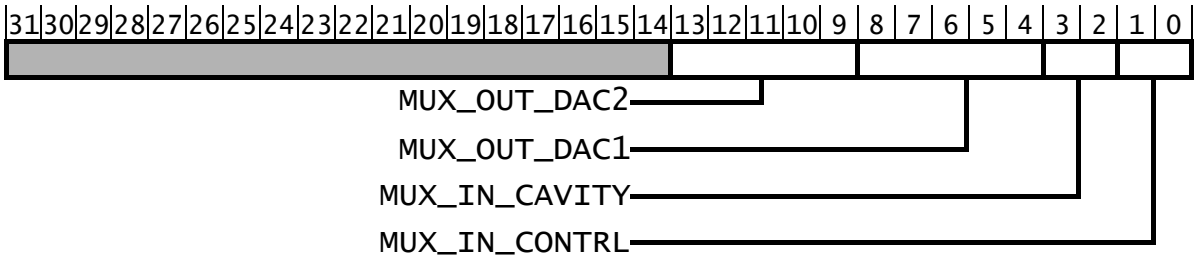
DAQ (0008H)



Packet DAQ contains control components for the DAQ process of the SIMCON system:

- DAQ_TIMER_START (RW): see chapter 10.2.3,
- DAQ_TIMER_STOP (RO): see chapter 10.2.3,
- DAQ_TIMER_ENA (RW): see chapter 10.2.3,
- DAQ_PROC_REQ (RW): see chapter 10.2.1,
- DAQ_PROC_ACK (RO): see chapter 10.2.1,
- DAQ_STROBE_ENA (RW): see chapter 10.2.3.

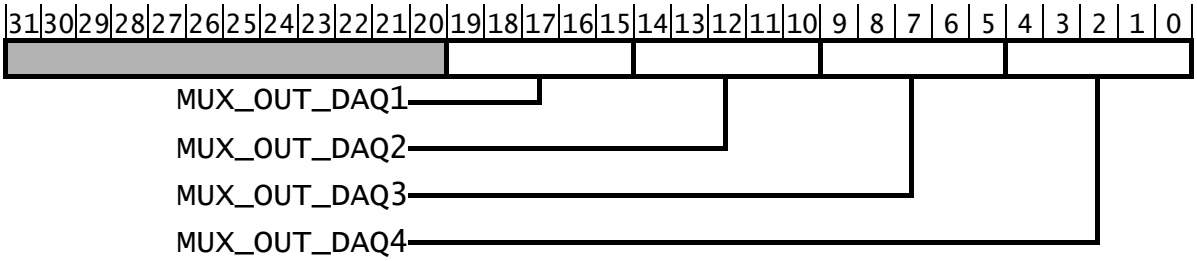
SIGNAL_MUX (0009H)



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

- MUX_IN_CONTRL (RW): see chapter 11.2,
- MUX_IN_CAVITY (RW): see chapter 11.2,
- MUX_OUT_DAC1 (RW): see chapter 12.2,
- MUX_OUT_DAC2 (RW): see chapter 12.2.

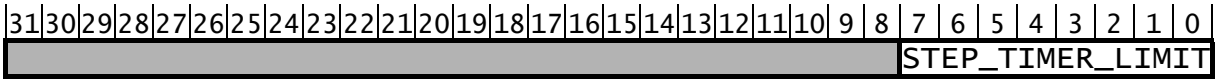
DAQ_MUX (000AH)



Packet DAQ_MUX contains control components for multiplexers for DAQ blocks:

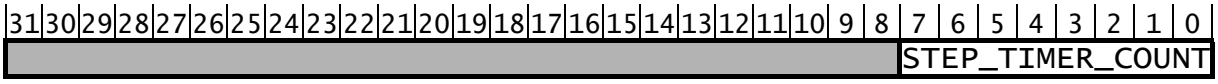
- MUX_OUT_DAQ1 (RW): see chapter 12.2,
- MUX_OUT_DAQ2 (RW): see chapter 12.2,
- MUX_OUT_DAQ3 (RW): see chapter 12.2,
- MUX_OUT_DAQ4 (RW): see chapter 12.2.

STEP_TIMER_LIMIT (000BH)



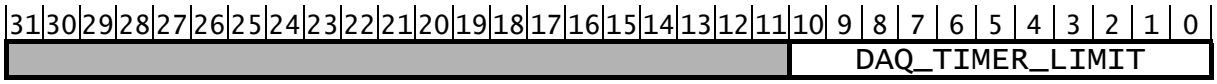
Register STEP_TIMER_LIMIT (RW) – see chapter 4.3.2.

STEP_TIMER_COUNT (000CH)



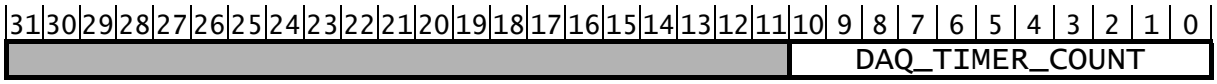
Register STEP_TIMER_COUNT (RO) – see chapter 4.3.2.

DAQ_TIMER_LIMIT (000DH)



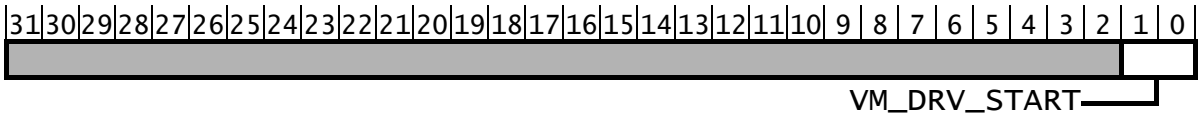
Register DAQ_TIMER_LIMIT (RW) – see chapter 10.2.3.

DAQ_TIMER_COUNT (000EH)



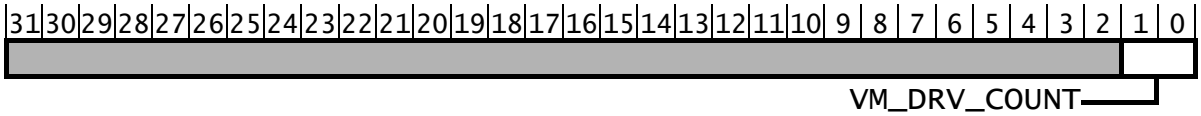
Register DAQ_TIMER_COUNT (RW) – see chapter 10.2.3.

VM_DRV_START (000FH)



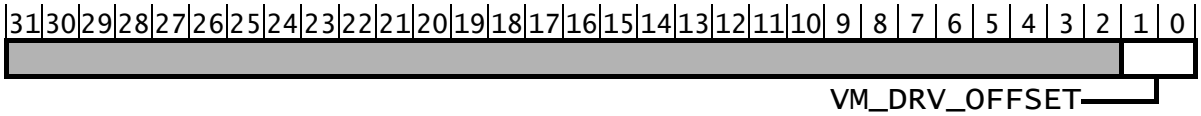
Register VM_DRV_START (RW) – see chapter 7.2.2.

VM_DRV_COUNT (0010H)



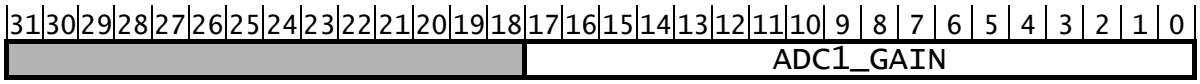
Register VM_DRV_COUNT (RW) – see chapter 7.2.2.

VM_DRV_OFFSET (0011H)



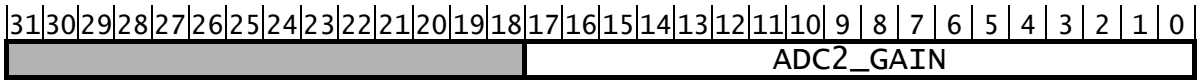
Register VM_DRV_OFFSET (RW) – see chapter 8.2.

ADC1_GAIN (0012H)



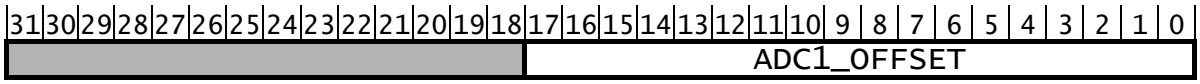
Register ADC1_GAIN (RW) – see chapter 5.2.

ADC2_GAIN (0013H)



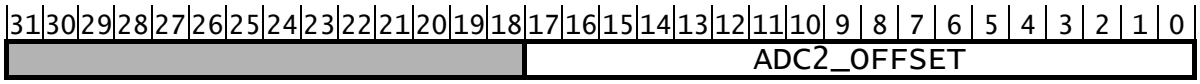
Register ADC2_GAIN (RW) – see chapter 5.2.

ADC1_OFFSET (0014H)



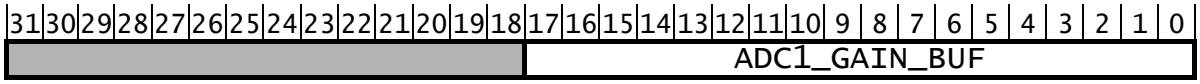
Register ADC1_OFFSET (RW) – see chapter 5.2.

ADC2_OFFSET (0015H)



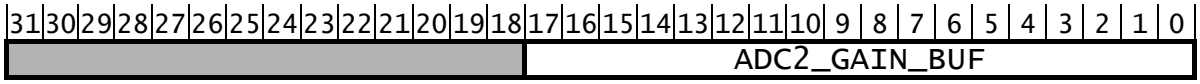
Register ADC2_OFFSET (RW) – see chapter 5.2.

ADC1_GAIN_BUF (0016H)



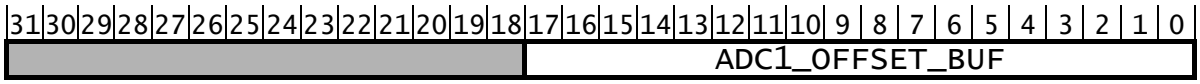
Register ADC1_GAIN_BUF (RW) – see chapter 5.2.

ADC2_GAIN_BUF (0017H)



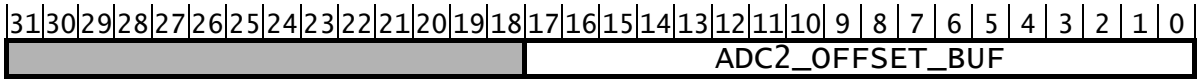
Register ADC2_GAIN_BUF (RW) – see chapter 5.2.

ADC1_OFFSET_BUF (0018H)



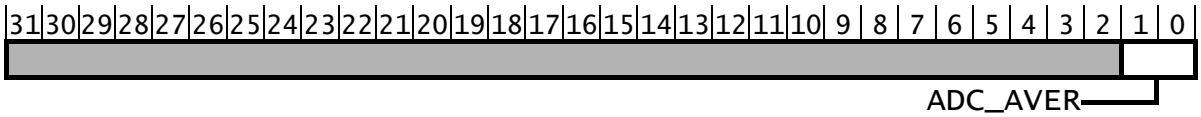
Register ADC1_OFFSET_BUF (RW) – see chapter 5.2.

ADC2_OFFSET_BUF (0019H)



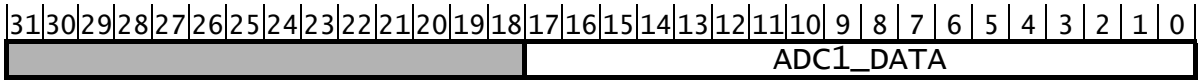
Register ADC2_OFFSET_BUF (RW) – see chapter 5.2.

ADC_AVER (001AH)



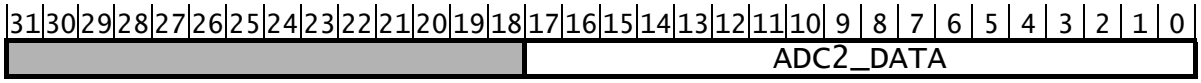
Register ADC_AVER (RW) – see chapter 5.2.

ADC1_DATA (001BH)



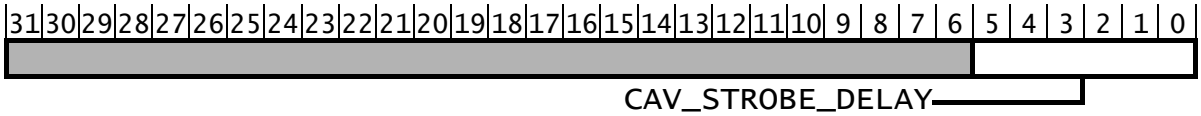
Register ADC1_DATA (RO) – see chapter 5.2.

ADC2_DATA (001CH)



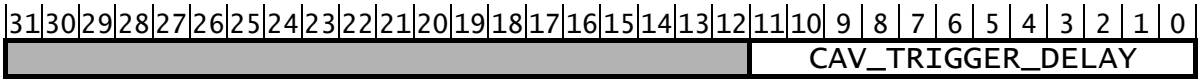
Register ADC2_DATA (RO) – see chapter 5.2.

CAV_STROBE_DELAY (001DH)



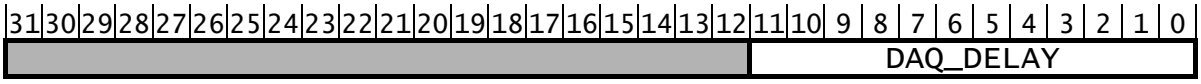
Register CAV_STROBE_DELAY (RW) – see chapter 4.3.3.

CAV_TRIGER_DELAY (001EH)



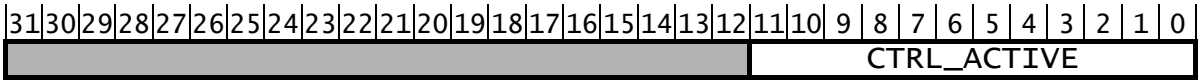
Register CAV_TRIGGER_DELAY (RW) – see chapter 4.3.3.

DAQ_DELAY (001FH)



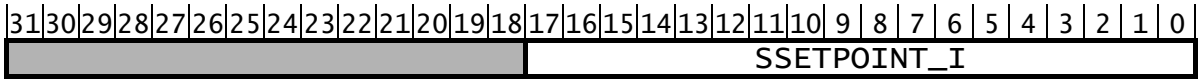
Register DAQ_DELAY (RW) – see chapter 4.3.3.

CTRL_ACTIVE (0020H)



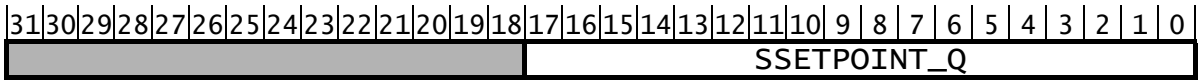
Register CTRL_ACTIVE (RW) – see chapter 9.2.

SSETPOINT_I (0021H)



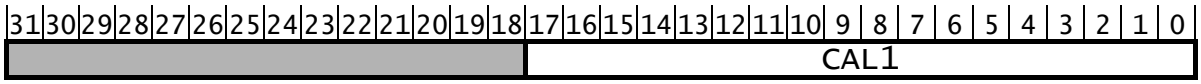
Register SSETPOINT_I (RW) – see chapter 7.2.5.

SSETPOINT_Q (0022H)



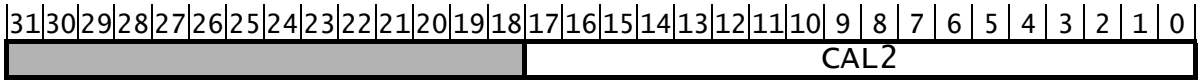
Register SSETPOINT_Q (RW) – see chapter 7.2.5.

CAL1 (0023H)



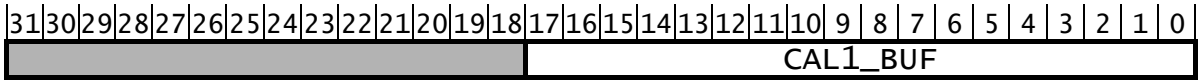
Register CAL1 (RW) – see chapter 7.2.4.

CAL2 (0024H)



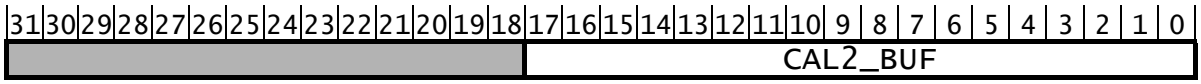
Register CAL2 (RW) – see chapter 7.2.4.

CAL1_BUF (0025H)



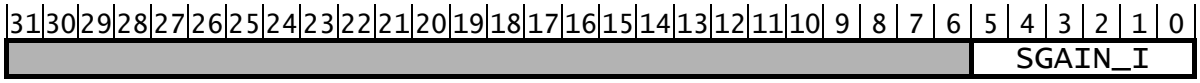
Register CAL1_BUF (RW) – see chapter 7.2.4.

CAL2_BUF (0026H)



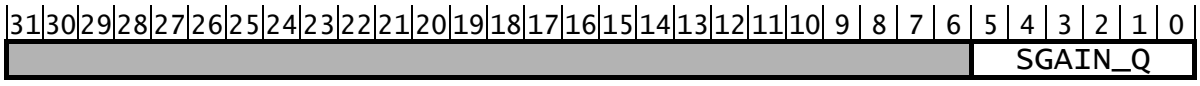
Register CAL2_BUF (RW) – see chapter 7.2.4.

SGAIN_I (0027H)



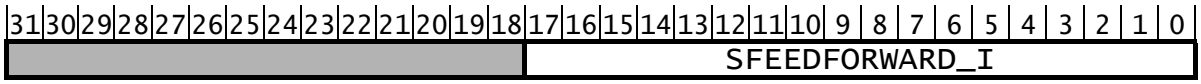
Register SGAIN_I (RW) – see chapter 7.2.5.

SGAIN_Q (0028H)



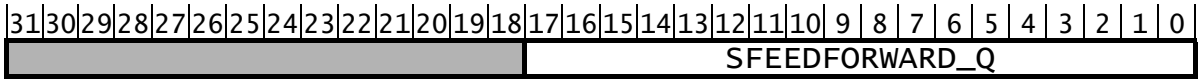
Register SGAIN_Q (RW) – see chapter 7.2.5.

SFEEDFORWARD_I (0029H)



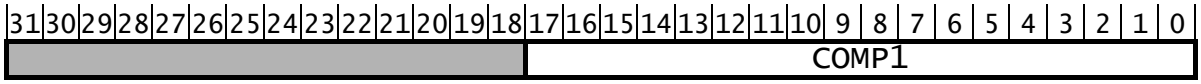
Register (RW) SFEEDFORWARD_I– see chapter 7.2.5.

SFEEDFORWARD_Q (002AH)



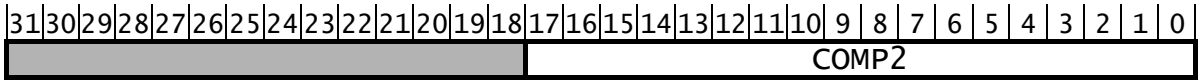
Register SFEEDFORWARD_Q (RW) – see chapter 7.2.5.

COMP1 (002BH)



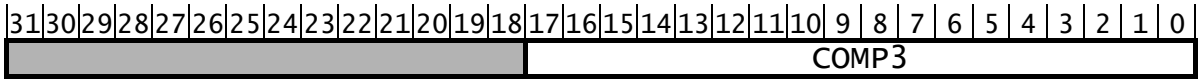
Register COMP1 (RW) –see chapter 7.2.5.

COMP2 (002CH)



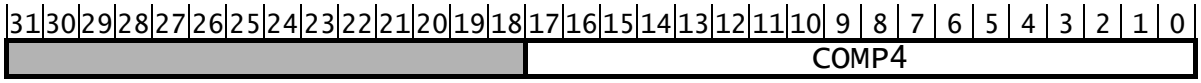
Register COMP2 (RW) – see chapter 7.2.5.

COMP3 (002DH)



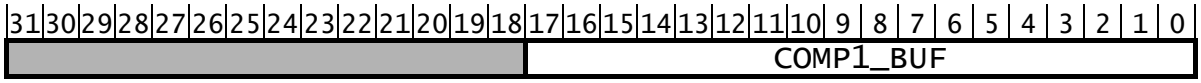
Register COMP3 (RW) – see chapter 7.2.5.

COMP4 (002EH)



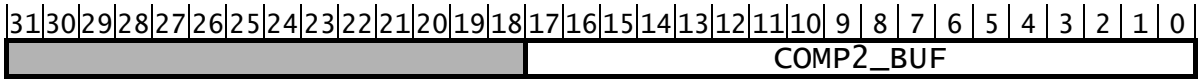
Register COMP4 (RW) – see chapter 7.2.5.

COMP1_BUF (002FH)



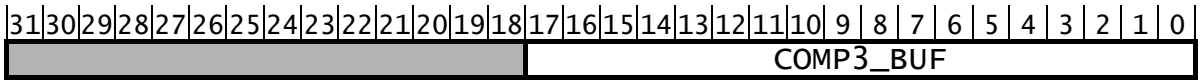
Register COMP1_BUF (RW) –see chapter 7.2.5.

COMP2_BUF (0030H)



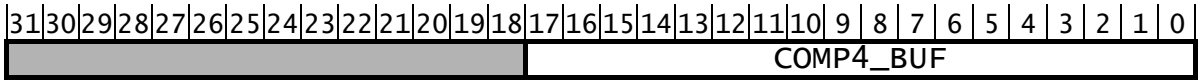
Register COMP2_BUF (RW) – see chapter 7.2.5.

COMP3_BUF (0031H)



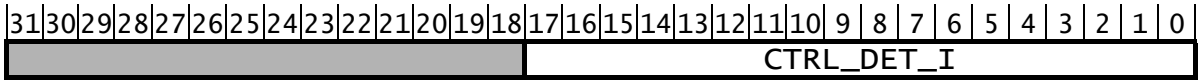
Register COMP3_BUF (RW) – see chapter 7.2.5.

COMP4_BUF (0032H)



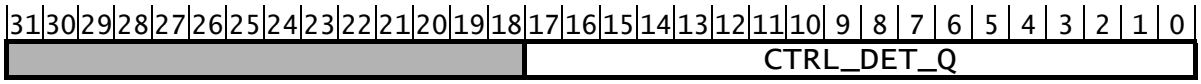
Register COMP4_BUF (RW) – see chapter 7.2.5.

CTRL_DET_I (0033H)



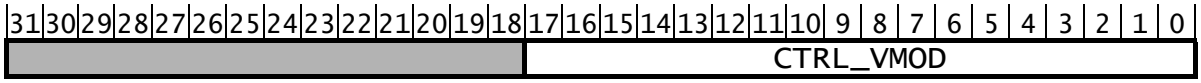
Register CTRL_DET_I (RW) – see chapter 9.2.

CTRL_DET_Q (0034H)



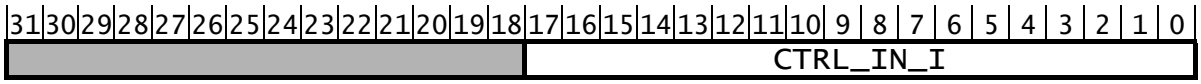
Register CTRL_DET_Q (RW) – see chapter 9.2.

CTRL_VMOD (0035H)



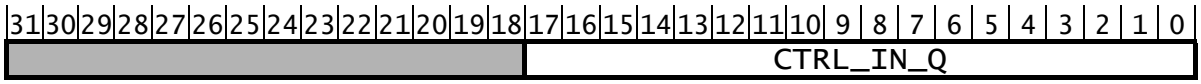
Register CTRL_VMOD (RW) – see chapter 9.2.

CTRL_IN_I (0036H)



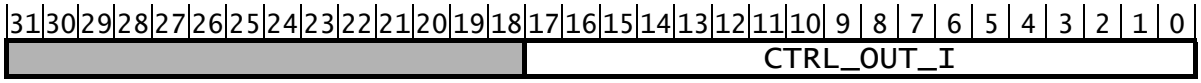
Register CTRL_IN_I (RW) – see chapter 9.2.

CTRL_IN_Q (0037H)



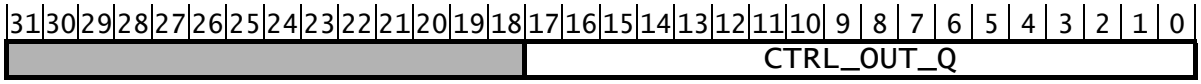
Register CTRL_IN_Q (RW) – see chapter 9.2.

CTRL_OUT_I (0038H)



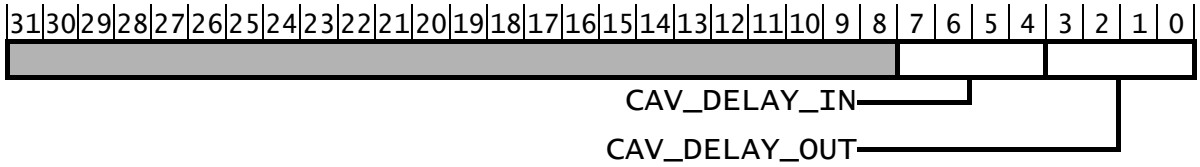
Register CTRL_OUT_I (RW) – see chapter 9.2.

CTRL_OUT_Q (0039H)



Register CTRL_OUT_Q (RW) – see chapter 9.2.

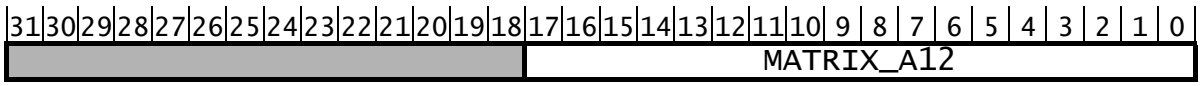
CAV_DELAY (003AH)



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

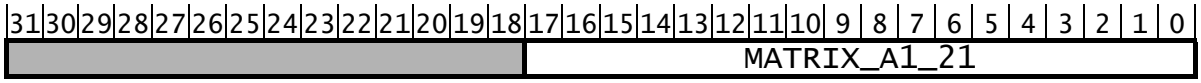
- CAV_DELAY_IN (RW): see chapter 8.2,
- CAV_DELAY_OUT (RW): see chapter 8.2,

MATRIX_A12 (003BH)



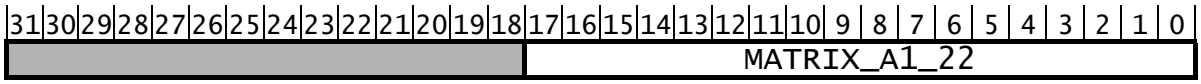
Register MATRIX_A12 (RW) – see chapter 7.2.4.

MATRIX_A1_21 (003CH)



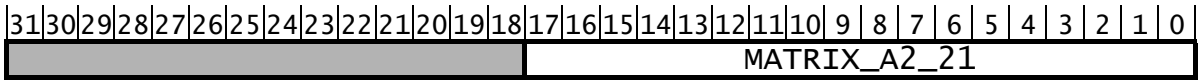
Register MATRIX_A1_21 (RW) – see chapter 7.2.4.

MATRIX_A1_22(003DH)



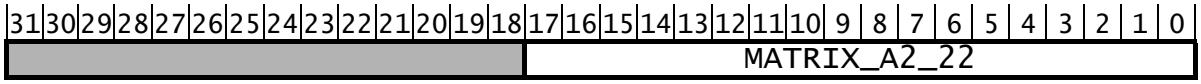
Register MATRIX_A1_22 (RW) – see chapter 7.2.4.

MATRIX_A2_21 (003EH)



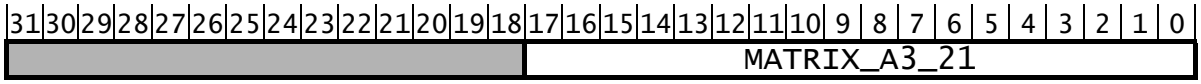
Register MATRIX_A2_21 (RW) – see chapter 7.2.4.

MATRIX_A2_22 (003FH)



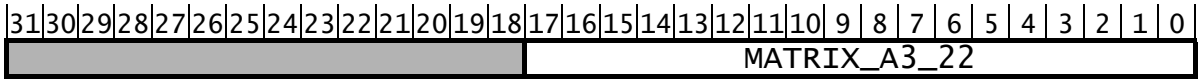
Register MATRIX_A2_22 (RW) – see chapter 7.2.4.

MATRIX_A3_21 (0040H)



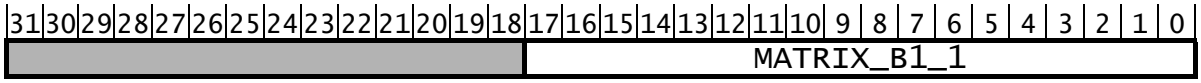
Register MATRIX_A3_21 (RW) – see chapter 7.2.4.

MATRIX_A3_22 (0041H)



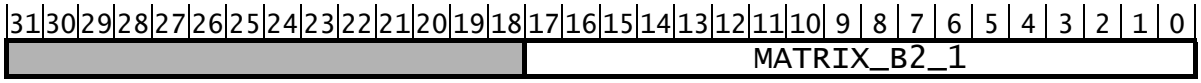
Register MATRIX_A3_22 (RW) – see chapter 7.2.4.

MATRIX_B1_1 (0042H)



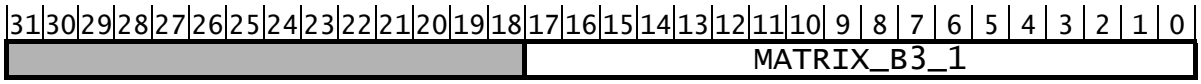
Register MATRIX_B1_1 (RW) – see chapter 7.2.4.

MATRIX_B2_1 (0043H)



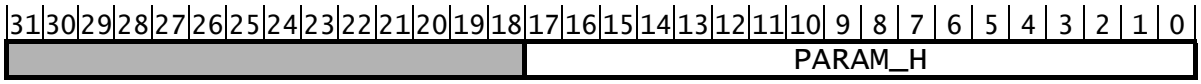
Register MATRIX_B2_1 (RW) – see chapter 7.2.4.

MATRIX_B3_1 (0044H)



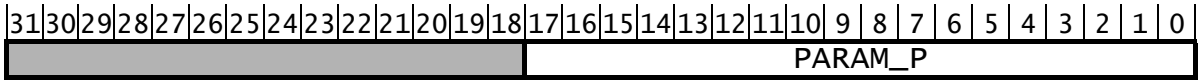
Register MATRIX_B3_1 (RW) – see chapter 7.2.4.

PARAM_H (0045H)



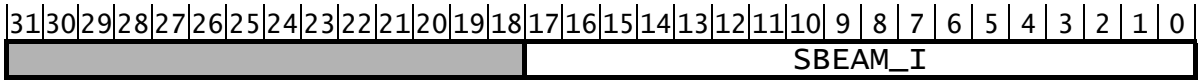
Register PARAM_H (RW) – see chapter 7.2.4.

PARAM_P (0046H)



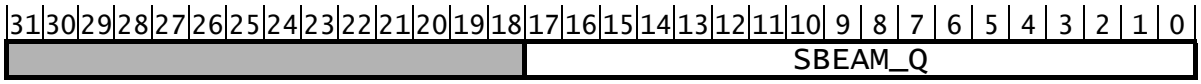
Register PARAM_P (RW) – see chapter 7.2.4.

SBEAM_I (0047H)



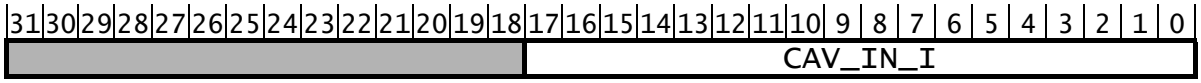
Register SBEAM_I (RW) – see chapter 7.2.4.

SBEAM_Q (0048H)



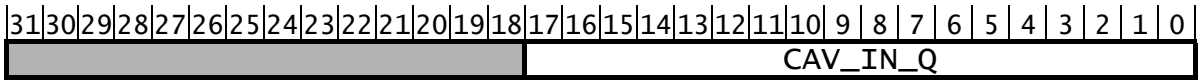
Register SBEAM_Q (RW) – see chapter 7.2.4.

CAV_IN_I (0049H)



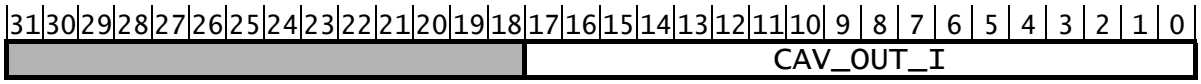
Register CAV_IN_I (RW) – see chapter 8.2.

CAV_IN_Q (004AH)



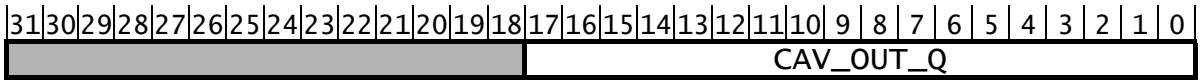
Register CAV_IN_Q (RW) – see chapter 8.2.

CAV_OUT_I (004BH)



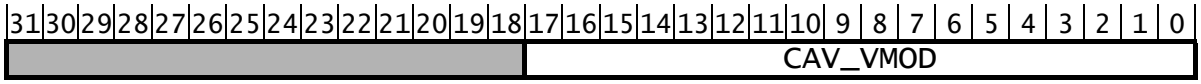
Register CAV_OUT_I (RO) – see chapter 8.2.

CAV_OUT_Q (004CH)



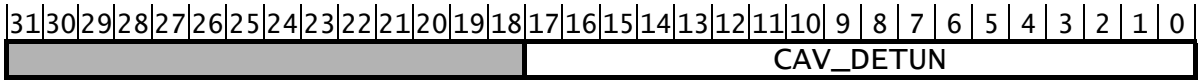
Register CAV_OUT_Q (RO) – see chapter 8.2.

CAV_VMOD (004DH)



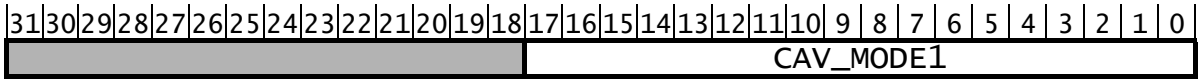
Register CAV_VMOD (RO) – see chapter 8.2.

CAV_DETUN (004EH)



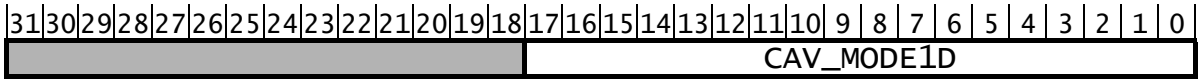
Register CAV_DETUN (RO) – see chapter 8.2.

CAV_MODE1 (004FH)



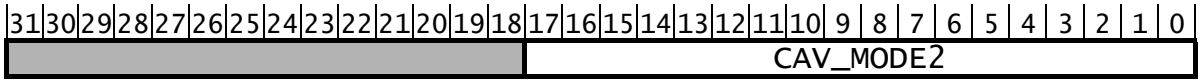
Register CAV_MODE1 (RO) – see chapter 8.2.

CAV_MODE1D (0050H)



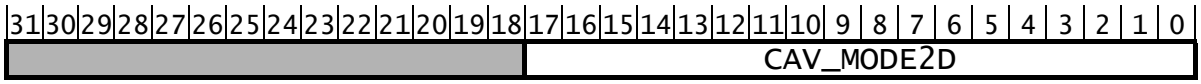
Register CAV_MODE1D (RO) – see chapter 8.2.

CAV_MODE2 (0051H)



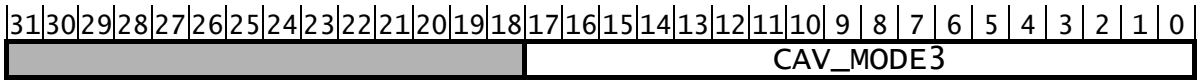
Register CAV_MODE2 (RO) – see chapter 8.2.

CAV_MODE2D (0052H)



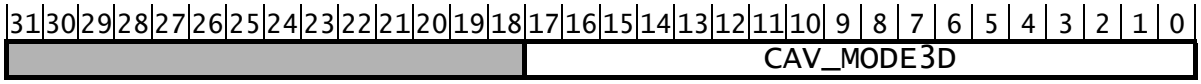
Register CAV_MODE2D (RO) – see chapter 8.2.

CAV_MODE3 (0053H)



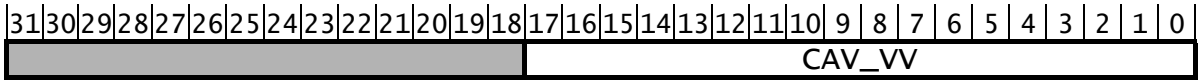
Register CAV_MODE3 (RO) – see chapter 8.2.

CAV_MODE3D (0054H)



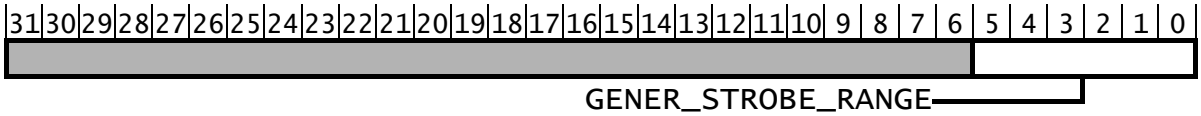
Register CAV_MODE3D (RO) – see chapter 8.2.

CAV_VV (0055H)



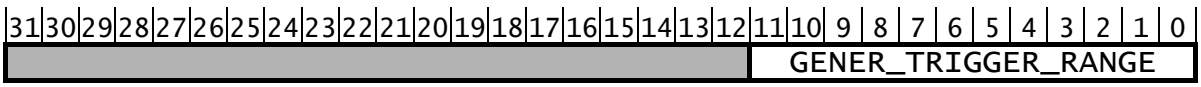
Register CAV_VV (RO) – see chapter 8.2.

GENER_STROBE_RANGE (0056H)



Register GENER_STROBE_RANGE (RW) – see chapter 4.3.1.

GENER_TRIGER_RANGE (0057H)



Register GENER_TRIGGER_RANGE (RW) – see chapter 4.3.1.

TSETPOINT_I (0800H-0FFFH)

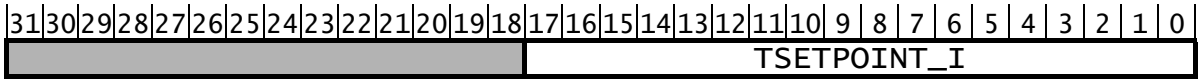


Table TSETPOINT_I (RW) – see chapter 7.2.5.

TSETPOINT_Q (1000H-17FFH)

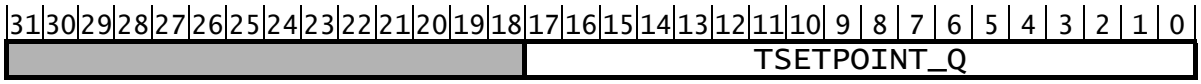


Table TSETPOINT_Q (RW) – see chapter 7.2.5.

TFEEDFORWARD_I (1800H-1FFFH)

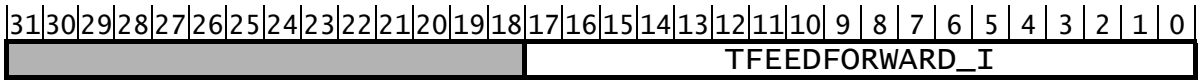


Table TFEEDFORWARD_I (RW) – see chapter 7.2.5.

TFEEDFORWARD_Q (2000H-27FFH)

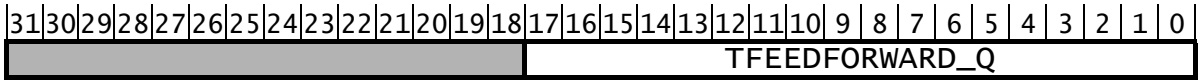


Table TFEEDFORWARD_Q (RW) – see chapter 7.2.5.

TGAIN_I (2800H-2FFFH)

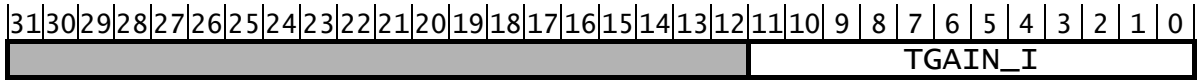


Table TGAIN_I (RW) – see chapter 7.2.5.

TGAIN_Q (3000H-37FFH)

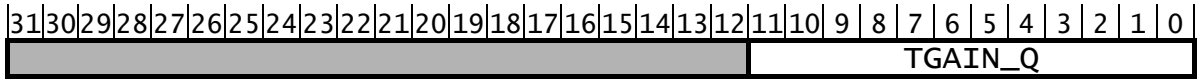


Table TGAIN_Q (RW) – see chapter 7.2.5.

TBEAM_I (3800H-3FFFH)

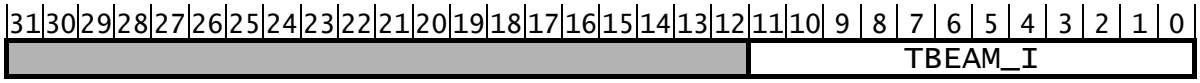


Table TBEAM_I (RW) – see chapter 7.2.4.

TBEAM_Q (4000H-47FFH)

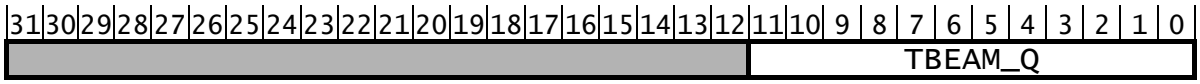


Table TBEAM_Q (RW) – see chapter 7.2.4.

DAQ1 (4800H-4FFFH)

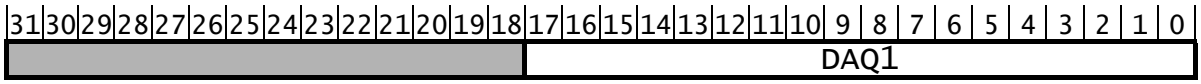


Table DAQ1 (RW) – see chapter 10.2.

DAQ2 (5000H-57FFH)

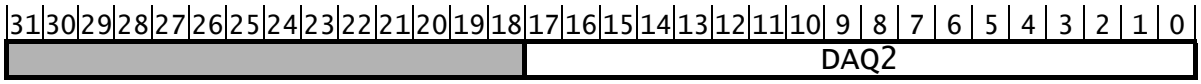


Table DAQ2 (RW) – see chapter 10.2.

DAQ3 (5800H-5FFFH)

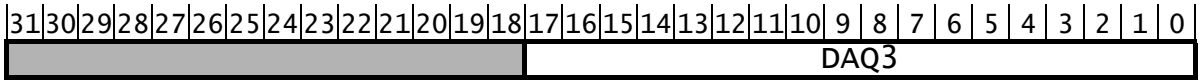


Table DAQ3 (RW) – see chapter 10.2.

DAQ4 (6000H-67FFH)

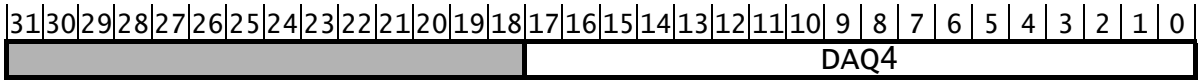


Table DAQ4 (RW) – see chapter 10.2.

13.2 I/O specification list by names

A VME INTERFACE

Realization of the basic physical communication layer with the FPGA chip positioned on the *XtremeDSP Development Kit* bases on the VMEbus standard [9]. The access of type „A24D32" was provided (i.e. 24 lines of address bus and 32 lines of data bus are used). The interface May work in the SLAVE (AM=39H) work mode as well as in MASTER (AM=3DH) work mode. The basic address of the board in the address space of VMEbus define the 4 oldest address bits (A23-A20). Application of the VMEbus interface stems form the requirements imposed by the DOOCS system [1]. The DOOCS is used for the accelerator control. The realized VME interface was presented in fig. 24.

The chips U5-U8 (74F1245) realize duplex data buffer. The VMEbus standard uses 0-5V signals. The FPGA chip from ACEX series accepts signals of 5V but works for I/O 3.3V. All the logic functionalities and direct control of the VME protocol is done by the FPGA ACEX chip.

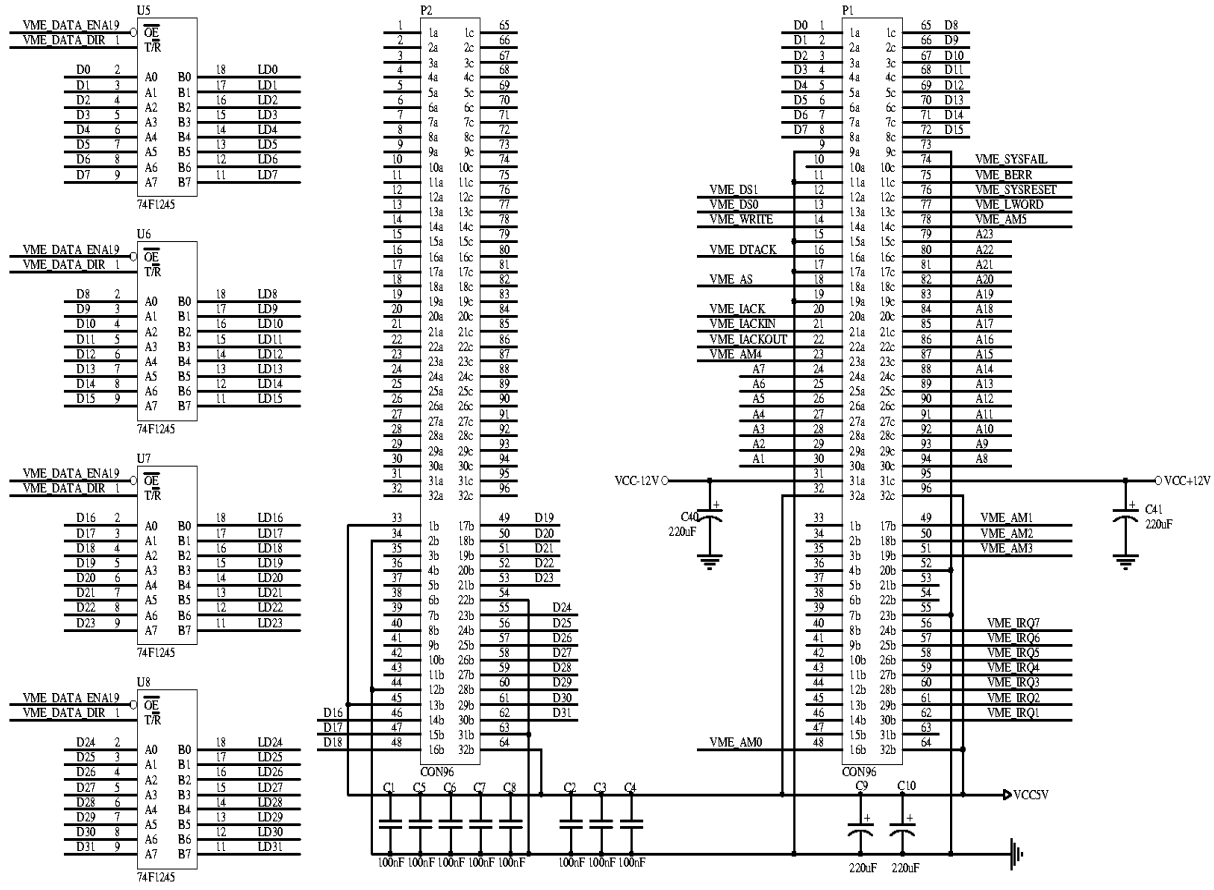


Fig. 24. VME interface schematic for FPGA Altera ACEX

The addresses, data and necessary control signals from the *VME-bus* are transmitted from the ACEX chip to the VIRTEX in the form of complex data packets. Packets contain a sequence of *6 bytes*. The two first bytes contain the *16-bit* address. The four following bytes contain the value of data word (respectively read or write). Next, a sequence of I/O operations is performed in the Internal Interface standard [8]. An example of the signal sequence for the write operation for *32-bit* data *12345678(hex)* under the *16-bit* address *0001(hex)*, for basic VME address *A00000(hex)* was presented in fig. 25.

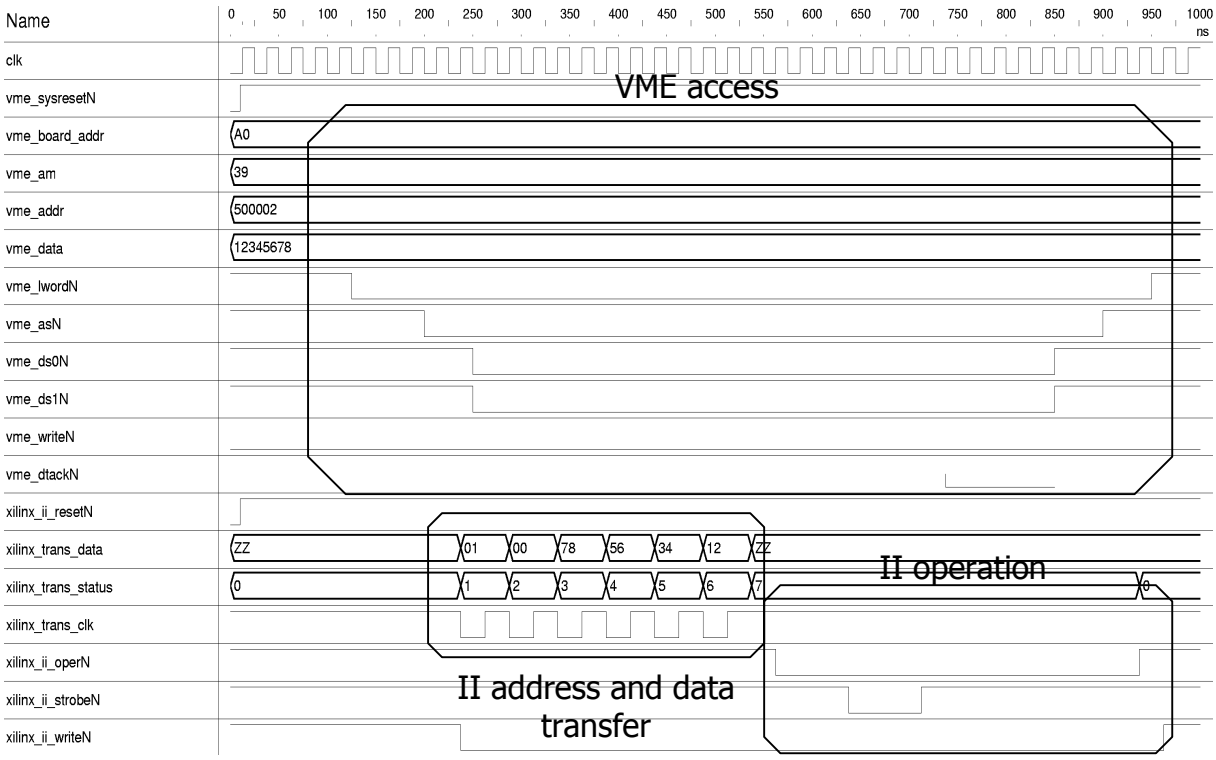


Fig. 25. Exemplary signal sequence for data transmission (write operation) via the VME interface to the II standard implemented in the FPGA Altera ACEX.

B EPP INTERFACE

The alternative way of physical communication layer between the FPGA chip situated on *XtremeDSP Development Kit* and a PC computer was realized using the EPP (Enhanced Parallel Port) transmission standard ver. 1.7 [12]. The applied configuration allows to obtain up to 500kB/s of data rate. The hardware interface to the FPGA ACEX was presented in fig. 26.

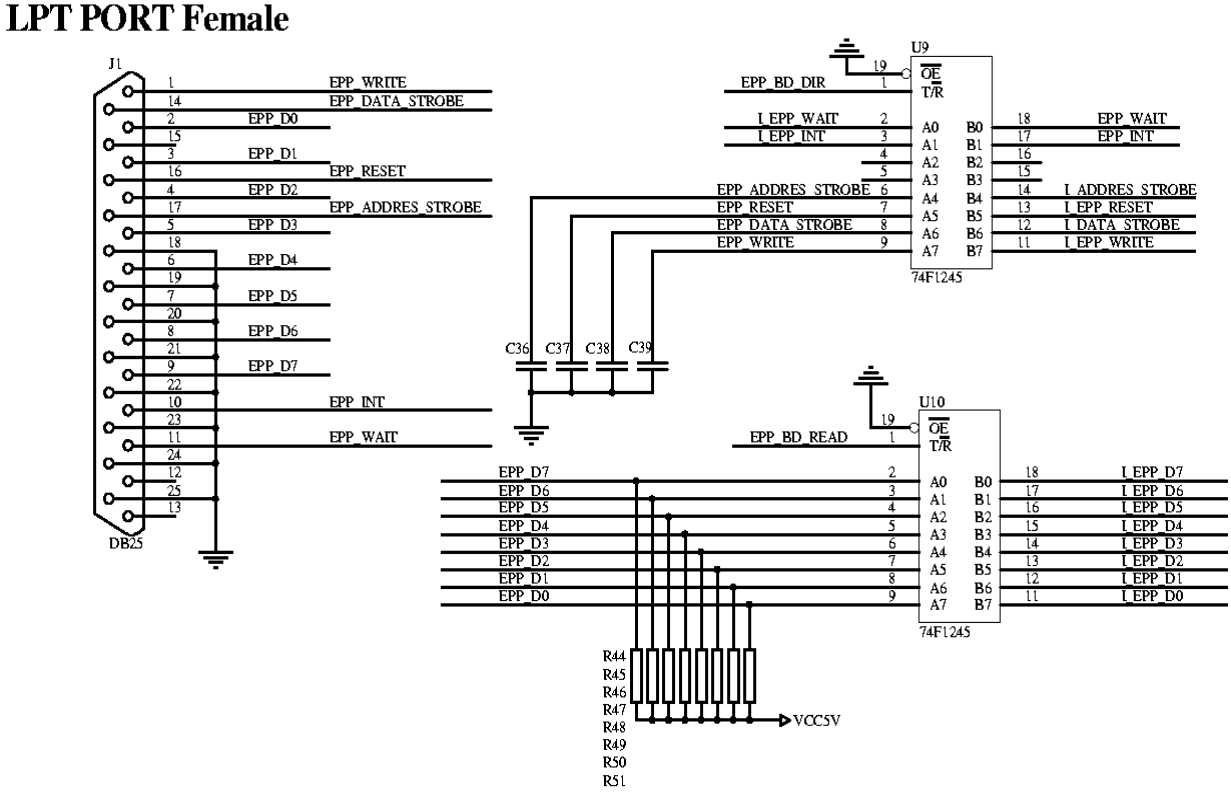


Fig. 26. VME interface schematic for FPGA Altera ACEX

The U10 (74LVT2245) chip realizes duplex data buffer. The EPP standard uses 0-5V signals. The ACEX chip accepts signal 0-5V but Works for I/O signals of 3,3V. The chip U9 (74F1245) is a buffer for control signals. All logical functions and direct control of the EPP protocol is performed in the ACEX.

The transmission standard bases on sending of 8-bit words or as data (active signal DATA STROBE) or as addresses (active signal ADDRESS STROBE). Sending of more complex packets of information to the address space of the FPGA is realized as a sequence of *8 bytes*. Two first bytes carry the information about the *16-bit* address, four following bytes contain the value of the data word (respectively write and read), next byte is a checksum, the last byte returns the state word of the realized transmission (among others, acknowledgement of the realized transmission by the FPGA chip). An example of the write sequence for *32-bit* data of the value *42372C21(hex)* under the *16-bit* address *160B(hex)* was presented in figure 27.

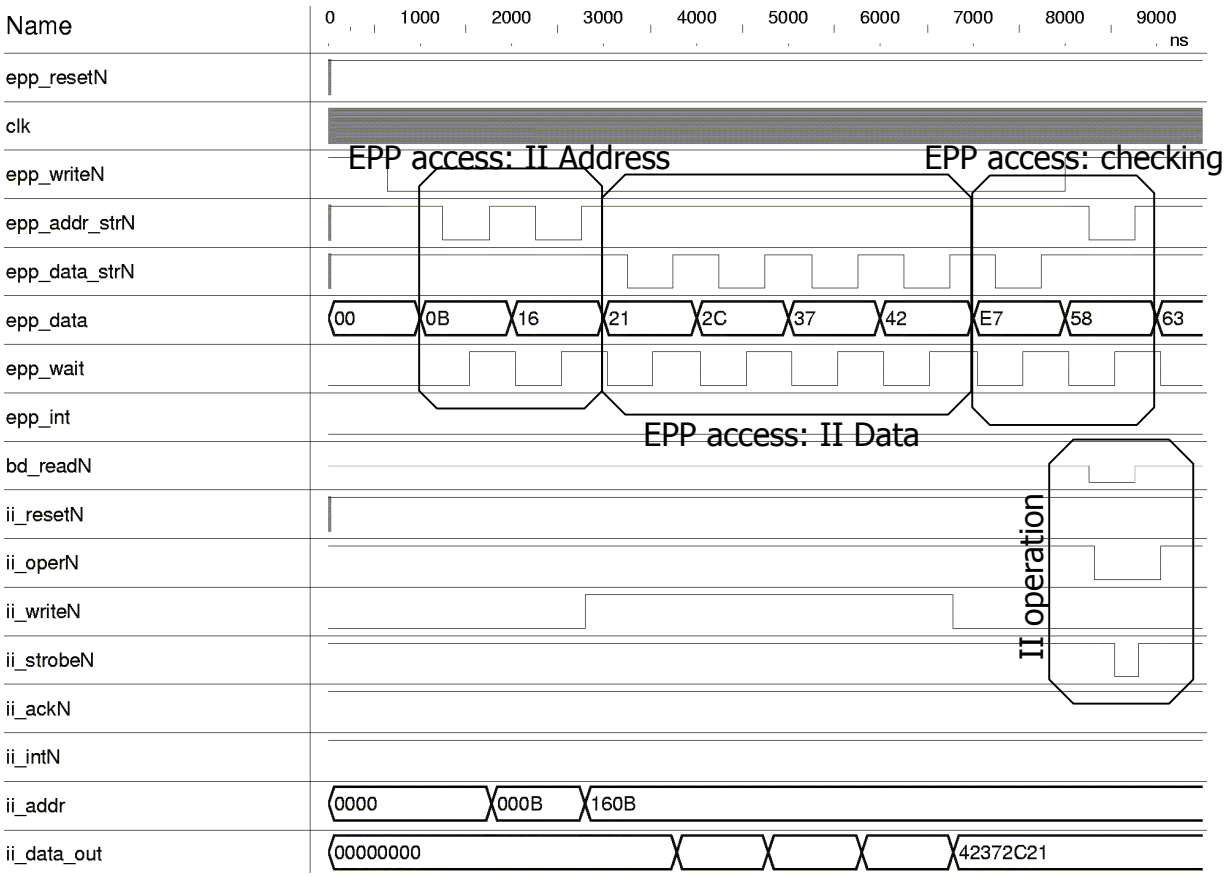


Fig. 27. Exemplary signal sequence for data transmission (write operation) via the EPP interface to the II standard implemented in the FPGA Altera ACEX.

C BenONE Overview

The BenONE - PCI [11] is a PCI, single slot DIME-II motherboard. It is classed as an entry-level motherboard and is capable of hosting a single width DIME-II module. The board has no FPGA resources available to the user; all resources are addressed on an attached module. It does however have the capability of a secondary connection to a host PC, for example USB (primary connection being PCI). This is achieved by the addition of an IO module on the motherboard. Another feature of the BenONE – PCI is that it can be used in standalone solutions using Compact Flash technology. The Compact Flash is a specific option that is not included as standard in the XtremeDSP Development Kit. The BenONE also performs housekeeping functions of the Programmable Power Supplies and PCI bus. Finally, connection to further Nallatech motherboards and modules is made possible by the inclusion of a pin header connection direct to the module site.

The key features of the BenONE - PCI are:

- PCI/Control Xilinx® Spartan-II FPGA, pre-configured with PCI/Control Firmware,
- Single DIME-II module expansion slot,
- 32 bit/33MHz PCI interface with expansion to 64bit/33Mhz via firmware upgrade,
- Two on-board clock nets,
- 2 Programmable clock sources,
- 1 Fixed Oscillator socket,
- Status LEDs,
- JTAG configuration headers,
- User selectable pin headers,
- Fixed or fully programmable power supplies,
- Nallatech FUSE Software for FPGA configuration over PCI,
- Nallatech FUSE Software Library for board interfacing & control,
- USB 1.1 I/O Module interface,
- Battery Backup for Virtex-II® Encryption Keys,
- Compact Flash using Xilinx® System Ace chipset,
- External oscillator input via mini coax connector.

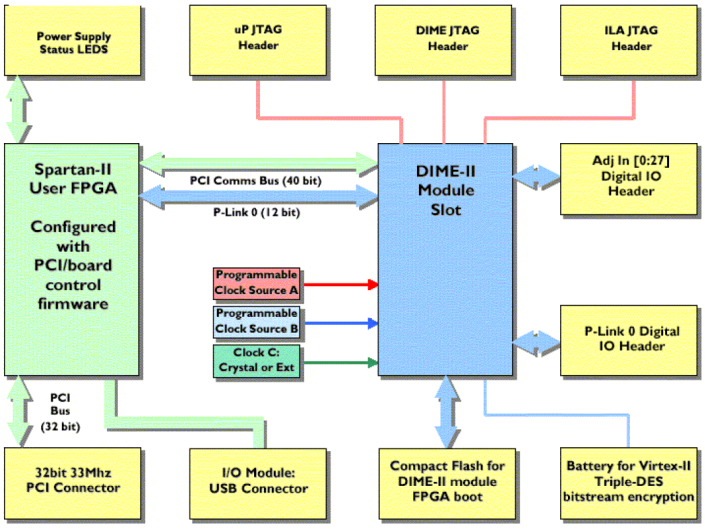


Fig. 28. BenONE - PCI Functional Diagram

D BenADDA Overview

The BenADDA DIME-II [11] module provides high-speed digital-to-analogue and analogue-to-digital conversion capability. As part of the scalable DIME-II family, the BenADDA can be easily integrated into systems, through the range of available DIME-II motherboards and associated software/firmware.

The module contains two high-speed ADC and two high-speed DAC channels, which allow for flexible, high-resolution data conversion for both baseband and IF applications. Key to the BenADDA's performance is the on-board Xilinx® Virtex™-II FPGA which provides you with a powerful data processing resource. Some of the main application areas for the BenADDA include mobile communications systems, infrared imaging, wideband cable systems and multi-channel, multi-mode receivers.

The key features of the XtremeDSP Development Kit (for Virtex V3000)are:

- On-board Xilinx Virtex-II FPGA,
- Various FPGA device packages, sizes and speed grade options available,
- Compatible with Nallatech's FUSE™ reconfigurable computing operating system,
- Two independent analogue capture channels,
- 2x 14-Bit ADC Resolution, up to 105MSPS per channel sampling rate,
- Two independent channels to extract analogue data,
- 2x 14-Bit DAC Resolution, up to 160MSPS per channel sampling rate,
- 8MB of ZBT SRAM memory, in two independent banks,
- Nallatech ZBT SRAM interfacing IP Core available,
- Multiple Clocking Options: Internal & External,
- Example designs and source code included,
- Status LEDs.

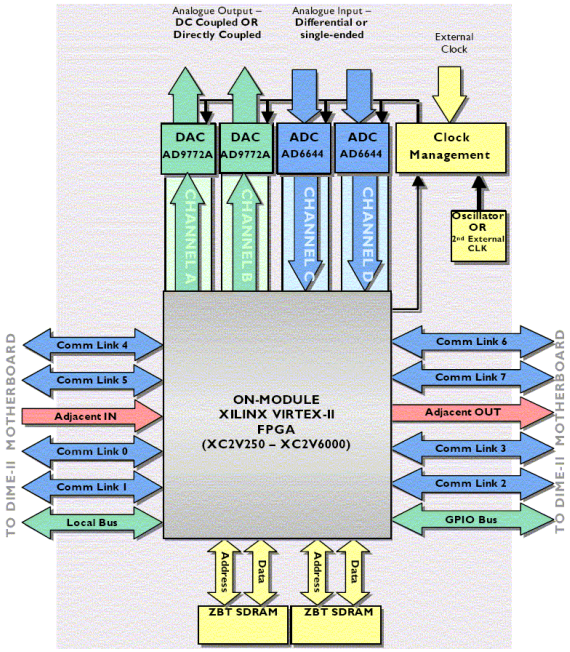


Fig. 29. BenADDA Functional Diagram

E Exemplary scope pictures of SIMCON system outputs

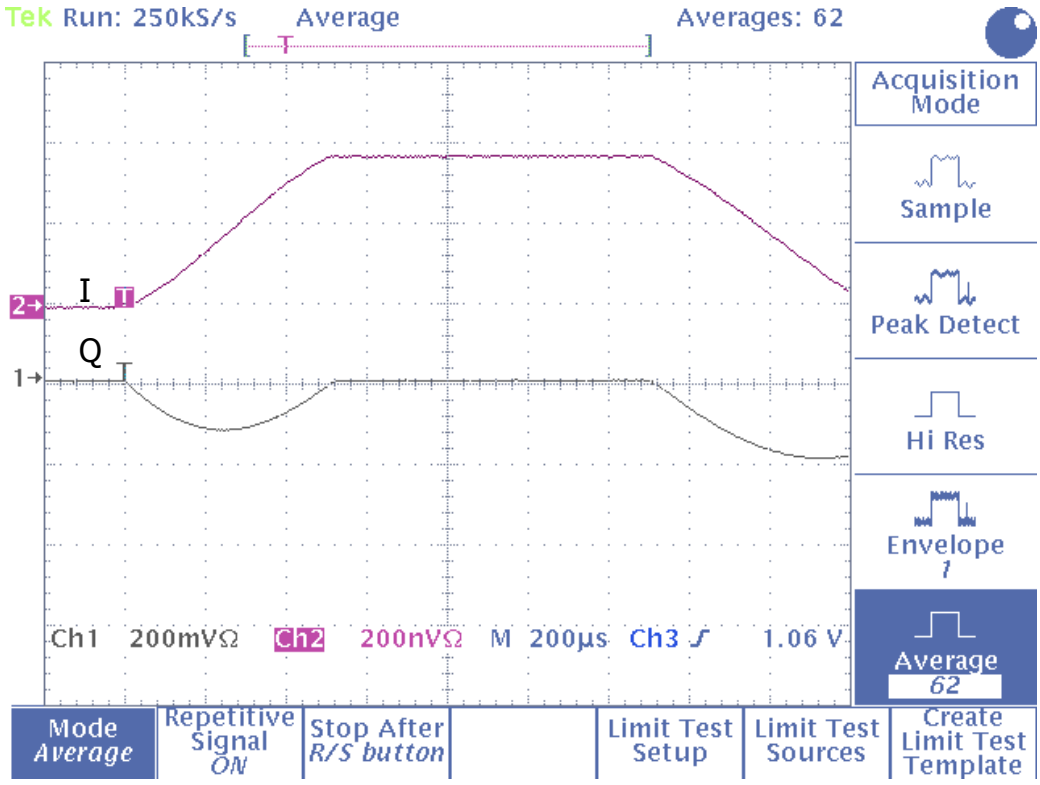


Fig. 30. I and Q outputs of cavity simulator driven by feedback and supported by feedforward

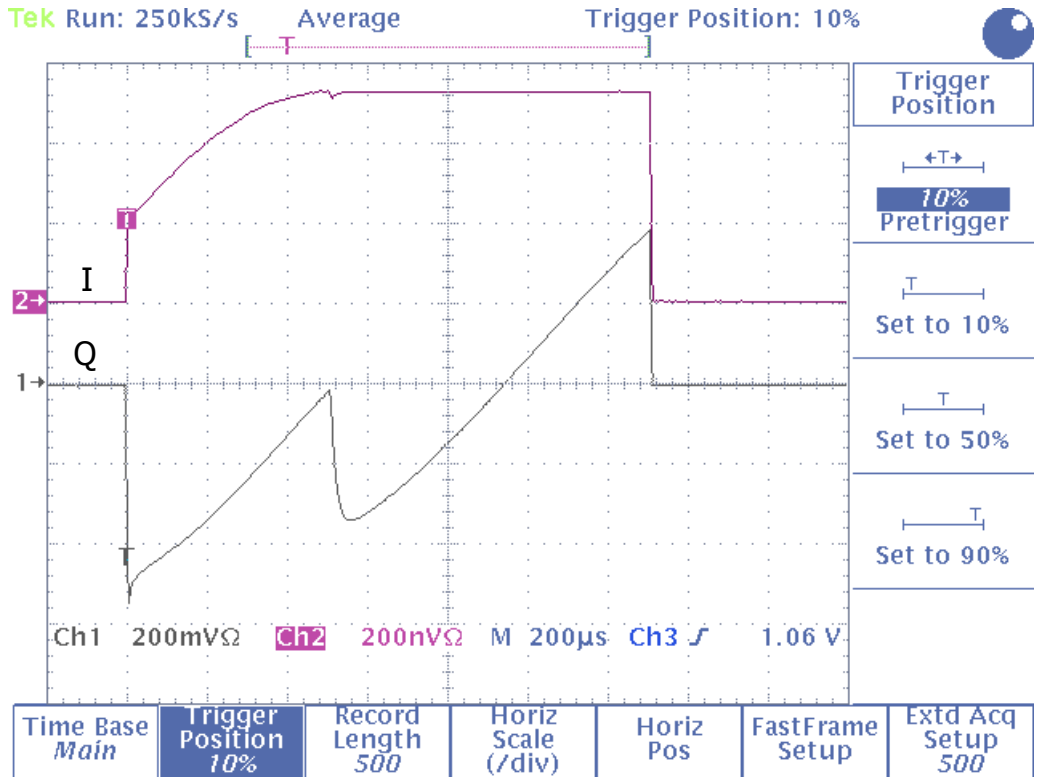


Fig. 31. I and Q outputs of cavity controller operated in feedback and feedforward mode

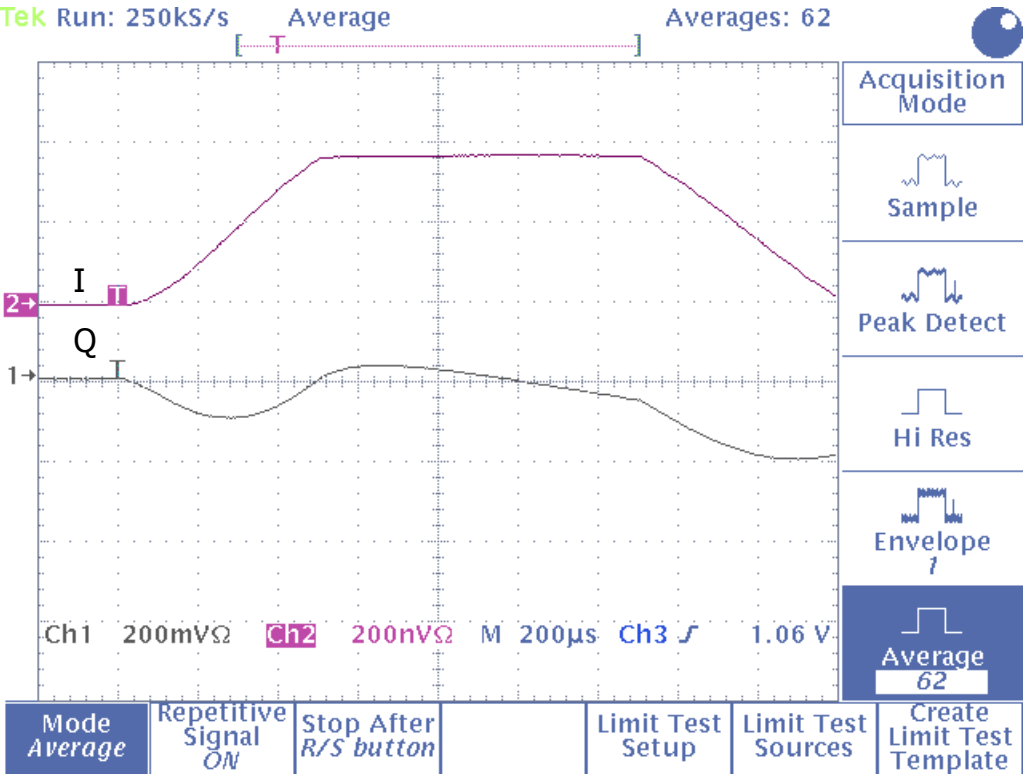


Fig. 32. I and Q outputs of cavity simulator driven by low gain feedback.

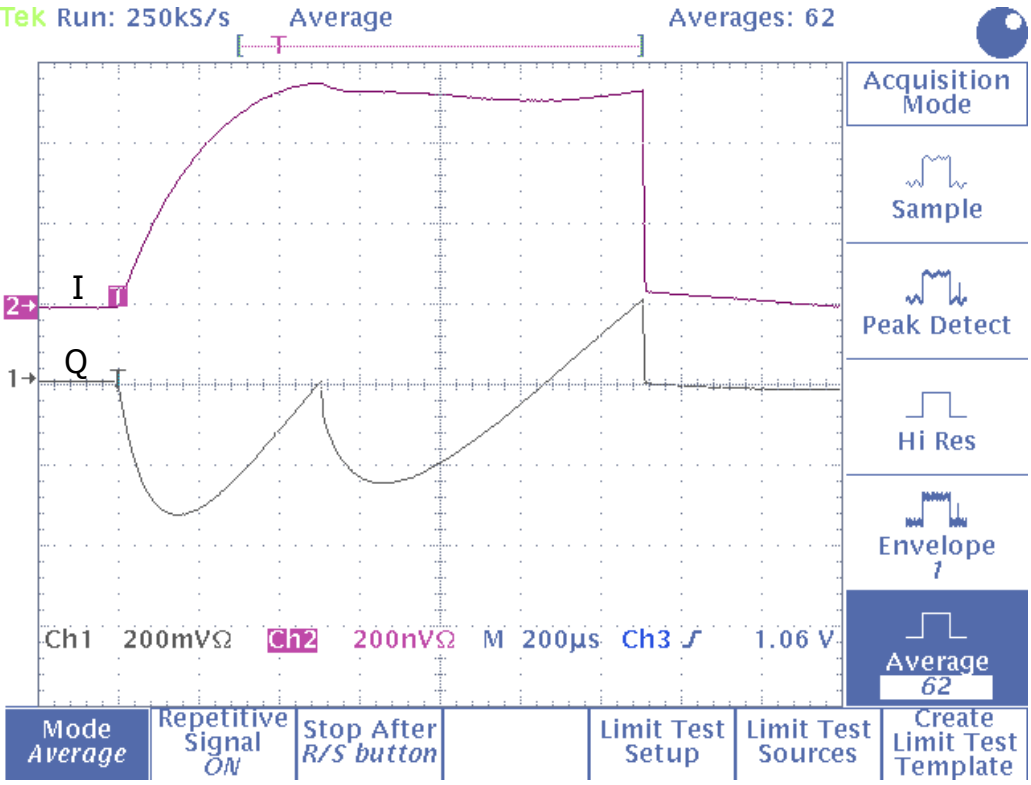


Fig. 33. I and Q outputs of cavity controller operated in low gain feedback mode.

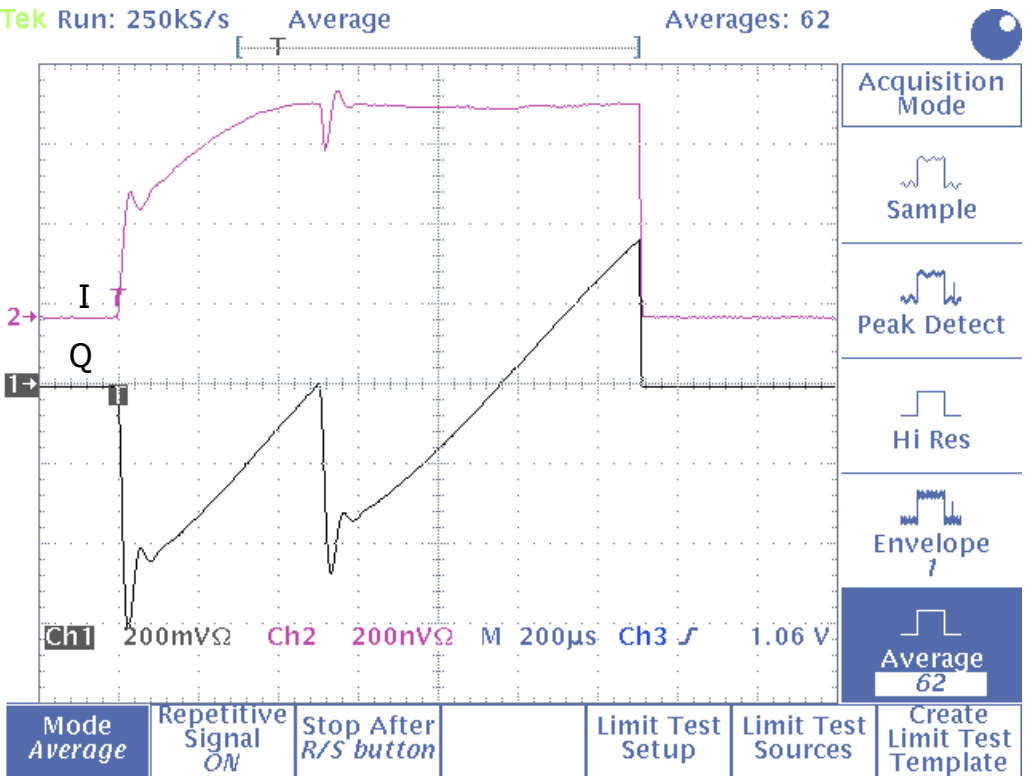


Fig. 34. I and Q outputs of cavity controller operated in delay loop condition.

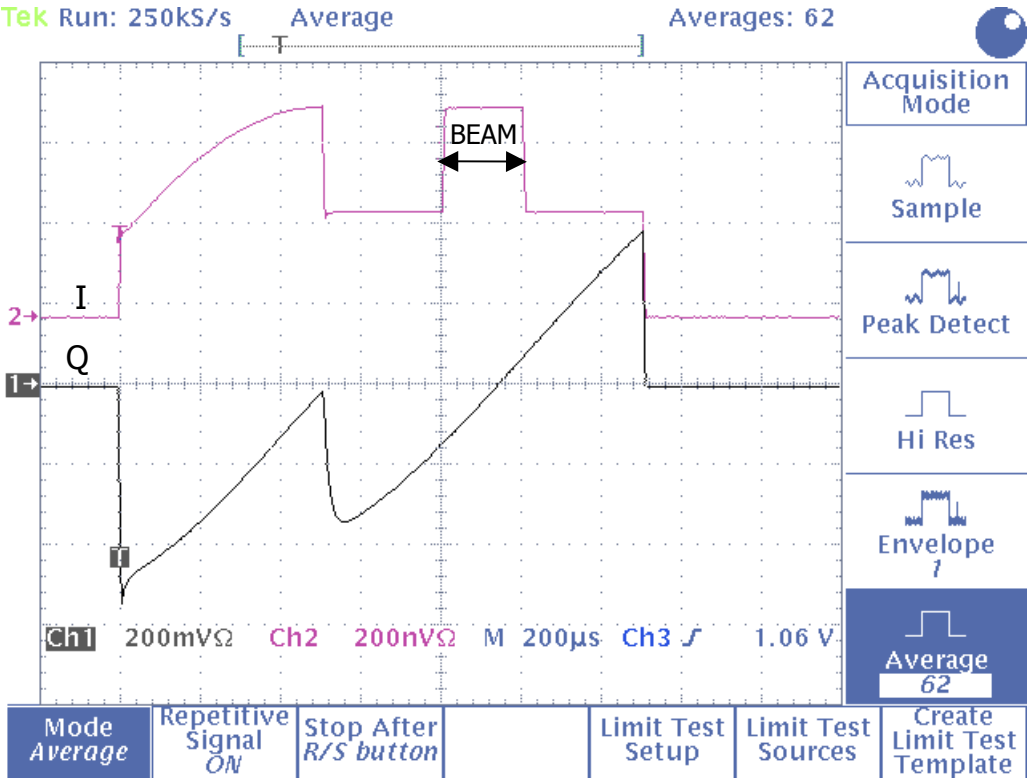


Fig. 35. I and Q outputs of cavity controller with the beam switched on during the flattop.

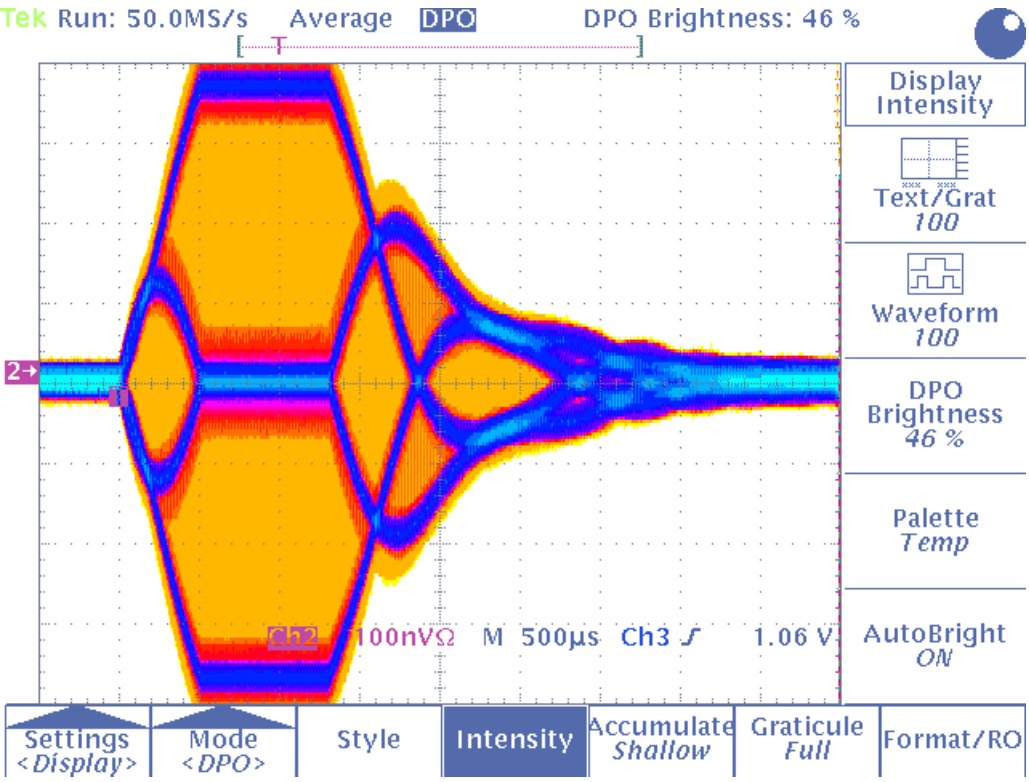


Fig. 36. Cavity simulator output for IF modulated signal (presented in DPO scope mode).

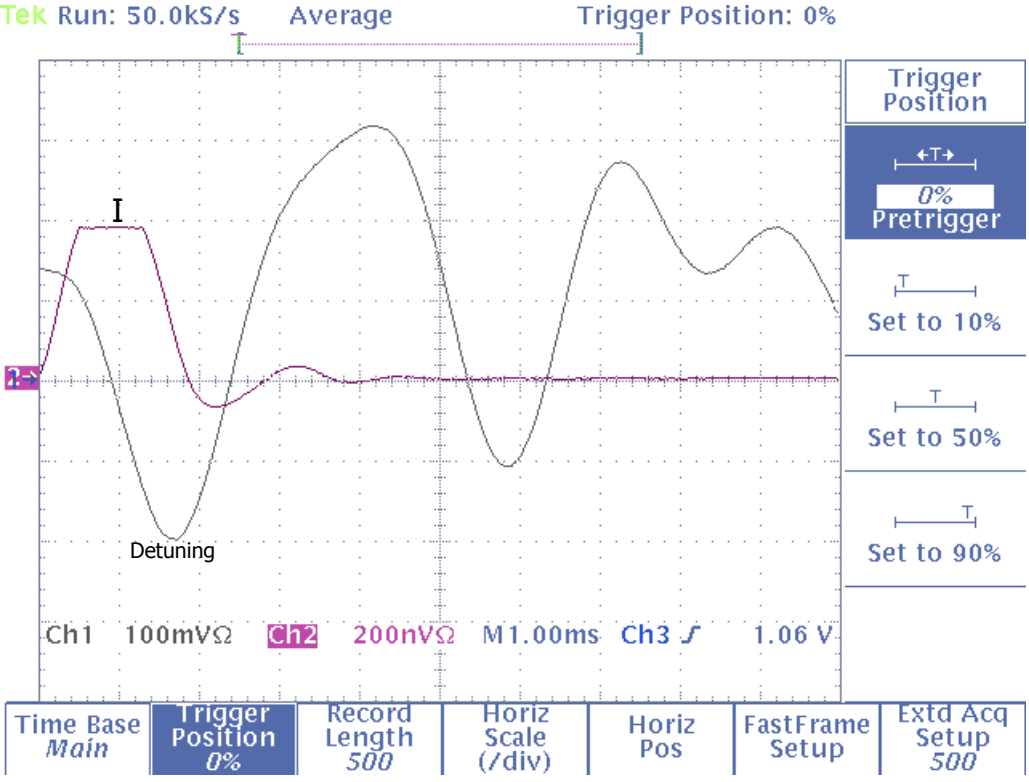


Fig. 37. Cavity simulator output for detuning signal related to I component of envelope.

F Exemplary results of CHECHIA real-time control

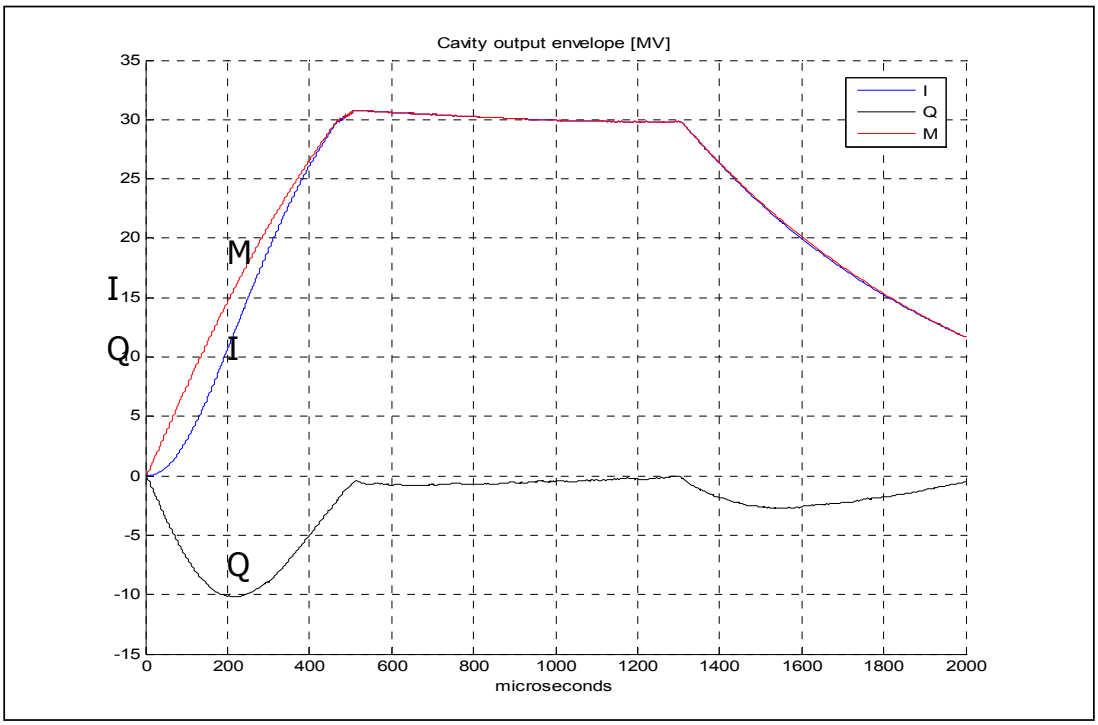


Fig. 38. Feed-forward cavity driving: selected readout of output envelope for 30 MV flattop level.

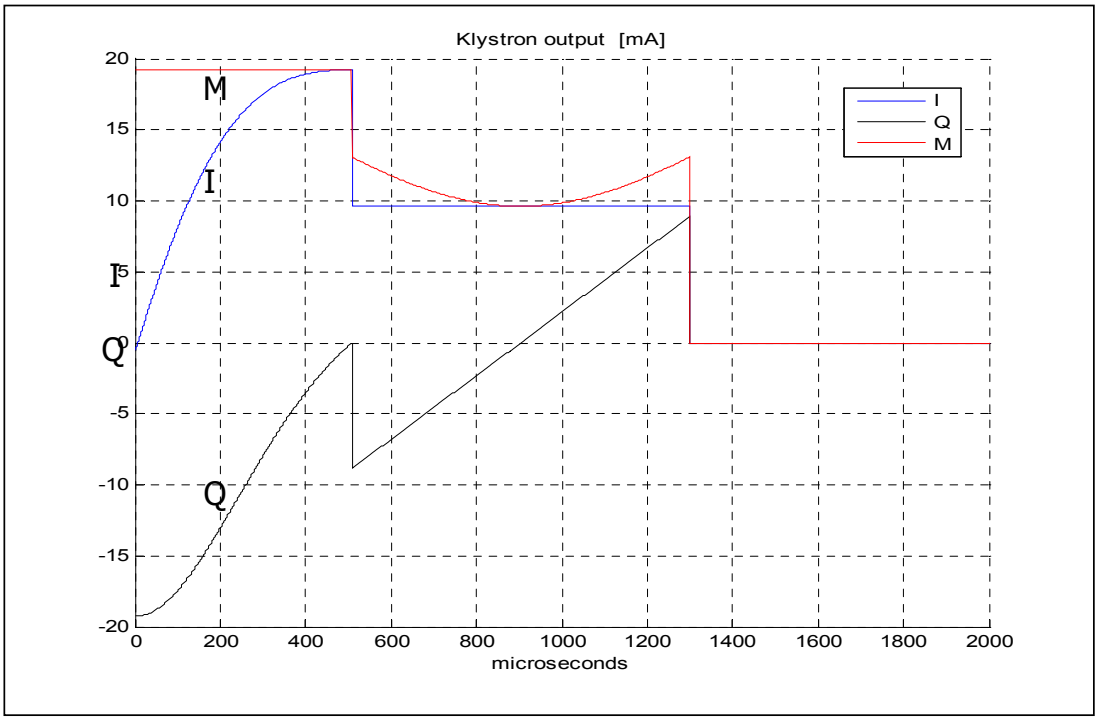


Fig. 39. Feed-forward cavity driving: selected readout of input envelope for 30 MV flattop level.

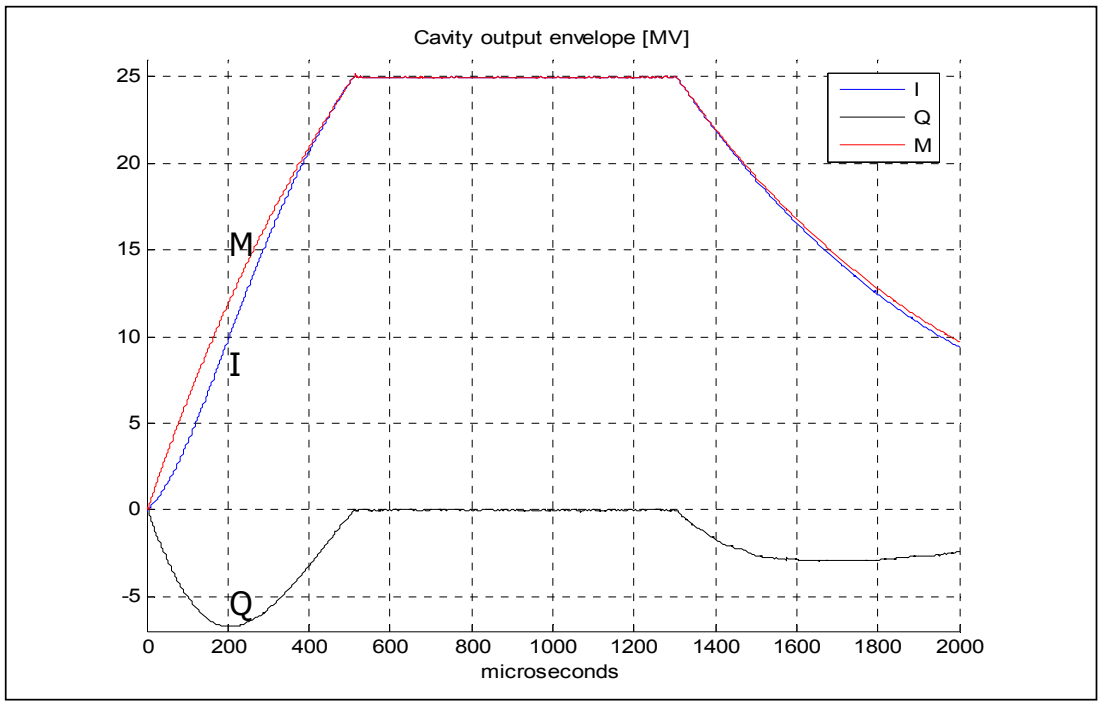


Fig. 40. Feed-forward with feedback cavity driving (gain = 100): selected readout of output envelope for 25 MV flattop level.

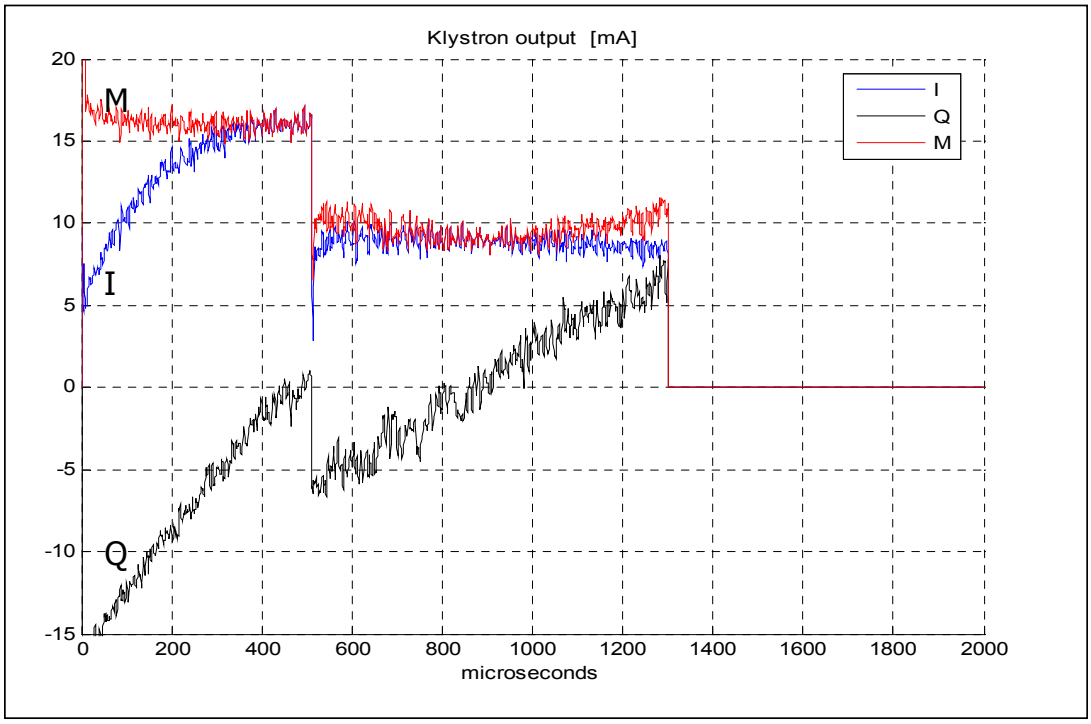


Fig. 41. Feed-forward with feedback cavity driving (gain = 100): selected readout of input envelope for 25 MV

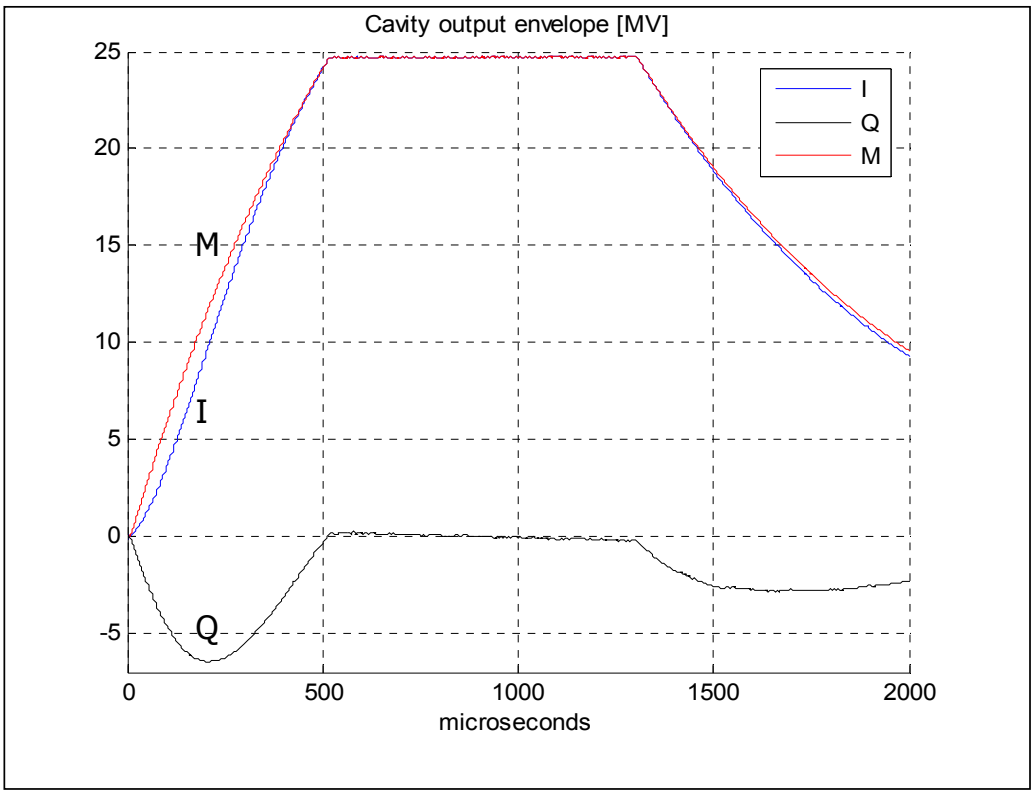


Fig. 42. Feedback cavity driving (gain = 100): selected readout of output envelope for 25 MV flattop level.

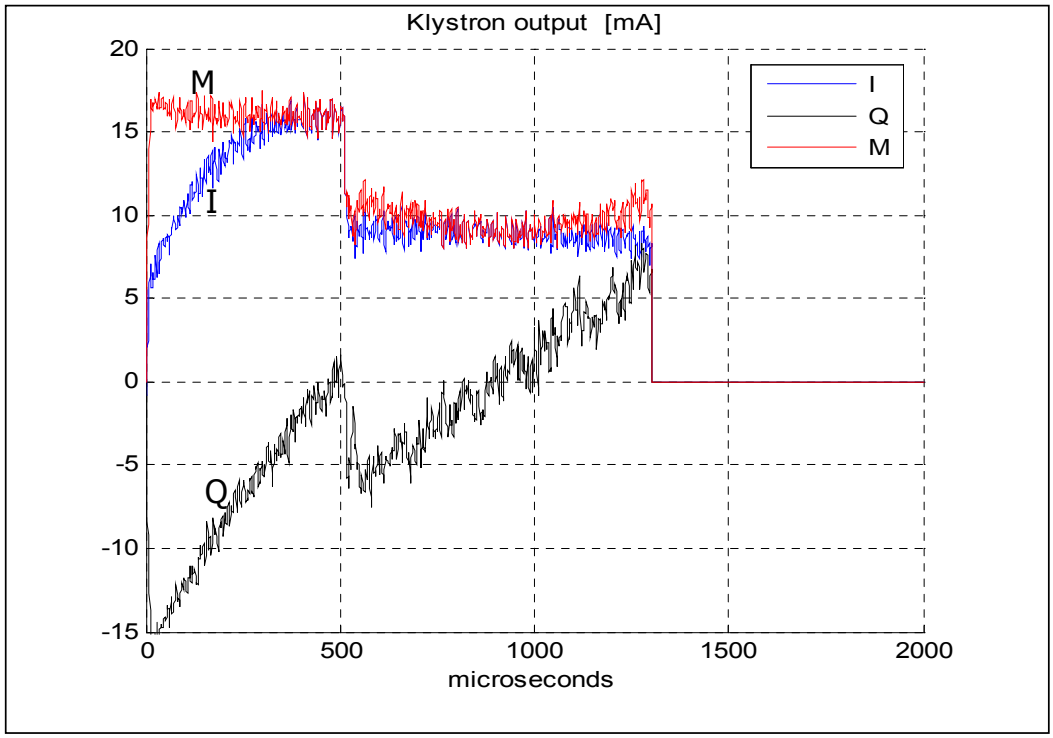


Fig. 43. Feedback cavity driving (gain = 100): selected readout of input envelope for 25 MV flattop level.

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 Cavity 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 Te
5. chnical Note, 2003-28
6. T.Czarski, R.S.Romaniuk, K.T.Pozniak S.Simrock “Cavity Control System Essential Modeling For TESLA Linear Accelerator”, TESLA Technical Note, 2003-08
7. T.Czarski, R.S.Romaniuk, K.T. Pozniak “Cavity Control System, Models Simulations For TESLA Linear Accelerator ”, TESLA Technical Note, 2003-09
8. 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
9. W. Petersen “The VMEbus Handbook”
10. http://www.xilinx.com/ [Xilinx Homepage]
11. http:// www.nallatech.com/ [Nallatech Homepage]
12. http:// http://www.beyondlogic.org/epp/ [EPP - Enhanced Parallel Port description]