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

SIMCON ver.2.1. Installation and Configuration Procedures

USER'S MANUAL

Waldemar Koprek, Piotr Pucyk, Tomasz Czarski,
Krzysztof T. Pozniak, Ryszard S. Romaniuk

wkoprek@ntmail.desy.de; ppucyk@ntmail.desy.de; tczarski@ntmail.desy.de;
pozniak@mail.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.

This document is intended to be used by end users and operators. It describes step by step how to install SIMCON in specific configuration, how and what software to copy to computer. There is described set of basic Matlab functions for developers of control algorithms.

This paper also contains brief description how to use Matlab function of one algorithm with its graphic user panels.

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.

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

This TESLA Report is in close relations with the following TESLA Reports published previously: 2005-05, 2005-02, 2004-10. Together, these Reports make a full SIMCON manual.

1. Introduction (SIMCON release policy)

The SIMCON hardware and software ver.2.1. is designed to control and simulate a single, TESLA technology based, superconducting cavity. The difference with ver.1.1. is that it can be used with a standard PC, via the LPT interface. This approach opens a path to control the system using standard web technologies.

The SIMCON hardware and software is a unity and is not provided by the designers separately or in parts. The reason is fast development of the next versions of the SIMCON system and not full compatibility between the versions. This kind of release policy is a must because there exist numerable versions of the test software for SIMCON, which are used internally for the development purposes. The designers are not able to support different products at the premises of different users of the SIMCON system. The system is in a developmental stage and not yet fully user friendly. In particular it has not got a fail safe mode of work.

Apart from this specific confinement the designers are eager to cooperate with potential SIMCON users and are ready to provide the whole system in a closed and fully supported form ready to be implemented at the user's premises.

2. Hardware and software installation of SIMCON system

The SIMCON system is predicted to work either in the VME crate or as a stand alone device. It can work with two VME controllers:

a) EPP-VME controller designed by ELHEP Group which can work in slot 0 of the VME crate. One side of the EPP-VME controller is terminated with a VME interface, while the other side is terminated with LPT port. The latter interface can be connected to a standard or embedded PC;
b) SUN Spark controller with the VME interface and Solaris operating system.

The SIMCON itself contains also the LPT port which can be directly connected to a PC. Due to this useful feature the SIMCON user does not need the VME crate to install the system. It can work as an independent device outside the VME crate. While working outside a crate, the SIMCON needs another source of power. The communication between the SIMCON and the possible system assembly configurations are shown in figure 1.

In reference to the above hardware configuration schemes there are possible different software configurations.

The lower layers of the software are dedicated libraries for FPGA solutions. One of these libraries (II Engine) parses "iid" (Internal Interface Definition) file, the others are responsible for communication between the channels.

Figure 2 presents the software configuration appropriate for hardware configuration from figure 1.a). In this case, the SUN station is used as a VME and software controller for the SIMCON. Only this configuration can be used with the DOOCS. The DOOCS software for the SIMCON is described in short in the following chapters.

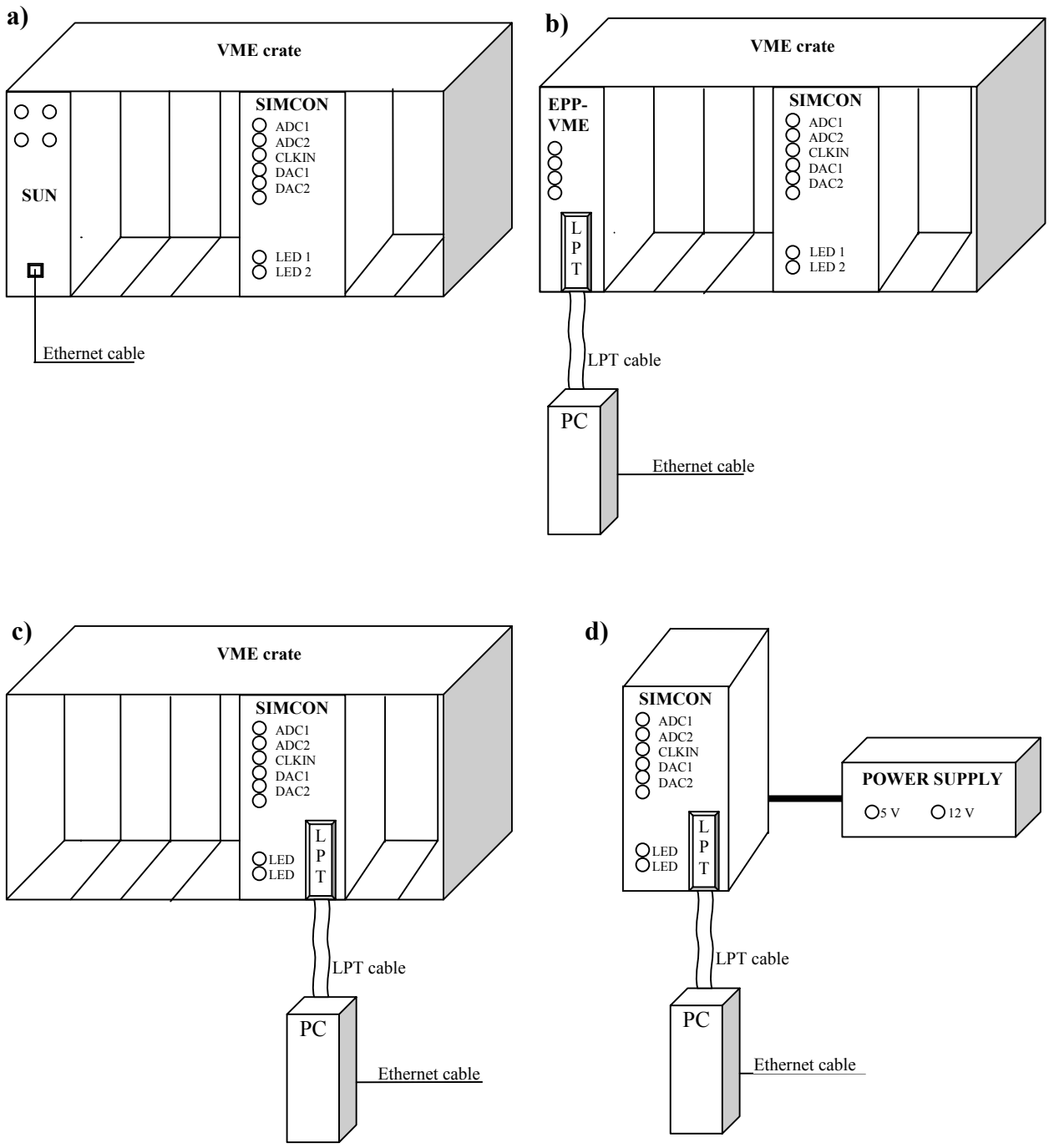


Figure 1. Hardware configuration for the SIMCON a) with SUN controller, b) with EPP-VME controller and PC, c) with local EPP interface, d) with independent power source

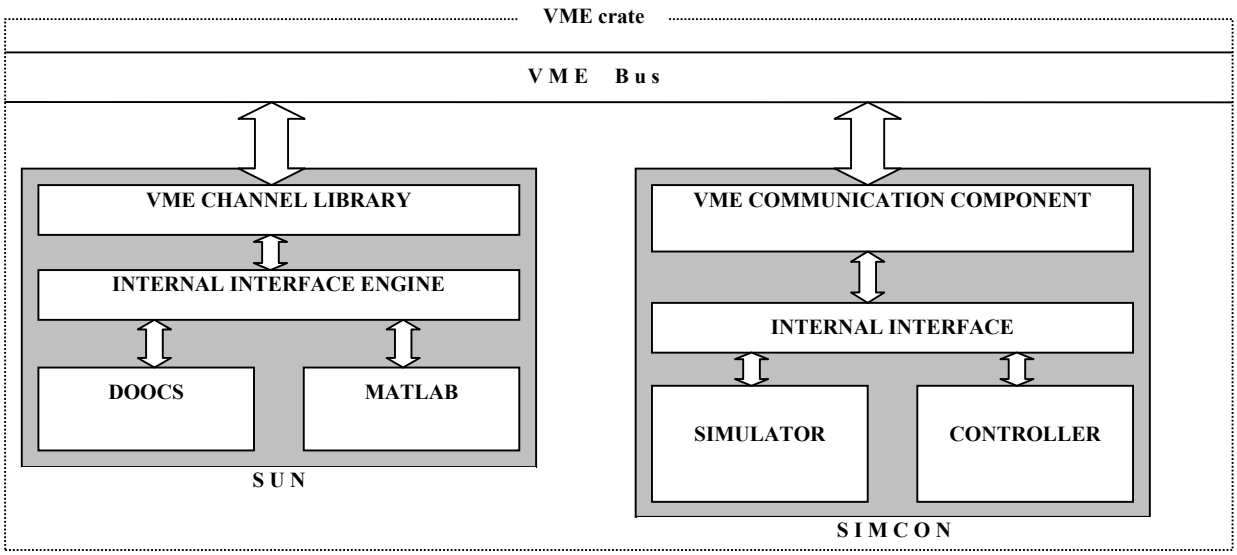


Figure 2. Software configuration with SIMCON and SUN controller

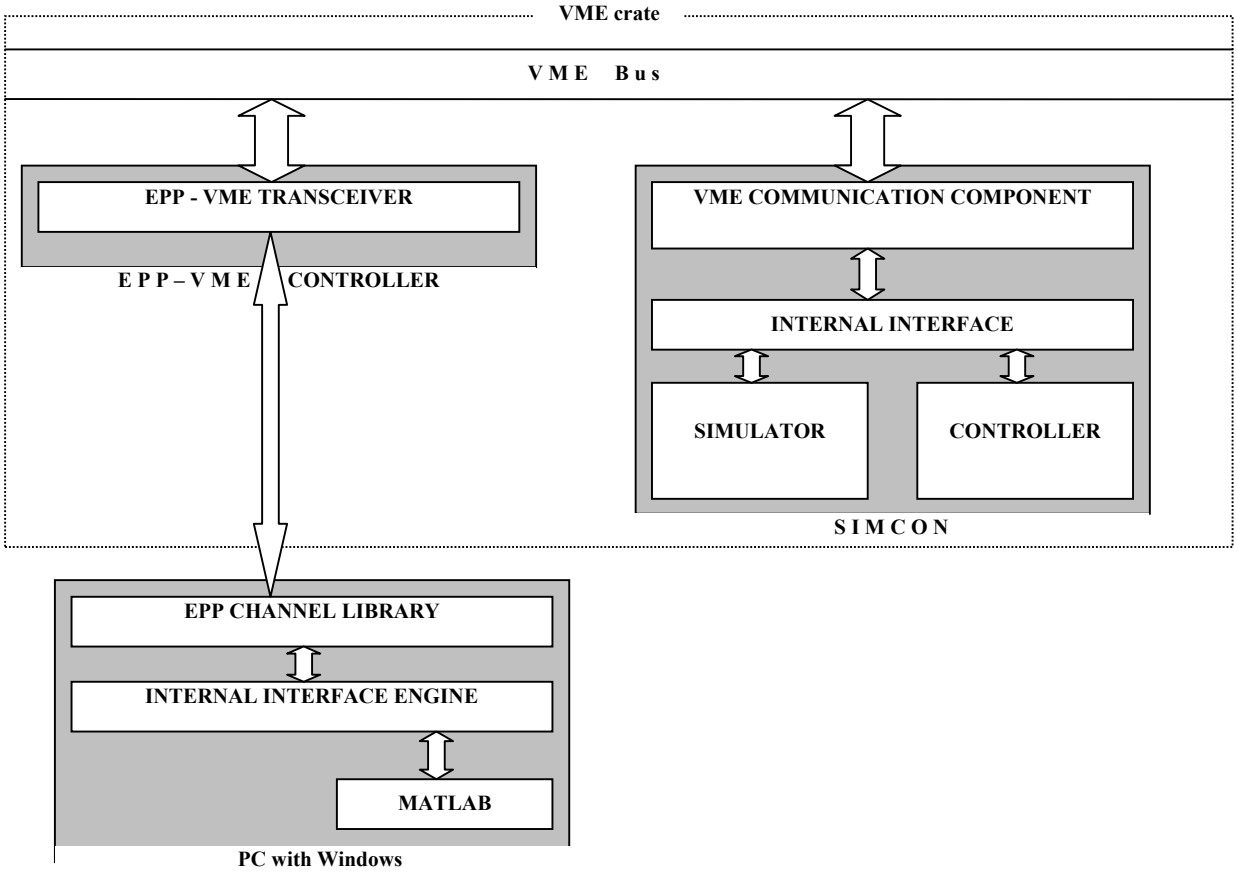


Figure 3. Software configuration with SIMCON and EPP-VME controller

2.1. SIMCON system configuration with SUN controller

2.1.1. Hardware installation

This is the most required type of hardware configuration, because it is used now commonly in the experiment.

Before installing the SIMCON in VME crate it is necessary to specify its base address. The SIMCON also needs 26624 bytes (hex 0x6800) of its address space within the SUN address space. The SUN uses VME address modifier 57 to communicate with devices. It means that the address has 24 bits and 16777216 bytes of address space. The developer should know the address space configuration, in particular of the VME crate, to make the address allocation for the SIMCON.

Figure 2 contains an example of exemplary address configuration. To set the base address, there is a dedicated switch SW1 on the SIMCON motherboard. It has 8 micro-switches which allow to set the eight most significant bits of the address (from 23rd down to 15th). In this case, there are used only four switches from S4 to S8 and only the address lines from 23rd to 20 on the VME bus are used.

For example, figure 3 presents how to set the base address 0xC00000 for SIMCON. It means that the SIMCON will react for each address from 0xC00000 to 0xC06800. When the address is asserted on the VME address bus by VME controller, the address monitor component inside Altera chip compares the four most significant lines from the bus with the four most significant switches from the address switches.

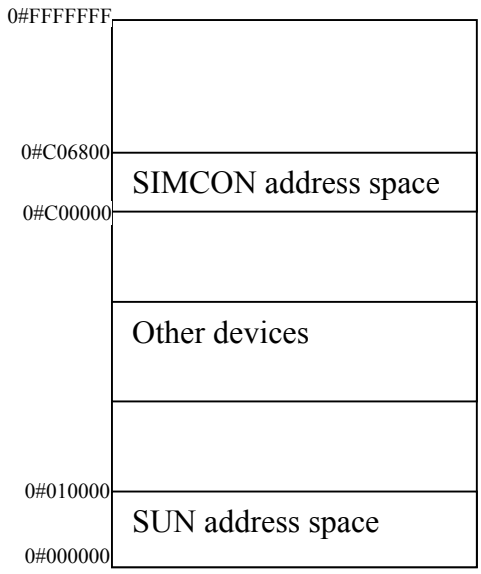


Figure 4. Exemplary configuration of address space

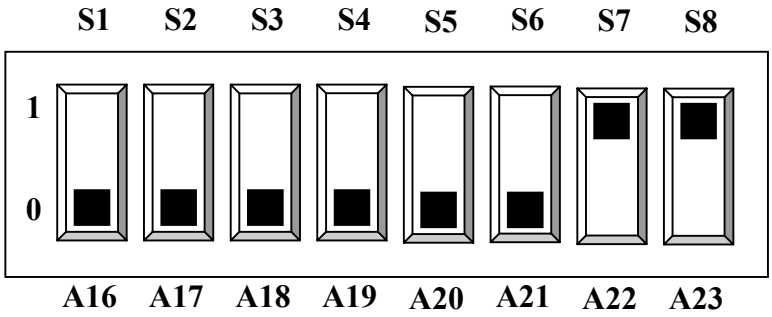


Figure 5. Setting for SIMCON base address 0xC00000 with the switch SW 1 (switch is positioned on SIMCON board).

After this introductory technical step, the SIMCON board can be installed in the VME crate in an arbitrary free slot, except the first one. The whole SIMCON module is 3 slots wide.

The SIMCON board consists of two sub-boards:

- VME-MB – carrier mother board with Altera chip and internal interface to the VME;
- Nallatech board – the evaluation DSP processing board with Xilinx chip (consisting of MB and DB).

Both boards have FPGA chips with internal logic and internal space of addresses. There is a need for a method to distinguish between these two addressing spaces. For this purpose, there are used lines from 19 to 16 on the VME address bus. The Altera chip has the address 0xF0000 and the Xilinx chip has the address 0x00000. In order to address the specific chip, the address of this chip should be added to the base address of the whole board. Therefore, for the base address 0xC00000 the chips have the following addresses:

- Altera - 0xC00000 + 0xF0000 = 0xCF0000;
- Xilinx - 0xC00000 + 0x00000 = 0xC00000.

The above addresses of the chips are also addresses of the first registers inside the specific FPGA chip. The principles of the usage of VME address bus are presented in figure 6.

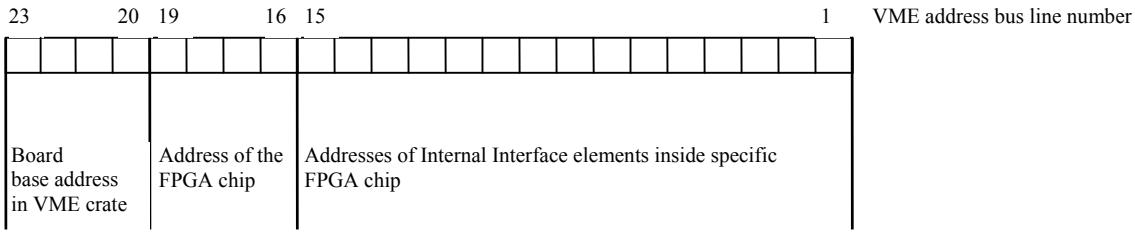


Figure 6. Usage of address lines on VME address bus

2.1.2. Software installation for Matlab on SUN

To install the software for Matlab it is necessary to copy only several files. The files can be divided into a few groups. The list of theses files can be found in attachment A. These files, except for GUI files, can work in Matlab version 6.5 or 7. The GUI files are appropriate only for Matlab 7, because they are incompatible with Matlab 6.5.

Functions providing direct access to FPGA registers

This kind of function works as MEX-type functions in the Matlab. They are the basis for all other m-files described in the following chapters. There is only one way to get access to the FPGA registers and memories through these functions. They are used to operate on the physical elements of FPGA system (i.e. bits, registers, memories). Direct access functions are described in details in TESLA Report [1].

Configuration files

Configuration files are used to perform the basic operation on the sets of FPGA registers. For example, to write a new version of the Feed-Forward table, there have to be performed many operation on the FPGA chip using direct access functions. In order to simplify such kind of operations, there were prepared m-functions. These m-functions can be used to operate on the functional blocks of FPGA controller and simulator (i.e. operating tables, rotation matrixes, IQ detector parameters, etc.). The system developers who have worked out their own control algorithms in Matlab can use these functions to download control tables and parameters to FPGA and make the result readouts.

Algorithm files

This set of m-files contains one version of worked out algorithm for FPGA controller. Details of this algorithm for simulator and controller are presented in TESLA Reports [3], [4] and [5].

GUI files

All files described above are used in the graphical user interfaces. The GUI were made using Matlab tool the“GUI Builder”. There are several GUI windows: for the simulator, the controller and for the readouts. These panels are described in further chapters.

The files listed in attachment A can be copied to one folder. Some files are different for different platform, therefore only files for proper platform should be copied.

2.1.3. Booting FPGA

There are two FPGA chips in the SIMCON 2.1. system. The Altera chip is placed on the VME mother board and is booted from the EPROM on the board automatically after the power on. Thanks to that, the VME interface is ready to work after a few milliseconds after the power is on.

The second FPGA chip is the Xilinx one on the Nallatech board. It can be programmed from external system like SUN or PC through the VME Bus and Altera chip on the mother board using JTAG standard.

In order to boot the Xilinx chip, there are used the files listed in attachment A. The base address in file vme.conf should have the value: BootAddress = BaseAddress + AlteraAddress (see chapter 1.1.1). For example, the base address of the board is 0xC00000, the AlteraAddress has address with value 0xF0000. Thus, the BootAddress should have the value 0xC00000 + 0xF0000 = 0xCF0000. The next step is to run the file 'run.sh' from the console. The program should show some messages about the progress of booting. The booting process takes about 1 minute and 40 seconds. When the program finishes the booting, the Xilinx and the whole SIMCON system are ready to use. The green configuration LED D1 on the Nallatech board should blink to indicate the successful configuration.

2.2. SIMCON system configuration with EPP-VME controller and PC-Windows

2.2.1. Hardware installation

The hardware installation in this configuration differs only in installation procedure of the VME bus controller. The installation of the SIMCON board is the same and it is described in chapter 1.1.1. The EPP-VME controller should be installed in slot 0 of the VME crate. There is LPT port in front panel of the controller which is used to connect a PC with the LPT cable. The corresponding hardware configuration is shown in figure 1 b). The software configuration is presented in figure 3. The LPT port of the PC must be configured to the EPP protocol in BIOS.

2.2.2. Software for Matlab installation on PC-Windows

The software files can be copied to one folder on the PC. The Windows version of communication files should be taken into account. The explanation of groups of files is in the chapter 1.1.2.

2.2.3. Booting FPGA from PC

The method of booting the SIMCON in the PC-Windows configuration is the same like for the SUN configuration and it is explained in chapter 1.1.2. The only difference is in the names of the booting files. The configuration file has name vmeii.ini and the boot file has name boot_simcon2.1-3000.bat for the Nallatech board with Xilinx xc2v3000 and boot_simcon2.1-2000.bat for Nallatech board with Xilinx xc2v2000.

The booting process takes much more time in this configuration than in the SUN configuration because of the speed of the LPT port.

3. FPGA configuration – Matlab files – manual

This chapter describes some sets of Matlab functions which allow to get access to the functional parts of the SIMCON or just to its functionalities. The TESLA note [1] deals with hardware description of SIMCON. It presents a detailed description of all registers and switches inside the FPGA. To set configuration of some working modes, there is a need to load many different values to different registers in the FPGA using basic MEX-functions like `ii_set_word(), ii_set_bits or ii_set_area()`. These tasks are quite complicated, because the user needs knowledge of all configuration registers and their configuration values. That is why there has appeared the need to create some set of higher level Matlab functions which would allow the user to set some modes of work. For example, in order to set the controller mode or cavity simulator mode, the user does not need to know how many and which registers should be set. He only needs to use one simple Matlab function `FPGASetMode(mode)` with one argument. The same method is used for the readouts. In order to read some signals from the FPGA, there should be done quite complicated combination of setting and reading registers and memories. Instead of this, there is a simple function `FPGAReadDAQ(Signals)`.

Many of these functions are described below, but there still appear new functions, which allow, in easy way, to perform some actions on the SIMCON. This set of functions can be used for Matlab control algorithms development or by programmers, like to create the GUI interfaces for the SIMCON.

This chapter described only several basic functions, because the number of these new functions changes quite fast and it depends on the requirements of new prepared algorithms based on the SIMCON.

Figure 7 presents block diagram of the SIMCON. It will be useful during the description of the details of the following functions.

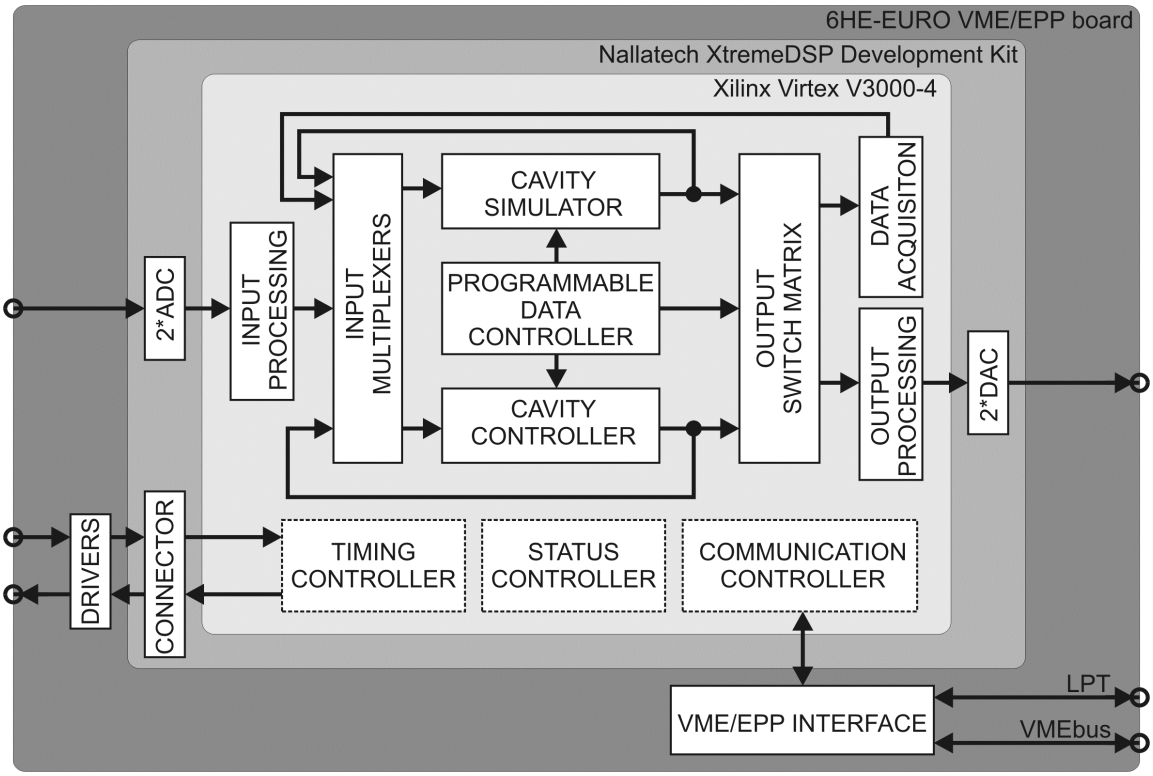


Figure 7. Block diagram of the SIMCON system

3.1. Functional parameters of SIMCON

This chapter presents the basic functions which can be used to set parameters and functionalities which read the status of different components of the SIMCON system.

3.1.1. Timing system

The timing system is used to provide 1MHz clock and trigger signal. Timing signals can be delivered from the internal clock system or from external system through the digital inputs. A single simple function is used for this purpose

```
FPGASetTiming(mode)
```

`mode` – 0: timing signals from internal clock system;
`mode` – 1: timing signals from external clock system;

```
result = FPGAReadTiming()
```

`result` – number, which indicates mode of timing system

When the internal source clock system is set, the two LEDs (name: 1MHz, TRG), which are located on the SIMCON front panel, should blink.

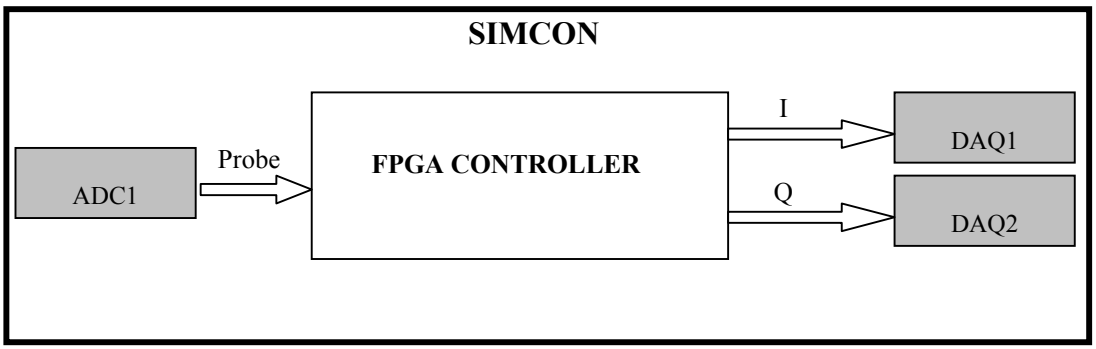
When the external source timing signals is set, then the same LEDs should blink if the timing signals are connected to digital inputs described as 1MHz and TRG on the front panel.

More about the timing system can be found in [1] chapter 4.

3.1.2. SIMCON function selection

The SIMCON system can work in three main modes:

- **controller mode** – in this mode, the FPGA controller is used to receive modulated signal from the real cavity and to drive klystron through the vector modulator. It means, that ADCs are connected to the input of the controller. Modulated signal from the cavity probe should be connected to ADC1. Output of the controller (I and Q control signals) are connected to DACs.



- Figure 8. Block diagram of SIMCON in the controller mode

- **cavity simulator mode** – in this case, the ADCs are connected to the input of cavity simulator. The control signals from external control system (I and Q signals) drive the cavity simulator. The output of cavity (modulated signal) is connected to DAC1 and the detuning signal is connected to DAC2.

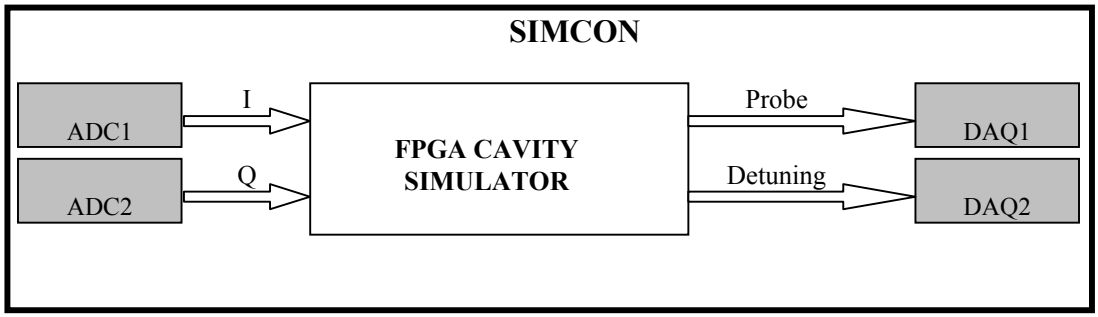


Figure 9. Block diagram of SIMCON in the cavity simulator mode

- **internal loop mode –** this mode allows to use the cavity simulator and controller together. It means that the output of cavity simulator is connected to the input of controller and outputs of controller are connected to the input of cavity simulator. All these connections are implemented inside the FPGA. No additional connections are required from outside of the FPGA. The ADCs are not used in this mode. The DACs can be used to provide any one from 22 internal signals from the FPGA structure as the analog signals.

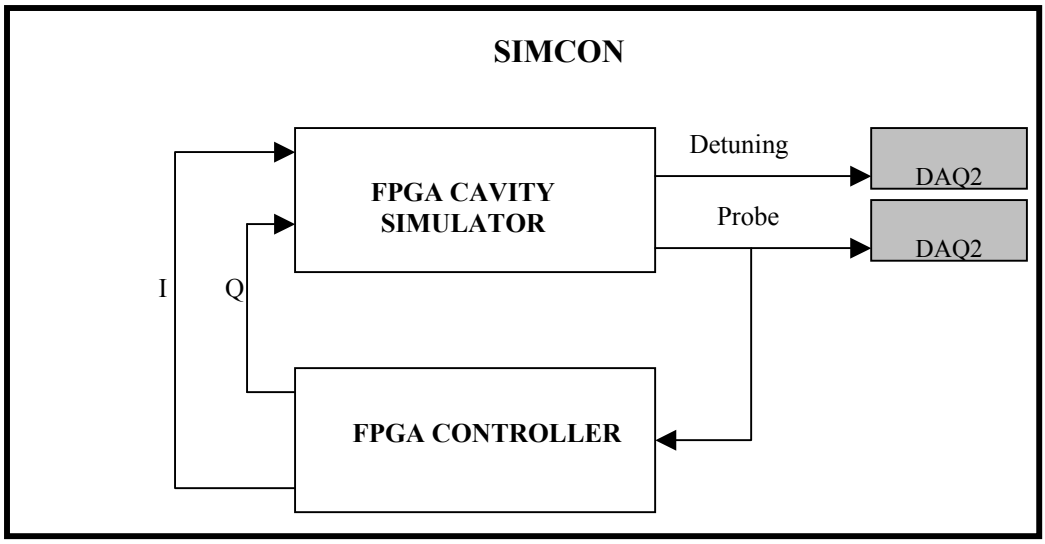


Figure 10. Block diagram of the SIMCON system in internal loop mode

Setting one of these modes needs configuration of many internal registers of the SIMCON. In order to do that there is used one function:

```
FPGASetMode(mode)
```

`mode` – 0: controller mode; 1: cavity simulator mode; 2: internal loop;

```
result = FPGAReadMode()
```

`result` – number, which indicates mode of SIMCON

Read function at the end of configuration process starts the work of selected block like controller and/or cavity simulator.

More about functional modes of the SIMCON can be found in [1] chapters 8 and 9.

3.1.3. DAC output signals

In the internal loop mode, the DACs can be used to observe (on the scope) the analog version of digital internal signals from the FPGA. Selected signal is provided to the input of DAC and then converted to analog signal. There can be observed 22 signals at the output of the DACs. In order to do that, there can be used the following function:

```
FPGASetDAC(dac1, dac2)
```

`dac1, dac2` – number of internal signals, which can be observed at the output of DAC;

```
result = FPGAReadDAC()
```

`result` – it is a vector of two elements. First number indicates signal on DAC1, the second one current signal on DAC2.

More about DAC can be found in [1] chapter 12.

3.1.4. Loading and exchanging of control tables

There can be specified 3 pairs of control tables for the controller: FEEDFORWARD_I, FEEDFORWARD_Q, SETPOINT_I, SETPOINT_Q, GAIN_I and GAIN_Q. Each table can be loaded independently. In this case, the tables are always loaded in "exchanging tables' mode". It means that the controller, simulator or both can work and the control tables can be replaced by new tables between pulses without disturbing the work of the system. That is why the table should be exchanged in pairs i.e. always I and Q of specific table. However, not all of these tables are required when the exchanging function is invoked.

```
result = FPGASetCtrlTables(tables)
```

`tables` – this is matrix which contains new data for the tables which will be replaced. The matrix consists of tables in the following order: FEEDFORWARD_I, FEEDFORWARD_Q, SETPOINT_I, SETPOINT_Q, GAIN_I and GAIN_Q.

This function can be execute even during the pulse, because the data is loaded to spare parallel tables, which correspond to the control tables. These parallel tables are inactive until the pulse finishes and then, before the next pulse, the current control tables are replaced by the parallel tables with new data.

The method of reading of control tables is described in the next chapter.

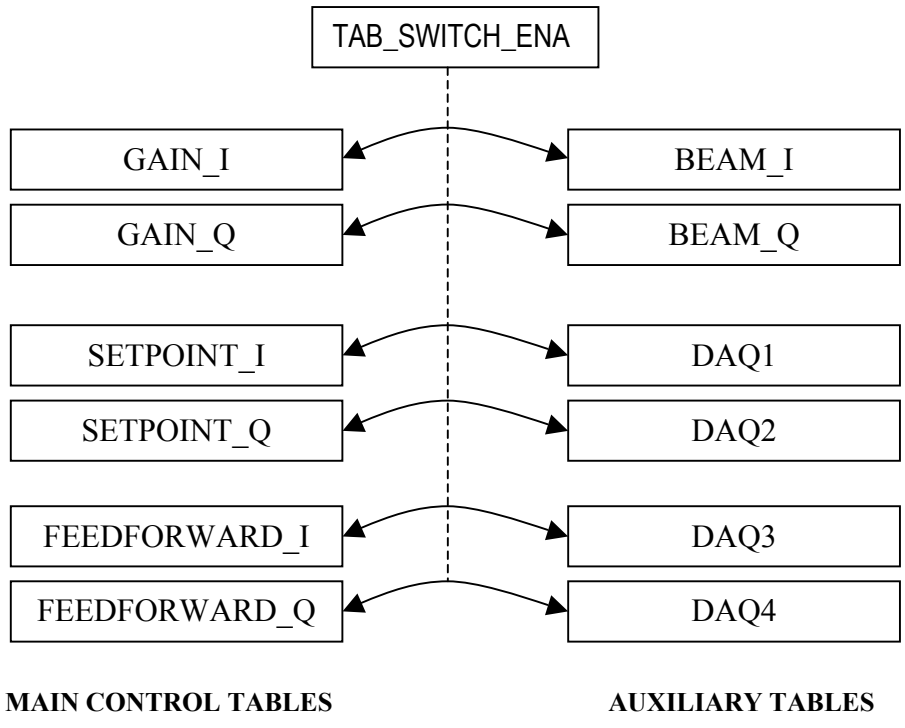


Figure 11. Scheme of control tables exchange mechanism

More about the architecture of SIMCON tables can be found in [1].

3.1.5. Loading and exchanging of beam table

The exchange mechanism of the beam table works in the same way like the control table exchanging. The difference is that the beam table can be changed in the cavity simulator mode and the function has only one parameter.

```
result = FPGASetBeam(tables)
```

`tables` – this is matrix which consists of two rows – BEAM_I and BEAM_Q, each one of 2048 samples.

As it was mentioned earlier, this table can be replaced only between the pulses.
The method of reading of the beam table is described in the further chapter.
More about the beam table can be found in [1] chapter 8.

3.1.6. Settings of calibration parameters

The SIMCON contains some blocks which are responsible for calibration of the output signals from the DACs and for calibration of the input signals going into the ADCs.

2.1.6.1 Setting of output rotation matrix D

This square matrix is situated right before the DACs and is used for adjusting the amplitude and phase of output control signals. It is used all the time. The matrix has the following coefficients [1 0; 0 1] as default values. It means that it does not have any influence on the output signals. The phase rotation and scaling of amplitude make sense when the output signals are I and Q from the controller. To set and read the matrix coefficient there are used the following functions:

```
FPGASetOutputMatrix(amplitude, phase)
```

`amplitude` – value can be within range <0;2), value ‘1’ doesn’t influence the output signals;
`phase` – value in radians and range from -Π to Π, value ‘0’ doesn’t influence the output signals.

```
result = FPGAReadOutputMatrix()
```

`result` – it is vector which consists of four numbers – coefficients of output rotation matrix, which has the following form [cos(x) –sin(x); sin(x) cos(x)].

Settings of D-matrix coefficients impacts the output signals immediately.
More about output the calibration matrix can be found in [1] chapter 6.

2.1.6.2 Setting of input calibration blocks

There are two input calibration blocks – one block per one ADC. They are located right after the ADCs outputs. One functionality of these blocks is used to compensate the offset of the input signals. Another functionality is to fit the amplitude of the input signals to the range of FPGA registers.

```
FPGASetInputCal(adc, amplitude, offset)
```

`adc` – values '0' or '1' indicate, which ADC will be adjusted;
`amplitude` – value in the range <0;2) allows to attenuate or to amplify the input signal; too big value can cause saturation of the input signal;
`offset` – value in the range $<-2^{17}; 2^{17}-1>$ and it is measured in bits;

```
result = FPGAReadInputCal(adc)
```

`adc` – values '0' or '1' indicate, which ADC will be adjusted;
`result` – it is a vector which consists of two numbers, first one is the calibration amplitude, the second one is offset calibration.

Settings of the input calibration block impacts the output signals immediately.
More about the input calibration can be found in [1] chapter 5.

2.1.6.3 Setting of input rotation matrix C

This square matrix is positioned right after the IQDetection block and is used for adjusting the amplitude and phase of the input signal from cavity. The I and Q signals from the output of IQ detector block are multiplied by matrix C. It is used all the time when the controller is running. As a default, the matrix has the following coefficients [1 0; 0 1]. It means that it does not have any influence on the incoming signal. To set and read the matrix coefficient there are used the following functions:

```
FPGASetInputMatrix(amplitude, phase)
```

`amplitude` – value can be within the range <0;2), value '1' does not influence the output signals;
`phase` – value in radians and range from -Π to Π, value '0' does not influence the output signals.

Settings of C-matrix coefficients impacts the output signals immediately.
More abut the input calibration matrix C can be found in [1] chapter 7.2.4.

3.1.7. Setting of IQ detection start point

As it was described in the previous documents [1-5], there is an IQ detection block, which uses the samples of modulated signal from the cavity every 1μs. These samples represent the in phase I and quadrature Q part of the complex signal and they appear in the

following order I, Q, -I, -Q. The parameter DRV_START shows what means the first sample of input signal, whether it is I, Q, -I or –Q.

```
FPGASetIQStart(start)
```

`start` – parameter accepts the values: 0, 1, 2, 3 which correspond to I, Q, -I, -Q samples. Changing of this parameter by 1 causes the change of phase of the input signal by 90 degrees;

```
result = FPGAReadIQStart()
```

`result` – it is the value of start point of IQ detector.

More about the IQ detection can be found in [1] chapter 9.

3.2. Readouts

The SIMCON system contains a special mechanism which allows to read many different signals from its internal structure. Figure 12 shows a block diagram of SIMCON and the points which can be monitored by the internal DAQ system. As it was mentioned in [1], the DAQ system consists of four parallel areas of memory with 2048 words each. Each word of the DAQ memory has 18-bit resolution and includes one sample. The samples are collected every 1μs. This mechanism allows to gather simultaneously four signals from 10 different points of the SIMCON during a single cavity pulse.

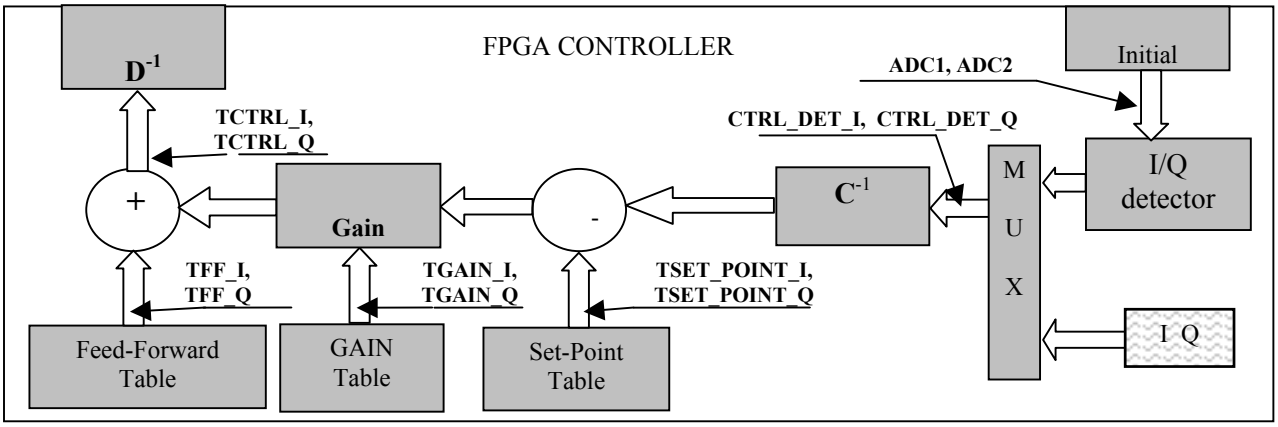


Figure 12. Signal points in the cavity controller for readout system

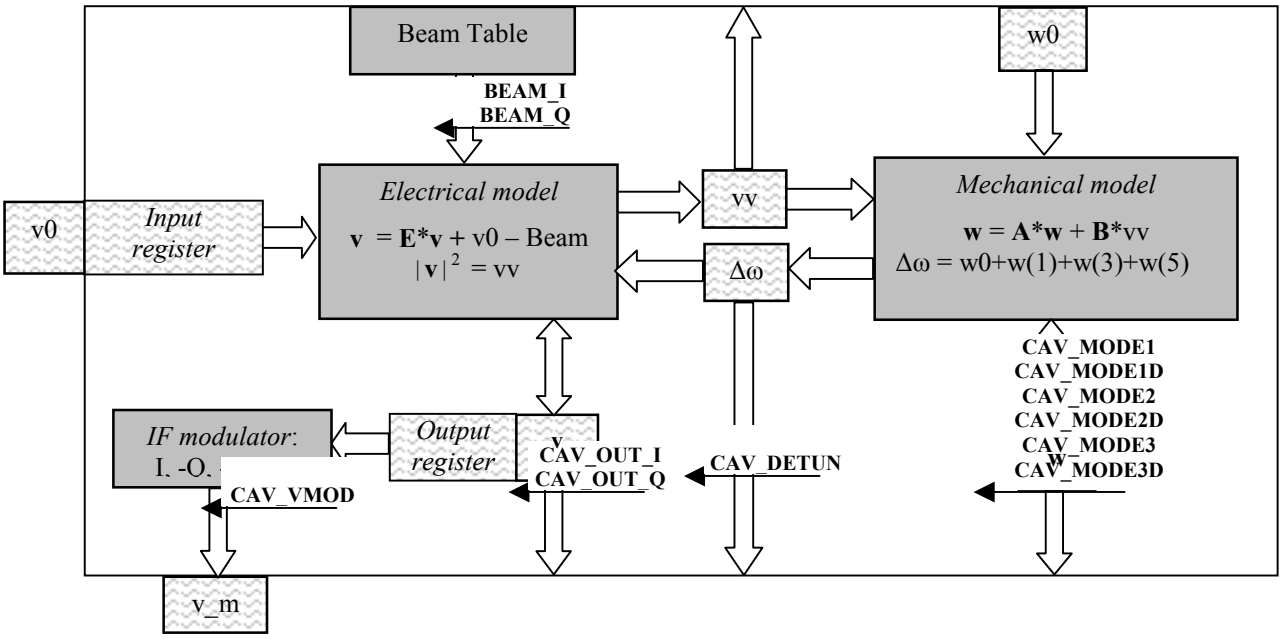


Figure 13. Signal points in cavity simulator for readout system

```
[Z, err] = FPGAReadDAQ(signals)
```

`signals` – vector, which can have from 1 to 4 number of signals. Signals numbers and names are gathered in the table in attachment B.
`Z` – matrix, which contains from 1 to 4 rows, each row means one signal with 2048 samples;
`err` – value '0' indicates that the readout has been completed, '1' means that there was time out.

This Matlab function can read from 1 to 4 signals simultaneously and number of signals to be read depends on the size of an argument *signals*. The number of rows of returned matrix Z depends on the size of *signals* argument as well. After invoking the function ReadDAQ, the SIMCON waits for the trigger. When the trigger comes, the SIMCON starts to collect samples every microsecond. The ReadDAQ is suspended during the collecting of samples. After 2048 μs the data are read out from the memories DAQ1-4 and put into the matrix Z. The function finishes its work.

Example:

```
[Z, err] = FPGAReadDAQ( 7, 8 );
if (err==0) then
     plot(Z(1,:))
     hold on
     plot(Z(2,:))
end
```

Result of this example is presented in figure 14.

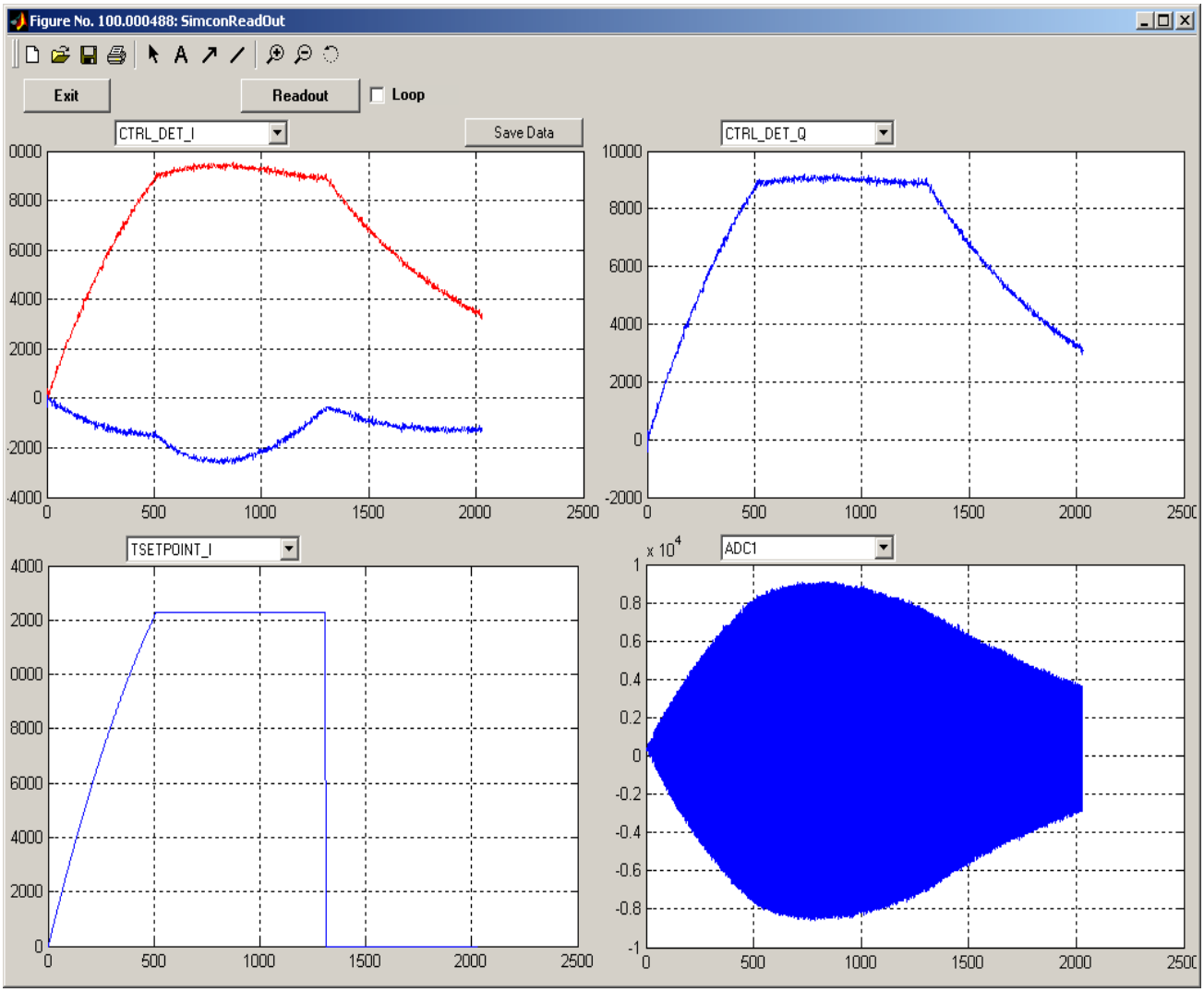


Figure 14. Plots achieved with readout systems of the SIMCON system

The values of readout samples of all data are calibrated in bits and its range is extending from -2^{17} to $2^{17}-1$ except for the tables GAIN and BEAM, which range is -2^{11} to $2^{11}-1$.

More about the readouts and the DAQ system can be found in [1] chapter 10.

4. Algorithm files in Matlab

The architecture of SIMCON is based on the algorithm of cavity simulator and controller. These algorithms were elaborated in Matlab. Matlab functions take as the input arguments several parameters, which define the condition of work of cavity simulator and controller. These algorithms return parameters and control tables for FPGA devices.

The principles of algorithm were described in [3], [4] and [5]. There are listed Matlab files which are used for controller and simulator. Each file has detailed description there.

`INPUT_D=INPUT_INIT()` – this function is used to initialize the structure of input parameters. There are several parameters, which define the condition of work of controller and cavity simulator. The output of this function is the pointer to the structure of parameters.

`[U,Y,W]=INITT(INPUT_D)` – this function prepares all parameters to initialize FPGA. The input parameters from structure INPUT_D are recalculated and rescaled. The results of this function are:

- U - vector of scalars, which are loaded to FPGA,
- V – matrix of tables, which are loaded to FPGA,
- W – structure of local variables, which are used by other functions in this algorithm.

`FPGAINIT(U, Y)` – this function loads all necessary parameters from vector U and tables from Y into FPGA. This function also sets up the mode of work of SIMCON.

`[Y,W]=RE_CONTROL1(W)` – this function is used by Adaptive Feed Forward algorithm to recalculate new control tables for controller. It uses as input argument the structure W and returns the same structure with modified parameters and the tables in Y with new control tables.

`FPGAEPPReloadTables(Y,W)` - the function loads to FPGA new control tables from variable Y and exchanges them with existing control tables between pulses.

`Z = FPGAReadDAQ(Sig)` – this function was described in chapter 2.2. In this case, it is used to read signal CTRL_I, CTRL_Q and CTRL_DET_I, CTRL_DET_Q – see chapter 2.2. The matrix Z consists of four rows, which correspond to specific signals. After this function the matrix Y is updated by signals from matrix Z.

`[Y,W]=RE_CONTROL2(Y,W)` – this function is the second part of Adaptive Feed Forward algorithm. It is used to estimate the half bandwidth and the detuning of cavity. These are crucial parameters, which are used to recalculate new control tables in function RE_CONTROL1.

5. Matlab graphical user interfaces

There are a few graphical user interfaces, which are used to work with the SIMCON for dedicated windows. The functionality of these panels is based on Matlab files from chapter 3 and algorithms described in [3], [4] and [5].

5.1. Controller

The control panel for the cavity controller consists of two files. FPGACTRL.m contains Matlab code of events called from the panel. The file FPGACTRL.fig contains definition of appearance of the control panel.

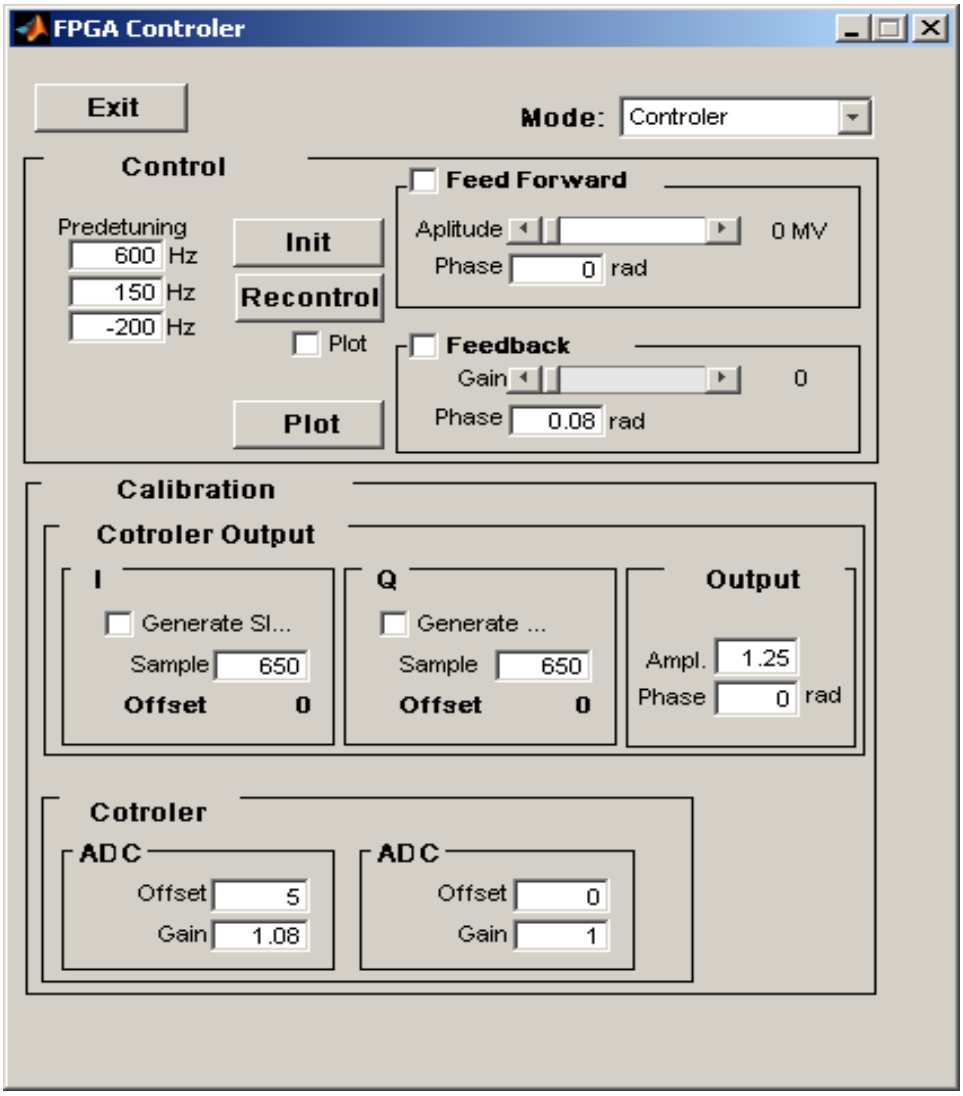


Figure 15. Control panel for cavity controller

The cavity controller can work in three modes. The mode can be chosen from the combo box 'Mode'. The possible configurations are: Calibration, Simulator and Controler.

The calibration mode is used to find the offset of DACs and to find the minimum value of the signal at the output of the vector modulator. It is very important to calibrate the offset,

because due to that the level of the signal from the vector modulator and the power going out from the klystron will be the lowest possible.

This mode is used to generate special signals at the output of SIMCON on DACs. When the 'Calibration' mode is chosen, one of two 'Generate slope' checkboxes should be checked in section **I** or **Q** in the frame 'Controller Output'. Only one of these checkboxes can be checked in the same time. When the checkbox is set, the slope will be generated on the output of DACs. Depending on I or Q checkbox the slope is generated and loaded into the table FEEDFORWARD_I or FEEDFORWARD_Q. The slope changes from the minimum to the maximum value, what means from -2^{17} to $2^{17}-1$ in two complementary codes and looks like the first graph in figure 16 (upper one).

This signal drives the vector modulator and the envelope of the signal at the output of VM looks like the second graph in figure 16 (lower one). This signal can be observed on the scope. Using the scope and cursor on the screen the point of the minimum can be determined. This point, measured in microseconds, is the number of the samples from FEEDFORWARD table and the value of this specific probe is the offset of the output. When the offset is determined, the user should write the number of the samples to the field 'Sample' corresponding to I or Q slope. The program automatically calculates the value of the offset and this calculated value is written to the appropriate offset register in FPGA. The method of adjusting the offset is identical for I and Q output (DAC1, DAC2) and must be done successively one after the other (in turn).

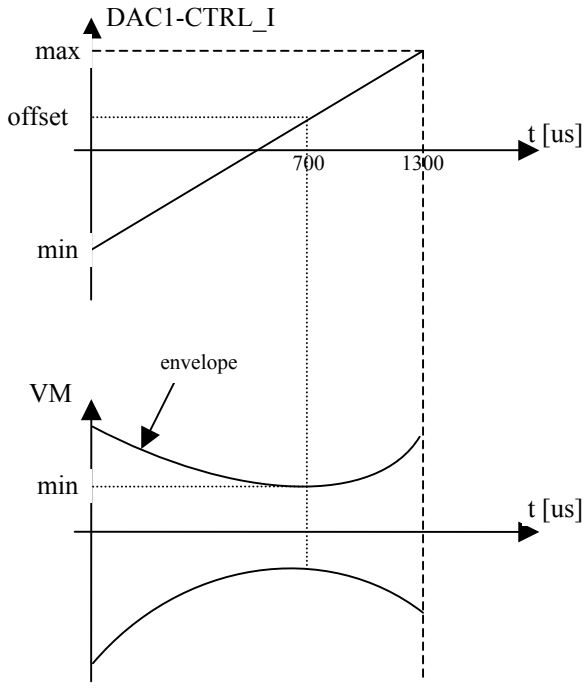


Figure 16. The method of calibration of the outputs of DACs offset

There are also other parameters of calibration. These parameters can be set in each mode of work. The frame 'Output' in section 'Calibration' is used to set coefficients of the output matrix. This matrix, as was mentioned above, is used to rotate and change the amplitude of the output signals CTRL_I and CTRL_Q. The user specifies the rotation phase and the amplification index and the program automatically calculates the coefficients of the matrix. This is done in the same way like in the function `FPGASetOutputMatrix` (see chapter 2.1.6.1).

The other calibration parameters are in the frame ‘Input’ in the section ‘Calibration’. These parameters are used to calibrate the signals just after the ADCs. There are two parameters Gain and Offset for each converter. These parameters can be changed in every mode of work. The parameters of section ‘Input’ and ‘Output’ can be changed any time but they are applied only by pushing the button ‘Init’.

The other modes of work of control panel are: ‘Controller’ and ‘Simulator’. The ‘Controller’ mode is used for work with the real cavity and works with the external timing system. The configuration of this mode is presented in figure 8. In the ‘Simulator’ mode, the controller works with internal cavity simulator and the timing signals are taken from the internal timing module. The configuration of this mode is presented in figure 10. The parameters of the cavity simulator are set automatically with default values. To set other values of parameters, the cavity simulator panel should be used.

In both modes of the ‘Controller’ and the ‘Simulator’ the usage of control panel is the same.

The three fields called ‘Presumed detuning’ are used to determine the detuning of the cavity (real one or simulator). The first value determines the detuning in the point of 0, the second one – at the beginning of the flattop and the third one – at the end of the flattop. The example of the presumed detuning is presented in figure 17 on the fourth graph (lower right) – the red curve. This presumed detuning is used in the algorithm to calculate the first control tables for the Adaptive Feed Forward algorithm.

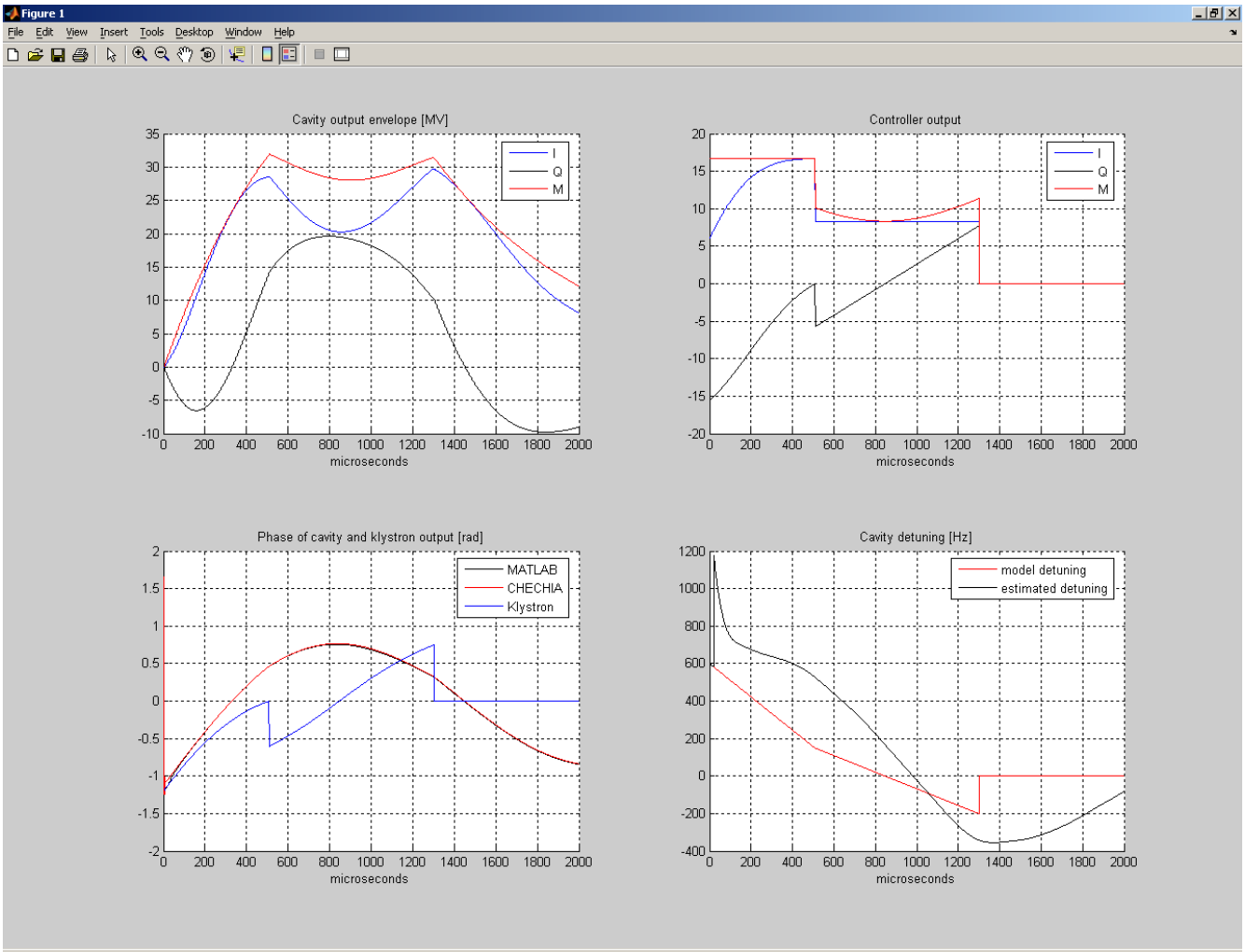


Figure 17. The result of the PLOTs function for cavity parameters

The ‘FeedForward’ section is used to set parameters of the work of the cavity (real one or simulator). The slider is used to set the gradient of the field during the flattop and the ‘Phase’ field is used to set the phase of the field during the flattop. The values of gradient and phase are taken into account by the algorithm files only when the checkbox ‘FeedForward’ is marked. Otherwise the FeedForward tables are zero.

The ‘FeedBack’ section is used to set the gain in fast feedback loop and the field phase is used to calibrate the phase of the feedback signal from the probe. The assumption is that, the phase of the signal from the probe is calibrated when the phase curve of the controller output and phase of the cavity output (probe signal) start from the same point. This situation is on the third graph in figure 17 (lower left). The phase of the controller (blue curve) and phase of the cavity (red curve) start from the same point.

When the all parameters are set in the control pane,l the button ‘Init’ is used to apply them. The function INITT.m is invoked. It calculates all control parameters for the configuration registers inside FPGA and loads them to the chip. All the control tables like gain, set point and feedforward are also calculated and loaded to FPGA. After loading all parameters and tables, the controller is started and can work.

This control system is based on the Adaptive Feed Forward (AFF) algorithm. The button ‘Recontrol’ can be used to invoke the next step of loop of the AFF algorithm. The system makes readout from FPGA. It reads signals CTRL_DET_I and CTRL_DET_Q from FPGA. These are the signals after the IQ Detector block, which detects I and Q signal from probe signal from the cavity. These signals are used by the algorithm to calculate half bandwidth and detuning of the cavity. These two parameters: half bandwidth and detuning are the main parameters of the cavity equations, which are used to calculate the new control tables. After the calculation of new control tables they are loaded again to FPGA. The new tables are loaded to FPGA using a special exchange method between the pulses (see chapter 2.1.4).

To see the results of work of the controller and the cavity, there is a special function PLOTs.m, which is invoked by pushing the button ‘PLOT’. The window looks exactly like in the figure 17. The legend explains the meaning of the curves.

In order to make the plots every time after pushing the button ‘Init’ or ‘Recontrol’, the checkbox ‘Plot’ should be marked.

5.2. Cavity simulator

Control panel for the cavity simulator is presented in figure 18. This screenshot is made from Solaris system and, like other panels, works either in Solaris or in Windows.

Usage of this window is very simple. Most of these parameters are self explaining or can be found in [3], [4] and [5]. After setting all the parameters, the button 'Run' should be pressed. The program recalculates all parameters using the input parameters and loads them into FPGA.

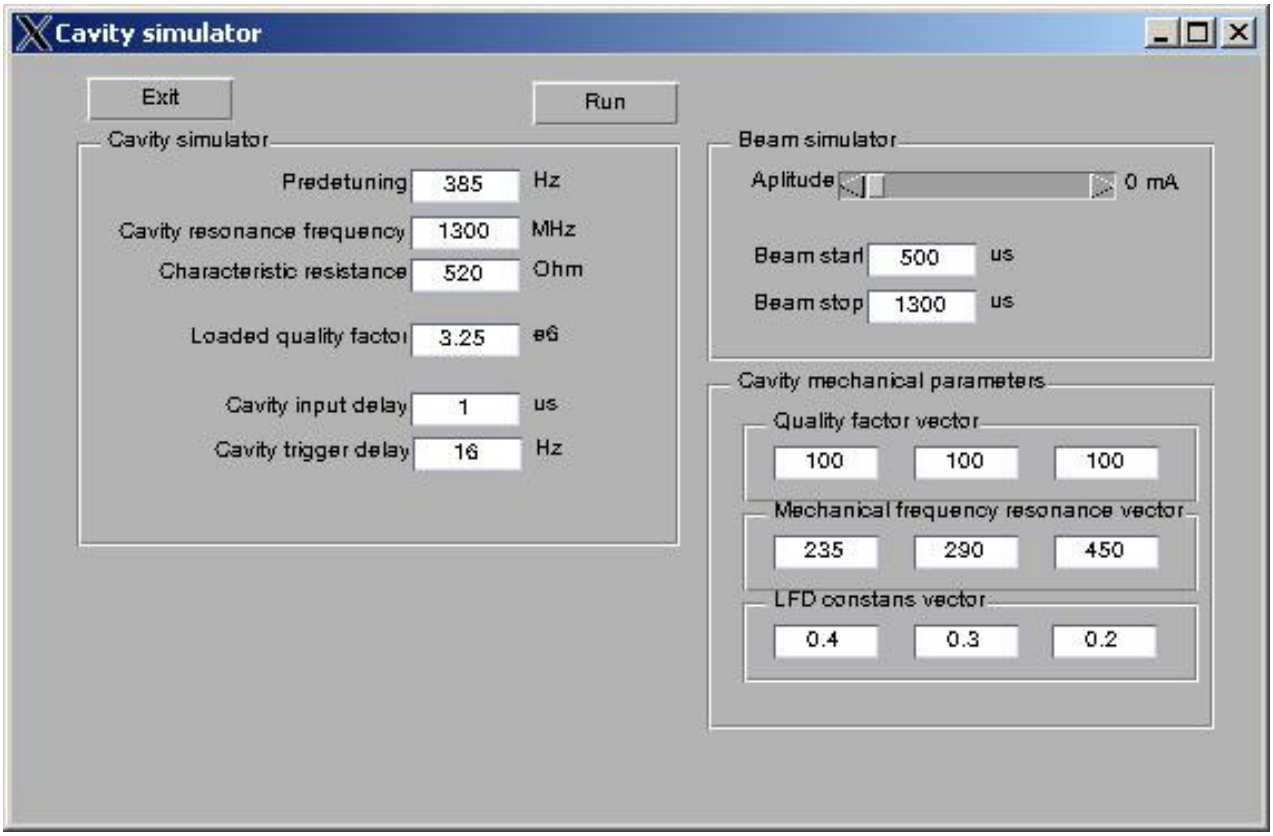


Figure 18. Cavity simulator control panel

This control panel can be used in parallel with panel for the controller in the mode 'Simulator'. When the SIMCON works in the mode 'Controller', the cavity simulator panel should not be used, because when the button 'Run' is pushed, the program sets the SIMCON in the 'Simulator' mode.

5.3. Readouts

The readouts panel is graphic presentation of the function FPGAReadDAQ, which was described in chapter 2.2. Every chart has its own combo box with a list of all available signals inside the FPGA. The signal can be chosen independently for each chart. The numbers of signals are the input parameters for the function FPGAReadDAQ and the result is plotted in these four charts. The readout is invoked by pushing the button 'Readout'. To make readout one after another there is a checkbox 'Loop' next to the button 'Readout'. When the checkbox is marked, the readouts are made in infinite loop until the checkbox will be unmarked.

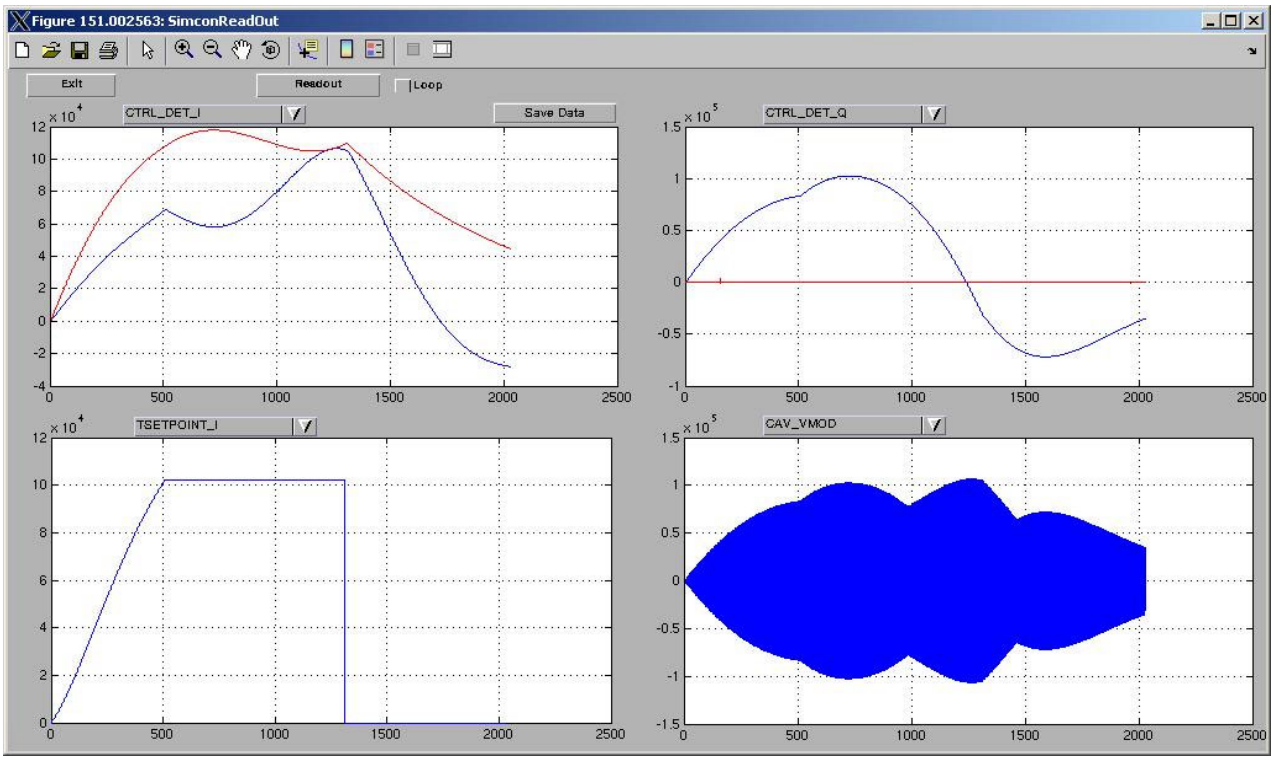


Figure 19. Window of program for readouts from cavity simulator and controller in the SIMCON system

6. DOOCS graphical user interfaces

The DOOCS software for SIMCON consists of two separated server application dedicated for the controller and simulator. This software is meant to be the operation environment instead of the developer software (mainly MATLAB). The detailed description of DOOCS environment will be soon presented in a separate Tesla Report in 2005. Graphical panels presented below were prepared with the DDD (Data DOOCS Display) – the tool dedicated for creating GUI for DOOCS servers.

6.1. Controller

The picture below presents the panel for DOOCS based controller software. It provides almost the same functionality as the Matlab environment. The only difference is the lack of the calibration mode in this version of server. It will be implemented in the future. The present implementation of the controller server works only in the step mode, it means that the *RECONTROL* function is called by the user by pressing *RECONTROL* button. All operation procedures in this system are the same as in the Matlab version.

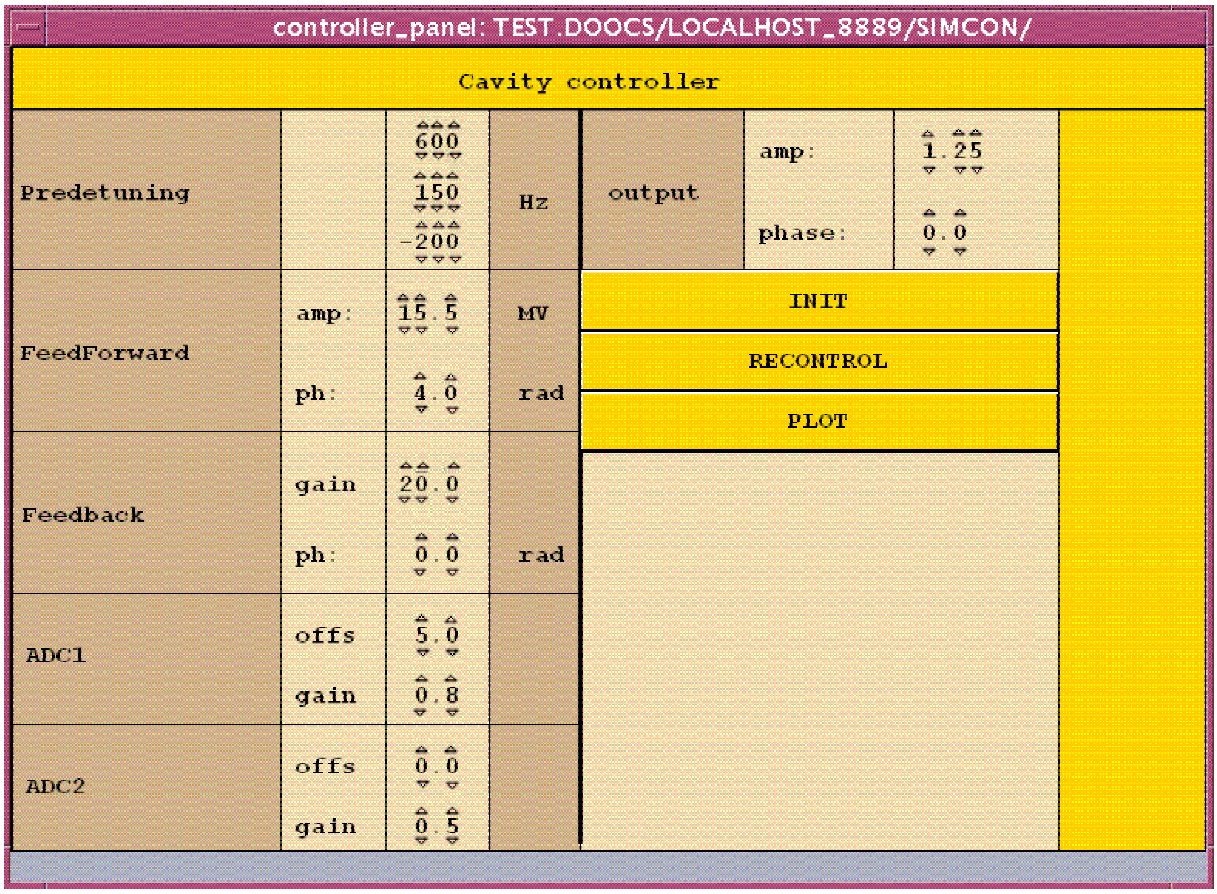


Figure 20. DOOCS based controller panel prepared in DDD.

6.2. Simulator

The simulator panel created in DDD also provides functionality equal to the Matlab version. The important difference is that it does not work in the step mode. Every change of any parameter on the panel triggers needed calculations and tables reloading.

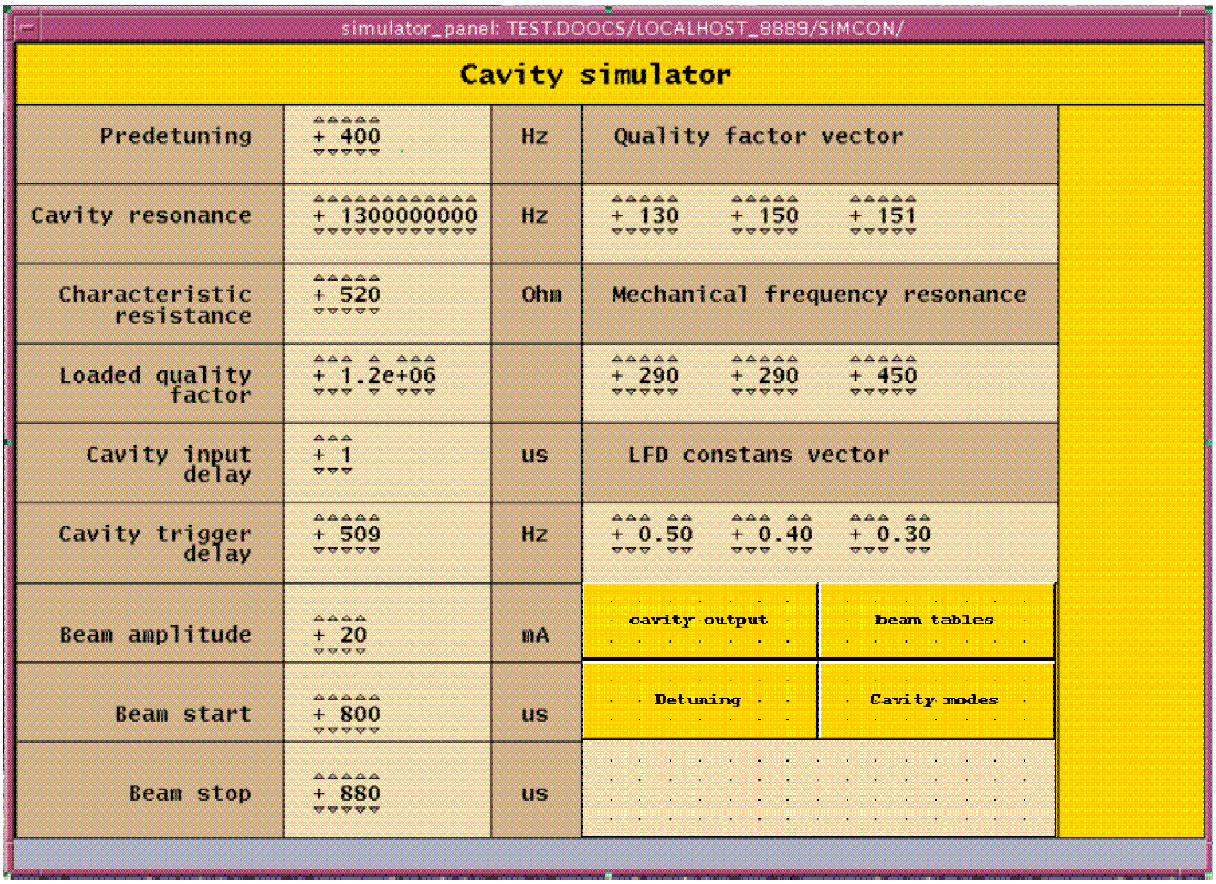


Figure 21. DOOCS based simulator panel prepared in DDD.

6.3. Readouts

Below, an example of readout from the simulator panel was presented. The server can present the raw data (i.e. I and Q components) as well as the pre-calculated plots (amplitude and phase).

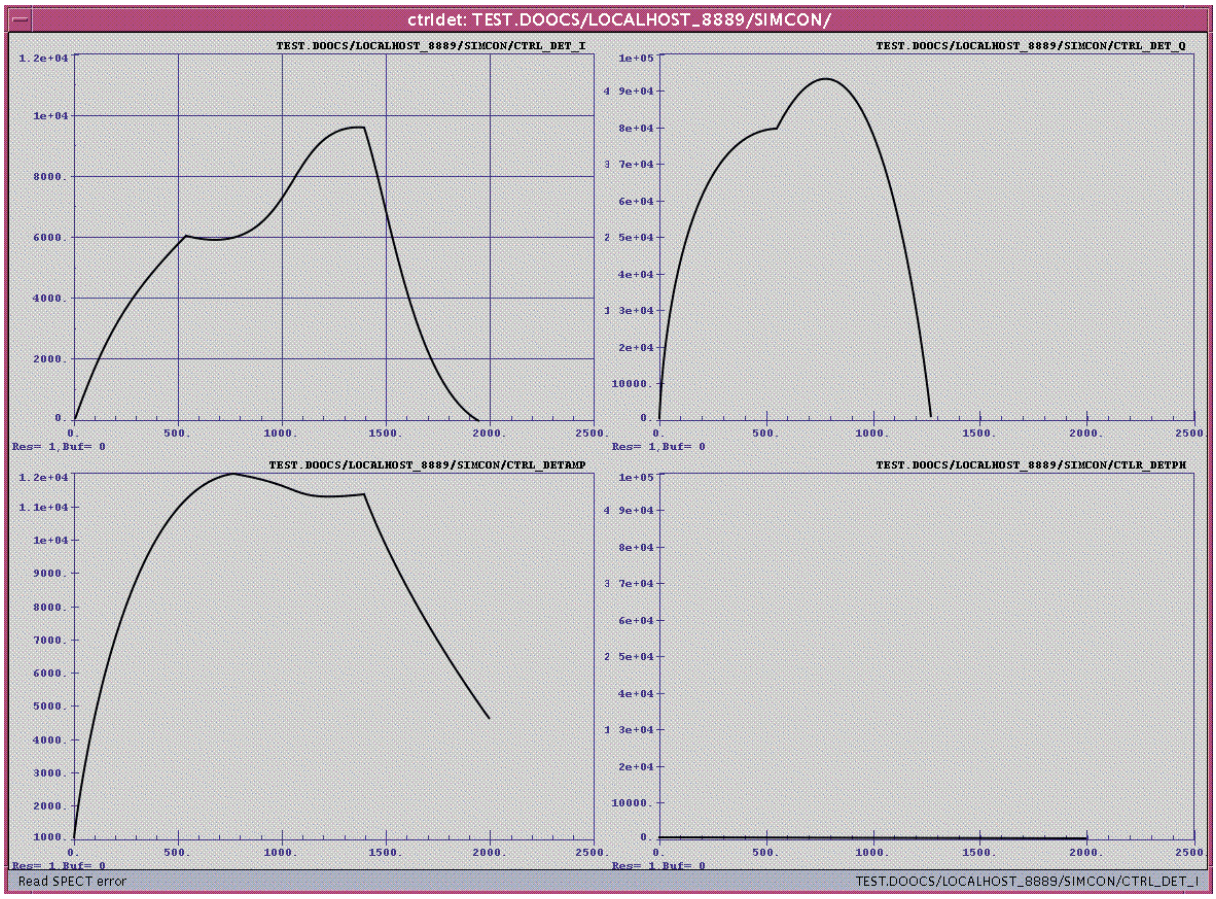


Figure 22. An example of readout performed in DOOCS SIMCON server.

7. Attachments

A. List of files for Matlab

List of files for Matlab on SUN	
Window	**Solaris**
Files used to boot SIMCON 2.1	
jambo.exe	jambo
simcon3000.jam	simcon3000.jam
boot_simcon2.1-3000.bat	run.sh
Direct access to FPGA registers files from Matlab	
ii_get_area.dll	ii_get_area.mexsol
ii_get_bits.dll	ii_get_bits.mexsol
ii_get_items.dll	ii_get_items.mexsol
ii_get_word.dll	ii_get_word.mexsol
ii_lock.dll	ii_lock.mexsol
ii_merge_bits.dll	ii_merge_bits.mexsol
ii_set_area.dll	ii_set_area.mexsol
ii_set_bits.dll	ii_set_bits.mexsol
ii_set_merged_bits.dll	ii_set_merged_bits.mexsol
ii_set_word.dll	ii_set_word.mexsol
ii_unlock.dll	ii_unlock.mexsol
channel.txt	channel.txt
vmeii.dll	libvme.so
vmeii.ini	vme.conf
libxiid.so	libxiid.so
source.txt	source.txt
LLRF_simcon_vdsp_vme.iid	LLRF_simcon_vdsp_vme.iid
LLRF_simcon_vdsp_config.iid	LLRF_simcon_vdsp_config.iid
Inpout32.dll	-
Configuration files in Matlab	
FPGASetTiming.m	
FPGAReadTiming.m	
FPGASetMode.m	
FPGAReadMode.m	
FPGASetDAC.m	
FPGAReadDAC.m	
FPGASetCtrlTables.m	
FPGASetBeam.m	
FPGASetOutputMatrix.m	
FPGASetInputCal.m	
FPGASetInputMatrix.m	
FPGASetIQStart.m	
FPGAReadIQStart.m	
FPGAReadDAQ.m	
Algorithm files in Matlab	
INPUT_INIT.m	

INITT.m
FPGAINIT.m
RE_CONTROL1.m
RE_CONTROL2.m
FPGAReloadTables.m
GUI files in Matlab (only for version 7)
FPGACAVITY.fig
FPGACAVITY.m
FPGACTRL.fig
FPGACTRL.m
HARDMON.fig
HARDMON.m
FPGAReadout.fig
FPGAReadout.m

B. List of signal available in DAQ system and for DACs

- *channel 0*: test signal from module *TEST GENERATOR*,
- *channel 1*: external signal *CAV_ OUT_I* ([1] - compare chapter 8.1),
- *channel 2*: internal signal *CAV_ OUT_Q* ([1] - compare chapter 8.1),
- *channel 3*: internal signal *CAV_ DETUN* ([1] - compare chapter 8.1),
- *channel 4*: internal signal *CAV_VMOD* ([1] - see chapter 8.1),
- *channel 5*: internal signal *CTRL_DET_I* ([1] - compare chapter 9.1),
- *channel 6*: internal signal *CTRL_DET_Q* ([1] - compare chapter 9.1),
- *channel 7*: internal signal *CTRL_I* ([1] - compare chapter 9.1),
- *channel 8*: internal signal *CTRL_Q* ([1] - compare chapter 9.1),
- *channel 9*: internal signal *TGAIN_I* ([1] - compare chapter 7.1),
- *channel 10:* internal signal *TGAIN_Q* ([1] - compare chapter 7.1),
- *channel 11*: internal signal *TSETPOINT_I* ([1] - compare chapter 7.1),
- *channel 12*: internal signal *TSETPOINT_Q* ([1] - compare chapter 7.1),
- *channel13* : internal signal *TFEEDFORWARD _I* ([1] - compare chapter 7.1),
- *channel 14*: internal signal *TFEEDFORWARD _Q* ([1] - compare chapter 7.1),
- *channel 15*: internal signal *TBEAM _I* ([1] - compare chapter 7.1),
- *channel 16*: internal signal *TBEAM _Q* ([1] - compare chapter 7.1),
- *channel 17*: internal signal *CAV_MODE1* ([1] - compare chapter 8.1),
- *channel 18*: internal signal *CAV_MODE1D* ([1] - compare chapter 8.1),
- *channel 19*: internal signal *CAV_MODE2* ([1] - compare chapter 8.1),
- *channel 20*: internal signal *CAV_MODE2D* ([1] - compare chapter 8.1),
- *channel 21*: internal signal *CAV_MODE3* ([1] - compare chapter 8.1),
- *channel 22*: internal signal *CAV_MODE3D* ([1] - compare chapter 8.1),
- *channel 23*: input signal *ADC1* ([1] - compare chapter 5.1),
- *channel 24*:input signal *ADC2* ([1] - compare chapter 5.1),

8. 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)

9. References

1. Krzysztof T. Pozniak, Tomasz Czarski, Waldemar Koprek, Ryszard S. Romaniuk - Institute of Electronic Systems, Warsaw University of Technology, "SC Cavity SIMCON ver. 2.1 rev. 1, 02.2005 – User's Manual", XFEL Report 2004-04;
2. Waldemar Koprek, Pawel Kaleta, Jaroslaw Szewinski, Krzysztof T. Pozniak, Tomasz Czarski, Ryszard S. Romaniuk - Institute of Electronic Systems, Warsaw University of Technology, *"*Software Layer for FPGA-Based TESLA Cavity Control System (Part I)*",* TESLA Report, 2004-10
3. T.Czarski, K.T.Pozniak, R.Romaniuk, S.Simrock: "TESLA Cavity Modeling and Digital Implementation with FPGA Technology Solution For Control System Purpose", TESLA Technical Note, 2003-28
4. T.Czarski, R.S.Romaniuk, K.T.Pozniak S.Simrock "Cavity Control System Essential Modeling For TESLA Linear Accelerator", TESLA Technical Note, 2003-08
5. 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, Bellingham, WA, USA;
6. http://www.mathworks.com