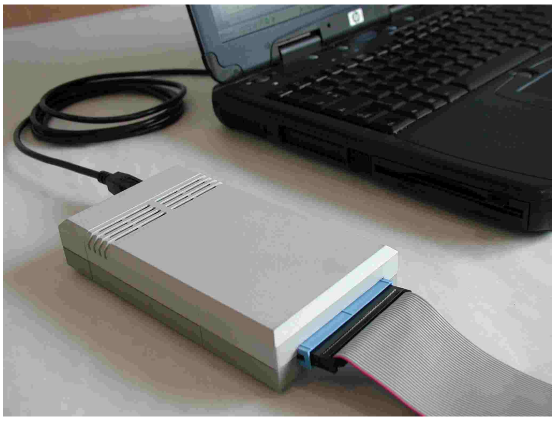
INTECO
Krakow, POLAND, inteco@kki.krakow.pl
www.inteco.com.pl

RT-DAC/USB

I/O Board

Board version 1.11



User's Manual

Kraków 2004

Table of contents

NOTES

1. GENERAL INFORMATION

The RT-DAC/USB is a multifunction analog and digital I/O board dedicated to real-time data acquisition and control in the Windows 95/98/NT/2000/XP environments. The board contains a Xilinx® FPGA chip. All boards are built as the OMNI version. It means the boards can be reconfigured to introduce a new functionality of all inputs and outputs without any hardware modification.

The default configuration of the FPGA chip accepts signals from incremental encoders and generates PWM outputs, typical for mechatronic control applications and is equipped with the general purpose digital input/outputs (GPIO), A/D and D/A converters, timer, counter and signal generator.

The RT-DAC/USB board is distributed in two versions: analog and digital (RT-DAC/USB) and digital only (RT-DAC/USB-D) .This manual contains description for the both versions. In the case, if any facility of the board relates to one version only, this fact is clearly marked.

1.1 Specification

Analog section (not available in the digital version)

Analog Inputs

Channels:	16 single-ended, multiplexed
Resolution:	12 bit
Input ranges:	±10V, programmable gain (x1, x2, x4, x8, x16)
Conversion time:	5.4 µs
Trigger:	all the A/D channels are scanned automatically when USB host requires data
Reference voltage:	on-board

Analog Outputs

Channels:	4
Resolution:	12 bit / 14 bit
Output range:	± 10V, ± 5V
Settling time:	10 µs (to 0.01%)
Reference voltage:	on-board

Digital section (version 1.11)

Digital Input/ Output

Channels:	26 bi-directional, direction setting; 4 channels shared with PWM outputs; 3 channels shared with generator block; 12 channels shared with incremental encoder inputs, 2 shared with counters
Direction:	bi-directional, individually software programmable
Input voltage:	V_{IH} = 2.0V÷3.6V, V_{IL} = - 0.5V÷0.8V
Output voltage:	V_{OH} = 2.4V (min), V_{OL} = 0.4V (max)
Output current:	2mA per channel
Standard:	LVTTL

Digital Timer/Counter

32 bit timer / counter :	2 channels, counts internal clock signal or external impulses . External pulse duration: min 50 μs

Digital Signal Generator

Channels:	1
Resolution:	32 bits of the H state; 32-bits of the L state
Max frequency:	20 MHz

Duty cycle	Software configurable
No of impulses	Software configurable or infinite generation mode
Trigger	Software or hardware
Gate signal	Software or hardware

PWM Outputs

Channels:	4
Resolution:	8/12 bits (software selected)
Base frequency:	programmable, initial 16-bits divider

Incremental encoders

Channels:	4
Output:	32 bit counter
Index	Software configured. 2 modes: with and without index, selectable active level of the index signal

USB features

USB 1.1, 2.0 full speed specification compliant.

The block diagram of the RT-DAC/USB board is shown in Fig. 1.1. The block diagram of the digital version is presented in Fig. 1.2.

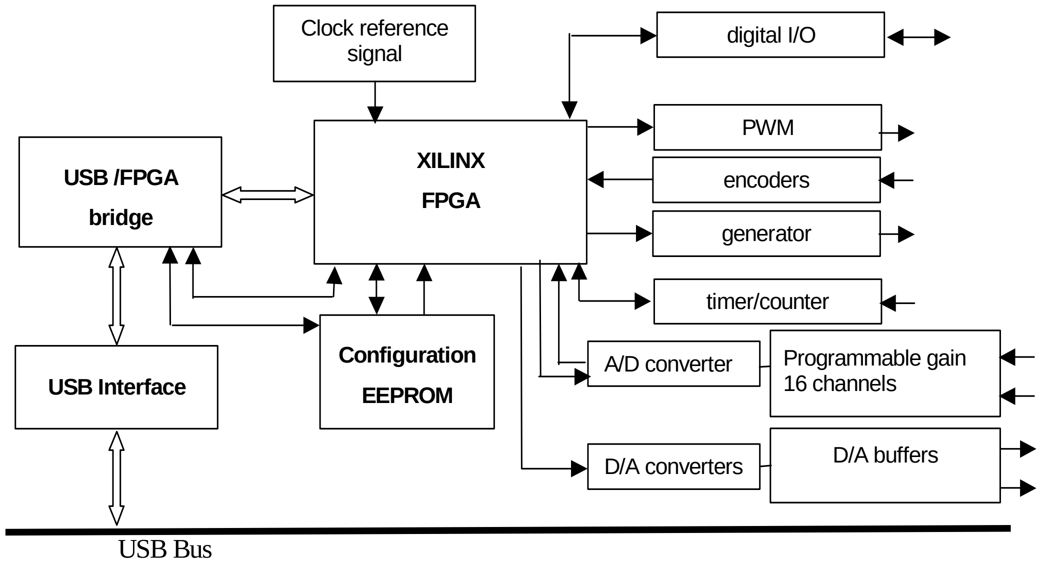


Fig. 1.1. General block diagram of the RT-DAC/USB board

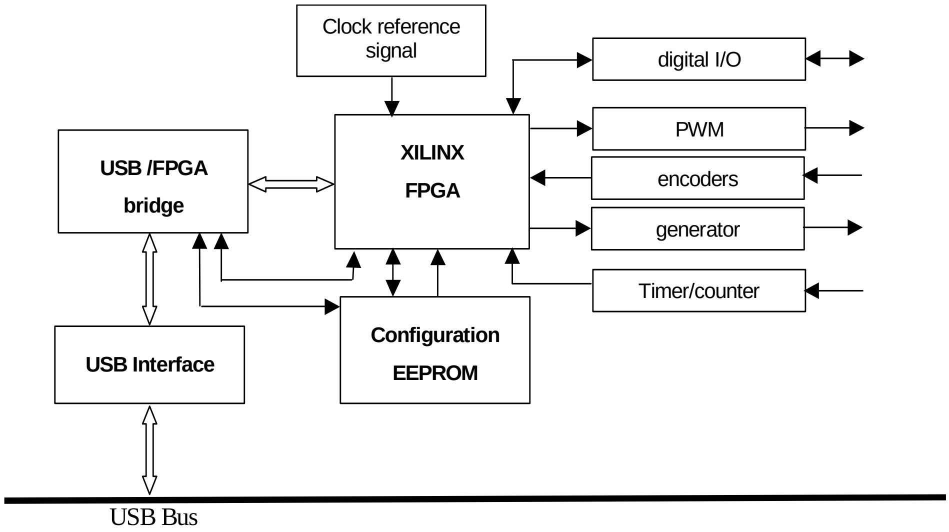


Fig. 1.2. General block diagram of the RT-DAC/USB-D board

The board is equipped with the 12-bit successive approximation A/D converters that give the 5 mV resolution within the input range ±10V. A finer resolution can be achieved by the gain definition using digitally programmable analog amplifier. The A/D conversion time of the RT-DAC/USB board is equal to 5.4 µs.
The board contains four 12-bits D/A converters connected to four analog output channels (optionally 14-bit D/A converters are available). All channels can be hardware configured to operate in the ± 10V or ± 5V mode. Each analog output channel can sink up to 10 mA.
There are 26 digital I/O lines at the RT-DAC/USB board. Digital I/O lines are LVTTL compatible. The direction of each digital I/O can be configured independently.
The default configuration of the RT-DAC/USB includes four PWM outputs and four input channels of the incremental encoders. The PWM outputs and encoders inputs turn the PC into a digital controller to be used in control of manipulators, servo systems, etc.
The digital signal generator can be applied to generate signals with an arbitrary duty cycle, frequency and number of pulses.
Reprogramming the XILINX FPGA chip at the boards can change functions of the board.

The board is equipped with two 40-pin ribbon cable connectors (CN1 and CN2), USB cable socket, 9V DC power socket and three signalling diodes. The digital version of the board is equipped with one 40-pin connector only (CN1). The detailed block diagram of the RT-DAC/USB board (including the connectors) is given in sections 3.

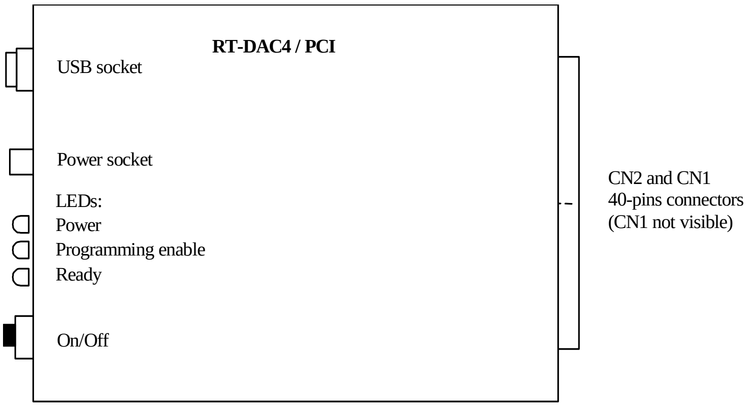
The next section contains the basic information necessary to install and test the board.
The information and specification how to reprogram XILINX FPGA is not included in this guide. Please, relate to *RTDAC/USB FPGA Programming Guide* distributed by INTECO separately.

2. BOARD INSTALLATION

The RT-DAC/USB setup contains:

- RT-DAC/USB board,
- Two 40-pin ribbon cable (only one cable when the digital version is distributed),
- USB cable
- CD containing a software and e-manuals,
- terminal wiring board (optional)
- 9V-12V DC / 4W stabilised power supply (optional). The plug dimensions are given in Fig. 2.2.

The layout of the RT-DAC/USB board is presented in Fig. 2.1



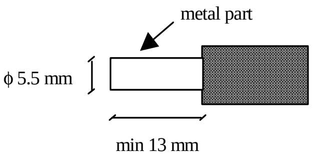


Fig. 2.1. The layout of the RT-DAC/USB board

Fig. 2.2. The plug of the DC power supply

The CN1 connector is not visible at the Fig. 2.1 because it is placed below the CN2 connector. The digital version of the board is equipped with the CN1 connector only. The CN2 connector contains pins connected to A/D inputs and D/A outputs only.

The *Power* signalling LED is emitting light when the *On/Off* switch is on, *Ready* LED indicates that communication between RT-DAC/USB board and computer is running and *Programming Enable* LED is emitting light when the board is ready to be programmed.

To install the board:

- install driver for the board (see below or CD:\DRIVER\readme*.txt),
- install RT-CON package,
- connect the board to the computer using the included USB cable,
- connect the national version of the DC 9V-12V stabilised power supply (not included). 9V DC voltage is recommended.
- test the board (see section 6).

3. DRIVER INSTALLATION

The driver for RTDAC/USB board has to be installed. The user with administrator privileges must install the drivers for Windows 2000 and Windows XP.

Windows 98/2000/XP installation

1. Start Microsoft Windows 98/2000/XP
2. Connect the RTDAC/USB device and turn power ON
3. System detects a new USB device
4. Select the file CD:\driver\ftd2xx.inf and then press OK

4. CONNECTOR PIN ASSIGNMENT

The digital version the RT-DAC/USBD is equipped with one 40-pin I/O connector CN1. The pin assignment of the connector is shown in Fig. 4.1.

CN1

DIO - Digital Input/Output
GND - Ground
ENC – Encoders signals
GEN – Generator signals
CNT – Counters/Timers
PWM – PWM Outputs

PWM0 / DIO0	1○	○2	GND
PWM1 / DIO1	3○	○4	GND
PWM2 / DIO2	5○	○6	GND
PWM3 / DIO3	7○	○8	GND
GEN0 / DIO4	9○	○10	GND
GEN0_TR / DIO5	11○	○12	GND
GEN0_GATE / DIO6	13○	○14	GND
DIO7	15○	○16	GND
DIO8	17○	○18	GND
DIO9	19○	○20	GND
ENC0_A / DIO10	21○	○22	DIO11 / ENC0_I
ENC0_B / DIO12	23○	○24	DIO13 / CNT0
ENC1_A / DIO14	25○	○26	DIO15 / ENC1_I
ENC1_B / DIO16	27○	○28	DIO17 / CNT1
ENC2_A / DIO18	29○	○30	DIO19 / ENC2_I
ENC2_B / DIO20	31○	○32	DIO21
ENC3_A / DIO22	33○	○34	DIO23 / ENC3_I
ENC3_B / DIO24	35○	○36	DIO25
GND	37○	○38	GND
+5V	39○	○40	+3.3

Fig. 4.1. RT-DAC/USB I/O CN1 connector

The analog & digital version of the RT-DAC/USB board is equipped additionally in the CN2 40-pin connector. The pin assignment of the connector is shown in Fig. 4.2.

CN2

A/I – A/D Input
A/O – D/A Output

A/I 0	1○	○2	GND
A/I 1	3○	○4	GND
A/I 2	5○	○6	GND
A/I 3	7○	○8	GND
A/I 4	9○	○10	GND
A/I 5	11○	○12	GND
A/I 6	13○	○14	GND
A/I 7	15○	○16	GND
A/I 8	17○	○18	GND
A/I 9	19○	○20	GND
A/I 10	21○	○22	GND
A/I 11	23○	○24	GND
A/I 12	25○	○26	GND
A/I 13	27○	○28	GND
A/I 14	29○	○30	GND
A/I 15	31○	○32	GND
A/O 0	33○	○34	GND
A/O 1	35○	○36	GND
A/O 2	37○	○38	GND
A/O 3	39○	○40	GND

Fig. 4.2. RT-DAC/USB I/O CN2 connector

The RT-DAC/USB is equipped with 16 multiplexed analog inputs located at the CN2 connector. They are denoted as A/I 0 up to A/I 15. The output of the analog multiplexer is connected to the input of the digital programmable analog amplifier. The board is also equipped with four 12-bit D/A converters. These outputs are denoted as A/O 0 up to A/O 3.

Only 26 pins at the CN1 connector are used as digital I/O signals (denoted as DIO0 to DIO25). The remaining pins are the ground and power (GND, +5V, +3.3V). The general purpose digital I/O signals can be separately configured to be either the input or output.

As the number of digital I/O signals is limited, 21 of them are shared with the signals of the specialized blocks. The specialized blocks are implemented as the modules in the on-board FPGA structure. It means that **the functions of the specialized block are hardware-implemented**. The specialized blocks are:

- PWM generators – there are four PWM blocks. The outputs are denoted: *PWM0*, *PWM1*, *PWM2* and *PWM3*,
- digital signal generator – The output is denoted *GEN0*. The generator contains input trigger signal (*GEN0_TR*) and input gating signal (*GEN0_GATE*),

- incremental encoders – the device contains four incremental encoder channels. Each channel requires three input signals – wave A (*ENC0_A*, *ENC1_A*, *ENC2_A* and *ENC3_A*), wave B (*ENC0_B*, *ENC1_B*, *ENC2_B* and *ENC3_B*) and index (*ENC0_I*, *ENC1_I*, *ENC2_I* and *ENC3_I*),
- counters – there are two counters available. The input signals are denoted *CNT0* and *CNT1* respectively.

There are two kinds of specialized blocks:

- the first kind of the specialized blocks contains digital output signals (PWM and the digital signal generator). In this case the appropriate pins of the CN1 connector can operate as the general purpose digital I/O signals or as the output of the specialized block. The operating mode is determined by a mode configuration register (ModConfReg). If they operate as general purpose digital I/Os their directions and states are determined by the software. If they operate as the outputs of the specialized blocks the state of the output is controlled by the block. The states of the output signals can be read by the software, (the software can check the PWM output),
- the second kind of the specialized blocks contains only the digital input signals (the incremental encoders and counters). In this case it is not necessary to select the operating mode of the encoder and counter signals. If the appropriate general purpose I/O signals are configured to be the inputs then their states can be read by the software and simultaneously the signals excites the specialized block (see Fig. 4.3). If the shared general purpose I/O signals are configured to be outputs their states can be set by the software and simultaneously the signals excites the specialized block (see Fig. 4.4). This operating mode can be applied for testing of the specialized blocks – for example the encoder can be excited and read in a software manner. This testing strategy is not significant during common device applications. It may be very useful when a user wants to design and test his own FPGA blocks (see *"USB Device XILINX Programming Guide"*)

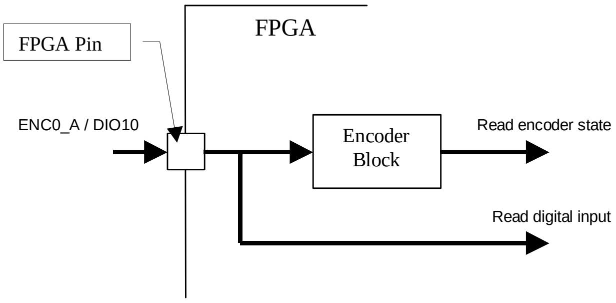


Fig. 4.3. The shared FPGA pin configured as the input

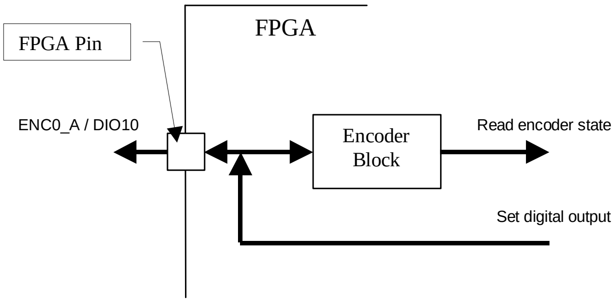


Fig. 4.4. The shared FPGA pin configured as the output

5. USB ACCESS FUNCTIONS

The RT-DAC/USB access functions are defined in the *rtdacusb.c* and the *rtdacusb.h* files. These files contain the API macro definitions and C API functions referred to the description of the board functions. In programs, which use API functions, user must include ftd2xx.h header and link ftd2xx.lib/ftd2xx.dll libraries.
Note, the *rtdacusb.c*, *rtdacusb.h*, *ftd2xx.h*, *ftd2xx.lib and ftd2xx.dll* files are accessible after installation of the *RT-DAC/USB* software.

The speed of the signals transferred between the PC and USB device depends on the way of communication. If communication is a one-way direction it lasts 1 millisecond (writing) or two milliseconds (reading). The transfer time does not depend on the size of a message up to the 4KB data frame. Otherwise, if the PC asks USB device and waits for answer the time to exchange the messages is twice longer than before. Minimal time period to transfer a message (USB time slot) is equal to 1 millisecond.

The special binary buffer placed at the board, in the FPGA chip, contains all information of the board. In this buffer are kept measurements and all settings of the board. The FPGA logic manages the binary buffer.
The data stored in the buffer can be read from the buffer as well as data can be written to the buffer.

Because the contents of the buffer is less than 4kB the communication with the board is realized by two main functions:

- the first reads binary buffer from USB board
- the second writes data to the binary buffer at USB board.

Using these two functions one can maintain all features of the USB board. From computer side the buffer is seen as a sequence of the bytes however the communication functions see the buffer as a nested structure of C language. These structures have the form:

```
#define BINARY_BUFFER_SIZE   (1024)
#define NO_OF_PWM            (4)
#define NO_OF_ENCODER        (4)
#define NO_OF_TMRCNT         (2)
#define NO_OF_GENERATOR      (1)
#define NO_OF_AD             (16)
#define NO_OF_DA             (4)

typedef struct {
        unsigned int Mode;
        unsigned int Prescaler;
        unsigned int Width;
} PWMType;

typedef struct {
        unsigned int Reset;
        unsigned int IdxActive;
        unsigned int IdxInvert;
        long int    Counter;
} EncoderType;

typedef struct {
        unsigned int St1Len;
        unsigned int St2Len;
        unsigned int NoOfPeriods;
        int       Enable;
        int       SwHwGateStartFlag;
        int       SwGate;
        int       SwStart;
        int       InvertGate;
        int       InvertStart;
```

```
            int       InfiniteGeneration;
            int       St1Level;
            int       InitLevel;
            int       TerminateLevel;
} GeneratorType;

typedef struct {
            unsigned int Reset;
            unsigned int Mode;
            unsigned int Counter;
} TmrCntType;

typedef struct {                    // not active in  digital version of the board
            unsigned int Gain;
            unsigned int Result;
} ADType;

typedef struct {
            unsigned int LogicVersion;
            char        ApplicationName[6];
            unsigned int CN1PinMode;
            unsigned int CN1Direction;
            unsigned int CN1Output;
            unsigned int CN1Input;
            PWMType      PWM[ NO_OF_PWM ];
            EncoderType  Encoder[ NO_OF_ENCODER ];
            TmrCntType   TmrCnt[ NO_OF_TMRCNT ];
            GeneratorType Generator[ NO_OF_GENERATOR ];
            ADType       AD[ NO_OF_AD ];               // not active in digital version of the board
            unsigned int  DA[ NO_OF_DA ];              // not active in digital version of the board
} RTDACUSBBufferType;
```

Before using the USB board the user has to initialise the communication with the board. Next, the board management is performed as follows:

- a contents of the *RTDACUSBBufferType* buffer is read,
- then an appropriate fields of the structure are set. Each field of the structure corresponds to a certain function of the board. The meaning of each field is described in the next sections.
- reading of the buffer allows to get measurements,
- writing to the buffer sets a new state of the device.

Closing the USB board finishes the current session of communication.

5.1 USB functions

int USBOpen()

DESCRIPTION
The function opens the connected RTDAC-USB device and must be invoked at the beginning before any of the USB operations. The function returns the number of connected RT-DAC-USB devices. If an error occurs or none of RTDAC-USB are connected the function returns a number equal to or less than zero.

ARGUMENTS
None

int USBClose()

DESCRIPTION
The function closes the connected RTDAC-USB device and should be called when all operations are finished.

ARGUMENTS
None

int USBLastError(char *Dsrp)

DESCRIPTION
The function returns a string containing the last error description. The description is returned as the *Drsp* reference. String "FT_OK" means that none error occurred during the last USB command execution.

ARGUMENTS

Dsrp	pointer to the string

int SetTimeouts(int RecTimeout, int SendTimeout);

DESCRIPTION
The function sets timeouts for operations of reading *(RecTimeout)* and writing *(SendTimeout)*. It is necessary to check if the time of these operations do not exceed the fixed limit.

ARGUMENTS

RecTimeout	32-bit integer
SendTimeout	32-bit integer

NOTES
RecTimeout and *RecTimeout* are in milliseconds.

int CommandSend_0111(RTDACUSBBufferType *Buff);

DESCRIPTION
The function sends the data placed in the structure to the buffer at the USB board.

ARGUMENTS

Buff	pointer to the structure

int CommandRead_0111(RTDACUSBBufferType *Buff);

DESCRIPTION
The function reads data from the buffer at the USB board and place data in the structure. The data are returned as the *Buff* reference.

ARGUMENTS

Buff	pointer to the structure

5.2 Description of the fields of the RTDACUSBBufferType structure

If the binary buffer is written every field of the structure sets a corresponding feature or a function of the board. If the binary buffer is read every field of the structure contains a current state of the corresponding feature or the function of the board. For this reason description of the fields contains “sets/means” phrase.

unsigned int LogicVersion;

DESCRIPTION
The field contains a number of the logic version applied in the FPGA chip. This field is read-only.

char ApplicationName[6];

DESCRIPTION
The field contains string of the application name (“USB01”for digital version of the RTDAC/USB). This field is read-only.

EXAMPLE:

```
RTDACUSBBufferType  RTDACUSBBuffer;

NoOfDetectedUSBDevices = USBOpen( );

if( NoOfDetectedUSBDevices <= 0) {
          printf ( "Can not detect any RT-DAC/USB device\n\n");
          printf(  "Application is not able to start\n" );
                                            }
else {
CommandRead_0111( &RTDACUSBBuffer );

 printf("Number of detected RT-DAC/USB devices: %d         "
                "Logic version: %04X / %s",
                 NoOfDetectedUSBDevices, RTDACUSBBuffer.LogicVersion,
                                  RTDACUSBBuffer.ApplicationName );
}            }
```

5.3 Operating mode of the shared pins (output signals)

Some pins are shared between the digital I/O lines and outputs of the specialized blocks. When a shared pin works as an output a problem may occur. It has to be determined whether the pin is controlled by a general purpose digital I/O or by a PWM block or a GEN0 block. For this reason the *CN1PinMode* field is applied. The field sets the operation mode of the shared pins.

unsigned int CN1PinMode

DESCRIPTION
The field sets/means a mode of the shared pins. Data determines the source of the output signals. Five bits corresponds to the GEN0, PWM3, PWM2, PWM1 and PWM0 blocks. If a bit is set to zero it means the pin is defined as the output of the general purpose digital I/O. If a bit is equal to “1” the corresponding pin is defined as the output of a specialized block.

5.4 Counter/timer

RT-DAC/USB contains two 32-bit timer/counter channels. In the timer mode the timer/counter channels count pulses of the internal board clock. The frequency of the clock corresponds to the board version (40 MHz is the default frequency value). In the counter mode the timer/counter channels count external pulses respectively from the CNT0 and CNT1 inputs. In the timer mode the blocks does not use any of external signals.
Both counter/timer channels can operate either in the counter or timer modes. The features of these channels are controlled by the *TmrCntType* substructure. The counter/timer value can be reset by setting the active state of the *TmrCntType.Reset* field.

In the counter mode the counter inputs are denoted: DIO13/CNT0 and DIO17/CNT1 and are located at the CN1 connector at pins 24 and 28. The inputs of the counters CTN0 and CNT1 are shared with the general purpose digital I/Os DIO13 and DIO17. If the DIO13/CNT0 or DIO17/CNT1 signals are defined to be the inputs, their states can be read (typically as the digital input) and simultaneously they excite the respective counter block. If the signals are set to be outputs, their states are determined by software (typically as the digital output) and simultaneously they excite the respective counter block (this operating mode can be applied to test the blocks in a programming manner).

Counter/timer fields:

The TmrCntType substructure controls the counter/timer features and has the form:

```
typedef struct {
        unsigned int Reset;
        unsigned int Mode;
        unsigned int Counter;
} TmrCntType;
```

unsigned int Reset;

DESCRIPTION
Its value equal to 1 sets/means that the counter is reset; 0 means the working mode. **The *Counter* field remains equal to 0 until the R*eset* field is equal 1.**

unsigned int Mode;

DESCRIPTION
The field sets/means the operating mode of the timer/counter. Its value equal to 0 means the counter mode, 1 means the timer mode.

unsigned int Counter;
DESCRIPTION
32-bit unsigned integer value of the counter. The field is read-only.

NOTES
In the counter mode the minimum width of the H and L states of the input wave is equal to 50ns each.

EXAMPLE
Set the first channel as the timer and the second channel as the counter. Reset both channels, start counting and read the values of both channels.

```
RTDACUSBBufferType  RTDACUSBBuffer;
unsigned int Input;
long int  Result;
```

```
NoOfDetectedUSBDevices = USBOpen( );

        if( NoOfDetectedUSBDevices <= 0) {
          printf( "Can not detect any RT-DAC/USB device\n\n"
                        "Application is not able to start" );
        }
        else {

        CommandRead_0111( &RTDACUSBBuffer );

         printf("Number of detected RT-DAC/USB devices: %d          "
                       "Logic version: %04X / %s",
                        NoOfDetectedUSBDevices, RTDACUSBBuffer.LogicVersion,
                                        RTDACUSBBuffer.ApplicationName );
        }

                RTDACUSBBuffer.TmrCnt[0]..Mode = 1;  // Set channel 0  as  timer
                RTDACUSBBuffer.TmrCnt[1].Mode = 0; // Set channel 1 as counter

                RTDACUSBBuffer.TmrCnt[0].Reset = 1;  // Reset timer
                RTDACUSBBuffer.TmrCnt[1].Reset = 1; // Reset counter

        CommandSend_0111( &RTDACUSBBuffer );

                RTDACUSBBuffer.TmrCnt[0].Reset = 0;  // Start  timer
                RTDACUSBBuffer.TmrCnt[1].Reset = 0; // Start counter



        // WAIT A TIME PERIOD …

        CommandRead_0111( &RTDACUSBBuffer );
        Result = RTDACUSBBuffer.TmrCnt[ 1 ].Counter
        printf("Counter value %d\n", Result );
        Result = RTDACUSBBuffer.TmrCnt[ 0 ].Counter
        printf("Timer value %d\n", Result );

USBClose();
```

5.5 Digital I/O

The RT-DAC/USB board contains 26 digital input/output lines denoted as DIO0, DIO2,…,DIO25. The digital I/O lines are connected to pins of the CN1 connector. The general purpose digital I/O signals can be separately configured to become either inputs or outputs. Only five lines (DIO7, DIO8, DIO9, DIO21 and DIO25) work independently, the remaining lines are shared with the signals of the specialized blocks. The direction of each line can be set independently using the field *CN1Direction*.

There are two ways to define the direction of the I/O lines. The first one is used if the line is working independently and the second one is applied if the line is shared with the PWM or GEN0 specialized blocks.
If a line is working as the general purpose digital I/O we define direction using *CN1Direction* field. The line becomes the input or output. Next we can read or set the state of the digital I/O line. If the line is shared with any specialized block and works as the input, the procedure is the same. However, if the line works as the output, at first we have to use *CN1PinMode* field to disconnect the PWM or GEN0 specialized blocks and after that we set the line as the output.

Digital I/O fields

unsigned int CN1Direction;

DESCRIPTION
The field sets/means digital I/O lines as inputs or outputs. At the beginning every shared line must be set as GPIO (see *CN1PinMode* field*)*. Only 26 bits are used. The last significant bit is referred to DIO0. If a bit is set to 0 it means that the digital I/O is set as the output. If the bit is equal to 1 the digital I/O is set as the input

unsigned int CN1Input;

DESCRIPTION
The field contains values of digital input lines. Only 26 bits are used. The last significant bit is referred to DIO0.

unsigned int CN1Output;

DESCRIPTION
The field sets values of digital output lines. Only 26 bits are used. The last significant bit links to DIO0.

EXAMPLE

To set DIO0-DIO15 lines as the outputs and to set DIO16, DIO17,…,DIO25 as the inputs, set all output lines to logic state '1' and read all 10 inputs, the following C-statements can be executed:

```
RTDACUSBBufferType  RTDACUSBBuffer;
unsigned int Input;

NoOfDetectedUSBDevices = USBOpen( );

      if( NoOfDetectedUSBDevices <= 0) {
        printf( "Can not detect any RT-DAC/USB device\n\n"
                   "Application is not able to start" );
      }
      else {
      CommandRead_0111( &RTDACUSBBuffer );

       printf("Number of detected RT-DAC/USB devices: %d       "
                     "Logic version: %04X / %s",
                      NoOfDetectedUSBDevices, RTDACUSBBuffer.LogicVersion,
                                      RTDACUSBBuffer.ApplicationName );
      }

      RTDACUSBBuffer.CN1PinMode = 0; // set all lines as general purpose input/outputs (GPIO)
      RTDACUSBBuffer.CN1Direction =  0x3FF0000;
      RTDACUSBBuffer.CN1Output =  0xFFFF ;

      CommandSend_0111( &RTDACUSBBuffer );

/ wait ….

      CommandRead_0111( &RTDACUSBBuffer );
      Input = RTDACUSBBuffer.CN1Input;
     Input = (Input >> 16) & 0x3FF;


USBClose();
```

5.6 A/D conversion (not active in the digital version of the board)

The board is equipped with the 12-bit successive approximation A/D converter that gives the 5 mV resolution within input range ±10V. A finer resolution can be achieved by the gain definition using the analog amplifier. The A/D conversion time of the RT-DAC/USB board is equal to 5.4 μs. The board logic automatically starts the A/D conversions from all analog inputs when the PC host requires data from the RT-DAC/USB device. There is possible to select individually the analog gain for each analog input channel.
The board contains four 12-bits D/A converters connected to four analog output channels. All channels can operate in the ±10V mode.

A/D input fields

The ADType substructure controls the A/D conversion and has the form:

```
typedef struct {
  unsigned int Gain;
  unsigned int Result;
} ADType;
```

unsigned int Gain;

DESCRIPTION
The field sets/means the gain of the selected A/D channel. The *Gain* is a 3-bit number in the 0 to 3 range. The number 0 means the gain equal to 1, 1 means the gain equal to 2, 2 means the gain equal to 4, 3 means the gain equal to 8. Any other number means the gain equal to 16.

unsigned int Result;

DESCRIPTION
The A/D conversion result (denoted as *Result* variable) is given in the form of a 12-bit (U2 coded) number (a read-only value). The number stored in *Result* can be transferred to *AnalogSignal* value given in volts. The following formula is applied:

```
if ( Result  > 2047 ) Result = Result –4096;
AnalogSignal = (10*Result/2048); // [V]
```

EXAMPLE

To start A/D conversion, set the gain of the third A/D input channel to 1 and read the conversion result, the following C-statements can be executed:

```
RTDACUSBBufferType  RTDACUSBBuffer;
long int  Result;

NoOfDetectedUSBDevices = USBOpen( );

          if( NoOfDetectedUSBDevices <= 0) {
            printf( "Can not detect any RT-DAC/USB device\n\n"
                              "Application is not able to start" );
          }
          else {

          CommandRead_0111( &RTDACUSBBuffer );

           printf("Number of detected RT-DAC/USB devices: %d          "
                            "Logic version: %04X / %s",
                             NoOfDetectedUSBDevices, RTDACUSBBuffer.LogicVersion,
                                                    RTDACUSBBuffer.ApplicationName );
          }

          RTDACUSBBuffer.ADType[2].Gain = 0; // set gain of the third A/D channel to 1

          CommandSend_0111( &RTDACUSBBuffer );

/ wait ….

          CommandRead_0111( &RTDACUSBBuffer );
          Result = RTDACUSBBuffer.ADType[ 2 ].Result
          printf("Output of  the third A/D channel%d\n", Result );
          if ( Result  > 2047 ) Result = Result –4096;
          AnalogSignal = (10*Result/2048); // [V]
          printf("Output of the A/D channel in [V]%d\n", AnalogSignal);
USBClose();
```

5.7 D/A conversion (not active in the digital version of the board)

The board contains four 12-bits D/A converter channels connected to the A/O 0, A/O 1, A/O 2 and A/O 3 pins. All channels can be hardware configured to operate in the ±5V or ±10V mode. Each analog output channel can sink up to 10 mA.

unsigned int DA[NO_OF_DA]

DESCRIPTION
The field sets a value to the selected D/A channel. *NO_OF_DA* is a number in the 0 to 3 range. The value written to the field is a 14-bit number in the natural binary code and two least significant bits of the number are neglected. The value of the output signal (*Vout)* (expressed in volts) corresponds to the number (*DA*) sent to the D/A converter. This value is calculated by the formula:

Vout = -5 + 10 DA[*] / 16384; [V]*, if the range of the output signal is ±10V

or

Vout = -10 + 20 DA[*] / 16384; [V]*, if the range of the output signal is ±5V.

.

EXAMPLE

The operating range of the A/D outputs is set to ±10V. To set the output of the third A/D channel equal to 5V and set the output of the fourth A/D channel to –5V, the following C-statements is executed:

```
RTDACUSBBufferType  RTDACUSBBuffer;

NoOfDetectedUSBDevices = USBOpen( );

        if( NoOfDetectedUSBDevices <= 0) {
          printf( "Can not detect any RT-DAC/USB device\n\n"
                        "Application is not able to start" );
        }
        else {

        CommandRead_0111( &RTDACUSBBuffer );

         printf("Number of detected RT-DAC/USB devices: %d        "
                       "Logic version: %04X / %s",
                        NoOfDetectedUSBDevices, RTDACUSBBuffer.LogicVersion,
                                      RTDACUSBBuffer.ApplicationName );
        }

        RTDACUSBBuffer.DA[2] = 0.75*16384;
        RTDACUSBBuffer.DA[3] = 0.25*16384;

        CommandSend_0111( &RTDACUSBBuffer );

USBClose();
```

5.8 Digital signal generator

The RT-DAC/USB board includes the digital signal generator. The output waves are generated on the basis of the default (40 MHz frequency) wave signal. It means that the resolution of generated pulses is 25 nanoseconds.

The following parameters of the generator can be tuned by the software:

- the generated wave may be triggered in a software manner or by an external signal. The name of the external trigger signal is GEN0_TR,
- the generator can operate in two modes: finite and infinite. The first one generates a defined number of pulses. The second one generates pulses until the generator operation is terminated by the software,
- the generator configuration functions can set the level of the generator output that immediately appears at the output and remains there until the trigger action takes place,
- when a finite number of the pulses is generated also the output level remaining after the generation can be set,
- the software sets the durations of the L and H states of the generated waves independently,
- the generated wave can start with a logical level selected by the configuration function,
- the generated wave may be gated. The external gate signal is denoted as GEN0_GATE. The gating is activated by software. The source (software or hardware) and the level of the gate signal can be defined.

The source of the gate and trigger signals may be both software or hardware. They come either from the external GEN0_TR and GEN0_GATE signals nor they are software stimulated.

The output of the generator GEN0/DIO4 is assigned to pin number 9 of the CN1 connector. The generator trigger signal GEN0_TR/DIO5 is assigned to the 11 pin of the CN1 connector. The generator gating signal GEN0_GATE/DIO6 is located at the 13 pin of the CN1 connector. All the generator signals are shared with the general purpose digital I/Os.

The generator operating mode is determined by the direction register and the mode configuration register. The direction register must set the GEN0/DIO4 signal to become the output, the GEN0_TR/DIO5 to become the input (it is required only if the external triggering is used) and GEN0_GATE/DIO6 to become the input (only if the external gating is required). Next, the configuration register determines that the GEN0/DIO4 signal is associated with the specialized generator block. In this case the state of the generator output is controlled by the generator block.

Digital signal generator fields

The *GeneratorType* substructure controls the digital generator features and has the form:

```
typedef struct {
        unsigned int St1Len;
        unsigned int St2Len;
        unsigned int NoOfPeriods;
        int      Enable;
        int      SwHwGateStartFlag;
        int      SwGate;
        int      SwStart;
        int      InvertGate;
        int      InvertStart;
        int      InfiniteGeneration;
        int      St1Level;
        int      InitLevel;
        int      TerminateLevel;
} GeneratorType;
```

unsigned int St1Len;

DESCRIPTION

length of the first part of the generated periods. The *St1Len* variable means a number of the intervals each one 25 nanoseconds long,

unsigned int St2Len;

DESCRIPTION
length of the second part of the generated periods. The *St2Len* variable means a number of the intervals each one 25 nanoseconds long,

unsigned int NoOfPeriods;

DESCRIPTION
number of the generated periods if the generator does not operate in the infinite mode. Each period consists of two parts. Their lengths are selected by the *St1Len* and *St2Len* arguments,

int Enable;

DESCRIPTION
The flag equal to “1” enables the generator to operate.

int SwHwGateStartFlag;

DESCRIPTION
If it flag is set to “0” it means that software becomes the source of the *Gate* and *Start* signals. If flag is set to “1” hardware becomes the source.

int SwGate , SwStart;

DESCRIPTION
“0” starts signal generation. “1” stops generation.

int InvertGat, InvertStart;

DESCRIPTION
This integer number equal to “1” sets the inverse logic of the Gate and Start signals.

int InfiniteGeneration;

DESCRIPTION
If this integer number is set to “1” it means that the infinite time of generation has been set. If this integer number is set to “0” it means that the number of generated periods is equal to *NoOfPeriods.*

int St1Level;

DESCRIPTION
Sets the level of the generated signal. Numbers 0 or 1 are allowed

int InitLevel, TerminateLevel;

DESCRIPTION
These two integer numbers denote the levels of the signal at the beginning and at the end of the generator pass. The admissible numbers are: 0 or 1.

NOTES
The generation starts after the trigger action.

5.9 PWM

The RT-DAC/USB board includes four output PWM channels denoted: PWM0/DIO0, PWM1/DIO1, PWM2/DIO2 and PWM3/DIO3, located at the CN1 connector at pins: 1, 3, 5 and 7. These pins are shared between PWM outputs and four general purpose digital I/Os. The PWM operating mode is determined by the contents of the direction register and the mode configuration register. The direction register must set the PWM0/DIO0, PWM1/DIO1, PWM2/DIO2 or PWM3/DIO3 signals to become the outputs. The configuration register determines wether the signals are associated with the specialized PWM blocks or operate as the general purpose digital IOs. In the first case the states of the outputs are constructed by the PWM block. In the second case the state of the outputs are defined by the software.

The basic PWM period and the period of the "H" state (width) of each channel are selected independently. The operating principle of the PWM channels is given in Fig. 5.1. The input basic frequency of the PWM channels is set to the default 40MHz value. This frequency is divided by the counter (called the prescaler), which creates the PWM basic period. The basic period wave excites the 8/12-bit counter. The output of the counter is compared to the 8/12-bit duration of the "H" state. The valid prescaler value is a number taken from the range [0 - 65535]. The PWM counters and the "H" state duration registers can operate in either 12 or 8-bit modes. The 8-bit mode allows PWM to operate in high speed and the 12-bit mode allows to achieve higher accuracy of the output.

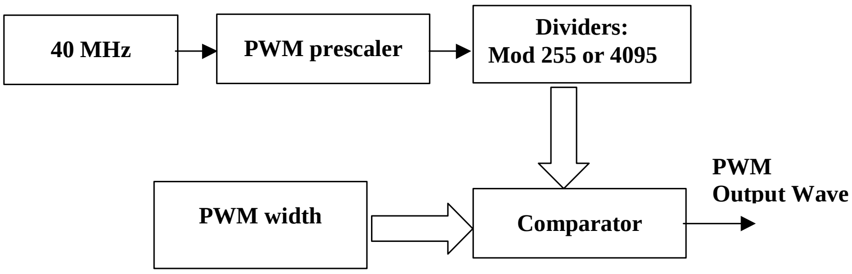


Fig. 5.1. Block diagram of the PWM generator

In the 12-bit mode a single PWM period contains 4095 impulses of the output prescaler. The duration of the 'H' state is set by a number from 0 to 4095. In the 8-bit mode a PWM period contains 255 impulses of the output prescaler. The duration of the 'H' state is set by a number from 0 to 255.
The frequency of the PWM wave is given by the formulas:

$$f_{PWM} = \frac{f_{basic}}{(prescaler+1)*255} \quad \text{for 8-bit mode}$$

$$f_{PWM} = \frac{f_{basic}}{(prescaler+1)*4095} \quad \text{for 12-bit mode}$$

PWM fields

The PWMType type controls the PWM channels and has a form:

```
typedef struct {
        unsigned int Mode;
        unsigned int Prescaler;
        unsigned int Width;
} PWMType;
```

There is the array of elements of the PWMType type. Each element of this array corresponds to one PWM channel. The fields of this type contain the PWM operating mode, prescaler value and the duration (width) of the "H" state. The mode value is placed in the *Mode* field. The *Prescaler* field contains the prescaler value. The *Width* field contains the duration of the "H" state.

unsigned int Mode;

DESCRIPTION
This integer number equal to “0” sets/means 8-bit PWM mode; If this integer value is equal to “1” then the 12-bit PWM mode is set.

unsigned int Prescaler;

DESCRIPTION
Sets/means prescaler of the PWM block (16-bit value),

unsigned int Width;

DESCRIPTION
This 12-bit unsigned value sets the duration of the “H” state in each PWM period. When the PWM channel operates in the 8-bit mode only the least significant 8-bits are applied.

NOTES
The maximum frequency of the PWM output is approximately equal to 156kHz and is generated in the 8-bit mode when the prescaler is equal to 0. The minimum frequency of the PWM output is approximately equal to 0.15Hz and is generated in the 12-bit mode when the prescaler is equal to 65535.

EXAMPLE

For the first PWM channel select the 8-bit operating mode, set the frequency to 300 Hz and set the duty cycle to 25% (the “H” state lasts 25% of the PWM period). For the second PWM channel select the 12-bit operating mode, set the frequency to 10 Hz and set the duty cycle to 75%. Read the mode of the third PWM channel.

```
RTDACUSBBufferType  RTDACUSBBuffer;

NoOfDetectedUSBDevices = USBOpen( );

        if( NoOfDetectedUSBDevices <= 0) {
          printf( "Can not detect any RT-DAC/USB device\n\n"
                          "Application is not able to start" );
        }
        else {

        CommandRead_0111( &RTDACUSBBuffer );

         printf("Number of detected RT-DAC/USB devices: %d          "
                         "Logic version: %04X / %s",
                          NoOfDetectedUSBDevices, RTDACUSBBuffer.LogicVersion,
                                         RTDACUSBBuffer.ApplicationName );
        }

                // Select channel 0 (first channel), mode argument equal to 0 sets the 8-bit mode
                // the prescaler value of 522 defines the 300Hz frequency
                // the width equal to 64 means 25% duty cycle (64 is 25% of 256)
        RTDACUSBBuffer.PWM [0 ].Mode = 0;
        RTDACUSBBuffer.PWM [0 ].Prescaler = 522;
        RTDACUSBBuffer.PWM [0 ].Width = 64;
```

```
                // Select channel 1 (second channel), mode argument equal to 1 sets the 12-bit mode
                // the prescaler value of 60 defines the 10Hz frequency
                // the width equal to 3072 means 75% duty cycle (3072 is 75% of 4096)

        RTDACUSBBuffer.PWM [ 1 ].Mode = 1;
        RTDACUSBBuffer.PWM [ 1 ].Prescaler = 60;
        RTDACUSBBuffer.PWM [ 1 ].Width = 3072;

        CommandSend_0111( &RTDACUSBBuffer );

/ wait ….

        CommandRead_0111( &RTDACUSBBuffer );
        Result = RTDACUSBBuffer.PWM[ 2 ].Mode
        If Result =0  printf("Third PWM mode: 8-bit \n",  );
        else
        printf("Third PWM mode: 12-bit \n",  );


USBClose();
```

5.10 Encoders

RT-DAC/USB includes four 32-bit incremental encoder channels. Each channel counts the changes of two input waves and optionally applies the input index signal to reset the encoder counter. The initial value of each encoder counter can be set to zero in a programmable way or using the active index signal. Encoders can work in two modes: with or without index signals. The software activates and deactivates the index signals and sets the active levels of the index signals as well.

Each encoder channel contains three inputs: wave A, wave B and index I. The respective signals are denoted as:

- ENC0_A, ENC0_B and ENC0_I for the first encoder,
- ENC1_A, ENC1_B and ENC1_I for the second encoder,
- ENC2_A, ENC2_B and ENC2_I for the third encoder and
- ENC3_A, ENC3_B and ENC3_I for the fourth encoder.

All the pins used by the encoder signals are also used by the general purpose digital I/Os. If the encoder signals are defined to be inputs, their states can be read (typically as the digital inputs) and simultaneously they excite the respective encoder block. If the signals are set to be outputs, their states are determined by the software (typically as the digital outputs) and simultaneously they excite the respective encoder block (this operating mode can be applied to test the blocks in a programming manner).

All encoder channels are assigned to the pins at the CN1 connector and are shared with the general-purpose digital input/output signals (see Fig. 4.1). If the direction of the DI/O's are set to be output the appropriate pins operate as the digital outputs. The encoders work if the shared pins are set to be inputs. In such a case they excite the encoder inputs and can be read simultaneously as the digital inputs.

Encoder fields

The EncoderType type controls the encoder channels and has a form:

```
typedef struct {
        unsigned int Reset;
        unsigned int IdxActive;
        unsigned int IdxInvert;
        long int     Counter;
} EncoderType;
```

There is the array of elements of the EncoderType type. Each element of this array corresponds to one encoder channel. The fields of this type contain the Reset value, counter of encoder, flag activating the index input and flag corresponding to the active level of the index signal.

unsigned int Reset;

DESCRIPTION
„1” resets the encoder. „0” starts counting.

unsigned int IdxActive;

DESCRIPTION
“1” activates the encoder index. It means that the active level of the external ENCx_I signal resets the encoder.   “0” deactivates the encoder index.

unsigned int IdxInvert;

DESCRIPTION

„0” sets the high level of the ENCx_I signal to become active. It means that the ENCx_I signal equal to “1” resets the encoder. *IdxInvert* equal to “1” sets the low level of the ENCx_I signal to become active. It means that the ENCx_I signal equal to “0” resets the encoder.

long int Counter;

DESCRIPTION
Counter of the encoder (read-only value).

NOTES
If power is turn on the all reset signals are equal to “1”.

EXAMPLE:

To reset the ENC2 encoder, operate with the index, start counting and read data of the encoder the following C-statements have to be executed:

```
RTDACUSBBufferType  RTDACUSBBuffer;
long int  Result;

NoOfDetectedUSBDevices = USBOpen( );

        if( NoOfDetectedUSBDevices <= 0) {
          printf( "Can not detect any RT-DAC/USB device\n\n"
                         "Application is not able to start" );
        }
        else {

        CommandRead_0111( &RTDACUSBBuffer );

         printf("Number of detected RT-DAC/USB devices: %d          "
                        "Logic version: %04X / %s",
                         NoOfDetectedUSBDevices, RTDACUSBBuffer.LogicVersion,
                                        RTDACUSBBuffer.ApplicationName );
        }

        RTDACUSBBuffer.Encoder[ 2 ].IdxActive = 1
        RTDACUSBBuffer.Encoder[ 2 ].IdxInvert =  0
        RTDACUSBBuffer.Encoder[ 2 ].Reset     =  1

        CommandSend_0111( &RTDACUSBBuffer );

        RTDACUSBBuffer.Encoder[ 2 ].Reset     =  0

        CommandSend_0111( &RTDACUSBBuffer );

/ wait ….

        CommandRead_0111( &RTDACUSBBuffer );
        Result = RTDACUSBBuffer.Encoder[ 2 ].Counter
        printf("Encoder value %d\n", Result );

USBClose();
```

6. TESTING

The software included to the RT-DAC/USB board contains six short programs which allow to test all the functions of the board. These programs works independently and are called separately by their names.

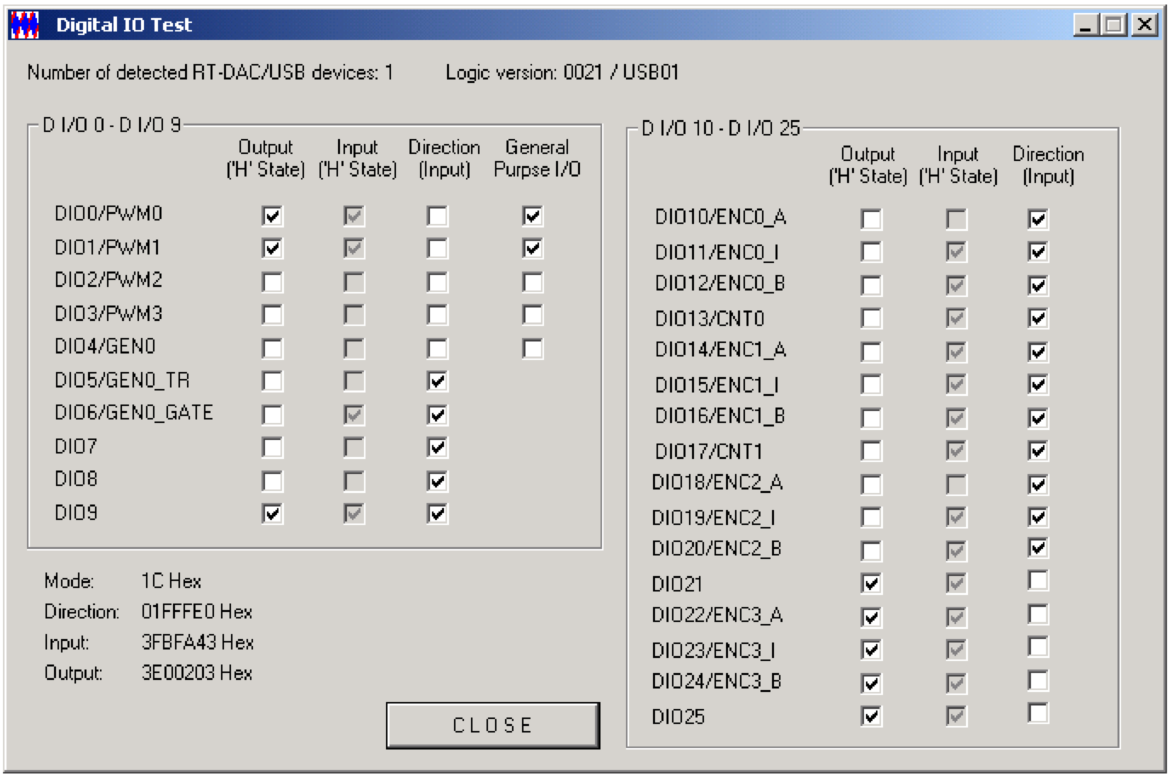
There are the following testing programs:

- DigitalIOTest,
- PWMTest,
- TimCntTest,
- EncoderTest,
- GenTest,
- AnalogTest.

The usage of these programs is self-explained, simple and fully intuitive. However, a short operating instruction for a user may be needed. It is assumed that a user is familiarised with the previous sections of this manual.

6.1 Digital IO Test

After calling the program the screen shown in Fig. 6.1 appears. Information about the states of all digital input/outputs lines is placed in the screen. Also we can choose a direction of each line independently marking the appropriate checkbox.
The checkboxes in the column *Output* allows to set the state of line if it works as the output. The column *Input* (read only checkboxes) shows the state of the line if it works as the input. The column *Direction* allows to set the direction of the line.
As it has been described in the section 5.5 the digital input/output lines denoted: DIO0/PWM0, DIO1/PWM1, DIO2/PWM20, DIO3/PWM3 and DIO4/GEN0 are shared with outputs of the PWM and GEN0 blocks. If a user wants to use only the digital IOs (without the outputs of the specialised blocks) the *General Purpose I/O* checkboxes have to be checked.



Number of detected RT-DAC/USB devices: 1

Logic version: 0021 / USB01

D I/O 0 - D I/O 9

	Output ('H' State)	Input ('H' State)	Direction (Input)	General Purpse I/O
DIO0/PWM0	☑	☑	□	☑
DIO1/PWM1	☑	☑	□	☑
DIO2/PWM2	□	□	□	□
DIO3/PWM3	□	□	□	□
DIO4/GEN0	□	□	□	□
DIO5/GEN0_TR	□	□	☑	
DIO6/GEN0_GATE	□	☑	☑	
DIO7	□	□	☑	
DIO8	□	□	☑	
DIO9	☑	☑	☑	

D I/O 10 - D I/O 25

	Output ('H' State)	Input ('H' State)	Direction (Input)
DIO10/ENC0_A	□	□	☑
DIO11/ENC0_I	□	☑	☑
DIO12/ENC0_B	□	☑	☑
DIO13/CNT0	□	☑	☑
DIO14/ENC1_A	□	☑	☑
DIO15/ENC1_I	□	☑	☑
DIO16/ENC1_B	□	☑	☑
DIO17/CNT1	□	☑	☑
DIO18/ENC2_A	□	□	☑
DIO19/ENC2_I	□	☑	☑
DIO20/ENC2_B	□	☑	☑
DIO21	☑	☑	□
DIO22/ENC3_A	☑	☑	□
DIO23/ENC3_I	☑	☑	□
DIO24/ENC3_B	☑	☑	□
DIO25	☑	☑	□

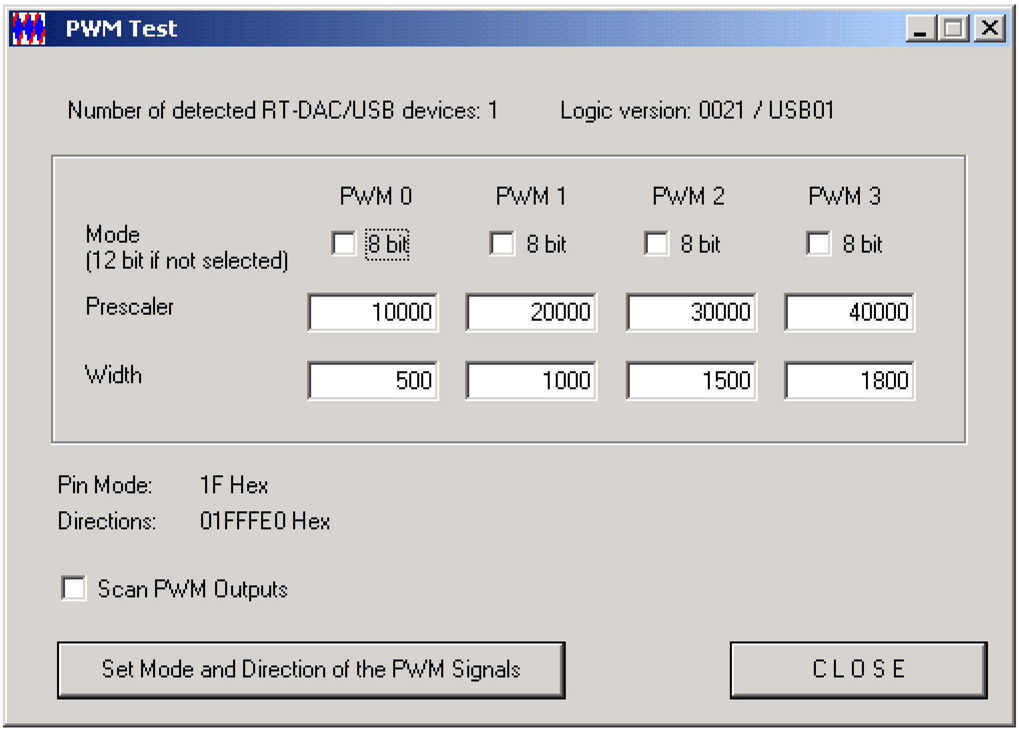
Mode: 1C Hex
Direction: 01FFFE0 Hex
Input: 3FBFA43 Hex
Output: 3E00203 Hex

Fig. 6.1. The *Digital IO Test* window

The analysis of the presented screen will help us to understand the idea of the testing program. In our case the first two lines are set as GPIOs. The column *Direction* for these lines are unchecked. It means that lines are simple outputs and the specialised blocks PWM0 and PWM1 are disconnected. The output states are set to “H”. It can be checked by an oscilloscope, for example. The next three lines are set as the outputs but are not declared as GPIOs. That means these lines are excited by the specialised blocks PWM2, PWM3 and GEN0. The lines denoted as DIO5 up to DIO9 are set as the inputs and their states are defined by the checkboxes in the column *Input (‘H’ State)*, which are read only type. The lines presented in the right hand side of the screen may be interpreted in the similar way.

6.2 PWM Test

The screen of this testing program is shown in Fig. 6.2. At the beginning click *Set Mode and Direction of the PWM Signals* button. This action set mode and direction of the DIO0/PWM0, DIO1/PWM1, DIO2/PWM2 and DIO3/PWM3 lines (shared with digital IOs) as the PWM outputs. From this time a user can observe the PWM outputs using an oscilloscope connected to appropriate pins at CN1 connector.
The all parameters of the PWM channels: *mode*, *prescaler* and *width* can be changed in the presented window and the introduced changes of the PWM outputs can be observed.
Clicking on *Scan PWM Outputs* causes that duty cycles for all channels are automatically changed in a fluent manner. It can be observed in the oscilloscope screen by the user.



Number of detected RT-DAC/USB devices: 1   Logic version: 0021 / USB01

	PWM 0	PWM 1	PWM 2	PWM 3
Mode (12 bit if not selected)	□ 8 bit	□ 8 bit	□ 8 bit	□ 8 bit
Prescaler	10000	20000	30000	40000
Width	500	1000	1500	1800

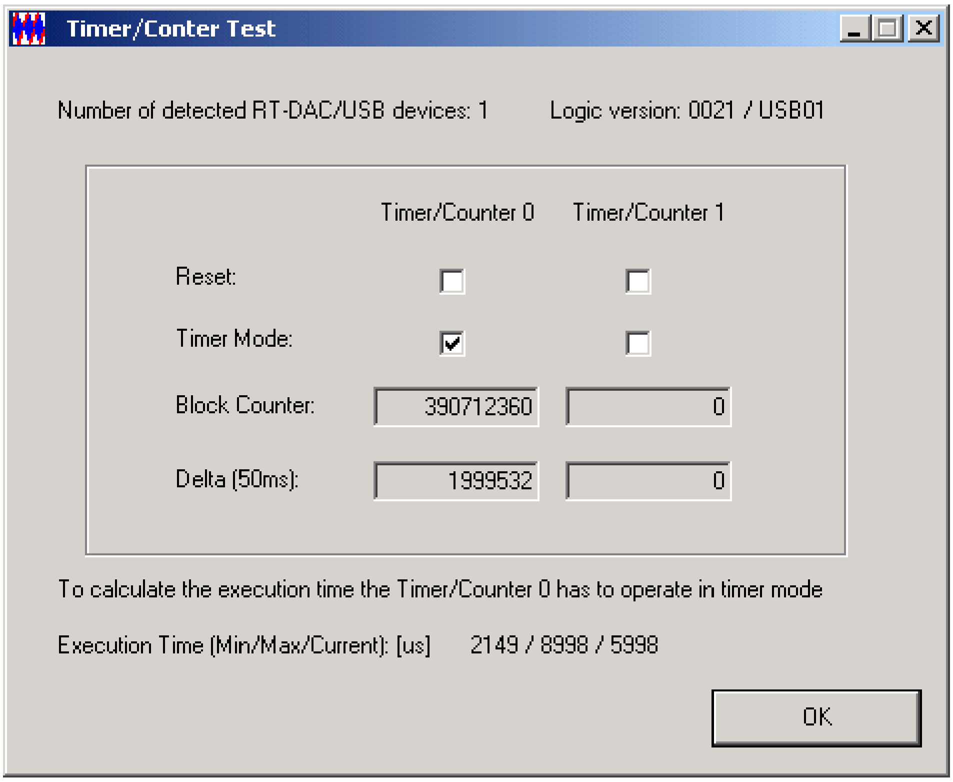
Pin Mode: 1F Hex
Directions: 01FFFE0 Hex

□ Scan PWM Outputs

Fig. 6.2. The *PWM Test* window

6.3 Timer Counter Test

After calling the program the screen shown in Fig. 6.3 appears and the timer/counter channels start. In the given case the first timer/counter channel works as the timer and the second channel works as the counter. The *Reset* checkbox can be used to reset the timer or counter. The timer/counter stops if the *Reset* is checked and starts to operate when *Reset* is unchecked. The counter value is equal to 0 because none of external signals is connected to the counter input. The timer value is different from zero. The *Delta* parameter shows increase of the timer value in the last 50 ms duration. In our case this value is almost equal to 2 000 000 because the basic frequency of the board is equal to 40MHz.



Number of detected RT-DAC/USB devices: 1 Logic version: 0021 / USB01

	Timer/Counter 0	Timer/Counter 1
Reset:	□	□
Timer Mode:	☑	□
Block Counter:	390712360	0
Delta (50ms):	1999532	0

To calculate the execution time the Timer/Counter 0 has to operate in timer mode

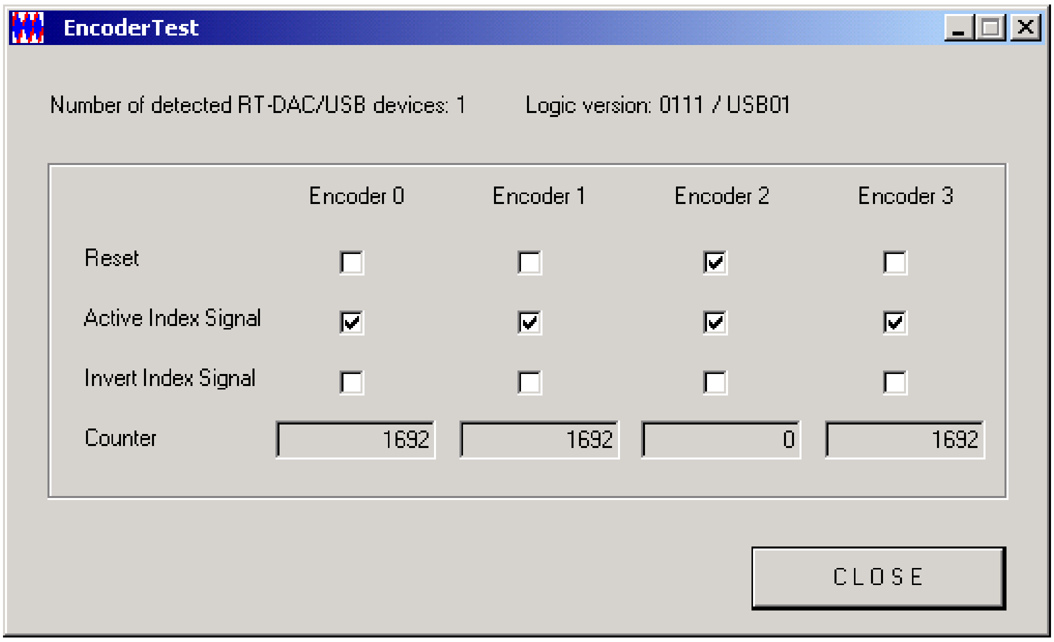
Execution Time (Min/Max/Current): [us] 2149 / 8998 / 5998

Fig. 6.3. The *Timer/Counter Test* window

At the bottom of the screen is placed information how fast the board works in the timer/counter mode. The minimal, maximal and current value of the execution time is shown. Notice, that execution time in the presented case changes from 2,15 millisecond to almost 9 milliseconds.

6.4 Encoder Test

The screen of this testing program is shown in Fig. 6.4. In the window shown below a user can set the



Number of detected RT-DAC/USB devices: 1 Logic version: 0111 / USB01

	Encoder 0	Encoder 1	Encoder 2	Encoder 3
Reset	□	□	☑	□
Active Index Signal	☑	☑	☑	☑
Invert Index Signal	□	□	□	□
Counter	1692	1692	0	1692

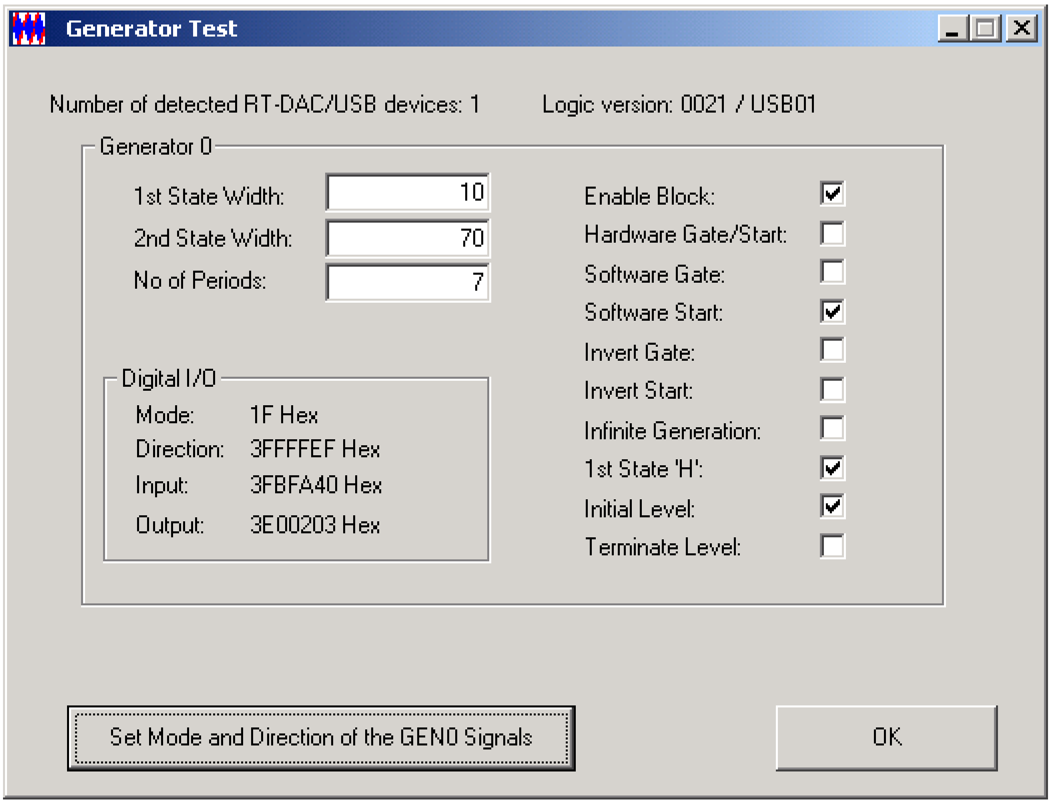
Fig. 6.4. The *Encoder Test* window

parameters of each encoder channels. If the *Reset* checkbox is checked the correspond encoder is reset. The unchecking starts counting of the encoder. The *Active Index Signal* activates and deactivates the index signals. The activating means that the active level of the external ENCx_I signal resets the encoder. The *Invert Index Signal* checkboxes set the active levels of the index signals as well. After connecting the tested encoder to one of the encoder channels the user can check if this channel works properly. The data read from the encoder are visible in the *Counter* field.

6.5 Generator Test

The screen of this testing program is shown in Fig. 6.5. At the beginning click *Set Mode and Direction of the GEN0 Signals* button. This action sets the *DI04* up to *DI06* lines as working in the generator mode. The oscilloscope is needed to observe the output of the generator. In this example the *Enable Block*, *1stState "H"* and *Initial Level* checkboxes are checked. This parameter setting means: the generator can operate, level before generating is "H" and the first state of the generated wave is "H". The values written in the edit fields mean: the first state ("H") of the generated signal is 10 microseconds long, the second state ("L") is 70 microseconds long and the number of the generated signals is seven. The generator starts when *Software Start* checkbox be checked.

A user can change any of the parameters and check the results in the screen of the oscilloscope.



Number of detected RT-DAC/USB devices: 1

Logic version: 0021 / USB01

Generator 0

1st State Width: 10

2nd State Width: 70

No of Periods: 7

Digital I/O

Mode: 1F Hex

Direction: 3FFFFEF Hex

Input: 3FBFA40 Hex

Output: 3E00203 Hex

Enable Block: ☑

Hardware Gate/Start: □

Software Gate: □

Software Start: ☑

Invert Gate: □

Invert Start: □

Infinite Generation: □

1st State 'H': ☑

Initial Level: ☑

Terminate Level: □

Fig. 6.5. The *Generator Test* window

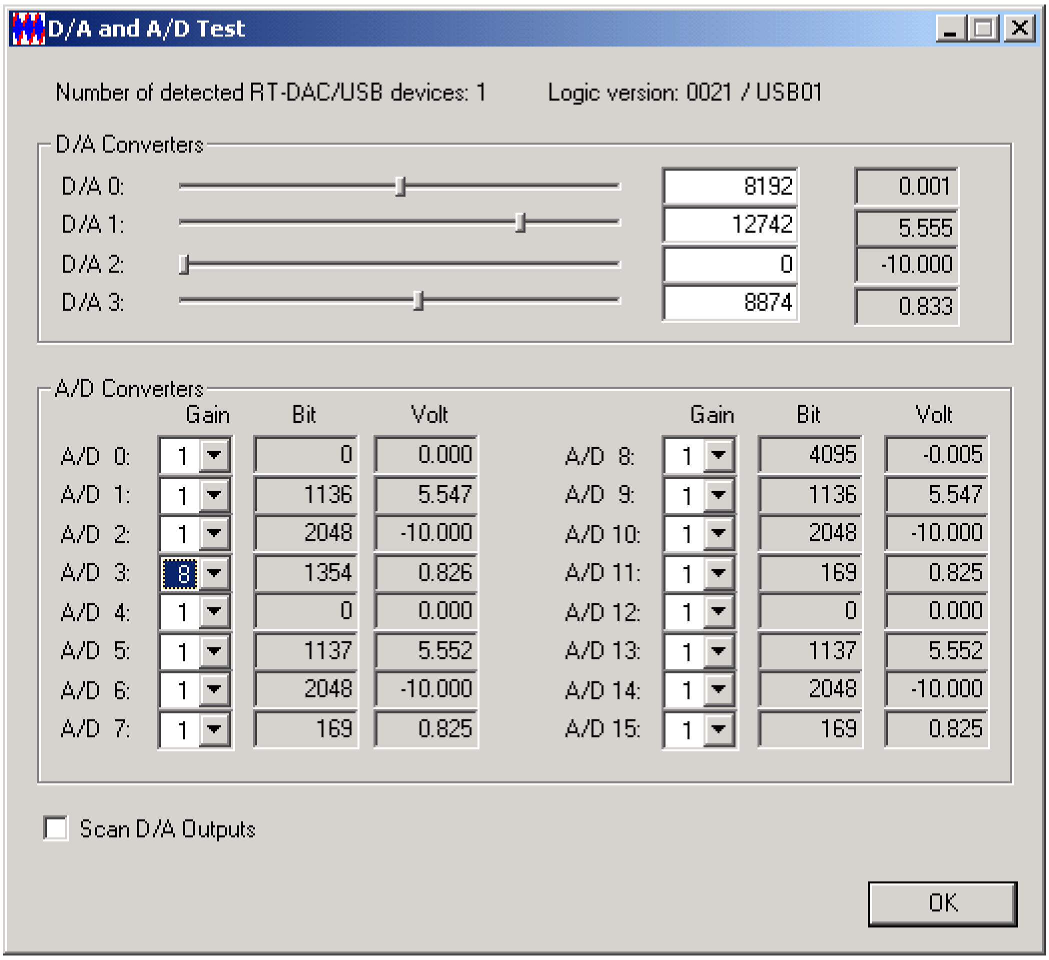
6.6 Analog Test

The screen of this testing program is shown in Fig. 6.4.

In the upper part of the window the user can set the numbers to conversion for the all D/A channels. It can be done by writing a number to the editing field or using the slider corresponding to the chosen channel. The value of the output expressed in volts is shown at the fields placed at the right side of any channel. Clicking on the *Scan D/A Outputs* checkbox causes that data for the all D/A channels are automatically changed in a fluent manner. The output values in volts are changing in the same manner.

In the lower part of the screen the user can set the gains for each A/D channel. The converted external signals expressed in bits and in volts are visible in the corresponding fields.

In the given test the D/A channels (D/A 0 up to D/A 3) are externally connected by the wires to the four A/D channels (A/D 0 up to A/D 3). Note that number 12742 sent to the D/A 1 converter results 5.555 [V] at the output and this value read by the A/D 1 channel is equal to 5.547 [V]. Simultaneously this value expressed in bits is equal to 1136.
Notice the behaviour of the fourth A/D 3 and eight A/D 7 channels. The values expressed in volts i.e. 0,826 and 0,825 are equal the same values expressed in bit but differs 8 times one from another i.e. 1354 and 169. This is due to the chosen gain for the A/D 3 channel.



Number of detected RT-DAC/USB devices: 1   Logic version: 0021 / USB01

D/A Converters

D/A 0:	8192	0.001
D/A 1:	12742	5.555
D/A 2:	0	-10.000
D/A 3:	8874	0.833

A/D Converters

	Gain	Bit	Volt		Gain	Bit	Volt
A/D 0:	1	0	0.000	A/D 8:	1	4095	-0.005
A/D 1:	1	1136	5.547	A/D 9:	1	1136	5.547
A/D 2:	1	2048	-10.000	A/D 10:	1	2048	-10.000
A/D 3:	8	1354	0.826	A/D 11:	1	169	0.825
A/D 4:	1	0	0.000	A/D 12:	1	0	0.000
A/D 5:	1	1137	5.552	A/D 13:	1	1137	5.552
A/D 6:	1	2048	-10.000	A/D 14:	1	2048	-10.000
A/D 7:	1	169	0.825	A/D 15:	1	169	0.825

□ Scan D/A Outputs

Fig. 6.6. The *Analog Test* window