TMS320VC5416 DSK Developing System

1. General Overview

TMS320VC5416 DSK is a static developing system. Allows the users to examine different characteristics of the C5416 DSPs, in order to observe if their requirements are met. The module is an excellent for developing and running programs for DSPs from the family TMS320VC5416.

A simplified scheme of the VC5416 DSK board is presented in figure 1:

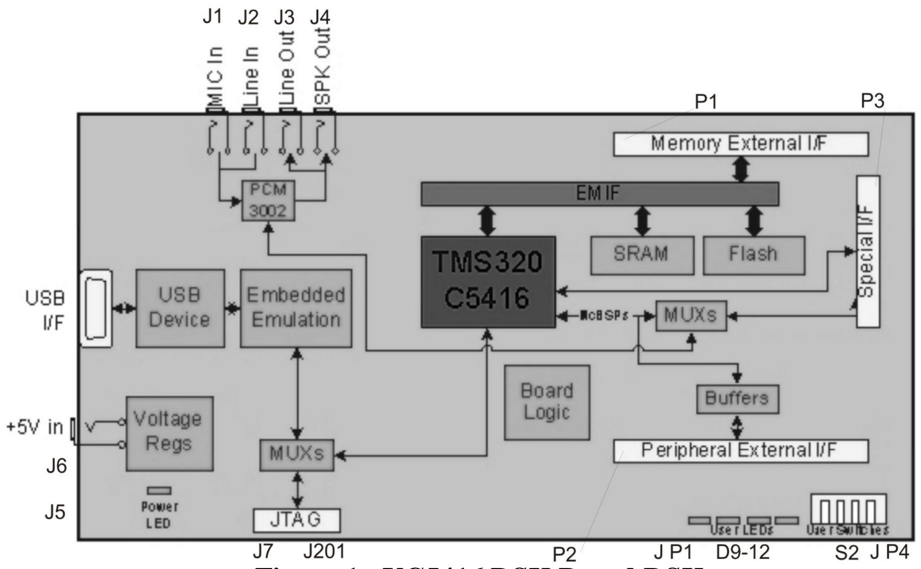


Figure 1 . VC5416 DSK Board DSK

The system has the following characteristics:

- 64k SRAM memory words
- 256k on-board flash ROM words
- 3 extension connectors (memory interface, peripheral interface, Host Port Interface - HPI)
- USB JTAG controller with plug and play drivers
- Stereo Burr Brown PCM 3002 Codec;
- +5V as input;
- VC5416 works a 16-160MHZ.

DSK 5416 is a multi-layer board of 210x115mm fed from an external power source of 5V. The is equipped with an external I/O interface which can have parallel I/O ports and serial-synchronous ports (equipped with buffers) on multiple channels. The important DSK interfaces include interfaces for ROM and RAM, FPGA, codec and expansion.

Four jack stereo correspond to inputs and outputs from the codec.

DSK assures the VC5416 code verification at maximum speed. With 64k memory words, 256 flash ROM memory words and a stereo Burr Brown PCM 3002 codec, the board represents the solution for a wide range of problems.

The central elements of the board are the TMS320VC5416 DSP whose advanced multibus architecture is presented bellow:

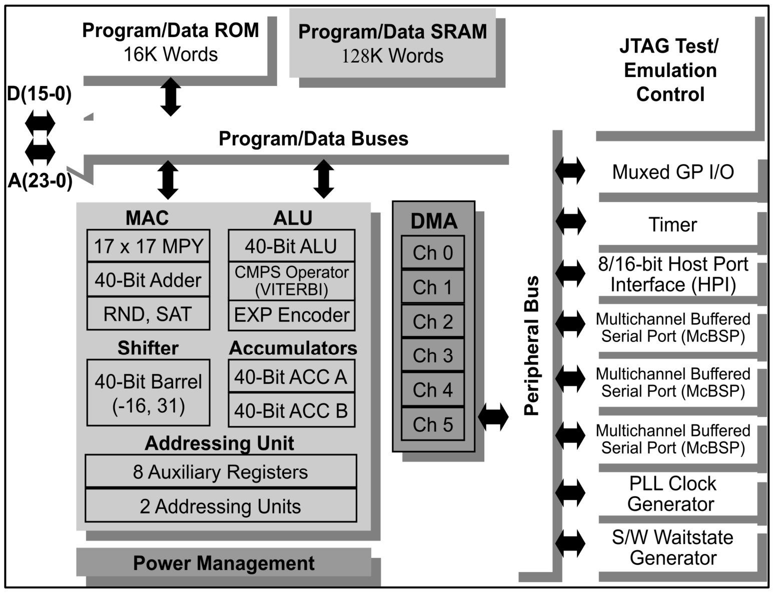


Figure 2. TMS320VC5416 DSP block scheme

- advanced multibus architecture (3 separate data memory on 16 bits buses and a program memory bus);
- arithmetical and logical unit on 40 bits, including 2 accumulator registers on 40 bits and a shifting one;
- parallel 17x17 bits multiplier coupled with a dedicated adder on 40 bits, for multiplication/accumulation operations in a single “not-pipelined” cycle;
- comparing, selecting, storage (CSSU) unit for selection, addition/comparison of Viterbi operator;
- exponential encoder used to compute an exponential value of a value from the 40 bits accumulator (one cycle);
- two address generators with 8 auxiliary registers and two auxiliary arithmetical register units;
- extended addressing mode for 8M x 16bits – maximum addressing space for an external program;
- maximum addressing memory space 192k x 16 bits (64k program words, 64k data words, 64k I/O words);
- ROM memory on chip with a few data program memory configurations.
- RAM memory on chip with dual access;
- instruction repeat and operation block repeat for program code;
- shifting memory blocks instructions – for a better programming and maneuvering of data;
- instructions on operands on 32 bit words;
- 2 or 3 operands reading instructions;
- arithmetical instructions with storage and parallel loading and conditional storage;
- fast return from interruptions;

1.2. System's memory map

DSK includes 64 SRAM memory words and 256k flash ROM words for loading. External memory decoding is made through a CPLD circuit. The associated program of this device selects the RAM memory, ROM flash or the peripherals on board.

Data Memory

Data memory space is extended on 64k words (2 pages of 32k words each). For DSK C5416 system the external data memory is mapped between the addresses 0x8000h and 0xFFFFh. DARAM (Dual Access RAM) memory is mapped in the data space (between the addresses 8000h and FFFFh) if DROM bit (the third bit from the PMST register) is set on "1". Otherwise external memory is mapped in this space.

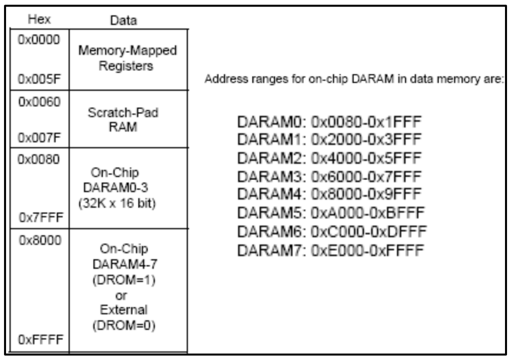


Figure 3. Data memory of DSK TMS320VC5416

Program memory

Program memory configuration can be done in two ways using the bit 5, OVLY of the PMST register. If OVLY=0, on-chip RAM memory is addressable in the data memory and if OVLY=1, RAM memory is mapped in the data memory and program memory also, as shown in the following figure:

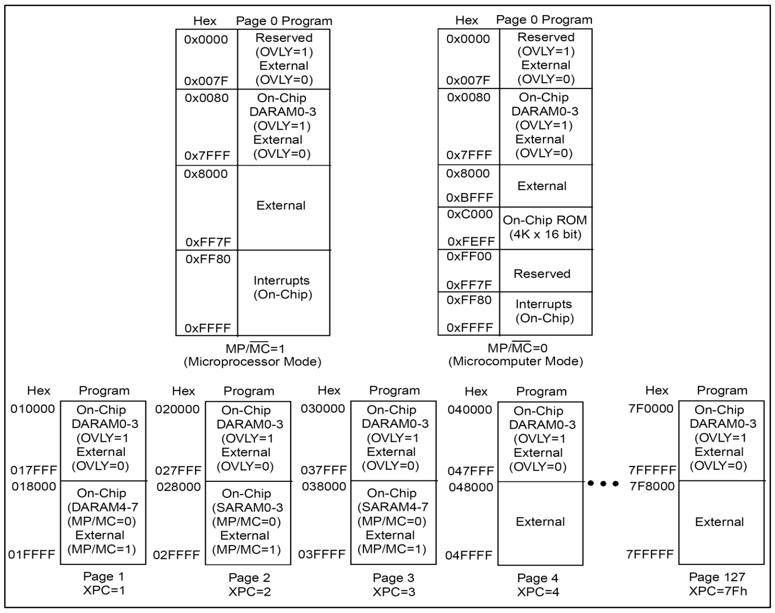

Hex
Page 0 Program
0x0000
0x007F
Reserved (OVLY=1) External (OVLY=0)
0x0080
0x7FFF
On-Chip DARAM0-3 (OVLY=1) External (OVLY=0)
0x8000
0xFF7F
External
0xFF80
0xFFFF
Interrupts (On-Chip)
MP/$\overline{MC}$=1 (Microprocessor Mode)
Hex
Page 0 Program
0x0000
0x007F
Reserved (OVLY=1) External (OVLY=0)
0x0080
0x7FFF
On-Chip DARAM0-3 (OVLY=1) External (OVLY=0)
0x8000
0xBFFF
External
0xC000
0xFEFF
On-Chip ROM (4K x 16 bit)
0xFF00
0xFF7F
Reserved
0xFF80
0xFFFF
Interrupts (On-Chip)
MP/$\overline{MC}$=0 (Microcomputer Mode)
Hex
Program
010000
017FFF
On-Chip DARAM0-3 (OVLY=1 External (OVLY=0)
018000
01FFFF
On-Chip (DARAM4-7 (MP/MC=0) External (MP/MC=1)
Page 1 XPC=1
Hex
Program
020000
027FFF
On-Chip DARAM0-3 (OVLY=1 External (OVLY=0)
028000
02FFFF
On-Chip (SARAM0-3 (MP/MC=0) External (MP/MC=1)
Page 2 XPC=2
Hex
Program
030000
037FFF
On-Chip DARAM0-3 (OVLY=1 External (OVLY=0)
038000
03FFFF
On-Chip (SARAM4-7 (MP/MC=0) External (MP/MC=1)
Page 3 XPC=3
Hex
Program
040000
047FFF
On-Chip DARAM0-3 (OVLY=1 External (OVLY=0)
048000
04FFFF
External
Page 4 XPC=4
Hex
Program
7F0000
7FFFFF
On-Chip DARAM0-3 (OVLY=1 External (OVLY=0)
7F8000
7FFFFF
External
Page 127 XPC=7Fh

Figure 4. Program memory for TMS320VC541

VC5416 uses the CPLD circuit (with 8 registers) in order to select (the interface) the ROM flash memory, SRAM memory, peripherals on the board or for the codec control.

CPLD's 8 registers are mapped in the I/O address space, starting from, 0x0000h address until 0x0007h.

Advantages of on-chip memory operations

-high performance because there are no wait stages;
-lower costs than of the external memory's;
-reduced power consumption than of the external memory's;
-the possibility of larger addressing space access;

1.3 PCM3002 Codec

DSK uses a stereo PCM3002 Codec, in order to provide analogical input and output. Codec interfacing is made through two channels, a control one and a data one.

DSK's CPLD circuit is used for the interfacing with the control channel through the two registers on 8 bits on I/O space. CODEC_L and CODEC_H through which commands on 16 bits are sent to the codec.

After sending the control word, CPLD automatically transmits the data, serial, to the control CODEC's interface through master control signals. Master serial control signals of the CODEC are: CODEC_MC (master clock), CODEC_MD (master data) and CODEC_ML (master load).

In addition CPLD generates all the clock signals for PCM3002 through an oscillator and through CODEC_CL K control register. The default system clock value is 12.288MHz. CPLD uses the internal system clock of the CODEC in order to generate a clock signal of 3.0122 MHz a frame synchronization signal of 48 kHz.
CODEC_CLK register allows the programmer to modify the sampling frequency through frequency rate division control of the PCM3002 codec. Data is transmitted to the codec through VC5416, McBSP2 interface.

A bit from the CPLD's MISC register will permit McBSP2 register to be routed to the expansion HPI connector; the main path is assured through PCM3002 Codec.

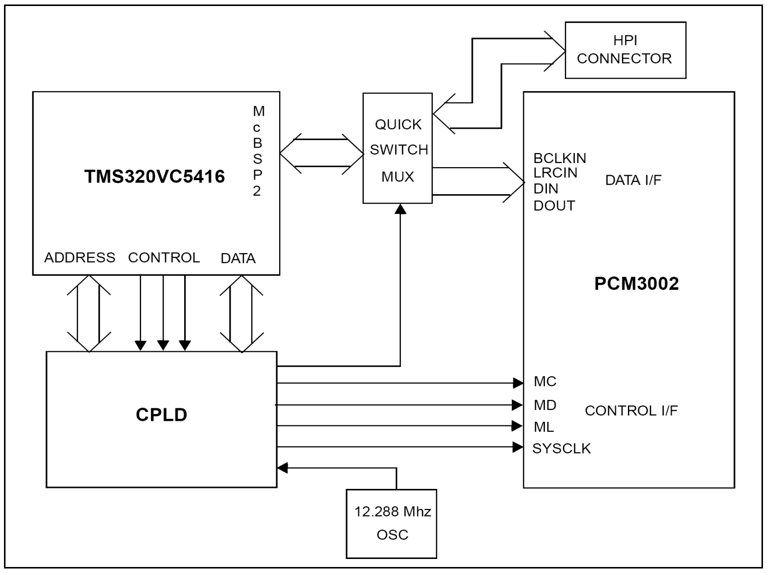


Figure 5. Interconnection between PCM3002, CPLD and DSP

PCM3002 the control interface has 4 internal registers which are software controlled through a serial interface.

Data interface programming

PCM3002 data interface is connected to the McBSP2 pins of VC5416

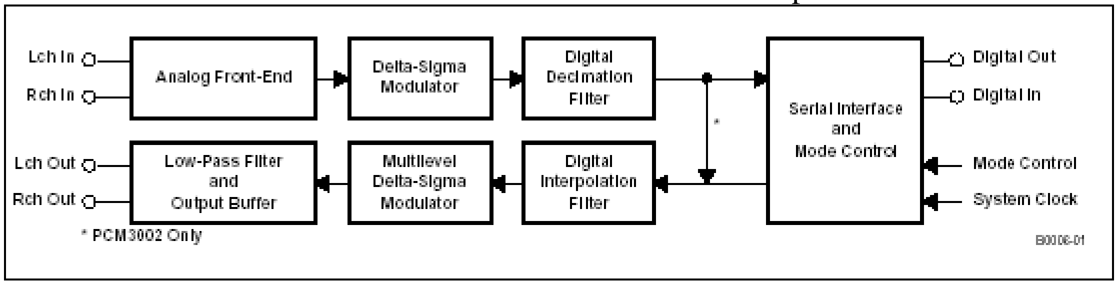

Lch In
Rch In
Analog Front-End
Delta-Sigma Modulator
Digital Decimation Filter
Serial Interface and Mode Control
Digital Out
Digital In
Lch Out
Rch Out
Low-Pass Filter and Output Buffer
Multilevel Delta-Sigma Modulator
Digital Interpolation Filter
Mode Control
System Clock
* PCM3002 Only
B0006-01

Figure 6. PCM 3002 Codec

PCM 3002 is a single chip audio codec (analog-digital and digital - analog converter).CAD and CDA involves delta-sigma modulation with a sampling rate of 64. CAD includes a digital decimation filter and CDA involves a digital interpolation filter for oversampling with 8. CDA also includes digital attenuation, zero infinite detection and the 'mute' soft function, all these making a complete subsystem. PCM3002 offers a power mode which functions independently with both converters.

PCM3002 is fabricated using an advanced CMOS process and in available SSOP packet on 24 pins. They are suitable for a wide range of applications, for which costs are important and for which performance is a priority.

For PCM3002, the programmable functions are software controlled.

Functions:

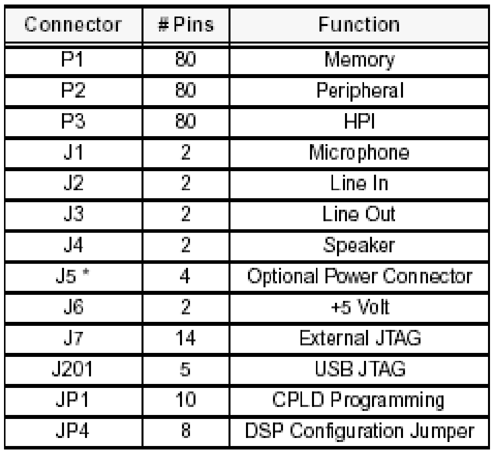
- CAD and CDA on 20 bits, monolithic
- Input/Output 16/20 bits
- Software control: PCM3002
- Special functions (PCM3002): digital attenuation (256 steps)
  soft mute
  digital loop back
  for alternative digital audio data formats
- Sampling rate: 4 kHz to 48 kHz
- Fed at 3V
- Portable/mobile audio applications.

CAD stereo	**CDA** stereo
- input for a single end fed - antialias filter - oversampling with 64 - high performances: SNR: 90 dB	- output for a single end fed - analogic low pass filter - oversampling with 64 - high performances: SNR: 94 dB

The special of the PCM3002 codec are controlled using 4 registers each having 16 bits. See Appendix 1.

1.4 System connectors

DSK board has 16 connectors through which the users' access to the board is realized. The connector, their size and each one's function are presented in the table bellow and their description in Appendix 2.



Connector	# Pins	Function
P1	80	Memory
P2	80	Peripheral
P3	80	HPI
J1	2	Microphone
J2	2	Line In
J3	2	Line Out
J4	2	Speaker
J5 *	4	Optional Power Connector
J6	2	+5 Volt
J7	14	External JTAG
J201	5	USB JTAG
JP1	10	CPLD Programming
JP4	8	DSP Configuration Jumper

Table 1. The connector of TMS320VC5416 DSK board

2. Applications

The following presented applications will be made using Code Compressor Studio C5416 Simulator. After describing the steps needed to create a new simulation project, for each application basic theoretical concepts will be described. Also we will describe the used formulas, block schemes of the applications, program sources and signal graphical representation before and after the processing.

2.1 The need to create a new simulation project

The following method was written for installing the template applications, which is stored in template.zip. For other projects, change the name 'template' in the one of the zip file used. (e.g. delay)

The execution of this procedure need a few minutes. Win.Zip program is needed.

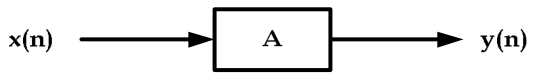
1.	Start Code Composer Studio for DSK TMS320C5416.
2.	Select *Project -> New*. For Project Name, type the word *template*. Click on button Finish. A new project was created. Keep in mind the directory in which the project was saved, then minimize Code Composer Studio.
3.	Using Windows Explorer, go in the directory in which the new project was saved, e.g. *C:\ti\myprojects\template*
4.	Copy *template.zip* file in the directory *C:\ti\myprojects\template*
5.	Extract files from *template.zip* using WinZip at current directory with option Extract Here.
6.	Go back at Code Composer Studio. *Select Project -> Add Files to Project.* Select path C:\ti\myprojects\template. Use Control+left click mouse for showing all files *.c*. Click on button Open for add them to the project.
7.	Select *Project -> Add Files to Project.* Using down arrow in Files at Type box, select Linker Command File (**.cmd*). Uses left click mouse for showing the file *template.cmd.* Click on button Open for add the file *template.cmd* at the project.

8.	Select *Project -> Add Files to Project*. Using down arrow in Files at Type box select Asm Source Files (**.a*, *.s*). Use Control + left click mouse for showing files *.asm* (if exists). If files *.asm* doesn't exist in project, click on button Open for adding files *.asm* at the project. If not click Cancel.
9.	Is not needed to add files *.h* at the project. This is done automatically.
10.	Select *Project -> Add Files to Project*. Go at C5400/cgtools/lib and load file rts.
11.	Select *Project -> Rebuild All*. This time project should be build successfully and file *template.out* and file could be loaded using *File ->Load Program*.
12.	Follow the steps described in the previous paper.

2.2 Gain control

The gain control is implemented in the project named VOLUME. An application which uses the CCS offered options [adds watch or GEL files] is created in order to adjust a variable gain applied to the input signal.

The implementation formula is: y(n) = A * x(n) ;



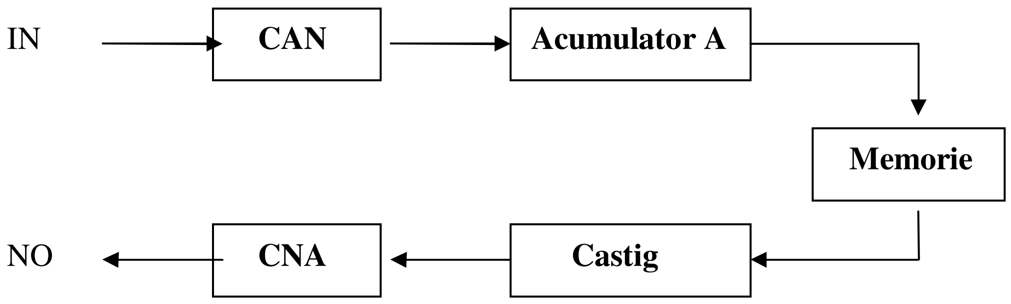


Figure7. Application block scheme

Observation when gain equals one, the output signal is the same with the input signal.

Source program is presented bellow:

```
/*    V O L U M E . C    */

#include <stdio.h>
#include "volume.h"

/* Global declaration */

int inp_buffer[BUFSIZE];                        /* data buffer processing */
int out_buffer[BUFSIZE];
int gain = MINGAIN;                             /* volume control variable */
unsigned int processingLoad = BASELOAD;         /* loading value of processing routine*/
/* Functions */

extern void load(unsigned int loadValue);
static int processing(int *input, int *output);
static void dataIO(void);
```

```
/*Main program*/
void main()
{       int *input = &inp_buffer[0];
        int *output = &out_buffer[0];
        puts("volume example started\n");                      /* message display */

    /* infinite loop */
    while(TRUE)
    {   dataIO();              /* Read the input data using probe-point connected to a host file.*/
        #ifdef FILEIO
        puts("begin processing")    /* display message at the beginning of the processing */
        #endif
        processing(input, output);                              /* the gain is applied through
processing function*/
    }
}

/*Processing function:  applies a transform (multiplied by a gain factor) to a received input signal;
PARAMETERS: input/output buffer addresses ; RETURN VALOAREA : TRUE.*/

static int processing(int *input, int *output)
{   int size = BUFSIZE;                                          /* size of buffer memory*/
    while(size--)                                                /* for all buffer samples... */
        {  *output++ = *input++ * gain;   /* output equals input multiplied by a coefficient,
                                                                and buffers pointers are incremented */
        }
    load(processingLoad);
    return(TRUE);
}

/*IO data function: reads the inputs signal  and writes the processed output signal.*/

static void dataIO()
{  return;  }
/***********************************************************************
******/
```

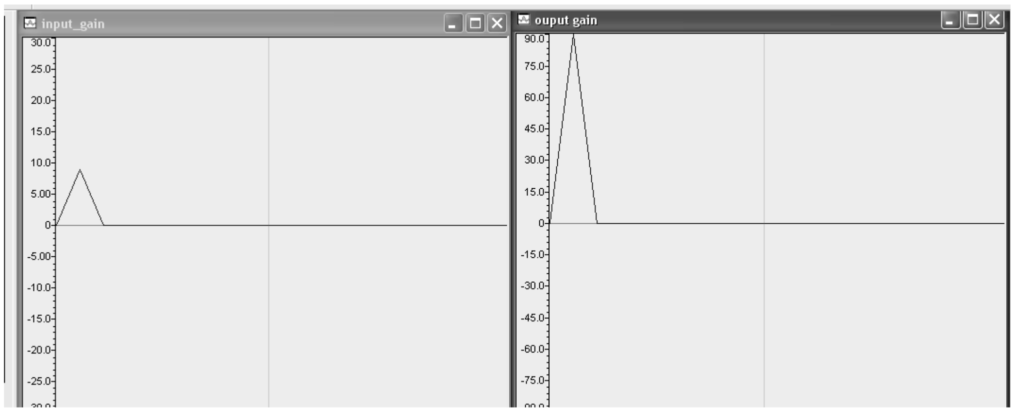
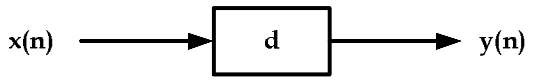


Fig. 8 Input signal **Fig. 9. Output signal**

2.3 Delay

Another phenomenon often met in signal processing is signal delay. This may appear due to unfavorable conditions, channel imperfections or different techniques applied to the signal.

The computation formula, block schema and application source are presented below



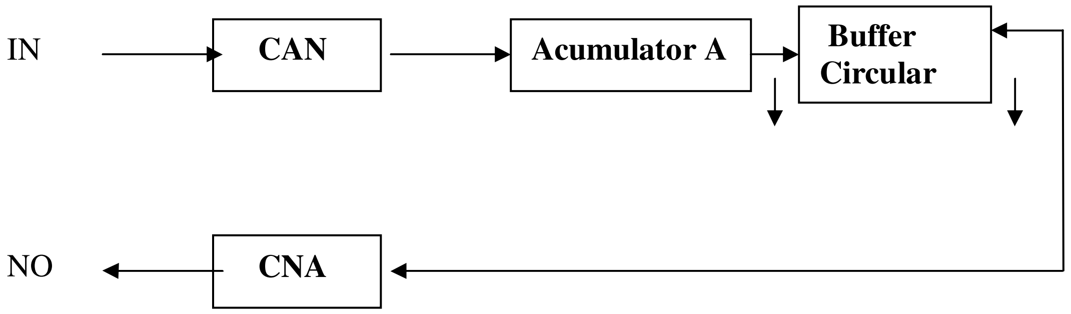


Figure 10. Application block scheme.

```
        /*    D E L A Y . C                                   */

        #include <stdio.h>
        #include "delay.h"

        /* Gobal declarations */
        int inp_buffer[BUFSIZE];                         /*volume data buffer processing*/
        int out_buffer[BUFSIZE];
        int delay = MINGAIN;                                    /* volume control variable */

        /* Functions */
        extern void load(unsigned int loadValue);
        static int processing(int *input, int *output);
        static void dataIO(void);

        /* main program */
        void main()
        {
           int *input = &inp_buffer[0];
           int *output = &out_buffer[0];
           puts("Delay example started\n");                    /* message display  */

           /* infinite loop */
           while(TRUE)
           {
              dataIO();      /*Reads the inputs data using a probe-point connected to host file.
                            Writes the output data in a graph connected through a probe-point.*/

              #ifdef FILEIO
              puts("begin processing")                         /* message display at the
beginning of the processing */
              #endif

              /* delay is applied using processing function */
              processing(input, output);
           }
        }

/* Processing function:applies a transform (attenuation and delay) at a received input signal.
PARAMETERS: input and output address buffers. RETURN VALUE: TRUE.*/
        static int processing(int *input, int *output)
        {
   int size = BUFSIZE;                                /* size of buffer memory */

   while(size--)                                      /* for all buffer samples*/
{  if (size<(BUFSIZE-delay))                          /* if the elements required are in the
memory*/

      *output = *(input-delay);                       /* delayed input is reproduced at the
output */
                          else
        *output = 0;                                  /* otherwise 0 is transmitted */

        output++;                                     /* increments buffers pointers */
        input++;
          }
          return(TRUE);
        }
```

/*IO data function :reads the input signal and writes the output processed signal. PARAMETERS: doesn't have; RETURN VALUE: doesn't have.*/

```
        static void dataIO()
        {
          return;
        }

        /*****************************************************************
************/
```

Used program allows us to fix the delay variable. For a value equivalent with 2ms obtaining the following graphics:

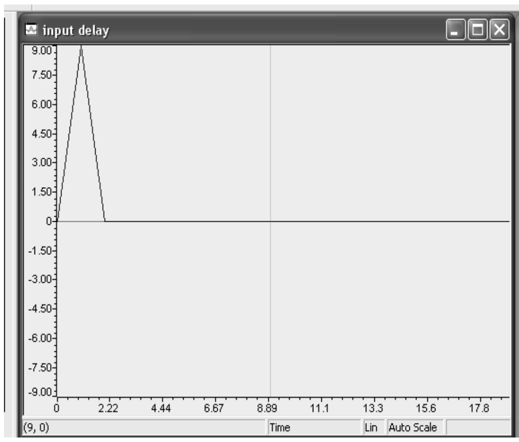


Fig. 11. Input signal

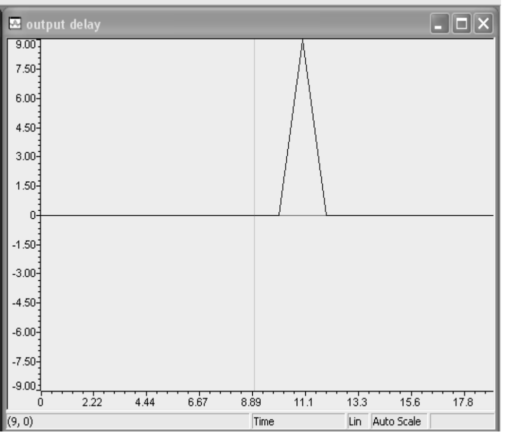


Fig. 12. Output signal

2.4 Reverberation

Reverberation is the sound persistence in a certain space after the original sound is removed. When the sound is produced in a certain space a large number echoes are produced and then slowly decreasing, while the sound is absorbed by the air and the wall, in this way the reverberation is created. This is especially observed when the source of the sound is stopped and the reflections carry on, decreasing in amplitude until they cannot be heard anymore.

The used formula in reverberation software implementation is:

$$y(n) = x(n-d) + 1/2 * x(n-2d) + 1/4 * x(n-3d) + 1/8 * x(n-4d) ;$$

Exercise: Using the previous programs make a project that implements reverberation using the following block skin:

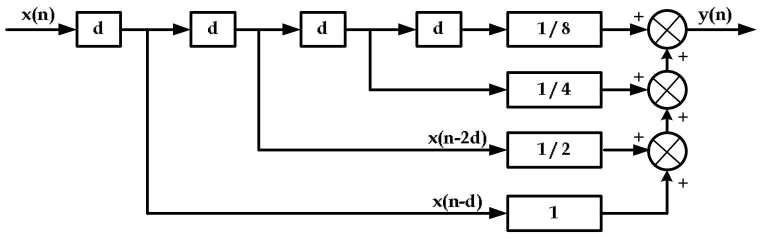
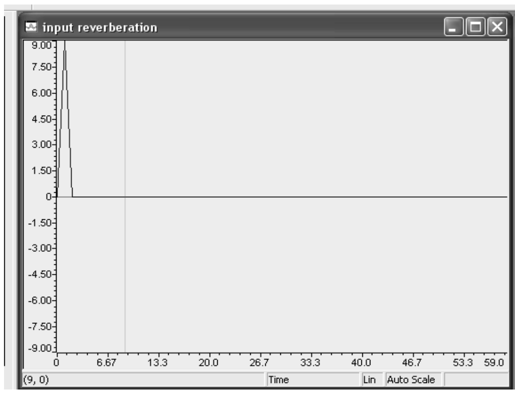


Figure 13: Application block scheme

The signal diagram



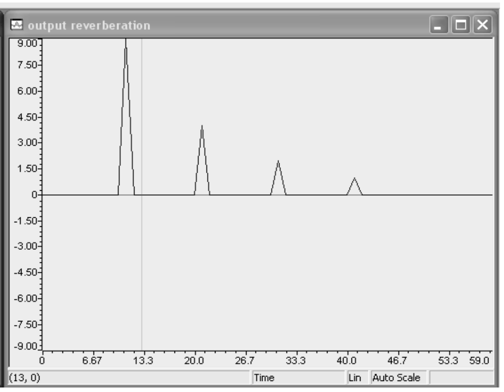


Figure 14: Input signal **Figure 15: Output signal**