

ANADOLU UNIVERSITY
DEPT. OF ELECTRICAL & ELECTRONICS ENGINEERING
EEM489 MICROPROCESSORS II LABORATORY
LAB 7: DAC - Wave Generation with Adjustable Frequency

TA: ŞÜKRÜ GÖRGÜLÜ

- PART 1: Generating A Specific Wave Using D/A Converter module.

In this part, you will generate four types of waves, sawtooth, triangle, square, and sine, according to the value of 8-position dip-switch array.

CASE 0: When the value of dip-switch array is null, it will not generate any signal.

CASE 1: When it is 1, it will generate sawtooth wave.

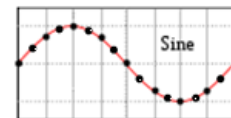
CASE 2: When it is 2, it will generate triangle wave.

CASE 4: When it is 4, it will generate square wave.

CASE 8: When it is 8, it will generate sine wave.

It is important to say that you will generate 16-point signals which means higher resolution with compared to the previous lab work. Add a periodical timer interrupt to your project and set its frequency as 16 Hz (actually, you should set the value to 32 Hz to match exact frequency). You will use this timer interrupt function for setting the value of DAC output.

The frequency of the signals are supposed to be 1 Hz. Therefore, you should have to repeat a full period of signal per second. This means your DAC module will generate 16 outputs per second.



You can easily generate an 16-point sine wave in **matlab** using the following code:

```
>> sine16 = sin(0 : pi / 8 : 2 * pi - pi / 8); % 16-point sine with amplitudes btw [-1,1]
```

You should convert/map these values into byte interval of [0,255]. Therefore, you have to add 1, multiply with 127(0x7f), and round them to integers to use them in your c-project.

```
>> sine16 = 127 * (sin(0 : pi / 8 : 2 * pi - pi / 8) + 1);
```

If you repeatedly send these samples to the output of DAC using timer interrupt, it generates a sine wave. In each interrupt, only one element of the array is sent. After sixteen interrupts, all samples are sent, and array starts from beginning. a whole sine wave is generated with 16 points of samples. With a timer frequency of 16 Hz, a sine wave of 1 Hz is generated.

```
extern byte sine16[]; // defined in main.c

void TI1_OnInterrupt(void)
{
    static int i=0;
    DA1_SetValue8(&sine16[i++%16]);
}
```

The same manner works for square, triangular, and sawtooth waves. To generate 16-point sample arrays, you can sample a full period of the wave with sixteen points. Generate and save them as byte arrays named: square16[], triang16[], and saw16[].

In the second part you will change the frequency of the periodic timer interrupt using data received from the computer terminal.

- PART 2: Setting The Frequency Of Timer From The Computer Terminal

In this part you will add RS232 serial communication bean to your project of part-1. This time, you will receive characters from the pc-terminal and use those data to set the frequency of the timer.

Using interrupts with events (**Bean AsynchroSerial**)

To avoid the necessity to continually check for incoming data the [Interrupt service/event](#) property can be enabled (so called interrupt mode). In this mode the AsynchroSerial bean uses the capability of the cpu to interrupt the flow of the program to execute a different part of code. The hardware of the asynchroserial module may generate the interrupt upon various occasions - when a new character is received, when sending of a character is completed, when a receive error is detected, etc.

The [events](#) of the AsynchroSerial bean are set of call-back functions, which are automatically invoked from the interrupt service routine. Because the body of the event function is defined by user it is possible to react immediately without wasting time in waiting loops.

The following example demonstrates usage of the [OnRxChar](#) event. The event is invoked when a character is received. The example implements simple repeater - when OnRxChar event is invoked the received character is read and sent back.

```
EVENTS.C

void AS1_OnRxChar(void)
{
    AS1_TComData ch;
    /* TComData type is defined in the
    AS1.h header file */

    /* Read received character and send
    it if no error is detected */
    if (AS1_RecvChar(&ch) == ERR_OK)
        AS1_SendChar(ch);
}
```

You can use the example code given in the help documents above, and modify it to get a four-digit numeric value from the terminal.

Your program should display a question which is asking for a numeric value to be entered.

For example: char message[] = “\n\rEnter the frequency (Hz) of the signal: \0”;

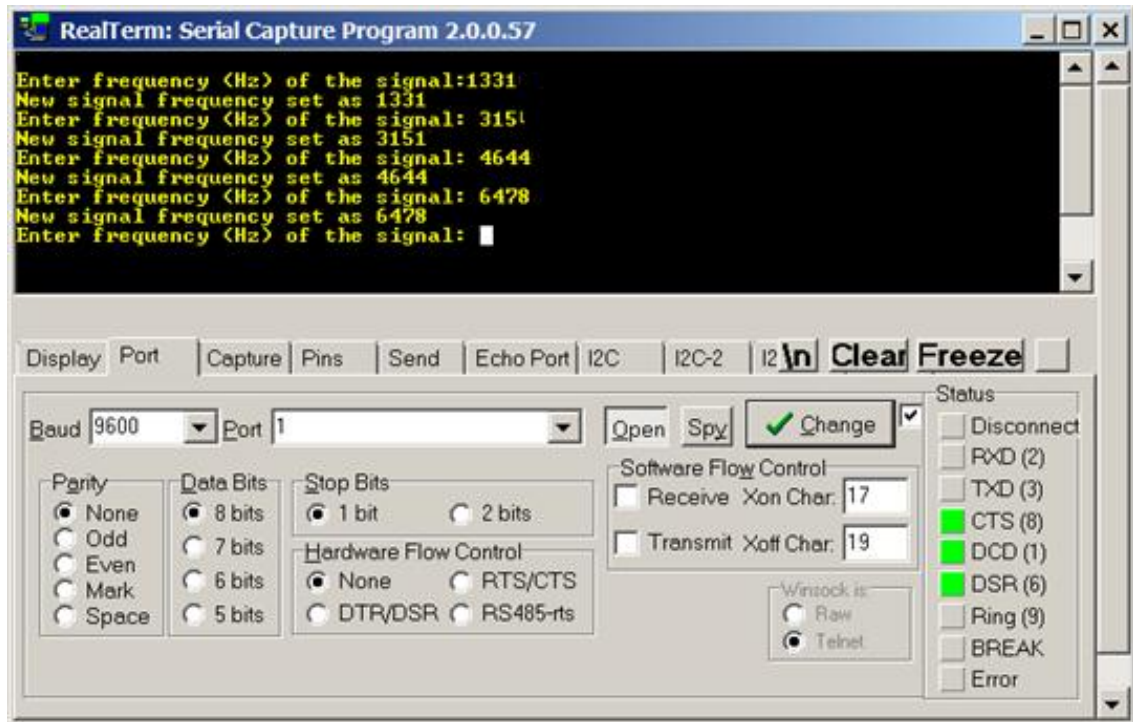
This message will be displayed on the terminal window and the system will be waiting for the four-digit numeric value.

There will be some criteria for your program :

- The receiver code should check incoming character. If it is not numeric, it'll discard it. For example, if user pushes “12sd23as”, then alphanumeric characters are discarded and “1223” should be accepted and displayed.

- After four digits received, the code should turn off receiving data (Hint: AS1_TurnRxOff) and send a new message “New signal frequency set as XXXX”. And set the frequency of the timer according to the received value.
- Following the new frequency setting, the program will ‘turn on’ receiving data and wait for a new value.

An example of terminal outputs for the program can be seen below.



You should also verify the correct frequency using an oscilloscope.