MSc. Systems Engineering and Engineering Management

MICROPROCESSOR BASED SYSTEMS

Prof. Dr. Ing. Werner Krybus

MODULE EEM4016

Assignment Number 2

Submitted by:

Md. Saiful Islam Sajol

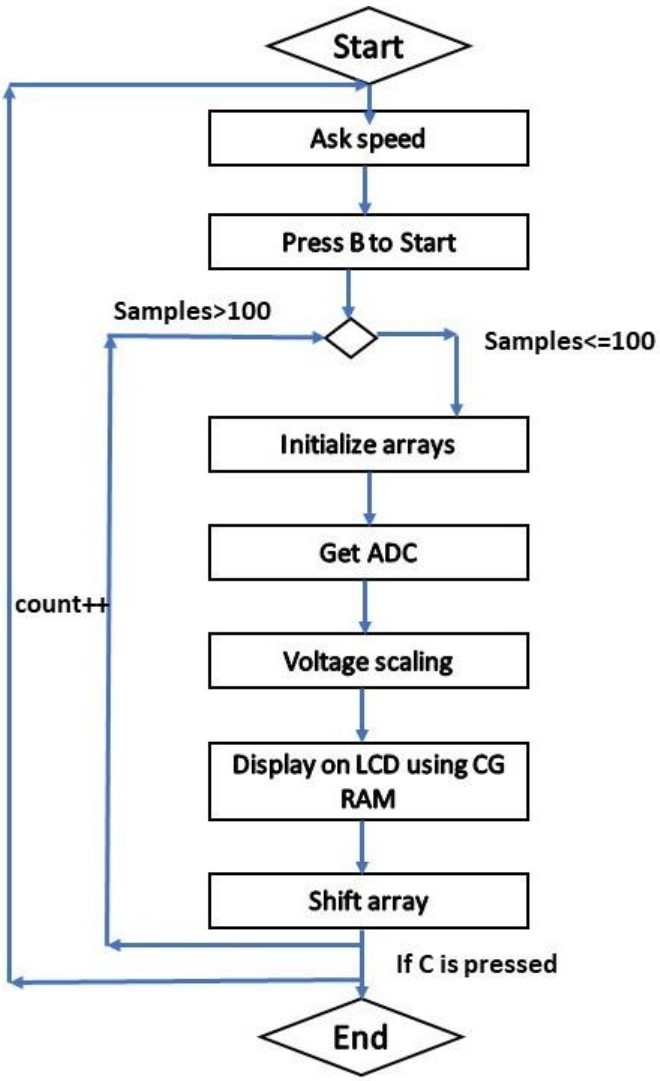
Matrikelnumber 30042836

02 August 2019

Introduction

A Data logger is needed to be built. The voltage of the analogue signal will be recorded and will be shown on the LCD display. When no samples have been recorded it will show dashes, after the initialization of record it will start showing corresponding bar graphs in the upper LCD line. The current sample will be displayed at the most right position and the previous samples will be shifted to the left. All samples need to be shown on an internal buffer of 100 samples. The data logger will start the recording when button B will be presses, and it will stop when button C is pressed. For the input voltage signal potentiometer 1 is used.

Basic Flow Chart:



Pseudo Code

1. Get the input voltage from potentiometer 1 by using getadc() function
2. Define a function to write the required string on LCD, print_string() in this assignment.
3. Define a function that builds user defined character (BAR in this case) at CGRAM then call the function, build_cgram() in this assignment.
4. Intialize 3 array variables of 100 element each. First array is for storing integer value, second array to show the value after decimal point and third one is for storing corresponding bar graph.
5. Define a function that converts the voltage came from the potentiometer into corresponding bar graph. voltage_to_cg() in this assignment
6. main function()
    - 6.1. Enable ADC channel
    - 6.2. set prescaler frequency
    - 6.3. set single conversion mode
    - 6.4. call the saved CGRAM
    - 6.5.
        - 6.5.1. Initialize LCD
        - 6.5.2. Write codes for Speed Setting and ask the user to select speed by using keypad
        - 6.5.3. Display the selected speed and ask to type Press B to continue
        - 6.5.4. Write the code to set sampling rate. The more the value of keypad pressed the slower will be the code.
        - 6.5.5. Write the code to display speed at lower left side on the LCD
        - 6.5.6. Before getting input voltage from potentiometer there will be only strips on the LCD. Write the code to show only Strips before getting any input voltage on the first 13 blocks from left side
        - 6.5.7. Initialize all the elements of array by putting zero to each element
        - 6.5.8. Write the code to continuously get the input voltage and stop until C is Pressed
            - 6.5.8.1.1. Add the feature when the recording exceeds the 100 of array, discard the oldest data
            - 6.5.8.1.2. Write code to continuously updating the new voltages after 100 samples
        - 6.5.9. Call the function . voltage_to_cg() to convert each samples of voltage to corresponding BARs
        - 6.5.10. Write the code to display final bar graph
        - 6.5.11. Write the code to display voltage in digits to upper right side and index number to the lower right side
7. End of the code

Analysis / Detailed Explanation of Code

```
#include <kamavr.h>
unsigned int getadc(int chan);
void print_string(char* str);
void build_cgram();
int voltage_to_cg(int val, int cnt);
void display_volt_cnt(int input);
int integer_array[100];
int decimal_array[100];
int cg_array[100];
int speed;
unsigned char key;
```

- First, getadc() function is used to get the input of potentiometer 1. Channel will be selected 0 , to select the first potentiometer.
- To write the string on LCD print_string() function is defined.
- voltage_to_cg() function is used to convert each samples of voltage to corresponding BAR.
- display_volt_cnt() is defined to show the voltage newest value and index number at upper and lower row of LCD respectively.
- Besides 3 arrays were declared for storing the values of voltage and there corresponding converted character stored in CGRAM.

.

```
int main(void)
{
//  Enanbling ADC
ADCSR = ADCSR | (1<< (ADEN));

//  set prescaler frequency
ADCSR |= 0b00000111;     // ADPS2, ADPS1, ADPS0 = 1

//  set single conversion
ADCSR&= ~(1<<(ADFR)); //ADFR = 0
init_lcd();

//  calling the saved cgram
build_cgram();
```

- To convert the analog signal coming from the potentiometer ADC channel is enabled.
- For prescaling division factore of 128 is used.
- Single channel conversion method is used.
- Then by calling build_cgram function the desired BAR graph pattern is written line by line to the CG RAM.

```
void build_cgram()
{
//point to the first cgram addresss
write_lcd(0,0x40);
//write from top line to bottom line
for(int i=0;i<7;i++)
write_lcd(1,0); //7 lines blank
write_lcd(1,0b00011111); //1 line full
for(int i=0;i<6;i++)
write_lcd(1,0); //6 lines blank
for(int i=0;i<2;i++)
write_lcd(1,0b00011111); //2 lines full
for(int i=0;i<5;i++)
write_lcd(1,0); //5 lines blank
for(int i=0;i<3;i++)
write_lcd(1,0b00011111); //3 lines full
for(int i=0;i<4;i++)
write_lcd(1,0); //4 lines blank
for(int i=0;i<4;i++)
write_lcd(1,0b00011111); //4 lines full
for(int i=0;i<3;i++)
write_lcd(1,0); //3 lines blank
for(int i=0;i<5;i++)
write_lcd(1,0b00011111); //5 lines full
for(int i=0;i<2;i++)
write_lcd(1,0); //2 lines blank
for(int i=0;i<6;i++)
write_lcd(1,0b00011111); //6 lines full
for(int i=0;i<1;i++)
write_lcd(1,0); //1 lines blank
for(int i=0;i<7;i++)
write_lcd(1,0b00011111); //7 lines full
for(int i=0;i<8;i++)
write_lcd(1,0b00011111); //8 lines full
}
```

- The concept of building BAR graph is shown above. Every LCD block is consists of 5*8 =40 ponits. Number of row is 8 and number of column is 5. So, total 8 levels of BAR can be designed. Using ´for loop´ every level is designed individually and are kept at CGRAM

```
//-------- CODES FOR :  SPEED SETTING   AND PRESS B TO CONTINUE ---------


//waiting until speed is selected by using keypad
do
{
write_lcd(0,0x80);
print_string("SET SPEED");
key=readkey();
speed=key;
}while(key>9);

//display the speed
write_lcd(0,0xC0);
write_lcd(1,0x30+speed);
delay_ms(250);
//display message "press B"
init_lcd();
write_lcd(0,0x80);
print_string("PRESS B");
```

- The code will continue to show ‘SET SPEED ’ . This can take any speed below 9
- The selected display will be shown to left side of the lower row. N.B. The symbol 0x30 is 0. It was used to show the speed digit.
- After getting the speed it will ask to press B , and proceed further command.

```
//----------- CODES FOR :  to show  SETTING SAMPLING RATE  --------


//set the delay, the bigger the selected value of keypad, the slower the sampling rate
int delay=(10+speed)*20;
    do{
    key=readkey();
    }while(key!=11);

init_lcd();


//------------- CODES :  To show S:Speed     on the LCD   ---------


//point to lower most left loaction of LCD
write_lcd(0,0xC0);
//write letter "S"
write_lcd(1,0x53);
//write symbol ":"
write_lcd(1,0x3A);
//write the speed value
write_lcd(1,0x30 + speed);
```

- Since it was asked maintain highest speed when lowest value is pressed and for lowest speed hightst value must be pressed. So, the value of 'delay' is determined using a simple equation delay= (10+speed)*20
  So , for speed selection 1 ; delay=(10+1)*20= 220ms
  And , , for speed selection 7 ; delay=(10+7)*20= 340ms
  That means the higher keypad selection the lower would be speed.
- The next ection is to show the delected speed on the lower left part on the LCD. 0x53 and 0x31 are the hexa value of the symbol 'S' and ':' respectively.

```
//------- CODES : TO SHOW  ""STRIPS"" TO THE FIRST 13 BLOCKS FROM LEFT SIDE ON LCD ----


//point to upper most left location of LCD for bar graph
write_lcd(0,0x80);
//the bar graph on LCD initially shows strips
for (int i=0; i<13;i++)
write_lcd(1,0x2D);




//-----------ARRAY INITIALIZATION-----------------

for (int i=0; i<100; i++)
{
integer_array[i] =0x30; //making array of zeros
decimal_array[i] =0x30; //making array of zeros
cg_array[i] =0x20; //making array of space characters
}

//initialize counter
int count=0;
```

- Before getting any voltage the blocks of LCD will show strips. This portion is to initilize strips on LCD to the first 13 block from left. The other 3 blocks will be configure to show the value of voltage.
- All the members of array variables –'arrays integer_array' and 'decimal_array' are filled with zero. And the cg_array is filled with space.

```
//--- CONTINUOUSLY GET THE INPUT UNTILL KEYPAD C IS NOT PRESSED ------

do
{
key=readkey();

count++;
    int voltage = getadc(0)*0.049; // (5/1023) *10
    int integer = voltage/10;
    int decimal = voltage%10;


//when the recording exceed the 100 of array, discard the oldest data,
//shift the data to the previous index,
  if (count>100)
  {
    for (int n=0;n<100-1;n++)
    {
    integer_array[n]=integer_array[n+1];
    decimal_array[n]=decimal_array[n+1];
    cg_array[n]=cg_array[n+1];
    }
    //make counte equal 100 to save the new 100 in the last index
    count=100;
  }

//the last index for new 100
    integer_array[count-1]=0x30 + integer;
    decimal_array[count-1]=0x30 + decimal;


//using the current voltage to get the CGRAM and save it in cg_array
    voltage_to_cg(voltage, count);
```

- The ADC voltage value that we are getting feom potentiometer 1 is scaled by a factor (5/1023), since 5 is the highest voltage that comes from potentiometer and to ease the round up it was multiplied by a factor of 10.
- Every sample value of the voltage will be called by voltage_to_cg() function to convert the voltage level to corresponding BAR graph.
- If the sample is more than 100 than it updates the newest value by discarding the oldest data. All the data of the arrays were given to theit previous memebers, so that the 100th member of the arrays remain free, then to set the newest member to that free space , count is equal to set 100 forcefully.
- By this the newest member is always set to the 100th member of the array.

```
//------------ VOLTAGE TO RESPECTIVE BAR CONVERSION  ---------------


int voltage_to_cg(int val, int cnt)
{ //There are 8 lines for 5V, split the voltage by 6 to create 8 sections
//In this case the voltage is multiplied by 10, therefore the range is from 0 -
//the index of cg_array start from 0
if (val<=6)
{
cg_array[cnt-1]=0x20; //blank, cgram 0
}
else if(val<=12)
{
cg_array[cnt-1]=0x00; //1 line, ASCII 0 = cgram address 1
}
else if(val<=18)
{
cg_array[cnt-1]=0x01; //2 lines, ASCII 1 = cgram address 2
}
else if(val<=24)
{
cg_array[cnt-1]=0x02; //3 lines, ASCII 2 = cgram address 3
}
else if(val<=30)
{
cg_array[cnt-1]=0x03; //4 lines, ASCII 3 = cgram address 4
}
else if(val<=36)
{
cg_array[cnt-1]=0x04; //5 lines, ASCII 4 = cgram address 5
}
else if(val<=42)
{
cg_array[cnt-1]=0x05; //6 lines, ASCII 5 = cgram address 6
}
else if(val<=48)
{
cg_array[cnt-1]=0x06; //7 lines, ASCII 6 = cgram address 7
}
else
{
cg_array[cnt-1]=0x07; //8 lines(full line), ASCII 7 = cgram address 8
}
return cg_array[cnt-1];
}
```

- This is the function that converts the voltage came from the getadc() channel is converted into corresponding BAR graph. For easier , round up the voltage is multiplied by a factor 10. So for the input voltage of 0 to 5 the range to code this corresponding bar graph would be 0 to 50- Since , maximum 8 rows or levels are possible to define, and 6*8=48, so range of each layer is 6.
- Now, for example the voltage is 3.5 then it would be multiplied by 10. So the value of val is equal to 35. When it enters into the loop it would check each condition. For a valuof 35 the conditions of line 5, line 6, line 7, line 8 would be fulfilled. So, all the address of these CGRAM would be stored to the corresponding member array variable. By this it would show the corresponding bar grpah for each of the sample.

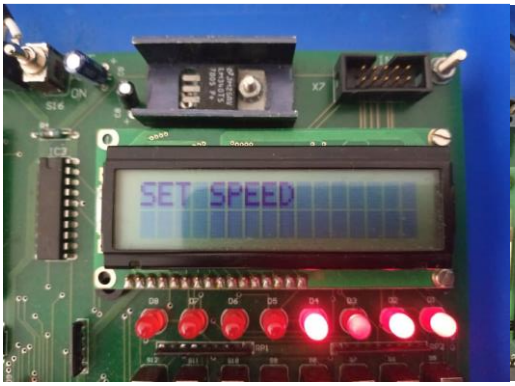
```
//------------CODE TO DISPLAY THE FINAL BAR GRAPH ------------------------


//the first 13 samples(16 slots on lcd, rightmost 3 are used for displaying voltage)
 if(count<14)
 {
    for(int n=0;n<count;n++)
    {
    write_lcd(0,0x8C - (count-n-1)); //showing for the first 13 samples of array

    write_lcd(1,cg_array[n]);
    }
 }


//display the bar graph after the 13th to 100th sample value
//shifting the bar as inserting new 100 to array
 else
 {
    for(int n=0; n<13; n++)
    {
    write_lcd(0,0x8C-n); // showing for the shifted samples after 13th sample values

    write_lcd(1, cg_array[count-n-1]);
    }
 }
```

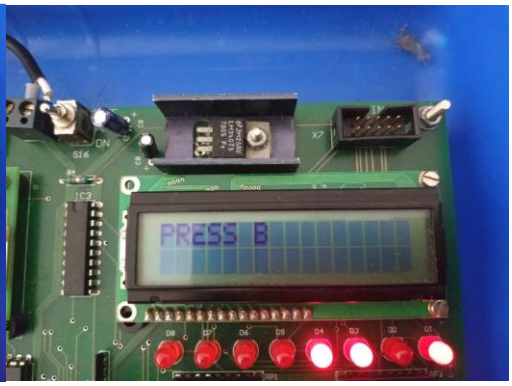
- When the sample is less than 13 it will start displaying the value starting from the position 8C and in each loop the older value would be shifted to the previous position like at first loop it would shifted to 8B, then after the following loop it would shifted 8A, then the position 89 etc.
- But when the sample number would be higher than 13, for example 14 , then all the members would be shifted leftwards so that only the newest 13 samples are shown on LCD.

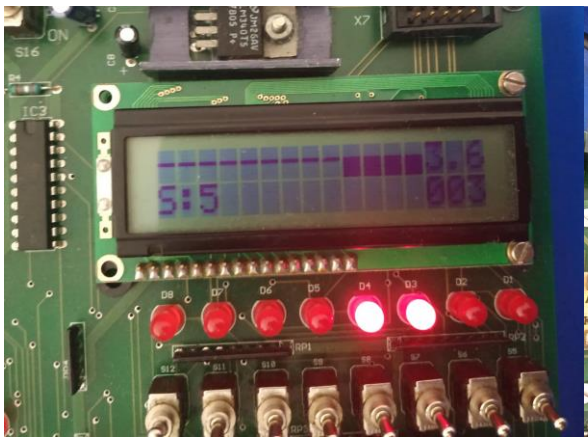
```
//----------  CODE TO DISPLAY VOLTAGE IN DIGITS AND INDEX NUMBER -----------


//display first number, komma sign, decimal number
 display_volt_cnt(count);
 delay_ms(delay);
}
 while(key!=12);      //  stop this activity when button C is pressed


}
}


//---- SHOW THE VOLTAGE AND INDEX AT THE UPPER AND LOWER RIGHT OF LCD  ---


void display_volt_cnt(int input)
{
//display the voltage
write_lcd(0,0x8D);
write_lcd(1,integer_array[input-1]);
write_lcd(1,0x2E);
write_lcd(1,decimal_array[input-1]);
//display the index,
write_lcd(0,0xCD);
write_lcd(1,0x30+((input-1)/100));
write_lcd(1,0x30+((input-1)%100 /10));
write_lcd(1,0x30+((input-1)%10));

}
```

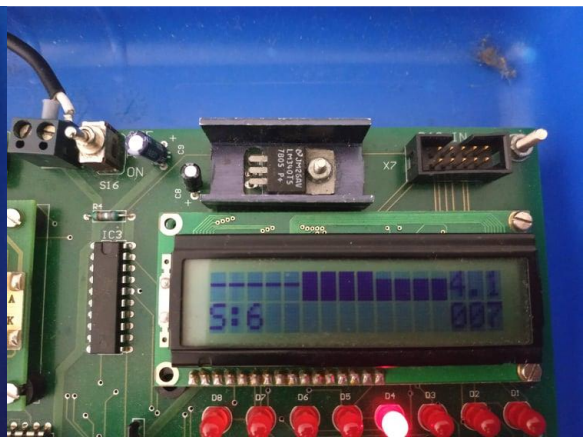
- Calling the display_volt_cnt(input) function will show the latest voltage value to the upper half of the LCD, starting from the position 8D to the next two blocks. 8D positio n is for intger part and 8F is for decimal part. 8E is fixed to '.', whose hexavalue is 0x2E.
- Similarly, the lower rightmost 3 blocks are coded to show index number upto 99.

Test Results









Observation

The program is showing results as expected. But, the code doesn't stop immediately when the C button is pressed. It has been seen that button C has to pressed 3/4 times to stop the sampling. Or, if the C button is pressed for 2-3 second then it works. Also, Further development is needed for adding shifting display window feature.

Conclusion

It was a great experience for me to let the work done. I am heartily sorry for my late submission. I have been going under tremendous pressure for last few days. But finally I overcame all the issues and let the work done.