Mini Report

KTH Royal Institute of Technology

Hristo Georgiev & Azad Mustafa

May 2017

IE1206 Embedded Electronics

Table of content

Mini report

Title of the project

PIC Caesar Cipher machine

Description of the proposed idea

The goal of the project is to program a Caesar Cipher machine that the user inputs a message through the UART tool, and then the program encrypt the message. Using a button on the board, the user can choose if the LCD screen will show the encrypted message or the decrypted (original) message. The user should hold the button when he sends the message through the UART tool.

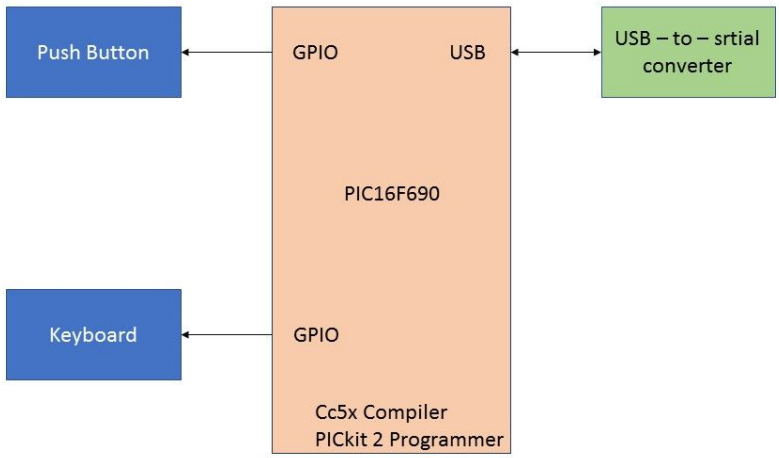


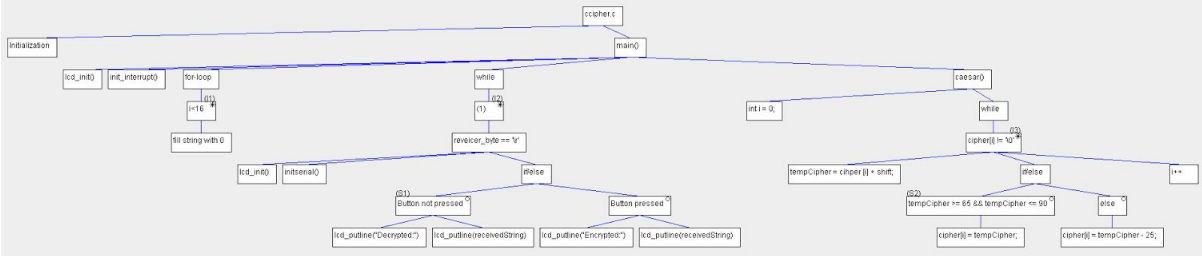
Figure 1: Block Diagram

Function prototype

The prototype can handle messages of the length of 16 characters due to that the display is 2x16 (rows x columns). Due to that the idea is to have one row of the display to show if the message is encrypted/decrypted, we are only able to use one row for the message. Which is limited to 16 character, due to the display width. Version 2.0 of the project could be that the code is improved in a way that more character can be used so the message wouldn't be limited to the LCD size. Another option is to use a extra screen or two.

Program check

To check that the program works we have inputted different messages and verified that the encryption is handled in way that program is written. The result of each execution of the messages is that we have manually checked if the letters are shifting the right amount of times, in our case it will shift 1 times (easier to check). The amount of shifts can be changed in the code.



https://gyazo.com/60b4df7f70863586af136b7689e7896d

Figure 2: Structure diagram

Published code and ASCII graphics

The code for our project can be found in the following GitHub repository.

Link: https://github.com/hristog121/ccipher

The used compiler is “CC5X C Compiler”.

Fritzing

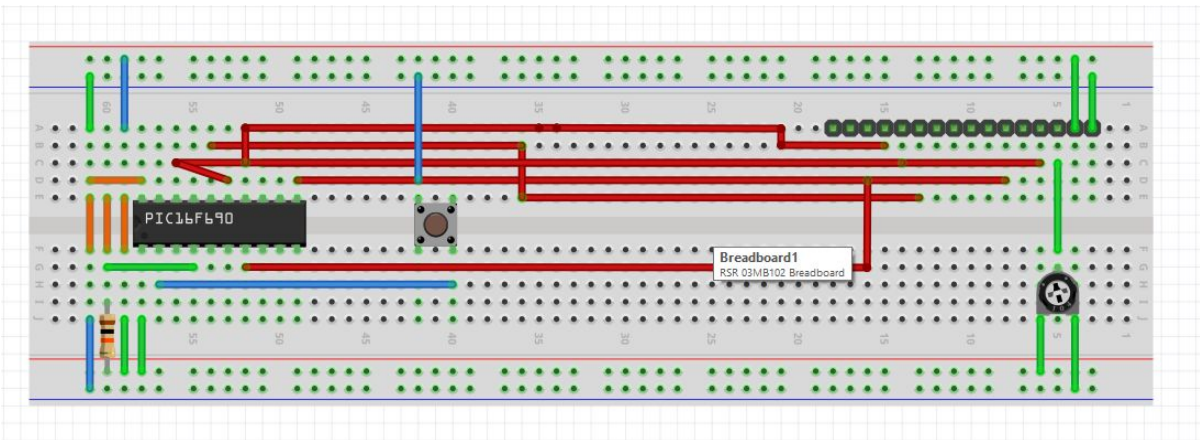


Figure 3: Fritzing Breadboard

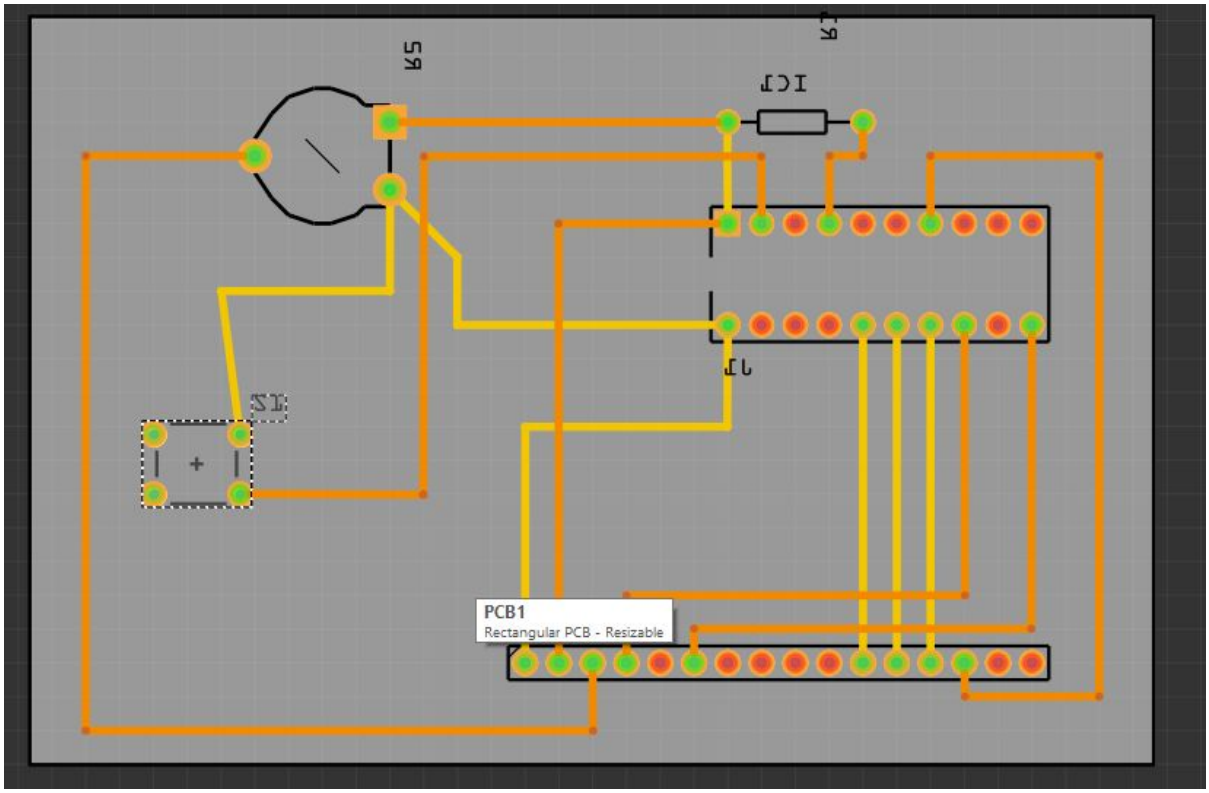


Figure 4: Fritzing PCB (Printed circuit board)

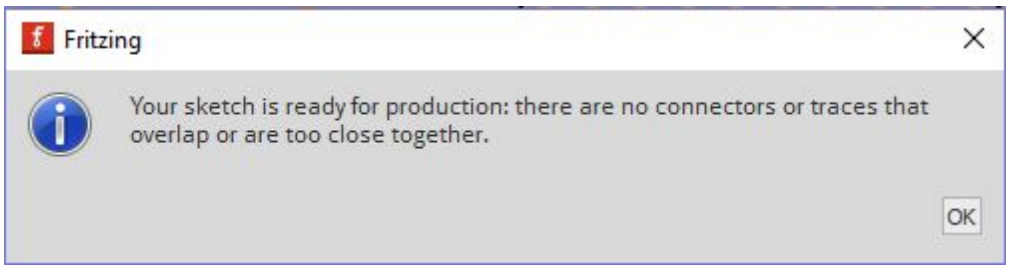


Figure 5: Fritzing Design Rule Check

Appendix

```
/* mini_hello_lcd.c  */
/* LCD two line 16 characthers display         */
/* Line 1 starts at adress 0x00 line 2 at 0xC0 */

#include "16F690.h"
#include "int16Cxx.h"
#pragma config |= 0x00D4
#define MAX_STRING 11

/* I/O-pin definitions                                */
/* change if you need a pin for a different purpose  */
#pragma bit RS  @ PORTB.4
#pragma bit EN  @ PORTB.6

#pragma bit D7  @ PORTC.3
#pragma bit D6  @ PORTC.2
#pragma bit D5  @ PORTC.1
#pragma bit D4  @ PORTC.0

void delay( char ); // ms delay function
void lcd_init( void );
void lcd_putchar( char );
void lcd_putline( char start, const char * text );
void caesar( char* cipher, int shift );
void string_in( char * string );
char getchar( void );
void putchar( char );
void initserial( void );
char randomChar( void );
void delay10( char n);
void init_interrupt( void );

bit receiver_flag;   /* Signal-flag used by interrupt routine   */
char receiver_byte;  /* Transfer Byte used by interrupt routine */


#pragma origin 4
interrupt int_server( void ) /* the place for the interrupt routine */
{
  int_save_registers
  /* New interrupts are automaticaly disabled            */
  /* "Interrupt on change" at pin RA1 from PK2 UART-tool */

  if( PORTA.1 == 0 )  /* Interpret this as the startbit  */
    {  /* Receive one full character   */
      char bitCount, ti;
      /* delay 1,5 bit 156 usec at 4 MHz         */
      /* 5+28*5-1+1+2+9=156 without optimization */
      ti = 28; do ; while( --ti > 0); nop(); nop2();
      for( bitCount = 8; bitCount > 0 ; bitCount--)
       {
         Carry = PORTA.1;
         receiver_byte = rr( receiver_byte);  /* rotate carry */
         /* delay one bit 104 usec at 4 MHz       */
         /* 5+18*5-1+1+9=104 without optimization */
         ti = 18; do ; while( --ti > 0); nop();
        }
```

```
      receiver_flag = 1; /* A full character is now received */
    }
  RABIF = 0;    /* Reset the RABIF-flag before leaving   */
  int_restore_registers
  /* New interrupts are now enabled */
}

void main( void)
{
    /* I/O-pin direction in/out definitions, change if needed  */
        ANSEL=0;                                        //  PORTC digital I/O
        ANSELH=0;
        TRISC = 0b1111.0000;            /* RC3,2,1,0 out*/
    TRISB.4=0;                                  /* RB4, RB6 out */
    TRISB.6=0;

        TRISA.5=1;                                      // set RA5 as input (BTN)


    lcd_init();
        initserial();
        init_interrupt();
        char receivedString[16];
        int counter = 0;
        int i;

        for (i = 0; i < 16; i++){
                receivedString[i] = '\0'; //fill the array with '0'
        }
        // Using the cipher with the button
        while(1) {
                if( receiver_flag ) {
                        if (receiver_byte == '\r') {
                                lcd_init();
                                initserial( );
                                        //If the button is not pressed
                                        if(PORTA.5 == 1){
                                                lcd_putline(0, "Decrypted: ");
                                                lcd_putline(0xC0, receivedString);
                                        }
                                        //If the button is pressed and hold
                                        if(PORTA.5 == 0){
                                                caesar(&receivedString[0], 1);
                                                lcd_putline(0, "Encrypted msg");
                                                lcd_putline(0xC0, receivedString);
                                        }
                                counter = 0;
                                for (i = 0; i < 16; i++){
                                        receivedString[i] = '\0';
                                }
                        } else {
                                receivedString[counter] = receiver_byte;
                                counter++;
                        }
                        receiver_flag = 0;

                }
    }
}
```

```
/* ********************************** */
/*            FUNCTIONS               */
/* ********************************** */

void lcd_init( void ) // must be run once before using the display
{
  delay(40);  // give LCD time to settle
  RS = 0;     // LCD in command-mode
  lcd_putchar(0b0011.0011); /* LCD starts in 8 bit mode          */
  lcd_putchar(0b0011.0010); /* change to 4 bit mode              */
  lcd_putchar(0b00101000);  /* two line (8+8 chars in the row)   */
  lcd_putchar(0b00001100);  /* display on, cursor off, blink off */
  lcd_putchar(0b00000001);  /* display clear                     */
  lcd_putchar(0b00000110);  /* increment mode, shift off         */
  RS = 1;    // LCD in character-mode
             // initialization is done!
}

void init_interrupt( void ) {
  IOCA.1 = 1; /* PORTA.1 interrupt on change */
  RABIE =1;   /* interrupt on change         */
  GIE = 1;    /* interrupt enable            */
  return;
}


void lcd_putchar( char data )
{
  // must set LCD-mode before calling this function!
  // RS = 1 LCD in character-mode
  // RS = 0 LCD in command-mode
  // upper Nybble
  D7 = data.7;
  D6 = data.6;
  D5 = data.5;
  D4 = data.4;
  EN = 0;
  nop();
  EN = 1;
  delay(5);
  // lower Nybble
  D7 = data.3;
  D6 = data.2;
  D5 = data.1;
  D4 = data.0;
  EN = 0;
  nop();
  EN = 1;
  delay(5);
}

void lcd_putline(char start, const char * text)
{
   RS = 0;  // LCD in command-mode
   lcd_putchar( start );  // move to text position
   RS = 1;  // LCD in character-mode
   char i, k;
   for(i = 0 ; ; i++)
    {
      k = text[i];
      if( k == '\0') return;   // found end of string
      lcd_putchar(k);
```

```
    }
   return;
}

void delay( char millisec)
/*
  Delays a multiple of 1 milliseconds at 4 MHz (16F690 internal clock)
  using the TMR0 timer
*/
{
    OPTION = 2;  /* prescaler divide by 8        */
    do  {
        TMR0 = 0;
        while ( TMR0 < 125)   /* 125 * 8 = 1000  */
            ;
    } while ( -- millisec > 0);
}

void caesar(char* cipher, int shift) {
  int i = 0;
  while (cipher[i] != '\0') {
  char tempCipher = cipher[i] + shift;
    if (tempCipher >= 65 && tempCipher <= 90) {
      cipher[i] = tempCipher;
    } else {
          //  Else tempCipher is always > 90 so we calculate the next one by 65 + tempCipher - 90
      cipher[i] = tempCipher - 25;
    }
    i++;
  }
}

void delay10( char n)
{
    char i;  OPTION = 7;
    do  {
        i = TMR0 + 39; /* 256 microsec * 39 = 10 ms */
        while ( i != TMR0)  ;
    } while ( --n > 0);
}

void initserial( void )  /* initialise PIC16F690 serialcom port */
{
   ANSEL.0 = 0; /* No AD on RA0             */
   ANSEL.1 = 0; /* No AD on RA1             */
   PORTA.0 = 1; /* marking line             */
   TRISA.0 = 0; /* output to PK2 UART-tool  */
   TRISA.1 = 1; /* input from PK2 UART-tool */
   receiver_flag = 0 ;


   return;
}

char getchar( void )  /* recieves one char, blocking */
{
   /* One start bit, one stop bit, 8 data bit, no parity = 10 bit. */
   /* Baudrate: 9600 baud => 104.167 usec. per bit.                */
   char d_in, bitCount, ti;
   while( PORTA.1 == 1 ) /* wait for startbit */ ;
      /* delay 1,5 bit 156 usec at 4 MHz         */
      /* 5+28*5-1+1+2+9=156 without optimization */
```

```
      ti = 28; do ; while( --ti > 0); nop(); nop2();
   for( bitCount = 8; bitCount > 0 ; bitCount--)
       {
        Carry = PORTA.1;
        d_in = rr( d_in);  /* rotate carry */
         /* delay one bit 104 usec at 4 MHz       */
         /* 5+18*5-1+1+9=104 without optimization */
         ti = 18; do ; while( --ti > 0); nop();
        }
   return d_in;
}

void putchar( char cha )  /* sends one char */
{
  char bitCount, ti;
  PORTA.0 = 0; /* set startbit */
  for ( bitCount = 10; bitCount > 0 ; bitCount-- )
   {
     /* delay one bit 104 usec at 4 MHz       */
     /* 5+18*5-1+1+9=104 without optimization */
     ti = 18; do ; while( --ti > 0); nop();
     Carry = 1;     /* stopbit                    */
     cha = rr( cha ); /* Rotate Right through Carry */
     PORTA.0 = Carry;
   }
  return;
}
```

```
/* ********************************** */
/*            HARDWARE                */
/* ********************************** */

/*
         ___________  ___________
        |           \/           |
  +5V---|Vdd     16F690       Vss|---GND
 BTN -<-|RA5        RA0/AN0/(PGD)|
        |RA4            RA1/(PGC)|
        |RA3/!MCLR/(Vpp)  RA2/INT|
        |RC5/CCP              RC0|->-D4
        |RC4                  RC1|->-D5
  D7 -<-|RC3                  RC2|->-D6
        |RC6                  RB4|->- RS
        |RC7               RB5/Rx|
        |RB7/Tx               RB6|->- EN
        |____________________________|
*/


/*
          LCD two lines, Line length 16 characters
          Internal ic: HD44780A00
          _________________
         |                 |
         |         Vss  1|--- GND
         |         Vdd  2|--- +5V
         |    Contrast  3|-<- Pot
         |          RS  4|-<- RB4
         |      RD/!WR  5|--- 0, GND
         |          EN  6|-<- RB6
         |          D0  7|
         |          D1  8|
         |          D2  9|
         |          D3 10|
         |          D4 11|-<- RC0
         |          D5 12|-<- RC1
         |          D6 13|-<- RC2
         |          D7 14|-<- RC3
         |_________________|
*/
```