VILNIUS GEDIMINAS
TECHNICAL UNIVERSITY
FACULTY OF ELECTRONICS
DEPARTMENT OF ELECTRONIC SYSTEMS

Keypad control using PIC16F887 MCU

Computers and computerized systems in automatics
Task 1

Done by: EKSfmu-16 gr. st. Arūnas Butkus
Checked by: prof. Algirdas Baškys

Vilnius, 2017

Work goal: create a program in C language for PIC16F887 MCU, which would allow printing hexdecimal value output and signs * and # on included LCD display using 4x4 keypad on EasyPIC 6 development system.

Algorithms used in tasks.

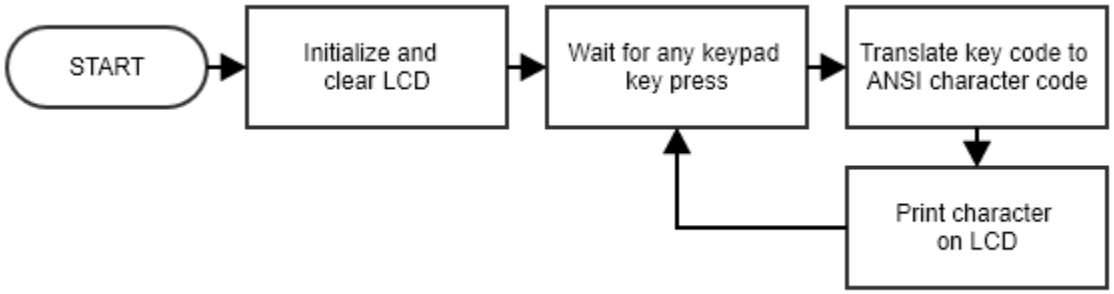


Fig. 1. Program algorithm.

Results analysis.

Main issue is figuring out how keypad presses could be pulled and how to print to LCD. MikroC libraries have the required function to pull keypad button press and it only requires that the port to which the keypad is connected be defined as "`keypadPort`". Having done that the key that was pressed code is pulled by `Keypad_Key_Click()` function. Key code is then translated into character ANSI code utilizing `switch` structure.

Data display on LCD is also handled by libraries provided by MikroC IDE. For operation this library requires 12 registers to be named specific names and then it works. To start of `Lcd_Init()` is called to initialize LCD display. Then few `Lcd_Cmd()` commands are called: parameter `_LCD_CLEAR` clears the LCD screen and parameter `_LCD_CURSOR_OFF` hides the cursor, that shows where would cursor utilizing commands output data.

For data output I used `Lcd_Chr()` function, but the way I implemented it instead I should've used `Lcd_Chr_CP()`. That would've saved a few memory segments and few machine cycles involved in comparison of registers and possible increment/assignment. The difference between these is that former requires user to specify coordinates where the character should be put, whereas the latter one prints at the cursor point.

Conclusions.

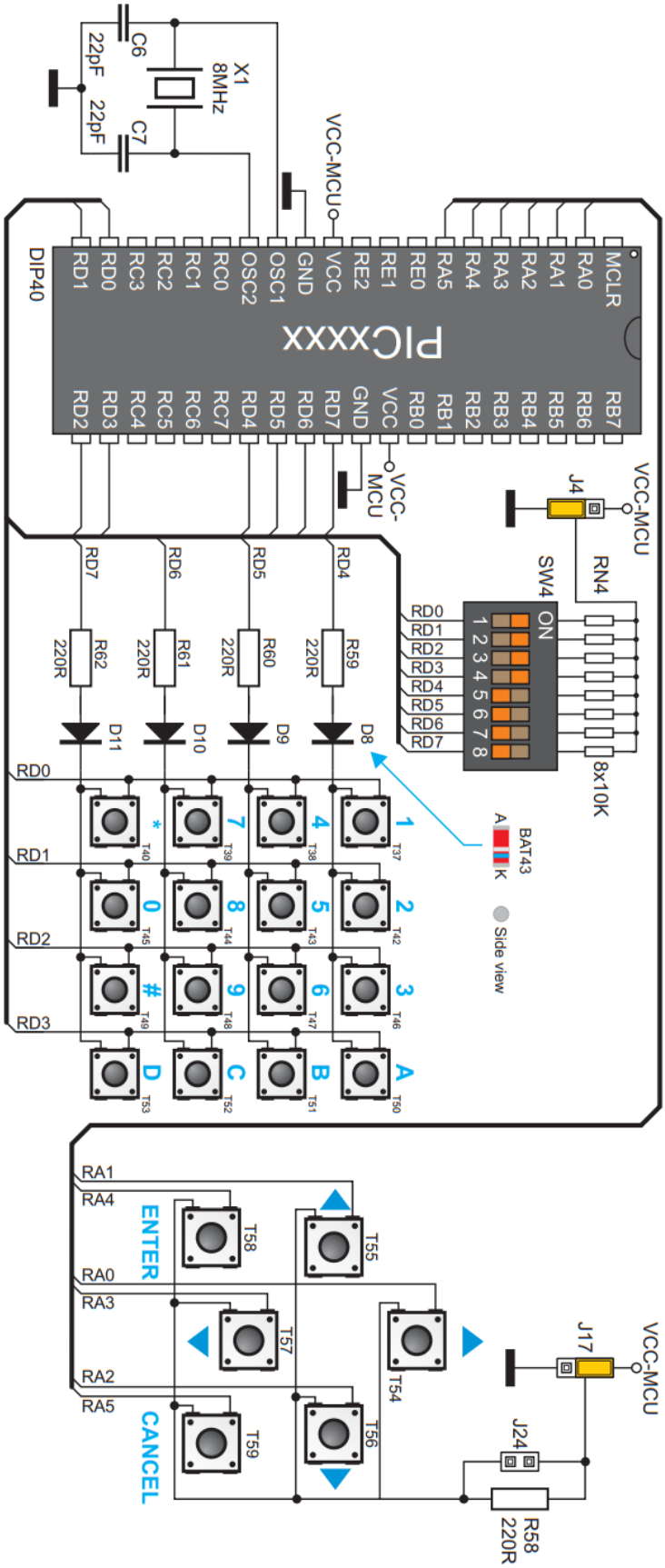
Having done task 2 I now see few locations for improvement in the solution of this task, but the initial solution is functional so I'll leave it as is for now.

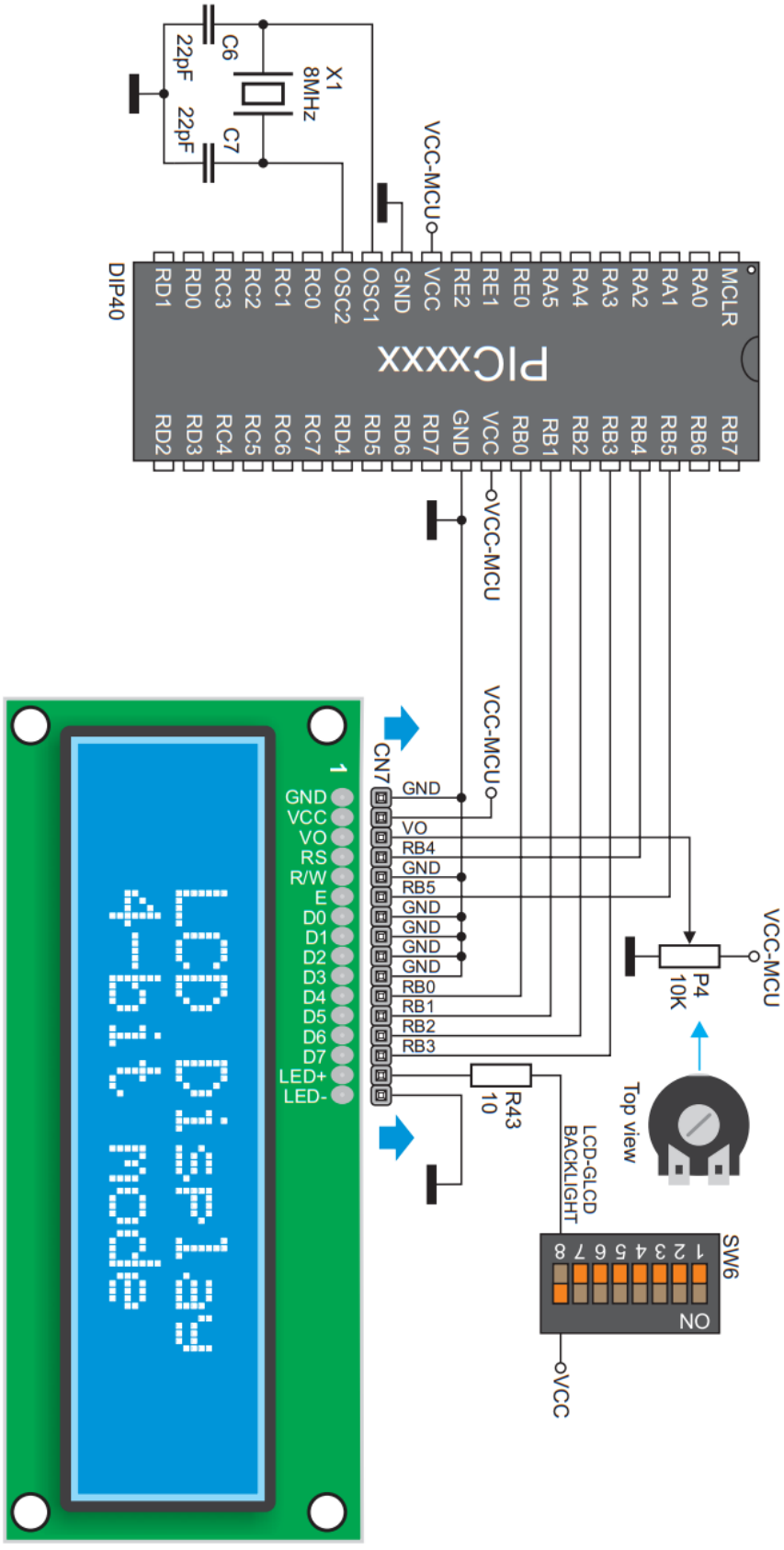
Source code.

```
/*
Sudarykite programa C kalba mikrovaldikliui PIC16F887,
kuri leistu naudojantis EasyPIC 6 stende esancia klaviatura
4x4 ikelti i LCD ekranA skaicius nuo 0 iki 9, raides A,B,C,D
ir ženklus*, #. Programa sudarykite naudodamiesi
Mikro C kompiliatoriumi.
*/

// LCD connection
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D7 at RB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;

// keypad connection
char keypadPort at PORTD;

unsigned short kp, col = 1, row = 1;

void main() {
  Keypad_Init(); // keypad init
  ANSEL  = 0; // make I/O digital
  ANSELH = 0;

  // LCD setup
  Lcd_Init();
  Lcd_Cmd(_LCD_CLEAR);
  Lcd_Cmd(_LCD_CURSOR_OFF);

  while (1) {
    kp = 0; // reset key press variable

    // wait for keypress
    while (!kp) {
      kp = Keypad_Key_Click();
    }

    // interpret the keypress
    switch (kp) {
      case  1: kp = 49; break; // 1
      case  2: kp = 50; break; // 2
      case  3: kp = 51; break; // 3
      case  4: kp = 65; break; // A
      case  5: kp = 52; break; // 4
      case  6: kp = 53; break; // 5
      case  7: kp = 54; break; // 6
      case  8: kp = 66; break; // B
      case  9: kp = 55; break; // 7
      case 10: kp = 56; break; // 8
      case 11: kp = 57; break; // 9
      case 12: kp = 67; break; // C
```

```
61        case 13: kp = 42; break; // *
62        case 14: kp = 48; break; // 0
63        case 15: kp = 35; break; // #
64        case 16: kp = 68; break; // D
65      }
66
67      Lcd_Chr(row, col, kp); // print it on LCD
68
69      // following is playing around
70      col++;
71      if (col > 16) {
72        row++;
73        col = 1;
74      }
75      if (row > 2) {
76        row = 1;
77      }
78    }
79  }
```

Electrical Schemes




C6
22pF
X1
8MHz
C7
22pF
VCC-MCU
DIP40
PICxxxx
MCLR
RA0
RA1
RA2
RA3
RA4
RA5
RE0
RE1
RE2
VCC
GND
OSC1
OSC2
RC0
RC1
RC2
RC3
RD0
RD1
RB7
RB6
RB5
RB4
RB3
RB2
RB1
RB0
VCC
GND
RD7
RD6
RD5
RD4
RC7
RC6
RC5
RC4
RD3
RD2
VCC-MCU
VCC-MCU
CN7
GND
VCC
VO
RS
R/W
E
D0
D1
D2
D3
D4
D5
D6
D7
LED+
LED-
GND
VO
RB4
GND
RB5
GND
GND
GND
GND
RB0
RB1
RB2
RB3
R43
10
LCD-GLCD
BACKLIGHT
VCC-MCU
P4
10K
Top view
SW6
1 2 3 4 5 6 7 8
NO
VCC
LCD Display
4-bit mode