VILNIUS GEDIMINAS
TECHNICAL UNIVERSITY
FACULTY OF ELECTRONICS
DEPARTMENT OF ELECTRONIC SYSTEMS

Calculator using PIC16F887 MCU

Computers and computerized systems in automatics
Task 2

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

Vilnius, 2017

Work goal: create a calculator program for PIC16F887 using C language which would be able to add two numbers and show them and the result on LCD screen.

Algorithms used in tasks.

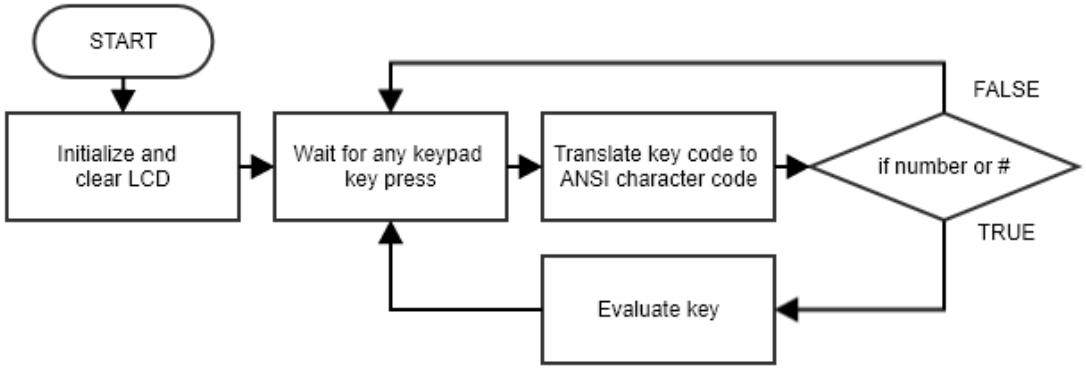


Fig. 1. Full program algorithm loop.

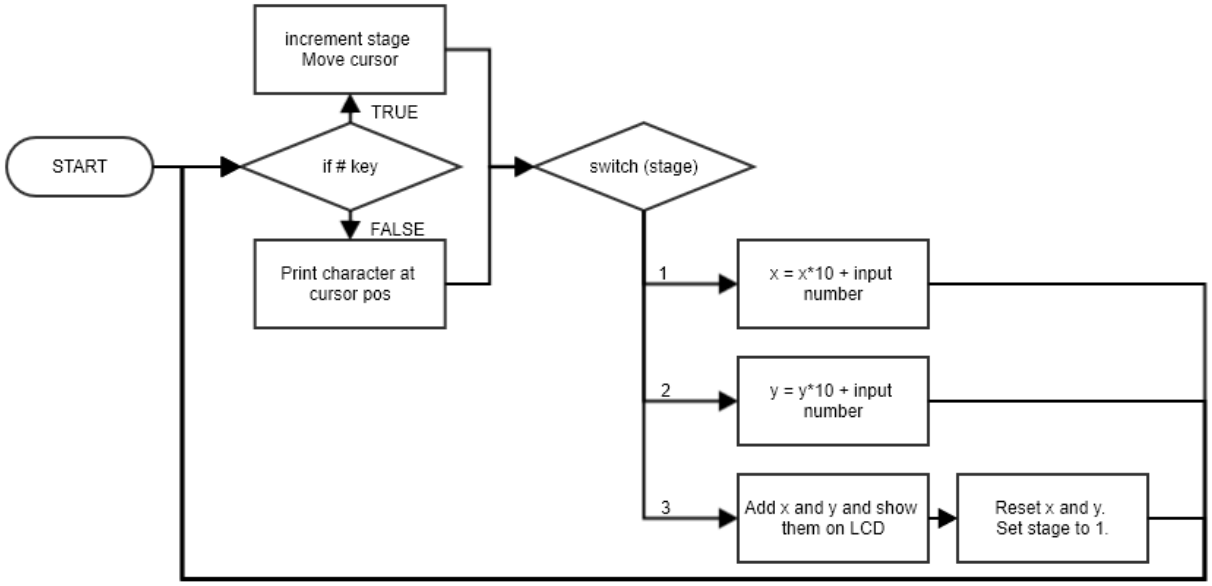


Fig. 2. Key evaluation algorithm.

Results analysis.

Building upon things achieved in Task 1 main issue here is design of effective algorithm for number input and display. I came up with a system using stages. First stage is variable `x` input. Second stage is variable `y` input. Third stage adds them, displays them and resets system to stage 1. I will not go into details here much because I find the algorithm (Fig. 2) quite self-explanatory. And source code is easily readable too.

As far as result images are concerned, I forgot to take any, but row one of LCD shows labels and values for `x` and `y`; row two – shows label and value for sum of said values.

Conclusions.

Not all input cases are accounted for and erroneous results/display could be noticed in cases, but knowing how to use this program and adhering to the limitation set in task, program works.

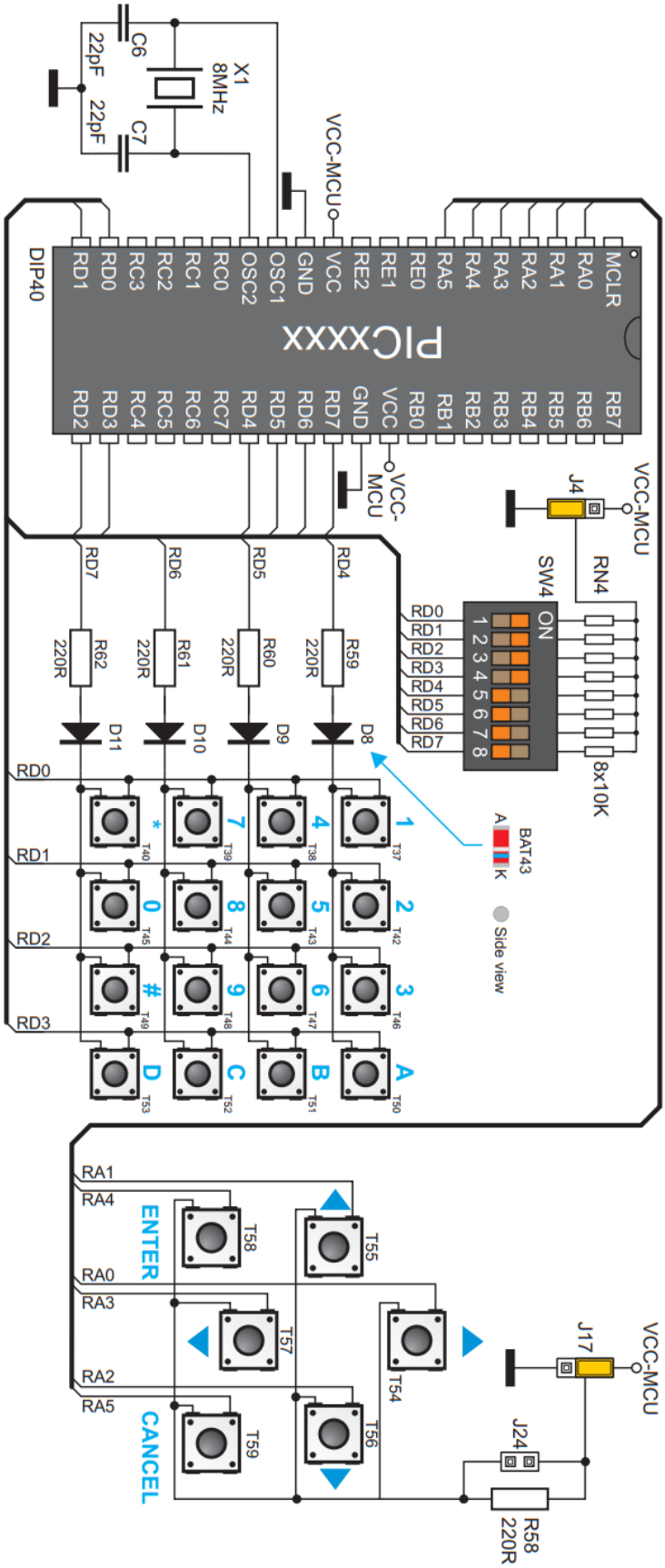
Source code.

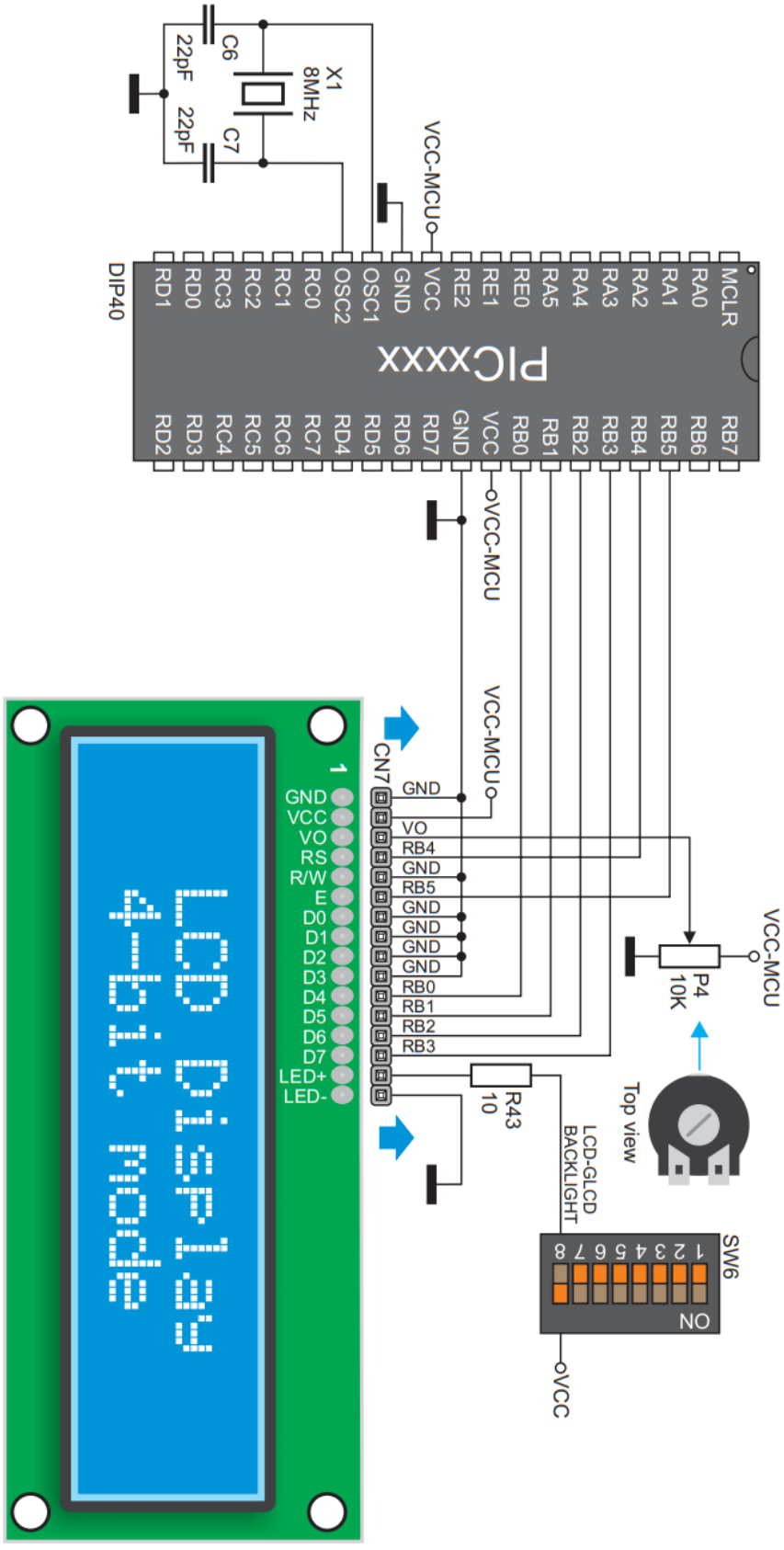
```
001 /*
002 Sudarykite kalkuliatoriaus programa C kalba mikrovaldikliui
003 PIC16F887, kuri sumuotu du skaicius ir sumuojamus skaicius
004 bei atsakyma atvaizduotu LCD ekrane. Maksimalus sumuojamas
005 skaicius 255. Skaiciu ivedimui panaudokite EasyPIC 6
006 stende esancia klaviatura 4x4.
007 */
008
009 // LCD connection
010 sbit LCD_RS at RB4_bit;
011 sbit LCD_EN at RB5_bit;
012 sbit LCD_D4 at RB0_bit;
013 sbit LCD_D5 at RB1_bit;
014 sbit LCD_D6 at RB2_bit;
015 sbit LCD_D7 at RB3_bit;
016
017 sbit LCD_RS_Direction at TRISB4_bit;
018 sbit LCD_EN_Direction at TRISB5_bit;
019 sbit LCD_D4_Direction at TRISB0_bit;
020 sbit LCD_D5_Direction at TRISB1_bit;
021 sbit LCD_D6_Direction at TRISB2_bit;
022 sbit LCD_D7_Direction at TRISB3_bit;
023
024 // keypad connection
025 char keypadPort at PORTD;
026
027 unsigned short
028   kp,
029   wasStage = 1,
030   stage = 1,
031   x = 0,
032   y = 0,
033   sum = 0,
034   val;
035 char txt[3];
036
037 // function prototypes
038 void LCD_move_cursor();
039 void LCD_inputs_reset();
040 void LCD_sum_reset();
041
042 void main() {
043   Keypad_Init(); // keypad init
044   ANSEL  = 0; // make I/O digital
045   ANSELH = 0;
046
047   // LCD setup
048   Lcd_Init();
049   Lcd_Cmd(_LCD_CLEAR);
050   Lcd_Out(1,9,"y = ");
051   Lcd_Out(2,1,"x+y = ");
```

```
052   Lcd_Out(1,1,"x = ");
053
054   while (1) {
055     kp = 0; // reset key press variable
056
057     // wait for keypress
058     while (!kp) {
059       kp = Keypad_Key_Click();
060     }
061
062     // interpret the keypress
063     switch (kp) {
064       case  1: kp = 49; val = 1; break; // 1
065       case  2: kp = 50; val = 2; break; // 2
066       case  3: kp = 51; val = 3; break; // 3
067       case  4: kp = 65; continue; break; // A
068       case  5: kp = 52; val = 4; break; // 4
069       case  6: kp = 53; val = 5; break; // 5
070       case  7: kp = 54; val = 6; break; // 6
071       case  8: kp = 66; continue; break; // B
072       case  9: kp = 55; val = 7; break; // 7
073       case 10: kp = 56; val = 8; break; // 8
074       case 11: kp = 57; val = 9; break; // 9
075       case 12: kp = 67; continue; break; // C
076       case 13: kp = 42; continue; break; // *
077       case 14: kp = 48; val = 0; break; // 0
078       case 15: kp = 35; val = 0; break; // #
079       case 16: kp = 68; continue; break; // D
080     }
081
082     if (kp == 35) { // if # then next stage
083       stage++;
084       LCD_move_cursor();
085     } else {
086       if (kp > 47 && kp < 58) { // if number pressed then write it down
087         Lcd_Chr_CP(kp);
088       }
089     }
090
091     // update values
092     switch (stage) {
093       case 1: // x input stage
094         x = x * 10 + val;
095         break;
096       case 2: // y input stage
097         y = y * 10 + val;
098         break;
099       case 3: // calculation stage
100         LCD_sum_reset();
101         LCD_move_cursor();
102         val = x+y;
103         WordToStr(val, txt);
104         Lcd_Out_CP(txt);
105         x = 0;
106         y = 0;
107         stage = 1;
108         LCD_inputs_reset();
109         LCD_move_cursor();
110         break;
111     }
112   }
```

```
}

void LCD_move_cursor(){
  switch (stage) {
    case 1: Lcd_Chr(1,4,32); break;
    case 2: Lcd_Chr(1,12,32); break;
    case 3: Lcd_Chr(2,6,32); break;
  }
}

void LCD_inputs_reset(){
  Lcd_Out(1,1,"x =     ");
  Lcd_Out(1,9,"y =     ");
}

void LCD_sum_reset(){
  Lcd_Out(2,1,"x+y =    ");
}
```

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