Министерство образования республики Беларусь
Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Институт информационных технологий

Специальность Программное обеспечение информационных технологий

КОНТРОЛЬНАЯ РАБОТА

По курсу Программное обеспечение встроенных систем

Вариант № 3

Студент-заочник 3 курса
Группы № 581072
ФИО Богданова Кристина Евгеньевна
Адрес Польша, г. Реда, ул. Млыньска 5Б-5
Тел. +48 576 335 295

Минск, 2018

Вариант 3. Генератор двоичной увеличивающейся последовательности

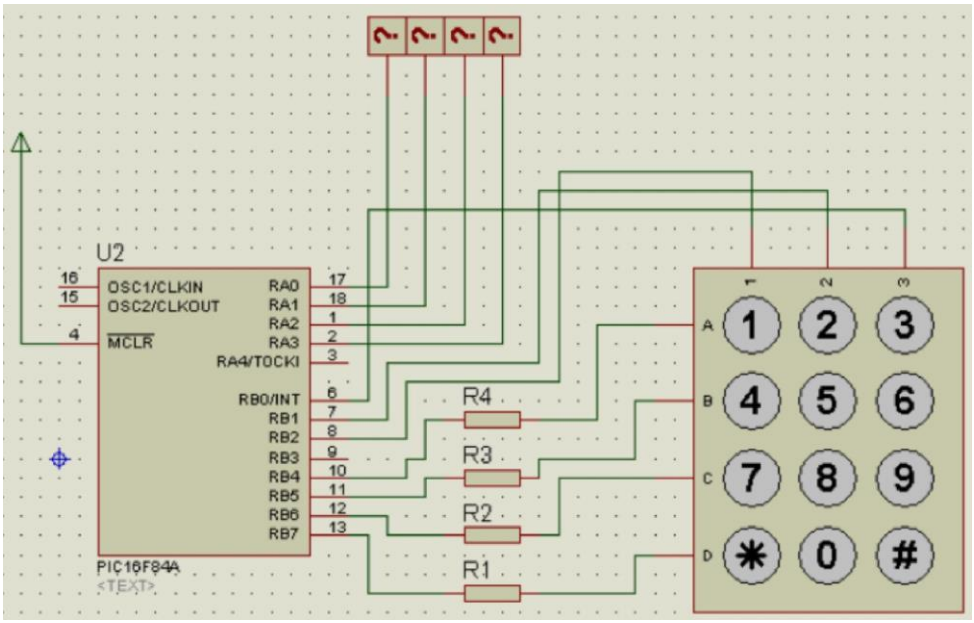
Генератор кодов состоит из микроконтроллера PIC16F84A, матричной клавиатуры и 4-х выводов для генерируемых кодов.

Увеличивающаяся двоичная последовательность (0000,…,1111);

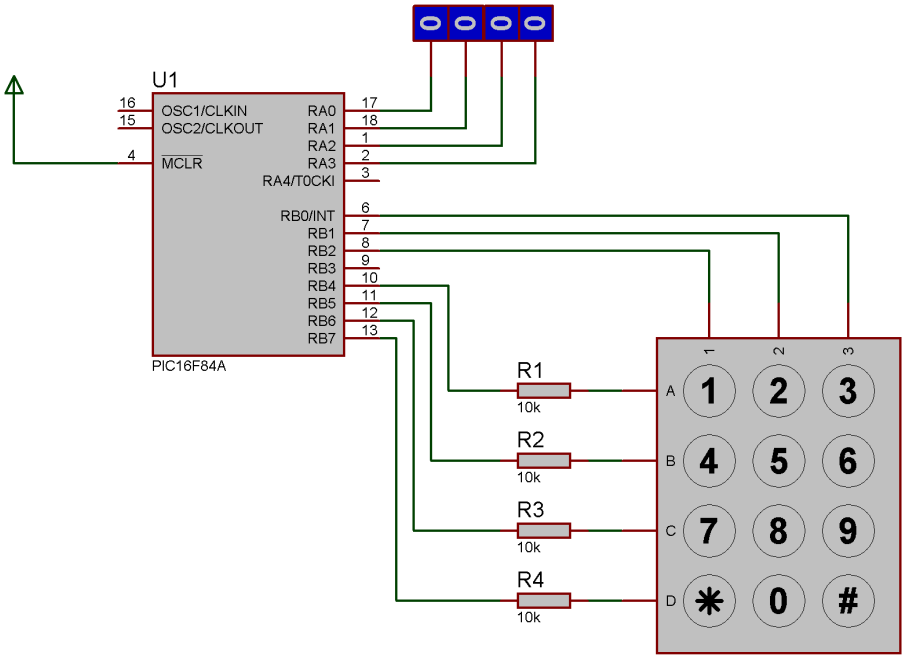
Микроконтроллер генерирует последовательность кодов при нажатии на клавиатуре кнопки «1».

Задержка между кодами равна 1 сек. При нажатии на другие кнопки микроконтроллер прекращает генерацию кодов.

Схема устройства:



Скриншот собранной схемы



Листинг программы

```
/*
 * File:   main.c
 * Author: Krystsina Bahdanava, 581072
 *
 * Created on 28 марта 2018 г., 18:05
 */


#include <stdio.h>
#include <stdlib.h>
#include <xc.h>

#define ASTERIX_SYM 10
#define HASH_SYM 11
#define NO_KEY_PRESSED 12

#define R1 RB4
#define R2 RB5
#define R3 RB6
#define R4 RB7

#define C1 RB2
#define C2 RB1
#define C3 RB0

#define INTERRUPT_FQ 4
#define TMR0_PRELOAD 12
#define PRESCALER_RATE 0b111

unsigned char codeType;
unsigned char code;

char readKey() {
    C1 = 1;
    C2 = 0;
    C3 = 0;
    if (R1) {
        while (R1) ;
        return 1;
    } else if (R2) {
        while (R2) ;
        return 4;
    } else if (R3) {
```

```
        while (R3) ;
        return 7;
    } else if (R4) {
        while (R4) ;
        return ASTERIX_SYM;
    }

    C1 = 0;
    C2 = 1;
    C3 = 0;
    if (R1) {
        while (R1) ;
        return 2;
    } else if (R2) {
        while (R2) ;
        return 5;
    } else if (R3) {
        while (R3) ;
        return 8;
    } else if (R4) {
        while (R4) ;
        return 0;
    }

    C1 = 0;
    C2 = 0;
    C3 = 1;
    if (R1) {
        while (R1) ;
        return 3;
    } else if (R2) {
        while (R2) ;
        return 6;
    } else if (R3) {
        while (R3) ;
        return 9;
    } else if (R4) {
        while (R4) ;
        return HASH_SYM;
    }

    while (TMR0 % 5) ;
    return NO_KEY_PRESSED;
}
```

```
void nextCode() {
    if (codeType == 1) {
        code += 1;
    }
    code &= 0xF;
}

void initCode() {
    if (codeType == 1) {
        code = 0b0000;
    }
    PORTA = code;
}

unsigned char interrupts = 0;
void interrupt tcInt() {
    if (TMR0IF) {
        interrupts++;
        if (interrupts == INTERRUPT_FQ) {
            interrupts = 0;
            nextCode();
            PORTA = code;
        }
        INTCONbits.TMR0IF = 0;
        INTCONbits.TMR0IE = 1;
        TMR0 = TMR0_PRELOAD;
    }
}

void startTimer() {
    interrupts = 0;
    INTCONbits.TMR0IF = 0;
    INTCONbits.TMR0IE = 1;
    TMR0 = TMR0_PRELOAD;
}

void stopTimer() {
    INTCONbits.TMR0IE = 0;
}

int main(int argc, char** argv) {
    codeType = 0;
    code = 0;

    OPTION_REGbits.T0CS = 0;
```

```
    OPTION_REGbits.PSA = 0;
    OPTION_REGbits.PS = PRESCALER_RATE;
    TMR0 = TMR0_PRELOAD;

    INTCON = 0;
    INTCONbits.TMR0IE = 0;
    INTCONbits.TMR0IF = 0;
    INTCONbits.GIE = 1;

    TRISA = 0b10000;
    TRISB = 0b11111000;

    PORTA = 0;
    PORTB = 0;

    char key;
    while (1) {
        while ((key = readKey()) == NO_KEY_PRESSED) ;
        codeType = key;
        initCode();
        if (codeType == 1) {
            startTimer();
        } else {
            stopTimer();
            PORTA = 0;
        }
    }
    return (EXIT_SUCCESS);
}
```