МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт цифровых технологий, электроники и физики

Кафедра вычислительной техники и электроники (ВТиЭ)

Лабораторная работа №2

Разработка микропроцессорных систем на базе микроконтроллера PIC16F84

Выполнил студент 595 гр.

_____ А.В. Лаптев

Проверил:

_____ В.В. Белозерских

Лабораторная работа защищена

«___» _____2023 г.

Оценка _____

Барнаул 2023

Цель работы:

Получение навыков программирования микропроцессорных систем на базе микроконтроллеров PIC16F84, при помощи среды разработки MPLAB.

Задачи:

Изучить основы программирования микропроцессорных систем на базе микроконтроллера PIC16F84. Написать и отладить управляющую программу с помощью пакета MPLAB и произвести программирование контроллера. Работоспособность созданного устройства проверить на лабораторном стенде.

Задание 1.

Написать программу сложения числа, вводимого с линий порта А МК, и числа 5. Вывод результатов осуществить в порт В.

Алгоритм:

1. Начало
2. Настраиваем порт А как вход
3. Настраиваем порт В как выход
4. Читаем старший байта из порта А
5. Переносим значение в память и сдвигаем на 4 позиции
6. Читаем младший байт из порта А и записываем его в свободные 4 бита ячейки памяти
7. Складываем полученное число с числом 5
8. Записываем результат в порт В
9. Очищаем регистр W
10. Заносим перенос в W
11. Записываем результат в порт В
12. Конец

Блок-схема:

Листинг:

```
include "p16f84.inc" ; подключаем файл с описанием регистров
;Переменные
First EQU 10h
;Устанавливаем вектор сброса
ORG 0
GOTO Start
Start
        BCF STATUS, RP1
        BSF STATUS, RP0 ; Выбираем банк памяти 1
        MOVLW B'00011111' ; Настраиваем порт A как вход
        MOVWF TRISA
        MOVLW B'00000000' ; Настраиваем порт B как выход
        MOVWF TRISB
        BCF STATUS, RP0 ; Выбираем банк памяти 0
Metka:
        CLRW
        CLRF First
        MOVF PORTA,0 ; Читаем старшую половину байта из порта A в W
        ANDLW 0x0F ; Убираем старшую половину байта
```

```
        MOVWF First ; Переносим значение в ячейку
        SWAPF First, 1
        MOVF PORTA,0 ; Читаем младшую половину байта
        ANDLW 0x0F ; Убираем старшую половину байта
        IORWF First,0 ; Собираем обе половины байта, результат в W
        ADDLW 5 ; Складываем W с числом 5, результат в W
        MOVWF PORTB ; Записываем результат в порт B
        CLRW ; Очищаем регистр W
        BTFSC STATUS,C
        INCF W,0
        MOVWF PORTB ; Записываем перенос в порт B
        GOTO Metka
END
```

Вывод: Задание выполнено успешно. В ходе выполнения задания была изучена работа портов ввода-вывода, а также применен на практике способ работы с числами, разрядность которых превышает разрядность порта ввода-вывода.

Задание 2.

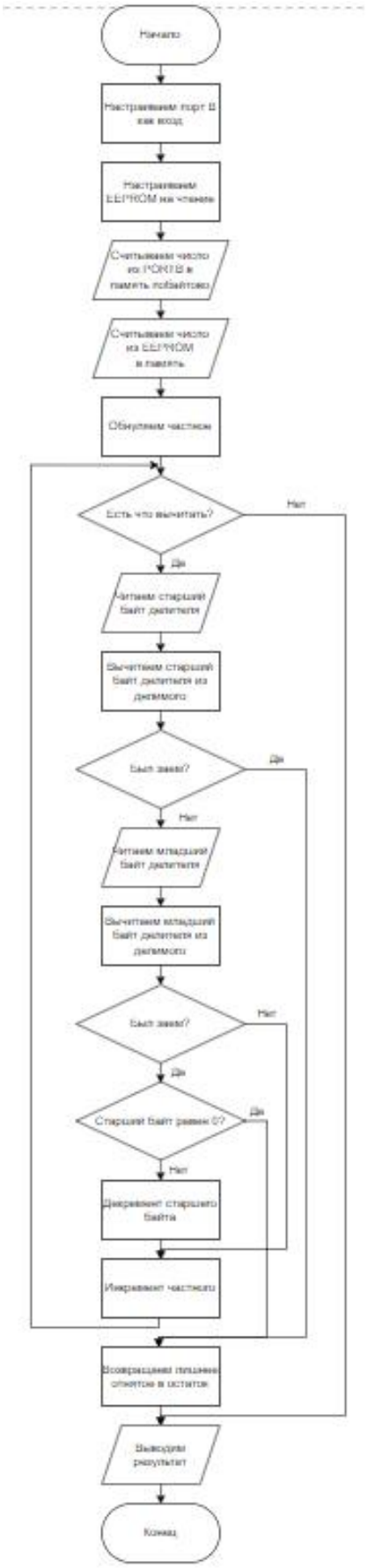
Написать программу деления двух двухбайтных чисел, одно последовательно считывается из порта B, другое находиться в EEPROM. Результат записать в память (частное и остаток от деления).

Алгоритм:

1. Начало
2. Настраиваем порт B как вход
3. Настраиваем EEPROM на чтение
4. Считываем число из PORTB побайтово и заносим в память
5. Считываем число из EEPROM и заносим в память
6. Обнуляем частное
7. Есть что вычитать? Если нет, то переход к п.18.
8. Читаем старший байт делителя в W
9. Вычитаем старший байт делителя из делимого
10. Был заем? Если был, то переход к п. 17.
11. Читаем младший байт делителя в W
12. Вычитаем младший байт делителя из делимого
13. Был заем? Если не был, то переход к п. 16.
14. Старший байт равен 0? Если да, то переход к п.17.
15. Уменьшаем старший байт на 1
16. Увеличиваем частное и переходим к п. 7
17. Возвращаем лишнее отнятое значение
18. Выводим результат

19. Конец

Блок-схема:



Листинг:

```
; используем процессор PIC16F84, система исчисления десятичная
include "p16f84.inc" ; подключаем файл с описанием регистров
address_data            EQU 24h
address_EE              EQU 25h
it                      EQU 23h
DelimoeH        EQU 010h
DelimoeL        EQU 011h
DelitelH                EQU 020h
DelitelL        EQU 021h
ChastnoeH               EQU 014h
ChastnoeL       EQU 013h
; Устанавливаем вектор сброса
ORG             0
GOTO            Start
Start:
        BCF             STATUS, RP1
        BSF             STATUS, RP0 ; Выбираем банк памяти 1
        MOVLW  B'11111111' ; Настраиваем порт B как вход
        MOVWF  TRISB
        ; берем 2 байтовое число из порта B
        BCF             STATUS, RP0 ; Выбираем банк памяти 0
        MOVF            PORTB, 0
        MOVWF  DelitelH
        MOVF            PORTB, 0
        MOVWF  DelitelL
        ; берем второе число из EEPROM
        BCF             STATUS, RP0    ; Bank 0
        MOVLW  10h
        MOVWF  EEADR        ; Address to read
        BSF             STATUS, RP0    ; Bank 1
        BSF             EECON1, RD     ; EE Read
        BCF             STATUS, RP0    ; Bank 0
        MOVF            EEDATA, 0
        MOVWF  DelimoeH
        BCF             STATUS, RP0    ; Bank 0
        MOVLW  11h
        MOVWF  EEADR        ; Address to read
        BSF             STATUS, RP0    ; Bank 1
        BSF             EECON1, RD     ; EE Read
        BCF             STATUS, RP0    ; Bank 0
        MOVF            EEDATA, 0
        MOVWF  DelimoeL
GOTO            Delenie
Delenie
        CLRF    ChastnoeH           ; Обнуляем частное и остаток
        CLRF    ChastnoeL
Cycle   ; Циклически отнимаем делитель из делимого
        MOVF            DelitelH,0 ; Читаем значение делителя в W
        SUBWF           DelimoeH,1          ; Вычитаем делитель из делимого, результат НЕ в W
        BTFSS           STATUS, C           ; Если при вычитании мы не занимали, то пропускаем обработку старшего регистра
GOTO    EndDiv
```

```
        MOVF        DelitelL,0 ; Читаем значение делителя в W
        SUBWF       DelimoeL,1      ; Вычитаем делитель из делимого, результат НЕ в W
        BTFSS       STATUS, C       ; Если при вычитании мы не занимали, то пропускаем обработку старшего регистра
GOTO    CheckHighByte       ; Иначе переходим на обработку старшего байта
RetCycle                    ; Метка возврата из CheckHighByte
        INCF        ChastnoeL,1     ; Увеличиваем частное
        BTFSC       STATUS, Z   ; Если при увеличении произошло переполнение, то...
        INCF        ChastnoeH,1     ; ...увеличиваем старший байт частного
GOTO Cycle          ; Повторяем до посинения
CheckHighByte       ; Проверка старшего байта
        MOVF        DelimoeH,1      ; Проверяем, не является ли старший байт нулем
        BTFSC       STATUS, Z   ; Если старший байт - не ноль, то пропускаем переход
        GOTO        EndDiv          ; Если старший байт - ноль (неоткуда занимать), то переходим в конец
        DECF        DelimoeH,1      ; Уменьшаем старший байт делимого
GOTO RetCycle       ; Возвращаемся обратно
EndDiv  ; Действие, если мы проскакали 0 (отняли лишнего) (деление закончено)
        MOVF        DelitelH, 0
        ADDWF   DelimoeH, 1
Endless:
        ;Выводим результат
        MOVLW       h'2'
        MOVWF       it
        MOVLW       h'11'
        MOVWF       address_EE
        MOVLW       h'13'
        MOVWF       address_data
Mark:
        MOVF        address_data, 0
        MOVWF   FSR
        MOVF        INDF, 0
        MOVWF   EEDATA
        MOVF        address_EE, 0
        MOVWF   EEADR
        BCF         STATUS, RP1 ; Выбираем банк памяти 1
        BSF         STATUS, RP0
        bsf         EECON1, 2
        movlw       h'55'
        movwf       EECON2
        movlw       h'AA'
        movwf       EECON2
        bsf         EECON1, WR ; установить WR бит, начать запись
Viv:
        BTFSS       EECON1, EEIF
        GOTO        Viv
        BCF         EECON1, EEIF
        BCF         STATUS, RP0 ; Выбираем банк памяти 0
        INCF        address_EE
        INCF        address_data
        DECFSZ      it, 1
GOTO    Mark
END
```

Вывод: Задание выполнено успешно. В ходе выполнения задания была изучена работа с числами, разрядность которых превышает разрядность АЛУ. Также было произведено ознакомление с работой разных видов памяти микроконтроллера.

Задание 3.

Написать программу временной задержки, используя программный алгоритм, основанный на применении циклов. Время задержки 3,3с.

Заданная задержка - 3,3с.

Расчет погрешности:

$$\Delta x = 3{,}3 - 3{,}29985 = 0.00015;$$

$$x_{отн} = \frac{0{,}00015}{3{,}29985} 100\% = 0.004\ \%.$$

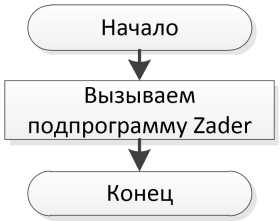
Алгоритм основной программы:

1. Начало
2. Вызываем подпрограмму задержки
3. Конец

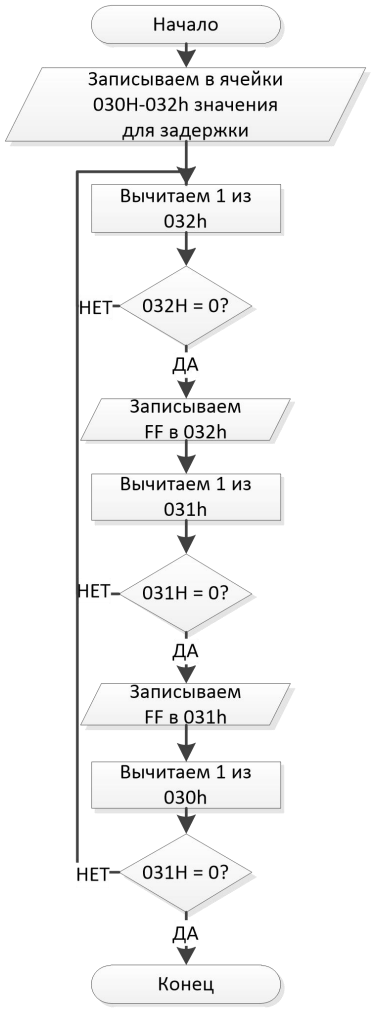
Алгоритм подпрограммы задержки:

1. Начало
2. Вводим начальные задержки для формирования задержки
3. Уменьшаем младший байт на 1
4. Если младший байт не равен 0, то п.3
5. Уменьшаем средний байт на 1
6. Если средний байт не равен 0, то п.3
7. Уменьшаем старший байт на 1
8. Если старший байт не равен 0, то п.3
9. Выход из подпрограммы

Блок-схема основной программы:



Блок-схема подпрограммы задержки:



Листинг:

```
; используем процессор PIC16F84, система исчисления десятичная
include "p16f84.inc" ; подключаем файл с описанием регистров
;Фстанавливаем вектор сброса
ORG 0
        GOTO Start
Start
        CALL Zader ;вызываем подпрограмму задержки
        GOTO Start
Zader
        ;вводим начальные данные для формирования задержки
        MOVLW h'12' ;старший байт
        MOVWF 030h
        MOVLW h'5E' ;средний байт
        MOVWF 031h
        MOVLW h'33' ;младший байт
        MOVWF 032h
Metka:
        DECFSZ 032h,1 ;уменьшаем значение в ячейке 032, пока она не станет разной нулю
        GOTO Metka
        MOVLW h'F9' ;записываем в ячейку 032 F9
        MOVWF 032h
        DECFSZ 031h,1 ;уменьшаем значение в ячейке 031
        GOTO Metka ;возвращаемся к уменьшению младшего байта, если средний не равен 0
        MOVLW h'FD' ;записываем в ячейку 031 FD
        MOVWF 031h
        DECFSZ 030h,1 ;уменьшаем значение в ячейке 030
        GOTO Metka ;возвращаемся к уменьшению младшего байта, если старший не равен 0
Return
END
```

Вывод: Задание выполнено успешно. В ходе работы была изучена реализация временной задержки с помощью подпрограмм. По результатам симуляции, относительная погрешность вышеописанной программы составляет 0.004%.

Задание 4.

Написать программу временной задержки, используя таймер/счетчик МК. Время задержки – 9,9с.

Заданная задержка - 9,9с.

Расчет погрешности:

$$\Delta x = 9{,}9 - 9{,}89994 = 0.00006;$$

$$x_{отн} = \frac{0.000139}{9{,}89994} 100\% = 0{,}0006\ \%.$$

Алгоритм основной программы:

1. Начало
2. Настраиваем таймер

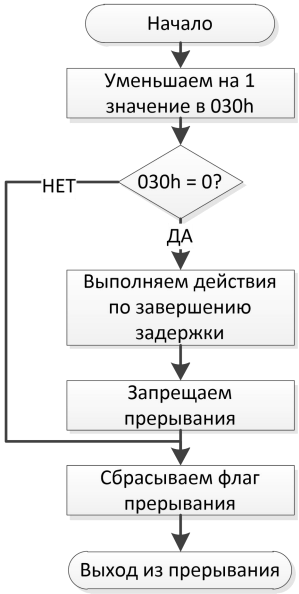
3. Разрешаем прерывания
4. Записываем FF в ячейку 030
5. Записываем значение 02 в TMR0
6. Запускаем таймер
7. Конец

Алгоритм обработки прерывания таймера:

1. Начало
2. Уменьшаем значение в ячейке 030 на 1
3. Если значение не равно 0, то п.6
4. Выполняем действие по завершению задержки
5. Запрещаем прерывания
6. Сбрасываем флаг прерывания
7. Выходим из прерывания

Блок-схема основной программы:

Блок-схема обработки прерывания от таймера:



Листинг:

```
; используем процессор PIC16F84, система исчисления десятичная
include "p16f84.inc" ; подключаем файл с описанием регистров
;Устанавливаем вектор сброса
ORG 0
GOTO Start
ORG 0004h
GOTO Timer
Timer:
        DECFSZ 030h, 1
        GOTO next
        MOVLW B'00000000' ;запрещаем прерывания
        MOVWF INTCON
next:
        BCF INTCON,02h
        RETFIE
Start
        BCF STATUS, RP1
        BSF STATUS, RP0 ; Выбираем банк памяти 1
        ;настраиваем таймер
        MOVLW B'00000111'
        MOVWF OPTION_REG
        BCF STATUS, RP0 ; Выбираем банк памяти 0
        MOVLW B'10100000' ;разрешаем прерывания
        MOVWF INTCON
        MOVLW h'98' ;записываем в ¤чейку 030 10
        MOVWF 030h
        MOVLW h'F0' ;записываем значение в TMR0
        MOVWF TMR0 ;запускаем таймер
loop:
        BTFSC INTCON, 5
        GOTO loop
        GOTO Start
END
```

Вывод: Задание выполнено успешно. В ходе работы была изучена реализация временной задержки с помощью таймера, т.е. аппаратным способом. Также было произведено ознакомление с работой прерываний. По результатам симуляции, относительная погрешность вышеописанной программы составляет 0.0006%.

Задание 5.

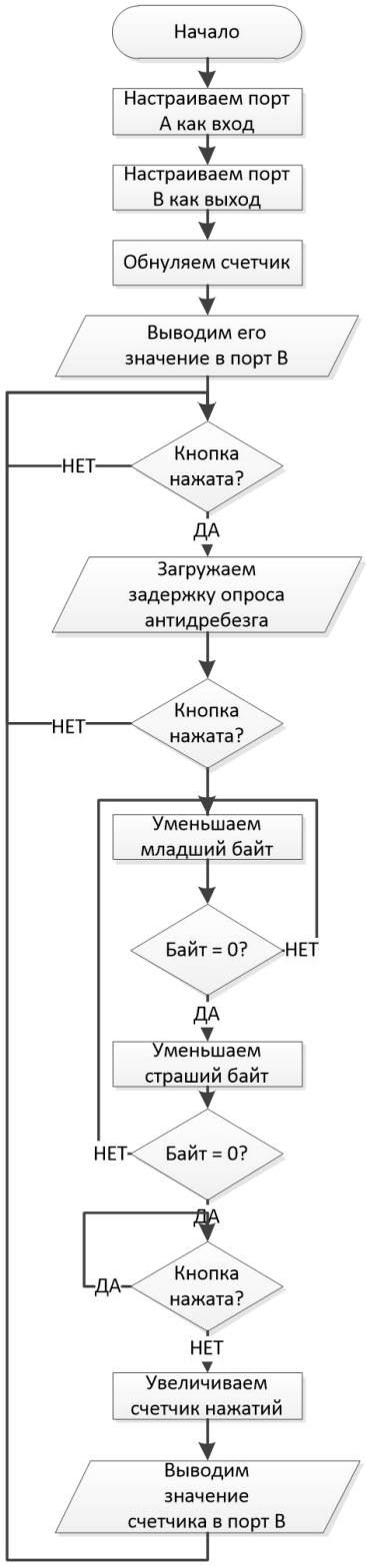
Разработать индикатор количества нажатий кнопки в двоичном коде (Обратный двоичный счетчик) номер кнопки — 0.

Алгоритм основной программы:

1. Начало
2. Настраиваем порт А как вход
3. Настраиваем порт В как выход
4. Обнуляем счетчик

5. Выводим его значение в порт B
6. Если кнопка не нажата, то п.6
7. Загружаем задержку опроса антидребезга
8. Если кнопка не нажата, то п.6
9. Уменьшаем младший байт на 1
10. Если младший байт не равен 0, то п.9
11. Уменьшаем старший байт на 1
12. Если старший байт не равен 0, то п.9
13. Если кнопка нажата, то п.13
14. Увеличиваем счетчик нажатий
15. Выводим значение счетчика нажатий
16. Переходим на п.6

Блок-схема:


Начало
Настраиваем порт A как вход
Настраиваем порт B как выход
Обнуляем счетчик
Выводим его значение в порт B
Кнопка нажата?
НЕТ
ДА
Загружаем задержку опроса антидребезга
Кнопка нажата?
НЕТ
Уменьшаем младший байт
Байт = 0?
НЕТ
ДА
Уменьшаем страший байт
Байт = 0?
НЕТ
ДА
Кнопка нажата?
ДА
НЕТ
Увеличиваем счетчик нажатий
Выводим значение счетчика в порт B

Листинг:

```
include "p16f84.inc"
PushCounter EQU 10h;Счетчик нажатий
Counter0 EQU 11h; Delay
Counter1 EQU 12h; Delay
Counter2 EQU 13h; Delay
org 0
GOTO Start
Start
        BSF STATUS, RP0
        MOVLW b'00011111'; A: 4-0 входы
        MOVWF TRISA; Заносим конфигурацию порта A
        MOVLW b'00000000'; B: 7-0 выходы
        MOVWF TRISB; Заносим конфигурацию порта B
        BCF STATUS, RP0; Выбираем банк памяти 0
        MOVLW B'00000000'; инициализация порта B нулем
        MOVWF PORTB
        MOVLW 0x00; инициализация счетчика нулем
        MOVWF PushCounter
CheckPushButton; проверка на нажатие 0 кнопки
        BTFSC PORTA,RA0
        GOTO CheckPushButton; если не нажата возвращаемся обратно
        CALL Delay ; 10мс если нажата кнопка переходим на подпрограмму задержки(дребезг)
CheckReleaseButton; кнопка отпущена
        BTFSS PORTA,RA0
        GOTO CheckReleaseButton
        INCF PushCounter; если кнопка отпущена инкрементируем счетчик
        MOVFW PushCounter
        MOVWF PORTB; вывод значения счетчика в порт B
        GOTO CheckPushButton
Delay
        MOVLW 02h
        MOVWF Counter0 ; мс
        MOVLW 0Dh
        MOVWF Counter1 ; мс
        MOVLW 0FFh
        MOVWF Counter2 ; мс
        Count0
        Count1
        Count2
        DECFSZ Counter2
        GOTO Count2
        DECFSZ Counter1
        GOTO Count1
        DECFSZ Counter0
        GOTO Count0 ; мс
RETURN ; мс
END
```

Вывод: Задание выполнено успешно. Работа программы протестирована на макетной плате. В ходе работы были получены навыки работы с программатором, а также изучены особенности взаимодействия микропроцессорной системы с пользователем.

Задание 6.

Разработать «бегущие огни» не менее, чем на 8 режимов с изменением направления движения при нажатии на кнопку.

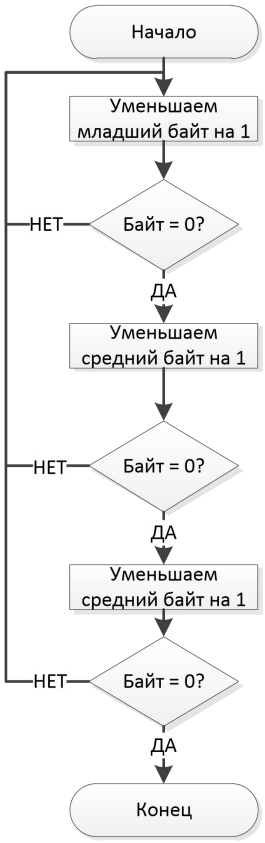
Алгоритм основной программы:

1. Начало
2. Настраиваем порт А как вход
3. Настраиваем порт В как выход
4. Обнуляем номер режима
5. Заносим 1 в счетчик
6. Выводим его значение в порт В
7. Запускаем первый режим
8. Если режим изменился, то переходим на п. 14
9. Если режим изменился на инверсный, то переходим к п.15.
10. Отрисовываем кадр режима
11. Загружаем биты задержки
12. Вызываем подпрограмму задержки
13. Переходим на п.8
14. Запускаем следующий по счету режим и переходим к п. 8
15. Запускаем инверсный режим того же номера и переходим к п. 8

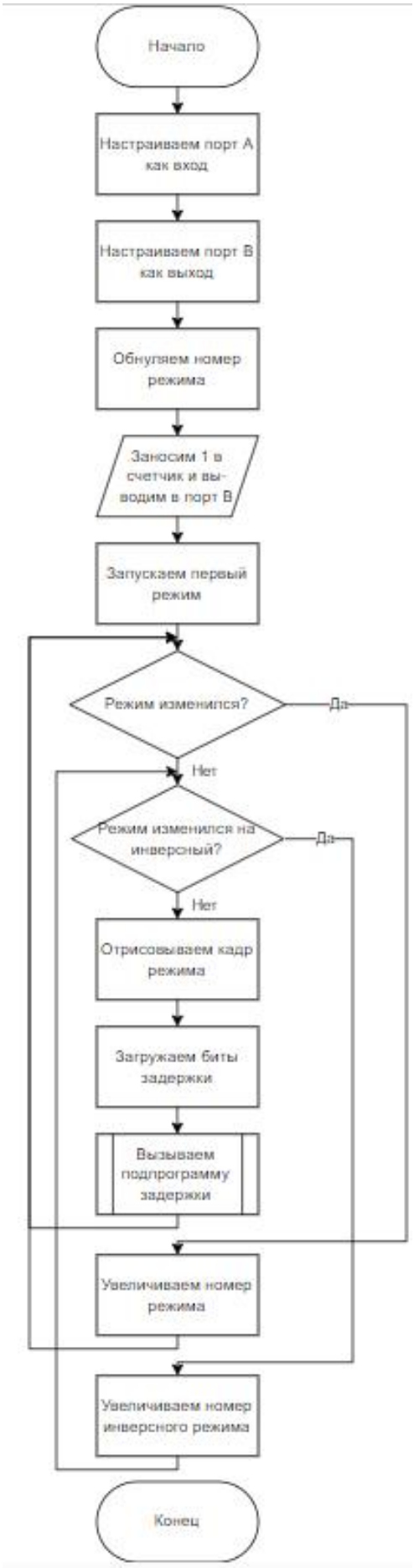
Алгоритм подпрограммы задержки:

1. Начало
2. Уменьшаем младший байт на 1
3. Если младший байт не равен 0, то п.2
4. Уменьшаем средний байт на 1
5. Если средний байт не равен 0, то п.2
6. Уменьшаем старший байт на 1
7. Если старший байт не равен 0, то п.2
8. Выход из подпрограммы

Блок-схема подпрограммы задержки:



Блок-схема основной программы:



Листинг:

```
include "p16f84.inc" ; подключаем файл с описанием регистров
; Переменные
Counter    EQU 10h                          ; Счетчик нажатий
DelJr      EQU 11h                          ; Младший байт задержки
DelMl      EQU 12h                          ; Средний байт задержки
DelSr   EQU 13h                             ; Старший байт задержки
ORG 0                                       ; Устанавливаем вектор сброса
           GOTO Start
; Основная программа
Start
           BCF      STATUS, RP1
           BSF      STATUS, RP0             ; Выбираем банк памяти 1
           MOVLW  B'00011111'               ; Маска порта A: 4-0 входы
           MOVWF  TRISA                     ; Заносим конфигурацию порта A
           MOVLW  B'00000000'               ; Маска порта B: 7-0 выходы
           MOVWF  TRISB                     ; Заносим конфигурацию порта B
           BCF      STATUS, RP0             ; Выбираем банк памяти 0
           MOVLW  1
           MOVWF  Counter                   ; заносим 1 в счетчик
           MOVF     Counter, 0
           MOVWF  PORTB                     ; Выводим его значение
           ;0 - кнопка нажата
           ;1 - кнопка не нажата
           GOTO     Mode_1
; Режимы
Mode_1
b31
           btfsc    PORTA, 3
           goto     b41
           call     Anti_bounce
           goto     Mode_2
b41
           btfsc    PORTA, 4
           goto     rez1
           call     Anti_bounce
           goto     Invert_mode_1
rez1
           RRF      Counter, 1      ; Сдвигаем вправо
           MOVF     Counter, 0
           MOVWF  PORTB                     ; Выводим новое положение
           MOVLW  0x2                               ; Загружаем задержку
           MOVWF  DelSr
           MOVLW  0x8
           MOVWF  DelMl
           MOVLW  0x7A
           MOVWF  DelJr
           CALL     Delay
           GOTO     Mode_1
Mode_2
b32
           btfsc    PORTA, 3
```

```
        goto    b42
        call    Anti_bounce
        goto    Mode_3
b42
        btfsc   PORTA, 4
        goto    rez2
        call    Anti_bounce
        goto    Invert_mode_2
rez2
        BSF             PORTB, RB0
        BSF             PORTB, RB1
        BCF             PORTB, RB2
        BCF             PORTB, RB3
        BSF             PORTB, RB4
        BSF             PORTB, RB5
        BCF             PORTB, RB6
        BCF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
b322
        btfsc   PORTA, 3
        goto    b422
        call    Anti_bounce
        goto    Mode_3
b422
        btfsc   PORTA, 4
        goto    rez22
        call    Anti_bounce
        goto    Invert_mode_2
rez22
        BSF             PORTB, RB0
        BSF             PORTB, RB1
        BSF             PORTB, RB2
        BSF             PORTB, RB3
        BCF             PORTB, RB4
        BCF             PORTB, RB5
        BSF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
        GOTO    Mode_2
Mode_3
b33
```

```
        btfsc   PORTA, 3
        goto    b43
        call    Anti_bounce
        goto    Mode_4
b43
        btfsc   PORTA, 4
        goto    rez3
        call    Anti_bounce
        goto    Invert_mode_3
rez3
        RRF     Counter, 1 ; Сдвигаем вправо
        RRF     Counter, 1 ; Сдвигаем вправо
        MOVF    Counter, 0
        MOVWF   PORTB           ; Выводим новое положение
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
        GOTO    Mode_3
Mode_4
b34
        btfsc   PORTA, 3
        goto    b44
        call    Anti_bounce
        goto    Mode_5
b44
        btfsc   PORTA, 4
        goto    rez4
        call    Anti_bounce
        goto    Invert_mode_4
rez4
        BSF             PORTB, RB0
        BSF             PORTB, RB1
        BSF             PORTB, RB2
        BCF             PORTB, RB3
        BSF             PORTB, RB4
        BSF             PORTB, RB5
        BSF             PORTB, RB6
        BCF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
b342
        btfsc   PORTA, 3
        goto    b442
        call    Anti_bounce
```

```
        goto    Mode_5
b442
        btfsc   PORTA, 4
        goto    rez42
        call    Anti_bounce
        goto    Invert_mode_4
rez42
        BCF             PORTB, RB0
        BCF             PORTB, RB1
        BCF             PORTB, RB2
        BSF             PORTB, RB3
        BCF             PORTB, RB4
        BCF             PORTB, RB5
        BCF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
        GOTO    Mode_4
Mode_5
b35
        btfsc   PORTA, 3
        goto    b45
        call    Anti_bounce
        goto    Mode_6
b45
        btfsc   PORTA, 4
        goto    rez5
        call    Anti_bounce
        goto    Invert_mode_5

rez5
        BSF             PORTB, RB0
        BSF             PORTB, RB1
        BSF             PORTB, RB2
        BSF             PORTB, RB3
        BSF             PORTB, RB4
        BCF             PORTB, RB5
        BCF             PORTB, RB6
        BCF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
b352
        btfsc   PORTA, 3
```

```
        goto    b452
        call    Anti_bounce
        goto    Mode_6
b452
        btfsc   PORTA, 4
        goto    rez52
        call    Anti_bounce
        goto    Invert_mode_5

rez52
        BCF             PORTB, RB0
        BCF             PORTB, RB1
        BCF             PORTB, RB2
        BCF             PORTB, RB3
        BCF             PORTB, RB4
        BSF             PORTB, RB5
        BSF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
        GOTO    Mode_5
Mode_6
b36
        btfsc   PORTA, 3
        goto    b46
        call    Anti_bounce
        goto    Mode_7
b46
        btfsc   PORTA, 4
        goto    rez6
        call    Anti_bounce
        goto    Invert_mode_6
rez6
        BCF             PORTB, RB0
        BCF             PORTB, RB1
        BCF             PORTB, RB2
        BCF             PORTB, RB3
        BCF             PORTB, RB4
        BCF             PORTB, RB5
        BCF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
```

```
b362
        btfsc   PORTA, 3
        goto    b462
        call    Anti_bounce
        goto    Mode_7
b462
        btfsc   PORTA, 4
        goto    rez62
        call    Anti_bounce
        goto    Invert_mode_6
rez62
        BSF             PORTB, RB0
        BSF             PORTB, RB1
        BSF             PORTB, RB2
        BSF             PORTB, RB3
        BSF             PORTB, RB4
        BSF             PORTB, RB5
        BSF             PORTB, RB6
        BCF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
        GOTO    Mode_6
Mode_7
b37
        btfsc   PORTA, 3
        goto    b47
        call    Anti_bounce
        goto    Mode_8
b47
        btfsc   PORTA, 4
        goto    rez7
        call    Anti_bounce
        goto    Invert_mode_7
rez7
        BCF             PORTB, RB0
        BSF             PORTB, RB1
        BSF             PORTB, RB2
        BSF             PORTB, RB3
        BSF             PORTB, RB4
        BSF             PORTB, RB5
        BSF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
```

```
        CALL    Delay
b371
        btfsc   PORTA, 3
        goto    b471
        call    Anti_bounce
        goto    Mode_8
b471
        btfsc   PORTA, 4
        goto    rez71
        call    Anti_bounce
        goto    Invert_mode_7
rez71
        BCF             PORTB, RB0
        BCF             PORTB, RB1
        BCF             PORTB, RB2
        BSF             PORTB, RB3
        BSF             PORTB, RB4
        BSF             PORTB, RB5
        BSF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
b372
        btfsc   PORTA, 3
        goto    b472
        call    Anti_bounce
        goto    Mode_8
b472
        btfsc   PORTA, 4
        goto    rez72
        call    Anti_bounce
        goto    Invert_mode_7
rez72
        BCF             PORTB, RB0
        BCF             PORTB, RB1
        BCF             PORTB, RB2
        BCF             PORTB, RB3
        BCF             PORTB, RB4
        BCF             PORTB, RB5
        BSF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
```

```
b373
        btfsc   PORTA, 3
        goto    b473
        call    Anti_bounce
        goto    Mode_8
b473
        btfsc   PORTA, 4
        goto    rez73
        call    Anti_bounce
        goto    Invert_mode_7
rez73
        BCF             PORTB, RB0
        BCF             PORTB, RB1
        BCF             PORTB, RB2
        BCF             PORTB, RB3
        BCF             PORTB, RB4
        BCF             PORTB, RB5
        BCF             PORTB, RB6
        BCF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
        GOTO    Mode_7
Mode_8
b38
        btfsc   PORTA, 3
        goto    b48
        call    Anti_bounce
        goto    Mode_1
b48
        btfsc   PORTA, 4
        goto    rez8
        call    Anti_bounce
        goto    Invert_mode_8
rez8
        BSF             PORTB, RB0
        BSF             PORTB, RB1
        BCF             PORTB, RB2
        BCF             PORTB, RB3
        BCF             PORTB, RB4
        BCF             PORTB, RB5
        BCF             PORTB, RB6
        BCF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
```

```
        CALL    Delay
b382
        btfsc   PORTA, 3
        goto    b482
        call    Anti_bounce
        goto    Mode_1
b482
        btfsc   PORTA, 4
        goto    rez82
        call    Anti_bounce
        goto    Invert_mode_8
rez82
        BSF             PORTB, RB0
        BSF             PORTB, RB1
        BSF             PORTB, RB2
        BSF             PORTB, RB3
        BCF             PORTB, RB4
        BCF             PORTB, RB5
        BCF             PORTB, RB6
        BCF             PORTB, RB7
        MOVLW   0x2                     ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
b383
        btfsc   PORTA, 3
        goto    b483
        call    Anti_bounce
        goto    Mode_1
b483
        btfsc   PORTA, 4
        goto    rez83
        call    Anti_bounce
        goto    Invert_mode_8
rez83
        BSF             PORTB, RB0
        BSF             PORTB, RB1
        BSF             PORTB, RB2
        BSF             PORTB, RB3
        BSF             PORTB, RB4
        BSF             PORTB, RB5
        BCF             PORTB, RB6
        BCF             PORTB, RB7
        MOVLW   0x2                     ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
```

```
b384
        btfsc   PORTA, 3
        goto    b484
        call    Anti_bounce
        goto    Mode_1
b484
        btfsc   PORTA, 4
        goto    rez84
        call    Anti_bounce
        goto    Invert_mode_8
rez84
        BSF             PORTB, RB0
        BSF             PORTB, RB1
        BSF             PORTB, RB2
        BSF             PORTB, RB3
        BSF             PORTB, RB4
        BSF             PORTB, RB5
        BSF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                     ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
        GOTO    Mode_8
; Обратные режимы
Invert_mode_1
b31i
        btfsc   PORTA, 3
        goto    b41i
        call    Anti_bounce
        goto    Mode_2
b41i
        btfsc   PORTA, 4
        goto    rez1i
        call    Anti_bounce
        goto    Mode_1
rez1i
        RLF     Counter, 1              ; Сдвигаем вправо
        MOVF    Counter, 0
        MOVWF   PORTB           ; Выводим новое положение
        MOVLW   0x2                     ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
        GOTO    Invert_mode_1
Invert_mode_2
b32i
```

```
        btfsc   PORTA, 3
        goto    b42i
        call    Anti_bounce
        goto    Mode_3
b42i
        btfsc   PORTA, 4
        goto    rez2i
        call    Anti_bounce
        goto    Mode_2
rez2i
        BCF             PORTB, RB0
        BCF             PORTB, RB1
        BSF             PORTB, RB2
        BSF             PORTB, RB3
        BCF             PORTB, RB4
        BCF             PORTB, RB5
        BSF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                     ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
b322i
        btfsc   PORTA, 3
        goto    b422i
        call    Anti_bounce
        goto    Mode_3
b422i
        btfsc   PORTA, 4
        goto    rez22i
        call    Anti_bounce
        goto    Mode_2
rez22i
        BSF             PORTB, RB0
        BSF             PORTB, RB1
        BCF             PORTB, RB2
        BCF             PORTB, RB3
        BSF             PORTB, RB4
        BSF             PORTB, RB5
        BSF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                     ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
        GOTO    Invert_mode_2
Invert_mode_3
```

```
b33i
        btfsc   PORTA, 3
        goto    b43i
        call    Anti_bounce
        goto    Mode_4
b43i
        btfsc   PORTA, 4
        goto    rez3i
        call    Anti_bounce
        goto    Mode_3
rez3i
        RLF             Counter, 1      ; Сдвигаем влево
        RLF             Counter, 1      ; Сдвигаем влево
        MOVF    Counter, 0
        MOVWF   PORTB           ; Выводим новое положение
        MOVLW   0x2                     ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
        GOTO    Invert_mode_3
Invert_mode_4
b34i
        btfsc   PORTA, 3
        goto    b44i
        call    Anti_bounce
        goto    Mode_5
b44i
        btfsc   PORTA, 4
        goto    rez4i
        call    Anti_bounce
        goto    Mode_4
rez4i
        BCF             PORTB, RB0
        BSF             PORTB, RB1
        BSF             PORTB, RB2
        BSF             PORTB, RB3
        BCF             PORTB, RB4
        BSF             PORTB, RB5
        BSF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                     ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
b342i
        btfsc   PORTA, 3
        goto    b442i
```

```
        call    Anti_bounce
        goto    Mode_5
b442i
        btfsc   PORTA, 4
        goto    rez42i
        call    Anti_bounce
        goto    Mode_4
rez42i
        BSF             PORTB, RB0
        BCF             PORTB, RB1
        BCF             PORTB, RB2
        BCF             PORTB, RB3
        BSF             PORTB, RB4
        BCF             PORTB, RB5
        BCF             PORTB, RB6
        BCF             PORTB, RB7
        MOVLW   0x2                     ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
        GOTO    Invert_mode_4
Invert_mode_5
b35i
        btfsc   PORTA, 3
        goto    b45i
        call    Anti_bounce
        goto    Mode_6
b45i
        btfsc   PORTA, 4
        goto    rez5i
        call    Anti_bounce
        goto    Mode_5
rez5i
        BCF             PORTB, RB0
        BCF             PORTB, RB1
        BCF             PORTB, RB2
        BSF             PORTB, RB3
        BSF             PORTB, RB4
        BSF             PORTB, RB5
        BSF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
b352i
        btfsc   PORTA, 3
```

```
        goto    b452i
        call    Anti_bounce
        goto    Mode_6
b452i
        btfsc   PORTA, 4
        goto    rez52i
        call    Anti_bounce
        goto    Mode_5
rez52i
        BSF             PORTB, RB0
        BSF             PORTB, RB1
        BSF             PORTB, RB2
        BCF             PORTB, RB3
        BCF             PORTB, RB4
        BCF             PORTB, RB5
        BCF             PORTB, RB6
        BCF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
        GOTO    Invert_mode_5
Invert_mode_6
b36i
        btfsc   PORTA, 3
        goto    b46i
        call    Anti_bounce
        goto    Mode_7
b46i
        btfsc   PORTA, 4
        goto    rez6i
        call    Anti_bounce
        goto    Mode_6
rez6i
        BSF             PORTB, RB0
        BCF             PORTB, RB1
        BCF             PORTB, RB2
        BCF             PORTB, RB3
        BCF             PORTB, RB4
        BCF             PORTB, RB5
        BCF             PORTB, RB6
        BCF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
b362i
```

```
        btfsc   PORTA, 3
        goto    b462i
        call    Anti_bounce
        goto    Mode_7
b462i
        btfsc   PORTA, 4
        goto    rez62i
        call    Anti_bounce
        goto    Mode_6
rez62i
        BCF             PORTB, RB0
        BSF             PORTB, RB1
        BSF             PORTB, RB2
        BSF             PORTB, RB3
        BSF             PORTB, RB4
        BSF             PORTB, RB5
        BSF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
        GOTO    Invert_mode_6
Invert_mode_7
b37i
        btfsc   PORTA, 3
        goto    b47i
        call    Anti_bounce
        goto    Mode_8
b47i
        btfsc   PORTA, 4
        goto    rez7i
        call    Anti_bounce
        goto    Mode_7
rez7i
        BSF             PORTB, RB0
        BSF             PORTB, RB1
        BSF             PORTB, RB2
        BSF             PORTB, RB3
        BSF             PORTB, RB4
        BSF             PORTB, RB5
        BSF             PORTB, RB6
        BCF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
```

```
b371i
        btfsc   PORTA, 3
        goto    b471i
        call    Anti_bounce
        goto    Mode_8
b471i
        btfsc   PORTA, 4
        goto    rez71i
        call    Anti_bounce
        goto    Mode_7
rez71i
        BSF             PORTB, RB0
        BSF             PORTB, RB1
        BSF             PORTB, RB2
        BSF             PORTB, RB3
        BSF             PORTB, RB4
        BCF             PORTB, RB5
        BCF             PORTB, RB6
        BCF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
b372i
        btfsc   PORTA, 3
        goto    b472i
        call    Anti_bounce
        goto    Mode_8
b472i
        btfsc   PORTA, 4
        goto    rez72i
        call    Anti_bounce
        goto    Mode_7
rez72i
        BSF             PORTB, RB0
        BSF             PORTB, RB1
        BCF             PORTB, RB2
        BCF             PORTB, RB3
        BCF             PORTB, RB4
        BCF             PORTB, RB5
        BCF             PORTB, RB6
        BCF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
b373i
```

```
        btfsc   PORTA, 3
        goto    b473i
        call    Anti_bounce
        goto    Mode_8
b473i
        btfsc   PORTA, 4
        goto    rez73i
        call    Anti_bounce
        goto    Mode_7
rez73i
        BCF             PORTB, RB0
        BCF             PORTB, RB1
        BCF             PORTB, RB2
        BCF             PORTB, RB3
        BCF             PORTB, RB4
        BCF             PORTB, RB5
        BCF             PORTB, RB6
        BCF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
        GOTO    Invert_mode_7
Invert_mode_8
b38i
        btfsc   PORTA, 3
        goto    b48i
        call    Anti_bounce
        goto    Mode_1
b48i
        btfsc   PORTA, 4
        goto    rez8i
        call    Anti_bounce
        goto    Mode_8
rez8i
        BCF             PORTB, RB0
        BCF             PORTB, RB1
        BCF             PORTB, RB2
        BCF             PORTB, RB3
        BCF             PORTB, RB4
        BCF             PORTB, RB5
        BSF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
```

```
b382i
        btfsc   PORTA, 3
        goto    b482i
        call    Anti_bounce
        goto    Mode_1
b482i
        btfsc   PORTA, 4
        goto    rez82i
        call    Anti_bounce
        goto    Mode_8
rez82i
        BCF             PORTB, RB0
        BCF             PORTB, RB1
        BCF             PORTB, RB2
        BCF             PORTB, RB3
        BSF             PORTB, RB4
        BSF             PORTB, RB5
        BSF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
b383i
        btfsc   PORTA, 3
        goto    b483i
        call    Anti_bounce
        goto    Mode_1
b483i
        btfsc   PORTA, 4
        goto    rez83i
        call    Anti_bounce
        goto    Mode_8
rez83i
        BCF             PORTB, RB0
        BCF             PORTB, RB1
        BSF             PORTB, RB2
        BSF             PORTB, RB3
        BSF             PORTB, RB4
        BSF             PORTB, RB5
        BSF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
b384i
```

```
        btfsc   PORTA, 3
        goto    b484i
        call    Anti_bounce
        goto    Mode_1
b484i
        btfsc   PORTA, 4
        goto    rez84i
        call    Anti_bounce
        goto    Mode_8
rez84i
        BSF             PORTB, RB0
        BSF             PORTB, RB1
        BSF             PORTB, RB2
        BSF             PORTB, RB3
        BSF             PORTB, RB4
        BSF             PORTB, RB5
        BSF             PORTB, RB6
        BSF             PORTB, RB7
        MOVLW   0x2                                     ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0x8
        MOVWF   DelMl
        MOVLW   0x7A
        MOVWF   DelJr
        CALL    Delay
        GOTO    Invert_mode_8
; Задержка
Delay
        DECFSZ DelJr,1          ; Уменьшаем младший байт, пока он не станет 0
        GOTO Delay
        DECFSZ DelMl,1          ; Уменьшаем средний байт, пока он не станет 0
        GOTO Delay
        DECFSZ DelSr,1          ; Уменьшаем старший байт, пока он не станет 0
        GOTO Delay
RETURN
Anti_bounce
        MOVLW   02h             ; Загружаем задержку
        MOVWF   DelSr
        MOVLW   0Dh
        MOVWF   DelMl
        MOVLW   0FFh
        MOVWF   DelJr
        CALL    Delay
RETURN
END
```

Вывод: Задание выполнено успешно. Работа программы протестирована на макетной плате. В ходе выполнения задания был получен навык написания программ, выполняющихся параллельно с воздействием пользователя.

Вывод: В ходе выполнения лабораторной работы были получены навыки программирования микропроцессорных систем на базе микроконтроллеров PIC16F84, при помощи среды разработки MPLAB.