Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**Лабораторная работа №4 (вариант 29)**

по дисциплине: «Автоматизация проектирования

микропроцессорных систем».

Выполнил:

студент 4 курса, гр. ИВТАПбд-41

Кондратьев Павел Сергеевич.

Проверил:

преподаватель кафедры ВТ

Игонин Андрей Геннадьевич.

г. Ульяновск, 2019

**Техническое задание**

Необходимо разработать модель схемы, выполняющую приведенные ниже функции. Моделирование производится с использование САПР Proteus 8 с использованием микропроцессора PIC16F814.

**Выполнение работы**

Прерывания в микроконтроллерах очень часто используются при разработке радиоэлектронных устройств, они позволяют реализовать многозадачность, то есть параллельное выполнение нескольких задач. Под прерыванием подразумевается условный переход из какой-либо области памяти программ (во время выполнения основной программы) на четко определенный адрес в этой памяти, данный адрес еще называют вектором прерываний. Векторов может быть несколько, но в микроконтроллерах PIC16 только один вектор прерываний, и это адрес в памяти программ равный 0004h. По этому адресу размещают подпрограмму обработки прерываний (обработчик прерываний).

Например, во время выполнения основной программы происходит прерывание от какого-либо источника, при этом в стек закладывается текущее значение счетчика команд PC (текущий адрес памяти программ на момент возникновения прерывания), после чего в счетчик команд загружается вектор прерываний и происходит переход на адрес 0004h в памяти программ. Здесь, как было сказано, находится подпрограмма обработки прерываний, в которой обычно выполняются другие задачи в отличии от основной программы, выход из обработчика прерывания выполняется командой retfie, при этом из стека в счетчик команд загружается ранее сохраненный адрес памяти программ, тем самым происходит возврат и дальнейшее выполнение основной программы. Таким образом, уходя на прерывания во время выполнения основной программы можно решать сразу несколько задач.

В микроконтроллерах PIC16 имеется несколько источников прерываний:

• внешнее прерывание по входу RB0/INT

• прерывание по переполнению таймера TMR0

• прерывание по изменению уровня сигнала на входах порта

Прерывания от периферийных модулей:

• прерывание по переполнению таймера TMR1

• прерывание по переполнению таймера TMR2

• прерывание от модуля компараторов

• прерывание по окончанию записи в EEPROM

В различных моделях микроконтроллеров реализовано разное количество источников прерываний, в зависимости от наличия тех или иных модулей. Наиболее часто используются (на моем опыте) следующие: внешнее прерывание по входу RB0/INT; прерывание по переполнению TMR0, TMR1; прерывание по изменению уровня сигнала на входах порта; реже прерывание от компараторов и по переполнению TMR2.

Каждое прерывание имеет собственный флаг (бит) состояния, при возникновении прерывания флаг устанавливается в 1 и происходит переход в обработчик прерываний. Флаги необходимо сбрасывать программно в подпрограмме обработки прерываний, если этого не сделать, то при выходе из подпрограммы произойдет повторный переход в обработчик. Прерывание для каждого источника можно разрешить или запретить. Регистры PIR1, PIR2 содержат флаги прерываний периферийных модулей, а биты для их разрешения находятся в регистрах PIE1, PIE2 (эти два регистра расположены в 1-ом банке памяти данных). В регистре INTCON содержатся биты разрешения и соответствующие флаги следующих прерываний: внешнее по входу RB0/INT, по переполнению TMR0, по изменению уровня сигнала на входах порта.

**Список литературы:**

1. Логические элементы И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ и их таблицы истинности <http://electricalschool.info/electronica/1918-logicheskie-jelementy-i-ili-ne-i-ne-ili.html> (Дата обращения 21.10.19).
2. Работа в Proteus. Сайт Паяльник <https://cxem.net/comp/comp117.php> (Дата обращения 21.10.19).
3. прерывания в pic16f628a https://alex-exe.ru/radio/microcontrollers/int-pic16f628a/ (Дата обращения 2.11.19).
4. Осваиваем простейший микроконтроллер PIC. Часть 1 https://habr.com/ru/post/97795/ (Дата обращения 23.11.19).
5. PIC16F877A Timer0 module and interrupt with CCS C compiler <https://simple-circuit.com/pic16f877a-timer0-module-interrupt/> (Дата обращения 23.11.19).
6. Прерывания https://radiolaba.ru/programmirovanie-pic-kontrollerov/ preryivaniya .html (Дата обращения 23.11.19).
7. PIC16 <https://radiolaba.ru/microcotrollers/chasy-na-gazorazryadnyh-indikatorah>. Html (Дата обращения 23.11.19).

**Приложение**

#define x1 PORTD.B0

#define x2 PORTD.B1

#define x3 PORTD.B2

#define x4 PORTD.B3

#define x5 PORTD.B4

#define t1 PORTC.B0

#define t2 PORTC.B1

#define t3 PORTC.B2

#define t4 PORTC.B3

#define y1 PORTB.B7

#define y2 PORTB.B1

#define y3 PORTB.B2

#define y4 PORTB.B3

#define y5 PORTB.B4

#define y6 PORTB.B5

#define y7 PORTB.B6

#define y8 PORTC.B4

#define y9 PORTC.B5

bit tmp;

char PIND;

volatile int timer1; //переменная значение которой может меняться из вне(из операционной системы или как в данной л/р по прерыванию)

void interrupt (void) //функция обработки прерываний

{

if (PIR2.EEIF) { //Прерывание по окончанию записи в ЕЕПРОМ. Тут считаем и выводим y11

if ((x1+x2+x3+x4+x5)%2==0) y9=1;

else y9=0;

PIR2.EEIF=0; //запрещаем прерфыывние EEPROM

}

else if (PIR1.TMR2IF)

{ //прерывание по переполнению Таймера 2, тут просто выводим y10 по таймеру

y8=x1&x3&x5;

PIR1.TMR2IF = 0;

}

else if (PIR1.TMR1IF) //прерывание по переполнению Таймера 1. Тут 3 с. ждем, если изменились входы, и выводим y1-y8.

{

if (++timer1==30)

{

y1=~x1;

y2=~x2;

y3=~x3;

y4=~x4;

y5=~x5;

timer1=0;

PIE1.TMR1IE = 0;// Запрещаем прерывание таймер1 (до тех пор, пока не изменится состояние входов х1-х8

}

TMR1H = 0x3C; // установка начальных значений Таймера1

TMR1L = 0xB0; // установка начальных значений Таймера1

PIR1.TMR1IF = 0;

}

}

void main()

{

TRISD=0xff; //Input x1-x8

TRISB=0x01;

TRISC=0x03;

TRISA=0x10;

TRISE=0x00;

PORTD=EEPROM\_Read(0);//Читаем из ЕЕПРОМ PORTD

PIND=PORTD; //считывание значения потртов в символьную переменную пинд

y1=~x1;

y2=~x2;

y3=~x3;

y4=~x4;

y5=~x5;

timer1=0;

T2CON = 0x05; //Преддделитель - 4, включение Т2. В приницпе величина предделителя не важна в нашем случае, можно брать любой

T1CON=0x11; //

TMR1H = 0x3C; // установка начальных значений Таймера1

TMR1L = 0xB0; // установка начальных значений Таймера1

PIE1.TMR1IE = 0; //Запрещаем прерывание Таймер1

PIE1.TMR2IE = 1; //Разрешаем прерывание Таймер2

PIE2.EEIE = 1; //Разрешаем прерывание по окончанию записи в ЕЕПРОМ

INTCON=0xC0; // Разрешить глобальные прерывания INTCON - регист управляющий прерываниями

while(1)

{

if (PIND!=PORTD)//проверка, изменились ли входы х1-х8 и стартуем таймер1

{

PIND=PORTD;

timer1=0;

TMR1H = 0x3C; // preset for timer1 MSB register

TMR1L = 0xB0; // preset for timer1 LSB register

PIE1.TMR1IE = 1;// Разрешаем прерывание переполнения таймер2

EEPROM\_Write(0,PORTD);//Пишем в ЕЕПРОМ PORTD

}

y6=(x1|x2|(x3&t2)|(x4&t4));

y7=(x1|x2|(x3&t2)|(x4&t4));

}

}

// PIC16F877A Timer1 interrupt

// Timer1 is used to interrupt every 500ms

// https://simple-circuit.com/

#include <16F877A.h>

#use delay(crystal=4000000)

#INT\_TIMER1

void timer1\_isr(void)

{

clear\_interrupt(INT\_TIMER1);

set\_timer1(3036);

output\_toggle(PIN\_B0);

}

void main()

{

setup\_timer\_1 ( T1\_INTERNAL | T1\_DIV\_BY\_8 ); // Internal clock and prescaler 8

set\_timer1(3036); // Preload value

clear\_interrupt(INT\_TIMER1); // Clear Timer1 interrupt flag bit

enable\_interrupts(INT\_TIMER1); // Enable Timer1 interrupt

enable\_interrupts(GLOBAL); // Enable global interrupts

output\_low(PIN\_B0);

while(TRUE) ; // Endless loop

}