ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
СРЕДНЕГО (ПОЛНОГО) ОБЩЕГО ОБРАЗОВАНИЯ
ЛИЦЕЙ ПРИ СПБГУТ

Вендор-ориентированный учебный курс в системе
«Старшая профильно-профессиональная школа-ВУЗ-Работодатель»:
«Программирование микроконтроллеров Microchip»

Богураев М.В.

«ЯЗЫК СИ ДЛЯ МИКРОКОНТРОЛЛЕРОВ PIC ОБРАБОТКА ПРЕРЫВАНИЙ»

Методические указания к выполнению
лабораторной работы

Санкт - Петербург
2012

Богураев М.В. «ЯЗЫК СИ ДЛЯ МИКРОКОНТРОЛЛЕРОВ PIC ОБРАБОТКА ПРЕРЫВАНИЙ». Методические указания к выполнению лабораторной работы №14(17). СПб: ГОУ «Лицей при СПбГУТ», 2012.

ЛАБОРАТОРНАЯ РАБОТА №14 «ЯЗЫК СИ ДЛЯ МИКРОКОНТРОЛЛЕРОВ PIC ОБРАБОТКА ПРЕРЫВАНИЙ»

Цель работы

Научиться обрабатывать прерывания микроконтроллеров среднего подсемейства PIC на языке СИ с использованием компилятора HI-TECH C. Научиться моделировать прерывания в среде разработки MPLAB IDE.

Теоретические основы

В этой работе рассматривается программирование на языке СИ с использованием прерывания. Прерывание (Interrupt) – это способ обработки событий, при котором как только наступает событие, работа основной программы приостанавливается и выполняется функция – обработчик прерывания, когда функция – обработчик прерывания завершается, выполнение основной программы возобновляется. Механизм прерывания следующий: если происходит событие (установка флага) в аппаратном стеке сохраняется текущее значение счётчика команд (Program Counter – PC). Затем значение PC устанавливается равным 0x04. Этот адрес в памяти команд называется вектором прерывания (Interrupt Vector Address) по этому адресу расположена функция – обработчик прерывания, то есть микроконтроллер переходит к функции – обработчику прерывания. После выполнения этой функции (после обработки прерывания) из стека в PC загружается тот адрес, на котором была прервана программа. После этого продолжается выполнение основной программы с адреса, загруженного в PC.

Источниками прерываний могут быть периферийные модули микроконтроллера или порты ввода/вывода. Чтобы настроить прерывания нужно установить биты разрешения прерываний в регистрах управления прерываниями (INTCON, PIE1 и PIE2). Комбинацию битов прерываний называют маской прерываний. Запрещённые прерывания называют замаскированными. Разрешённые прерывания называют незамаскированными. То есть программист посредством маски указывает, какое событие будет обработано как прерывание.

Биты разрешения прерывания имеют название вида ???E (Enable); например, бит T0IE разрешает прерывание от нулевого таймера (Timer 0 Interrupt Enable). Функция бита GIE (Global Interrupt Enable) в регистре INTCON заключается в разрешении или запрете абсолютно всех прерываний. В момент входа в прерывание микроконтроллер сбрасывает этот бит. Оператор return разрешает следующее прерывание, так как устанавливает бит GIE.

Источник прерывания определяют по флагу – специальному биту, который устанавливается после наступления события. Каждому источнику прерывания соответствует свой флаг. Флаги имеют название вида ???F (Flag);. Флаги устанавливаются независимо от того, разрешены прерывания или нет. Флаги прерываний сбрасывает программист в функции – обработчике прерывания, чтобы не было повторного вхождения в прерывание по ранее установленному флагу – так как прерывание по этому флагу будет обработано. Например, бит T0IF (Timer 0 Interrupt Flag) устанавливается (принимает значение «1») если произошло событие – переполнение нулевого таймера. Если T0IE и T0IF установлены, то в результате переполнения нулевого таймера возникнет прерывание.

Обработчик прерывания – это функция (подпрограмма), которая выполняется по прерыванию, и заканчивается выходом из прерывания. Обработчик прерывания располагают по адресу 0x04, или по 0x04 располагают команду перехода на подпрограмму обработчика прерывания.

Для написания работающей программы необходимо подключить файл pic.h, задать слово конфигурации директивой __CONFIG, написать функцию main. Внутри функции

main нужно инициализировать микроконтроллер – настроить регистры TRIS, разрешить прерывания, настроить маску прерываний, настроить периферийные модули.

Затем пишут текст основной программы и текст обработчика прерывания. Так как микроконтроллер программируют на языке СИ, то компилятор сам размещает функцию обработчика прерывания по адресу вектора прерывания. Компилятор сам делает контекстное сохранение регистров W, STATUS и PCLATH. Внимание – флаги прерываний, в функции – обработчике прерывания, должен сбрасывать программист.

Функция обработчика прерывания в общем виде выглядит следующим образом:

```
void interrupt <Имя функции> (void)
{
        if (XXIE && XXIF) // определить источник прерывания
        {
        XXIF=0;// сбросить флаг источника прерывания
        <оператор1>; // тело функции
        <оператор2>;
                …
        <оператор n>;
        return;
        }
        else if (YYIF && YYIE)
        {
        YYIF = 0;
        <оператор1>; // тело функции
        <оператор2>;
                …
        <оператор k>;
        return;
        }
        else if …
}
```

Чтобы компилятор распознал функцию прерывания, она должна иметь тип void interrupt и не должна иметь параметров. Поскольку в микроконтроллерах среднего подсемейства только один вектор прерывания, то и функция прерывания может существовать только одна. Для неё компилятор СИ располагает точку входа по адресу 0x04 (вектор прерываний). Аппаратное прерывание вызывает функцию void interrupt. Из функции прерывания могут вызываться другие функции. Внутри этой функции по флагам определяют источники прерываний и выполняют соответствующие действия для каждого из источников.

Сохранение нужных регистров на время прерывания обеспечивается компилятором автоматически. Если функция прерывания вызывает другие функции, и эти функции описаны перед кодом прерывания в этом же программном модуле, тогда все переменные, используемые этими функциями, тоже будут сохранены. Все сохранённые переменные автоматически восстанавливаются перед выходом из функции обработки прерывания. Следует помнить об опасности переполнения стека. Нельзя заполнять стек более чем на восемь уровней.

Задание

На диске C в папке C_Projects создайте папку C_Project6. В эту папку скопируйте файл C_Project6.c. Откомпилируйте проект. Промоделируйте работу микроконтроллера в среде MPLAB IDE. Запрограммируйте лабораторный макет и предъявите результат.

Порядок выполнения

На диске C:\ в папке C_Projects создайте папку C_Project6. В папку C_Project6 скопируйте файл C_Project6.c. Запустите MPLAB IDE и создайте проект на языке СИ в этой папке. Откомпилируйте программу. Выберите симулятор MPLAB SIM (рис. 1). В окно Watch выведите регистры COUNTER, TMR0, INTCON, PORTB, PORTD (рис.2).

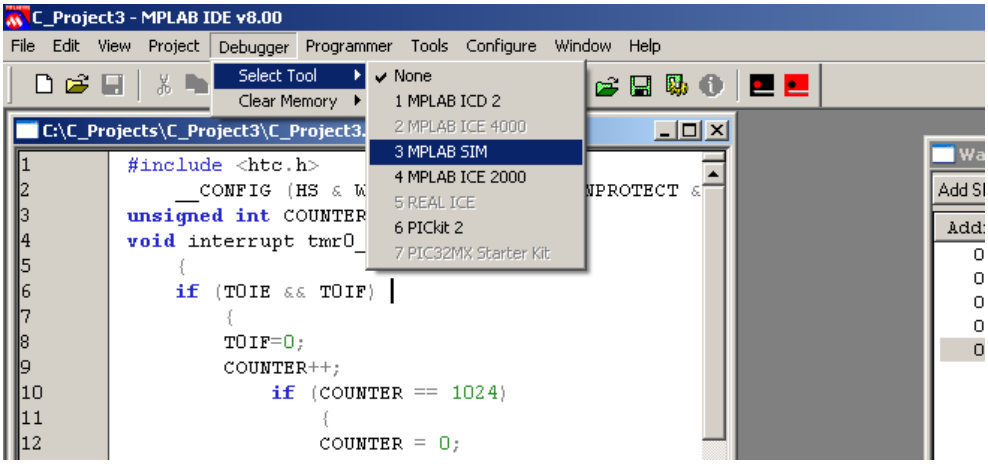

Рис. 1. Выбор симулятора MPLAB SIM.

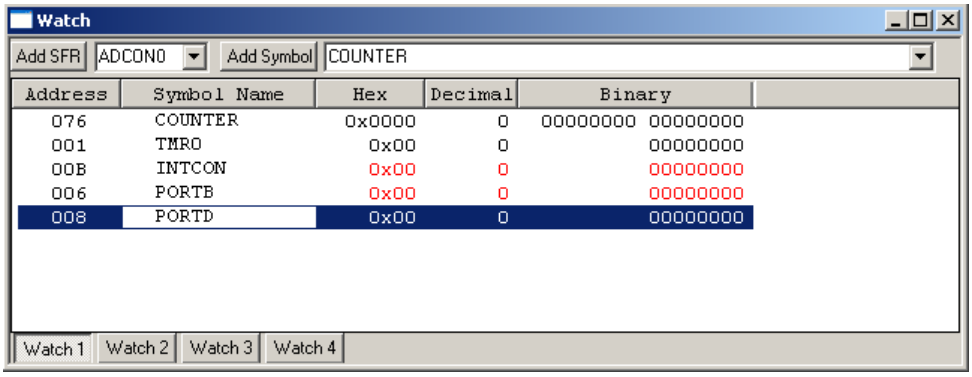

Рис. 2. Вид настроенного окна Watch.

Создайте файл стимулов для RB0. Для этого выберите Debugger/Stimulus/New Workbook (рис. 3).

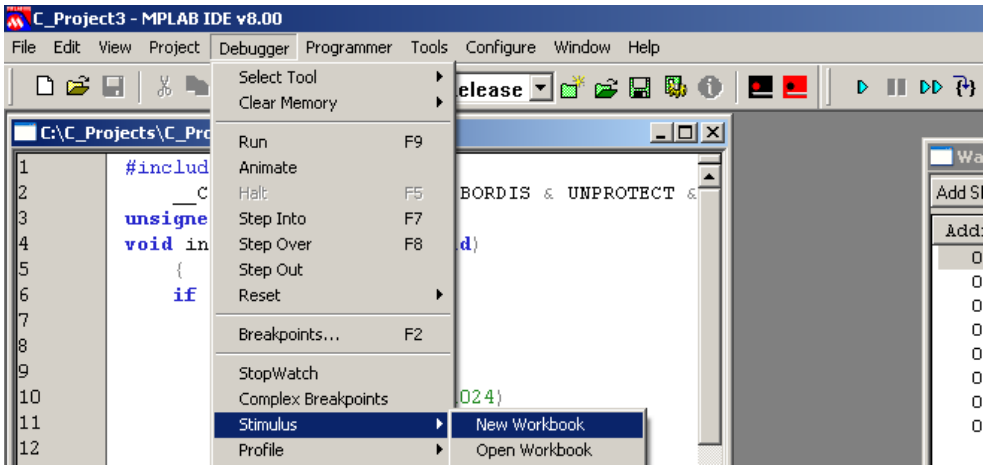

Рис. 3. Создание файла стимулов.

В открывшемся окне выберите вкладку Asynch, и в столбце Pin/SFR выберите RB0 (рис. 4).

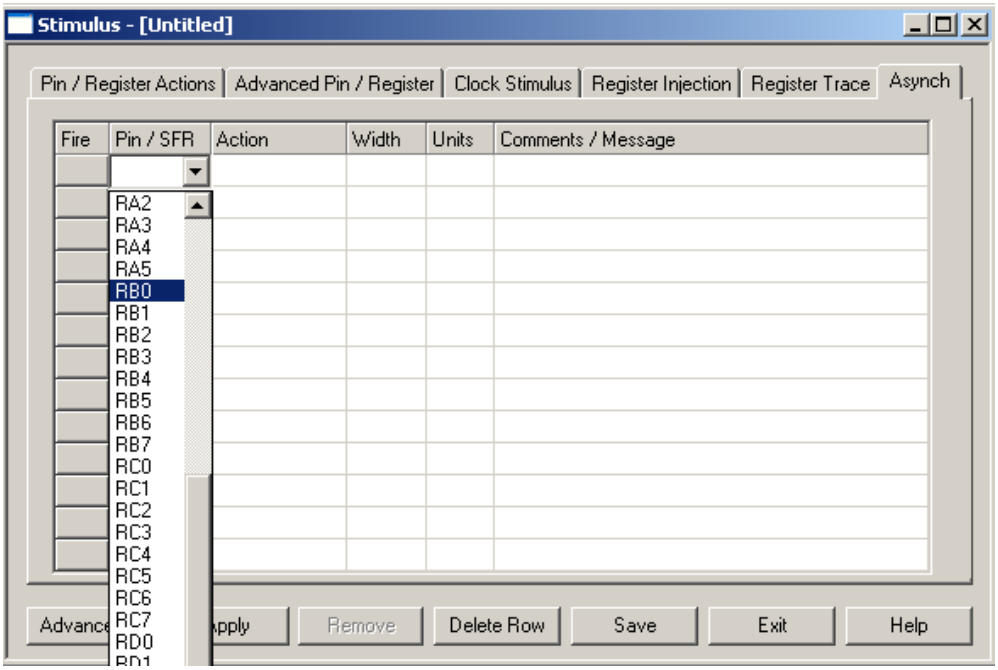

Рис. 4. Выбор вывода RB0.

В столбце Action выберите Toggle – переключение (рис. 5).

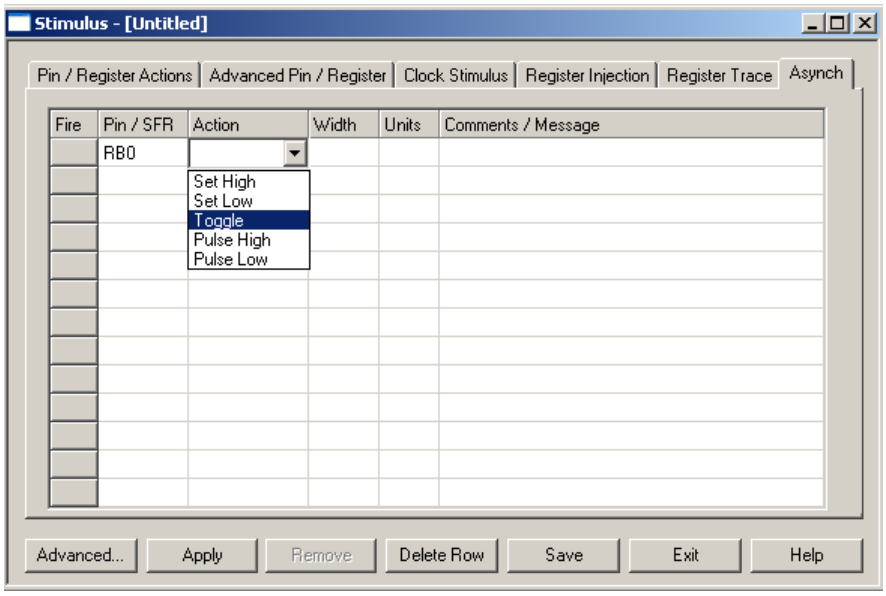

Рис. 5. Выбор воздействия Toggle.

Теперь, нажимая на «>» в графе Fire, вы будете включать или выключать единицу на выводе RB0 (симуляция нажатия кнопки).

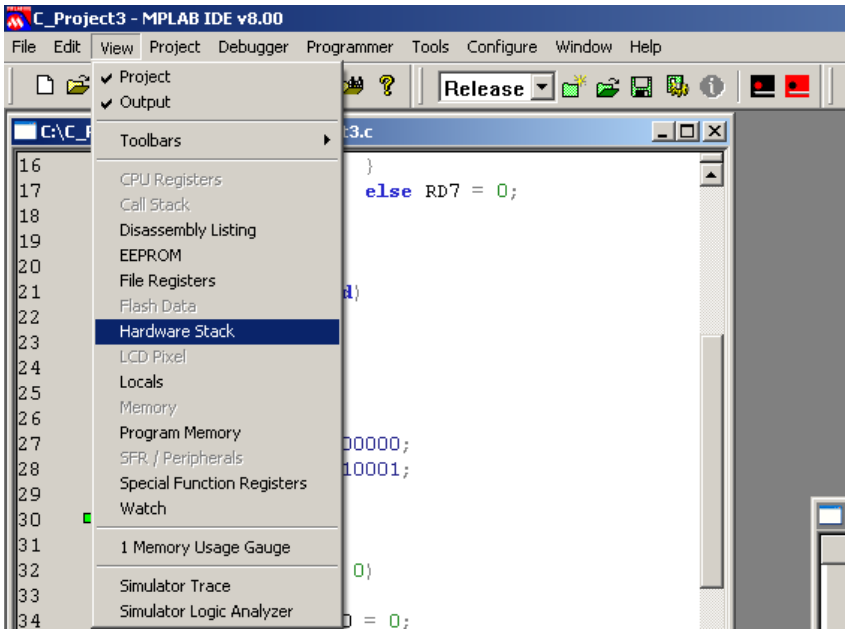

Рис. 6. Выбор окна просмотра аппаратного стека.

Выберите View/Hardware Stack (рис. 6), появившееся окно содержит восемь уровней аппаратного стека (рис. 7).

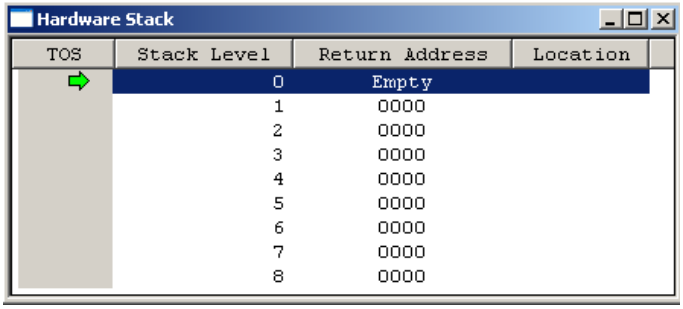

Рис. 7. Окно симулятора, в котором отображается содержимое стека.

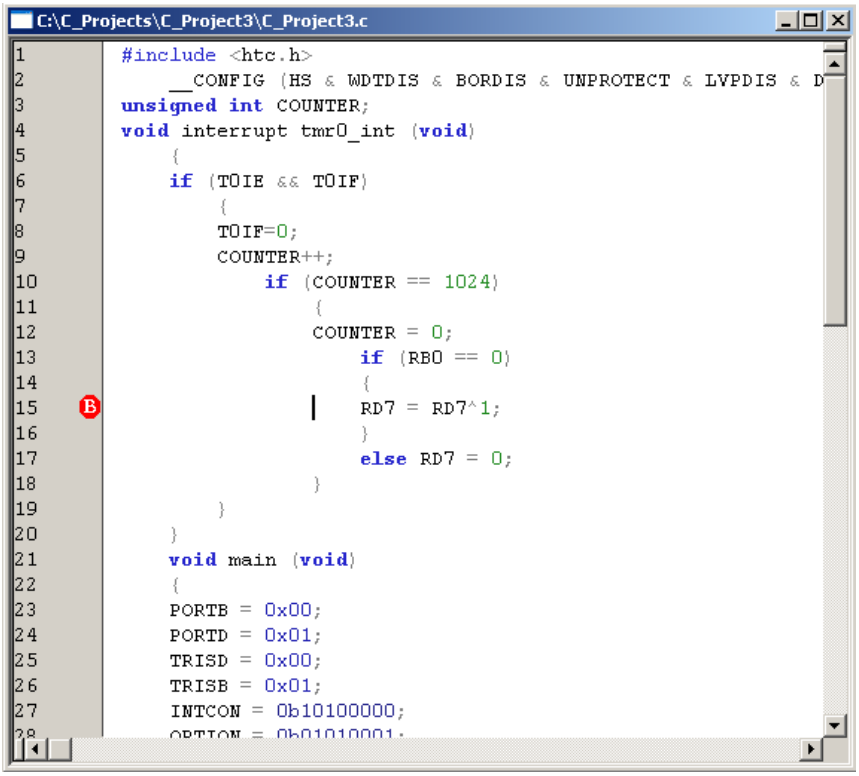

Рис. 8. Установка точки останова.

На строке RD7 = RD7^1 поставьте точку останова для чего дважды щёлкните левой кнопкой мыши (рис. 8).

Запустите симулятор и промоделируйте работу контроллера. После каждой остановки обращайте внимание на содержимое стека, затем запускайте программу заново и нажимайте «>» в графе Fire. Следите за изменениями в окне Watch. Объясните изменения регистров TMR0, COUNTER, PORTD, PORTB. Поясните изменения содержимого стека.

После удачной симуляции соберите на макете схему (рис. 9) как показано на рис. 10. Запрограммируйте лабораторный макет и предъявите результат преподавателю.

Аппаратное обеспечение

Эта работа выполняется и на компьютере в среде разработки MPLAB IDE, и на макете. Принципиальная схема изображена на рис. 9. Схема, собранная на макете, изображена на рис. 10.

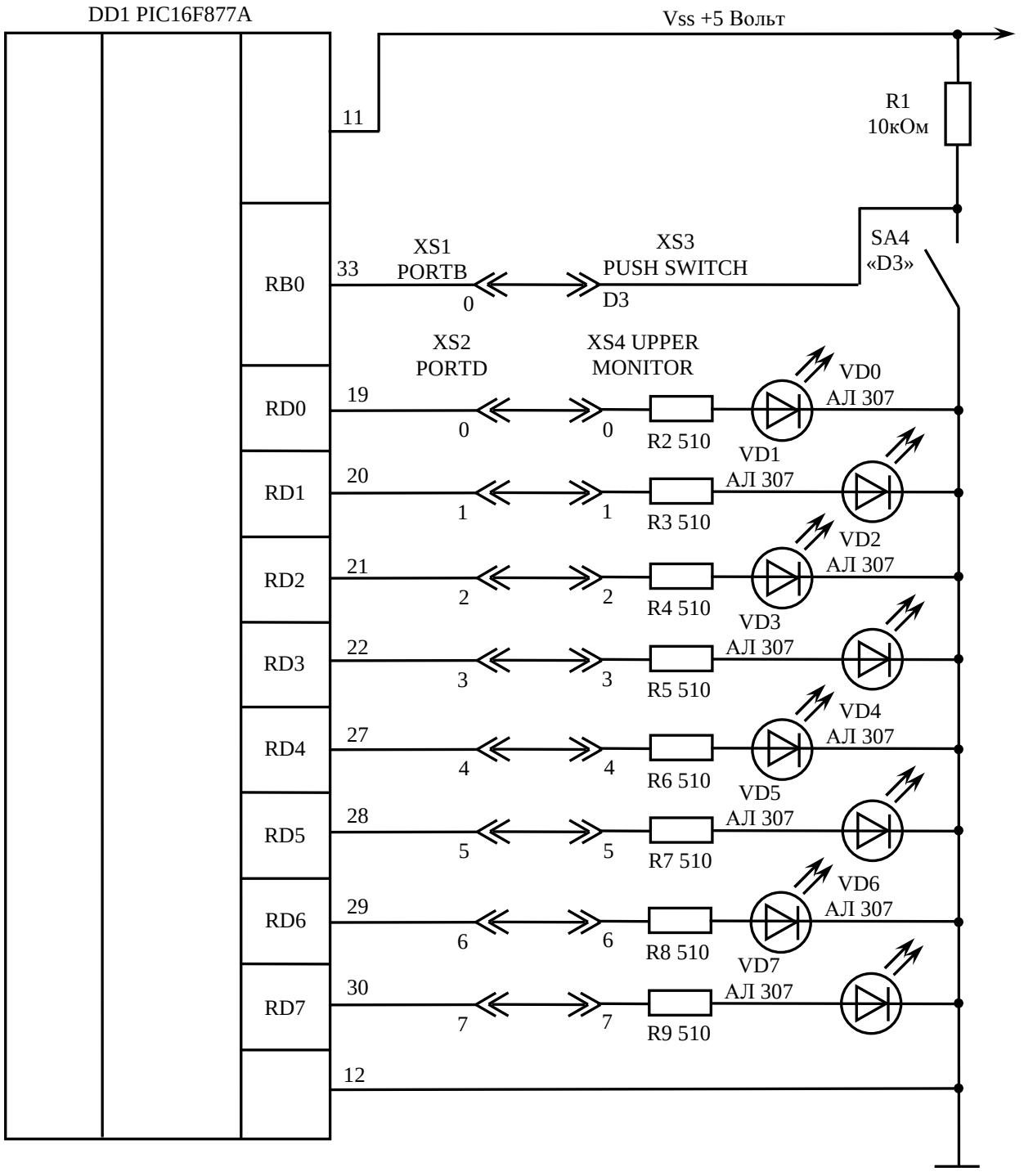

Рис. 9. Схема электрическая принципиальная.

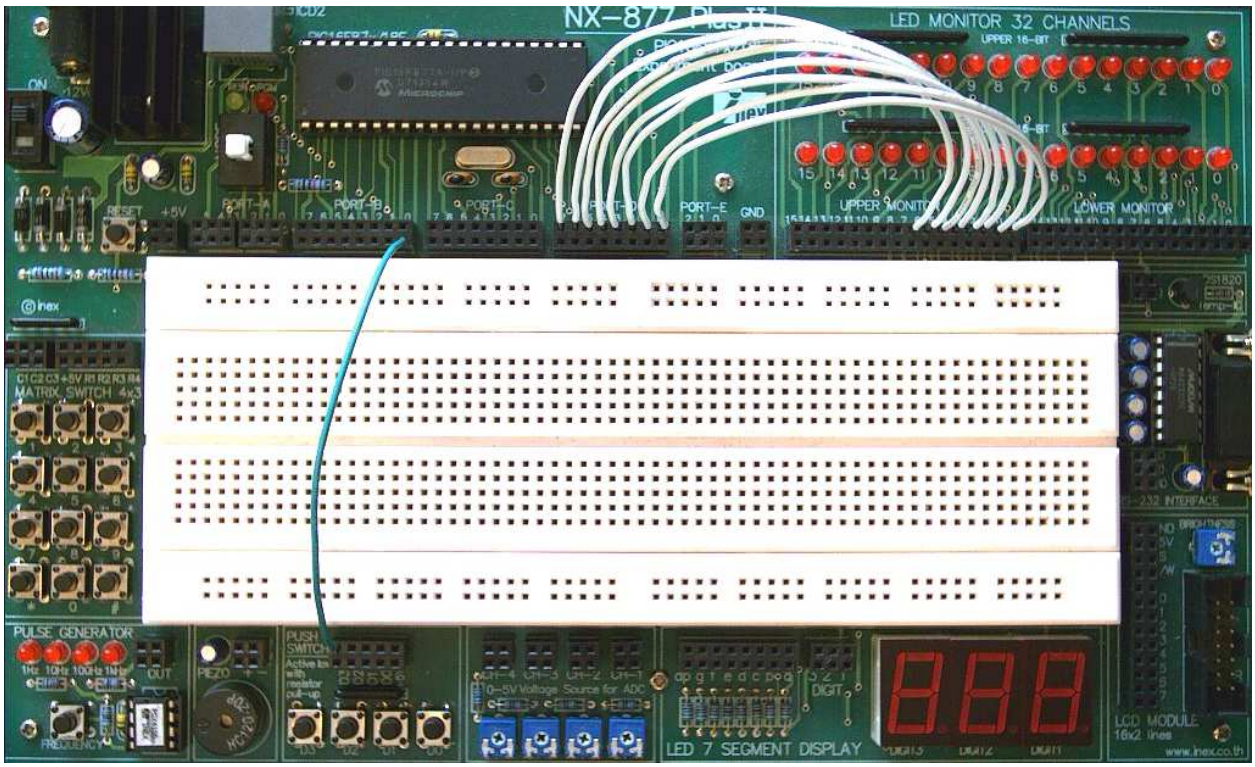

Рис. 10. Схема, собранная на макете.

Программное обеспечение

Текст файла C_Projec3.c

```
#include <htc.h>
__CONFIG (HS & WDTDIS & BORDIS & UNPROTECT & LVPDIS & DEBUGEN);
unsigned int COUNTER;
void interrupt tmr0_int (void)
                        {
                        if (T0IE && T0IF)
                        {
                                T0IF=0;
                                COUNTER++;
                                if (COUNTER == 1024)
                                {
                                        COUNTER = 0;
                                        if (RB0 == 0)
                                        {
                                        RD7 = RD7^1;
                                        }
                                else RD7 = 0;
                                }
                                }
                        }
                        void main (void)
                        {
                        PORTB = 0x00;
                        PORTD = 0x01;
                        TRISD = 0x00;
                        TRISB = 0x01;
                        INTCON = 0b10100000;
                        OPTION = 0b01010001;
                        COUNTER = 0;
                                while (1)
                                {
                                if (RB0 == 0)
                                        {
                                        RD0 = 0;
                                        }
                                else
                                        {
                                        RD0 = 1;
                                        }
                                }
```

Индивидуальные задания

Заставьте моргать светодиод на RB6 таким образом: если кнопка отпущена, он моргает, если кнопка нажата, моргает светодиод RB7.

Попробуйте заставить моргать разные комбинации диодов по нажатой и отпущенной кнопке.

Контрольные вопросы

1. Что такое прерывания?
2. Объясните механизм прерывания.
3. Что такое вектор прерывания?
4. Зачем нужен стек?
5. Что может быть источником прерываний?
6. Что такое маска прерывания?
7. Какова функция регистров INTCON, PIR1, PIR2?
8. Как определить источник прерывания?
9. Что такое флаг прерывания?
10. Что такое обработчик прерывания?
11. Должен ли программист сбросить флаг прерывания в обработчике прерывания?
12. Почему регистр COUNTER видится в окне Watch как два байта?

Оглавление:

Список рисунков: