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 ПЕРИФЕРИЙНЫЙ МОДУЛЬ ПЕРВЫЙ ТАЙМЕР»

Методические указания к выполнению лабораторной работы

Санкт - Петербург
2012

Богураев М.В. «ЯЗЫК СИ ДЛЯ МИКРОКОНТРОЛЛЕРОВ PIC ПЕРИФЕРЙИНЫЙ МОДУЛЬ ПЕРВЫЙ ТАЙМЕР». Методические указания к выполнению лабораторной работы № 5. СПб: ГОУ «Лицей при СПбГУТ», 2012.

ЛАБОРАТОРНАЯ РАБОТА №13
«ЯЗЫК СИ ДЛЯ МИКРОКОНТРОЛЛЕРОВ PIC
ПЕРИФЕРИЙНЫЙ МОДУЛЬ ПЕРВЫЙ ТАЙМЕР»

Цель работы

Научиться включать периферийные модули. Ознакомиться с работой периферийного модуля первый таймер. Освоить моделирование этого модуля в MPLAB IDE. Овладеть навыками программирования таймеров PIC микроконтроллеров.

Теоретические основы

Периферийные модули – это устройства, которые располагаются на одном кристалле с ядром микроконтроллера. Каждый периферийный модуль выполняет одну или несколько функций. Настройка и подключение этих модулей производятся программно. Периферийные модули могут быть подключены к ядру микроконтроллера и к ножкам корпуса микроконтроллера. Включают и настраивают периферийные модули программно.

Счётчики и таймеры один из видов периферийных модулей. Счётчик – это периферийный модуль, на вход которого подают напряжения логических уровней. При каждом спаде или нарастании напряжения содержимое счётчика меняется на единицу. Принципиальной разницы между таймером и счётчиком нет. Таймер – это частный случай счётчика. Если сделать так, что нарастания (спады) напряжения будут происходить с равным периодом, то счётчик станет таймером. Таймеры – это счётчики, при помощи которых измеряют время.

О включении и режимах периферийного модуля TIMER1 можно прочитать в документации на микроконтроллер (Data Sheet – доступен на сайте производителя: файл 39582b.pdf). Периферийный модуль TIMER1 шестнадцатиразрядный, отображается в память данных как две ячейки памяти – TMR1H и TMR1L. Это позволяет работать в более широком диапазоне чисел: от 0x0000 до 0xFFFF. То есть можно отсчитывать более длительные интервалы времени чем, те которые можно отсчитать восьмиразрядным таймером. Модуль TIMER1 можно настроить или как счётчик или как таймер.

Первый таймер может работать с импульсами, которые поступают на вход RC0/T1OSO/T1CKI. В модуле первого таймера имеется цепь тактового генератора. Эта цепь рассчитана на подключение кварцевого резонатора с частотой резонанса 32768 Герц («часовой кварц»). Резонатор подключается между выводами T1OSI и T1OSO (RC1 и RC0, соответственно) и снабжается двумя конденсаторами. В модуле имеется предделитель. Предделитель позволяет увеличивать период (делить «частоту») импульсов, которые изменяют содержимое регистров TMR1H и TMR1L.

Счёт в таймере происходит следующим образом: при каждом изменении (нарастании) напряжения на входе первого таймера содержимое ячейки памяти данных TMR1L увеличивается на единицу. Когда содержимое ячейки становится равным 0xFF и приходит очередной импульс – значение в регистре TMR1L обнуляется (TMRlL=0x00), а в регистре TMR1H устанавливается единица. Если счёт продолжается, то при следующем обнулении регистра TMR1L содержимое регистра TMR1H снова увеличивается на единицу. Переполнение первого таймера происходит в тот момент, когда в TMR1H и TMRlL устанавливается значение 0xFFFF и приходит очередной импульс. Тогда обе ячейки обнуляются и устанавливается флаг (бит) TMR1IF.

Чтобы пользоваться первым таймером его нужно настроить и включить. Затем в программе отслеживать флаг (бит) TMR1IF. Если первый таймер настроен и включен, то отслеживать бит TMR1IF можно с помощью прерываний. Прерывания – это способ обработки событий. В нашем случае событием является установка флага. Если при таком способе обработки происходит событие, то: выполнение основной программы

прерывается и выполняется функция – обработчик прерывания. Когда функция – обработчик прерываний завершается, работа основной программы продолжается с того места, в котором основная программа была прервана.

В общем случае в тексте программы функция – обработчик прерывания описывается 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 …
}
```

В такой функции interrupt – ключевое слово, по которому компилятор определяет, что это функция – обработчик прерывания. Слово name заменяют своим названием функции. Биты XIF и YIF – флаги прерывания от различных периферийных модулей. Биты XIE и YIE – биты разрешения прерывания от соответствующих периферийных модулей. Флаги прерываний должны сбрасываться программистом в программе обработки прерывания. Это необходимо для исключения повторного вхождения в прерывание по ранее установленному флагу – так как прерывание по этому флагу уже обработано. Прерывания от периферийных модулей разрешены, если в регистре INTCON биты GIE и PEIE единицы. Бит GIE разрешает все прерывания, бит PEIE разрешает прерывания от периферийных модулей.

Настройка модуля TIMER1 осуществляется в регистре T1CON. Коэффициент деления предделителя устанавливается битами <T1CKPS1:T1CKPS0>. Если бит T1OSCEN установлен, тактовый генератор модуля TIMER1 включается и начинает генерировать импульсы на входе RC0/T1OSO/T1CKI. После включения генератора, программист должен сделать программную задержку (паузу) для запуска генератора. Бит #T1SYNC устанавливают, если не предполагается синхронизация генератора таймера и генератора микроконтроллера. Бит TMR1CS устанавливают, если предполагается использование внешнего источника тактовых сигналов со входа RC0/T1OSO/T1CKI.

Если бит TMR1ON установлен, содержимое регистра TMR1L будет увеличиваться на один на каждом импульсе, который придёт на вход RC0/T10SO/T1CKI (если предделитель установлен в положение 1:1, бит TMR1CS=1 и генератор таймера работает). То есть таймер начинает счёт сразу же после установки бита TMR1ON.

Прерывания от периферийного модуля первый таймер настраивают в регистрах INTCON – биты GIE и PEIE и в регистре PIE1 – бит TMR1IE. Флаг (бит) прерывания

TMR1IF находится в регистре PIR1 и устанавливается при переполнении первого таймера. Этот бит программист должен сбросить сам, иначе микроконтроллер будет постоянно входить в прерывание.

В этой лабораторной работе TIMER1 конфигурируется следующим образом. В регистре T1CON (регистр настроек первого таймера):

Биты <T1CKPS1:T1CKPS0> нули. То есть коэффициент деления предделителя 1:1.

Бит T1OSCEN единица. То есть тактовый генератор модуля TIMER1 включен и генерирует импульсы на входе RC0/T1OSO/T1CKI.

Бит #T1SYNC единица. Генератор таймера и генератор микроконтроллера не синхронизируются.

Бит TMR1CS единица. То есть: источник тактовых импульсов внешний – генератор модуля TIMER1 (External clock). Импульсы генератора первого таймера поступают на вход RC0/T1OSO/T1CKI.

Бит TMR1ON единица – таймер считает. Содержимое регистра TMR1L увеличивается на один на каждом импульсе, который приходит на RC0/T10SO/T1CKI (предделитель установлен в положение 1:1; бит TMR1CS=1 – генератор таймера запущен и работает).

В этой лабораторной работе прерывания от периферийного модуля первый таймер разрешены. В регистре INTCON биты PEIE и GIE единицы; в регистре PIE1 бит TMR1IE единица. На каждом прерывании происходит следующее: проверяется источник прерывания (TMR1IE && TMR1IF). Если прерывание пришло от первого таймера, тогда нужно сбросить флаг TMR1IF, затем настроить таймер на секунду и увеличить на единицу содержимое переменной ClockSeconds. Говорят, что содержимое переменной ClockSeconds увеличивается на единицу раз в секунду по прерыванию от первого таймера. Чтобы первый таймер отсчитывал секундные интервалы нужно после каждого сброса флага TMR1IF записывать в регистр TMR1H число 0x80 – заполнять таймер наполовину, иначе прерывания будут происходить раз в две секунды. Это так, потому что максимальное число, которое может разместиться в двух регистрах первого таймера – это 0xFFFF. Оно же десятеричное 65535 – или 2 раза по 32768. А 32768 Герц – это частота кварцевого резонатора, который подключен к генератору первого таймера. Значит, за одну секунду будет совершено 32768 приращений, а потом будет выставлен флаг прерывания. Это как раз будет секундой, это требуется в лабораторной работе.

Три семисегментных индикатора будут переключаться по нулевому таймеру (катоды подключены к ножкам RC2, RC3, RC4). После каждого сброса флага TMR0IF катод одного из семисегментных индикаторов соединяется с общим проводом источника напряжения. Когда подключается катод RC2 индикатора DIGIT1, на индикатор (аноды всех трёх индикаторов подключены к PORTD) выводится содержимое ячейки DispSeconds; когда подключается катод RC3 индикатора DIGIT2, на него выводится содержимое ячейки DispDesSeconds; когда подключается катод RC4 индикатора DIGIT3, на него (в PORTD) выводится содержимое ячейки DispMinutes. Внутри функции Translate() происходит преобразование числа секунд в коды семисегментных индикаторов.

То есть отсчёт секунд происходит по первому таймеру, а динамическая индикация осуществляется по нулевому таймеру.

Задание

Изучите теоретические основы этой лабораторной работы (дополнительно можно изучить файл 39582b.pdf, раздел TIMER1). Заполните таблицу включения и настройки первого таймера. На диске C в папке C_Projects создайте папку C_Project5. В эту папку скопируйте файл C_Project5.c. Затем создайте проект C_Project5. Откомпилируйте текст программы C_Project5.c. В симуляторе MPLAB SIM промоделируйте работу устройства. Запустите программу в симуляторе MPLAB IDE и промоделируйте работу программы в окне Watch, а каналы T1CKI, RC2, RC3, RC4 и PORTD в Logic Analyzer.

На лабораторном макете подключите три семисегментных индикатора к разъёму PORTD микроконтроллера. Выводы DIGIT1, DIGIT2, DIGIT3 подключите к разъёму PORTC к битам 2, 3, 4 соответственно. Запрограммируйте микроконтроллер и продемонстрируйте работу программы на лабораторном макете.

Порядок выполнения

На диске C в папке C_Projects создайте папку C_Project5. В эту папку скопируйте файл C_Project5.c. Затем создайте проект C_Project5. Откомпилируйте текст программы C_Project5.c.

Заполните таблицу включения и настройки первого таймера.

Табл. 1. Включение и настройка первого таймера.

Регистр	Название, номер бита, значение бита.								После POR	После сброса
	7	6	5	4	3	2	1	0		
INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
			0	0	0	0	0	0		
T1CON	-	-	T1CKPS1	T1CKPS0	T1OSCEN	#T1SYNC	TMR1CS	TMR1ON	--00 0000	--uu uuuu
	-	-	0	0						
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
	0	0	0	0	0	0	0			
RIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
	0	0	0	0	0	0	0			
TMR1L	Младший байт регистра модуля Timer1								xxxx xxxx	uuuu uuuu
TMR1H	Старший байт регистра модуля Timer1								xxxx xxxx	uuuu uuuu

Примечание: POR – power on reset после включения, x = неизвестно, u = не изменяется.

Выберите в качестве отладчика симулятор MPLAB SIM. (рис. 1). Затем из меню View откройте Simulator Logic Analyzer (рис. 2). В появившемся окне (рис. 3) нажмите кнопку Channels. Появится окно Configure Channels. Для отображения каналов T1CKI, RC2, RC3, RC4 нужно выделить добавляемый канал и нажать кнопку Add (рис. 4). Выберите каналы T1CKI, RC2, RC3, RC4 (Рис. 5). Теперь, для отображения содержимого регистра PORTD сформируем шину. Для этого в окне Configure Channels нажмем кнопку Configure Bus(s) и в появившемся окне Configure Bus (рис. 6) нажмём кнопку New Bus. Появится окно Bus Name. Введём туда название шины PORTD (рис. 6) и нажмём кнопку OK. В окне Configure Bus выделим RD0 – RD7 зажав клавишу Shift, и нажмём кнопку Add. После этого шина будет сформирована (рис. 8), нажмём кнопку OK. Добавим шину: в окне Configure Channels выбираем PORTD и нажимаем кнопку Add (рис. 9) и затем нажимаем кнопку OK.

Теперь в окне Logic Analyzer будет отображаться состояние выводов T1CKI, RC2, RC3, RC4 и PORTD (Рис. 10).

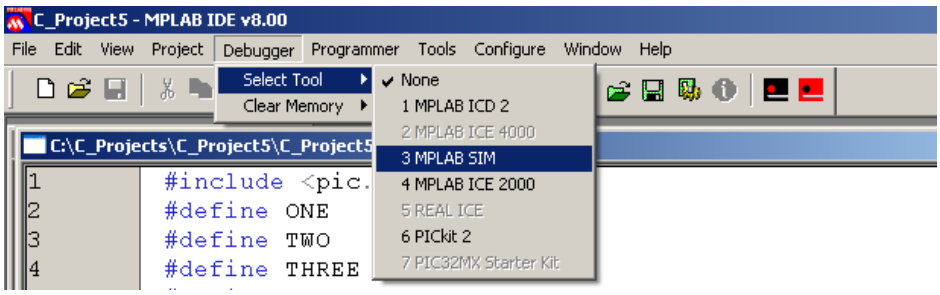

Рис. 1. Выбор симулятора MPLAB SIM.

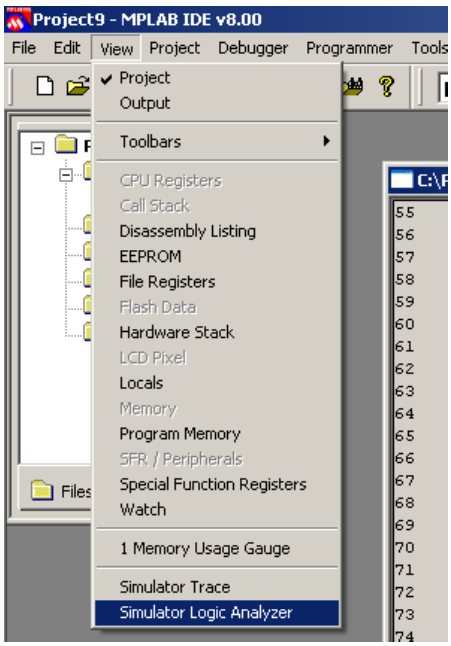

Рис. 2. Запуск логического анализатора.

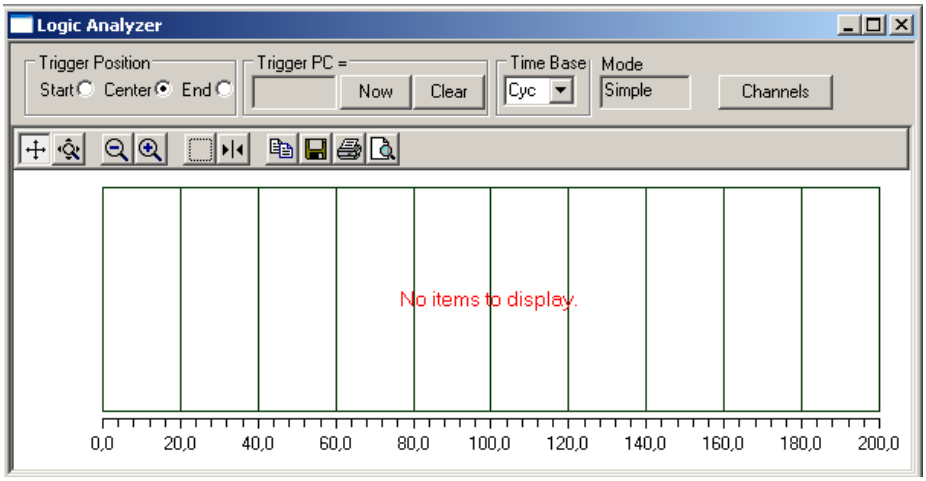

Рис. 3. Вид окна логического анализатора.

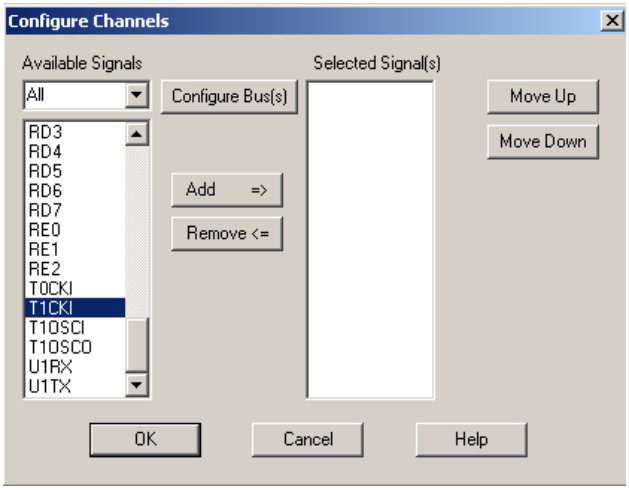

Рис. 4. Выбор каналов.

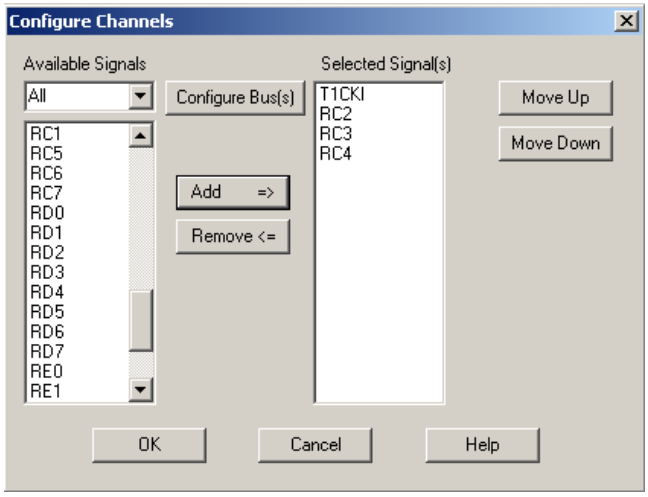

Рис. 5. Вид окна Configure Channels после выбора T1CKI, RC2, RC3 и RC4.

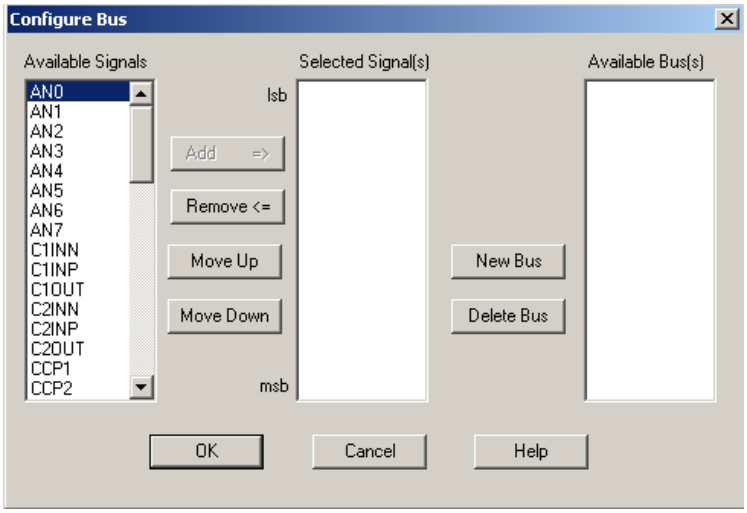

Рис. 6. Окно Configure Bus.

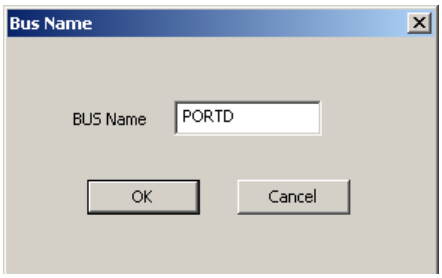

Рис. 7. Окна Bus Name для ввода названия шины.

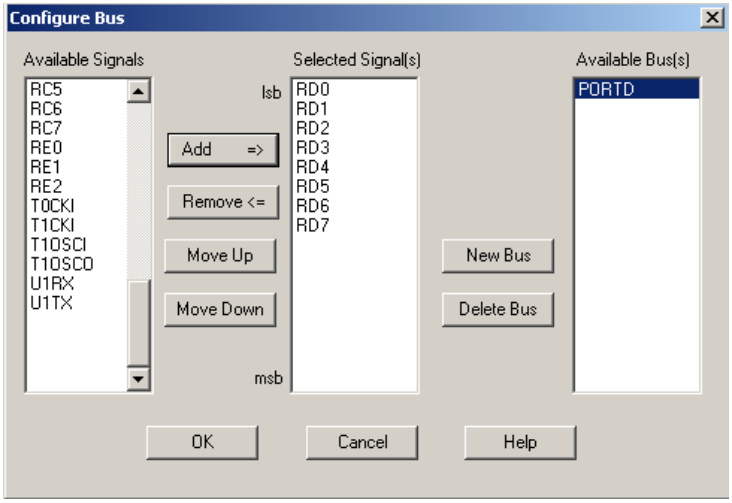

Рис. 8. Формирование шины PORTD.

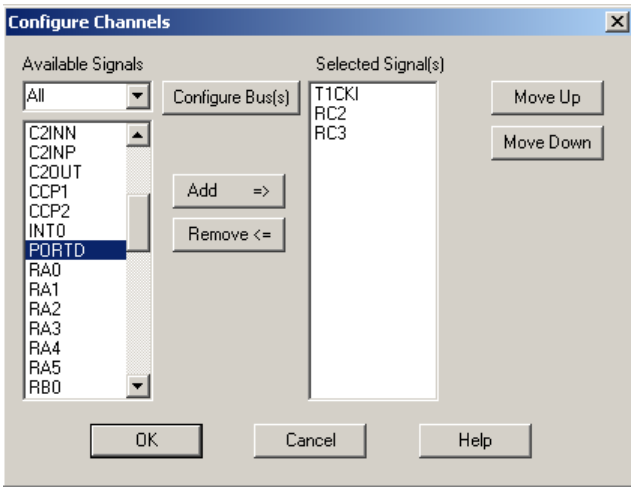

Рис. 9. Добавление сформированной шины PORTD.

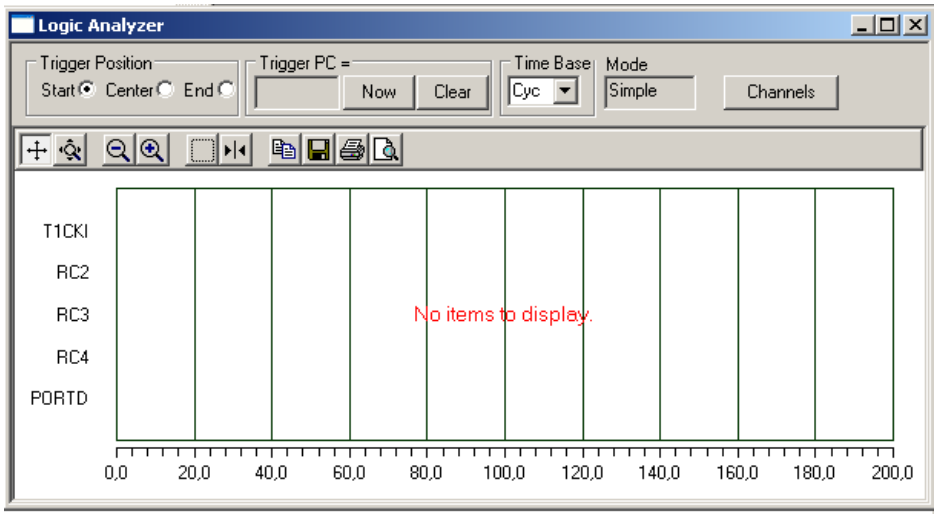

Рис. 10. Вид окна Logic Analyzer после настройки T1CKI, RC2, RC3, RC4 и PORTD.

Для симуляции работы модуля TIMER1 создадим файл стимулов. Для этого в меню Debugger в разделе Stimulus выберем графу New Workbook (рис. 11). В появившемся окне стимулов выберем вкладку Clock Stimulus. В графе PIN выберем T1CKI (рис. 12). В графе Initial выберем значение Low – низкий уровень. В графах Low Cyc и High Cyc впишем единицы, в графе Begin выберем At Start, в графе End выберем Never (рис. 13). Эти установки не имеют абсолютно никакого отношения к реальному таймеру и выбраны исключительно исходя из соображений удобства симуляции, иначе симуляция займёт чрезмерно много времени. Рассмотрим значение этих установок: Initial – это значение, которое будет на ножке T1CKI в начале симуляции. Low Cyc – длительность низкого уровня в машинных циклах. High Cyc – длительность высокого уровня в машинных циклах. Begin – начало применения стимула. End – окончание применения стимула.

После того, как файл стимулов настроен, необходимо нажать кнопку Apply для применения текущих установок. В окне Output должно появиться сообщение об успешном применении стимулов (рис.14). Если стимулы не применить, то симуляция не будет происходить.

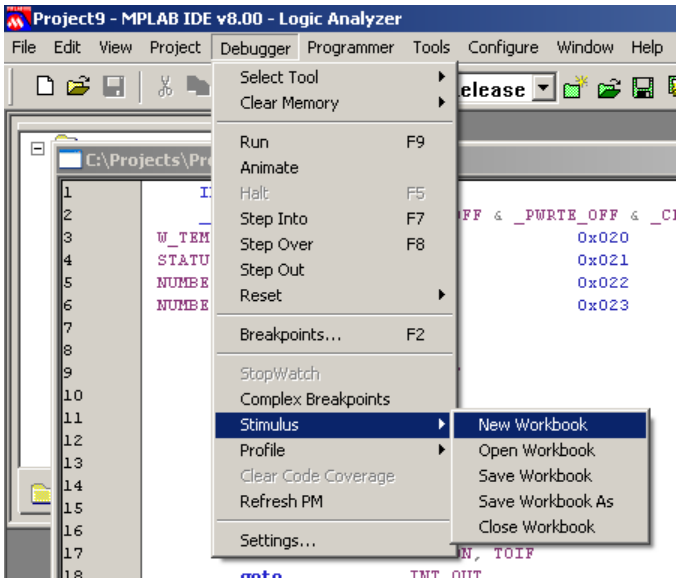

Рис. 11. Создание нового файла стимулов.

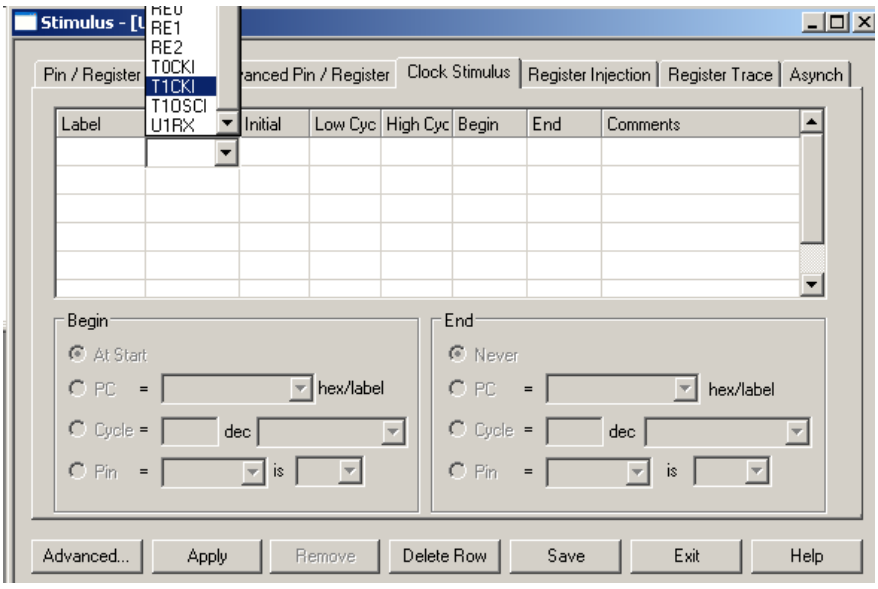

Рис. 12. Выбор симуляции напряжения на ножке T1CKI.

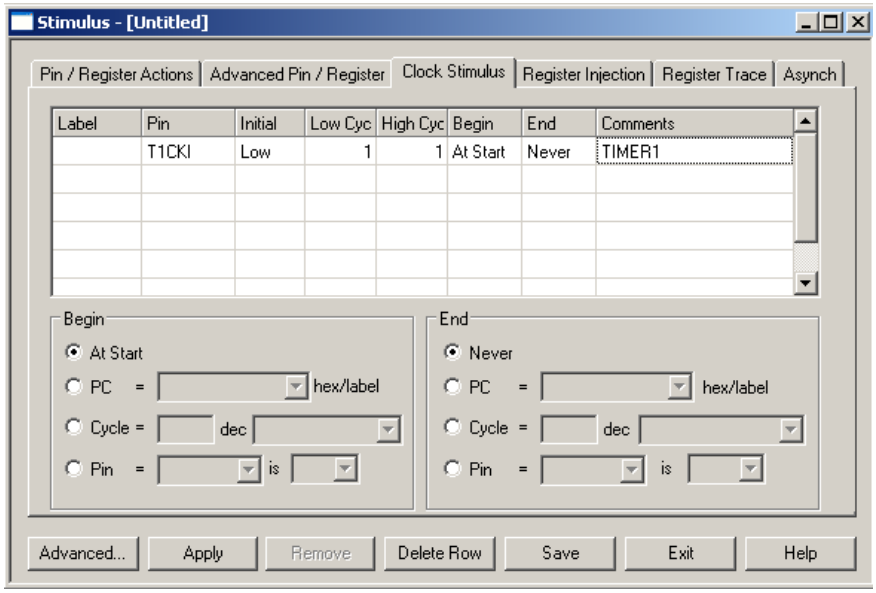

Рис. 13. Вид настроенного окна стимулов.

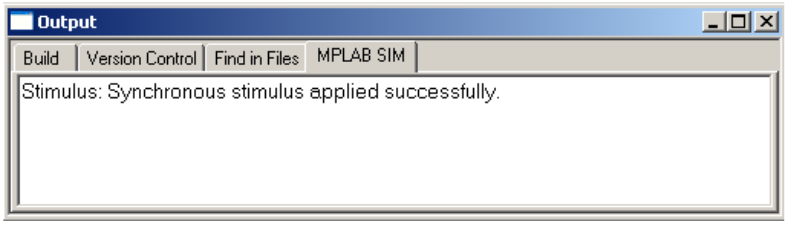

Рис. 14. Сообщение о применении синхронного стимула.

Чтобы симуляция происходила в удобном для наблюдения темпе (Debugger/Settings), выберем время одного шага в Animate режиме 1 миллисекунду, и поставим галочку напротив Enable Realtime watch updates (Рис. 15). Настроим окно Watch (View/Watch) для отображения регистров PORTC, PORTD и двух переменных ClockSeconds и ClockDesSeconds (рис. 16). Кнопкой Animate запустим симулятор (рис. 17). После начала симуляции Logic Analyzer будет отображать состояние выводов T1CKI, RC2, RC3, RC4 и PORTD (Рис. 18). Изменения регистров PORTC, PORTD и переменных ClockSeconds и ClockDesSeconds будут видны и в окне Watch.

После того, как будет получено несколько периодов сигнала в окне Logic Analyzer, остановите симулятор кнопкой Halt (рис. 19).

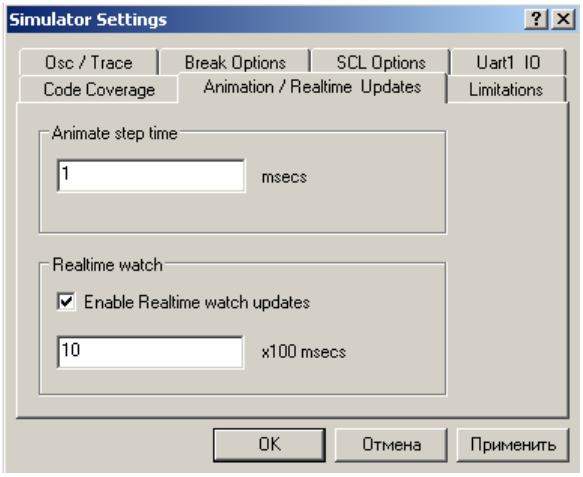

Рис. 15. Выбор времени одного шага в Animate режиме.

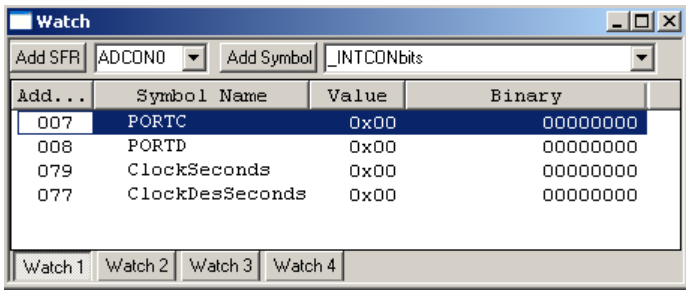

Рис. 16. Вид настроенного окна Watch.

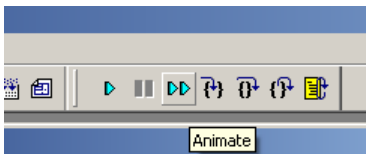

Рис. 17. Кнопка Animate.

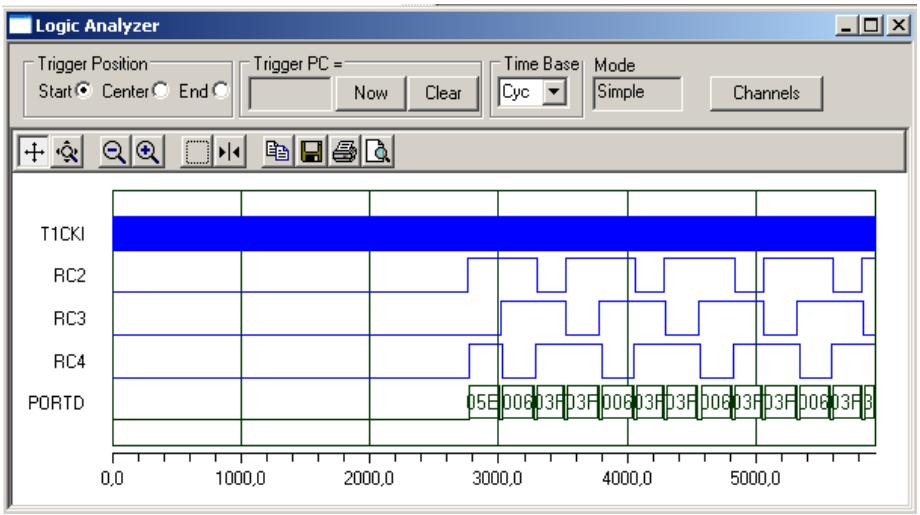

Рис. 18. Вид окна Logic Analyzer в процессе симуляции.

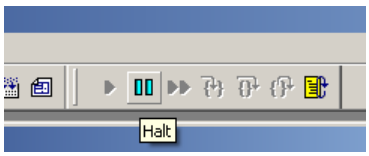

Рис. 19. Кнопка Halt для остановки симулятора.

Проанализируйте полученный результат: постарайтесь понять, что и когда происходит с пользовательскими регистрами NUMBER_ONE и NUMBER_TWO, какое число и когда будет подано на PORTD; какие при этом уровни будут на ножках RC2 и RC3. Предскажите результат работы программы на макете.

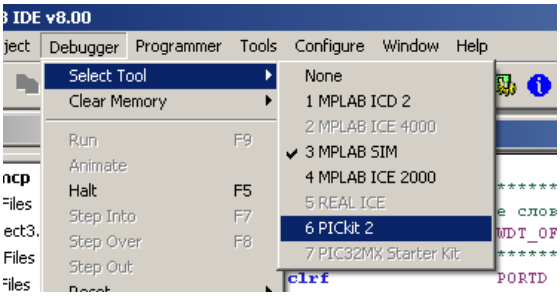

Рис. 20. Выбор отладчика в среде разработки.

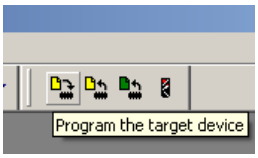

Рис. 21. Кнопка для программирования лабораторного макета.

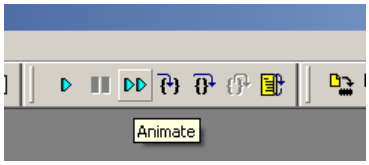

Рис. 22. Кнопка для запуска программы в режиме Animate.

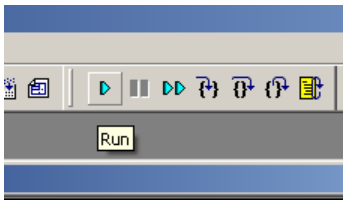

Рис. 23. Кнопка для запуска программы в режиме Run.

Соберите схему рис. 24 на лабораторном макете как показано на рис. 25. Когда схема собрана, выберите в среде разработки имеющийся у вас отладчик, например PICkit 2 (рис. 19). Запрограммируйте лабораторный макет (рис. 20). Запустите программу в режиме RUN (рис. 22) на макете и предъявите результат.

Аппаратное обеспечение

В этой работе используется три семисегментных индикатора. Семисегментный индикатор состоит из восьми светодиодов и каждый из них работает как обычный светодиод. Только в семисегментном индикаторе светодиоды выполнены в виде полосок. Полоски имеют определённое положение на плоскости индикатора. Комбинируя горящие и не горящие светодиоды можно получать различные символы. Катоды у всех диодов объединены и подключаются к выводам PORTC. Принципиальная электрическая схема семисегментных индикаторов лабораторного макета изображена на рис. 24. Принципиальная электрическая схема для выполнения лабораторной работы изображена на рис. 25. На рис. 26 показаны соединения на плате лабораторного макета.

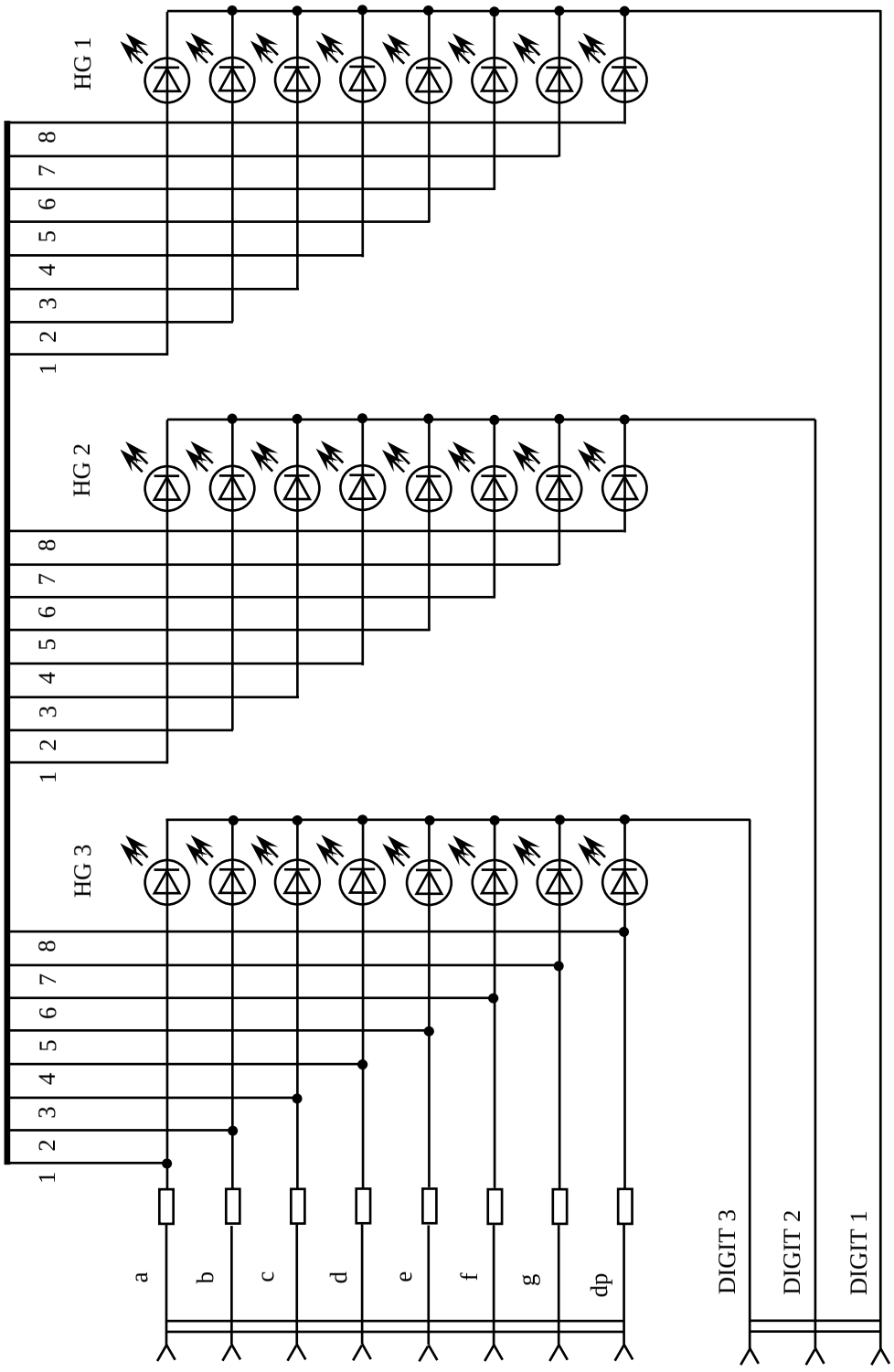

Рис. 24. Принципиальная электрическая схема семисегментных индикаторов.

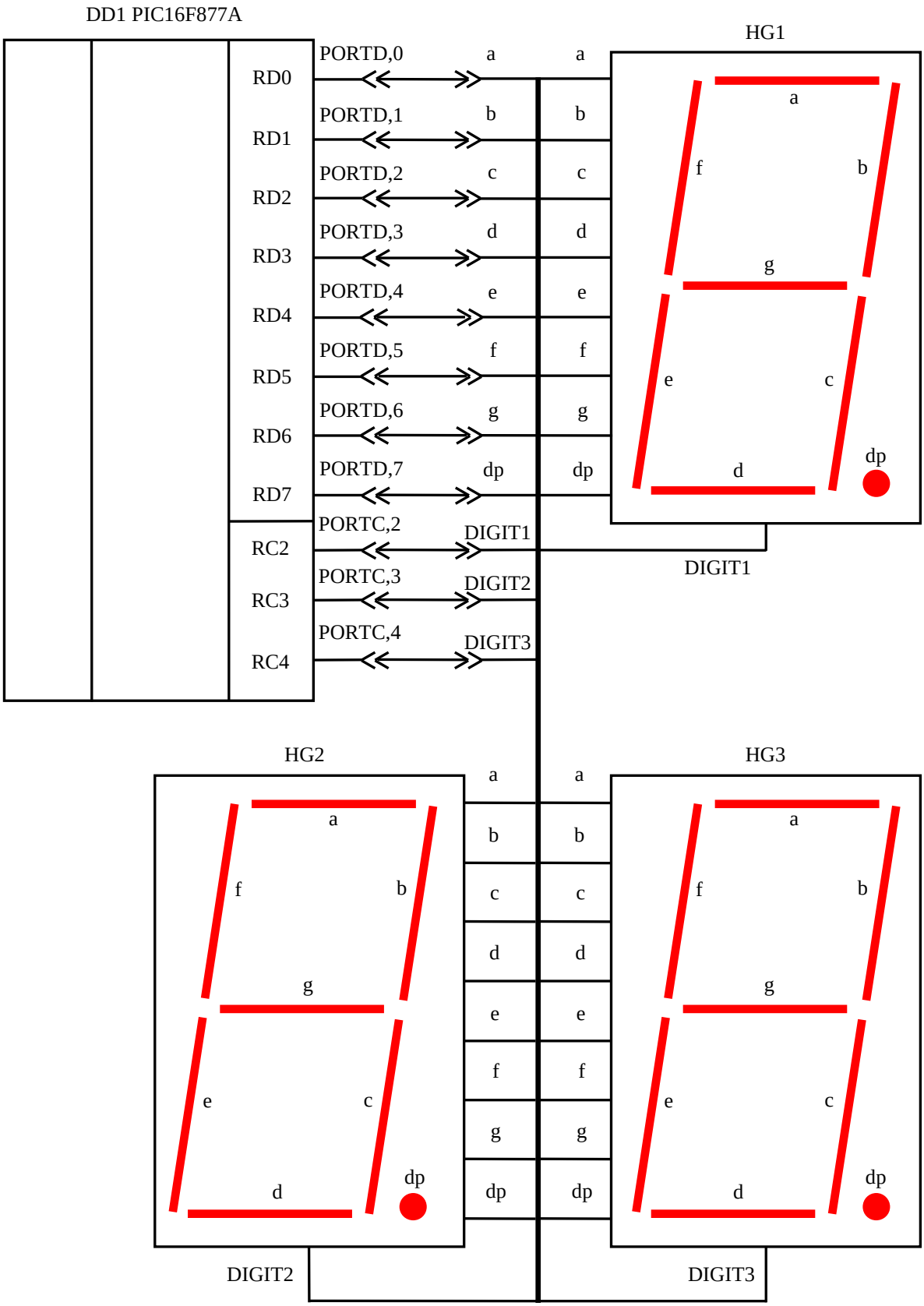

Рис. 25. Принципиальная электрическая схема для выполнения лабораторной работы.

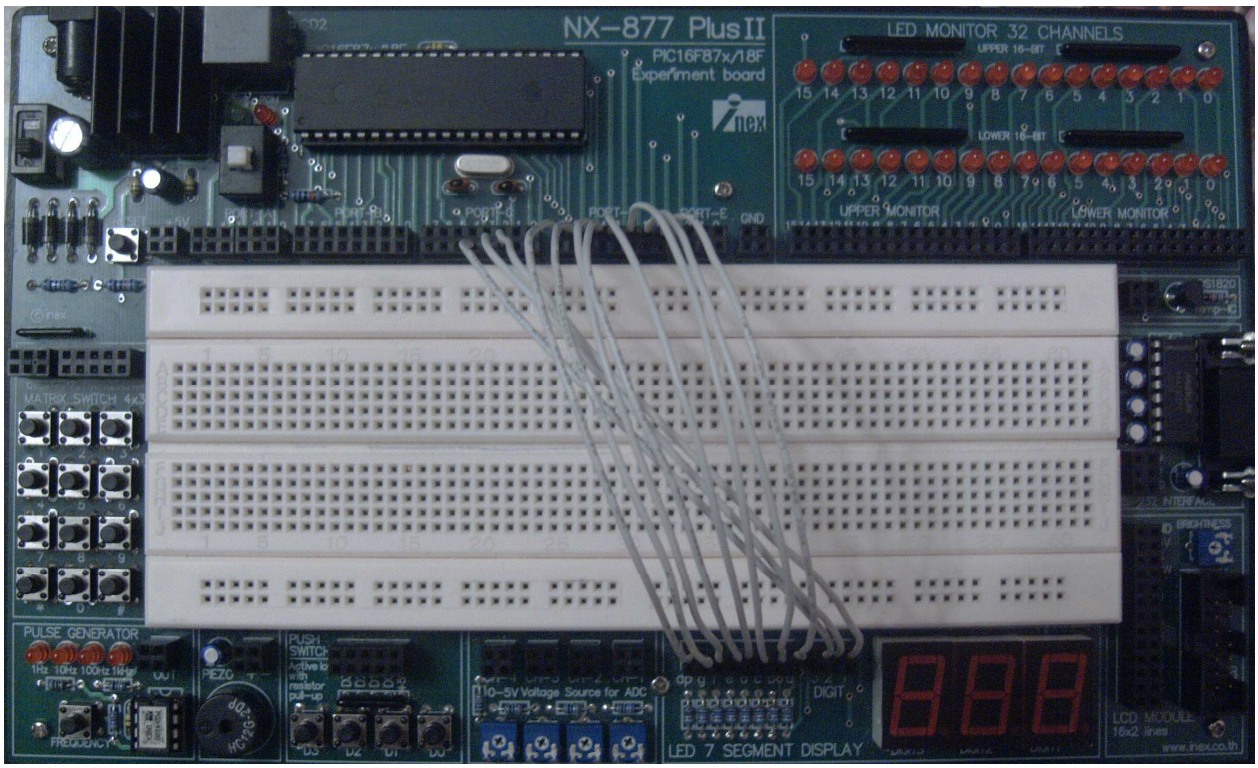

Рис. 26. Схема, собранная на лабораторном макете.

Текст программы Project5.c

```
#include <pic.h>
#define ONE     0x06
#define TWO     0x5B
#define THREE   0x4F
#define FOUR    0x66
#define FIVE    0x6D
#define SIX     0x7D
#define SEVEN   0x07
#define EIGHT   0x7F
#define NINE    0x6F
#define ZERO    0x3F
 __CONFIG (HS & WDTDIS & LVPDIS & DEBUGEN);
 unsigned char ClockSeconds           = 0;
 unsigned char ClockDesSeconds        = 0;
 unsigned char ClockMinutes           = 0;
 unsigned char DispState              = 0;
 unsigned char DispSeconds            = THREE;
 unsigned char DispDesSeconds         = TWO;
 unsigned char DispMinutes            = ONE;

   void init (void)
      {
         PORTD = 0x00;
         PORTC = 0x00;
         TRISC = 0x03;
         TRISD = 0x00;
      }

   void initTMR0 (void)
      {
                              TMR0=0x00;
                              T0CS=0;
                              PSA=1;
      }

   void initTMR1 (void)
      {
                              TMR1H      = 0x00;
                              TMR1L      = 0x00;
/* инициализация таймера:
*  T1CON = 0b00001111;*/
                              T1CKPS1    = 0;
                              T1CKPS0    = 0;
         T1OSCEN    = 1;
         T1SYNC     = 1;
         TMR1CS     = 1;
/*
*  Если бит T1OSCEN установлен,
*  тактовый генератор модуля TIMER1 включается.
*  После включения генератора,
```

```
*  программист должен сделать программную задержку
*  (паузу) для запуска генератора.
*/
        unsigned char i = 0;
        for (i = 255; i > 0; i --) NOP();
        TMR1ON      = 1;
        PIR1        = 0b00000000;
        TMR1L       = 0x00; /* задаем время первого таймера */
        TMR1H       = 0x80; /* задаем время первого таймера */
        PIE1        = 0b00000001;/*Нулевой бит TMR1IE*/
        INTCON      = 0b11000000;//разрешим все прерывания (INTCON) GIE = 1; PEIE = 1;
    }

  void Disp (void)
    {
                                TMR0IF=0;
                                        if (DispState==0)//RC2==0
                                    {
                                        PORTD = 0x00;
                                        RC2=1;
                                        RC3=0;//Выводим DispDesSeconds
                                        RC4=1;
                                        PORTD=DispDesSeconds;
                                    }
                                    else if (DispState==1)//RC3==0
                                    {
                                        PORTD = 0x00;
            RC3=1;
                                        RC2=1;
                                        RC4=0;//Выводим DispMinutes
            PORTD = DispMinutes;
                                    }
                                    else if (DispState==2)//RC4==0
                                    {
                                        PORTD = 0x00;
            RC4=1;
                                        RC3=1;
                                        RC2=0;//Выводим DispSeconds
            PORTD = DispSeconds;
                                    }
                                    DispState++;
            if (DispState > 2) DispState = 0;
                                }
 void Translate (void)
   {
      switch (ClockSeconds)
        {
          case 0x00: DispSeconds     = ZERO;     break;
          case 0x01: DispSeconds     = ONE;      break;
          case 0x02: DispSeconds     = TWO;      break;
          case 0x03: DispSeconds     = THREE;    break;
          case 0x04: DispSeconds     = FOUR;     break;
```

```
            case 0x05: DispSeconds      = FIVE;      break;
            case 0x06: DispSeconds      = SIX;       break;
            case 0x07: DispSeconds      = SEVEN;     break;
            case 0x08: DispSeconds      = EIGHT;     break;
            case 0x09: DispSeconds      = NINE;      break;
         }
      switch (ClockDesSeconds)
         {
            case 0x00: DispDesSeconds   = ZERO;      break;
            case 0x01: DispDesSeconds   = ONE;       break;
            case 0x02: DispDesSeconds   = TWO;       break;
            case 0x03: DispDesSeconds   = THREE;     break;
            case 0x04: DispDesSeconds   = FOUR;      break;
            case 0x05: DispDesSeconds   = FIVE;      break;
         }
   }

void Clock (void)
   {
      if (ClockSeconds > 9)
         {
            ClockSeconds = 0;
            ClockDesSeconds ++;
         }
            if (ClockDesSeconds > 5)
               {
                  ClockDesSeconds = 0;
                  ClockMinutes++;
               }
                  if (ClockMinutes > 9)
                     ClockMinutes = 0;
   }

void interrupt timer1_isr (void)
   {
      if (TMR1IF && TMR1IE) // Если прерывание от первого таймера TMR1F (PIR1)
обработка прерывания.
         {
            TMR1IF = 0;
            TMR1L = 0x00;//задаём время работы таймера до следующего срабатывания.
            TMR1H = 0x80;//задаём время работы таймера до следующего срабатывания.
            ClockSeconds ++;
            Clock();
            return;
          }
      else return;// Если прерывание не от первого таймера TMR1F (PIR1), то выход.
   }

void main (void)
{
   init();
   initTMR0();
```

```
   initTMR1();
   while (1)
   {
   while (!TMR0IF) continue;
      Disp();
      Translate();
   }
}
```

Индивидуальные задания

Исправьте функцию Translate() так, чтобы таймер считал минуты, десятки секунд и секунды.

Соберите секундомер, с тремя кнопками. По нажатию первой кнопки начинается счёт, по нажатию второй кнопки счёт останавливается, по нажатию третьей кнопки показания секундомера сбрасываются.

Соберите секундомер с одной кнопкой: по перовому нажатию кнопки начинается счёт, по второму нажатию кнопки счёт останавливается, по третьему нажатию кнопки показания секундомера сбрасываются.

Контрольные вопросы

1. Что такое периферийные модули?
2. Как включают и настраивают периферийные модули?
3. Что такое счётчики, что такое таймеры?
4. Какова разрядность первого таймера в PIC16F877A?
5. Какой диапазон чисел может уместиться в шестнадцати разрядах?
6. Что такое предделитель?
7. Каким образом происходит счёт в первом таймере?
8. При каких условиях устанавливается флаг TMR1IF?
9. Что такое прерывания?

Оглавление:

Список рисунков: