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., Кисляков С.В.

«ОТНОСИТЕЛЬНАЯ АДРЕСАЦИЯ»

Методические указания к выполнению лабораторной работы

Санкт - Петербург
2009

Богураев М.В., Кисляков С.В. «ОТНОСИТЕЛЬНАЯ АДРЕСАЦИЯ». Методические указания к выполнению лабораторной работы №8(10). СПб: ГОУ «Лицей при СПбГУТ», 2009.

ЛАБОРАТОРНАЯ РАБОТА №8
«ОТНОСИТЕЛЬНАЯ АДРЕСАЦИЯ»

Цель работы

Научиться использовать относительную адресацию в своих программах для табличного преобразования данных. Освоить моделирование в MPLAB IDE. Овладеть навыками программирования PIC микроконтроллеров.

Теоретические основы

Для преобразования данных одного вида в данные другого вида удобно пользоваться табличными преобразованиями. Например, нужно подсчитывать события – это данные одного вида. Количество событий надо отображать на индикаторе – код семисегментного индикатора – это данные другого вида. Для преобразования количества в код составляют таблицу, с кодами индикатора. Коды в таблице располагают в порядке возрастания: 0,1,2…9. Для преобразования считывают содержимое таблицы по адресу начало_таблицы+[смещение]. Смещение определяется ячейкой подсчёта. Тогда адрес ячейки с кодом индикатора определяется числом из ячейки подсчёта событий, поэтому смещение от начала таблицы равно количеству событий. Для преобразования потребуются две ячейки памяти – одна для подсчёта событий, а вторая для кода семисегментного индикатора, считанного из адреса равному количеству событий. В микроконтроллерах PIC среднего подсемейства есть команда RETLW, которая как раз и позволяет считать число (константу или литерал) из памяти команд в регистр W. Чтобы выбрать нужную инструкцию используют команду CALL. Например, CALL TABLE. Метка TABLE указывает на команду, которая расположена в начале таблицы, состоящей из последовательности команд RETLW. Первая команда после метки TABLE (ADDWF PCL,F) прибавляет содержимое регистра W к содержимому счётчика команд (PC). Перед командой CALL TABLE в регистр W помещают содержимое ячейки подсчёта событий, тогда к PC прибавляется значение (смещение) из ячейки подсчёта событий. После вызова команды CALL TABLE счётчик команд показывает на начало таблицы, а после выполнения команды ADDWF PCL,F (сложения PCL с W) возвращает в регистр W код семисегментного индикатора. Затем из регистра W полученный код нужно поместить в порт ввода – вывода для отображения на семисегментном индикаторе.

В этой лабораторной работе первый таймер настроен на одну секунду. Подсчёт ведётся в диапазоне 0 – 9 секунд. Прерывание от первого таймера происходит раз в секунду. По этому прерыванию увеличивается на единицу содержимое ячейки SECONDS. Получается, что частота изменения содержимого регистра SECONDS составляет один раз за секунду. В основной программе происходит бесконечный цикл преобразования SECONDS -> NUMBER_TWO. Поэтому ячейка NUMBER_TWO обновляется часто, а вот изменяется ячейка один раз в секунду, как только изменится SECONDS. А вот ячейка NUMBER_ONE задаётся в начале программы, содержит код буквы S, и в течение всей программы не изменяется.

По нулевому таймеру осуществляется динамическая индикация. То есть переключаются два вывода PORTC – RC2 и RC3. Когда RC2 равно нулю, тогда RC3 равно единице и наоборот, когда RC2 равно единице, тогда RC3 равно нулю. Аппаратное обеспечение собрано так, что RC2 соединяет с нулём правый индикатор. Поэтому, если на RC2 ноль, то на PORTD подаётся содержимое NUMBER_TWO. Вывод RC3 соединяет с нулём левый индикатор. Поэтому, если на RC3 ноль, то на PORTD подаётся NUMBER_ONE. И тогда будет видно, что левый индикатор (DIGIT2, который соединён с RC3) всё время отображает символ S, а правый (DIGIT1, который соединён с RC2) показывает число секунд от нуля до девяти.

Задание

Изучите AN556 (файл 00556e.pdf). Создайте проект Project10, и откомпилируйте текст программы Project10. Запустите программу в симуляторе MPLAB IDE и промоделируйте работу программы в окне Watch, а каналы T1CKI, RC2, RC3 и PORTD в Logic Analyzer.

Подключите два семисегментных индикатора на лабораторном макете к порту PORTD микроконтроллера. Выводы DIGIT1 и DIGIT2 подключите к порту PORTC (RC2, RC3) микроконтроллера. Запрограммируйте микроконтроллер и продемонстрируйте работу программы на лабораторном макете.

Порядок выполнения

На диске C:\ в папке Projects создайте папку Project10. В эту папку скопируйте файл Project10 и создайте проект с названием Project10. Откомпилируйте программу. Выберите отладчик MPLAB SIM (Debugger/Select Tool/MPLAB SIM) и настройте симулятор (Debugger/Settings). В окне настройки симулятора выберите вкладку Animation/Realtime Updates. В графе Animate step time установите значение 1 мс, поставьте галочку напротив Enable Realtime watch updates и введите цифру 1 в графе x100 msecs.

Затем из меню View откройте Simulator Logic Analyzer (рис. 1). В появившемся окне (рис. 2) нажмите кнопку Channels. Появится окно Configure Channels. Для отображения каналов T1CKI, RC2, RC3 нужно выделить добавляемый канал и нажать кнопку Add (рис. 3). Выберите каналы T1CKI, RC2, RC3 (Рис. 4). 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. 5) 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. 7), нажмём кнопку OK. Остаётся только добавить шину, для чего в окне Configure Channels выбираем PORTD и нажимаем кнопку Add (рис. 8) и затем кнопку OK.

Теперь в окне Logic Analyzer будет отображаться состояние выводов T1CKI, RC2, RC3 и PORTD (Рис. 9).

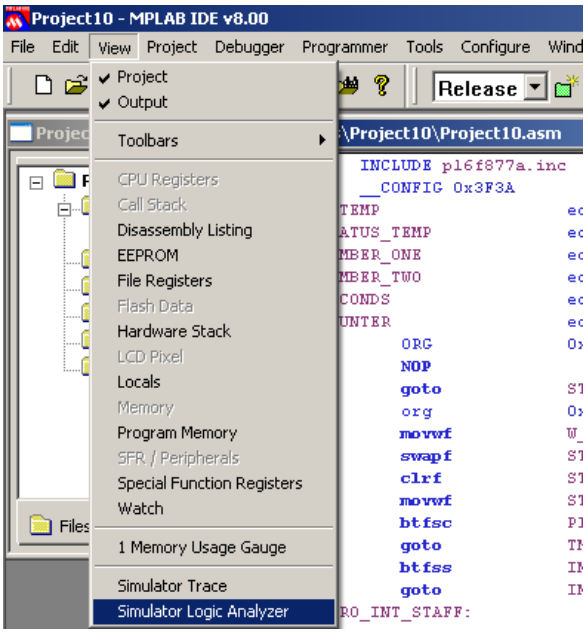

Рис. 1. Запуск логического анализатора.

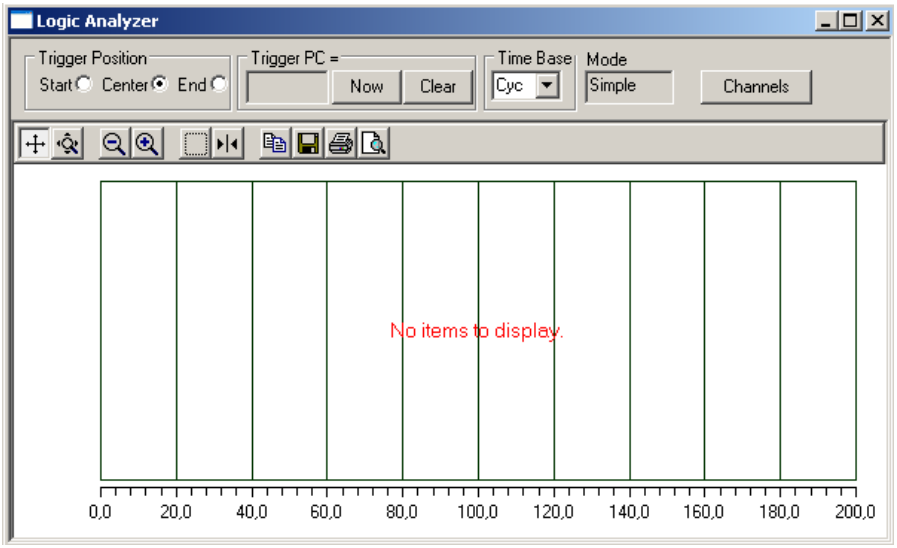

Рис. 2. Окно логического анализатора.

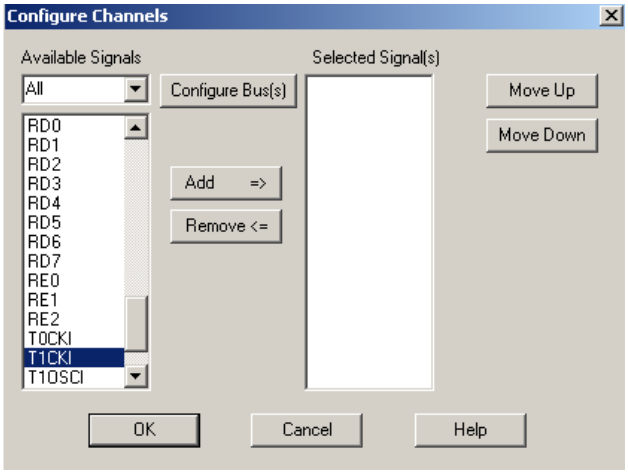

Рис. 3. Добавление канала.

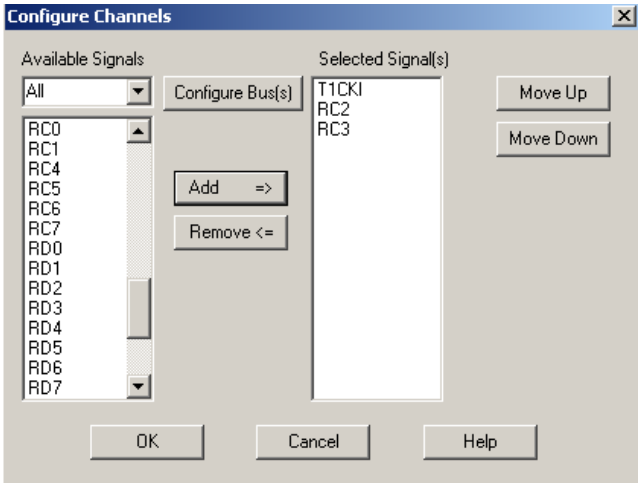

Рис. 4. Настроены каналы T1CKI, RC2, RC3.

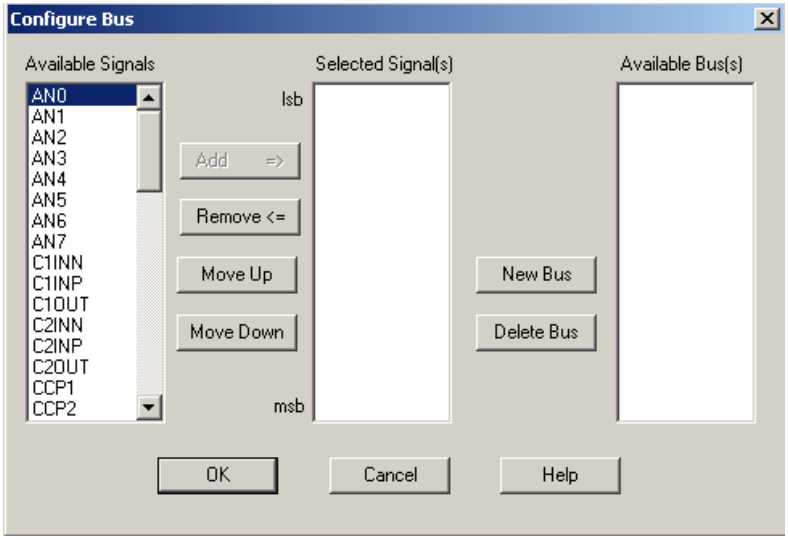

Рис. 5. Формирование шины.

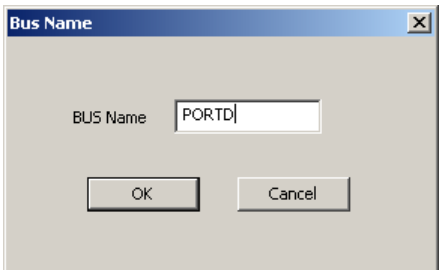

Рис. 6. Ввод названия шины.

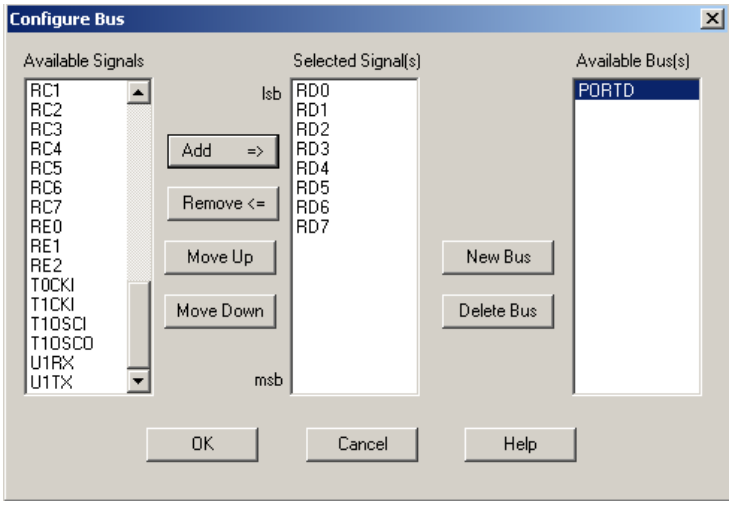

Рис. 7. Сформированная шина PORTD.

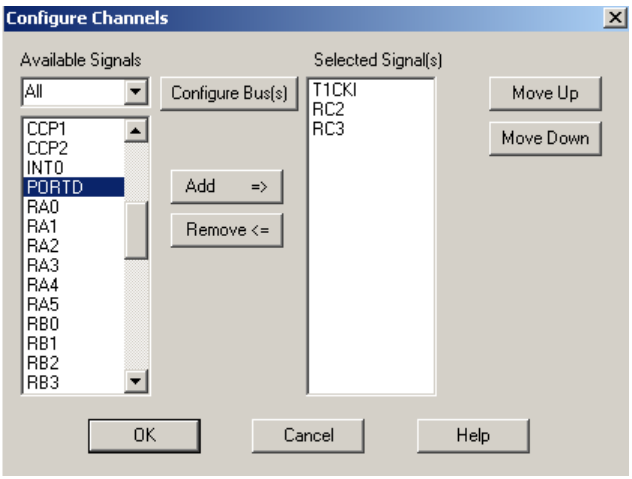

Рис. 8. Добавление сформированной шины.

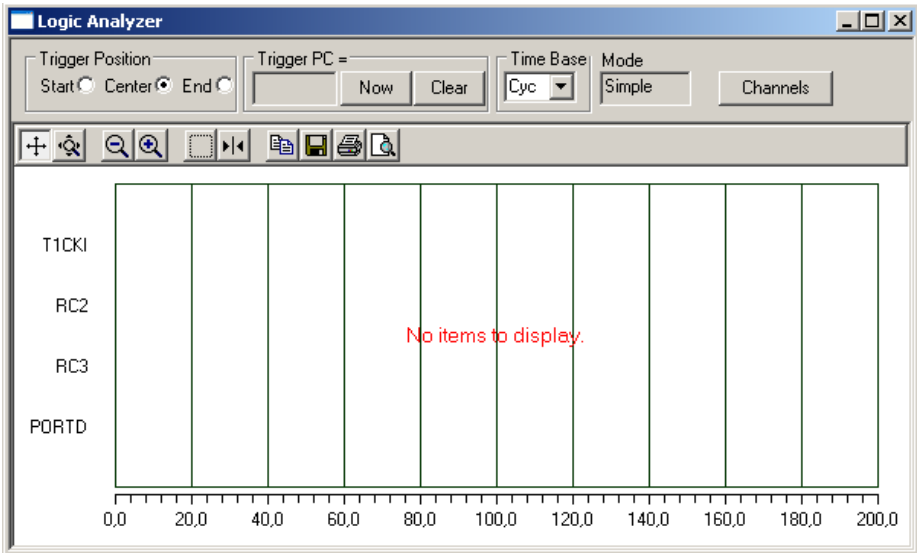

Рис. 9. Настроенное окно Logic Analyzer.

Для симуляции работы модуля TIMER1 создадим файл стимулов (Debugger/Stimulus/New Workbook – рис. 10). В появившемся окне стимулов выберем вкладку Clock Stimulus. В графе PIN выберем T1CKI (рис. 11). 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. 12). 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. Если стимулы не применить, то симуляция не будет происходить.

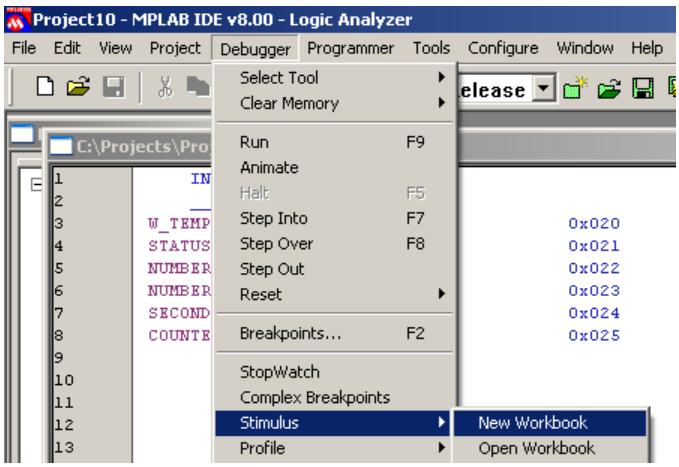

Рис. 10. Создание нового файла стимулов.

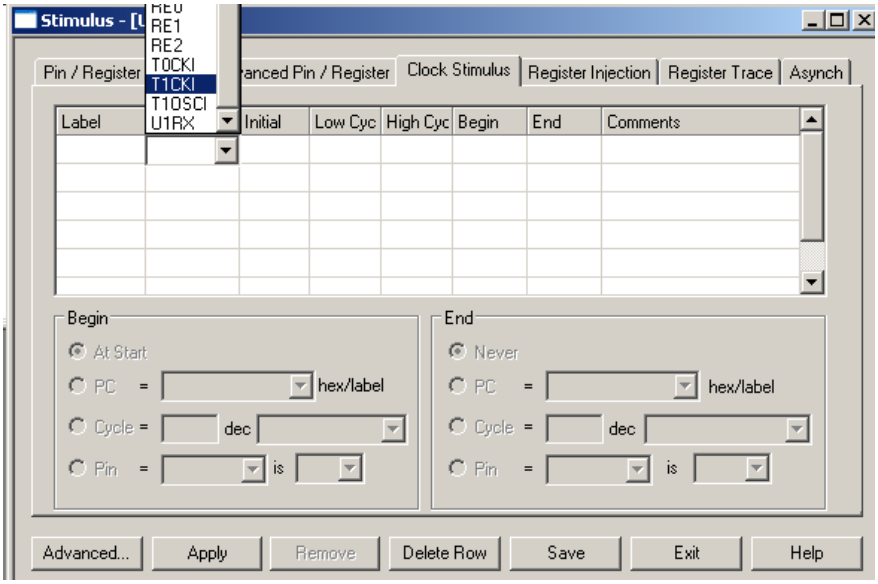

Рис. 11. Выбор симуляции напряжения на ножке T1CKI.

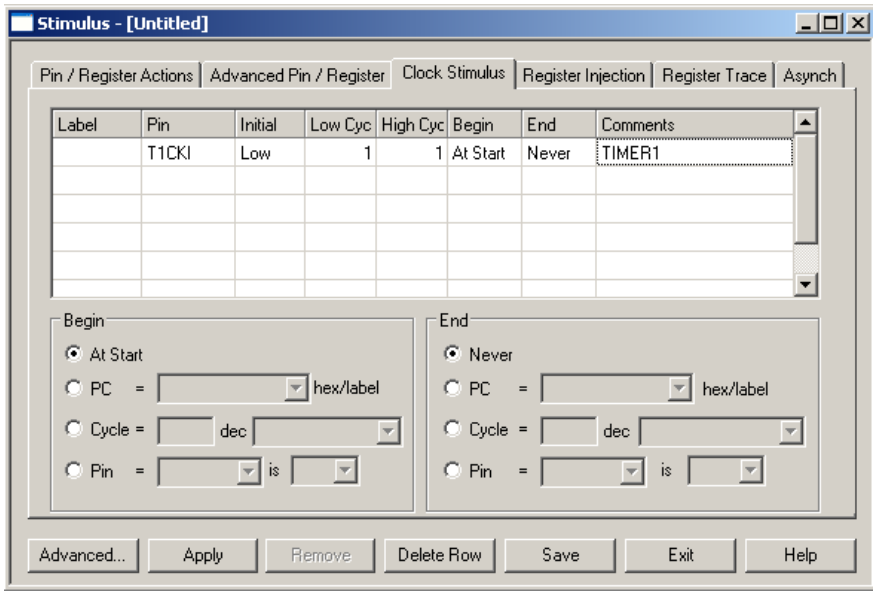

Рис. 12. Вид настроенного окна стимулов.

Чтобы симуляция происходила в удобном для наблюдения темпе, выберем время одного шага в Animate режиме 1 миллисекунду (Debugger/Settings), и поставим галочку напротив Enable Realtime watch updates (Рис. 14). Настроим окно Watch (View/Watch) для отображения регистров PORTC, PORTD, TMR1L, TMR1H и двух пользовательских регистров NUMBER_ONE и NUMBER_TWO (рис. 15). Кнопкой Animate запустим симулятор (рис. 16). После начала симуляции Logic Analyzer будет отображать состояние выводов T1CKI, RC2, RC3 и PORTD (Рис. 17). Изменения регистров PORTC, PORTD, TMR1H и TMR1H будут видны и в окне Watch.

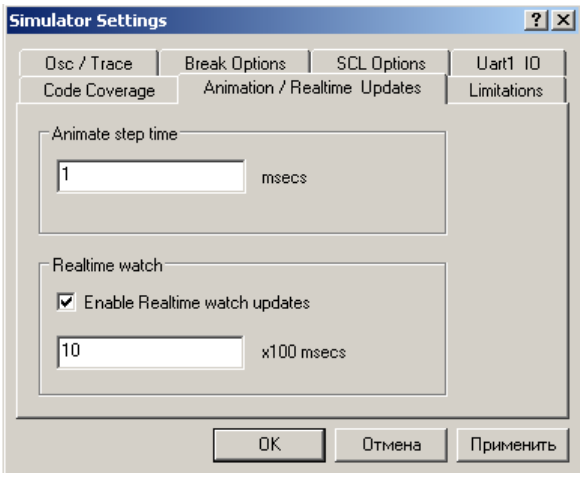

Рис. 14. Выбор времени одного шага в Animate режиме.

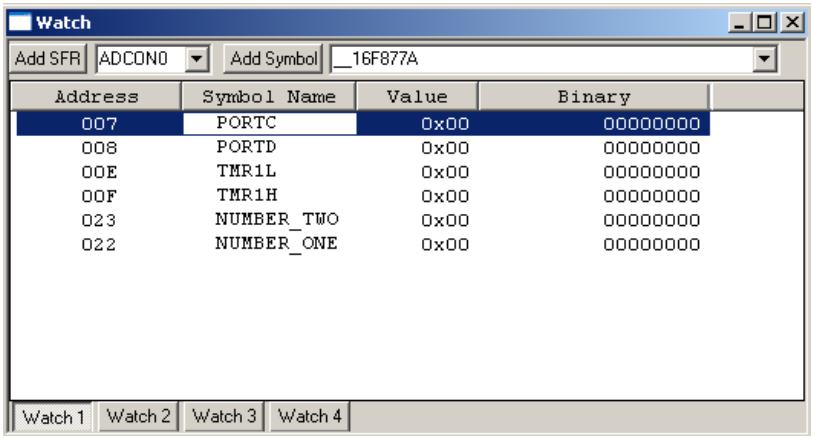

Рис. 15. Вид настроенного окна Watch.

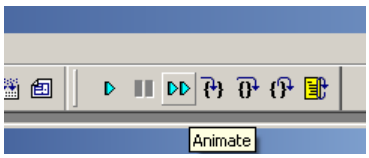

Рис. 16. Кнопка Animate.

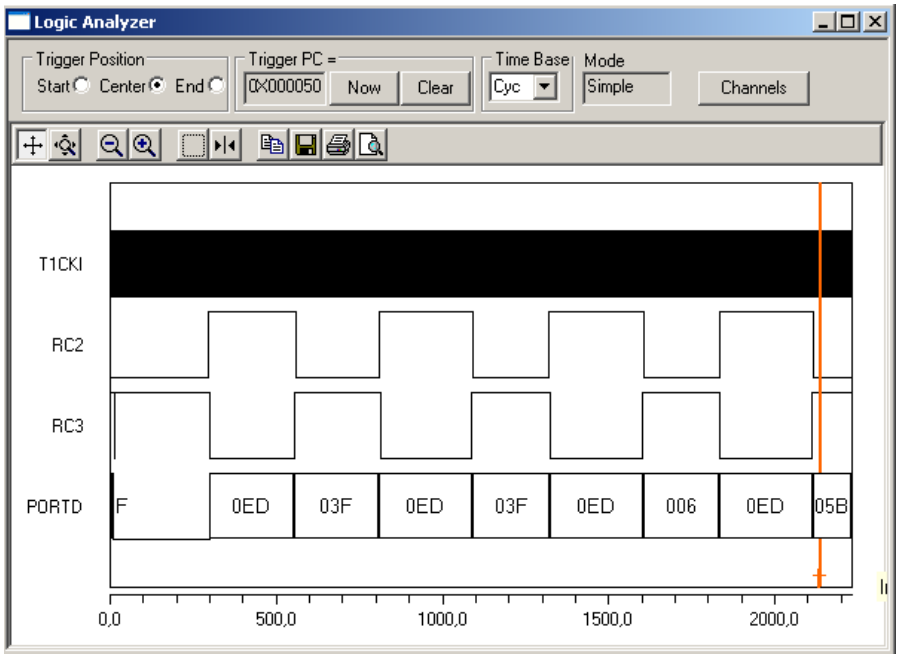

Рис. 17. Вид окна Logic Analyzer в процессе симуляции.

После того, как будет получено несколько периодов сигнала в окне Logic Analyzer, остановите симулятор кнопкой Halt (рис. 18).

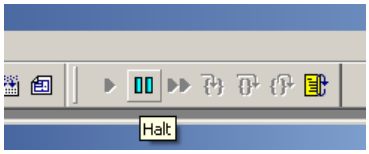

Рис. 18. Кнопка Halt для остановки симулятора.

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.

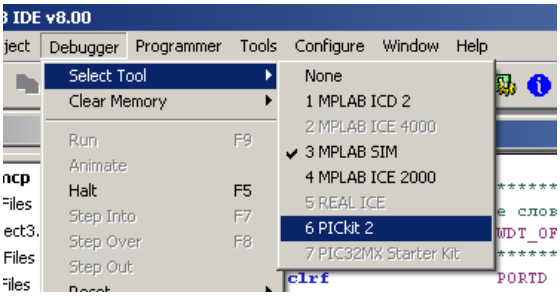

Рис. 19. Выбор отладчика в среде разработки.

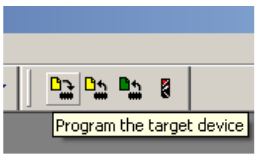

Рис. 20. Кнопка для программирования лабораторного макета.

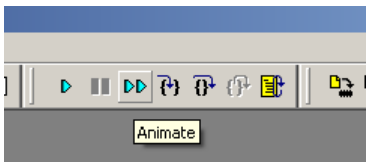

Рис. 21. Кнопка для запуска программы в режиме Animate.

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

После симуляции измените текст программы: строки movlw 0xFE закомментируйте поставив ; перед строкой, а строки movlw 0x080 раскомментируйте убрав ; После переделки программы 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 Animate (рис. 21). Затем запустите программу в режиме RUN (рис. 22) на макете и предъявите результат.

Аппаратное обеспечение

Аппаратное обеспечение аналогично предыдущей работе. В этой работе используется два 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. По нулевому таймеру осуществляется динамическая индикация: переключаются два вывода PORTC – RC2 и RC3. Первый таймер настроен на одну секунду. Подсчёт ведётся в диапазоне 0 – 9 секунд. Первый индикатор всё время отображает символ S, а второй показывает число секунд от нуля до девяти.

Принципиальная электрическая схема для выполнения лабораторной работы изображена на рис. 23. На рис. 24 показаны соединения на плате лабораторного макета.

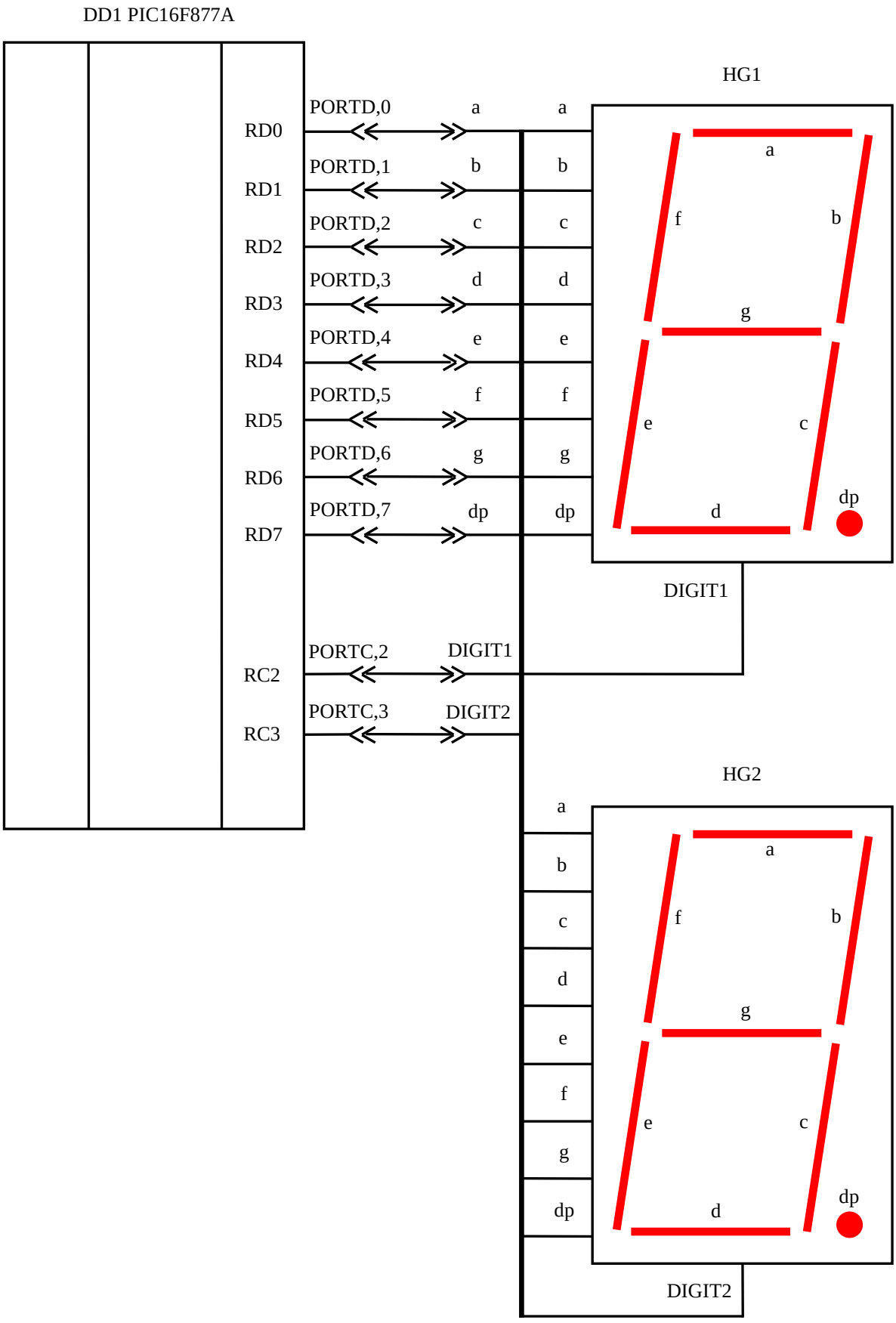

Рис. 23. Схема электрическая принципиальная.

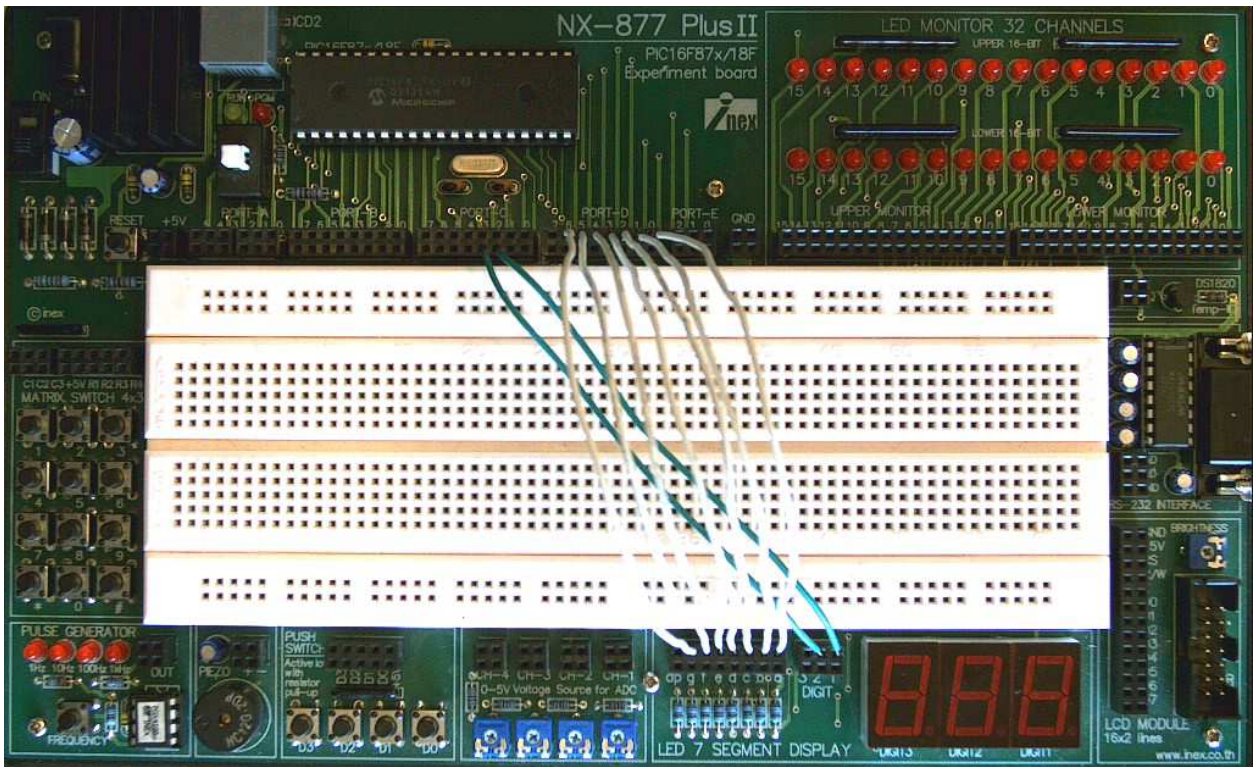

Рис. 24. Схема, собранная на лабораторном макете.

Программное обеспечение

Алгоритм основной программы изображён на рис. 25, алгоритм обработчика прерывания на рис. 26.

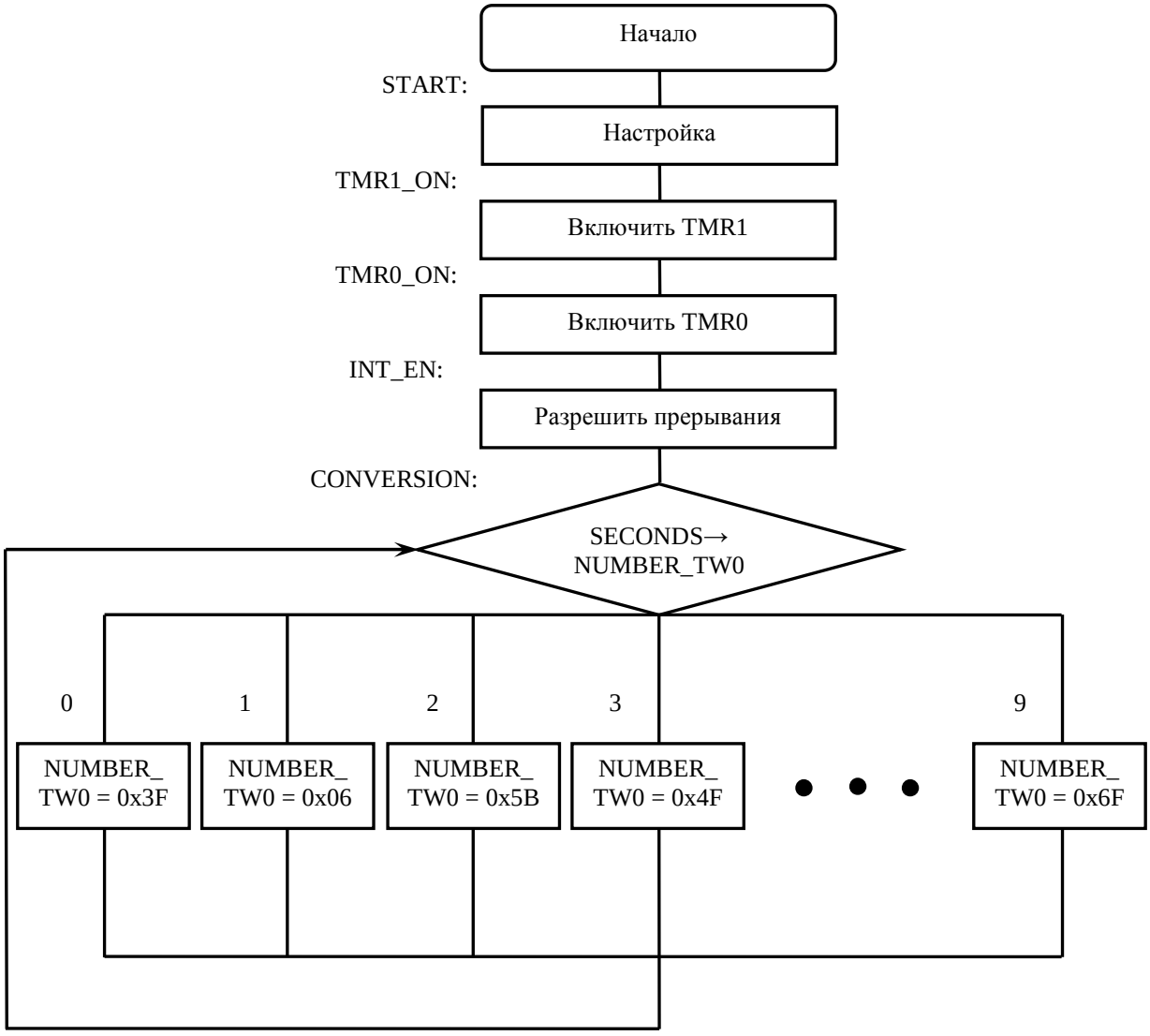

Рис. 25. Алгоритм основной программы Project10.

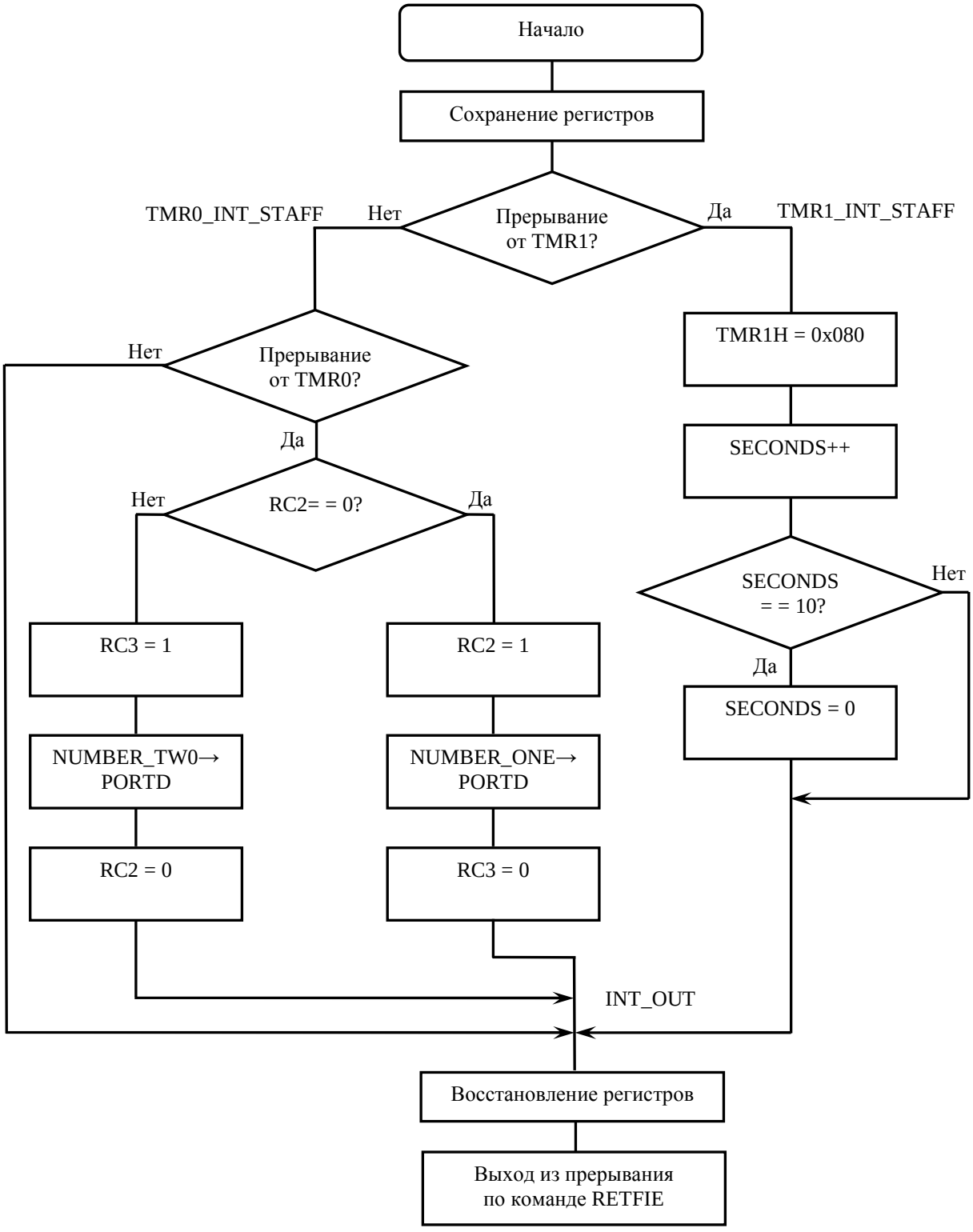

Рис. 26. Алгоритм обработчика прерывания программы Project10.

Текст файла Project10.ASM

```
                INCLUDE     p16f877a.inc
                __CONFIG _CP_OFF & _DEBUG_ON & _WRT_OFF & _CPD_OFF
& _LVP_OFF & _BODEN_OFF & _PWRTE_ON & _WDT_OFF & _HS_OSC
W_TEMP          equ         0x020
STATUS_TEMP     equ         0x021
NUMBER_ONE      equ         0x022
NUMBER_TWO      equ         0x023
SECONDS         equ         0x024
                ORG         0x00
                NOP
                goto        START
                org         0x04
                movwf       W_TEMP
                swapf       STATUS,W
                clrf        STATUS
                movwf       STATUS_TEMP
                btfsc       PIR1,TMR1IF
                goto        TMR1_INT_STAFF
                btfss       INTCON,T0IF
                goto        INT_OUT
TMR0_INT_STAFF:
                bcf         INTCON,T0IF
                btfsc       PORTC,2
                goto        NUMBER_TWO_OUT
                bsf         PORTC,2
                movf        NUMBER_ONE,W
                movwf       PORTD
                bcf         PORTC,3
                goto        INT_OUT
NUMBER_TWO_OUT:
                bsf         PORTC,3
                movf        NUMBER_TWO,W
                movwf       PORTD
                bcf         PORTC,2
                goto        INT_OUT
TMR1_INT_STAFF:
                bcf         PIR1,TMR1IF
                ;movlw      0x080
                movlw       0xFF
                movwf       TMR1H
                incf        SECONDS,F
                movlw       0x0A
                subwf       SECONDS,W
                btfsc       STATUS,Z
                goto        SECONDS_CLEAR
                goto        INT_OUT
SECONDS_CLEAR:
                clrf        SECONDS
INT_OUT:
                swapf       STATUS_TEMP,W
                movwf       STATUS
```

```
                swapf       W_TEMP,F
                swapf       W_TEMP,W
                retfie
START:
                clrf        PORTC
                clrf        PORTD
                clrf        NUMBER_ONE
                clrf        NUMBER_TWO
                clrf        SECONDS
                bsf         STATUS,RP0
                movlw       0x00
                movwf       TRISD
                movlw       b'00000011'
                movwf       TRISC
                bcf         STATUS,RP0
                bsf         PORTC,3
                movlw       b'11101101'
                movwf       NUMBER_ONE
TMR1_ON:
                clrf        TMR1L
                clrf        TMR1H
                bsf         T1CON, T1OSCEN
                bsf         T1CON, T1SYNC
                bsf         T1CON, TMR1CS
                ;movlw      0x80
                movlw       0xFE
                movwf       TMR1H
                bsf         T1CON, TMR1ON
TMR0_ON:
                clrf        TMR0
                bsf         STATUS, RP0
                bcf         OPTION_REG,T0CS
                bcf         STATUS, RP0
INT_EN:
                bsf         INTCON, T0IE
                bsf         INTCON, PEIE
                bsf         STATUS, RP0
                bsf         PIE1, TMR1IE
                bcf         STATUS, RP0
                bsf         INTCON, GIE
CONVERSION:
                movf        SECONDS,W
                call        TABLE
                movwf       NUMBER_TWO
                goto        CONVERSION
TABLE:
                ADDWF       PCL,F
                RETLW       0x3F; 0
                RETLW       0x06; 1
                RETLW       0x5B; 2
                RETLW       0x4F; 3
                RETLW       0x66; 4
```

```
RETLW       0x6D; 5
RETLW       0x7D; 6
RETLW       0x07; 7
RETLW       0x7F; 8
RETLW       0x6F; 9
RETLW       0x77; A
RETLW       0x7C; B
RETLW       0x39; C
RETLW       0x5E; D
RETLW       0x79; E
RETLW       0x71; F
END
```

Индивидуальные задания

Напишите программу для подсчёта числа прерываний от первого таймера (числа секунд) в диапазоне от 0 до 59 секунд. Первая цифра показывает десятки секунд, вторая цифра показывает единицы секунд.

Контрольные вопросы

1. Что происходит в результате выполнения команды ADDWF PCL,F?
2. Что происходит в результате выполнения команды RETLW k?
3. Как часто меняется содержимое пользовательского регистра NUMBER_ONE?
4. Как часто меняется содержимое пользовательского регистра NUMBER_TWO?
5. При каких уровнях RC2, RC3 на PORTD будет подано содержимое регистра NUMBER_ONE?
6. При каких уровнях RC2 и RC3 на PORTD будет подано содержимое регистра NUMBER_TWO?
7. Подправьте программу так, чтобы счёт секунд вёлся в порядке 0,1,2…9, A,B,C,D,E,F.

Оглавление:

Список рисунков: