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.

«ВВОД ИНФОРМАЦИИ С КЛАВИАТУРЫ»

Методические указания к выполнению лабораторной работы

Санкт - Петербург
2013

Богураев М.В. «ВВОД ИНФОРМАЦИИ С КЛАВИАТУРЫ». Методические указания к выполнению лабораторной работы №4. СПб: ГБОУ «Лицей при СПбГУТ», 2013.

ЛАБОРАТОРНАЯ РАБОТА №4
«ВВОД ИНФОРМАЦИИ С КЛАВИАТУРЫ»

Цель работы

Научиться считывать информацию из устройства ввода, если в качестве устройства ввода используется клавиатура. Овладеть навыками программирования PIC микроконтроллеров, приобрести навыки работы в интегрированной среде разработки MPLAB IDE. Освоить моделирование устройства ввода при симуляции программы в MPLAB IDE.

Теоретические основы

Для подключения клавиатуры в микроконтроллерах PIC16F877A рекомендуется использовать PORTB и прерывание по изменению сигнала на входах PORTB <RB7:RB4>. Чаще всего клавиатура представляет собой матрицу из кнопок. Кнопка может замыкать свой столбец и свою строку. Клавиатура из 12 кнопок содержит четыре строки и три столбца. Если нумерацию кнопок, строк и столбцов вести с нуля (см. схему электрическую принципиальную клавиатуры), то номер кнопки можно определить по формуле:

$$A = (N \cdot i) + j\,, \qquad (1)$$

где A – номер замкнутой кнопки, N – количество столбцов клавиатуры, i – номер опрашиваемой строки, j – номер столбца, в котором обнаружена замкнутая кнопка. Формула описывает тот факт, что с увеличением номера опрашиваемой строки номер кнопки увеличивается на количество столбцов (в нашем случае N = 3). А номер кнопки – это сумма номера столбца нажатой кнопки и количества опрошенных кнопок. То есть с каждой опрошенной строкой количество опрошенных кнопок увеличивается на количество кнопок в строке (в нашем случае 3 кнопки). Клавиатура собрана так, что на столбцы через балластные резисторы всё время подано напряжение логической единицы. Ножки порта, подключенные к строкам, настроены на выход и управляют подачей нулей на строки, поэтому их называют драйверами строк. Ножки порта, подключенные к столбцам, настроены на вход и принимают информацию, поэтому их называют сенсорами столбцов. Опрос клавиатуры происходит по строкам и столбцам. Нажатая кнопка определяется как ноль на одном из столбцов. Для этого сначала подают логический ноль на нулевую строку и проверяют столбцы на ноль – если кнопка замкнута, тогда на одном из столбцов появится уровень логического ноля. Затем подают ноль на первую строку и проверяют столбцы на ноль – если кнопка замкнута, тогда на одном из столбцов появится уровень логического ноля и так далее по всем строкам. Последовательный опрос состояния выводов контроллера в зарубежной литературе иногда называют polling.

В этой лабораторной работе клавиатура опрашивается постоянно, поэтому прерывания не используются. Но приведённый текст можно модифицировать и использовать в подпрограмме обработки прерывания. Операцию умножения ($N \cdot i$ в (1)) можно реализовать путём сложения. В нашем случае добавляется тройка после каждой опрошенной строки. Если кнопка была нажата и её номер определён, делается табличное преобразование номера кнопки в семисегментный код. Этот код передаётся на индикатор то тех пор, пока не будет нажата другая кнопка. Алгоритм опроса построен таким образом, что определяется только одна нажатая кнопка. Наибольший приоритет у кнопки «1». Если зажать эту кнопку, то нажатие других кнопок не будет влиять на работу программы.

Выводы RB6 и RB7 используются и для программирования и для клавиатуры. После программирования работу макета проверяют в режиме RUN (горит зелёный светодиод).

Задание

Создайте проект, откомпилируйте программу Project4. Промоделируйте в симуляторе MPLAB SIM нажатие кнопки. Соберите схему на лабораторном макете. Запрограммируйте микроконтроллер, переведите макет в режим RUN и запустите программу на лабораторном макете. Продемонстрируйте результат работы программы.

Порядок выполнения

На диске C:\ в папке Projects создайте папку Project4. В эту папку скопируйте файл Project4 и создайте проект с названием Project4. Откомпилируйте программу. Настройте симулятор и стимулы, как описано ниже.

Выберите отладчик MPLAB SIM (Debugger/Select Tool/MPLAB SIM).

Поскольку аппаратное обеспечение собрано так, что на три старших бита PORTB постоянно подается логическая единица, то при симуляции нужно задать высокие уровни на ножках <RB7:RB5>. Для этого создайте файл стимулов (Debugger/Stimulus/New Workbook). В появившемся окне выберите вкладку Asinch и выберите RB7 в выпадающем списке столбца Pin/SFR (рис.1).

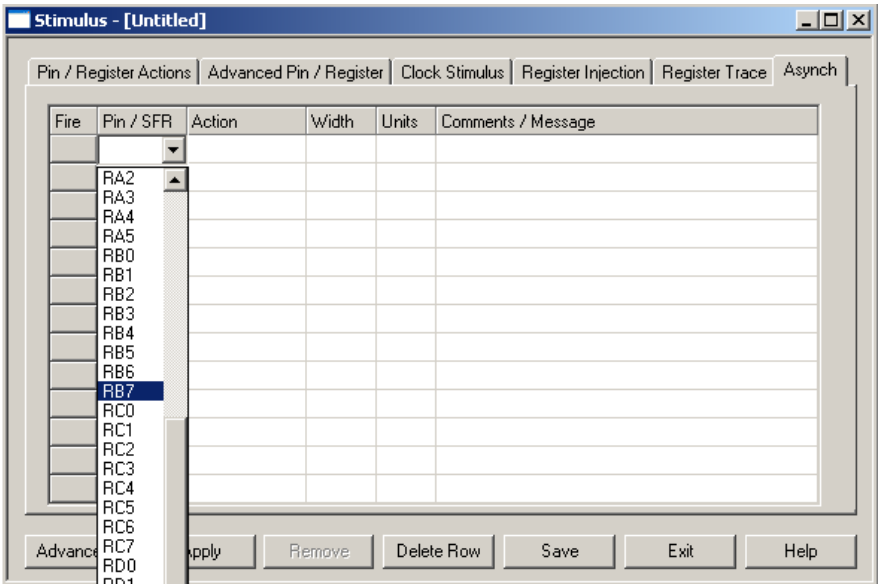

Рис. 1. Выбор выводов контроллера для симуляции.

В столбце Action выберите тип воздействия Toggle – переключение, как показано на рис. 2.

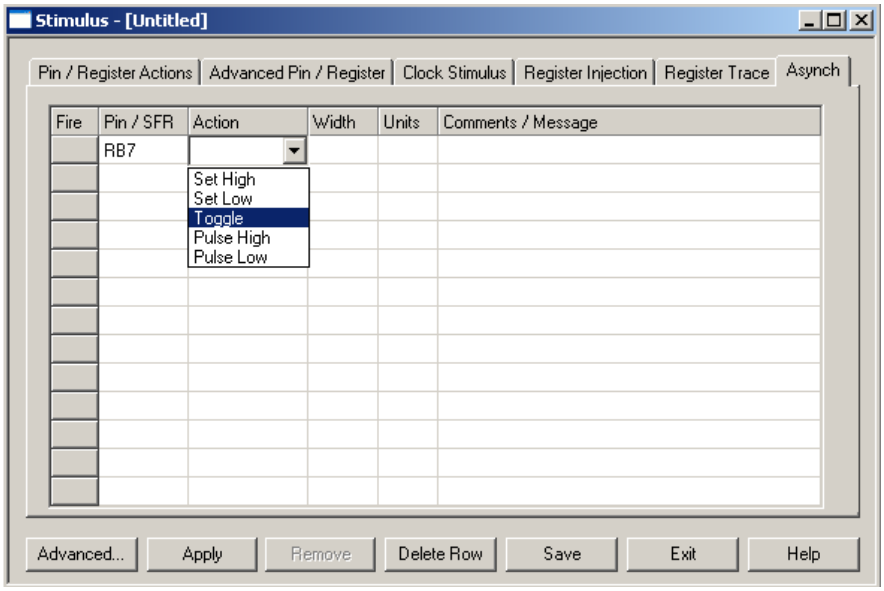

Рис. 2. Выбор типа воздействия на выбранном выводе.

То же самое проделайте для выводов RB6 и RB5 (рис. 3). Теперь выводы <RB7:RB5> подготовлены к симуляции.

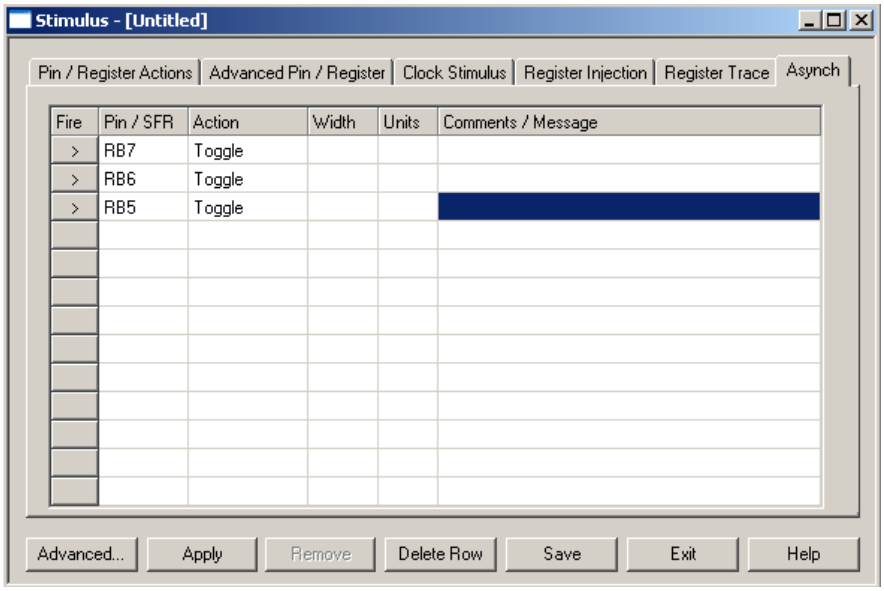

Рис. 3. Вид окна Stimulus после настройки вкладки Asynch.

Симуляция сенсоров столбцов <RB7:RB5> подготовлена. Теперь разберёмся с драйверами строк <RB4:RB1> – подготовим для них стимулы нажатия кнопки. Пусть это будет кнопка SA7. Если кнопка SA7 нажата, то на RB6 устанавливается ноль, но только в том случае, если на RB3 появился ноль (см. аппаратное обеспечение). То есть RB6 = 0, если PORTB = 0xF7 (0b’1111 0111’) и, соответственно, RB6 = 1, если PORTB не равно 0xF7. Нажатие кнопки SA7 должно привести к появлению на семисегментном индикаторе числа 8. Или, что то же самое, на PORTD выводится число 0b’0111 1111’.

Для настройки стимулов откройте в окне Stimulus вкладку Advanced Pin/Registers (рис. 4).

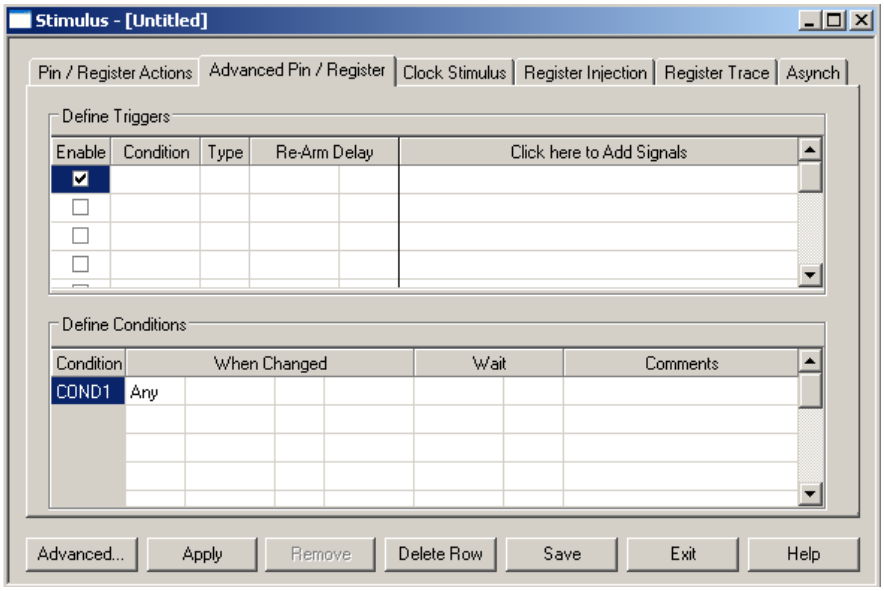

Рис. 4. Вкладка Advanced Pin/Registers окна Stimulus.

Вкладка состоит из двух областей: Define Conditions и Define Triggers. В области Define Conditions задаются условия, по которым будет меняться содержимое ячеек памяти

виртуального микроконтроллера в симуляторе MPLAP SIM. То, какие ячейки памяти будут изменяться в MPLAB SIM, определяют в области Define Triggers.

Зададим условия в области Define Condition. Для этого нужно щёлкнуть левой кнопкой мышки в поле When Changed в столбце Any и выбрать SFR (рис. 5) потому как PORTB находится в этой области памяти.

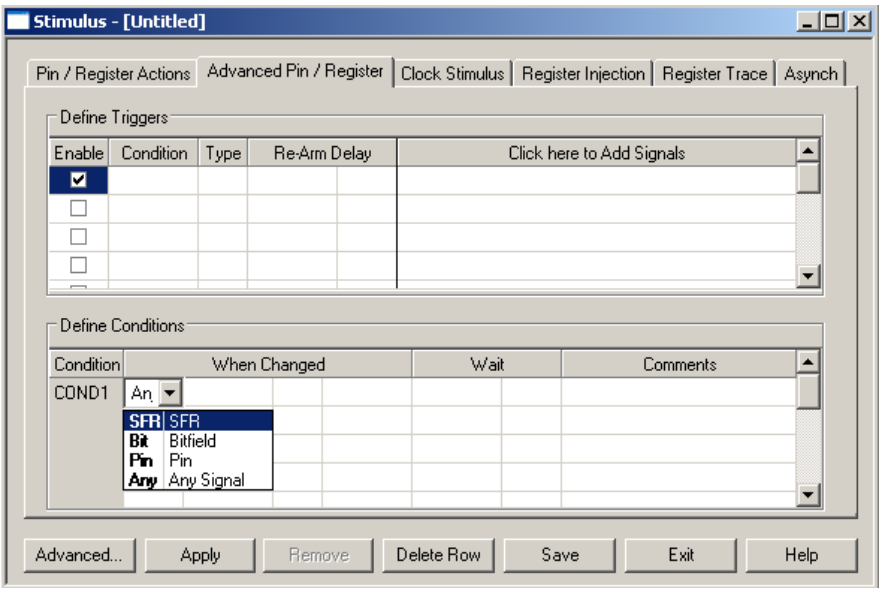

Рис. 5. Выбор области памяти SFR.

Затем нужно выбрать аргумент условия в нашем случае – PORTB. Для этого щелкаем левой кнопкой мыши на следующем столбце и выбираем PORTB (рис. 6).

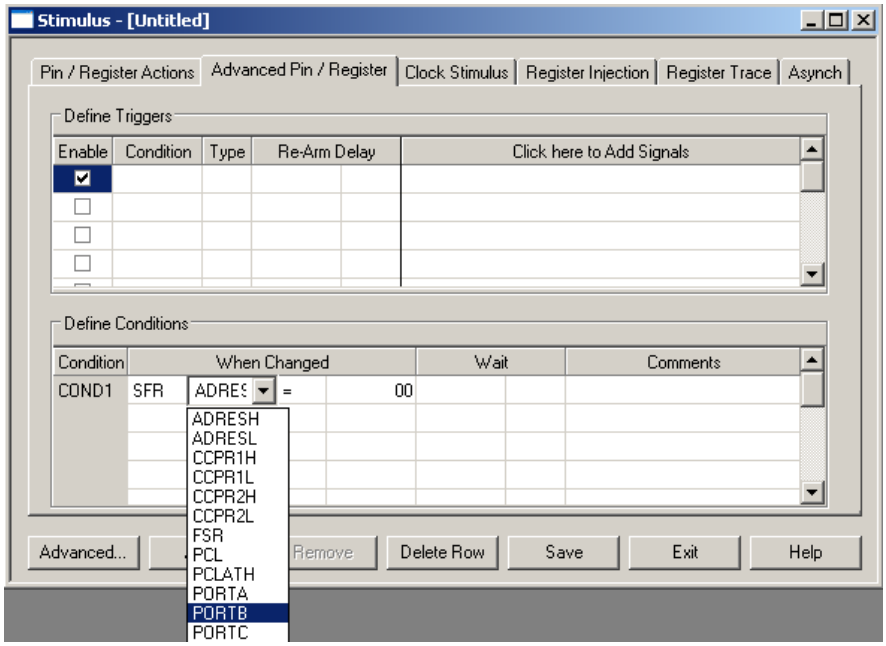

Рис. 6. Выбор аргумента условия COND1.

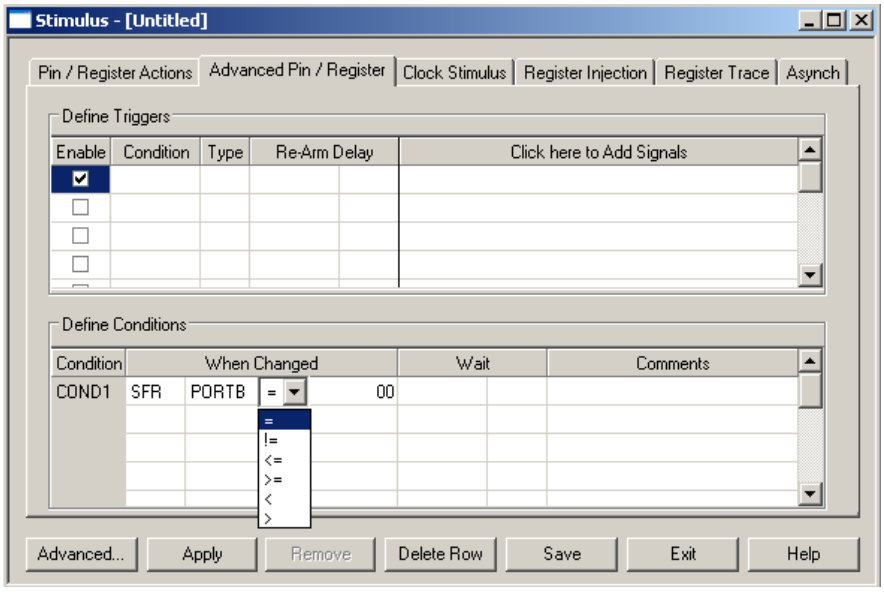

Рис. 7. Выбор условия применения стимула COND1.

Затем выбираем условие применения стимула (рис. 7). Это условие PORTB = 0xF7. Если равенство соблюдено, стимул будет применён. Чтобы вписать число нужно дважды щёлкнуть левой кнопкой мыши в четвёртом столбце поля When Changed, ввести число и нажать enter (рис. 8).

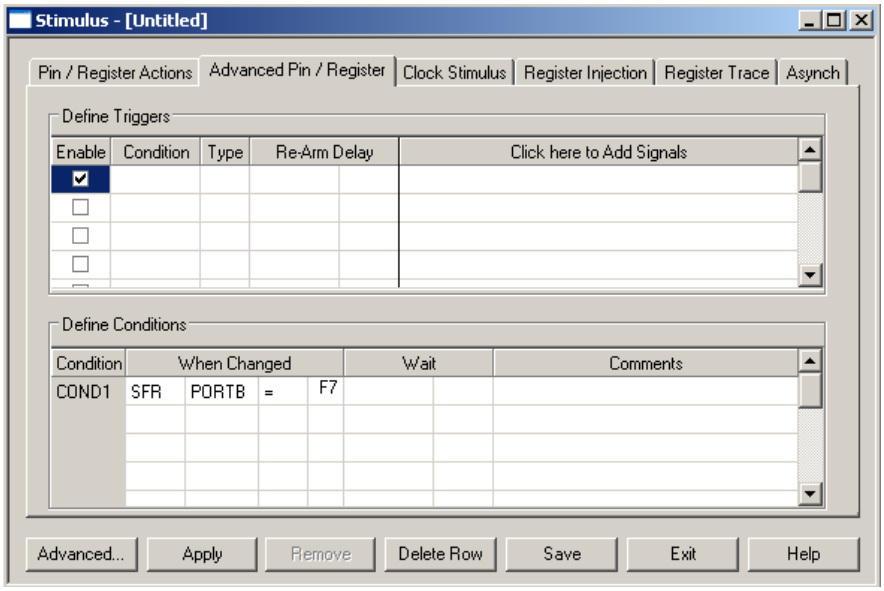

Рис. 8. Условие PORTB = F7 для стимула COND1.

Теперь заполним столбцы поля Wait. В первом столбце введём длительность 10 (нужно дважды щёлкнуть левой кнопкой мыши, ввести число и нажать enter). Во втором столбце поля Wait выберем машинные циклы (рис. 9).

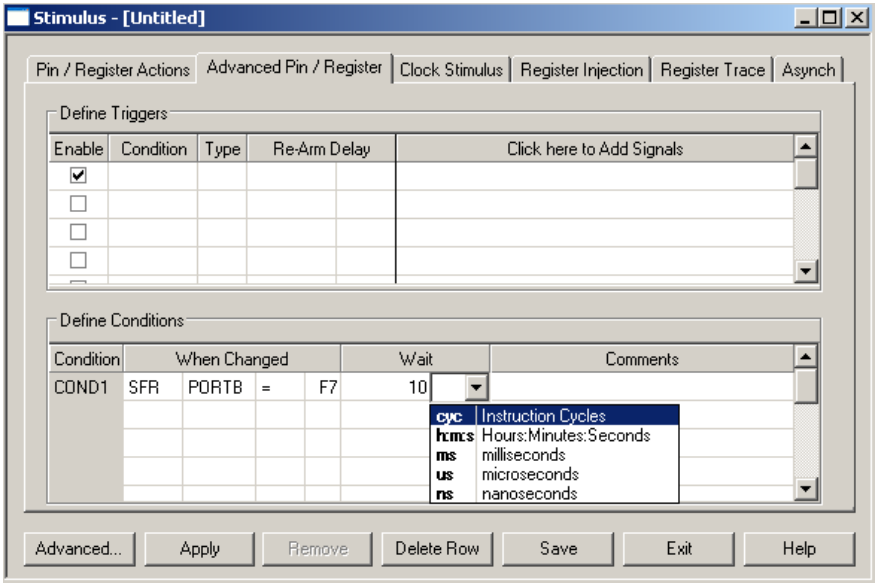

Рис. 9. Условие для стимула COND1 полностью сформулировано.

Условия применения стимула определены. Зададим сами изменения. Для этого в области Define Triggers щёлкнем левой кнопкой мыши в столбце Condition и выберем в выпадающем списке COND1 (рис. 10). Это название первого условия, которое мы только что сформировали.

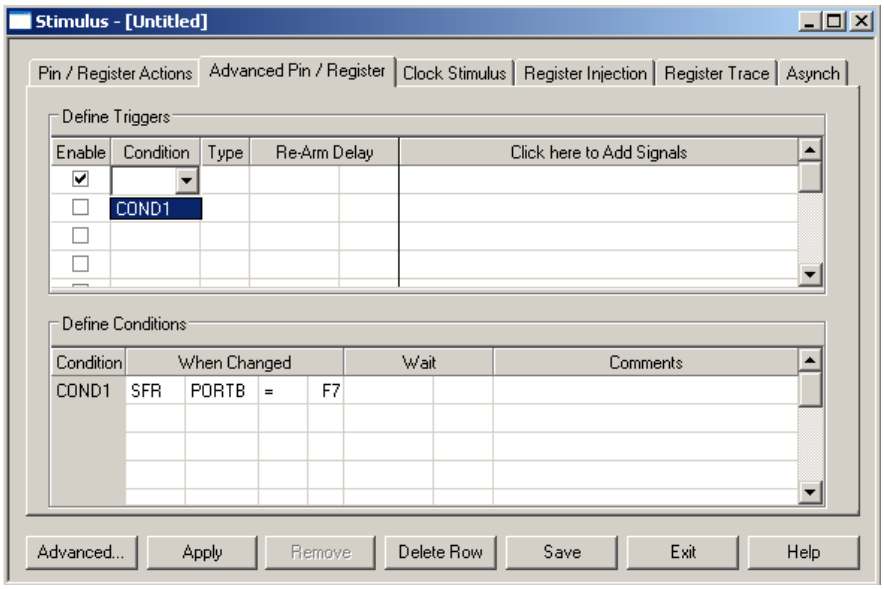

Рис. 10. Настройка изменений для случая PORTB = 0xF7.

Затем выбираем тип применения стимула. Для этого в графе Type выбираем 1x One Time (рис. 11). Это значит, что стимул будет применён однократно.

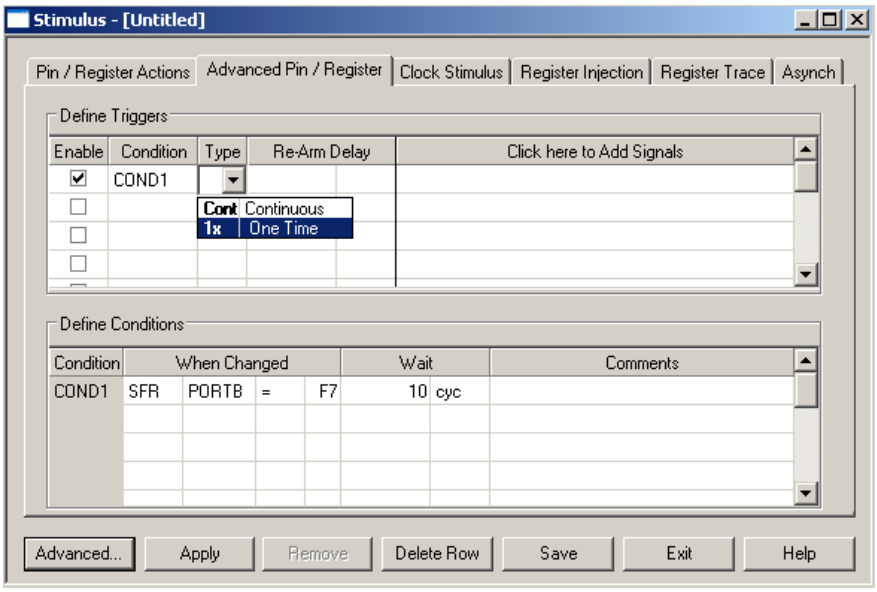

Рис. 11. Выбор типа применения стимула.

Затем кликнем левой кнопкой мыши на надписи Click here to Add Signals. В появившемся окне выберем RB6. Нажмём кнопку Add, затем OK (рис. 12).

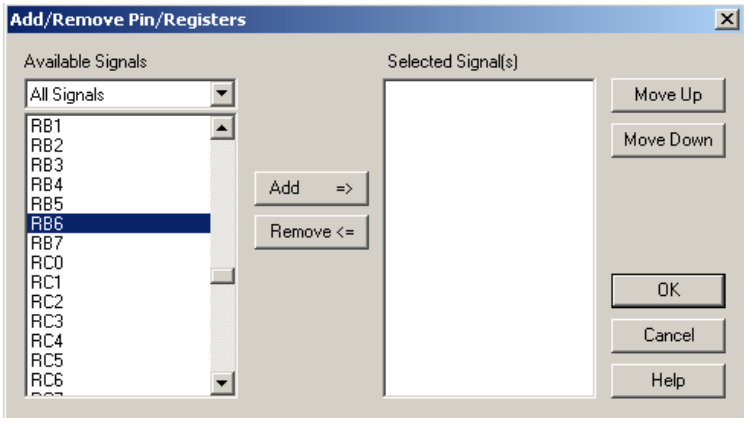

Рис. 12. Добавление RB6.

После того, как в окне Stimulus появится столбец RB6, в этом столбце на пересечении со строкой COND1 нужно щёлкнуть дважды левой кнопкой мыши, набрать ноль и нажать enter (рис. 13). Теперь, если значение PORTB станет равным 0xF7, то через 10 машинных циклов виртуального микроконтроллера на RB6 будет установлен ноль.

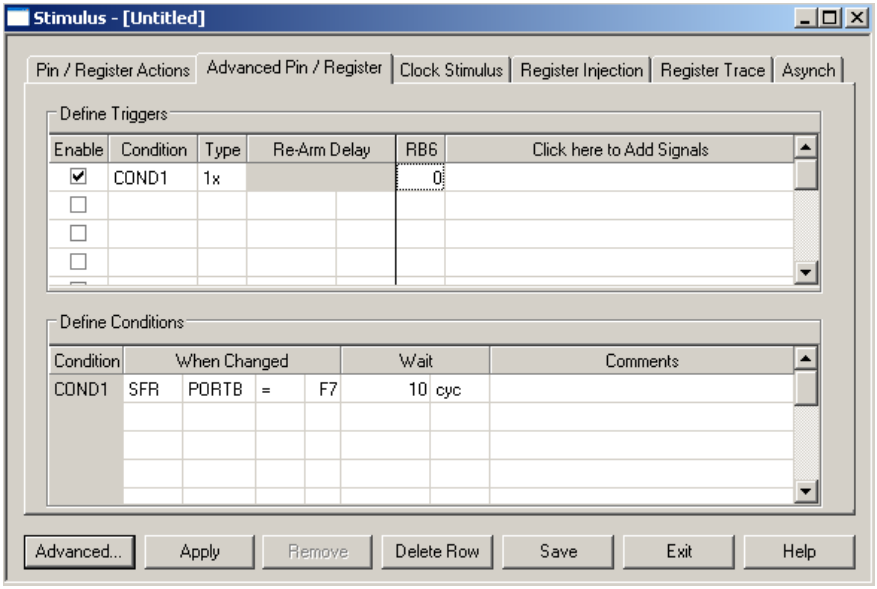

Рис. 13. Настройка значения RB6 после наступления условий COND1.

Далее настроим вторую часть стимула RB6. Как и в предыдущем случае щёлкнем левой кнопкой мыши на первом столбце поля When Changed и выберем SFR (рис. 14).

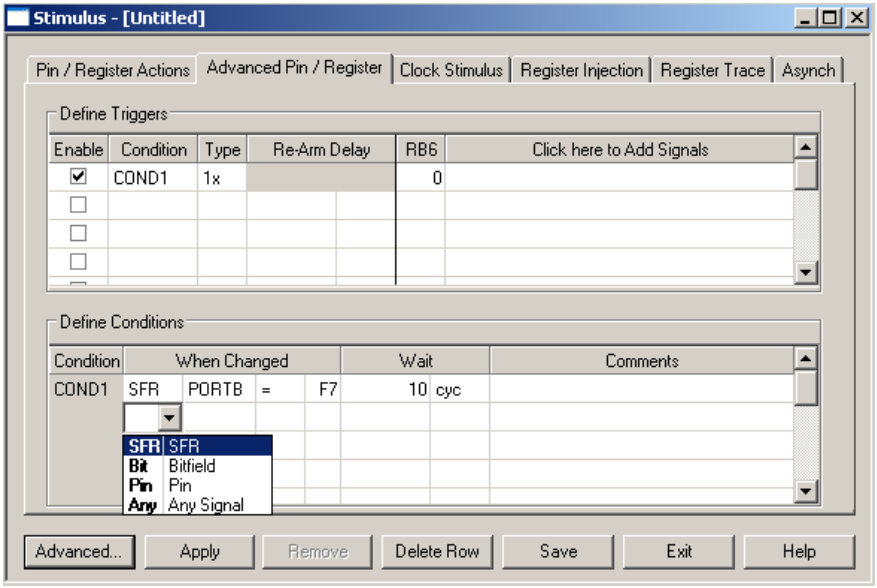

Рис. 14. Настройка второй части стимула – выбор SFR.

В следующем столбце выберем PORTB (рис. 15).

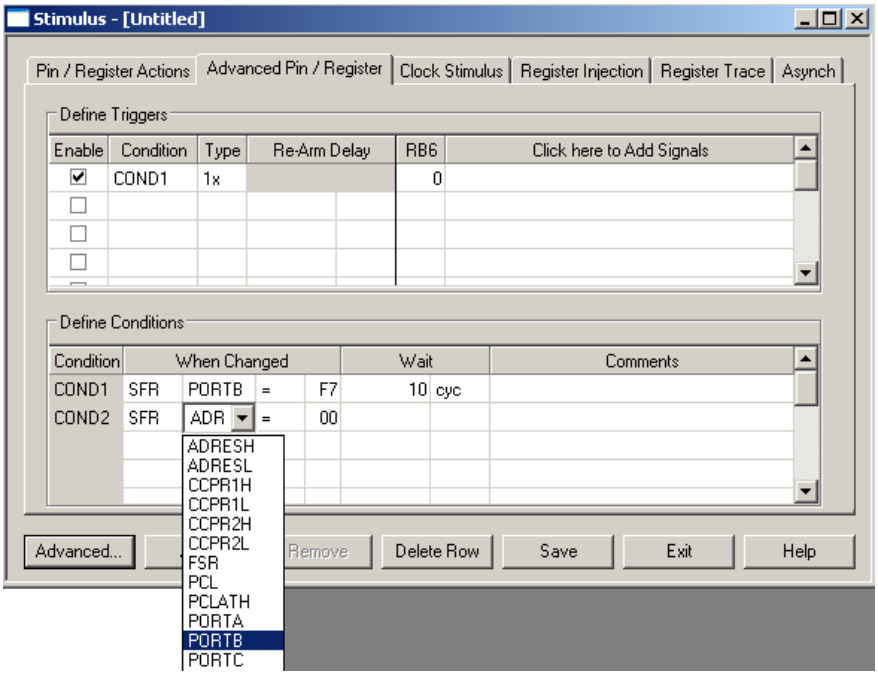

Рис. 15. Выбор PORTB для условия COND2.

Далее выбираем условие != то есть «не равно» (рис. 16). В следующем столбце дважды щёлкнем левой кнопкой мыши, впишем 0xF7 и нажмём enter.

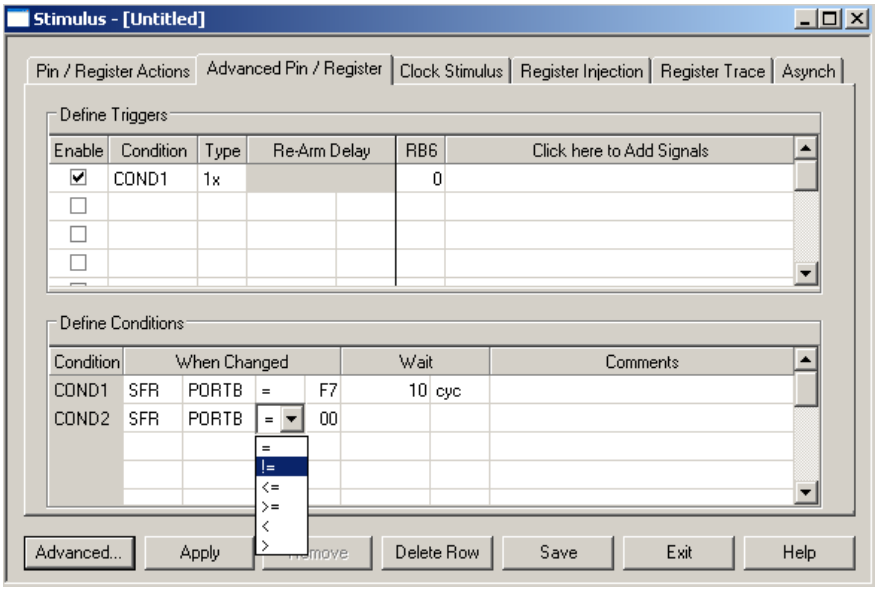

Рис. 16. Выбор условия «не равно» для применения стимула.

Чтобы стимул по условию COND2 выполнялся, в первом столбце графы Wait дважды щёлкнем левой кнопкой мыши, впишем число 1 и нажмём enter. Во втором столбце графы Wait выберем Instruction Cycles (рис. 17). Это определит время применения стимула.

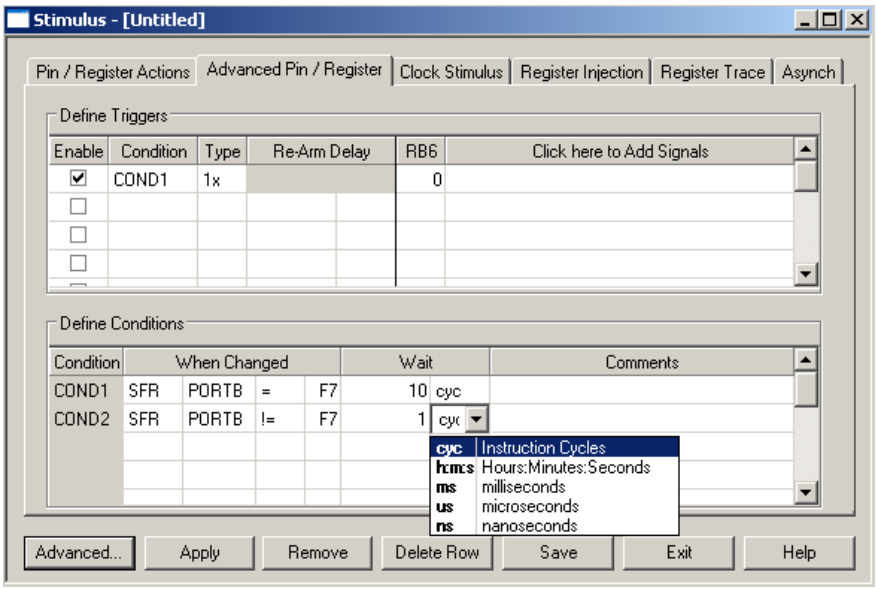

Рис. 17. Условие COND2 сформулировано.

Условия COND2 заданы, теперь определим, что будет изменяться, если PORTB не равно F7. Для этого в области Define Triggers во второй строке в столбце Condition выберем COND2 (рис. 18).

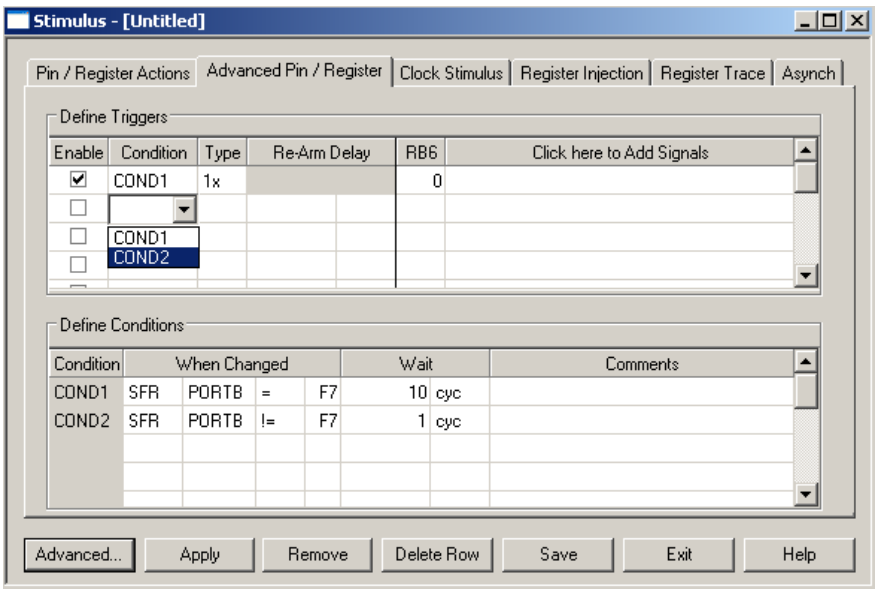

Рис. 18. Определение изменения по состоянию COND2.

В столбце Type выбираем Continuous (рис. 19). В поле Re-Arm Delay вписываем 0, в следующем столбце выбираем циклы. Затем в столбце RB6 вписываем значение единица – файл стимулов настроен для симуляции программы (рис. 20).

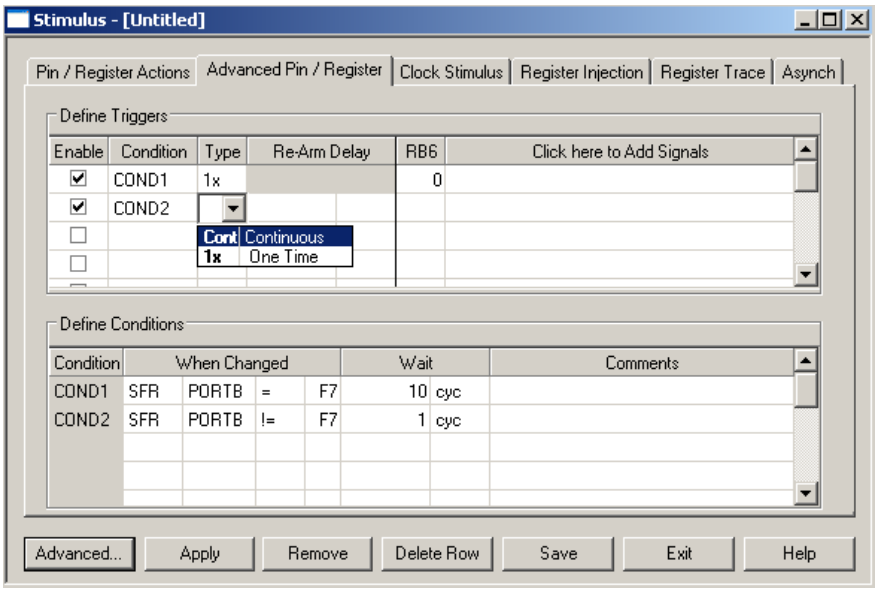

Рис. 19. Выбор типа стимула по условию COND2.

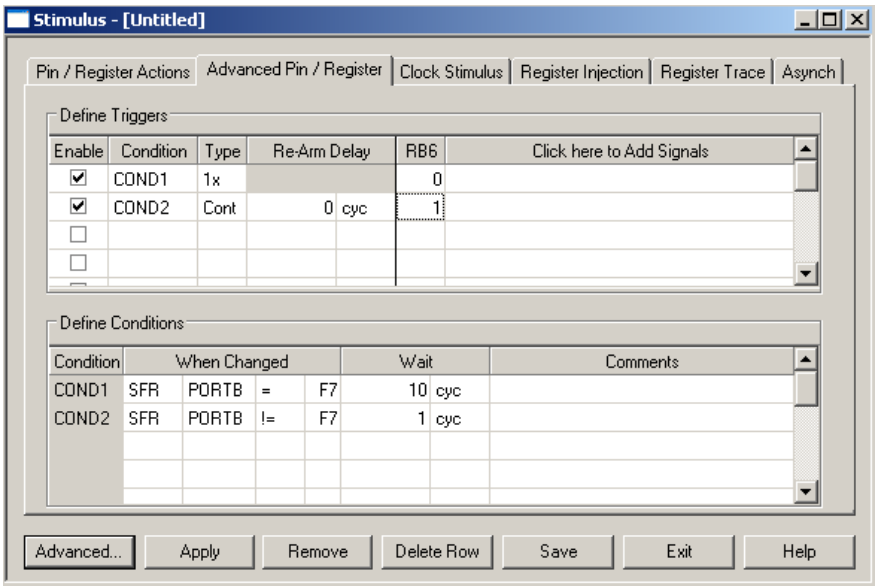

Рис. 20. Вид настроенного файла стимулов.

Теперь настроим окно Watch (View/Watch). Для наблюдения содержимого регистров. Выберем в выпадающем списке PORTB и нажмем Add SFR. Аналогичным образом добавим PORTD (рис. 21).

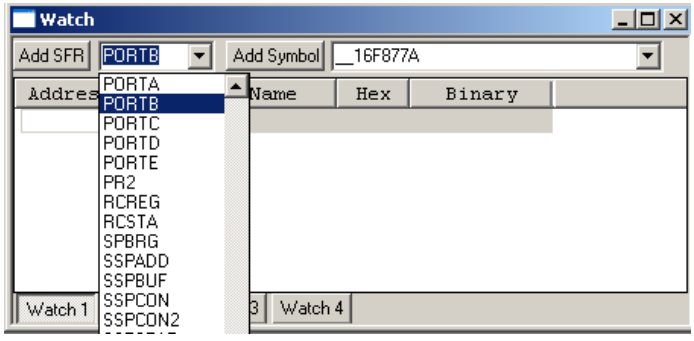

Рис. 21. Настройка окна Watch.

Теперь проведём симуляцию с применением стимулов. Выполнение программы проводим в пошаговом режиме (рис. 22). Вместо кнопки на мониторе можно использовать F7 на клавиатуре.

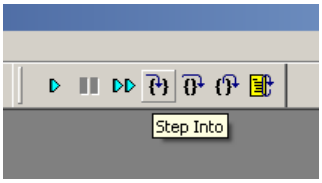

Рис. 22. Кнопка «один шаг».

Нажатие кнопки SA7 на микроконтроллере должно привести к появлению на семисегментном индикаторе числа 8. Или, что то же самое, на PORTD выводится число 0b’0111 1111’.

Выполним первую команду, нажав Step Into или F7 на клавиатуре. В окне Watch PORTB и PORTD будут равны нулю. Чтобы установить единицы на <RB7:RB5> в окне Stimulus выбираем вкладку Asinch и нажимаем кнопки Fire напротив RB7, RB6,RB5 (рис. 23). В окне Output должны появиться сообщения о применении асинхронных стимулов (рис. 24).

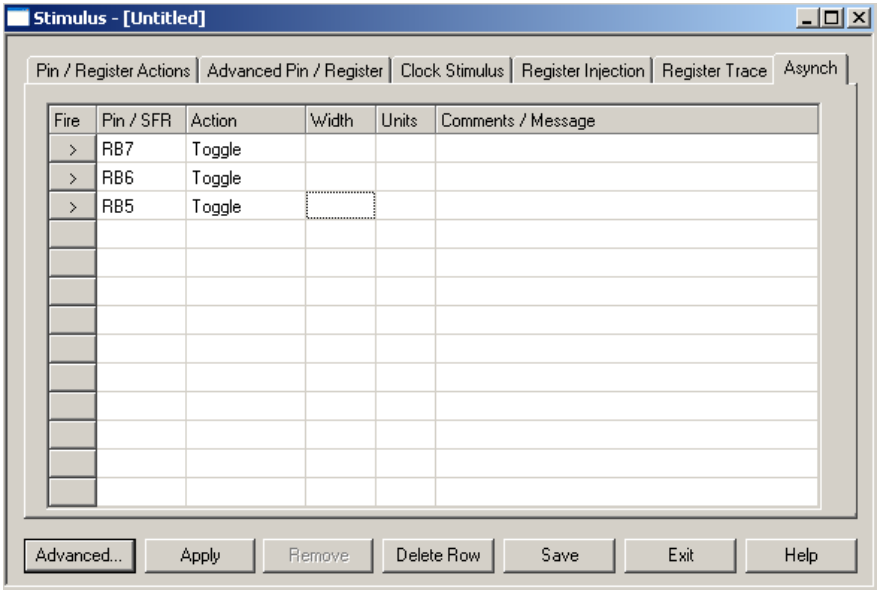

Рис. 23. Кнопки Fire вкладки Asynch.

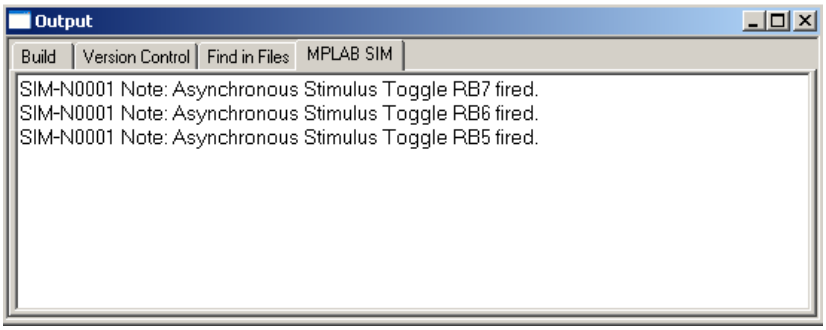

Рис. 24. Сообщения о применении асинхронных стимулов.

Теперь в окне Stimulus выберем вкладку Advanced Pin/Registers и нажмём кнопку Apply. В окне Output появится соответствующее сообщение (рис. 25).

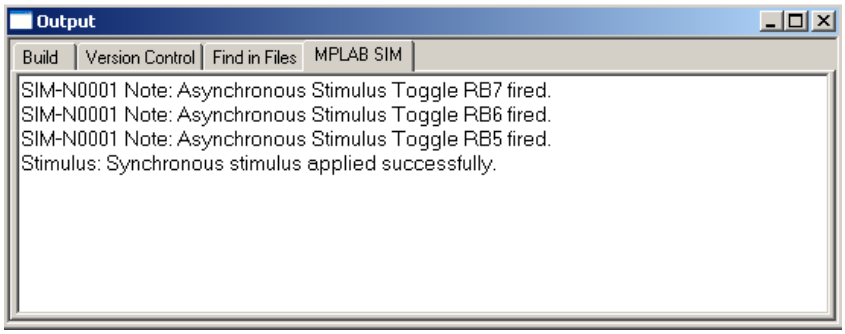

Рис. 25. Сообщение о применении синхронного стимула.

Сделаем ещё шаг, нажав F7, и убедимся, что три старшие бита PORTB установлены в единицы (рис. 26).

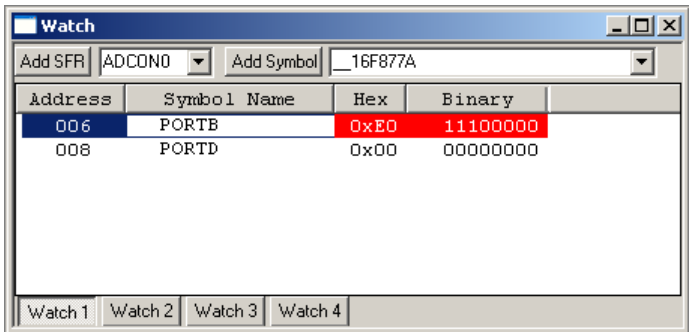

Рис. 26. Старшие биты PORTB установлены в единичное состояние.

Будем нажимать Step Into до тех пор, пока в окне Watch не появиться значение PORTB = F7. Как только появилось это значение, начнётся отсчёт десяти машинных циклов (мы их задали в области Define Condition в первой графе поля Wait), после чего будет применён стимул RB6 = 0. Но так как некоторые команды выполняются за два машинных цикла, до применения стимула на кнопку F7 придётся нажать меньше десяти раз (семь раз). И когда выполнение команды подойдёт к строке BTFSS PORTB,6 значение RB6 станет нулевым (рис. 27).

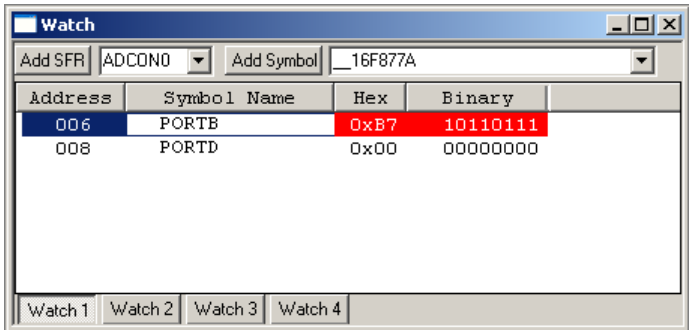

Рис. 27. Стимул применён: RB6 = 0.

Команда BTFSS PORTB,6 зафиксирует нулевое значение RB6. По нажатию F7 выполнение программы продолжится на следующей команде GOTO FIND_SW1 (рис. 28). И в этот же момент RB6 станет равным единице (рис. 29). Этот параметр 0 циклов (0 cyc) мы определили для COND2 в области Define Triggers в поле Re-Arm Delay.

```
15      LOOP_SCAN
16          BTFSS       PORTB,5     ; Read column 1
17          GOTO        FIND_SW0
18          BTFSS       PORTB,6     ; Read column 2
19          GOTO        FIND_SW1
20          BTFSS       PORTB,7     ; Read column 3
21          GOTO        FIND_SW2
22      NEXT_ROW
```

Рис. 28. Выполнение программы после применения стимула.

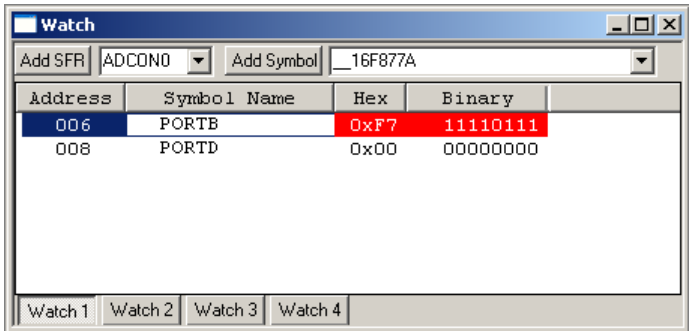

Рис. 29. В случае PORTB != F7 на RB 6 сразу же установится единица.

Затем программа изменит регистр COUNTER, произведёт табличное преобразование и в PORTD попадёт значение b’0111 1111’ (рис. 30).

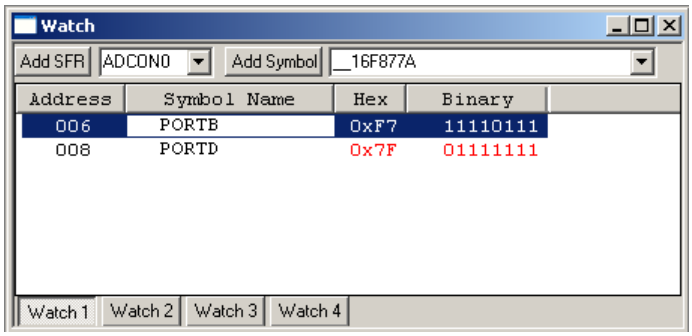

Рис. 30. Результат работы программы: кнопка нажата – PORTD = 7F.

После этого программа будет выполняться в бесконечном цикле, изменений на PORTD происходить не будет. Это связано с тем, что было выбрано однократное применение стимула COND1.

После проведения симуляции можно утверждать, что программа, скорее всего, отреагирует на нажатие кнопки SA7 так, как было задумано и описано в алгоритме. В идеале нужно проверить симуляцию всех кнопок. Для моделирования нажатия других кнопок нужно составить стимулы для каждой кнопки в соответствии с аппаратным обеспечением. И проверить работу программы в симуляторе с разными стимулами.

После компиляции и симуляции соберите схему рис. 35 на лабораторном макете как показано на рис. 36. После этого в среде разработки выберите имеющийся у вас программатор, например PICkit 2 (рис. 31).

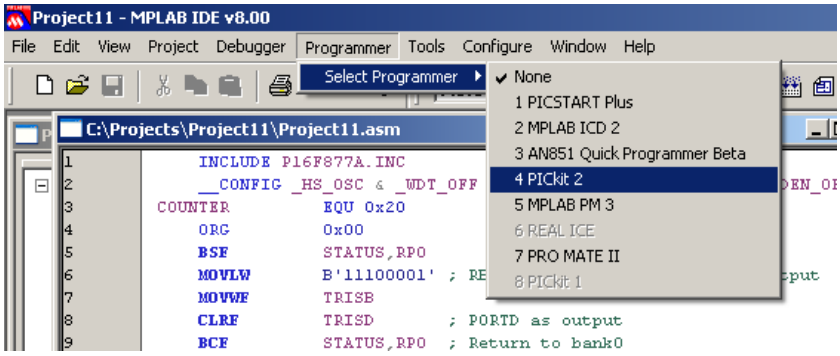

Рис. 31. Выбор программатора.

Запрограммируйте лабораторный макет (рис. 32). Затем выключите питание, отсоедините программатор, переведите макет в режим RUN (рис. 33), подайте питание – программа начнёт выполняться. Проверьте работу всех кнопок клавиатуры.

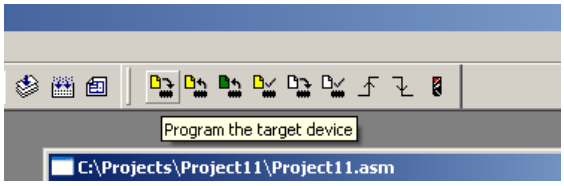

Рис. 32. Кнопка для программирования лабораторного макета.

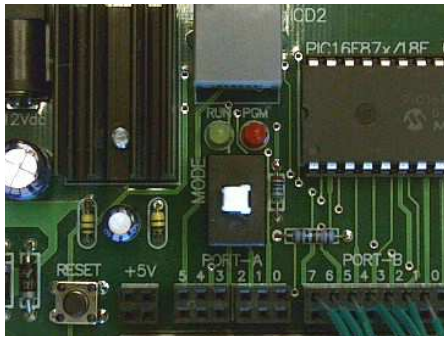

Рис. 33. Переключатель MODE RUN/PGM на плате макета.

Аппаратное обеспечение

Соединения PORTB микроконтроллера и клавиатуры перечислены в таблице 1. Соединения PORTD и семисегментного индикатора приведены в таблице 2. Контакт порта соединяют проводом с контактом разъёма, который указан в той же строке, но другого столбца таблицы.

Табл. 1. Соединения PORTB и клавиатуры.

Контакт разъёма PORTB	Контакт разъёма клавиатуры
PORTB,1	R1
PORTB,2	R2
PORTB,3	R3
PORTB,4	R4
PORTB,5	C1
PORTB,6	C2
PORTB,7	C3

Табл. 2. Соединения PORTD и семисегментного индикатора.

Контакт разъёма PORTD	Контакт разъёма индикатора
PORTD,0	a
PORTD,1	b
PORTD,2	c
PORTD,3	d
PORTD,4	e
PORTD,5	f
PORTD,6	g
DIGIT1 соединяют с разъёмом GND	

Запускают и проверяют программу после программирования макета. Выводы RB6 и RB7 используются и для программирования, и для клавиатуры. Поэтому переключатель MODE, расположенный рядом с разъёмом программатора, должен быть в положении RUN (горит зелёный светодиод).

На рис. 36 показаны соединения на плате лабораторного макета.

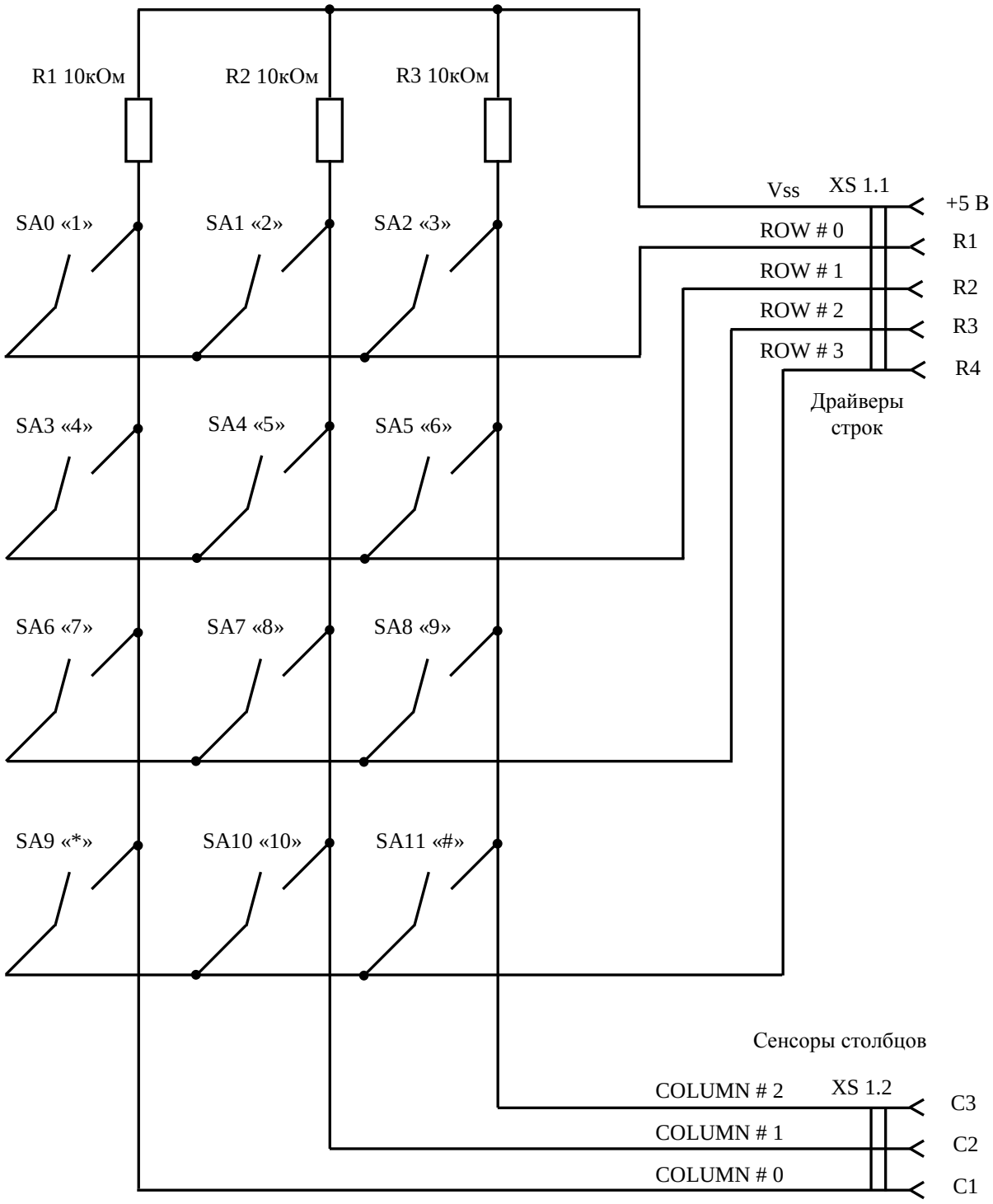

Рис. 34. Клавиатура. Схема электрическая принципиальная.

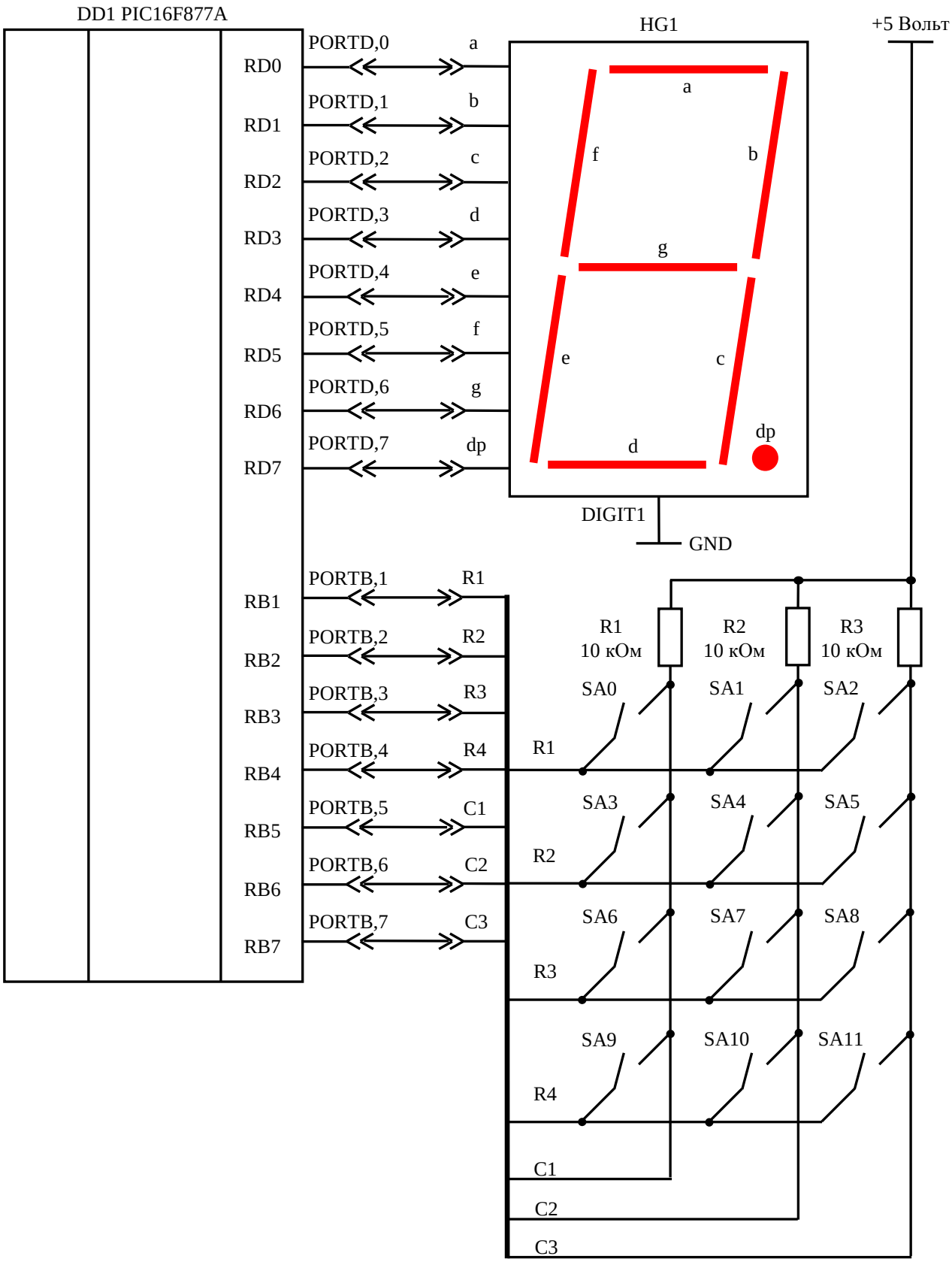

Рис. 35. Схема электрическая принципиальная для выполнения работы.

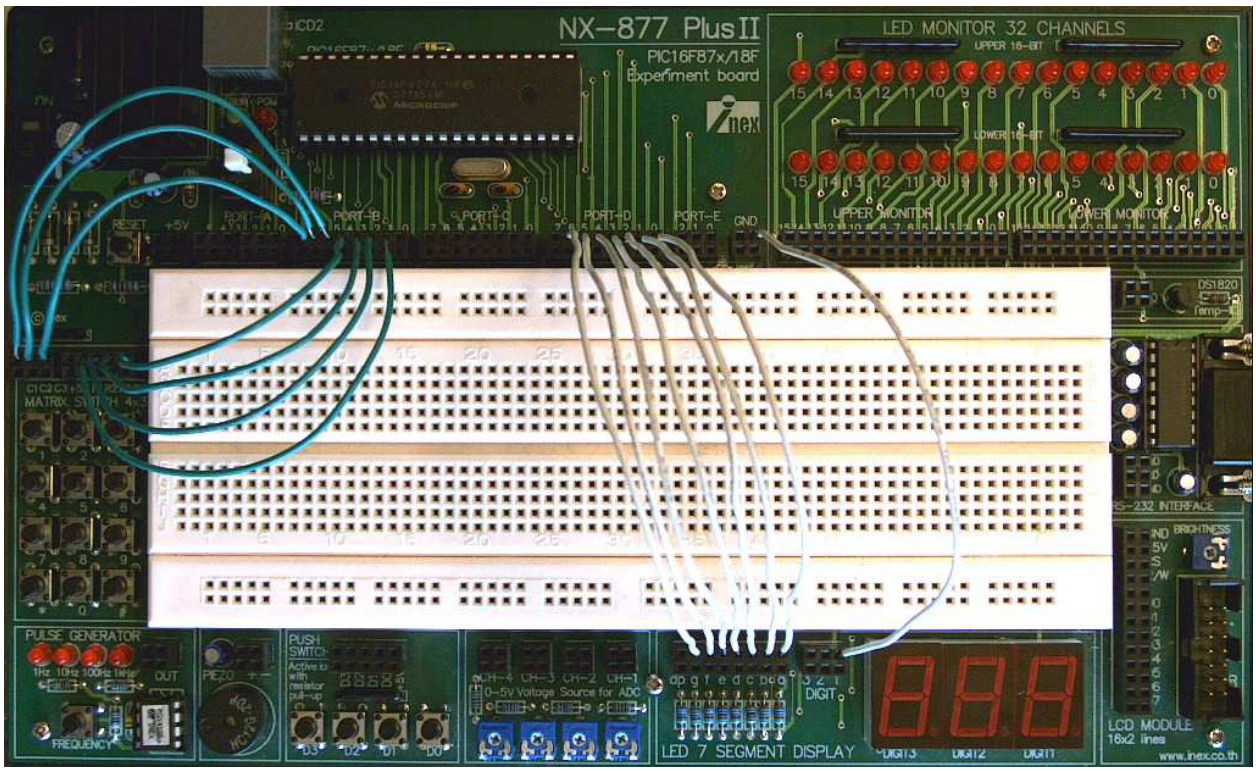

Рис. 36. Схема, собранная на лабораторном макете.

Алгоритм программы изображён на рис. 37.

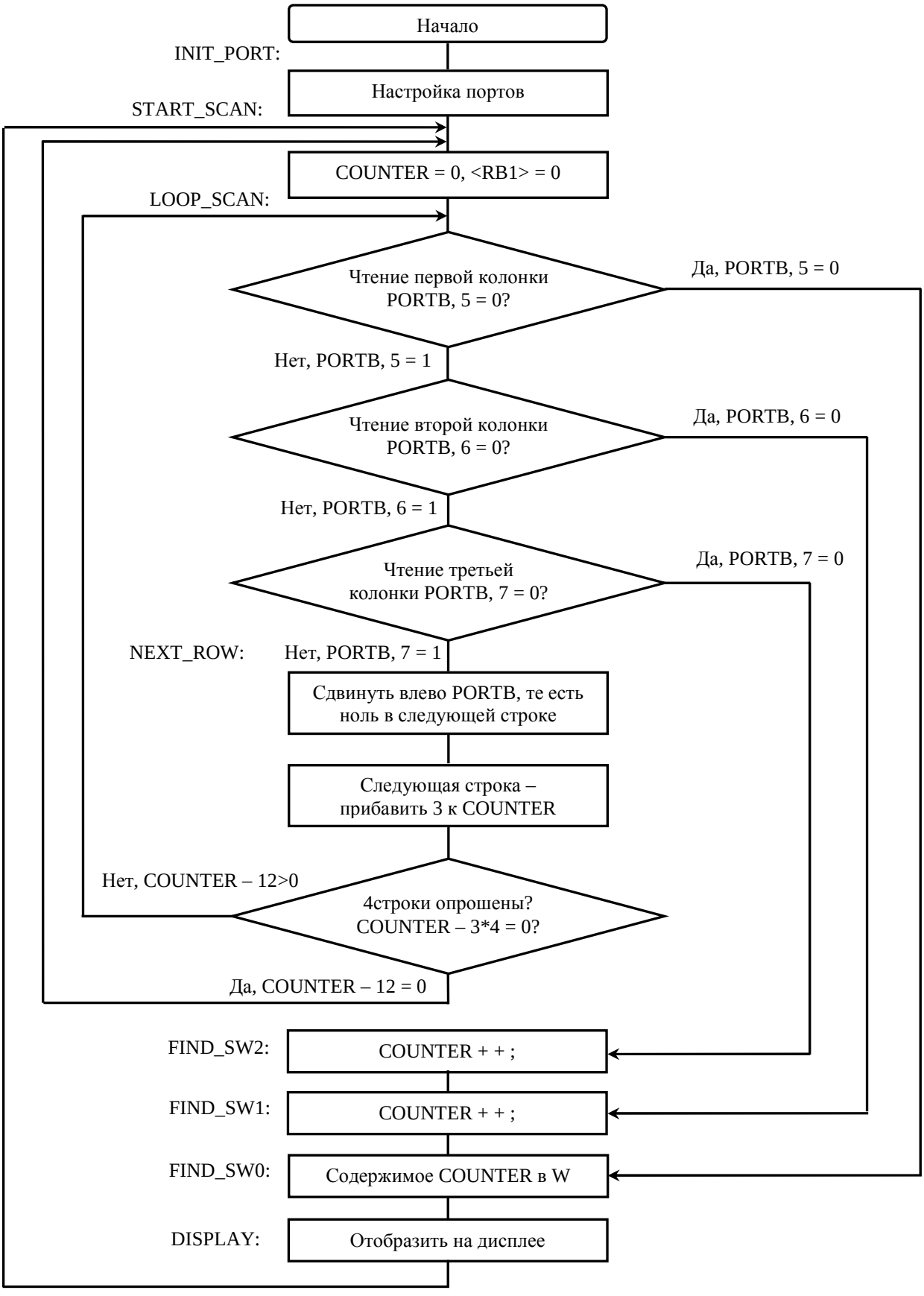

Рис. 37. Алгоритм программы Project11.

Текст файла Project4.c

```
#include <htc.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
#define STAR    0x76
#define SHARP   0xEE
__CONFIG (HS & WDTDIS & BORDIS & UNPROTECT & LVPDIS & DEBUGEN);
unsigned char BUTTON = 0;
unsigned char Klav (void)
{
   unsigned char Key = 0;

   TRISB = 0b11100001;

   PORTB = 0b11111101;//ROW # 1 "1,2,3"
   if (0 == RB5)          Key = 1;//COLUMN 1
   else if (0 == RB6)     Key = 2;//COLUMN 2
   else if (0 == RB7)     Key = 3;//COLUMN 3

   PORTB = 0b11111011;//ROW # 2 "4,5,6"
   if (0 == RB5)          Key = 4;//COLUMN 1
   else if (0 == RB6)     Key = 5;//COLUMN 2
   else if (0 == RB7)     Key = 6;//COLUMN 3

   PORTB = 0b11110111;//ROW # 3 "7,8,9"
   if (0 == RB5)          Key = 7;//COLUMN 1
   else if (0 == RB6)     Key = 8;//COLUMN 2
   else if (0 == RB7)     Key = 9;//COLUMN 3

   PORTB = 0b11101111;//ROW # 4 "*,0,#"
   if (0 == RB5)          Key = 10;//COLUMN 1
   else if (0 == RB6)     Key = 11;//COLUMN 2
   else if (0 == RB7)     Key = 12;//COLUMN 3

   return Key;
}
void main (void)
{
                          TRISD = 0x00;
                          PORTD = 0x00;
while (1)
   {
   BUTTON = Klav();
   switch (BUTTON)
```

```
  {
    case   0:  PORTD = 0x00; break;
    case   1:  PORTD = ONE; break;
    case   2:  PORTD = TWO; break;
    case   3:  PORTD = THREE; break;
    case   4:  PORTD = FOUR; break;
    case   5:  PORTD = FIVE; break;
    case   6:  PORTD = SIX; break;
    case   7:  PORTD = SEVEN; break;
    case   8:  PORTD = EIGHT; break;
    case   9:  PORTD = NINE; break;
    case  10:  PORTD = STAR; break;
    case  11:  PORTD = ZERO; break;
    case  12:  PORTD = SHARP; break;
  }
  }

}
```

Индивидуальные задания

Напишите программу, которая по нажатию чётных клавиш будет выводить соответствующие цифры на одном индикаторе. А по нажатию нечётных клавиш будет выводить соответствующие цифры на другом индикаторе. Используйте динамическую индикацию.

Контрольные вопросы

1. Какой порт PIC16F877A рекомендуется использовать для клавиатуры?
2. Что такое драйверы строк?
3. Что такое сенсоры столбцов?
4. На одном из столбцов обнаружен ноль – кнопка нажата или отпущена?
5. Что такое polling?
6. Как можно реализовать операцию умножения?
7. Почему работу программы можно проверить только в RUN режиме работы макета?
8. Как промоделировать нажатие других кнопок?

Оглавление:

Список рисунков: