В.Н. Гололобов

Первая схема...

...с микроконтроллером

Оглавление

Предисловие

Не могли радиолюбители пройти мимо такого удобного и интересного компонента электрической схемы, как микроконтроллер. Не могли.

Однако начинающему радиолюбителю следует знать, что для работы с микроконтроллером ему предстоит выбрать не только сам микроконтроллер, благо их моделей в продаже много, но и среду разработки программы. Микросхема контроллера без загруженной в неё программы мертва. Для загрузки написанной программы потребуется программатор – устройство, которое считывает файл с программой и переносит его в память микроконтроллера.

Прежде, чем сделать свой выбор, потратьте немного времени за компьютером – это поможет вам сделать свой выбор.

Глава 1. Выбор микроконтроллера

В профессиональной практике выбор микроконтроллера определяется в первую очередь не «крутизной» модели, советы подобного рода можно встретить на форумах, а поставленной перед разработчиком задачей. Задача определяет ряд параметров, по которым можно начать выбор. Затем разработчик смотрит на такие аспекты, как доступность модели, возможный срок жизни (не снимут ли модель с производства в ближайшие годы), цена и качество модели в тех условиях, в которых устройство будет эксплуатироваться.

Для начинающего радиолюбителя важнейшими параметрами могут стать цена и доступность микросхемы, стоимость и доступность программатора. При всей универсальности программатора он не программирует все модели микроконтроллеров. И, главное, для начинающего работать с микроконтроллером следует определиться с тем, как он будет создавать код программы.

Для написания кода программы достаточно блокнота из состава Windows любой версии (или текстового редактора Linux). Но это для опытных программистов и простой программы.

Есть много сред разработки программ для микроконтроллеров, но, как и программаторы, эти удобные редакторы зависят от выбранной модели микроконтроллера. При выборе среды разработки следует обратить внимание и на такой атрибут программы, как отладчик. Хотя некоторые программаторы имеют встроенную систему отладки. Но начать всё-таки лучше со среды разработки и её отладчика.

Глава 2. Выбор среды разработки программы

Почти все производители микроконтроллеров предлагают бесплатные среды разработки на языке ассемблер для производимых ими моделей. Многие производители предоставляют в пользование и компиляторы с языка Си, которые работают в выпускаемых ими средах разработки.

Не только ассемблер и язык Си, но и другие языки высокого уровня используются при создании программ для микроконтроллеров. Есть и современные среды разработки, использующие графический язык программирования.

Таким образом, перед начинающим серьёзная проблема, какой язык программирования освоить.

Вместе с тем, если начинающий любитель не имел опыта программирования, то ему следует подумать не о выборе языка программирования, а выборе подходящего учебника по программированию. Есть учебники по программированию на языках низкого и высокого уровня, есть учебники и книги, описывающие программирование микроконтроллеров. Но…

Учебники по программированию, как правило, ориентируются на создание прикладных программ для компьютеров. А учебники по программированию микроконтроллеров много внимания уделяют описанию самих микроконтроллеров. И то, и другое очень важно и полезно, но тогда, когда начинающий, сделав первые шаги в работе с микроконтроллерами, решит продолжить изучение предмета.

Поэтому, это только моё мнение, начинающему следует в первую очередь понять, что любая программа, какой бы язык вы не выбрали, описывает ряд операций, осуществляемых процессором микроконтроллера, по опросу входов микроконтроллера и принятию решения о состоянии выходов микроконтроллера на основании состояния его входов. В подавляющем большинстве схем входы принимают два значения: значения высокого и низкого уровня сигнала. Это состояние, считанное на входе, может подвергаться логической или арифметической обработке, чем занимается процессор, а результат, за некоторыми исключениями, выводится на выходы, которые тоже принимают два значения: высокого и низкого уровня.

Если вы впервые решили программировать микроконтроллер, то выберите самую простую программу, которая вам понятна без дополнительных пояснений. Например, самой простой программой (мне нравится это встреченное где-то название) будет «Hello, world» для микроконтроллера. Ещё её называют «помигать светодиодом». Пусть вас не смущает, что только помигать светодиодом. Выводя данные через встроенный модуль USART, используя протокол, скажем, One-Wire или USB, вы будете, пусть сложным образом, но не более, чем «мигать светодиодом».

Глава 3. Выбор своей первой программы

Если вы согласны со мной в вышенаписанном, то давайте мигать светодиодом. Если нет, то выбрасывайте этот рассказ. Он вас больше разочарует, чем поможет в ваших изысканиях.

Конечно, начать можно со сложного, продвигаясь к простому – максимум усилий в начале, минимум, когда появилась усталость. Но всё-таки начнём с простого, продвигаясь к более сложному.

Как можно описать мигание светодиода?

Мы включаем микроконтроллер, подав на него питающее напряжение, светодиод, который подключен к одному из выводов через резистор, должен загореться. После небольшой, скажем в секунду, паузы он должен погаснуть. И после такой же паузы загореться. Давайте на этом пока остановимся.

Итак. Чтобы светодиод загорелся, вывод микроконтроллера должен быть направлен на выход. И на этом выводе следует установить высокий уровень (уровень логической единицы). Чтобы светодиод погас, на выводе следует установить низкий уровень (уровень логического нуля).

Вот основные операции, которые мы должны осуществить. Правда, есть ещё операция, которую мы определили как паузу.

Пауза – это такое состояние микроконтроллера, когда он, как бы, замыкается в себе, поглядывая на свои собственные часы и ожидая, когда выйдет время, и ему разрешат продолжить другие операции. С паузой мы разберёмся немного позже. А сейчас начнём с простого, используя простой для понимания графический язык. Я воспользуюсь демо-версией программы Flowcode 6.

Как и другие демо-версии, она ограничена – нельзя получить работающий код программы для загрузки через программатор в микросхему. Но к этой проблеме мы вернёмся позже. Кстати, скачать пробную версию, которая полностью функциональна в течение 30 дней, можно сайте производителя: http://www.matrixmultimedia.com/flowcode.php.

Чтобы установить высокий уровень на выводе достаточно такой программы:

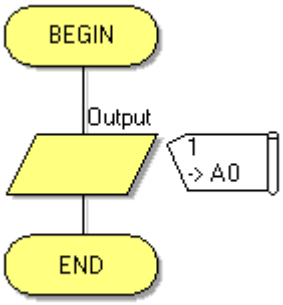

Рис. 1. Программа установки высокого уровня на выводе A0

Как многие программы, наша программа начинается вводным словом BEGIN, а заканчивается END. Сама программа описывается как 1 -> A0 для выхода (Output). Отладчик программы Flowcode позволяет проверить работу программы, но на системную панель следует добавить светодиод и подключить его (в панели свойств) к выводу A0.

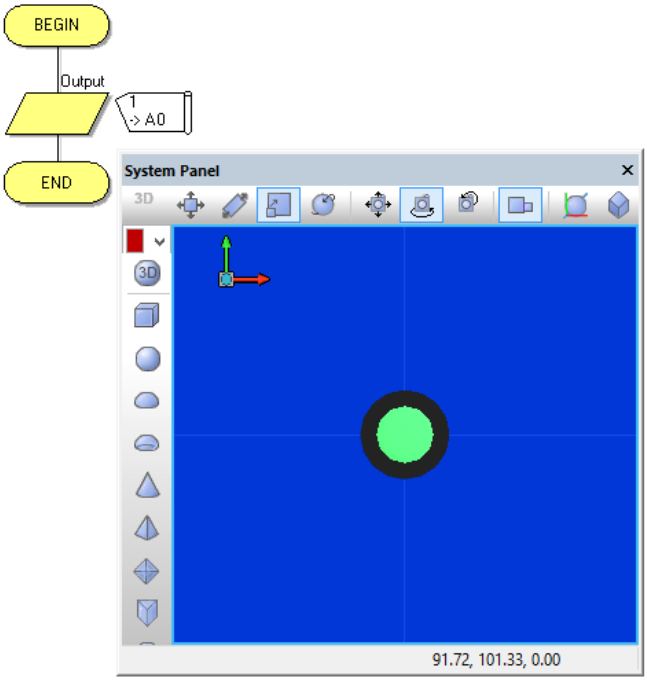

Рис. 2. Проверка работы программы

Глава 4. Завершение создания первой программы

Используя графический язык, завершим создание первой программы.

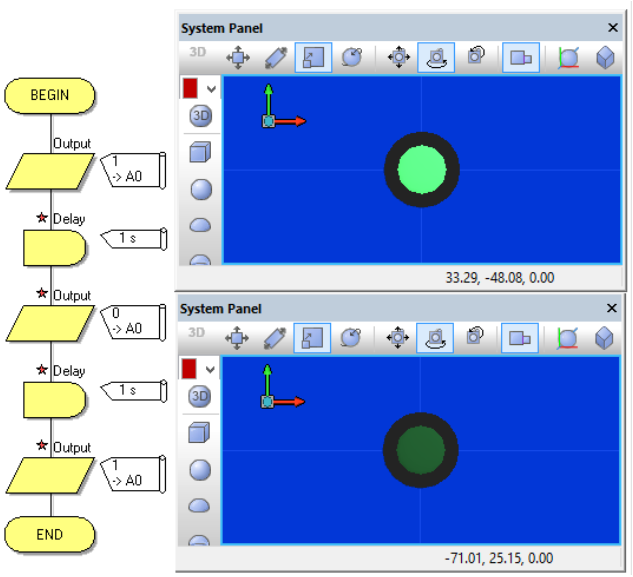

Рис. 3. Полная версия первой программы

Вся программа буквально повторяет то, что записано выше на обычном языке: подать на вывод высокий уровень, сделать паузу в секунду, подать низкий уровень, сделать паузу в секунду, подать высокий уровень.

Выбрав то, как будет организован тактовый генератор, а любому процессору нужен тактовый генератор, для выбранной модели микроконтроллера, мы получили бы готовую программу.

Выбор организации работы тактового генератора и некоторые другие параметры микроконтроллера записываются в слове конфигурации. Но об этом позже.

Если бы версия Flowcode 6 была рабочая, то, проверив работу программы, её можно было бы откомпилировать и сразу загрузить в программатор, подключённый к компьютеру (для программатора, работающего с Flowcode). Осталось бы запаять микросхему (лучше использовать панельку, в которую вставить микросхему), подпаять питание и резистор со светодиодом, чтобы получить своё первое устройство, использующее микроконтроллер.

Глава 5. Первое усовершенствование первой программы

Ни один радиолюбитель не остановится на достигнутом. Мы тоже.

Мигнуть светодиодом можно и без контроллера – подключили к питанию, он зажёгся, подождали секунду, выключили питание. Просто и без программы.

Изменим программу, добавив цикл. Цикл – это повторяющаяся последовательность операций. Циклы бывают разные, часто цикл определяется некоторым условием: пока условие выполняется, цикл работает, операции повторяются; когда условие перестало выполняться, работа цикла прекращается и программа «выходит» из цикла. Таким образом, если задать условие, которое будет выполняться всегда, цикл будет работать всегда. Такой цикл называют бесконечным. И это обычный образ жизни микроконтроллера. После подачи питающего напряжения он начинает работать в бесконечном цикле, выполняя, порой, очень сложную программу.

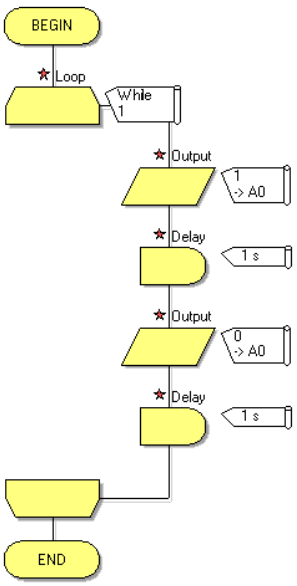

Рис. 4. Добавление бесконечного цикла

Цикл While отмечен условием в начале цикла, while 1, и будет выполняться всегда, поскольку единица условия не меняется в программе. А программа включает и выключает светодиод. И светодиод будет мигать, пока мы не выключим питание микроконтроллера.

Глава 6. Запись программы на языке Си

Графический язык программирования (повторюсь, это только моё мнение) достаточно удобен для начинающих осваивать микроконтроллер. Из бесплатных программ, работающих с микроконтроллером, я знаю ещё только KTechlab, работающую в Linux, но, скорее, работавшую. Программа KTechlab обучающая, позволят поработать и с аналоговыми, и с цифровыми устройствами, но создатель программы, когда операционная система сильно изменилась, перестал переделывать программу, и сегодня, насколько я знаю, с микроконтроллером программа не работает. Хотя можно использовать предыдущие версии программы и операционной системы.

Гораздо меньше вы будете зависеть от срока действия лицензии или от финансов, если научитесь программировать на языке высокого уровня. Есть множество сред разработки, бесплатных и удобных, использующих разные языки высокого уровня. Хотя любой язык программирования, как и модели микроконтроллеров, наилучшим образом подходит к поставленным задачам, наиболее часто используют язык Си для программирования микроконтроллеров.

Я начну с того, что приведу запись программы, которую мы создали ранее, на языке Си.

```
void main() {
      TRISA = 0xFE;
      while(1) {
            PORTA.F0 = 1;
            Delay_ms(1000);
            PORTA.F0 = 0;
            Delay_ms(1000);
      }
}
```

Запись программы на языке Си будет зависеть (хотя и незначительно) от компилятора, который использует среда разработки. Приведённая выше программа записана (откомпилирована и проверена) в среде MicroC Pro. Для другого компилятора она может иметь несколько иной вид:

```
#define _XTAL_FREQ 8000000

void main(void) {
   TRISA = 0xFE;
      while(1) {
            RA0 = 1;
            __delay_ms(1000);
            RA0 = 0;
            __delay_ms(1000);
      }
    return;
}
```

Так программа выглядит в среде разработки MPLABX.

Глава 7. От слов к делу

Я немного погорячился, определив называние главы, но тем не менее. Если вы решили овладеть основами программирования на языке Си, то вам пора определиться с двумя составляющими последующей работы: выбрать модель микроконтроллера и среду разработки.

Я не советчик в этих вопросах, поскольку во многом считаю делом вкуса для начинающих остановить свой выбор на микроконтроллерах PIC, AVR, ARM или «более крутых». Но я опишу работу с микроконтроллером PIC16F628A в среде MicroC Pro.

Почему эта модель микроконтроллера? Для начинающих важно иметь справку по микросхеме, а для этой модели datasheet есть на русском языке. У микроконтроллера есть встроенные дополнительные модули, освоить управление которыми и интересно, и полезно. Такие же модули вы встретите и в других моделях. Альтернативой, хотя не всем это нравится, я счёл бы модуль Arduino, который тоже хорошо оснащён всем, включая встроенный программатор.

Если вы только начинаете осваивать микроконтроллер, вам придётся потратить некоторое время на осуществление первых шагов к успеху, вам придётся потратить некоторое количество денег, поскольку не только за компьютером, хотя дочитать книгу можно и за компьютером, но и на макетной плате что-то полезно попробовать. Выбор микроконтроллера определит для вас и выбор программатора, и выбор программы, с которой вы будете работать.

Итак. Программа MicroC Pro бесплатная, но ограничивает вашу программу объёмом в 2 КБайта. Много это или мало? Ни то, ни другое – вы можете создать много интересных и полезных устройств, не переходя предел ограничения.

Глава 8. Немного о программе MicroC Pro

Программа доступна на сайте производителя: http://www.mikroe.com/mikroc/pic/

У каждого могут быть свои предпочтения, но, рассказывая о роботе РОВОРІСА, я использовал эту программу, она пришла вместе с роботом. Если программу можно использовать в работе с роботом, пусть и очень простым, она вполне подойдёт для начинающего осваивать язык Си.

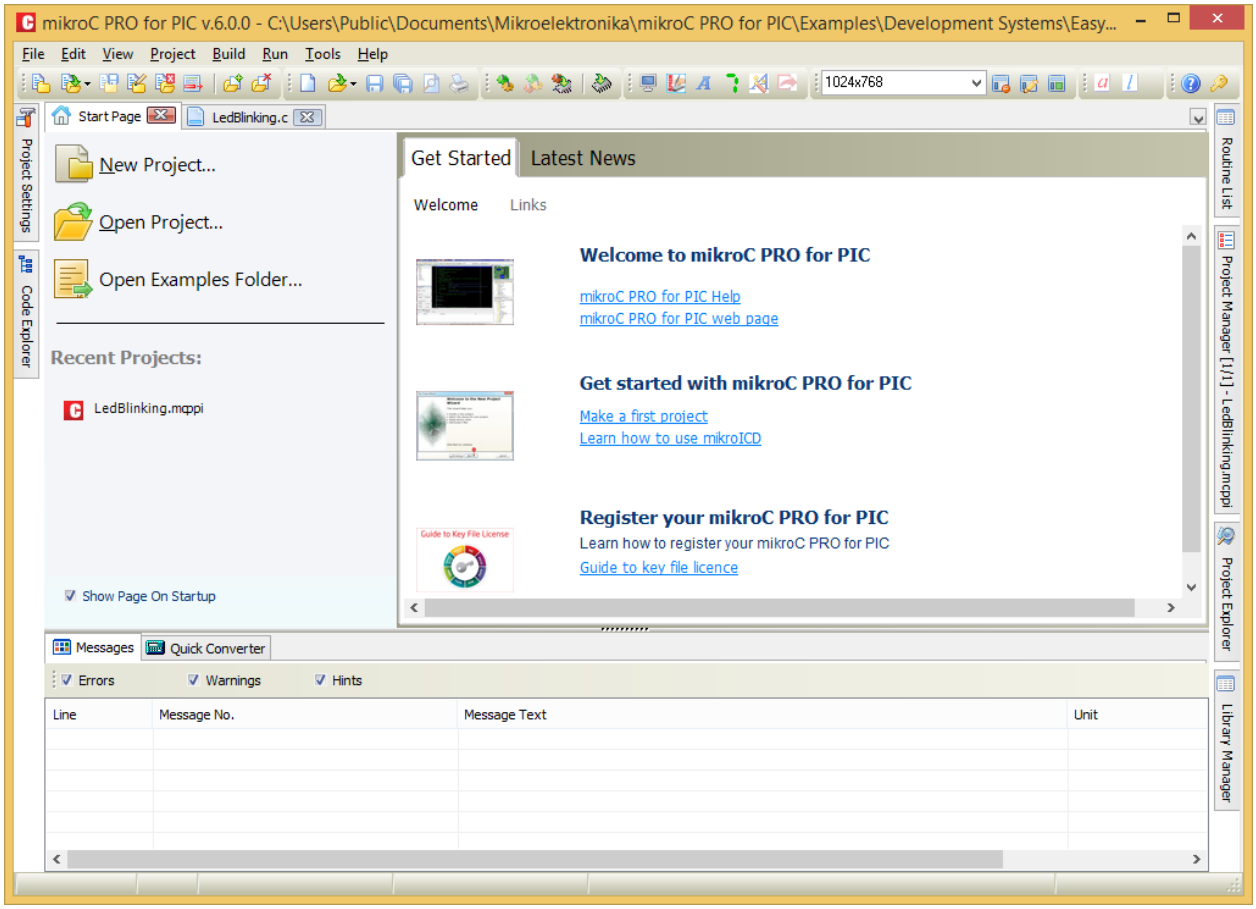

Рис. 5. Запуск программы MicroC Pro

Само написание кода программы – это чисто технический аспект работы, для этого достаточно блокнота. Но создание программ, прикладных ли для компьютера, рабочих для электронных устройств, задача более сложная, чем запись кода программы.

Я уже говорил, что к вопросу о паузе в нашей первой программе мы вернёмся позже. Вот и вернулись. Дело в том, что у процессора нет такой команды, такой операции как пауза в 1 секунду. Эту часть программы вам надо создавать самостоятельно. Если…

Если нет возможности воспользоваться готовыми функциями среды разработки, которые написаны профессионалами для вас. Программа MicroC Pro имеет обширную библиотеку (Library Manager), позволяющую вам без проблем работать с математическими функциями, разными протоколами обмена данными и дополнительными устройствами, как, например, ЖКИ.

Кроме текста программы на языке Си в подсказке есть и пример подключения ЖКИ к микроконтроллеру. Таким образом, если у вас другой микроконтроллер, то вы можете исправить текст программы сообразно с изменениями на физическом уровне. На картинке ниже не поместится вся программа работы с ЖКИ и схема подключения, но вы сами можете посмотреть всё, если захотите использовать программу MicroC Pro.

```
  Delay_ms(2000);

  // Moving text
  for(i=0; i<4; i++) {
    Lcd_Cmd(_LCD_SHIFT_RIGHT);
    Move_Delay();
  }

  while(1) {
    for(i=0; i<8; i++) {
      Lcd_Cmd(_LCD_SHIFT_LEFT);
      Move_Delay();
    }

    for(i=0; i<8; i++) {
      Lcd_Cmd(_LCD_SHIFT_RIGHT);
      Move_Delay();
    }
  }
}
```

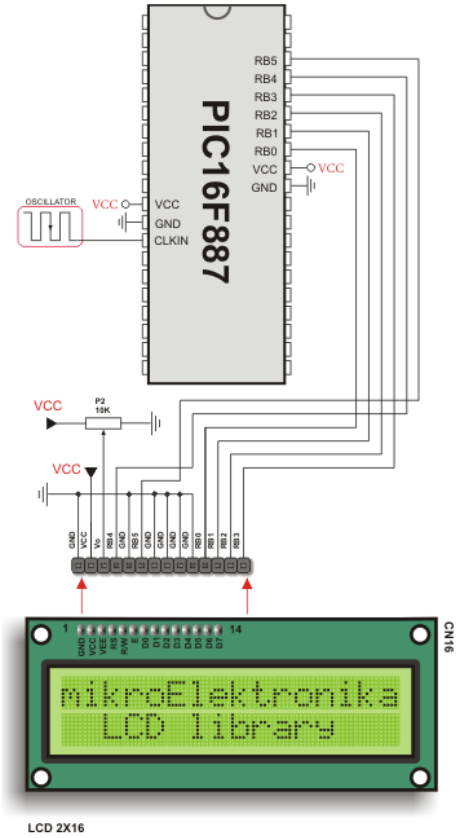

Рис. 6. Пример работы с ЖКИ из Library Manager

Глава 9. По лестнице вниз

Кроме языков высокого уровня можно использовать ассемблер. Не раскрою секрета, если скажу, что процессор микроконтроллера не понимает не только графический язык, но и язык Си. И ассемблер он тоже не понимает. Поэтому ваш проект будет транслироваться с языка высокого уровня на язык более низкого уровня, а окончательный вариант будет записан в числах. Если вы откроете любой hex-файл, то увидите нечто похожее на это:

```
:020000001628C0
:0E000600831203132108 8A00200882000800DC
:1000140003208A110A128000840AA00A0319A10A83
:08002400F003031D0A28080087
:10002C00FE308316031385008312051 40B30FB007E
:10003C002630FC005D30FD00FD0B2228FC0B222835
:10004C00FB0B22280000000005100B30FB002630B3
:10005C00FC005D30FD00FD0B3128FC0B3128FB0B47
:0A006C003128000000001A283A288D
:02400E00182177
:00000001FF
```

Это шестнадцатеричная запись двоичных чисел команд, которые только и понимает процессор, вместе с командами программатору. Это почти самый низкий уровень программы, ниже только двоичная запись команд. С такой программой трудно работать, поэтому программисты придумали язык ассемблер. На языке ассемблера команды записываются с помощью

аббревиатуры, которую легче запомнить, чем двоичные или шестнадцатеричные числа. Вот пример записи программы на ассемблере.

```
_main:

;tut1.c,1 ::           void main() {
;tut1.c,2 ::           TRISA = 0xFE;
      MOVLW      254
      MOVWF      TRISA+0
;tut1.c,3 ::           while(1) {
L_main0:
;tut1.c,4 ::           PORTA.F0 = 1;
      BSF        PORTA+0, 0
;tut1.c,5 ::           Delay_ms(1000);
      MOVLW      11
      MOVWF      R11+0
      MOVLW      38
      MOVWF      R12+0
      MOVLW      93
      MOVWF      R13+0
L_main2:
      DECFSZ     R13+0, 1
      GOTO       L_main2
      DECFSZ     R12+0, 1
      GOTO       L_main2
      DECFSZ     R11+0, 1
      GOTO       L_main2
      NOP
      NOP
;tut1.c,6 ::           PORTA.F0 = 0;
      BCF        PORTA+0, 0
;tut1.c,7 ::           Delay_ms(1000);
      MOVLW      11
      MOVWF      R11+0
      MOVLW      38
      MOVWF      R12+0
      MOVLW      93
      MOVWF      R13+0
L_main3:
      DECFSZ     R13+0, 1
      GOTO       L_main3
      DECFSZ     R12+0, 1
      GOTO       L_main3
      DECFSZ     R11+0, 1
      GOTO       L_main3
      NOP
      NOP
;tut1.c,8 ::           }
      GOTO       L_main0
;tut1.c,9 ::           }
L_end_main:
      GOTO       $+0
; end of _main
```

Запись программы сделана при трансляции кода на языке Си на ассемблер в среде MicroC Pro. Если вы самостоятельно будете писать эту программу на ассемблере, то вам придётся приложить больше усилий. Но познакомиться с ассемблером очень необходимо, даже если вы не намерены кодировать программы на ассемблере.

Глава 10. Почему «помигать светодиодом» – это первая программа?

Я уже сказал выше, что большая часть работы с микроконтроллером – это помигать светодиодом.

Давайте посмотрим, как можно написать на языке Си что-то, что позволит без встроенного модуля USART отправить команду другому микроконтроллеру или компьютеру по протоколу RS232. Правда, для работы с COM-портом компьютера понадобится микросхема интерфейса RS232. Но это уже не относится к программе.

Начнём с того, что посмотрим, как выглядит отправка байта 10101010 по протоколу RS232 на выходе микроконтроллера при наличии встроенного модуля USART. Есть несколько путей, чтобы это выполнить. На физическом уровне нам потребуется запрограммировать микроконтроллер, подключить осциллограф, подать питающее напряжение и посмотреть на экран осциллографа.

Есть физический наполовину путь – использовать возможности программы Flowcode 6 работать с COM-портом компьютера, а к порту компьютера подключить осциллограф.

И, наконец, можно использовать программу ISIS (Proteus), которая очень удобна для завершения отладки схем с микроконтроллером. На этом пути всё получится в виртуальном пространстве компьютера. Написать программу можно на любом языке, важно получить hex-файл программы.

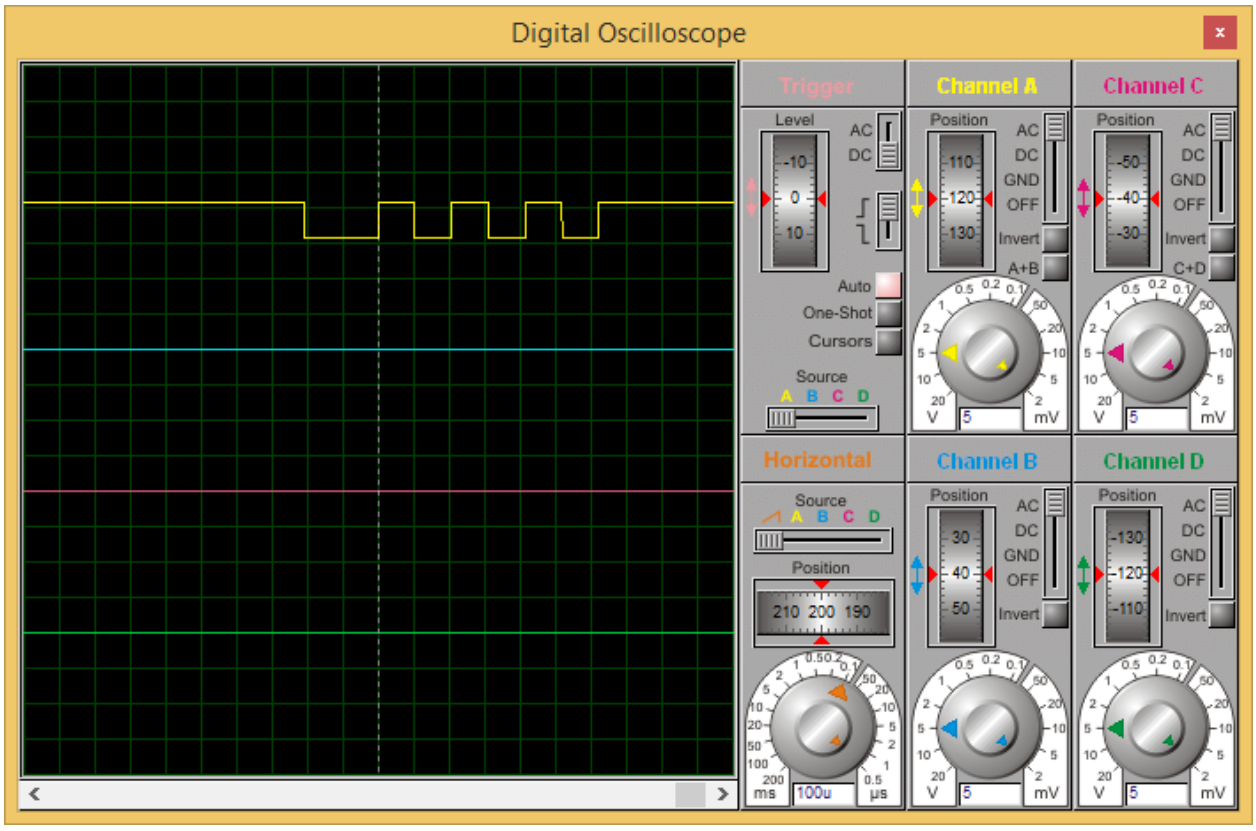

Рис. 7. Отправка символа через модуль USART

Передаваемый сигнал выглядит, что бы ни говорили, как: низкий уровень, пауза, высокий уровень, пауза и т.д. Не более чем «помигать светодиодом».

В отсутствии передачи на выводе поддерживается высокий уровень. Первый сигнал – это стартовый бит низкого уровня. Передаваемый по RS232 байт отправляется младшим битом вперёд, в нашем случае нулём. То есть, ноль – это низкий уровень, единица – высокий уровень.

Каждый импульс продолжается 100 мкс. Длительность импульсов (паузы) определяется скоростью передачи, 9600 на рисунке, и зависит от тактовой частоты, в данном случае 4 МГц.

Давайте помигаем светодиодом так, чтобы передать символ «1». Шестнадцатеричное число символа 31, а двоичное 00110001. Программа для передачи этого байта выглядит так: установить низкий уровень, пауза 100 мкс, установить высокий уровень, пауза 100 мкс, установить низкий уровень и т.д. У нас есть первая программа для среды программирования MicroC Pro. Давайте уподобимся опытным программистам и переделаем её для нашей текущей цели.

```
void main() {
      TRISA = 0xFE;
      PORTA.F0 = 1;
      while(1) {
            PORTA.F0 = 0;
            Delay_us(100);
            PORTA.F0 = 1;
            Delay_us(100);
            PORTA.F0 = 0;
            Delay_us(100);
            PORTA.F0 = 0;
            Delay_us(100);
            PORTA.F0 = 0;
            Delay_us(100);
            PORTA.F0 = 1;
            Delay_us(100);
            PORTA.F0 = 1;
            Delay_us(100);
            PORTA.F0 = 0;
            Delay_us(100);
            PORTA.F0 = 0;
            Delay_us(100);
            PORTA.F0 = 1;
            Delay_ms(4);
      }
}
```

Вы можете проверить, оттранслировав программу, что микроконтроллер отправляет символ «1».

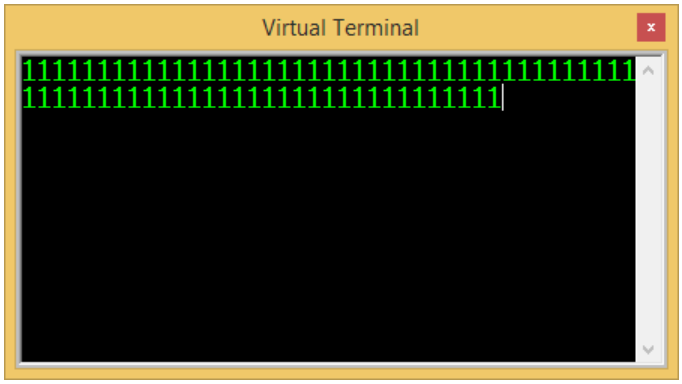

Рис. 8. Передача символа «1» по RS232

Скорость передачи 9600 при тактовой частоте (внутренний генератор PIC16F628A) 4 МГц.

Заключение

Если вы не только читали этот рассказ за компьютером, но и проделали всё, что в нём описано за компьютером, то…

Вы приобрели начальный опыт работы с микроконтроллером. Вы можете определиться с выбором модели микроконтроллера (я бы советовал приобрести на первый случай самый дешёвый, он позже пригодится). Вы можете определиться с выбором среды разработки и языком программирования, который лучше всего подходит лично вам. И вы можете, если не спешите с покупкой микросхемы, продолжить создавать несложные программы.

Например, вы можете создать программу передачи любого байта по RS232. Достаточно дополнить программу, например, массивом (char buf[8]; для MicroC Pro) и переменной (char out = 0x31;), которую следует записать побитово в массив. Для определения бита вы можете использовать операцию «И» и сдвиг переменной, скажем, так: `buf[2] = (out>>2) & 0x1;`

Затем, когда массив заполнен, вы можете передать его в порта А командами: `PORTA.F0 = buf[2];` и т.д. Вы можете попробовать сразу использовать операции сдвига и логического умножения, не используя массив, чтобы понять, зачем нужен массив.

И ещё, вы можете попробовать принимать данные по RS232, подобно тому, как данные передавались, используя один из выводов на вход и сканируя его состояние с записью результата в массив или переменную.

Если вы оформите функцию инициализации процесса, где учтёте скорость передачи и тактовую частоту, если оформите функцию заполнения буфера для нужного числа перед передачей, вы поймёте пользу от использования функций.

Словом, вы можете (не забудьте про datasheet!) узнать и понять многое, многому научиться, создавая и проверяя свои программы за компьютером, даже до покупки микросхемы и первой пайки. Было бы желание это сделать.