Создание проекта Theremin на базе стартового набора PICkit 2 Debug Express

Колин ГРИВЗ (Colin GREAVES)

Очень компактный отладчик от Microchip на первый взгляд больше похож на брелок, однако внутри располагается все, что необходимо для разработки проекта на базе микроконтроллера.

Стартовый набор разработчика PICkit 2 Debug Express уже упоминался в прошлых выпусках обзорных статей о PICkit 2, где анализировались возможности среды разработки, особенности отладчика и маленькой демо-платы, входящей в состав набора.

Автор предлагает рассмотреть некоторые из этих особенностей при создании следующего проекта на базе данного набора. Добавив всего 3 внешних компонента, которые почти всегда имеются под рукой у инженера-разработчика, мы создадим звуковой контроллер (теремин), который представляет довольно сложное устройство. PICkit 2 Debug Express позволит микроконтроллеру выполнить всю работу.

Правильный подход к процессу разработки подразумевает разбиение проекта на мелкие последовательные шаги. Большинство инженеров начинают работу с банального мигания светодиодами, чтобы убедиться в работоспособности устройства. Выполнив данный шаг, вы можете приступать к реализации своих функций. На демо-плате имеются светодиоды, подключенные к выводам порта PORT D микроконтроллера. С них мы и начнем.

Для разработок на базе PIC-контроллеров Microchip предоставляет бесплатную среду разработки MPLAB IDE. Она имеется на компакт-диске в составе стартового набора, однако можно проверить на сайте разработчика, есть ли там последняя версия. Специальный «мастер» для создания проекта в MPLAB позволит быстро создать новый проект. На демо-плате установлен микроконтроллер PIC16F887, что необходимо указать при настройке проекта «мастером». Следующий шаг — выбор toolsuite. В нашем случае нужно выбрать MPASM, так как код проекта будет написан на ассемблере. Но вы можете установить и C-компилятор.

Далее необходимо указать директорию проекта. Создав проект с названием Theremin в корневом каталоге, вы создадите файл проекта (Theremin.MCP) и файл рабочей области MPLAB (Theremin.MCW), где будут сохранены индивидуальные настройки о местоположении открытых окон (рис. 1).

На данной стадии в проекте нет ни одного файла, пропустите шаг 'add files', и «мастер» закончит работу. Эти файлы проекта должны быть созданы пользователем, но необязательно «с чистого листа». Нам потребуется файл с исходным кодом (*.ASM) и включаемый файл, определяющий все используемые регистры и названия конкретных битов, указанных в спецификации (даташите). ASM-файл может быть создан на базе файлов из папки C:\Program Files\Microchip\MPASM Suite\Template\Code. В данной папке есть заготовки под все процессоры, и, скопировав файл16F887TEMP.ASM в новую директорию C:\Theremin, мы сэкономим время на создание шаблонной заготовки. Переименуйте этот файл в Theremin.ASM (рис. 2).

Откройте проект Theremin.MCP в среде разработки MPLAB. Для добавления исход-

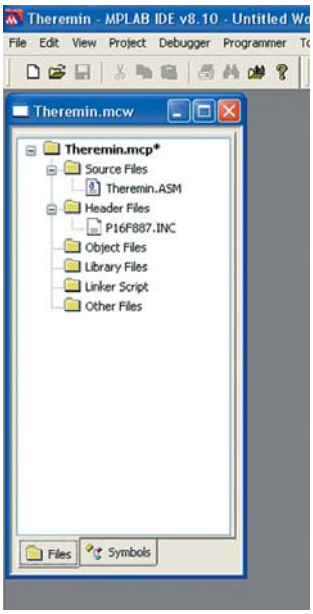

Рис. 1. Создание файла проекта

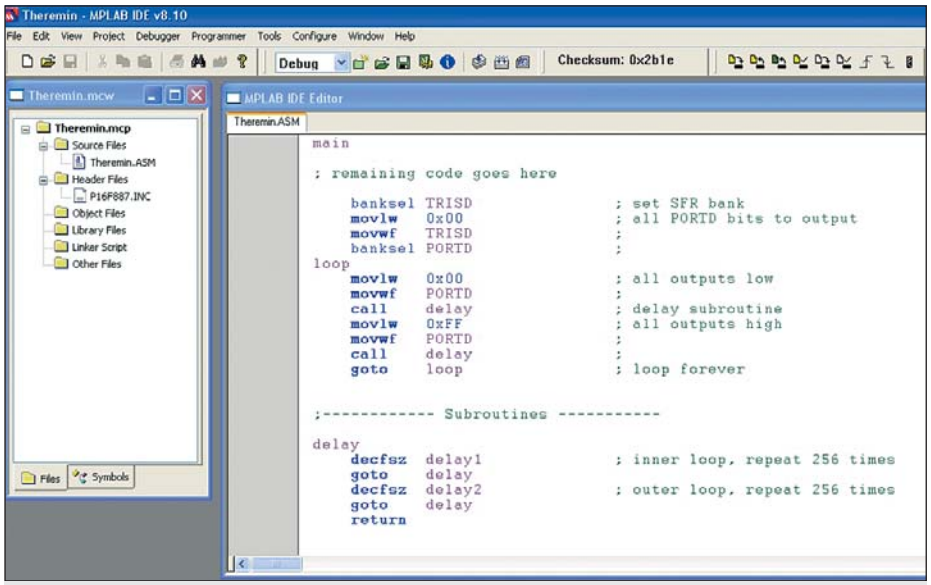

Рис. 2. Создание файла с исходным кодом

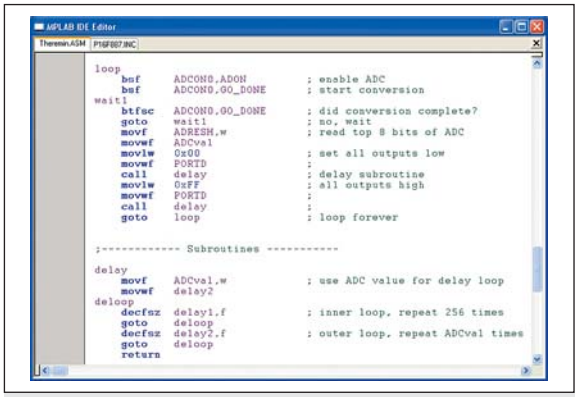

Рис. 3. Добавление включаемого файла для P16F887.INC

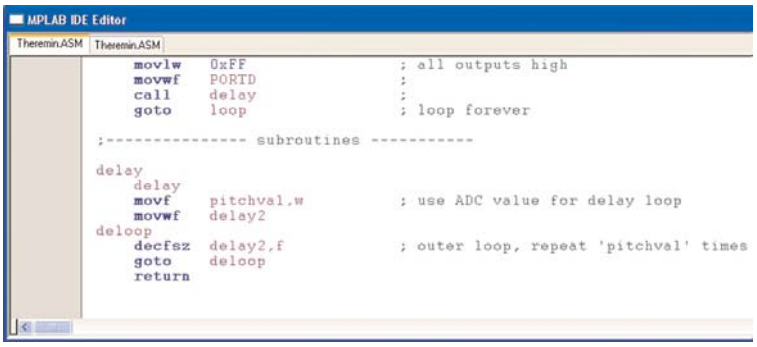

Рис. 4. Пример исходного ассемблерного кода

ных файлов щелкните правой кнопкой мыши на вкладке source file. Откройте папку проекта и выберите Theremin.ASM в качестве исходного кода. Аналогично во вкладке header file добавьте включаемый файл для P16F887.INC из директории C:\Program Files\Microchip\MPASM Suite (рис. 3). Проект готов. Мы можем попробовать его скомпилировать. Нажмите кнопку Build All, компилятор попытается собрать проект и создать его образ. По окончании откроется окно Output, в котором появится сообщение Build Succeeded. Если компиляция закончилась неудачей (failed), возможно, вы присоединили неверный файл.

Открыв Theremin.ASM, мы можем увидеть исходный ассемблерный код (рис. 4). Первое, на что следует обратить внимание, — настройки CONFIG. Это шаблон инициализации контроллера: необходимо убедиться, что он удовлетворяет вашим требованиям. Некоторые из важнейших установок — тип осциллятора, функция вывода MCLR (вывод сброса), настройки сторожевого таймера. В нашем случае — осциллятор внутренний (internal), функция MCLR включена и сторожевой таймер отключен. По умолчанию установки такие, как нам и требуются: ничего изменять не нужно.

Следующий шаг — определение переменных, но мы вернемся к этому позже, примите это во внимание. Вектор сброса и обработчик прерываний настроены. Вектор сброса осуществляет переход к метке 'main'. Прерывания в нашем случае не востребованы, поэтому игнорируем их в коде. В общем, этот простой шаблон образует полнофункциональную структуру, необходимую для начала добавления своего кода.

Первый шаг — установка направления портов ввода/вывода. Выводы PORTD подключены к светодиодам, поэтому настраиваем их как выходы. Регистр, отвечающий за направление портов ввода/вывода, называется TRISD. Установив 0 во все 8 бит данного регистра, мы назначаем все выводы на выход. Обратите внимание, что управляющие регистры находятся в банках, поэтому необходимо выбрать нужный банк перед записью или чтением соответствующего регистра. Для этого служит команда 'banksel'. Например, для записи в TRISD необходимо предварительно выполнить команду 'banksel TRISD'.

Далее создаем примитивный бесконечный цикл, в котором будут записаны в выходные биты 0 и 1. Чтобы увидеть изменение состояния светодиодов, необходимо замедлить процесс, добавив функцию задержки в цикл. Процедура задержки — отдельная подпрограмма вне main, назовем ее 'delay'. Данный код может быть вызван командой 'call', он осуществляет обратный отсчет от 255 до 0 и возвращает программу к месту вызова команды 'call'. Так как одна задержка — маленький временной промежуток (PIC работает на скорости 8 МГц!), можно использовать второй цикл задержки, тем самым увеличив задержку до 255×255 инструкций. Этого уже достаточно.

Перекомпилируйте проект и посмотрите, что получилось. Это необходимо сделать, чтобы убедиться, что все предыдущие действия выполнены верно. Мы видим несколько ошибок: переменные, используемые в подпрограмме задержки, не описаны. Эти переменные необходимы для реализации обратного отсчета и должны быть объявлены в блоке объявления переменных в верхней части исходного кода. Добавьте их и выберите местоположение в памяти. Бегло просматривая спецификацию, можно найти таблицу специальных регистров (SFR), и в конце первого банка, за регистром 20h память свободна. Это хорошее место для хранения переменных. Спецификация имеется на компакт-диске, включенном в состав набора, вместе с другими полезными руководствами. Конечно же, более новую документацию и новые примеры применений можно скачать с сайта Microchip. Также рекомендуется перед началом разработки ознакомиться с последним выпуском Errata на микроконтроллер: там содержится информация об опечатках и ошибках в спецификации, изменениях некоторых функций контроллера и т. п.

Описав переменные, попробуем еще раз скомпилировать проект. Подстройка и компиляция — на первый взгляд, ленивый способ написания кода, но он позволил вам быстро получить готовый проект. Теперь попробуем запрограммировать контроллер.

Подключите отладчик PICkit 2 к демонстрационной плате и к ПК по USB-кабелю. Питание для платы будет подаваться от USB. Когда вы впервые подключаете стартовые наборы от Microchip к ПК, Windows попросит вас установить драйверы. Microchip предоставляет специализированные драйверы. (Не устанавливайте стандартные драйверы Windows!) Они расположены в папках с соответствующим названием в директории Microchip\MPLAB IDE. Однако PICkit 2 — стандартное HID-устройство и не требует установки специальных драйверов.

После подключения PICkit 2 может быть выбран в качестве программатора. Выберите в меню Programmer — Select Programmer — PICkit 2. В окне output будет выведен отчет о процессе инициализации PICkit 2 и обновлении его ОС. Выбрав в меню Programmer — Program, вы начинаете загрузку вашего кода в контроллер демо-платы. Теперь в окне output отображается прогресс загрузки. Любая ошибка на данном этапе наверняка является следствием плохого соединения между PICkit 2 и демо-платой.

Теперь выберите в меню Programmer — release from reset, и вы увидите, как светодиоды замигают с частотой 2–3 Гц. Итак, все работает!

Кстати, все пункты меню Programmer, которые мы использовали, имеются на панели инструментов MPLAB для быстрого доступа.

Теперь посмотрим, как отладчик может помочь в отладке кода, непосредственно выполняющегося контроллером. Сбросьте микроконтроллер, нажав на иконку спадающего фронта (или выберите в меню Programmer — Hold in Reset): светодиоды перестанут светиться. PICkit 2 может быть переведен в режим отладки: для этого необходимо выбрать в меню Debugger – Select Tool – PICkit 2. Вы увидите сообщение о том, что нельзя выбрать один инструмент (PICkit 2) в качестве программатора и отладчика одновременно. Режим отладки требует часть памяти контроллера для отслеживания статусов регистров и связи

с MPLAB. Эта часть кода должна быть запрограммирована в память PIC-контроллера: для этого необходимо перепрограммировать контроллер в режиме debug. На панели инструментов появятся новые иконки для отладки — RUN, STEP и RESET. Иконка RUN служит для начала выполнения кода, HALT — для остановки PIC-контроллера. Так как 99% времени наша программа находится в процессе выполнения подпрограммы задержки, скорее всего останов контроллера будет приводить к появлению зеленой стрелки на инструкции подпрограммы задержки. Эта инструкция — следующая, которая будет выполнена. Если вы наведете курсор мыши на переменные delay1 или delay2, то увидите их текущие значения. Чтобы остановиться на одной из инструкций основной программы, удобно воспользоваться точками останова (breakpoint).

Двойной щелчок на строчке с инструкцией movlw 0x00 (следующая за строчкой с меткой 'loop') приведет к появлению символа B в красном кружке справа от этой строки. Это значит, что на данной строчке установлена точка останова. Нажмите на иконке RUN, и PIC-микроконтроллер будет выполнять код, пока не дойдет до строчки, на которой установлена точка останова. Наведите курсор мыши на название регистра PORTD в коде программы и увидите текущее значение 0xFF (все 1), то есть все светодиоды горят. Теперь нажмем на иконку RUN, контроллер один раз прокрутит полный цикл и остановится: на этот раз значение PORTD равно 0x00, и все светодиоды не горят.

Пока мы только ознакомились с тем, как управлять аппаратными средствами, а теперь сделаем следующий шаг — рассмотрим регистры специального назначения (SFR) PIC-микроконтроллера, выбрав в меню View — Special Function Registers. Здесь приведены все управляющие регистры и их значения. Дважды кликните на регистре PORT, значение которого отображается как 0x00 или 00000000 в двоичном формате. Попробуйте поменять значение одного из двоичных битов с 0 на 1 — вы можете это сделать. Теперь взгляните на плату: светодиод загорелся! Такой доступ к программе, портам ввода/вывода и многому другому помогает разработчику понять, что происходит, и упрощает процесс отладки проекта.

Для создания Theremin нам потребуется некий аналоговый интерфейс. Наиболее простым решением было бы использование потенциометра, подключенного к одному из аналоговых входов PIC16F887, но в соответствии с определением Theremin лучше использовать оптический сенсор, детектирующий близость человеческих рук. Для этого подойдет обычный фоторезистор (в составе делителя напряжения), значение которого определяется освещенностью комнаты, где будет использоваться Theremin. В нашем случае получилось так: когда руки почти целиком

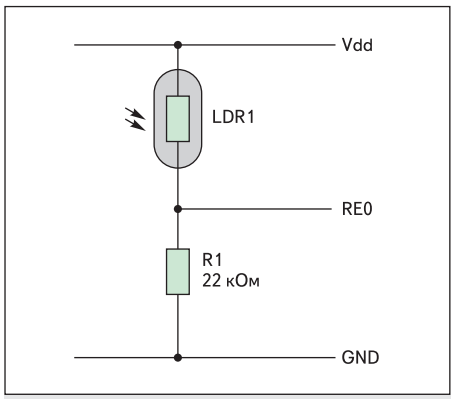

Рис. 5. Схема для проекта Theremin

покрывают фоторезистор, его сопротивление составляет порядка 200–300 кОм, без рук — порядка 2 кОм. Для обеспечения оптимального напряжения на входе АЦП резистор с сопротивлением 20 кОм будет использован в качестве фиксированного сопротивления делителя напряжения. Это сопротивление вдесятеро больше минимального сопротивления фоторезистора и вдесятеро меньше максимального.

Схема получается предельно простая (рис. 5), выберем расположение так, чтобы проще было монтировать. Вывод RE0 наилучшим образом подходит для поставленной задачи, так как позволяет подсоединить резисторы делителя к Vdd и GND соответственно без переходных отверстий (рис. 6) и имеет функцию аналогового входа AN5. В данном контроллере АЦП имеет 14 входных каналов, выбираемых мультиплексором. Мы будем использовать пятый.

Глава 9 спецификации на PIC16F887 содержит всю необходимую информацию для работы с АЦП. К настройкам АЦП относятся скорость преобразования, формат результата, источник опорного напряжения и другие. Работа с АЦП идентична для множества PIC-контроллеров, поэтому время, потраченное на изучение данного раздела, окупится в дальнейшем, и не раз. Сразу можно сказать, что

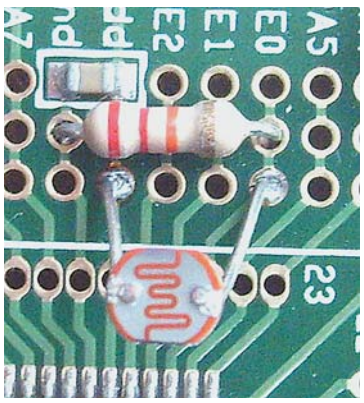

Рис. 6. Подсоединение резистора без переходных отверстий

нам достаточно 8-разрядного результата, поэтому форматирование результата следующее: старшие биты помещаются в 1 регистр, а младшие игнорируются. Частота преобразования — Fosc/2 (4 МГц), в качестве опорного напряжения будем использовать Vdd, так как у нас нет необходимости в высокой точности измерения. Итак, мы будем получать 8-битный результат в соответствии с уровнем напряжения на выводе AN5 от Vdd (+5 В) до GND.

Для настройки АЦП в коде программы (рис. 4) необходимо выставить значения трех регистров — ADCON0, ADCON1 и ADSEL. Обратите внимание, что и тут понадобится команда 'banksel'. Нажмите кнопку 'build all' на панели инструментов, чтобы проверить код на наличие синтаксических ошибок.

Теперь АЦП настроено для работы, и нам необходимо изменить цикл мигания светодиодами, чтобы использовать АЦП. Для начала преобразования АЦП необходимо разрешить начало преобразования в регистре ADCON0. Изначально оно разрешается, и лишь затем дается команда начала преобразования. Эти 2 бита — для разрешения и начала преобразования — находятся в одном регистре, однако их устанавливать необходимо в две операции: сначала разрешить, после стартовать. Установка в 1 бита GO_DONE начинает преобразование. АЦП производит измерение и помещает результат в регистры ADRESH и ADRESL (если результат 10-битный). По окончании бит GO_DONE сбрасывается в 0, и результат можно считать. Нам достаточно значения в регистре ADRESH (8-битный результат). Считываем это значение и записываем в регистр с соответствующим названием — pitchval. Это значение будет использоваться проектом Theremin.

В последней части основного цикла мы видим, что выходы переключаются с задержкой, определяемой значением АЦП. Перекомпилировав проект ('build all') мы видим ошибку: не определена новая переменная pitchval.

В итоге, чтобы установить зависимость между частотой переключений и значением pitchval, мы должны добавить эту переменную в цикл задержки. Процедура задержки изменена, чтобы обойтись одним циклом, а не циклом в цикле. Необходимо подобрать количество переключений из области звуковых частот, а не единиц герц, чтобы результат можно было услышать. Подпрограмма задержки становится совсем простой: несколько инструкций, чтобы загрузить значение pitchval в переменную delay2. Нажимаем 'build all'. Задача решена! Программирование займет порядка секунды и проект можно запустить. Обычный пьезодинамик, подключенный к выводу RD0 порта PORTD, завершит конструкцию (рис. 7). Теперь мы услышим изменение тона в зависимости от того, насколько близко расположены руки к фоторезистору. Используйте инструкцию nop (No Operation) после метки deloop, чтобы по-

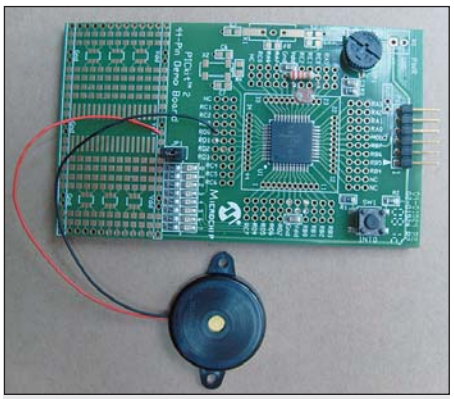

Рис. 7. Подключение пьезодинамика

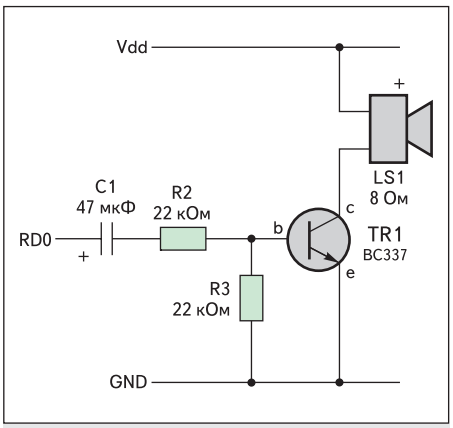

Рис. 8. Схема усилителя

догнать частоту переключений к наиболее слышимому диапазону частот.

Прежде чем показывать ваш музыкальный шедевр друзьям, неплохо бы увеличить громкость. Качество звука из пьезоизлучателя оставляет желать лучшего. Усилитель можно построить на транзисторе BC337 (рис. 8) и нескольких резисторах. (Запитать схему усилителя лучше независимым источником питания, так как USB не сможет обеспечить необходимый ток.) Конденсатор добавлен для предотвращения выгорания, если PIC-контроллер будет остановлен, когда все выходы имеют высокий уровень. Дешевые 8-Ом динамики дадут хороший результат. Не забудьте перепрограммировать PIC-микроконтроллер в режиме program (а не debug), чтобы использовать устройство стационарно (без PICkit 2). Когда программа отлажена и работает, режим debug вам больше не нужен. Теперь вы имеете полнофункциональный проект Theremin, созданный собственными руками.

Для создания проекта Theremin или другого сопутствующего проекта, посмотрите дополнительное специальное предложение от компании Microchip на следующей странице.