

Вопросы и ответы по программированию на С для микроконтроллеров PIC.

Для начинающих

Первая попытка (моя) сделать FAQ по программированию на С для контроллеров PIC. Использованы материалы сайта [MicrochipC](#), техническая документация от [Microchip Tech Inc](#), [High Tech Software](#). Предложения и вопросы жду [почтой](#). В ближайших планах - расширение FAQ и небольшой цикл статей для начинающих.

- *Все представленные материалы являются некоммерческими и могут использоваться при условии сохранения ссылок на источник.*

Автор: [Виктор Леонтьев](#).

Последнее обновление: 26 Января 2007 г.

Оглавление

1. [Почему лучше программировать на С?](#)
2. [Что и где можно почитать по программированию на С?](#)
3. [Где можно найти информацию на русском языке по микроконтроллерам PIC?](#)
4. [Что такое embedded system?](#)
5. [Можно ли писать на Си программы для процессоров с небольшим объемом программной и оперативной памяти?](#)
6. [Нужно ли мне изучать систему команд процессора, если я пишу на С?](#)
7. [Какой компилятор лучше?](#)
8. [Что такое RTOS?](#)
9. [Где можно взять компилятор Hi-Tech PICC?](#)
10. [Где взять документацию по компилятору Hi-Tech PICC?](#)
11. [Где найти примеры программ на Си для микроконтроллеров PIC?](#)
12. [Как разрабатывать программы на С с помощью среды разработки MPLAB?](#)
13. [Как создать простой проект на Си?](#)
14. [Почему MPLAB 6.X не выдает сообщения об ошибках в программе?](#)
15. [Как я могу из своей программы на С обращаться к ресурсам процессора?](#)
16. [Какие типы данных поддерживает компилятор HT PICC?](#)

1. Почему лучше программировать на С?

Во избежание излишней полемики сразу акцентирую внимание, что здесь изложено мнение автора и оно не претендует на абсолютную истину.

С – это быстрый, устоявшийся, достаточно простой и переносимый язык и что самое главное – он выдержал испытание временем. В языке С гармонично сочетаются возможности программирования низкого уровня со свойствами языка высокого уровня. Возможность низкоуровневого программирования позволяет легко оперировать непосредственно аппаратными средствами, а свойства языка высокого уровня позволяют создавать читаемый и модифицируемый программный код. Поскольку в С делается акцент на эффективность, то это язык, в котором применяются наиболее сложная оптимизация кода времени компиляции. Это справедливо и в отношении компилятора Hi-Tech PICC, который генерирует весьма компактный и эффективный код. Возможность применения ассемблерных вставок делает этот компилятор еще более привлекательным для разработки программ.

И последнее – на дворе 21 век!!!

[в начало](#)

2. Где и что можно почитать по программированию на Си?

Невзирая на то, что в последнее время значительно увеличилось количество литературы по C++, все же время от времени можно купить и отличные книги по C.

Естественно, основополагающим трудом является книга Б. Кернигана и Д. Ричи – “Язык программирования C”. Найти этот бестселлер достаточно легко, например [здесь](#) в разделе “UNIX and GNU documentation Library”.

Очень хорошая книга для начинающих – “Язык СИ”, М. Уэйт, С. Прата, Д. Мартин, 1988 г, издательство “Мир”. Она является раритетом, но можно попытаться поискать ее на полках программистов C++, которые наверняка смогут с ней расстаться.

“Искусство программирования на C” – отличный фундаментальный труд известных специалистов Р. Хэзфилда и Л. Кирби. Книга издана в 2001 году издательством “ДиаСофт”.

Многое можно найти в Интернет, в частности на сайте www.citforum.ru в разделе “Программирование”. Ну а дальше – поисковые машины еще никто не отменил.

P.S. Безусловно, самого пристального внимания заслуживает фундаментальный труд Д. Кнутта – “Искусство программирования”

[в начало](#)

3. Где можно найти информацию на русском языке по микроконтроллерам PIC?

Огромная и необычайно полезная работа делается на сайте по технической поддержке продукции Microchip Technology Inc. www.microchip.ru, где можно найти переводы фирменной технической документации. Много интересной информации содержится на сайте официального представителя компании Microchip Technology Inc. – www.gamma.spb.ru.

[в начало](#)

4. Что такое embedded system?

Дословно этот термин переводится как “внедренная система”. В этом случае компьютер приспособляется для весьма специфического использования, поэтому стандартные компоненты настольных систем, как правило, отсутствуют. Во встроенных системах используется любые доступные программируемые аппаратные средства, как на основе специализированных процессоров, так и на основе обычных микропроцессоров, известных в мире настольных вычислительных систем. К основным признакам встроенных систем наверно можно отнести следующее:

- компьютер приспособлен к конструкции машины, в которую встроен
- имеются необычные устройства ввода-вывода

- система разработана исключительно для выполнения, программное обеспечение создано вне ее.

[в начало](#)

5. Можно ли писать на Си программы для процессоров с небольшим объемом программной и оперативной памяти (например, семейство 12C/12F)?

Никаких принципиальных ограничений язык программирования не накладывает, все зависит от качества программирования. В литературе описаны системы, построенные на очень скромных ресурсах, но программное обеспечение для них написано на С (хороший пример для изучения – светорегулятор на базе PIC 12C508, документ PICREF-4 с www.microchip.com). Очень поучителен для изучения проект на базе 12C509A (41 байт RAM, 1024 байт ROM) для удаленного управления вентилятором, построенный на базе операционной системы реального времени (конечно, написанной на С), реализующий помимо удаленного управления еще много разнообразных функций (application notes AN-6 на сайте www.pumpkininc.com).

[в начало](#)

6. Нужно ли мне изучать систему команд процессора, если я пишу на С?

Безусловно, да! Программирование встроенных систем имеет свои особенности, связанные в первую очередь с тесным взаимодействием с периферийными средствами. Знание системы команд позволит проанализировать код, сгенерированный компилятором, даст возможность определить узкие места и в ряде случаев, понять причину возникающих ошибок. Некоторые программные секции, с целью оптимизации скорости и/или размера кода, придется писать на ассемблере, что невозможно без знания системы команд процессора.

[в начало](#)

7. Какой компилятор лучше?

В настоящее время на рынке представлено достаточно много компиляторов С для процессоров PIC. В этом документе будет вестись разговор о достаточно популярном компиляторе Hi-Tech PICC v.8.0X компании HI-TECH Software. Хороший обзор различных компиляторов для процессоров PIC можно найти на сайте www.gamma.spb.ru.

[в начало](#)

8. Что такое RTOS?

Real Time Operation System (RTOS) – операционная система реального времени. В ней есть практически все функции, присущие современным операционным системам, таким как UNIX и Windows, как то: многозадачность, многопоточность, межпроцессорная коммуникация. Основная особенность RTOS – гарантированное время отклика на внешние и/или внутренние события в системе, что не могут обеспечить обычные операционные системы. Типичный пример компактной, но достаточно мощной RTOS – это SALVO компании [Pumpkin Inc.](http://PumpkinInc.com) Эта операционная система занимает удивительно мало места (372 байта ROM, 33 байта RAM на процессоре типа 16F84, по материалам фирмы), но предоставляет весьма значительные возможности.

[в начало](#)

9. Где можно взять компилятор Hi-Tech PICC?

Демо-версия компилятора находится на сайте разработчика – компании [HI-TECH Software](#). Реализуют полноценные версии компилятора несколько фирм, например ["ООО Микрочип"](#).

[в начало](#)

10. Где взять документацию по компилятору Hi-Tech PICC?

Документация на компилятор HT PICC можно взять с [сайта разработчика](#), она достаточно подробная, там есть что почитать, но, к сожалению, только на английском языке! Насколько мне известно, целенаправленно ее переводом никто не занимался, я ее перевожу кусками, в основном для личного пользования. Если нахожу что-то интересное – кладу сюда.

[в начало](#)

11. Где найти примеры программ на Си для микроконтроллеров PIC?

В первую очередь – каталог samples компилятора. Там есть несколько достаточно интересных для начала примеров, практически по всему спектру задач. К сожалению, в этих примерах иногда встречаются ошибки, так что будьте внимательны. Следующий источник – примеры с сайта Microchip Technology Inc., раздел ["Application Notes, Code Examples & Templates"](#). Многие из этих примеров содержат хорошо комментированный исходный код на C. Естественно, на английском языке. Очень интересный сайт – www.microchipC.com, содержит массу полезной информации. Много для этого FAQ я почерпнул оттуда. Язык тоже не наш.

P.S. Чтобы не тратить деньги в дебрях микрочиповского сайта, закажите себе бесплатно CD_ROM с его копией. Это можно сделать, заполнив несложную форму на www.microchip.com или на www.microchip.ru. В последнем случае вам пришлют английскую версию и массу интересных переводов на русский язык.

[в начало](#)

12. Как разрабатывать программы на C с помощью среды разработки MPLAB?

Среда разработки MPLAB бесплатно предлагается фирмой Microchip Technology Inc. и позволяет создавать и отлаживать программные проекты на ассемблере. Для разработки программ на C необходимо дополнительно установить компилятор. Достаточно популярен и широко используется компилятор HT-PICC компании [HI-TECH Software](#). Для интеграции компилятора нужно сделать следующее.

Устанавливаем компилятор, запустив на исполнение файл с компилятором, никаких трудностей быть не должно. По умолчанию компилятор устанавливается в каталог C:\HT-PIC.

Устанавливаем MPLAB. Дальнейшее зависит от выбора версии MPLAB. Сейчас предлагаются и используются две версии – версия 5.70 и 6.30. Первый вариант достаточно отработан но имеет весьма слабый редактор и менеджер проектов. Версия 6.30 выглядит гораздо привлекательней, но багов в ней достаточно много!

Для версии 5.70.XX. После установки MPLAB заходим в меню project => install language tools. В появившемся окне в Language Suite выбираем HI-TECH PICC, в Tool Name – PICC compiler. В строке Executable указываем путь к компилятору, по умолчанию это C:\HT-PIC\BIN\PICC.EXE. После чего в Tool Name выбираем PICC assembler и на вопрос Save Tool Setting? отвечаем YES. Для PICC assembler и PICC linker в строке Executable указываем тот же путь и файл, что и для компилятора (C:\HT-PIC\BIN\PICC.EXE). Сохраняем. Во всех трех случаях должен стоять флажок на Command-line. Все готово для работы.

Для версии 6.30. После установки MPLAB заходим в меню project => set language tool location. В открывшемся окне в дереве выбираем HI-TECH PICC Toolsuite и в нем открываем пункт Executables. Появляется три строчки про ассемблер, компилятор и линковщик. Для каждого из них в строке Location прописываем путь к исполняемому файлу (можно руками, можно через кнопку Browse). Во всех трех случаях указываем одну и ту же строку с ссылкой на файл picc.exe, например C:\HT-PIC\BIN\PICC.EXE. Нажимаем ОК. Готово!

Аналогично устанавливаем и PICC 18, если есть необходимость.

[В начало](#)

13. Как создать простой проект на Си?

Для MPLAB 5.70.XX. Предварительно на диске создаем каталог, в котором будут размещаться файлы проекта. В меню Project выбираем пункт New Project. В открывшемся окне выбираем нужный каталог и задаем имя проекта (в строке File Name). Нажимаем ОК. Появляется окно редактирования свойств проекта. В Target Filename должно стоять имя нашего проекта с расширением hex. В меню Language Tool Suite выбираем HI-TECH PICC, а в окне Development Mode выбираем нужный тип процессора, нажав Change. Появляется окно Development Mode с закладками. В закладке Tools выбираем нужный процессор и систему эмуляции (в простейшем случае, при отсутствии аппаратных средств отладки, выбираем MPLAB SIM Simulator). Под типом процессора есть окошко, в котором пишут об ограничениях симулятора на данный процессор. В закладке Clock выбираем тип тактового генератора и его частоту. Остальные закладки можно для начала не трогать. Нажимаем ОК. Появляется сердитая надпись, что не было построено hex-файла для этого проекта. Выбор небогатый, нажимаем ОК. Потом еще раз внимательно читаем такое же сообщение и опять нажимаем ОК. В следующем сообщении нас предупреждают, что "после изменения режимов будет очищена память программ и что надо перекомпилировать проект". Опять предлагают богатый выбор в виде одной кнопки ОК. Соглашаемся. Мы вернулись в окно редактирования проекта (Edit Project). В окне Project Files ставим курсор на имеющейся там файл "имя_проекта[.hex] и нажимаем внизу кнопку Node Properties. Появляется окно с таким же названием. В Language Tool выбираем PICC Linker. Ниже, в окне Description, ставим галочку на строке Generate Debug Info. Больше ничего не меняем, нажимаем ОК. В окне Edit Project появилась кнопка Add Node. Нажимаем и выбираем файл, который нужно добавить в проект. Окно не позволяет создавать файлы, поэтому предварительно нужно создать текстовый файл с расширением ".c". Эту сложную операцию можно сделать, например в Notepad или нажав Shift-F4 в FAR-е. Добавляем столько файлов, сколько нужно для нашего проекта (для начала хватит и одного). В окне Project Files появился еще один файл. Ставим на него курсор и нажимаем Node Properties. В окне Description тоже ставим галочку на строке Generate Debug Info и больше ничего не трогаем. ОК и еще раз ОК.

Теперь нам надо хоть что-то написать. Для этого, открываем наш исходный файл с расширением C (можно через меню File=>Open, можно через Window=>Project и дважды щелкнуть на синем имени файла).

В файл пишем могучий текст в виде:

```
#include <pic.h>

void main (void)

{

}
```

и нажимаем кнопку "Make the current project" (в виде воронки справа на панели инструментов). Появляется окно с результатами компиляции. Если внизу есть строка "Build completed successfully" – то все сделано верно. Если строка не появилась – читаем сообщения об ошибках и разбираемся.

Для MPLAB 6.30. Из меню Project запускаем Project Wizard. Нажимаем Next и выбираем нужный процессор. Опять Next. Выбираем в окне Active Toolsuite HI-TECH PICC Toolsuite. Далее, в следующем окне вводим имя проекта и выбираем каталог размещения. Если каталога нет, MPLAB предложит его создать (wow, мы научились делать это!). В следующем окне кнопкой Add добавляем нужные файлы в проект (если файлов нет, то их нужно предварительно создать, MPLAB пока умеет создавать только каталоги, sic!). Если на добавленных файлах поставить птичку, то они будут скопированы в каталог проекта. После этого слева появиться менеджер проекта, в окне которого будут все исходные файлы. Открыв нужный файл, вводим туда текст нашей первой программы (см. выше) и нажимаем кнопку Build All (справа на панели). Свершилось!

P.S. Учтите, что ваша первая программа не только не делает ничего полезного, она вообще ничего не делает!

[В начало](#)

14. Почему MPLAB 6.X не выдает сообщения об ошибках в программе?

Для лечения этой проблемы нужно сделать следующее. Первое – проверить, что в установках проекта стоит опция Generate Debug Info (Project=>Build Options=>Project, закладка HI-TECH C Compiler, Categories => Output @ Debug, в закладке HI-TECH Linker тоже должно стоять Produce MPLAB debugging info). Если не помогло – возможно, что стоит старый вариант PICC, его нужно пропатчить. Лекарство лежит на сайте Hi Tech, называется mp16fix.zip. Этот файл нужно распаковать и переписать в каталог bin компилятора. Все должно заработать.

[В начало](#)

15. Как я могу из своей программы на C обращаться к ресурсам процессора?

Прежде чем вы начнете писать код, ознакомьтесь с соответствующим header файлом. Его можно найти в каталоге include компилятора. Как правило, имя файла достаточно легко ассоциируется с процессором, и вдобавок, в его начале всегда пишется, для каких процессоров он предназначен. Например,

для процессора 16F627/628 файл называется `pic16f6x.h` (я обычно распечатываю этот файл и держу его под рукой, пока не запомню всю мнемонику для регистров конкретного процессора). Дальше все очень просто. Например:

```
// установить RB0...3 на ввод, RB4...7 - на вывод

TRISB = 0b00001111;

// установить значение порта RB5 в '1'

RB5 = 1;

// установить значение порта RB4

// в зависимости от значения RB0

if (RB0 == 0)

    RB4 = 1;

// функция начальной инициализации (пример)

void Init (void)

{

    INTCON = 0;    // никаких прерываний, настраиваемся...

    PIE1 = 0;

    // один компаратор, RA1: Vin-, RA2: Vin+

    CMCON = 0b00000101;

    // настройка источника опорного напряжения

    VRCON = 0b11000000;

    T0CS = 0;

    PORTA = 0;

    PORTB = 0;

    TRISA = 0b00000110;

    TRISB = 0b00000111;    // RB0...2 - ввод, RB3...7 - вывод

    ...

    ...

} // end Init
```

Причем можно работать как с регистром целиком, так и с отдельными битами. Обратите внимание, что всю возню с переключением банков памяти компилятор делает сам.

[в начало](#)

16. Какие типы данных поддерживает компилятор HT PICC?

Следующие типы:

Тип данных	Синтаксис	Размер, байт	Диапазон
битовая переменная	bit s;	-	0 или 1
беззнаковое целое 8 бит	unsigned char i;	1	от 0 до 255
знаковое целое 8 бит	signed char i;	1	-127 до 127
беззнаковое целое 16 бит	unsigned int k;	2	от 0 до 65535
знаковое целое 16 бит	signed int k;	2	от -32768 до 32767
беззнаковое целое 32 бит	unsigned long m;	4	от 0 до 4294967295
знаковое целое 32 бит	long m;	4	от -2147483648 до 2147483647
знаковое с плавающей запятой 24 бита	float x;	6	
знаковое с плавающей запятой 24/32 бита	double y;	6/8 *	

* - может быть 24 или 32 бита, в зависимости от опции -D24 или -D32 компилятора.

[в начало](#)