Новая архитектура среднего семейства контроллеров Microchip

Многие годы компания Microchip с успехом выпускает 8-разрядные PIC-контроллеры и продолжает вкладывать средства в их развитие для поддержания широкой номенклатуры продукции, которая удовлетворит потребности нынешних и будущих клиентов. Новая усовершенствованная архитектура среднего семейства контроллеров PIC12 и PIC16 взяла все самое лучшее от существующей архитектуры и получила дополнительные возможности.

Илья АФАНАСЬЕВ
ilya@gamma.spb.ru

Архитектура 8-битных PIC-микроконтроллеров среднего семейства PIC12 и PIC16 имеет:

- увеличенный объем памяти программ и данных (до 32 кбайт инструкций и более 4 кбайт памяти данных);
- улучшенный аппаратный стек (увеличен до 16 уровней, программный доступ);
- модернизированный метод переключения между страницами памяти и банками данных;
- дополнительные источники сброса;
- 14 дополнительных программных инструкций, включая команды для более эффективной работы С-компилятора;
- расширенную периферию;
- уменьшенное время входа в прерывание;
- увеличенную тактовую частоту ядра микроконтроллера.

Ядро контроллеров среднего семейства новой улучшенной архитектуры

Новые команды

Микроконтроллеры нового семейства в дополнение к 35 командам существующего среднего семейства получили 14 новых инструкций. Новые команды (табл. 1) позволяют получить более быстрый и более компактный код программы и предоставляют эффективный доступ к данным.

К прежним арифметическим командам сложения и вычитания добавились команды сложения и вычитания с учетом флага переноса/заема. Данные команды улучшают эффективность кода для многобайтной арифметики. К командам циклического сдвига добавились команды логического и арифметического сдвигов. Конечно же, логический сдвиг можно сделать через циклический сдвиг, но для этого понадобятся дополнительные команды.

В новых контроллерах с улучшенной архитектурой среднего семейства увеличено адресуемое пространство памяти программ и ОЗУ. Теперь не нужно устанавливать биты RP0 и RP1 в регистре STATUS для того, чтобы переключить банк памяти, достаточно загрузить адрес в регистр переключения банков BSR. Аналогично и с переключением страниц памяти: добавлена команда загрузки регистра PCLATH.

Новые команды позволяют использовать меньше команд для переключения банков ОЗУ и страниц памяти программ. Для обеспечения совместимости кода между контроллерами нового поколения и старой архитектуры лучше использовать специальные макросы ассемблера PAGESEL и BANKSEL (табл. 2), которые в зависимости от выбранного контроллера будут соответствующим образом интерпретированы компилятором.

Основное достоинство применения команд относительных переходов — это уменьшение размера кода, связанное с отсутствием необходимости отслеживания переходов через границы страниц памяти программ. Иногда, на старом ядре, простой цикл может

Внимательный читатель может задать вопрос: «А почему нет команды арифметического сдвига влево?» Ответ: арифметический сдвиг влево эквивалентен логическому сдвигу влево. Для арифметического сдвига влево вы можете использовать псевдокоманду ассемблера ASLF.

Таблица 1. Новые команды

Новые команды	Мнемоника	Описание
Новые арифметические команды и команды сдвигов	ADDWFC	Сложить значение рабочего регистра W с регистром F с учетом флага переноса
	SUBWFB	Вычесть из регистра F значение рабочего регистра W с учетом флага заема
	LSLF	Логический сдвиг влево
	LSRF	Логический сдвиг вправо
	ASRF	Арифметический сдвиг вправо
Одноцикловое переключение страниц и банков памяти	MOVLP	Загрузить константу в регистр PCLATH
	MOVLB	Загрузить константу в регистр BSR
Относительный переход	BRA	Относительный переход (знаковый)
	BRW	Переход PC+W (беззнаковый)
Вызов подпрограммы с адресом в W	CALLW	Вызов PCLATH:W
Программный сброс	RESET	Сброс программы и периферии
Дополнительные команды косвенной адресации	ADDFSR	Добавить константу к FSR (знаковая команда)
	MOVIW	Копировать косвенно в W
	MOVWI	Копировать W по указателю

Таблица 2. Примеры кода переключения банков ОЗУ и страниц памяти программ

Ядро Mid-Range	Ядро Enhanced Mid-Range	Пример совместимого кода
Вызов функции по любому адресу памяти программ		
movlw high My_Function movwf PCLATH call My_Function	movlp high My_Function call My_Function	PAGESEL My_Function call My_Function
Доступ к произвольной переменной (с переключением банков)		
bsf STATUS, RP0 bcf STATUS, RP1 addwf Var1	movlb HIGH Var1 addwf Var1	BANKSEL Var1 addwf Var1

перестать работать, если сдвинуть программу так, что цикл попадет на границу страницы памяти программ. Таких сюрпризов можно избежать при использовании команды относительных переходов.

Многие знакомы и с организацией вычисляемых переходов на стандартном ядре Mid-Range, где нужно либо усложнять код, либо следить, чтобы таблицы данных не попали на границы блоков в 256 байт. Команда BRW осуществляет переход относительно текущего значения счетчика команд на беззнаковое значение, находящееся в рабочем регистре W. С этой командой операции табличного чтения становятся намного проще:

```
constants
        brw
        retlw DATA1
        retlw DATA2
        retlw DATA3
        retlw DATA4
my_function
        ;… код программы…
        movlw DATA_INDEX
        call constants
        ;… константа DATAx в W регистре
```

Команда BRA аналогична команде BRW, только осуществляет переход относительно текущего значения счетчика команд на 9-битную константу с учетом знака (переход в пределах от –256 до +255 команд).

С командой CALLW становится доступен вызов процедуры по адресу, указанному в регистре W, что может пригодиться для быстрого поиска по таблицам, организации машины состояний и вызова функций по указателю.

Следующие три команды расширяют возможности косвенной адресации.

Команда ADDFSR производит знаковое сложение константы с указателем FSR, значение смещения должно лежать в диапазоне от –32 до +31. Команды MOVIW и MOVWI более сложные. Обе эти команды могут выполняться с пре- и постинкрементом и декрементом указателя, а также со смещением указателя.

Рассмотрим более подробно синтаксис команд MOVWI (команда MOVIW имеет тот же синтаксис, только работает в обратном порядке: пересылает данные из косвенно адресуемого регистра в рабочий регистр W):

- MOVWI 0[INDF0] — перенос значения из W в косвенно адресуемый регистр. Эта операция может быть выполнена также старыми командами MOVWF INDF0 или MOVF INDF0,W. Нет никаких преимуществ у того или иного способа. Команда MOVWI 0[INDF0] не меняет значение указателя.
- MOVWI ++INDF0 — перенос значения из W в косвенно адресуемый регистр с преинкрементом. Если перед INDF0 или INDF1 написать ++, то данная команда произведет инкремент указателя FSR до того, как значение из регистра W будет помещено в косвенно адресуемый регистр. Подобные команды позволяют обновить указатель до записи или чтения.
- MOVWI INDF0++ — перенос значения из W в косвенно адресуемый регистр с постинкрементом. Аналогичны и команды с пре- и постинкрементом: MOVWI-- INDF0 и MOVWI INDF0--. Комбинация пре- и постдекремента — это быстрый способ программного построения стека или циклических буферов данных.
- MOVWI k [INDF0] — перенос значения из W в косвенно адресуемый регистр со смещением относительно указателя. Можно обратиться к регистру со смещением относительно указателя. Данная команда не изменяет значение самого указателя, она востребована для загрузки элемента структуры в рабочий регистр W или получения доступа к регистрам специального назначения в различных банках. Значение смещения может лежать в диапазоне от –32 до +31 относительно адреса, загруженного в указатель FSR.

Память данных контроллеров

В новых контроллерах с улучшенной архитектурой среднего семейства значительно расширена память данных. Теперь они имеют 32 банка памяти общим размером до 4 кбайт. Как и прежде, память данных разделена на регистры специального назначения (Special Function Registers, SFR), которые служат для настройки прерываний, периферии и пр., и регистры общего назначения (General Purpose Registers, GPR), которые служат для хранения пользовательских данных.

Регистры GPR банков памяти с 0 по 30 могут использоваться по усмотрению программиста, банк 31 служит для специальных функций. Назначение этого банка рассмотрим позднее.

В каждом банке памяти отображаются 12 общих регистров специального назначения (табл. 3), таким образом, можно получить доступ к этим регистрам из любого банка данных. Еще одна особенность новых контроллеров в том, что выделенные в таблице 3 регистры автоматически сохраняются при возникновении любого прерывания. Сохраненные при входе в прерывание регистры доступны программисту в 31-м банке.

Таблица 3. Первые 12 регистров специальных функций каждого банка

	Адрес	Регистр	Функция
	0x00	INDF0	Регистр косвенной адресации 0
New!	0x01	INDF1	Регистр косвенной адресации 1
	0x02	PCL	Счетчик команд (младший байт)
	0x03	STATUS	Регистр статуса
New!	0x04	FSR0 Low	Указатель косвенной адресации 0 (младший байт)
New!	0x05	FSR0 High	Указатель косвенной адресации 0 (старший байт)
New!	0x06	FSR1 Low	Указатель косвенной адресации 1 (младший байт)
New!	0x07	FSR1 High	Указатель косвенной адресации 1 (старший байт)
New!	0x08	BSR	Регистр выбора банка
New!	0x09	WREG	Рабочий регистр
	0x0A	PCLATH	Защелка счетчика команд (старший байт)
	0x0B	INTCON	Регистр контроля прерываний

Память контроллеров. Новые свойства указателя FSR

В новых контроллерах с улучшенной архитектурой среднего семейства теперь два регистра FSR (File Select Register), которые к тому же стали 16-битными. Теперь через регистры указателей FSR программист получает доступ ко всей памяти микроконтроллера — памяти данных и памяти программ. Доступ к константам в памяти программ абсолютно такой же, как и к регистрам в ОЗУ: как только вы загрузили указатель, следующей инструкцией вы можете считать значение из памяти программ.

Интересная новая возможность контроллеров — это линейная адресация всех регистров общего назначения. Как видно на рис. 1, регистры общего назначения GPR имеют адреса 20h–6Fh, A0h–EFh, 120h–16Fh и т. д. Такое «дырчатое» распределение регистров общего назначения неудобно для организации больших массивов данных и их адресации. В новых контроллерах добавлен режим линейной адресации памяти (рис. 2), позволяющий при установке указателя FSR на адреса 2000h–29AFh адресовать всю область регистров общего назначения без «дыр». Новый режим адресации позволяет упростить создание больших массивов в памяти данных.

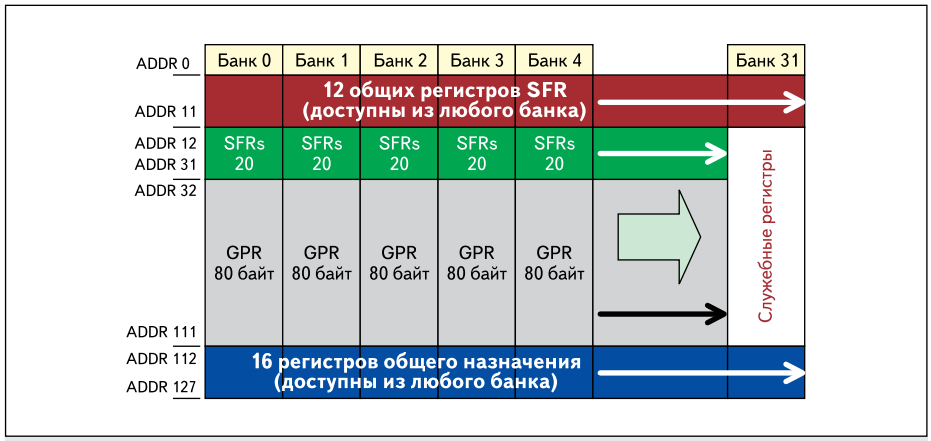

Рис. 1. Память данных нового ядра PIC16

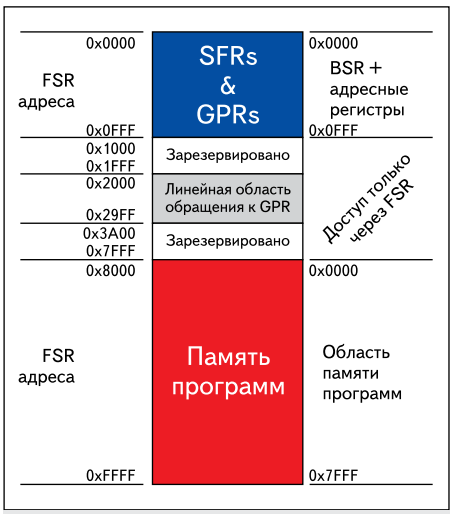

Рис. 2. Организация памяти новых контроллеров. Области адресации памяти данных, программ и доступ при косвенной адресации

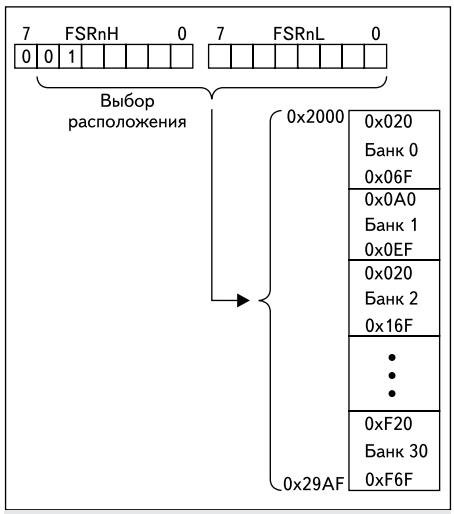

Рис. 3. Линейное отображение регистров общего назначения

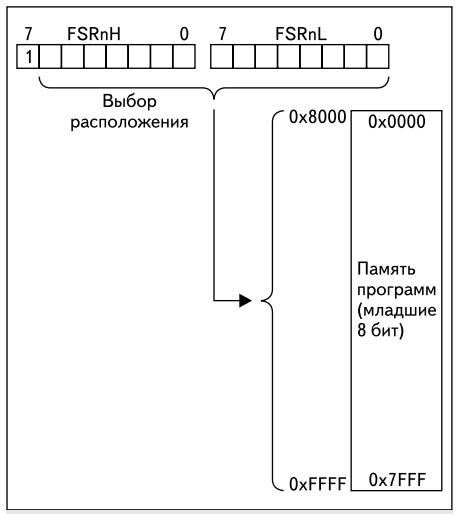

Рис. 4. Адресация памяти программ через регистры косвенной адресации

Если старший бит регистра FSR установлен в «1», то это означает, что регистр FSR будет указывать на область памяти программ (рис. 3, 4).

Стек контроллеров улучшенной архитектуры

Контроллеры среднего семейства PIC16 (табл. 4) имеют кольцевой 8-уровневый стек.

Таблица 4. Сравнение контроллеров среднего семейства (Mid-Range) и улучшенного среднего семейства (Enhanced Mid-Range)

Ядро	Mid-Range	Enhanced Mid-Range
Длина инструкций, бит	14	14
Адресуемая память программ	8 кбит инструкций	32 кбит инструкций
Максимальный объем ОЗУ и регистров, байт	446	>4096
Число инструкций	35	49
Аппаратный стек	8-уровневый	16-уровневый с дополнительными возможностями
Сохранение контекста	Программно	Аппаратно

Если происходит переполнение стека, то происходит и «затирание» самого первого адреса вхождения. В новых контроллерах стек стал больше: он способен хранить 16 адресов. Кроме этого, контроллеры получили возможность сброса при переполнении или опустошении стека. Более того, стек в новой архитектуре доступен программисту через регистры, находящиеся в 31-м банке, — SPTR (указатель стека); текущее значение стека, он состоит из двух байтов — TOSH и TOSL. Программный доступ к стеку может пригодиться при создании операционных систем реального времени или безопасной отладке программ.

Для внутрисхемной отладки кода в старых контроллерах резервируется один уровень стека, в то время как новое ядро имеет отдельный стек для отладки. Это позволяет использовать весь стек в режиме отладки кода, при этом, когда включен режим отладки, сброс по переполнению стека становится точкой останова кода, что позволяет определить причину переполнения стека.

Результаты улучшений

Новая архитектура контроллеров позволяет снизить размер кода на 40% и увеличить производительность микроконтроллера на 50% за счет расширенного набора команд, новых возможностей и увеличенной тактовой частоты.

На рис. 5, 6 приведены сравнительные данные по уменьшению числа циклов и снижению размера кода, которые необходимы для реализации различных алгоритмов.

Тестирование результатов улучшения производилось на следующих задачах:

- RAM to RAM copy: копирование строки между двумя областями GPR-регистров. Данная задача эквивалентна команде strcopy() в Си. С использованием двух регистров косвенной адресации FSR, новых команд и улучшения в работе косвенной адресации данная задача по копированию данных выполняется быстрее и занимает меньший размер памяти программ.

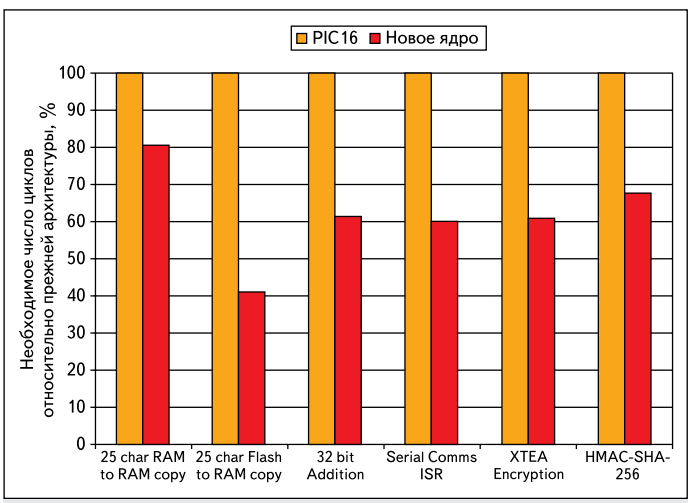

Рис. 5. Сравнение быстродействия контроллеров среднего семейства прежней и улучшенной архитектуры

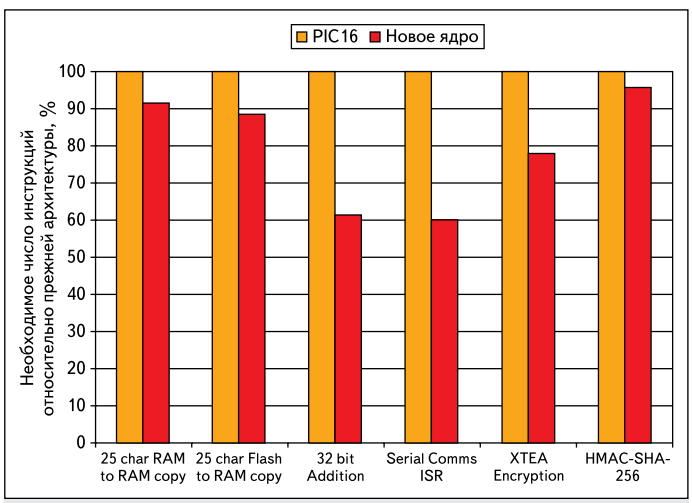

Рис. 6. Сравнение размера кода для контроллеров среднего семейства прежней и улучшенной архитектуры

- Flash to RAM copy. Этот тест аналогичен предыдущему — осуществляет копирование строки между Flash памятью программ и регистрами общего назначения GPR. Подобная задача часто используется для чтения констант, сообщений и других данных из памяти программ микроконтроллера. Примечательно то, что в новых контроллерах функция копирования данных из Flash памяти программ и ОЗУ будет иметь одинаковый программный код за счет возможности адресации данных и памяти программ с помощью регистров косвенной адресации, что позволит еще больше уменьшить размер программы.
- 32 bit addition: сложение 32-разрядных чисел. Улучшенное ядро с новыми командами позволяет существенно сократить размер кода и ускорить арифметические операции с многобайтными данными.
- Serial ISR: подпрограмма обработки данных, получаемых с последовательного порта в прерываниях. Подпрограмма копирует данные из регистра UART в кольцевой буфер. Новые команды работы с регистрами косвенной адресации, а также автоматическое сохранение контекста при входе в прерывание позволяют ускорить работу и сократить размер подобных подпрограмм.
- Алгоритм шифрования XTEA. Это простой и быстрый алгоритм, позволяющий осуществлять шифрацию и дешифрацию данных с помощью секретного пароля.

Таблица 5. Номенклатура новых микроконтроллеров

Наименование	Память			Внутренний генератор	АЦП, каналов	MSSP		EUART	ЖКИ, сегментов	Uпит, В	Число портов	Число выводов
	Flash, кбайт	RAM, байт	EEPROM, байт			SPI	I²C					
PIC12F1822*	3,5	128	256	32 МГц, 32 кГц	4	1	1	1	–	1,8–5,5	6	8
PIC16F1823*	3,5	128	256		8	1	1	1	–		12	14
PIC16F1826*	3,5	256	256		12	1	1	1	–		16	18
PIC16F1827*	7	384	256		12	2	2	1	–		16	18
PIC16F1933*	7	256	256		11	1	1	1	60		25	28
PIC16F1934	7	256	256		14	1	1	1	96		36	44
PIC16F1936	14	512	256		11	1	1	1	60		25	28
PIC16F1937	14	512	256		14	1	1	1	96		36	44
PIC16F1938*	14	1024	256		11	1	1	1	60		25	28
PIC16F1939*	28	1024	256		14	1	1	1	96		36	44
PIC16F1946*	14	512	256		17	2	2	2	184		53	64
PIC16F1948*	28	1024	256		17	2	2	2	184		53	64

Примечание. * Контроллеры находятся на стадии подготовки к массовому производству.

- Алгоритм HMAC-SHA-256. Это алгоритм хеширования, который не осуществляет кодирование или декодирование данных, но проводит аутентификацию (подтверждение подлинности) и подтверждение целост-ности (подтверждение того, что сообщение не было изменено). Оба алгоритма используют ключи, константы, 32- и 64-битную арифметику, а новое ядро с улучшенной системой команд позволяет увеличить быстродействие и снизить требуемый размер кода.

Периферия

Новые контроллеры содержат множество разнообразных периферийных модулей:

- Контроллер ЖК-индикаторов (в контроллерах PIC16F19xx).
- Модуль емкостных датчиков (CSM, Capacitive Sensing Module) поможет в построении сенсорных клавиатур.
- 10-разрядный АЦП.
- Источник опорного напряжения с программируемыми значениями напряжения 1,024, 2,048 и 4,096 В может работать совместно с модулем АЦП, компараторами и 5-разрядным ЦАП.
- Таймеры: четыре 8- и один 16-разрядный, с возможностью подключения кварца 32 кГц и построения микропотребляющих часов реального времени.
- Модули захвата, сравнения и ШИМ (до трех стандартных и до двух расширенных, со специальными режимами управления электродвигателями).
- Модуль синхронного последовательного порта (MSSP): поддерживаются интерфейсы SPI, I²C, совместим с интерфейсами SMBUS и PMBUS.
- Модуль универсального синхронного/асинхронного приемника-передатчика (USART): поддерживаются интерфейсы RS-232, RS-485 и LIN.
- RS-триггер.
- Два аналоговых компаратора.

Номенклатура контроллеров среднего семейства новой улучшенной архитектуры

Компания Microchip начала производство нескольких микроконтроллеров среднего семейства с улучшенной архитектурой. Контроллеры доступны для заказа образцов и серийных партий (табл. 5). Все контроллеры имеют максимальную тактовую частоту 32 МГц, при этом такую частоту можно получить от встроенного программируемого RC-генератора.

Контроллеры PIC16F1xxx выполнены по микропотребляющей технологии XLP и имеют ядро с максимальным напряжением питания 3,6 В, поэтому микроконтроллеры LF-версий имеют диапазон питания 1,8–3,6 В, а контроллеры F-версий содержат встроенный стабилизатор питания ядра и могут работать в расширенном диапазоне питаний от 1,8 до 5,5 В.

Средства разработки для новых контроллеров

Для новых микроконтроллеров созданы как бесплатная среда разработки MPLAB IDE, так и внутрисхемные программаторы-отладчики PICkit3, ICD-3 и REAL ICE.

Поддержку новых контроллеров осуществляют несколько производителей программного обеспечения: компании Hi-Tech, CCS и Byte Craft имеют Си-компиляторы для новых PIC16F1xxx, а компания microEngineering Labs использует новое ядро в компиляторе PICBASIC PRO.

Литература

1. www.microchip.com
2. PIC16F193X/LF193X Data Sheet.
3. PIC1xF1xxx Software Migration.