Обзор Расширенного Среднего семейства

Enhanced PIC12, PIC16

Обзор семейства PIC16F1XXX

- Обзор
- Распределение памяти
- Новые Инструкции
- Улучшенная косвенная адресация

Цели по улучшению PIC16F1xxx

- Увеличение адресуемой памяти программ
- Увеличение места под периферию
- Увеличение максимального размера памяти данных.
- Улучшение метода переключения страниц/банков
- Улучшение эффективности для ‘C’
- Максимальная совместимость.

Сравнение 1-ых страниц PIC16XXX и PIC16F1XXX

Старое семейство

- **Высокопроизводительное RISC ядро:**
- **35 инструкций**
  - Все команды выполняются за 1 цикл, исключая команд переходов
- **Тактовая частота:**
  - DC – 20 MHz генератор/внешний вход
  - DC – 200 нсек командный цикл
- **Прерывания**
- **8-и уровневый аппаратный стек**
- **Прямая, Косвенная и Относительная адресации**

Новое

- **Высокопроизводительное RISC ядро :**
- **49 инструкций**
  - Все команды выполняются за 1 цикл, исключая команд переходов
- **Тактовая частота :**
  - DC – *32* MHz генератор/внешний вход
  - DC – *125* нсек командный цикл
- **Прерывания с автоматическим сохранением контекста**
- **16-и уровневый аппаратный стек со сбросом по переполнению/опустошению**
- **Прямая, Косвенная и Относительная адресации**
  - Два 16-bit File Select Registers (FSR)
  - FSR-ы обращаются к памяти данных и программ.

Быстрое сравнение

	PIC16	Enhanced PIC16	PIC18
Max GPR/SFR	336 / 110	2496 / 316	4096/159+ More if the SFRs are outside of the access bank.
Max Program	8Kx14	32Kx14 16K is likely the largest device	1Mx16
FSRs	1	2 can access Program Memory	3
Instruction Count	35	49	75 83 including the optional extended instructions
Stack	8	16 with over/under flow reset	31 with over/under flow reset
Interrupts	1	1 hardware context save	2 optional hardware context save
Program Memory Read	All devices via RETLW. Some devices via EEPROM interface	All devices via RETLW or FSR. All devices via EEPROM interface.	All devices via TABLRD instructions.

Новая карта Памяти Данных

- 32 банков регистров
- 15 банков зарезервированы для будущего

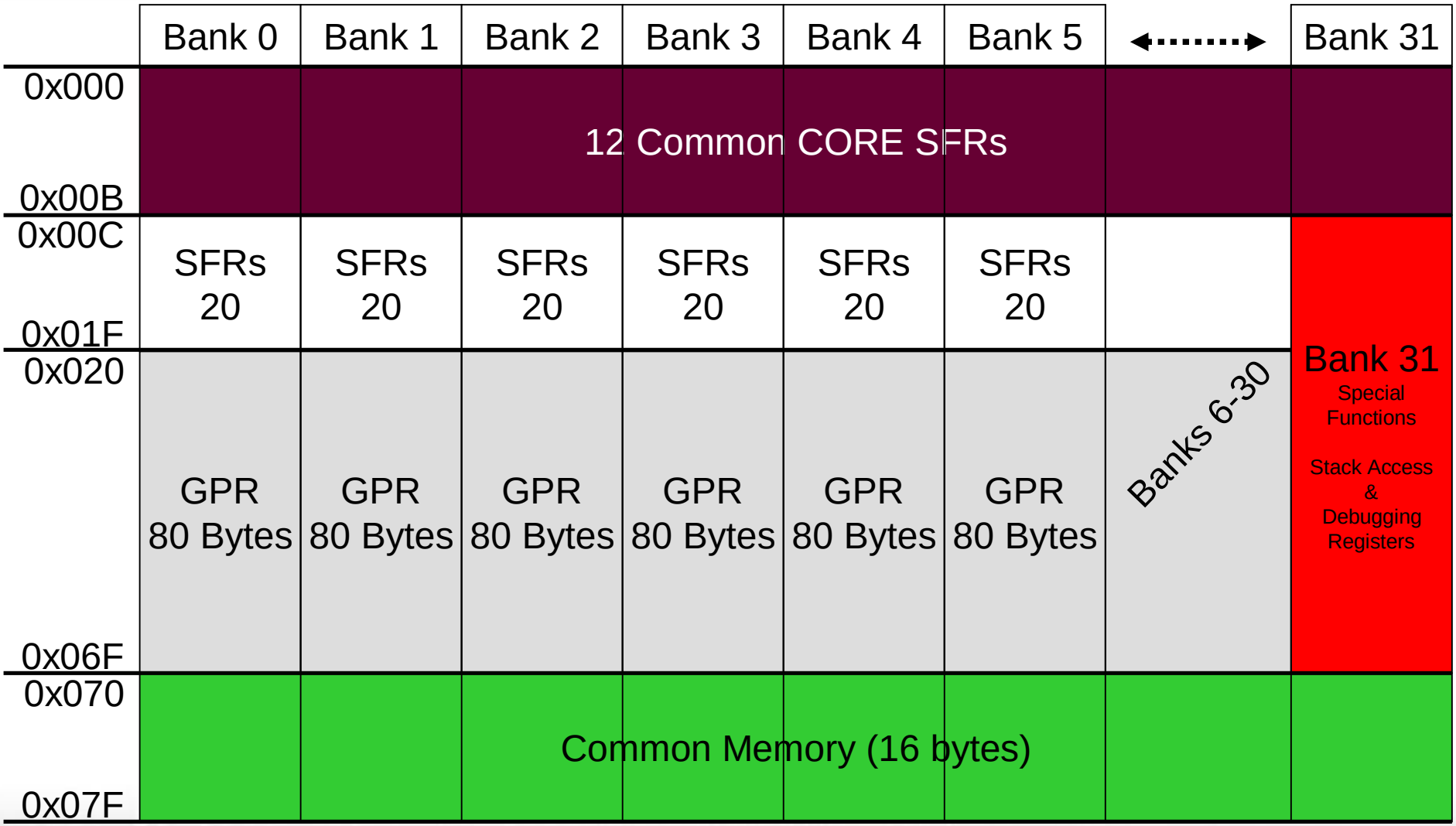
Распределение памяти:

- Нижние 16 байт каждого банка - общие
- Первые 12 байт каждого банка для регистров ЦПУ
- SFR расположены по адресам 12-31 в каждом банке

Новые особенности

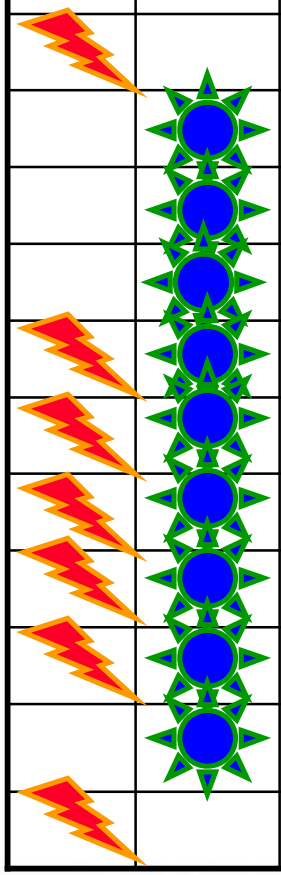
- Рабочий регистр W адресуется как “wreg”
- Банки 16-30 зарезервированы для будущего
- 31-й банк имеет особые функции

Карта Памяти Данных



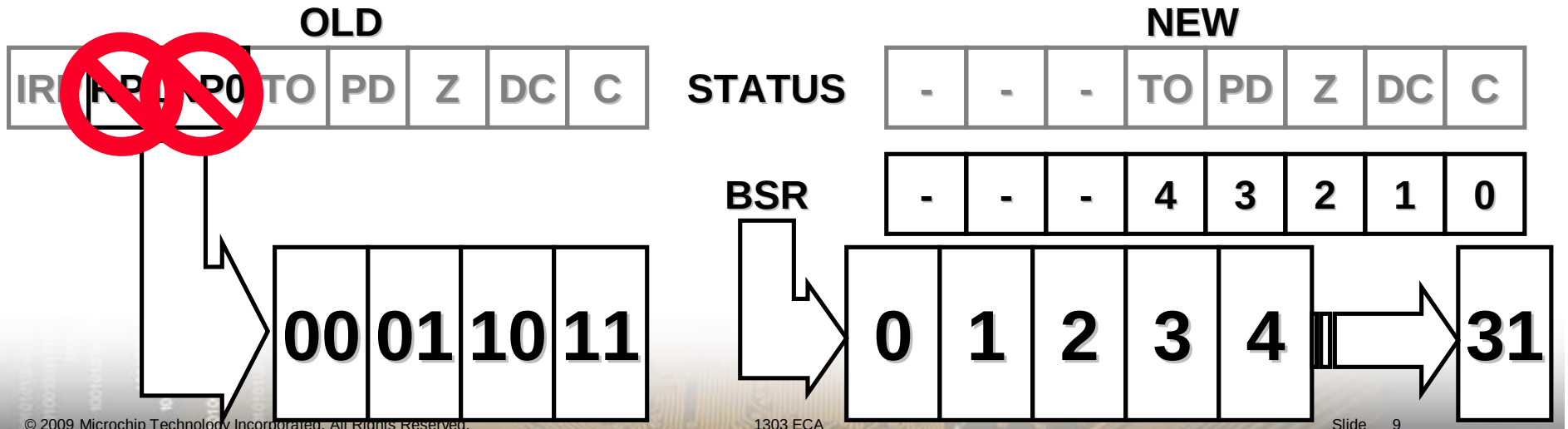
Общие регистры каждого банка

New	Saved	Address	Register	Function
		0x00	INDF0	Indirect Register 0
		0x01	INDF1	Indirect Register 1
		0x02	PCL	Program Counter Low
		0x03	STATUS	Status Register
		0x04	FSR0 Low	File Select Register 0 Low Byte
		0x05	FSR0 High	File Select Register 0 High Byte
		0x06	FSR1 Low	File Select Register 1 Low Byte
		0x07	FSR1 High	File Select Register 1 High Byte
		0x08	BSR	Bank Select Register
		0x09	WREG	Working Register
		0x0A	PCLATH	Program Counter Latch High
		0x0B	INTCON	Interrupt Control Register



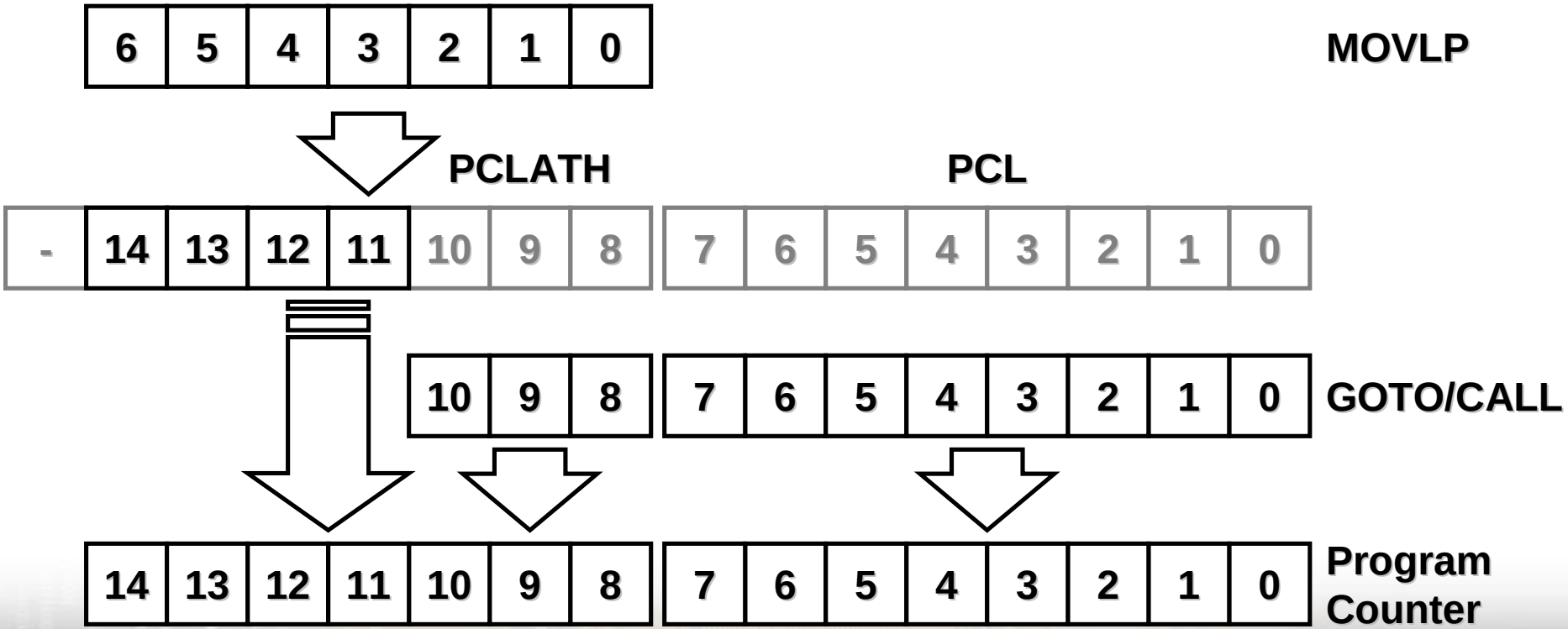
Банки памяти данных

- Старое ядро требует переключения банков через RP0 и RP1 в регистре Status
- Этих битов больше НЕТ!
- Теперь банки переключаются через регистр BSR
- Новая команда MOVLB выбирает банк за один цикл



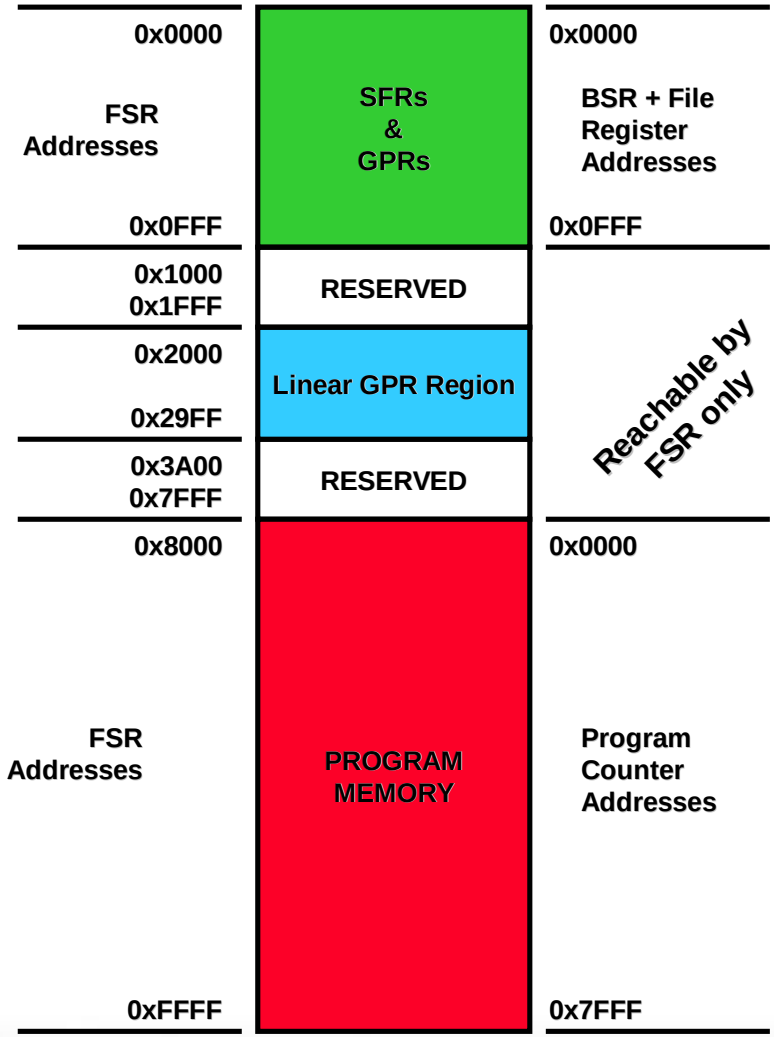
Память программ

- Память программ расширена до 16 страниц по 2 Кбайт
- Переключение банков командой MOVLP



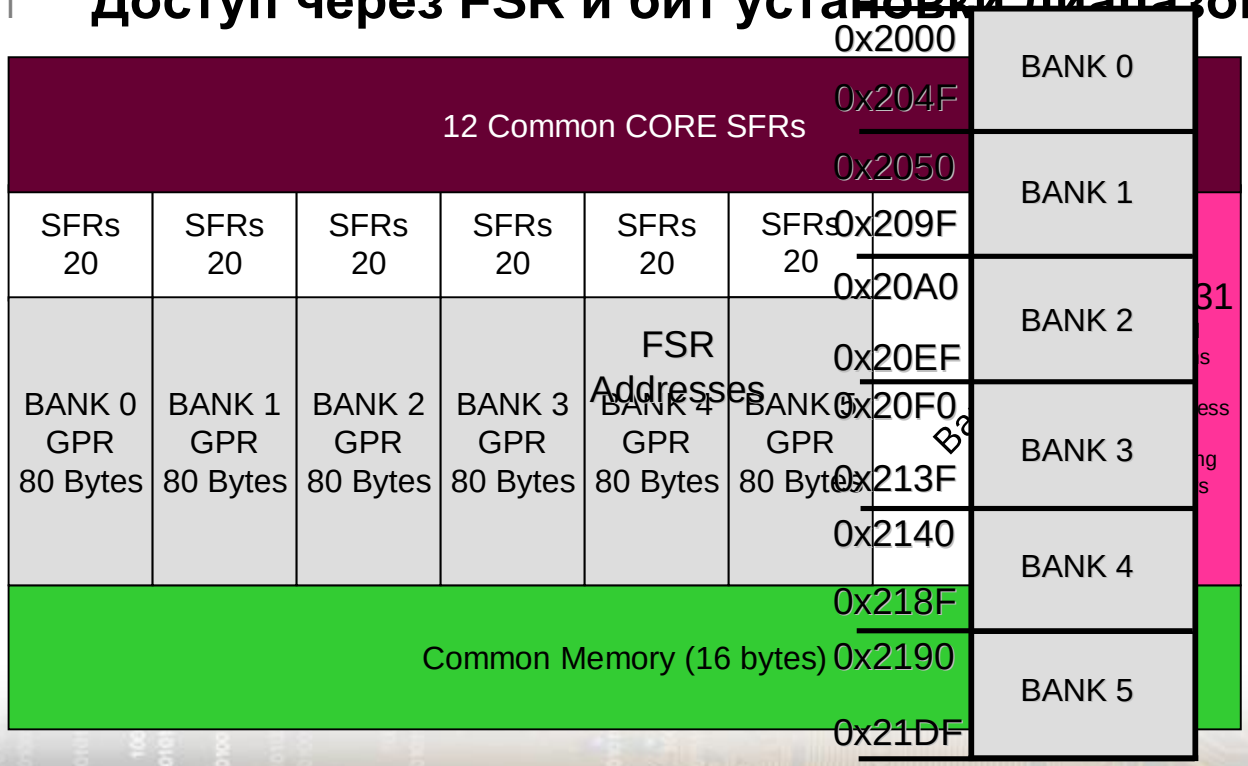
Новый FSR

- Два 16-битных FSR
- FSR может адресовать все регистры И память программ
- Новые FSR позволяют использовать один указатель для всей памяти
- FSR поддерживается новыми инструкциями



Линейная адресация GPR

- Отображает блоки по 80 байт GPR в линейный массив
- Сохраняет работу FSR внутри области GPR
- Позволяет организовать большие Стек, массивы, буферы и т.п.
- Доступ через FSR и бит установки диапазона адресов

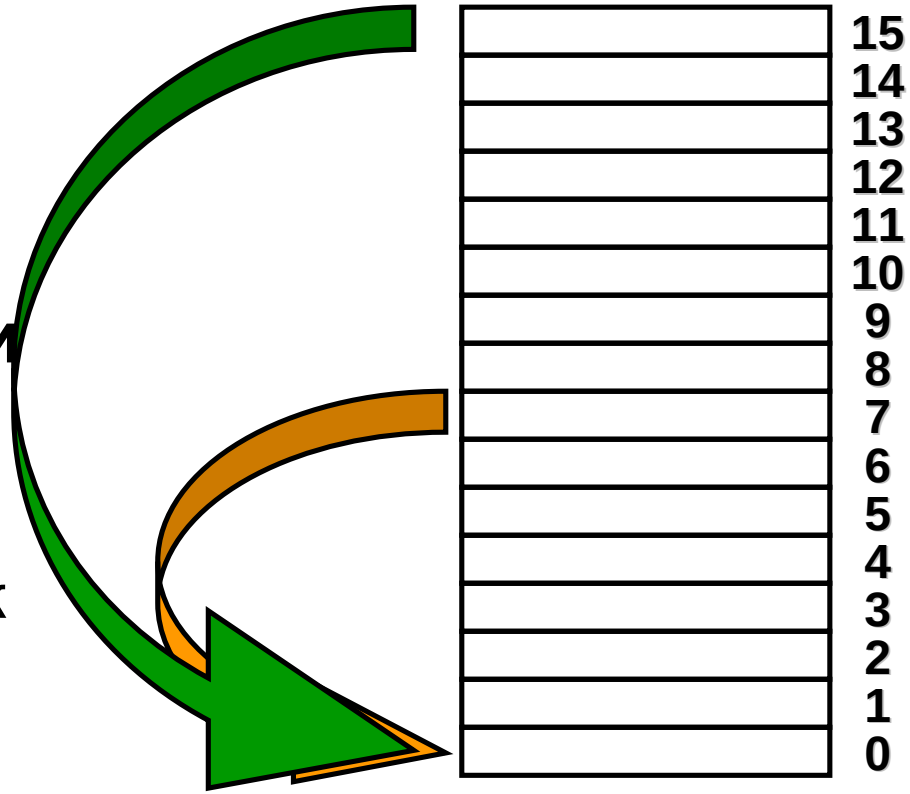


Быстрое сохранение контекста

- **Прерывание автоматически сохраняет контекст**
  - W
  - STATUS
  - BSR
  - FSRs
  - PCLATH
- **RETFIE автоматически восстанавливает контекст**
- **Вы не можете отменить сохранение контекста**
  - Но можете поработать с сохраненным контекстом

Стек

- **16-и уровневый стек**
- **Сброс по переполнению/опустошению (опц.)**
- **Пользовательский и ICD Стек доступен в Банке 31**
  - Чтение/Запись в стек в Банке 31
    - **Применимо для RTOS или при отладке кода**



Режим Сброса Стека
Stack Reset Mode

- **Бит в слове конфигурации STRVEN разрешает Режим Сброса Стека**
- **Сброс контроллера по Стеку происходит:**
  - Если выполняется Return когда стек пустой
  - Если выполняется Call или Прерывание при заполненном Стеке

Чтение вершины стека когда Стек пустой возвращает 0

Нормальный Режим
Normal Mode

- **Стек работает точно так же как и в старых контроллерах, плюс дополнительные возможности:**
  - 16-ть уровней стека
  - Доступ к стеку через STKPTR & TOSH/TOSL

Новые команды

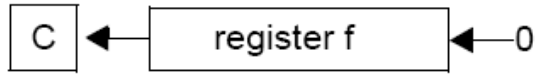
- ADDWFC – Add W+F with Carry
- SUBWFB – Subtract F-W with Borrow
- LSLF – Logical Shift Left
- LSRF – Logical Shift Right
- ASRF – Arithmetic Shift Right
- MOVLP – Move Literal to PCLATH
- MOVLB – Move Literal to BSR
- BRA – Branch Relative (signed)
- BRW – Branch PC + W (unsigned)
- CALLW – Call PCLATH:W
- ADDFSR – Add Literal to FSRn (signed)
- MOVIW – Move indirect to W
- MOVWI – Move W to Indirect
- RESET – Reset Hardware & Software

Арифметические с Переносом

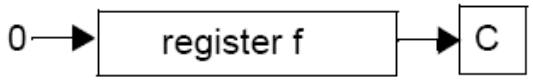
- **ADDWFC**
  - Сложение с учетом переноса
- **SUBWFB**
  - Вычитание с заемом
- **Не поддерживаются операции с константами (Literal) и переносом/заемом**

New: Арифметический Сдвиг

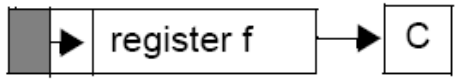
- **LSLF, LSRF, ASRF**
  - Логический сдвиг влево



    - **сдвиг влево, MSB в флаг переноса, LSB = 0**
    - **То же самое что и арифметический сдвиг влево**



  - Логический сдвиг вправо
    - **сдвиг вправо, MSB = 0, LSB LSB в флаг переноса**



  - Арифметический сдвиг вправо
    - **сдвиг вправо, знак переходит в MSB, LSB в флаг переноса**

New: Страницы/Банки

- **MOVLP**
  - Помещает 7-bit константу в PCLATH

    MOVLP HIGH *LABEL*
  - PAGESEL за 1 цикл без изменения W
  - MOVLP + CALL/GOTO занимает 3 цикла и 2 команды, НО работает В ЛЮБОМ МЕСТЕ ПАМЯТИ ПРОГРАММ
- **MOVLB**
  - Помещает 5-bit константу в BSR
  - BANKSEL за 1 цикл без изменения W для ЛЮБОГО банка
  - Биты *IRP, RP0, RP1 больше не существуют*

NEW: Относительные Переходы

- **Относительные переходы позволяют осуществлять переходы независимо от Страничной организации памяти.**
- **BRA N**
  - Всегда переход на PC + N (знаковый)
  - Диапазон -256 <= N <= 255
  - PC + N это 15-bit математика – нет сложностей со страницами.
- **BRW**
  - Всегда переход на PC + W (беззнаковый)
  - Быстрая работа с таблицами / Автомат состояний
  - PC + W это 15-bit математика – нет сложностей со страницами.
- **CALLW**
  - Вызов (Call) с адресом = PCLATH : W
  - Быстрая работа с таблицами / Автомат состояний / указатели на функции
  - PCLATH:W прямой адрес

NEW: Команды косвенной адресации

- **Команда ADDFSR**
  - Добавляет знаковую константу к выбранному FSR
  - Диапазон от -32 до +31
- **MOVIW / MOVWI – Перенос из косвенно адресуемого регистра в W и Перенос W в косвенно адресуемый регистр**
  - Специальные режимы
    - **Pre/Post Increment**
    - **Pre/Post Decrement**
    - **Относительный сдвиг**
      - Диапазон от -32 до +31

Синтаксис MOVIW / MOVWI

- **Standard**
  `MOVWI 0[INDF0]`
- **Pre Increment**
  `MOVIW ++INDF0`
- **Post Increment**
  `MOVWI INDF0++`
- **Pre Decrement**
  `MOVIW --INDF0`
- **Post Decrement**
  `MOVWI INDF0--`
- **Offset**
  `MOVWI k[INDF0]`
  -32 <= k <= 31

- Модифицируют FSR
- Не меняют FSR

NEW: Дополнительные команды

- **RESET**
  - Не нужен GOTO 0
  - Сбрасывается вся периферия
  - Программная версия сброса по MCLR
  - Бит PCON доступен для определения программного сброса
- **Чтение Программной Памяти (PMR) и Запись аналогична записи в EEPROM**
  - Есть во всех контроллерах
- **Device/Revision ID, User ID и Config Words теперь могут быть прочтены программно**

Переход на новые контроллеры

- Страницы
- Банки
- Прерывания
- Косвенная адресация

Страницы

- **Используйте макрос PAGESEL или**
  - Применяйте команду MOVLP
- **Обновите весь код работы с PCLATH**
- **Перейдите на относительные переходы**
  - Устранит большинство проблем со страницами

PAGESEL

My ASSEMBLY Code

```
My_Function
  movlw 0x04
  movwf  delay_cntr
My_function_loop
  decfsz  delay_cntr
  goto My_function_loop
  return

Main
  ...
  PAGESEL My_Function
  call My_Function
  ...

  end
```

PAGESEL MACRO PIC16

```
My_Function
  movlw 0x04
  movwf  delay_cntr
My_function_loop
  decfsz  delay_cntr
  goto My_function_loop
  return

Main
  ...
  movlw high My_Function
  movwf PCLATH
  call My_Function
  ...
  end
```

PAGESEL MACRO ENHANCED PIC16

```
My_Function
  movlw 0x04
  movwf  delay_cntr
My_function_loop
  decfsz  delay_cntr
  goto My_function_loop
  return

Main
  ...
  movlp high My_Function
  call My_Function
  ...

  end
```

Банки

- **Используйте макрос BANKSEL или**
  - Применяйте команду MOVLB
- **Замените запись в регистр STATUS (RP0, RP1) записью в BSR**

BANKSEL

My ASSEMBLY Code

```
 data
Var1 res 1
Var2 res 1
Var3 res 1

 code

Main
   ...
   BANKSEL Var1
   addwf  Var1
   ...

   end
```

BANKSEL MACRO PIC16

```
 data
Var1 res 1
Var2 res 1
Var3 res 1

 code

Main
   ...
   bsf STATUS,RP0
   bcf STATUS,RP1
   addwf  Var1
   ...

   end
```

BANKSEL MACRO ENHANCED PIC16

```
 data
Var1 res 1
Var2 res 1
Var3 res 1

 code

Main
   ...
   movlb Var1 >> 7
   addwf  Var1
   ...

   end
```

Сохраняет 1 команду
И доступ к нужному банку

Always works

Прерывания

- RETFIE работает с небольшими отличиями
- Убедитесь что прерывания не возвращают параметр в W (это плохая практика)
- Уберите код с сохранением и восстановлением контекста

Косвенное обращение к памяти

- IRP не существует
- Доступ к более чем 256 байт требует обновление старшей части FSR<x>H
- Быстрый метод обновления регистра FSR<x>H требует модификации W
- BANKISEL выполняет несколько команд BCF и BSF

Переход на новые контроллеры. Итог

- Переход на PIC16F1XXX это очень просто
- Обратный переход с PIC16F1XXX может быть сложным
- Применение макросов BANKSEL и PAGESEL будет большим подспорьем

Новые программные Трюки

- Относительные переходы
- Табличное чтение
- 16-и разр. арифметика
- Увеличение надежности/устойчивости

Относительные переходы

ORIGINAL ASSEMBLY Code

```
Additional Code
My_Function
   movlw 0x04
   movwf  delay_cntr
My_function_loop
   decfsz  delay_cntr
   goto My_function_loop
   return

Main
   ...
   call My_Function
   ...
```

Граница страницы в этом месте будет требовать применения PAGESEL в My_function_loop

NEW ASSEMBLY ENHANCED PIC16

```
My_Function
   movlw 0x04
   movwf  delay_cntr
My_function_loop
   decfsz  delay_cntr
   bra My_function_loop
   return

Main
   ...
   PAGESEL My_Function
   call My_Function
   ...
   end
```

Относительный переход делает этот код рабочим для любой страницы

No relative CALL support. CALLW is not relative.

Табличное чтение

- **PIC16F1XXX имеет новые методы доступа к таблицам в ROM**
  - FSR
  - Относительные переходы

Таблицы с использованием FSR

- **Длинный код предустановки**

```
The_CODE
        movlw  high Table_start
        movwf  FSR0H
        movlw  low Table_start
        movwf  FSR0L
        movlw  3
        addwf  FSR0L
        movf   INDF0,w


Table_start
        DT 3,4,5,6,7,8,9
```

Быстрая работа с таблицами

- Этот код возвращает константу из таблицы (выровненную по границам в 256 слов)

```
Table_Function
        movlw  high Table_start
        movwf  PCLATH
        movlw  3
        movwf  PCL
Table_start
        DT 3,4,5,6,7,8,9
```

```
The_CODE
        movlp  high Table_start
        movlw  3
        callw


Table_start
        DT 3,4,5,6,7,8,9
```

Более быстрый способ

- Возвращает константу из таблицы
- НЕТ НЕОБХОДИМОСТИ в выравнивании
- Начало таблицы может располагаться где угодно

```
The_CODE
        movlw  high Table_start
        movwf  PCLATH
        movlw  low Table_start
        addwf  3
        btfss  STATUS,C
        incf   PCLATH,f
        movwf  PCL
Table_start
        DT 3,4,5,6,7,8,9
```

```
The_CODE
        movlp  high Table_start
        movlw  3
        brw
Table_start
        DT 3,4,5,6,7,8,9
```

16-и разрядная Арифметика

- **Новые команды ускоряют работу 16-bit арифметики**

```
movf   val_a_l, W
addwf  val_b_l, F
btfsc  STATUS,  C
incf   val_b_h, F
movf   val_a_h, W
addwf  val_b_h, F
```

```
movf   val_a_l, W
addwf  val_b_l, F
movf   val_a_h, W
addwfc val_b_h, F
```

Увеличение надежности/устойчивости

- Применение команды RESET
- Сброс по переполнению / опустошению Стека

Для Продвинутых

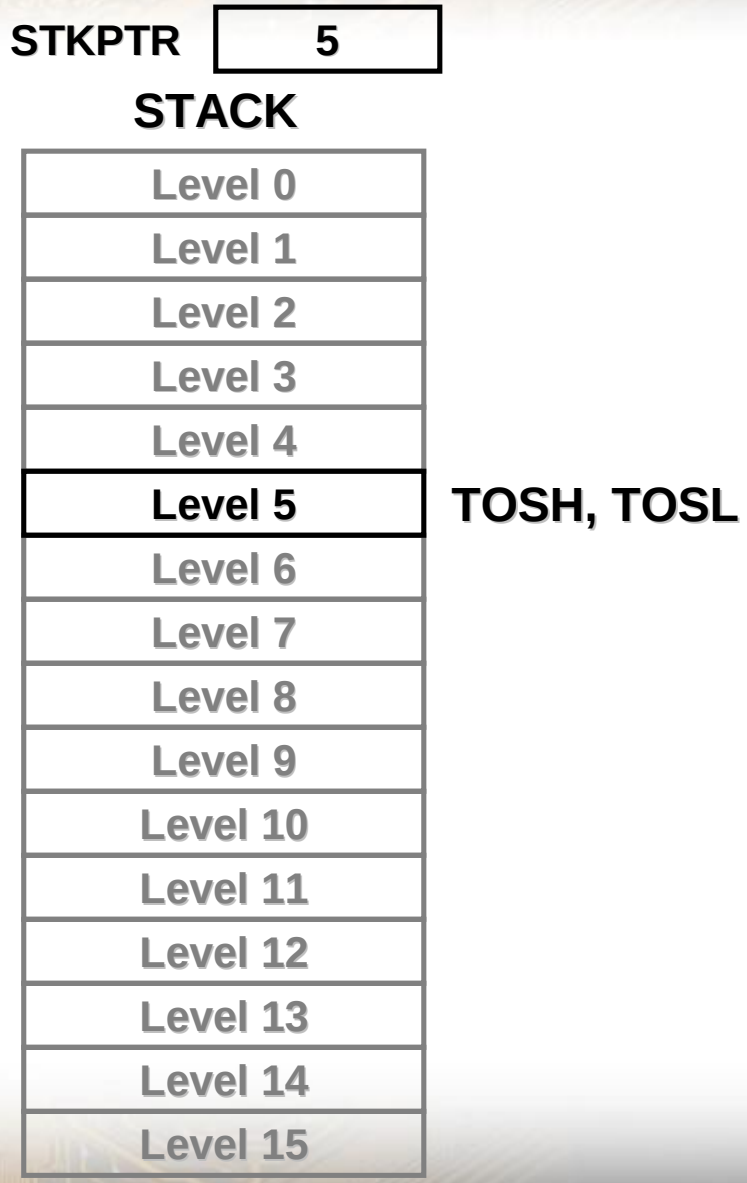
- Доступ к Стеку
- Доступ к сохраненному контексту
- Чтение Device ID
- Приоритетная многозадачность
- Диагностика ошибок

Доступ к Стеку

- Стек доступен через регистры TOS и STKPTR
- STKPTR это текущее значение указателя стека
- TOS указатель на вершину
- Оба регистра доступны для чтения и записи
- TOS состоит из TOSH и TOSL - 15-битное значение счетчика команд (PC)

Доступ к Стеку

- Регистр STKPTR указывает на текущую позицию стека
- TOSH,TOSL это содержимое стека, на которое указывает STKPTR
- Изменение STKPTR изменит TOSH,TOSL
- STKPTR – 5-и битный



Доступ к контексту

- **Регистры сохраненного контектса при входе в прерывание доступны для чтения и записи в 31-м банке**

STATUS	STATUS_SHAD
FSR0 Low	FSR0L_SHAD
FSR0 High	FSR0H_SHAD
FSR1 Low	FSR1L_SHAD
FSR1 High	FSR1H_SHAD
BSR	BSR_SHAD
WREG	WREG_SHAD
PCLATH	PCLATH_SHAD

Device ID

- Некоторые регистры в области конфигурации имеют тот же способ доступа, что и EEPROM
- User ID, Device/Revision ID и Слово конфигурации могут быть прочтены программно.

Приоритетная многозадачность

- Доступ к стеку и контексту позволяет прерыванию заменить текущую задачу другой задачей
- Это позволяет упростить создание RTOS для новых контроллеров.

Диагностика ошибок

- Доступ к стеку и к сохраненному контексту позволяет осуществлять самодиагностику программы
- Проверка стека для безопасных и критических применений

PIC16F1937

- **Наш первый «всем напичканный» контроллер с новым расширенным ядром**
- **Другие контроллеры:**
  - PIC16F193x & PIC16F194x – больше выводов, больше памяти, больше периферии и т.д.
  - PIC16F182x – мало выводов, 8-выводов у PIC12F1822

Дополнительная литература

- **Класс 1304 ECP**
  - Обзор и использование периферии контроллеров с расширенным ядром PIC16F1937
- **DS41364A**
  - PIC16F1937 Data Sheet
- **DS41375A**
  - PIC1xF1xxx Software Migration Document

Вопросы?

Trademarks