PIC18FXX2

Однокристальные 8-разрядные FLASH CMOS микроконтроллеры с 10 – разрядным АЦП компании Microchip Technology Incorporated

- **PIC18F242**
- **PIC18F252**
- **PIC18F442**
- **PIC18F452**

Часть 11
(Описание системы команд PIC18FXX2)

Перевод основывается на технической документации DS39564A компании Microchip Technology Incorporated, USA.

PIC18FXX2
Data Sheet
High Performance, Enhanced FLASH Microcontrollers with 10-Bit A/D

28/40-выводные высокоскоростные FLASH микроконтроллеры с 10-разрядным АЦП

Высокоскоростной RISC микроконтроллер:

- Оптимизированная архитектура и система команд для написания программ на языке C
- Система команд совместима с командами семейств PIC16C, PIC17C и PIC18C
- Линейное адресное пространство памяти программ 32кбайта
- Линейное адресное пространство памяти данных 1.5кбайт

Устройство	Память программ		Память данных (байт)	EEPROM память данных (байт)
	Flash (байт)	Команд		
PIC18F242	16к	8192	768	256
PIC18F252	32к	16384	1536	256
PIC18F442	16к	8192	768	256
PIC18F452	32к	16384	1536	256

- Быстродействие до 10MIPS:
  - Тактовая частота от DC до 40МГц
  - Частота генератора с вкл. PLL от 4МГц до 10МГц
- 16-разрядные команды, 8-разрядные данные
- Система приоритетов прерываний
- Аппаратное умножение 8x8 за один машинный цикл

Характеристика периферийных модулей:

- Высокая нагрузочная способность портов ввода/ вывода
- Три входа внешних прерываний
- Модуль TMR0: 8/16-разрядный таймер/счетчик с программируемым 8-разрядным предделителем
- Модуль TMR1: 16-разрядный таймер/счетчик
- Модуль TMR2: 8-разрядный таймер/счетчик с 8-разрядным регистром периода (основной для ШИМ)
- Модуль TMR3: 16-разрядный таймер/счетчик
- Вторичный генератор тактового сигнала на основе TMR1/TMR3
- Два модуля CCP
  - Выводы модуля CCP могут работать как:
  - 16-разрядный захват, максимальная разрешающая способность 6.25нс (TCY/16)
  - 16-разрядное сравнение, максимальная разрешающая способность 100нс (TCY)
  - ШИМ, разрядность от 1 до 10 бит, Максимальная частота ШИМ 156кГц@8 бит; 39кГЦ@10 бит

Характеристика периферийных модулей (продолжение):

- Модуль ведущего последовательного синхронного порта (MSSP)
  - 3-х проводной интерфейс SPITM (поддерживает 4 режима)
  - I2CTM (ведущий и ведомый режим)
- Адресуемый модуль USART, поддержка интерфейса RS-485 и RS-232
- Модуль PSP, ведомый параллельный порт

Аналоговые периферийные модули:

- Модуль 10-разрядного АЦП:
  - Высокая скорость преобразования
  - Работа модуля АЦП в SLEEP режиме микроконтроллера
  - DNL = ±1Lsb, INL = ±1Lsb
- Программируемый детектор пониженного напряжения (PLVD)
  - При обнаружении снижения напряжения возможна генерация прерываний
- Программируемый сброс по снижению напряжения питания

Особенности микроконтроллеров

- 100 000 гарантированных циклов стирание/запись памяти программ
- 1 000 000 гарантированных циклов стирание/запись EEPROM памяти данных
- Возможность самопрограммирования
- Сброс по включению питания (POR), таймер включения питания (PWRT), таймер запуска генератора (OST)
- Сторожевой таймер WDT с отдельным RC генератором
- Программируемая защита кода программы
- Режим пониженного энергопотребления и режим SLEEP
- Выбор режима работы тактового генератора, включая:
  - 4 x PLL (от основного генератора)
  - Вторичный генератор (32кГц)
- Внутрисхемное программирование по двухпроводной линии (ICSP) с одним напряжением питания 5В
- Внутрисхемная отладка по двухпроводной линии (ICD)

КМОП технология

- Высокоскоростная энергосберегающая КМОП технология
- Полностью статическая архитектура
- Широкий диапазон напряжений питания (от 2.0В до 5.5В)
- Промышленный и расширенный температурные диапазоны

Содержание

20. Описание системы команд

Набор команд PIC18FXXX несколько расширен по сравнению с предыдущими версиями PICmicro, но обеспечивает легкость переноса программы, написанной для микроконтроллеров PICmicro среднего семейства.

Большинство команд занимают одно слово в памяти программ (16 бит), но есть 4 команды, для которых необходимо два слова в памяти программ.

Каждая команда состоит из одного 16-разрядного слова, разделенного на код операции (OPCODE), определяющий тип команды и один или несколько операндов, определяющие операцию команды.

Система команд ортогональна, все команды разделена на четыре основных группы:

- **Байт-ориентированные** команды
- **Бит-ориентированные** команды
- Операции с **константами**
- **Управляющие** команды

Сводный список команд PIC18FXXX смотрите в таблице 20-2. Описание полей команды и дескрипторов смотрите в таблице 20-1.

Байт-ориентированные команды имеют следующие операнды:

- Регистр (‘f’), к которому выполняется обращение
- Размещение результата команды (‘d’)
- Доступ к памяти (‘a’)

Для байт-ориентированных команд 'f' является указателем регистра, а 'd' указателем адресата результата. Указатель регистра определяет, какой регистр должен использоваться в команде. Указатель адресата определяет, где будет сохранен результат. Если 'd'=0, результат сохраняется в регистре W. Если 'd'=1, результат сохраняется в регистре, который используется в команде.

Бит-ориентированные команды имеют следующие операнды:

- Регистр (‘f’), к которому выполняется обращение
- Бит в регистре (‘b’)
- Доступ к памяти (‘a’)

В бит-ориентированных командах 'b' определяет номер бита участвующего в операции, а 'f' - указатель регистра, который содержит этот бит.

Команды операций с **константами** могут иметь операнды:

- Константа, которая будет загружена в регистр (‘k’)
- Регистр FSR, чтобы загрузить значение в указываемый регистр
- Нет операндов (‘-‘)

Команды **управления** могут содержать следующие операнды:

- Адрес в памяти программ (‘n’)
- Режим выполнения команд вызова (Call) или возврата (Return) (‘s’)
- Режим выполнения команд табличного чтения/записи (‘m’)
- Нет операндов (‘-‘)

Практически все команды занимают одно слово в памяти программ, кроме четырех команд, занимающих два слова в памяти программ. Эти команды занимают два слова из-за большого числа операндов (команда 32 бита). Во втором слове 4 старших бита всегда равны ‘1’. Если второе слово команды будет выполнено как отдельная команда, то вместо каких-либо операций выполняется пустой цикл NOP.

Все однословные команды выполняются за один машинный цикл, кроме команд, условие которых истинно или в результате исполнения команды изменяется счетчик команд PC. В этом случае команда исполняется за два цикла, дополнительно выполняя пустой цикл NOP.

Двухсловные команды выполняются за два цикла.

Один машинный цикл состоит из 4-х периодов тактового сигнала. Таким образом, при тактовой частоте 4МГц нормальная длительность выполнения команды 1мкс. Если условие команды истинно или изменяется значение счетчика команд, то команда выполняется за два машинных цикла 2мкс.

На рисунке 20-1 представлены общие форматы команд.

Во всех примерах применяется формат чисел “nnh’, чтобы представить шестнадцатеричное число, где h - шестнадцатеричная цифра.

Мнемоника команд, поддерживаемая ассемблером MPASM, показана в таблице 20-2.

Подробное описание каждой команды смотрите в разделе 20.1.

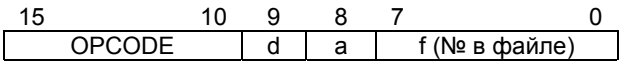
Таблица 20-1. Описание полей команды и дескрипторов

Обозначение	Описание
a	Бит доступа к памяти: a = 0 : Обращение к ОЗУ быстрого доступа (значение регистра BSR игнорируется) a = 1 : Обращение к ОЗУ с учетом значения регистра BSR
bbb	Номер бита в 8-разрядном регистре (от 0 до 7)
BSR	Регистр выбора банка памяти. Используется для выбора текущего банка памяти.
d	Бит размещения результата команды: d = 0 : Результата помещается в регистр WREG d = 1 : Результат сохраняется в регистре ‘f’
dest	Адресат результата: регистр WREG или регистр ‘f’ в ОЗУ
f	8-разрядный адрес регистра (от 0x00 до 0xFF)
fs	12-разрядный адрес регистр источника (от 0x000 до 0xFFF)
fd	12-разрядный адрес регистр приемника (от 0x000 до 0xFFF)
k	Константа или метка (8-, 12- или 20-разрядное значение)
label	Имя метки
mm * *+ *- +*	Режим работы регистр TBLPTR при операциях табличного чтения/записи (может использоваться только в операциях табличного чтения/записи). Значение регистра TBLPTR не изменяется Значение регистра TBLPTR пост - инкрементируется Значение регистра TBLPTR пост - декрементируется Значение регистра TBLPTR пред - инкрементируется
n	Относительный адрес (знаковый) для команд относительного перехода или абсолютный адрес для команд вызова подпрограмм, перехода и возврата
PRODH	Старший байт результата умножения
PRODL	Младший байт результата умножения
s	Бит быстрого вызова подпрограммы/возврата s = 0 : значение дополнительных регистров не используется s = 1 : значение некоторых регистров сохраняется/восстанавливается в дополнительных регистрах (быстрый режим)
u	Не используется или не реализовано
WREG	Рабочий регистр (аккумулятор)
TBLPTR	21-разядный табличный указатель (указатель в памяти программ)
TABLAT	8-разрядная защелка табличного чтения/записи
TOS	Вершина стека
PC	Счетчик команд
PCL	Младший байт счетчика команд
PCH	Старший байт счетчика команд
PCLATH	Защелка старшего байта счетчика команд
PCLATU	Защелка верхнего байта счетчика команд
GIE	Бит глобального разрешения прерываний
WDT	Сторожевой таймер
-TO	Бит переполнения WDT
-PD	Бит включения питания
C, DC, Z, OV, N	Флаги АЛУ: перенос, десятичный перенос, нуль, переполнение, отрицательный результат
[ ]	Опционально
( )	Контекст
$\rightarrow$	Присвоение
< >	Битовое поле
$\in$	Из набора

Рисунок 20-1. Общий формат команд

Байт ориентированные команды с регистрами

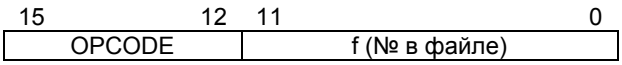
Пример

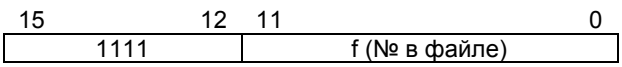


ADDWF MYREG, W, B

d = 0 - результата помещается в регистр WREG
d = 1 - результат сохраняется в регистре ‘f’
a = 0 : Обращение к ОЗУ быстрого доступа
a = 1 : Обращение к ОЗУ с учетом значения регистра BSR
f = 8-разрядный адрес регистра

Команды перемещения **Байт-Байт**

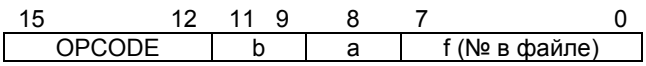




MOVFF MYREG1, MYREG2

f = 12-разрядный адрес регистра

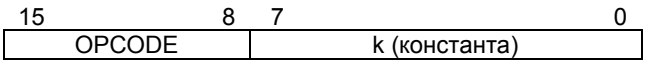
Бит ориентированные операции с регистрами



BSF MYREG, bit, B

b = номер бита в 8-разрядном регистре
a = 0 : Обращение к ОЗУ быстрого доступа
a = 1 : Обращение к ОЗУ с учетом значения регистра BSR
f = 8-разрядный адрес регистра

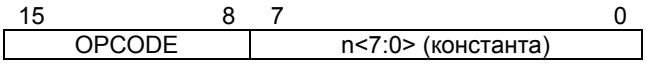
Операции с **константами**

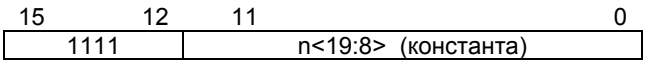


MOVLW 0x7F

k = 8-разрядная константа

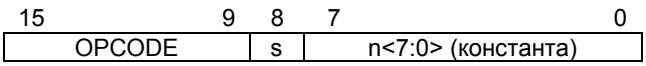
Команды управления

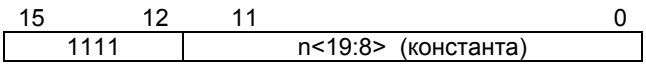




GOTO Label

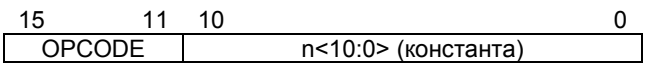
n = 20-разрядная константа



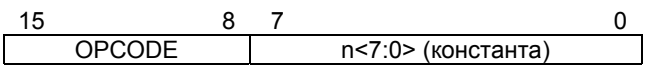


CALL MYFUNC

s = бит быстрого вызова подпрограммы/возврата



BRA MYFUNC



BC MYFUNC

Таблица 20-2. Список команд PIC18FXXX

Мнемоника и операнды		Описание	Циклов	Слово команды (16 бит) Ст.			Мл.	Воздействие на флаги АЛУ	Прим.
Байт ориентированные команды с регистрами									
ADDWF	f, d, a	Сложение WREG и f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Сложение WREG, f и бита C	1	0010	00da	ffff	ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	Логическое И WREG и f	1	0001	01da	ffff	ffff	Z, N	1, 2
CLRF	f, a	Очистка f	1	0110	101a	ffff	ffff	Z	2
COMF	f, d, a	Инверсия f	1	0001	11da	ffff	ffff	Z, N	1, 2
CPFSEQ	f, a	Сравнить WREG и f, проп. если =	1(2 или 3)	0110	001a	ffff	ffff	-	4
CPFSGT	f, a	Сравнить WREG и f, проп. если >	1(2 или 3)	0110	010a	ffff	ffff	-	4
CPFSLT	f, a	Сравнить WREG и f, проп. если <	1(2 или 3)	0110	000a	ffff	ffff	-	4
DECF	f, d, a	Декремент f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1,2,3,4
DECFSZ	f, d, a	Декремент f, пропустить если 0	1(2 или 3)	0010	11da	ffff	ffff	-	1,2,3,4
DCFSNZ	f, d, a	Декремент f, пропустить если не 0	1(2 или 3)	0100	11da	ffff	ffff	-	1,2,3,4
INCF	f, d, a	Инкремент f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1,2,3,4
INCFSZ	f, d, a	Инкремент f, пропустить если 0	1(2 или 3)	0011	11da	ffff	ffff	-	1,2,3,4
INFSNZ	f, d, a	Инкремент f, пропустить если не 0	1(2 или 3)	0100	10da	ffff	ffff	-	1,2,3,4
IORWF	f, d, a	Логическое ИЛИ WREG и f	1	0001	01da	ffff	ffff	Z, N	1, 2
MOVF	f, d, a	Переместить f	1	0101	00da	ffff	ffff	Z, N	1
MOVFF	fs, fd	Переместить fs (источник) 1-е слово в fd (приемник) 2-е слово	2	1100	ffff	ffff	ffff	-	
				1111	ffff	ffff	ffff		
MOVWF	f, a	Переместить WREG в f	1	0110	111a	ffff	ffff	-	
MULWF	f, a	Умножение WREG и f	1	0000	001a	ffff	ffff	-	
NEGF	f, a	Негативное значение f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	1, 2
RLCF	f, d, a	Сдвиг влево через перенос	1	0011	01da	ffff	ffff	C, Z, N	
RLNCF	f, d, a	Сдвиг влево без переноса	1	0100	01da	ffff	ffff	Z, N	1, 2
RRCF	f, d, a	Сдвиг вправо через перенос	1	0011	00da	ffff	ffff	C, Z, N	
RRNCF	f, d, a	Сдвиг вправо без переноса	1	0100	00da	ffff	ffff	Z, N	1, 2
SETF	f	Установить все биты f	1	0110	100a	ffff	ffff	-	
SUBFWB	f, d, a	Вычитание f из WREG с заемом	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
SUBWF	f, d, a	Вычитание WREG из f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	
SUBWFB	f, d, a	Вычитание WREG из f с заемом	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	1, 2
SWAPF	f, d, a	Поменять местами полубайты в f	1	0011	10da	ffff	ffff	-	4
TSTFSZ	f, a	Тест f, пропустить если 0	1(2 или 3)	0110	011a	ffff	ffff	-	1, 2
XORWF	f, d, a	Лог. исключающее ИЛИ WREG и f	1	0001	10da	ffff	ffff	Z, N	1, 2
Бит ориентированные операции с регистрами									
BCF	f, b, a	Сброс бита в f	1	1001	bbba	ffff	ffff	-	1, 2
BSF	f, b, a	Установка бита в f	1	1000	bbba	ffff	ffff	-	1, 2
BTFSC	f, b, a	Тест бита, пропустить если ‘0’	1(2 или 3)	1011	bbba	ffff	ffff	-	3, 4
BTFSS	f, b, a	Тест бита, пропустить если ‘1’	1(2 или 3)	1010	bbba	ffff	ffff	-	3, 4
BTG	f, b, a	Инверсия бита в f	1	0111	bbba	ffff	ffff	-	1, 2

Примечания:

1. При выполнении операции «чтение-модификация-запись» с портом ввода вывода (например, MOVF PORTB, 1, 0) исходное значение считывается с выводов порта, а не из выходных защелок. Например, в защелке данных записано ‘1’, а вывод настроен как вход и на этом входе сигнал с уровнем ‘0’, обратно в защелку будет записано значение ‘0’.
2. При записи в TMR0 (и в команде бит d = 1) предделитель TMR0 сбрасывается, если он подключен к TMR0.
3. Если условие истинно, или изменяется счетчик команд PC, то команда выполняется за два цикла. Во втором цикле выполняется NOP.
4. Некоторые команды состоят из двух 16-разрядных слов. Если по каким-то причинам счетчик команд попадет на 2-е слово команды, то оно будет выполнено как NOP.
5. Если производится запись во внутреннюю память, то следующая команда не начнет выполняться до тех пор, пока не закончится цикл записи.

Таблица 20-2. Список команд PIC18FXXX (продолжение)

Мнемоника и операнды		Описание	Циклов	Слово команды (16 бит) Ст.			Мл.	Воздействие на флаги АЛУ	Прим.
Команды управления									
BC	n	Переход, если перенос (C = 1)	1(2)	1110	0010	nnnn	nnnn	-	
BN	n	Переход, если нег. результ. (N = 1)	1(2)	1110	0110	nnnn	nnnn	-	
BNC	n	Переход, если нет переноса (C = 0)	1(2)	1110	0011	nnnn	nnnn	-	
BNN	n	Переход, если пол. результ. (N = 0)	1(2)	1110	0111	nnnn	nnnn	-	
BNOV	n	Переход, если нет переполн. (OV = 0)	1(2)	1110	0101	nnnn	nnnn	-	
BNZ	n	Переход, если не нуль (Z = 0)	2	1110	0001	nnnn	nnnn	-	
BOV	n	Переход, если переполнение (OV = 1)	1(2)	1110	0100	nnnn	nnnn	-	
BRA	n	Безусловный переход	1(2)	1101	0nnn	nnnn	nnnn	-	
BZ	n	Переход, если нуль (Z = 1)	1(2)	1110	0000	nnnn	nnnn	-	
CALL	n, s	Переход на подпрограмму. 1-е слово	2	1110	110s	kkkk	kkkk	-	
		2-е слово		1111	kkkk	kkkk	kkkk		
CLRWDT	-	Сбросить сторожевой таймер	1	0000	0000	0000	0100	-TO, -PD	
DAW	-	Десятичная коррекция WREG	1	0000	0000	0000	0111	C	
GOTO	n	Переход по адресу, 1-е слово	2	1110	1111	kkkk	kkkk	-	
		2-е слово		1111	kkkk	kkkk	kkkk		
NOP	-	Нет операции	1	0000	0000	0000	0000	-	
NOP	-	Нет операции	1	1111	xxxx	xxxx	xxxx	-	4
POP	-	Чтение вершины стека возврата TOS	1	0000	0000	0000	0110	-	
PUSH	-	Запись в вершину стека возврата TOS	1	0000	0000	0000	0101	-	
RCALL	n	Короткий переход на подпрограмму	2	1101	1nnn	nnnn	nnnn	-	
RESET	-	Программный сброс	1	0000	0000	1111	1111	все	
RETFIE	s	Возврат из пп с разреш. прерываний	2	0000	0000	0001	000s	GIEH/GIEL	
RETLW	k	Возврат из пп с загрузкой WREG	2	0000	1100	kkkk	kkkk	-	
RETURN	s	Возврат из подпрограммы	2	0000	0000	0001	001s	-	
SLEEP	-	Переход в SLEEP режим	1	0000	0000	0000	0011	-TO, -PD	
Операции с константами									
ADDLW	k	Прибавить константу к WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW	k	Логическое И константы и WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	Логическое ИЛИ константы и WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f, k	Поместить константу (12 бит) в	2	1110	1110	00ff	kkkk	-	
		FSR (2 слова)		1111	0000	kkkk	kkkk		
MOVLB	k	Поместить константу в BSR<3:0>	1	0000	0001	0000	kkkk	-	
MOVLW	k	Поместить константу в WREG	1	0000	1110	kkkk	kkkk	-	
MULLW	k	Умножение константы на WREG	1	0000	1101	kkkk	kkkk	-	
RETLW	k	Возврат из пп с загрузкой WREG	2	0000	1100	kkkk	kkkk	-	
SUBLW	k	Вычитание WREG из константы	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Лог. исключ. ИЛИ константы и WREG	1	0000	1010	kkkk	kkkk	Z, N	
Операции память данных ↔ память программ									
TBLRD*		Табличное чтение	2	0000	0000	0000	1000	-	
TBLRD*+		Табличное чтение с пост-инкрементом	2	0000	0000	0000	1001	-	
TBLRD*-		Табличное чтение с пост-декрементом	2	0000	0000	0000	1010	-	
TBLRD+*		Табличное чтение с пред-инкрементом	2	0000	0000	0000	1011	-	
TBLWT*		Табличная запись	2	0000	0000	0000	1100	-	5
TBLWT*+		Табличная запись с пост-инкрементом	2	0000	0000	0000	1101	-	5
TBLWT*-		Табличная запись с пост-декрементом	2	0000	0000	0000	1110	-	5
TBLWT+*		Табличная запись с пред-инкрементом	2	0000	0000	0000	1111	-	5

Примечания:

1. При выполнении операции «чтение-модификация-запись» с портом ввода вывода (например, MOVF PORTB, 1, 0) исходное значение считывается с выводов порта, а не из выходных защелок. Например, в защелке данных записано ‘1’, а вывод настроен как вход и на этом входе сигнал с уровнем ‘0’, обратно в защелку будет записано значение ‘0’.
2. При записи в TMR0 (и в команде бит d = 1) предделитель TMR0 сбрасывается, если он подключен к TMR0.
3. Если условие истинно, или изменяется счетчик команд PC, то команда выполняется за два цикла. Во втором цикле выполняется NOP.
4. Некоторые команды состоят из двух 16-разрядных слов. Если по каким-то причинам счетчик команд попадет на 2-е слово команды, то оно будет выполнено как NOP.
5. Если производится запись во внутреннюю память, то следующая команда не начнет выполняться до тех пор, пока не закончится цикл записи.

20.1 Подробное описание команд

ADDLW Прибавить константу к WREG

Синтаксис: [*label*] ADDLW k

Операнды: $0 \le k \le 255$

Операция: $(W) + k \rightarrow W$

Измен. флаги: N, OV, C, DC, Z

Код:

0000	1111	kkkk	kkkk

Описание: Содержимое регистра W складывается с 8-разрядной константой ‘k’, результат сохраняется в регистр W

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘k’	Выполнение	Запись в регистр W

Пример:

ADDLW 0x15

До выполнения команды

W = 0x10

После выполнения команды

W = 0x25

ADDWF Сложение WREG и f

Синтаксис: [*label*] ADDWF f[,d[,a]]

Операнды: $0 \le f \le 255$

$d \in [0,1]$

$a \in [0,1]$

Операция: $(W) + (f) \rightarrow dest$

Измен. флаги: N, OV, C, DC, Z

Код:

0010	01da	ffff	ffff

Описание: Сложение содержимого регистров W и ‘f’. Если d=0, результат сохраняется в регистре W, если d=1, то в регистре’ f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа. Если a = 1, используется BSR.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись результата

Пример:

ADDWF REG, 0, 0

До выполнения команды

W = 0x17
REG = 0xC2

После выполнения команды

W = 0xD9
REG = 0xC2

ADDWFC Сложение WREG, f и бита C

Синтаксис: [*label*] ADDWFC f[,d[,a]]

Операнды: $0 \le f \le 255$

$d \in [0,1]$

$a \in [0,1]$

Операция: (W) + (f) + (C) → dest

Измен. флаги: N, OV, C, DC, Z

Код:

0010	00da	ffff	ffff

Описание: Сложение содержимого регистров W и ‘f’ и бита C. Если d=0, результат сохраняется в регистре W, если d=1, то в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа. Если a = 1, используется BSR.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись результата

Пример: ADDWFC REG, 0, 1

До выполнения команды
C = 1
REG = 0x4D
W = 0x02

После выполнения команды
C = 0
REG = 0x02
W = 0x50

ANDLW Логическое И константы и WREG

Синтаксис: [*label*] ANDLW k

Операнды: $0 \le k \le 255$

Операция: (W) . AND . k → W

Измен. флаги: N, Z

Код:

0000	1011	kkkk	kkkk

Описание: Операция И с содержимым регистра W и 8-разрядной константой ‘k’. Результат операции сохраняется в регистре W.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘k’	Выполнение	Запись в регистр W

Пример: ANDLW 0x5F

До выполнения команды
W = 0xA3

После выполнения команды
W = 0x03

ANDWF Логическое И WREG и f

Синтаксис: [*label*] ANDWF f[,d[,a]]

Операнды: $0 \le f \le 255$

$d \in [0,1]$

$a \in [0,1]$

Операция: (W) . AND . (f) → dest

Измен. флаги: N, Z

Код:

0001	01da	ffff	ffff

Описание: Логическая операция поразрядного И регистров W и ‘f’. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа. Если a = 1, используется BSR.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись результата

Пример: ANDWF REG, 0, 0

До выполнения команды
W = 0x17
REG = 0xC2

После выполнения команды
W = 0x02
REG = 0xC2

BC Переход, если перенос (C = 1)

Синтаксис: [*label*] BC n

Операнды: $-128 \le n \le 127$

Операция: Если C = 1
(PC) + 2 + 2n → PC

Измен. флаги: Нет

Код:

1110	0010	nnnn	nnnn

Описание: Если C=1, то происходит переход по адресу PC+2+2n (это действие выполняется за два такта). Если условие ложно, то выполняется следующая команда.

Слов: 1

Циклов: 1(2)

Выполнение команды по тактам

Если переход

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Запись в PC
Нет операции	Нет операции	Нет операции	Нет операции

Если нет перехода

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Нет операции

Пример: HERE BC 5

До выполнения команды
PC = адрес (HERE)

После выполнения команды
Если C = 1 PC = адрес (HERE + 12)
Если C = 0 PC = адрес (HERE + 2)

BCF Сброс бита в f

Синтаксис: [*label*] BCF f,b[,a]

Операнды: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Операция: $0 \rightarrow f<b>$

Измен. флаги: Нет

Код:	1001	bbba	ffff	ffff

Описание: Сброс бита ‘b’ в регистре ‘f’. Если a = 0, выбран банк быстрого доступа. Если a = 1, используется BSR.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись в регистр ‘f’

Пример: BCF FLAG_REG, 7, 0

До выполнения команды
FLAG_REG = 0xC7

После выполнения команды
FLAG_REG = 0x47

BN Переход, если нег. резулт. (N = 1)

Синтаксис: [*label*] BN n

Операнды: $-128 \le n \le 127$

Операция: Если N = 1
$(PC) + 2 + 2n \rightarrow PC$

Измен. флаги: Нет

Код:	1110	0110	nnnn	nnnn

Описание: Если N=1, то происходит переход по адресу PC+2+2n (это действие выполняется за два такта). Если условие ложно, то выполняется следующая команда.

Слов: 1

Циклов: 1(2)

Выполнение команды по тактам

Если переход

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Запись в PC
Нет операции	Нет операции	Нет операции	Нет операции

Если нет перехода

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Нет операции

Пример: HERE BN Jump

До выполнения команды
PC = адрес (HERE)

После выполнения команды
Если N = 1 PC = адрес (Jump)
Если N = 0 PC = адрес (HERE + 2)

BNC Переход, если нет переноса (C = 0)

Синтаксис: [*label*] BNC n

Операнды: $-128 \le n \le 127$

Операция: Если C = 0
(PC) + 2 + 2n → PC

Измен. флаги: Нет

Код:

1110	0011	nnnn	nnnn

Описание: Если C=0, то происходит переход по адресу PC+2+2n (это действие выполняется за два такта). Если условие ложно, то выполняется следующая команда.

Слов: 1

Циклов: 1(2)

Выполнение команды по тактам

Если переход

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Запись в PC
Нет операции	Нет операции	Нет операции	Нет операции

Если нет перехода

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Нет операции

Пример: HERE BNC Jump

До выполнения команды
PC = адрес (HERE)

После выполнения команды
Если C = 0 PC = адрес (Jump)
Если C = 1 PC = адрес (HERE + 2)

BNN Переход, если пол. результ. (N = 0)

Синтаксис: [*label*] BNN n

Операнды: $-128 \le n \le 127$

Операция: Если N = 0
(PC) + 2 + 2n → PC

Измен. флаги: Нет

Код:

1110	0111	nnnn	nnnn

Описание: Если N=0, то происходит переход по адресу PC+2+2n (это действие выполняется за два такта). Если условие ложно, то выполняется следующая команда.

Слов: 1

Циклов: 1(2)

Выполнение команды по тактам

Если переход

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Запись в PC
Нет операции	Нет операции	Нет операции	Нет операции

Если нет перехода

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Нет операции

Пример: HERE BNN Jump

До выполнения команды
PC = адрес (HERE)

После выполнения команды
Если N = 0 PC = адрес (Jump)
Если N = 1 PC = адрес (HERE + 2)

BNOV Переход, если нет переполн. (OV = 0)

Синтаксис: [*label*] BNOV n

Операнды: $-128 \le n \le 127$

Операция: Если OV = 0
(PC) + 2 + 2n → PC

Измен. флаги: Нет

Код:

1110	0101	nnnn	nnnn

Описание: Если OV=0, то происходит переход по адресу PC+2+2n (это действие выполняется за два такта). Если условие ложно, то выполняется следующая команда.

Слов: 1

Циклов: 1(2)

Выполнение команды по тактам

Если переход

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Запись в PC
Нет операции	Нет операции	Нет операции	Нет операции

Если нет перехода

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Нет операции

Пример: HERE BNOV Jump

До выполнения команды
PC = адрес (HERE)

После выполнения команды
Если OV=0 PC = адрес (Jump)
Если OV=1 PC = адрес (HERE + 2)

BNZ Переход, если не нуль (Z = 0)

Синтаксис: [*label*] BNZ n

Операнды: $-128 \le n \le 127$

Операция: Если Z = 0
(PC) + 2 + 2n → PC

Измен. флаги: Нет

Код:

1110	0001	nnnn	nnnn

Описание: Если Z=0, то происходит переход по адресу PC+2+2n (это действие выполняется за два такта). Если условие ложно, то выполняется следующая команда.

Слов: 1

Циклов: 1(2)

Выполнение команды по тактам

Если переход

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Запись в PC
Нет операции	Нет операции	Нет операции	Нет операции

Если нет перехода

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Нет операции

Пример: HERE BNZ Jump

До выполнения команды
PC = адрес (HERE)

После выполнения команды
Если Z = 0 PC = адрес (Jump)
Если Z = 1 PC = адрес (HERE + 2)

BRA Безусловный переход

Синтаксис: [*label*] BRA n

Операнды: $-1024 \le n \le 1023$

Операция: $(PC) + 2 + 2n \rightarrow PC$

Измен. флаги: Нет

Код:

1101	0nnn	nnnn	nnnn

Описание: Командой выполняется безусловный переход. Значение PC будет равно PC + 2 + 2n. Команда исполняется за 2 цикла.

Слов: 1

Циклов: 2

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Запись в PC
Нет операции	Нет операции	Нет операции	Нет операции

Пример:

```
HERE    BRA     Jump
```

До выполнения команды

PC = адрес (HERE)

После выполнения команды

PC = адрес (Jump)

BSF Установка бита в f

Синтаксис: [*label*] BSF f,b[,a]

Операнды: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Операция: $1 \rightarrow f<b>$

Измен. флаги: Нет

Код:

1000	bbba	ffff	ffff

Описание: Установка бита ‘b’ в регистре ‘f’. Если a = 0, выбран банк быстрого доступа. Если a = 1, используется BSR.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись в регистр ‘f’

Пример:

```
BSF     FLAG_REG, 7, 1
```

До выполнения команды

FLAG_REG = 0x0A

После выполнения команды

FLAG_REG = 0x8A

BTFSC Тест бита, пропустить если ‘0’

Синтаксис: [*label*] BTFSC f,b[,a]

Операнды: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Операция: Пропустить, если (f<b>) = 0

Измен. флаги: Нет

Код:

1011	bbba	ffff	ffff

Описание: Если бит ‘b’ в регистре ‘f’ = 0, вместо следующей команды выполняется пустая операция (NOP), растягивая выполнение команды на 2 цикла. Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1(2) 3 цикла, если пропуск двухсловной команды

Выполнение команды по тактам

	Q1	Q2	Q3	Q4
	Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись в регистр ‘f’
Если пропуск	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
Если пропуск 2-х словной команды	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
	Нет операции	Нет операции	Нет операции	Нет операции

Пример:

```
HERE    BTFSC FLAG, 1, 0
FALSE   :
TRUE    :
```

До выполнения команды
PC = адрес (HERE)
После выполнения команды
FLAG<1>=0 PC = адрес (TRUE)
FLAG<1>=1 PC = адрес (FALSE)

BTFSS Тест бита, пропустить если ‘1’

Синтаксис: [*label*] BTFSS f,b[,a]

Операнды: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Операция: Пропустить, если (f<b>) = 1

Измен. флаги: Нет

Код:

1010	bbba	ffff	ffff

Описание: Если бит ‘b’ в регистре ‘f’ = 1, вместо следующей команды выполняется пустая операция (NOP), растягивая выполнение команды на 2 цикла. Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1(2) 3 цикла, если пропуск двухсловной команды

Выполнение команды по тактам

	Q1	Q2	Q3	Q4
	Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись в регистр ‘f’
Если пропуск	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
Если пропуск 2-х словной команды	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
	Нет операции	Нет операции	Нет операции	Нет операции

Пример:

```
HERE    BTFSS FLAG, 1, 0
FALSE   :
TRUE    :
```

До выполнения команды
PC = адрес (HERE)
После выполнения команды
FLAG<1>=0 PC = адрес (FALSE)
FLAG<1>=1 PC = адрес (TRUE)

BTG Инверсия бита в f

Синтаксис: [*label*] BTG f,b[,a]

Операнды: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Операция: -(f<b>) → f<b>

Измен. флаги: Нет

Код:

0111	bbba	ffff	ffff

Описание: Инверсия бита ‘b’ в регистре ‘f’. Если a = 0, выбран банк быстрого доступа. Если a = 1, используется BSR.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись в регистр ‘f’

Пример: BTG PORTC, 4, 0

До выполнения команды
PORTC = 0x75

После выполнения команды
PORTC = 0x65

BOV Переход, если переполнение (OV = 1)

Синтаксис: [*label*] BOV n

Операнды: $-128 \le n \le 127$

Операция: Если OV = 1
(PC) + 2 + 2n → PC

Измен. флаги: Нет

Код:

1110	0100	nnnn	nnnn

Описание: Если OV=1, то происходит переход по адресу PC+2+2n (это действие выполняется за два такта). Если условие ложно, то выполняется следующая команда.

Слов: 1

Циклов: 1(2)

Выполнение команды по тактам

Если переход

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Запись в PC
Нет операции	Нет операции	Нет операции	Нет операции

Если нет перехода

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Нет операции

Пример: HERE BOV Jump

До выполнения команды
PC = адрес (HERE)

После выполнения команды
Если OV=1 PC = адрес (Jump)
Если OV=0 PC = адрес (HERE + 2)

BZ Переход, если нуль (Z = 1)

Синтаксис: [*label*] BZ n

Операнды: $-128 \le n \le 127$

Операция: Если Z = 1
(PC) + 2 + 2n → PC

Измен. флаги: Нет

Код:

1110	0000	nnnn	nnnn

Описание: Если Z=1, то происходит переход по адресу PC+2+2n (это действие выполняется за два такта). Если условие ложно, то выполняется следующая команда.

Слов: 1

Циклов: 1(2)

Выполнение команды по тактам

Если переход

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Запись в PC
Нет операции	Нет операции	Нет операции	Нет операции

Если нет перехода

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’	Выполнение	Нет операции

Пример:

```
HERE    BZ      Jump
```

До выполнения команды
PC = адрес (HERE)

После выполнения команды
Если Z = 1 PC = адрес (Jump)
Если Z = 0 PC = адрес (HERE + 2)

CALL Переход на подпрограмму

Синтаксис: [*label*] CALL k[,s]

Операнды: $0 \le k \le 1048575$
$s \in [0,1]$

Операция: (PC) + 4 → TOS,
k → PC <20:1>,
Если s = 1,
(W) → WS,
(STATUS) → STATUSS
(BSR) → BSRS

Измен. флаги: Нет

Код:1-е слово k<7:0>	1110	110s	kkkk	kkkk
2-е слово k<19:8>	1111	kkkk	kkkk	kkkk

Описание: Вызов подпрограммы во всем диапазоне адресуемой памяти (2 мегабайта). В начале, адрес возврата из подпрограммы (PC+4) сохраняется в стеке. Если s=1, то содержимое регистров W, STATUS, BSR сохраняются в спец. регистрах WS, STATUSS, BSRS. При s=0 обновление регистров не производится (по умолчанию). 20-разрядная константа загружается в PC<20:1>. Команда выполняется за 2 цикла.

Слов: 2

Циклов: 2

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы k<7:0>	Запись PC в стек	Чтение k<19:8>, Запись в PC
Нет операции	Нет операции	Нет операции	Нет операции

Пример:

```
HERE    CALL    THERE, 1
```

До выполнения команды

PC = адрес (HERE)

После выполнения команды

PC = адрес (THERE)
TOS = адрес (HERE)
WS = W
BSRS = BSR
STATUSS = STATUS

CLRF Очистка f

Синтаксис: [*label*] CLRF f[,a]

Операнды: $0 \le f \le 255$
$a \in [0,1]$

Операция: 000h → f
1 → Z

Измен. флаги: Z

Код:

0110	101a	ffff	ffff

Описание: Очистка содержимого регистра ‘f’. Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись в регистр ‘f’

Пример:

```
CLRF    FLAG_REG, 1
До выполнения команды
        FLAG_REG = 0x5A
После выполнения команды
        FLAG_REG = 0x00
```

CLRWDT Сбросить сторожевой таймер

Синтаксис: [*label*] CLRWDT

Операнды: Нет

Операция: 000h → WDT
000h → предделитель WDT
1 → -TO
1 → -PD

Измен. флаги: -TO, -PD

Код:

0000	0000	0000	0100

Описание: Сброс сторожевого таймера WDT и предделителя WDT. – TO = 1, -PD = 1.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Нет операции	Выполнение	Нет операции

Пример:

```
        CLRWDT
До выполнения команды
        Счетчик WDT = ?
После выполнения команды
        Счетчик WDT = 0x00
        Предделитель WDT = 0
        -TO = 1
        -PD = 1
```

COMF Инверсия f

Синтаксис: [*label*] COMF f[,d[,a]]

Операнды: $0 \le f \le 255$

$d \in [0,1]$

$a \in [0,1]$

Операция: (-f) → dest

Измен. флаги: N, Z

Код:

0001	11da	ffff	ffff

Описание: Инвертирование битов регистра ‘f’. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись результата

Пример:

```
COMF    REG, 0, 0
```

До выполнения команды

REG = 0x13

После выполнения команды

W = 0x13
REG = 0xEC

CPFSEQ Сравнить WREG и f, проп. если f = W

Синтаксис: [*label*] CPFSEQ f[,a]

Операнды: $0 \le f \le 255$
$a \in [0,1]$

Операция: (f) – (W)
Пропустить, если (f) = (W)
(незнаковое сравнение)

Измен. флаги: Нет

Код:

0110	001a	ffff	ffff

Описание: Сравнивается значение регистра ‘f’ с содержимым регистра W. Если ‘f’=W, вместо следующей команды выполняется пустая операция (NOP), растягивая выполнение команды на 2 цикла. Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1(2) 3 цикла, если пропуск двухсловной команды

Выполнение команды по тактам

	Q1	Q2	Q3	Q4
	Декодирование команды	Чтение регистра ‘f’	Выполнение	Нет операции
Если пропуск	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
Если пропуск 2-х словной команды	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
	Нет операции	Нет операции	Нет операции	Нет операции

Пример:

```
HERE      CPFSEQ REG, 0
NEQUAL    :
EQUAL     :
```

До выполнения команды
PC = адрес (HERE)
После выполнения команды
REG = W PC = адрес (EQUAL)
REG ≠ W PC = адрес (NEQUAL)

CPFSGT Сравнить WREG и f, проп. если f > W

Синтаксис: [*label*] CPFSGT f[,a]

Операнды: $0 \le f \le 255$
$a \in [0,1]$

Операция: (f) – (W)
Пропустить, если (f) > (W)
(незнаковое сравнение)

Измен. флаги: Нет

Код:

0110	010a	ffff	ffff

Описание: Сравнивается значение регистра ‘f’ с содержимым регистра W. Если ‘f’ > W, вместо следующей команды выполняется пустая операция (NOP), растягивая выполнение команды на 2 цикла. Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1(2) 3 цикла, если пропуск двухсловной команды

Выполнение команды по тактам

	Q1	Q2	Q3	Q4
	Декодирование команды	Чтение регистра ‘f’	Выполнение	Нет операции
Если пропуск	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
Если пропуск 2-х словной команды	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
	Нет операции	Нет операции	Нет операции	Нет операции

Пример:

```
HERE      CPFSGT REG, 0
NGREATER  :
GREATER   :
```

До выполнения команды
PC = адрес (HERE)

После выполнения команды
REG > W PC = адрес (GREATER)
REG ≤ W PC = адрес (NGREATER)

CPFSLT Сравнить WREG и f, проп. если f < W

Синтаксис: [*label*] CPFSLT f[,a]

Операнды: $0 \le f \le 255$
$a \in [0,1]$

Операция: (f) – (W)
Пропустить, если (f) < (W)
(незнаковое сравнение)

Измен. флаги: Нет

Код:

0110	000a	ffff	ffff

Описание: Сравнивается значение регистра ‘f’ с содержимым регистра W. Если ‘f’ < W, вместо следующей команды выполняется пустая операция (NOP), растягивая выполнение команды на 2 цикла. Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1(2) 3 цикла, если пропуск двухсловной команды

Выполнение команды по тактам

	Q1	Q2	Q3	Q4
	Декодирование команды	Чтение регистра ‘f’	Выполнение	Нет операции
Если пропуск	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
Если пропуск 2-х словной команды	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
	Нет операции	Нет операции	Нет операции	Нет операции

Пример:

```
HERE    CPFSLT REG, 1
NLESS   :
LESS    :
```

До выполнения команды
PC = адрес (HERE)
После выполнения команды
REG < W PC = адрес (LESS)
REG ≥ W PC = адрес (NLESS)

DAW Десятичная коррекция WREG

Синтаксис: [*label*] DAW

Операнды: Нет

Операция: Если [W<3:0> > 9] или [DC = 1],
то (W<3:0>) + 6 → (W<3:0>);
иначе (W<3:0>) → (W<3:0>);

Если [W<7:4> > 9] или [C = 1],
то (W<7:4>) + 6 → (W<7:4>);
иначе (W<7:4>) → (W<7:4>);

Измен. флаги: C

Код:

0000	0000	0000	0111

Описание: Десятичная коррекция 8 бит регистра W, результата сложения двух переменных (в формате BCD) в корректный BCD формат.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра W	Выполнение	Запись в регистр W

Пример 1:

```
DAW
```

До выполнения команды

```
W = 0xA5
C = 0
DC = 0
```

После выполнения команды

```
W = 05
C = 1
DC = 0
```

Пример 2:

До выполнения команды

```
W = 0xCE
C = 0
DC = 0
```

После выполнения команды

```
W = 0x34
C = 1
DC = 0
```

DECF Декремент f

Синтаксис: [*label*] DECF f[,d[,a]]

Операнды: $0 \le f \le 255$

$d \in [0,1]$

$a \in [0,1]$

Операция: (f) – 1 → dest

Измен. флаги: C, DC, N, OV, Z

Код:

0000	01da	ffff	ffff

Описание: Декремент значения регистра ‘f’. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись результата

Пример:

```
DECF    CNT, 1, 0
```

До выполнения команды

CNT = 0x01
Z = 0

После выполнения команды

CNT = 0x00
Z = 1

DECFSZ Декремент f, пропустить если 0

Синтаксис: [*label*] DECFSZ f[,d[,a]]

Операнды: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Операция: (f) – 1 → dest
Пропустить, если результат 0

Измен. флаги: Нет

Код:

0010	11da	ffff	ffff

Описание: Декремент значения регистра ‘f’. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если ‘f’ = 0, вместо следующей команды выполняется пустая операция (NOP), растягивая выполнение команды на 2 цикла. Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1(2) 3 цикла, если пропуск двухсловной команды

Выполнение команды по тактам

	Q1	Q2	Q3	Q4
	Декодирование команды	Чтение регистра ‘f’	Выполнение	Нет операции
Если пропуск	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
Если пропуск 2-х словной команды	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
	Нет операции	Нет операции	Нет операции	Нет операции

Пример:

```
HERE    DECFSZ CNT, 1, 1
        GOTO LOOP
CONTINUE
```

До выполнения команды
PC = адрес (HERE)
После выполнения команды
CNT = CNT - 1
CNT = 0 PC = адрес (CONTINUE)
CNT ≠ 0 PC = адрес (HERE + 2)

DECFSNZ Декремент f, пропустить если не 0

Синтаксис: [*label*] DECFSNZ f[,d[,a]]

Операнды: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Операция: (f) – 1 → dest
Пропустить, если результат не 0

Измен. флаги: Нет

Код:

0100	11da	ffff	ffff

Описание: Декремент значения регистра ‘f’. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если ‘f’ ≠ 0, вместо следующей команды выполняется пустая операция (NOP), растягивая выполнение команды на 2 цикла. Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1(2) 3 цикла, если пропуск двухсловной команды

Выполнение команды по тактам

	Q1	Q2	Q3	Q4
	Декодирование команды	Чтение регистра ‘f’	Выполнение	Нет операции
Если пропуск	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
Если пропуск 2-х словной команды	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
	Нет операции	Нет операции	Нет операции	Нет операции

Пример:

```
HERE    DECFSNZ TEMP, 1, 0
ZERO    :
NZERO   :
```

До выполнения команды
PC = адрес (HERE)
После выполнения команды
TEMP = TEMP - 1
TEMP = 0 PC = адрес (ZERO)
TEMP ≠ 0 PC = адрес (NZERO)

GOTO Переход по адресу

Синтаксис: [*label*] GOTO k

Операнды: $0 \le k \le 1048575$

Операция: $k \rightarrow PC <20:1>$

Измен. флаги: Нет

Код:1-е слово k<7:0>	1110	1111	kkkk	kkkk
2-е слово k<19:8>	1111	kkkk	kkkk	kkkk

Описание: Безусловный переход по любому адресу в пределах 2-х мегабайтного адресного пространства. 20-разрядное значение ‘k’ загружается в PC<20:1>. Команда выполняется за два цикла.

Слов: 2

Циклов: 2

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы k<7:0>	Нет операции	Чтение k<19:8>, Запись в PC
Нет операции	Нет операции	Нет операции	Нет операции

Пример:

```
GOTO    THERE
```

После выполнения команды

PC = адрес (THERE)

INCF Инкремент f

Синтаксис: [*label*] INCF f[,d[,a]]

Операнды: $0 \le f \le 255$

$d \in [0,1]$

$a \in [0,1]$

Операция: $(f) + 1 \rightarrow dest$

Измен. флаги: C, DC, N, OV, Z

Код:	0010	10da	ffff	ffff

Описание: Инкремент значения регистра ‘f’. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись результата

Пример:

```
INCF    CNT, 1, 0
```

До выполнения команды

CNT = 0xFF
Z = 0
C = ?
DC = ?

После выполнения команды

CNT = 0x00
Z = 1
C = 1
DC = 1

INCFSZ Инкремент f, пропустить если 0

Синтаксис: [*label*] INCFSZ f[,d[,a]]

Операнды: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Операция: (f) + 1 → dest
Пропустить, если результат 0

Измен. флаги: Нет

Код:

0011	11da	ffff	ffff

Описание: Инкремент значения регистра ‘f’. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если ‘f’ = 0, вместо следующей команды выполняется пустая операция (NOP), растягивая выполнение команды на 2 цикла. Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1(2) 3 цикла, если пропуск двухсловной команды

Выполнение команды по тактам

	Q1	Q2	Q3	Q4
	Декодирование команды	Чтение регистра ‘f’	Выполнение	Нет операции
Если пропуск	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
Если пропуск 2-х словной команды	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
	Нет операции	Нет операции	Нет операции	Нет операции

Пример:

```
HERE    INCFSZ CNT, 1, 0
NZERO   :
ZERO    :
```

До выполнения команды
PC = адрес (HERE)
После выполнения команды
CNT = CNT + 1
CNT = 0 PC = адрес (ZERO)
CNT ≠ 0 PC = адрес (NZERO)

INCFSNZ Инкремент f, пропустить если не 0

Синтаксис: [*label*] INCFSNZ f[,d[,a]]

Операнды: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Операция: (f) + 1 → dest
Пропустить, если результат не 0

Измен. флаги: Нет

Код:

0100	10da	ffff	ffff

Описание: Инкремент значения регистра ‘f’. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если ‘f’ ≠ 0, вместо следующей команды выполняется пустая операция (NOP), растягивая выполнение команды на 2 цикла. Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1(2) 3 цикла, если пропуск двухсловной команды

Выполнение команды по тактам

	Q1	Q2	Q3	Q4
	Декодирование команды	Чтение регистра ‘f’	Выполнение	Нет операции
Если пропуск	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
Если пропуск 2-х словной команды	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
	Нет операции	Нет операции	Нет операции	Нет операции

Пример:

```
HERE    INCFSZ REG, 1, 0
ZERO    :
NZERO   :
```

До выполнения команды
PC = адрес (HERE)
После выполнения команды
REG = REG + 1
REG = 0 PC = адрес (ZERO)
REG ≠ 0 PC = адрес (NZERO)

IORLW Логическое ИЛИ константы и WREG

Синтаксис: [*label*] IORLW k

Операнды: $0 \le k \le 255$

Операция: (W) . OR . k → W

Измен. флаги: N, Z

Код:	0000	1011	kkkk	kkkk

Описание: Операция ИЛИ с содержимым регистра W и 8-разрядной константой ‘k’. Результат операции сохраняется в регистре W.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘k’	Выполнение	Запись в регистр W

Пример:

```
IORLW   0x35
До выполнения команды
        W = 0x9A
После выполнения команды
        W = 0xBF
```

IORWF Логическое ИЛИ WREG и f

Синтаксис: [*label*] IORWF f[,d[,a]]

Операнды: $0 \le f \le 255$

$d \in [0,1]$

$a \in [0,1]$

Операция: (W) . OR . (f) → dest

Измен. флаги: N, Z

Код:	0001	00da	ffff	ffff

Описание: Логическая операция поразрядного ИЛИ регистров W и ‘f’. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа. Если a = 1, используется BSR.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись результата

Пример:

```
IORWF   RESULT, 0, 1
До выполнения команды
        W = 0x91
        RESULT = 0x13
После выполнения команды
        W = 0x93
        RESULT = 0x13
```

LFSR Поместить константу (12 бит) в FSR

Синтаксис: [*label*] FSR f, k

Операнды: $0 \le f \le 2$
$0 \le k \le 4095$

Операция: $k \rightarrow FSRf$

Измен. флаги: Нет

Код: слово 1 k<11:8>	1110	1110	00ff	kkkk
2-е слово k<7:0>	1111	0000	kkkk	kkkk

Описание: 12-разрядная константа ‘k’ загружается в регистр FSR (указатель косвенной адресации)

Слов: 2

Циклов: 2

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы k<11:8>	Нет операции	Запись k<11:8> в FSRfH <11:8>
Нет операции	Чтение константы k<7:0>	Нет операции	Запись k<7:0> в FSRfL <7:0>

Пример:

```
LFSR 2, 0x3AB
```

После выполнения команды

```
LFSRH = 0x03
LFSRL = 0xAB
```

MOVF Переместить f

Синтаксис: [*label*] MOVF f[,d[,a]]

Операнды: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Операция: $(f) \rightarrow dest$

Измен. флаги: N, Z

Код:	0101	00da	ffff	ffff

Описание: Содержимое регистра ‘f’ пересылается в зависимости от состояния бита d. Если d=0, то значение сохраняется в регистре W, если d=1, то значение сохраняется в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа. Если a = 1, используется BSR.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись результата

Пример:

```
MOVF REG, 0, 0
```

До выполнения команды

```
W = 0xFF
REG = 0x22
```

После выполнения команды

```
W = 0x22
REG = 0x22
```

MOVFF Переместить из f в f

Синтаксис: [*label*] MOVFF fs, fd

Операнды: $0 \le fs \le 4095$
$0 \le fs \le 4095$

Операция: (fs) → fd

Измен. флаги: Нет

Код: источник fs	1100	ffff	ffff	ffff
приемник fd	1111	ffff	ffff	ffff

Описание: Содержимое регистра fs пересылается в регистр fd. Регистры fs и fd могут находиться в любом месте адресного пространства размером в 4096 байт (000h-FFFh). В качестве fs и fd может использоваться W. Команда MOVFF может применяться для пересылки данных в периферийные устройства, такие, как буфер передатчика, порт ввода/вывода и др. В качестве fs в команде MOVFF нельзя использовать регистры: PCL, TOSU, TOSH и TOSL.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘fs’	Выполнение	Нет операции
Нет операции	Нет операции	Нет операции	Запись в регистр ‘fd’

Пример:

```
MOVFF   REG1, REG2
До выполнения команды
        REG1= 0x33
        REG2 = 0x11
После выполнения команды
        REG1= 0x33
        REG2 = 0x33
```

MOVLB Поместить константу в BSR<3:0>

Синтаксис: [*label*] MOVLB k

Операнды: $0 \le k \le 255$

Операция: k → BSR

Измен. флаги: Нет

Код:	0000	0001	kkkk	kkkk

Описание: 8-разрядная константа ‘k’ загружается в регистр BSR (регистр выбора банка памяти).

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘k’	Выполнение	Запись в регистр BSR

Пример:

```
MOVLB   5
До выполнения команды
        BSR = 0x02
После выполнения команды
        BSR = 0x05
```

MOVLW Поместить константу в WREG

Синтаксис: [*label*] MOVLW k

Операнды: $0 \le k \le 255$

Операция: $k \rightarrow W$

Измен. флаги: Нет

Код:

0000	1110	kkkk	kkkk

Описание: 8-разрядная константа ‘k’ загружается в регистр W.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘k’	Выполнение	Запись в регистр W

Пример: MOVLW 0x5A

После выполнения команды

W = 0x5A

MOVWF Переместить WREG в f

Синтаксис: [*label*] MOVWF f[,a]

Операнды: $0 \le f \le 255$

$a \in [0,1]$

Операция: $(W) \rightarrow f$

Измен. флаги: Нет

Код:

0110	111a	ffff	ffff

Описание: Пересылка содержимого регистра W в регистр ‘f’. Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра W	Выполнение	Запись в регистр ‘f’

Пример: MOVWF REG, 0

До выполнения команды

W = 0x4F

REG = 0xFF

После выполнения команды

W = 0x4F

REG = 0x4F

MULLW Умножение константы на WREG

Синтаксис: [*label*] MULLW k

Операнды: $0 \le k \le 255$

Операция: (W) x k → PRODH:PRODL

Измен. флаги: Нет

Код:

0000	1101	kkkk	kkkk

Описание: Умножение содержимого регистра W и 8-разрядной константы. 16-разрядный результат помещается в регистровую пару PRODH:PRODL (регистр PRODH содержит старший байт). Выполнение команды не изменяет содержимого регистра W и не влияет на флаги АЛУ. Нулевой результат возможен, но он не детектируется.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘k’	Выполнение	Запись в PRODH: PRODL

Пример:

```
MULLW   0xC4
До выполнения команды
        W = 0xE2
        PRODH = ?
        PRODL = ?
После выполнения команды
        W = 0xE2
        PRODH = 0xAD
        PRODL = 0x08
```

MULWF Умножение WREG и f

Синтаксис: [*label*] MULWF f[,a]

Операнды: $0 \le f \le 255$

$a \in [0,1]$

Операция: (W) x (f) → PRODH:PRODL

Измен. флаги: Нет

Код:

0000	001a	ffff	ffff

Описание: Умножение содержимого регистров W и ‘f’. 16-разрядный результат помещается в регистровую пару PRODH:PRODL (регистр PRODH содержит старший байт). Выполнение команды не изменяет содержимого регистров W, ‘f’ и не влияет на флаги АЛУ. Нулевой результат возможен, но он не детектируется. Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись в PRODH: PRODL

Пример:

```
MULWF   REG, 1
```

До выполнения команды

```
W = 0xC4
REG = 0xB5
PRODH = ?
PRODL = ?
```

После выполнения команды

```
W = 0xC4
REG = 0xB5
PRODH = 0x8A
PRODL = 0x94
```

NEGF Негативное значение f

Синтаксис: [*label*] NEGF f[,a]

Операнды: $0 \le f \le 255$

$a \in [0,1]$

Операция: $(-f) + 1 \rightarrow f$

Измен. флаги: N, OV, C, DC, Z

Код:

0110	110a	ffff	ffff

Описание: Негативное значение ‘f’ в формате дополнения до 2. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись в регистр ‘f’

Пример: NEGF REG, 1

До выполнения команды

REG = 0x3A

После выполнения команды

REG = 0xC6

NOP Нет операции

Синтаксис: [*label*] NOP

Операнды: Нет

Операция: Нет операции

Измен. флаги: Нет

Код:

0000	0000	0000	0000
1111	xxxx	xxxx	xxxx

Описание: Нет операции.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Нет операции	Нет операции	Нет операции

Пример: Нет

POP Чтение вершины стека возврата TOS

Синтаксис: [*label*] POP

Операнды: Нет

Операция: (TOS) →

Измен. флаги: Нет

Код:

0000	0000	0000	0110

Описание: Перемещение всего содержимого стека на один уровень вверх. Предыдущее значение, находящееся на вершине стека, утрачивается. Данная команда может использоваться для программного управления стеком.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Нет операции	POP TOS	Нет операции

Пример:

```
POP
GOTO    NEW
```

До выполнения команды

TOS = 0031A2h
Стек (на 1 уровень ниже) = 014332h

После выполнения команды

TOS = 014332h
PC = NEW

PUSH Запись в вершину стека возврата TOS

Синтаксис: [*label*] PUSH

Операнды: Нет

Операция: (PC+2) → TOS

Измен. флаги: Нет

Код:

0000	0000	0000	0101

Описание: Данная команда помещает в вершину стека (TOS) значение PC+2. Предыдущее значение, находящееся на вершине стека перемещается на один уровень вниз.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	PUSH PC+2	Нет операции	Нет операции

Пример:

```
PUSH
```

До выполнения команды

TOS = 00345Ah
PC = 000124h

После выполнения команды

PC = 000126h
TOS = 000126h
Стек (на 1 уровень ниже) = 00345Ah

RCALL Короткий переход на подпрограмму

Синтаксис: [*label*] RCALL n

Операнды: $-1024 \le n \le 1023$

Операция: (PC) + 2 → TOS,
(PC) + 2 + 2n → PC

Измен. флаги: Нет

Код:

1101	1nnn	nnnn	nnnn

Описание: Производится вызов подпрограммы, находящийся в пределах 1кб от текущей позиции. В стек помещается адрес возврата (PC+2). После этого в программный счетчик загружается новый адрес PC+2+2n. Команда выполняется за два цикла.

Слов: 1

Циклов: 2

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘n’ PUSH PC	Выполнение	Запись в регистр PC
Нет операции	Нет операции	Нет операции	Нет операции

Пример:

```
HERE    RCALL   Jump
```

До выполнения команды
PC = адрес (HERE)
После выполнения команды
PC = адрес (Jump)
TOS = адрес (HERE-2)

RESET Программный сброс

Синтаксис: [*label*] RESET

Операнды: Нет

Операция: Сброс всех регистров и флагов аналогично MCLR

Измен. флаги: Все

Код:

0000	0000	1111	1111

Описание: Команда делает возможным сброс микроконтроллера, аналогичный аппаратному сбросу MCLR.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Старт программного сброса	Нет операции	Нет операции

Пример:

```
RESET
```

После выполнения команды
Регистры = значение после сброса
ОЗУ = значение после сброса

RETFIE Возврат из пп с разреш. прерываний

Синтаксис: [*label*] RETFIE [s]

Операнды: s ∈ [0, 1]

Операция: (TOS) → PC,
1 → GIE/GIEH или PEIE/GIEL,
Если s = 1,
(WS) → W,
(STATUSS) → STATUS,
(BSRS) → BSR,
PCLATU, PCLATH не изменяются

Измен. флаги: GIE/GIEH, PEIE/GIEL

Код:

0000	0000	0001	000s

Описание: Возврат из прерывания. Значение, находящееся на вершине стека (TOS), загружается в PC. Прерывания включаются установкой бита глобального разрешения прерываний высокого/низкого приоритета. Если s=1, содержимое скрытых регистров WS, STATUSS и BSRS загружаются в соответствующие регистры W, STATUS и BSR. Если s=0, то загрузки не производится (по умолчанию).

Слов: 1

Циклов: 2

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Нет операции	Нет операции	POP PC GIEH =1 или GIEL = 1
Нет операции	Нет операции	Нет операции	Нет операции

Пример:

RETFIE 1

После выполнения команды

PC = TOS
W = WS
BSR = BSRS
STATUS = STATUSS
GIE/GIEH, PEIE/GIEL = 1

RETLW Возврат из пп с загрузкой WREG

Синтаксис: [*label*] RETLW k

Операнды: $0 \le k \le 255$

Операция: $k \rightarrow W$,
(TOS) $\rightarrow$ PC,
PCLATU, PCLATH are unchanged

Измен. флаги: Нет

Код:

0000	1100	kkkk	kkkk

Описание: В регистр W записывается 8-разрядная константа ‘k’. Программный счетчик загружается значением, взятым с вершины стека (адресом возврата). Защелка старшего байта адреса (PCLATH) при этом не меняет своего содержимого.

Слов: 1

Циклов: 2

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘k’	Выполнение	POP PC, запись в W
Нет операции	Нет операции	Нет операции	Нет операции

Пример:

```
CALL    TABLE   ; W = смещение таблицы
                ; после выполнения
                ; W = значение из таблицы
:
TABLE
        ADDWF   PCL
        RETLW k0
        RETLW k1
        :
        :
        RETLW kn
```

До выполнения команды

W = 0x07

После выполнения команды

W = значение kn

RETURN Возврат из подпрограммы

Синтаксис: [*label*] RETURN [s]

Операнды: s ∈ [0, 1]

Операция: (TOS) → PC,
Если s = 1,
(WS) → W,
(STATUSS) → STATUS,
(BSRS) → BSR,
PCLATU, PCLATH не изменяются

Измен. флаги: Нет

Код:

0000	0000	0001	001s

Описание: Возврат из подпрограммы. Значение, находящееся на вершине стека (TOS), загружается в PC. Если s=1, содержимое скрытых регистров WS, STATUSS и BSRS загружаются в соответствующие регистры W, STATUS и BSR. Если s=0, то загрузки не производится (по умолчанию).

Слов: 1

Циклов: 2

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Нет операции	Выполнение	POP PC
Нет операции	Нет операции	Нет операции	Нет операции

Пример:

```
RETURN  1
```

После выполнения команды

PC = TOS

RLCF Сдвиг влево через перенос

Синтаксис: [*label*] RLCF f [,d [,a]

Операнды: $0 \le f \le 255$

$d \in [0, 1]$

$a \in [0, 1]$

Операция: (f<n>) → dest<n+1>,

(f<7>) → C,

(C) → dest <0>

Измен. флаги: C, N, Z

Код:

0011	01da	ffff	ffff

Описание: Содержимое регистра ‘f’ циклически сдвигается на один бит влево через флаг переноса. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись значения

Пример:

```
RLCF    REG, 0, 0
До выполнения команды
        REG = 1110 0110
        C = 0
После выполнения команды
        REG = 1110 0110
        W = 1100 1100
        C = 1
```

RLNCF Сдвиг влево без переноса

Синтаксис: [*label*] RLNCF f [,d [,a]

Операнды: $0 \le f \le 255$

$d \in [0, 1]$

$a \in [0, 1]$

Операция: (f<n>) → dest<n+1>,

(f<7>) → dest <0>

Измен. флаги: N, Z

Код:

0100	01da	ffff	ffff

Описание: Содержимое регистра ‘f’ циклически сдвигается на один бит влево. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись значения

Пример: RLNCF REG, 1, 0

До выполнения команды

REG = 1010 1011

После выполнения команды

REG = 0101 0111

RRCF Сдвиг вправо через перенос

Синтаксис: [*label*] RRCF f [,d [,a]

Операнды: $0 \le f \le 255$

$d \in [0, 1]$

$a \in [0, 1]$

Операция: (f<n>) → dest<n-1>,

(f<0>) → C,

(C) → dest <7>

Измен. флаги: C, N, Z

Код:

0011	00da	ffff	ffff

Описание: Содержимое регистра ‘f’ циклически сдвигается на один бит вправо через флаг переноса. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись значения

Пример:

```
RRCF    REG, 0, 0
До выполнения команды
        REG = 1110 0110
        C = 0
После выполнения команды
        REG = 1110 0110
        W = 0111 0011
        C = 0
```

RRNCF Сдвиг вправо без переноса

Синтаксис: [*label*] RRNCF f [,d [,a]

Операнды: $0 \le f \le 255$

$d \in [0, 1]$

$a \in [0, 1]$

Операция: (f<n>) → dest<n-1>,

(f<0>) → dest <7>

Измен. флаги: N, Z

Код:

0100	00da	ffff	ffff

Описание: Содержимое регистра ‘f’ циклически сдвигается на один бит вправо. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись значения

Пример 1: RRNCF REG, 1, 0

До выполнения команды

REG = 1101 0111

После выполнения команды

REG = 1110 1101

Пример 2: RRNCF REG, 0, 0

До выполнения команды

W = ?

REG = 1101 0111

После выполнения команды

W = 1110 1011

REG = 1101 0111

SETF Установить все биты f

Синтаксис: [*label*] SETF f [,a]

Операнды: $0 \le f \le 255$

$a \in [0, 1]$

Операция: FFh → f

Измен. флаги: Нет

Код:

0110	100a	ffff	ffff

Описание: Записать в регистр значение 0xFF. Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись в регистр ‘f’

Пример:

```
SETF    REG,    1,
```

До выполнения команды

REG = 0x5A

После выполнения команды

REG = 0xFF

SLEEP Переход в SLEEP режим

Синтаксис: [*label*] SEEP

Операнды: Нет

Операция: 00h → WDT,

0 → WDT,

1 → - TO,

0 → - PD

Измен. флаги: - TO, - PD

Код:

0000	0000	0000	0011

Описание: Биты -PD=0, -TO=1. Сброс сторожевого таймера WDT. Переход в режим SLEEP с остановкой тактового генератора.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Нет операции	Выполнение	

Пример:

```
SLEEP
```

До выполнения команды

- TO = ?

- PD = ?

После выполнения команды

- TO = 1 *

- PD = 0

SUBFWB Вычитание f из WREG с заемом

Синтаксис: [*label*] SUBFWB f [,d [,a]

Операнды: $0 \le f \le 255$

$d \in [0, 1]$

$a \in [0, 1]$

Операция: (W) – (f) – (-C) → dest

Измен. флаги: N, OV, C, DC, Z

Код:

0101	01da	ffff	ffff

Описание: Из регистра W вычитается значение регистра ‘f’ вместе с флагом переноса. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись результата

Пример 1: SUBFWB REG, 1, 0

До выполнения команды

REG = 3

W = 2

C = 1

После выполнения команды

REG = FF

W = 2

C = 0

Z = 0

N = 1

Пример 2: SUBFWB REG, 0, 0

До выполнения команды

REG = 2

W = 5

C = 1

После выполнения команды

REG = 2

W = 3

C = 1

Z = 0

N = 0

Пример 3: SUBFWB REG, 1, 0

До выполнения команды

REG = 1

W = 2

C = 0

После выполнения команды

REG = 0

W = 2

C = 1

Z = 1

N = 0

SUBLW Вычитание WREG из константы

Синтаксис: [*label*] SUBLW k

Операнды: $0 \le k \le 255$

Операция: $k - (W) \rightarrow W$

Измен. флаги: N, OV, C, DC, Z

Код:

0000	1000	kkkk	kkkk

Описание: Содержимое W вычитается из 8-разрядной константы ‘k’. Результат помещается в регистр W.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘k’	Выполнение	Запись в W

Пример 1:
SUBLW 0x02
До выполнения команды
W = 1
C = ?
После выполнения команды
W = 1
C = 1
Z = 0
N = 0

Пример 2:
SUBLW 0x02
До выполнения команды
W = 2
C = ?
После выполнения команды
W = 0
C = 1
Z = 1
N = 0

Пример 3:
SUBLW 0x02
До выполнения команды
W = 3
C = ?
После выполнения команды
W = FF
C = 0
Z = 0
N = 1

SUBWF Вычитание WREG из f

Синтаксис: [*label*] SUBWF f [,d [,a]

Операнды: $0 \le f \le 255$

$d \in [0, 1]$

$a \in [0, 1]$

Операция: (f) - (W) → dest,

Измен. флаги: N, OV, C, DC, Z

Код:

0101	11da	ffff	ffff

Описание: Содержимое регистра W вычитается из регистра ‘f’. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись результата

Пример 1: SUBWF REG, 1, 0

До выполнения команды

REG = 3

W = 2

C = ?

После выполнения команды

REG = 1

W = 2

C = 1

Z = 0

N = 0

Пример 2: SUBWF REG, 0, 0

До выполнения команды

REG = 2

W = 2

C = ?

После выполнения команды

REG = 2

W = 0

C = 1

Z = 1

N = 0

Пример 3: SUBWF REG, 1, 0

До выполнения команды

REG = 1

W = 2

C = ?

После выполнения команды

REG = FFh

W = 2

C = 0

Z = 0

N = 1

SUBWFB Вычитание WREG из f с заемом

Синтаксис: [*label*] SUBWFB f [,d [,a]

Операнды: $0 \le f \le 255$

$d \in [0, 1]$

$a \in [0, 1]$

Операция: (f) - (W) – (-C) → dest,

Измен. флаги: N, OV, C, DC, Z

Код:

0101	10da	ffff	ffff

Описание: Из регистра ‘f’ вычитается значение регистра W вместе с флагом переноса. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись результата

Пример 1: SUBWFB REG, 1, 0

До выполнения команды

REG = 0x19 (0001 1001)

W = 0x0D (0000 1101)

C = 1

После выполнения команды

REG = 0x0C (0000 1011)

W = 0x0D (0000 1101)

C = 1

Z = 0

N = 0

Пример 2: SUBWFB REG, 0, 0

До выполнения команды

REG = 0x1B (0001 1011)

W = 0x1A (0001 1010)

C = 0

После выполнения команды

REG = 0x1B (0001 1011)

W = 0x00

C = 1

Z = 1

N = 0

Пример 3: SUBWFB REG, 1, 0

До выполнения команды

REG = 0x03 (0000 0011)

W = 0x0E (0000 1101)

C = 1

После выполнения команды

REG = 0xF5

W = 0x0E

C = 0

Z = 0

N = 1

SWAPF Поменять местами полубайты в f

Синтаксис: [*label*] SWAPF f [,d [,a]

Операнды: $0 \le f \le 255$

$d \in [0, 1]$

$a \in [0, 1]$

Операция: (f<3:0>) → dest<7:4>,

(f<7:4>) → dest <3:0>

Измен. флаги: Нет

Код:

0011	10da	ffff	ffff

Описание: Старший и младший полубайт значения регистра ‘f’ меняются местами. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись результата

Пример: SWAPF REG, 1, 0

До выполнения команды

REG = 0x53

После выполнения команды

REG = 0x35

TBLRD Табличное чтение

Синтаксис: [*label*] TBLRD (*; *+; *-; +*)

Операнды: Нет

Операция: Если TBLRD*,
(Память программ (TBLPTR)) → TABLAT;
TBLPTR – не изменяется;
Если TBLRD*+,
(Память программ (TBLPTR)) → TABLAT;
TBLPTR +1 → TBLPTR;
Если TBLRD*-,
(Память программ (TBLPTR)) → TABLAT;
TBLPTR –1 → TBLPTR;
Если TBLRD+*,
(TBLPTR) +1 → TBLPTR;
(Память программ (TBLPTR)) → TABLAT;

Измен. флаги: Нет

Код:

0000	0000	0000	10nn nn=0 * =1 *+ =2 *- =3 +*

Описание: Инструкция используется для чтения содержимого памяти программ. Для доступа к памяти используется указатель TBLPTR. Данный (21-разрядный) указатель позволяет адресовать любой байт адресного пространства размером 2Мбайта.
TBLPTR[0]=0: Младший байт слова, находящегося в памяти программы.
TBLPTR[0]=1: Старший байт слова, находящегося в памяти программы.
Команда TBLRD может изменять значение указателя TBLPTR следующим образом:
- не изменять
- увеличение на 1 после выполнения команды
- уменьшение на 1 после выполнения команды
- увеличение на 1 перед выполнением команды

Слов: 1

Циклов: 2

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Нет операции	Нет операции	Нет операции
Нет операции	Нет операции (Чтение программы памяти)	Нет операции	Нет операции (запись TABLAT)

Пример 1: TBLRD *+ ;

До выполнения команды
TABLAT = 0x55
TBLPTR = 0x00A356
MEMORY(0x00A356) = 0x34

После выполнения команды
TABLAT = 0x34
TBLPTR = 0x00A357

Пример 2: TBLRD +* ;

До выполнения команды
TABLAT = 0xAA
TBLPTR = 0x01A357
MEMORY(0x00A357) = 0x12
MEMORY(0x00A358) = 0x34

После выполнения команды
TABLAT = 0x34
TBLPTR = 0x01A358

TBLWT Табличная запись

Синтаксис: [*label*] TBLWT (*; *+; *-; +*)

Операнды: Нет

Операция: Если TBLWT *,
(TABLAT)→ Память программ (TBLPTR) или рег. защелки;
Если TBLWT*+,
(TABLAT)→ Память программ (TBLPTR) или рег. защелки;
TBLPTR +1 → TBLPTR;
Если TBLWT*-,
(TABLAT)→ Память программ (TBLPTR) или рег. защелки;
TBLPTR –1 → TBLPTR;
Если TBLWT+*,
(TBLPTR) +1 → TBLPTR;
(TABLAT)→ Память программ (TBLPTR) или рег. защелки;

Измен. флаги: Нет

Код:

0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*

Описание: Инструкция используется для записи в память программ. Для доступа к памяти используется указатель TBLPTR. Данный (21-разрядный) указатель позволяет адресовать любой байт адресного пространства размером 2Мбайта.
TBLPTR[0]=0: Младший байт слова, находящегося в памяти программы.
TBLPTR[0]=1: Старший байт слова, находящегося в памяти программы.
Команда TBLRD может изменять значение указателя TBLPTR следующим образом:
- не изменять
- увеличение на 1 после выполнения команды
- уменьшение на 1 после выполнения команды
- увеличение на 1 перед выполнением команды

Слов: 1

Циклов: 2 (больше, если длинная запись в память)

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Нет операции	Нет операции	Нет операции
Нет операции	Нет операции (Чтение TABLAT)	Нет операции	Нет операции (зап. в рег.защ. или память)

Пример 1: TBLWT *+ ;

До выполнения команды
TABLAT = 0x55
TBLPTR = 0x00A356
MEMORY или рег.защ.(0x00A356) = 0xFF

После выполнения команды
TABLAT = 0x55
TBLPTR = 0x00A357
MEMORY или рег.защ.(0x00A356) = 0x55

Пример 2: TBLWT +* ;

До выполнения команды
TABLAT = 0x34
TBLPTR = 0x01389A
MEMORY или рег.защ.(0x01389A) = 0xFF
MEMORY или рег.защ.(0x01389B) = 0xFF

После выполнения команды
TABLAT = 0x34
TBLPTR = 0x01389B
MEMORY или рег.защ.(0x01389A) = 0xFF
MEMORY или рег.защ.(0x01389B) = 0x34

TSTFSZ Тест f, пропустить если 0

Синтаксис: [*label*] TSTFSZ f [,a]

Операнды: $0 \le f \le 255$
$a \in [0,1]$

Операция: Пропустить, если f = 0

Измен. флаги: Нет

Код:

0110	011a	ffff	ffff

Описание: Если ‘f’=0, вместо следующей команды выполняется пустая операция (NOP), растягивая выполнение команды на 2 цикла. Если a = 0, выбран банк быстрого доступа (значение BSR игнорируется). Если a = 1, используется регистр BSR для выбора банка памяти данных.

Слов: 1

Циклов: 1(2) 3 цикла, если пропуск двухсловной команды

Выполнение команды по тактам

	Q1	Q2	Q3	Q4
	Декодирование команды	Чтение регистра ‘f’	Выполнение	Нет операции
Если пропуск	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
Если пропуск 2-х словной команды	Q1	Q2	Q3	Q4
	Нет операции	Нет операции	Нет операции	Нет операции
	Нет операции	Нет операции	Нет операции	Нет операции

Пример:

```
HERE    TSTFSZ CNT, 1
NZERO   :
ZERO    :
```

До выполнения команды

PC = адрес (HERE)

После выполнения команды

Если CNT = 0x00, PC = адрес (NZERO)

Если CNT ≠ 0x00, PC = адрес (ZERO)

XORLW Лог. исключ. ИЛИ константы и WREG

Синтаксис: [*label*] XORLW k

Операнды: $0 \le k \le 255$

Операция: (W) . XOR . k → W

Измен. флаги: N, Z

Код:

0000	1010	kkkk	kkkk

Описание: Операция исключающего ИЛИ с содержимым регистра W и 8-разрядной константой ‘k’. Результат операции сохраняется в регистре W.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение константы ‘k’	Выполнение	Запись в регистр W

Пример:

```
XORLW   0xAF
```

До выполнения команды

W = 0xB5

После выполнения команды

W = 0x1A

XORWF Логическое исключающее ИЛИ WREG и f

Синтаксис: [*label*] XORWF f [,d [,a]

Операнды: $0 \le f \le 255$

$d \in [0,1]$

$a \in [0,1]$

Операция: (W) . XOR . (f) → dest

Измен. флаги: N, Z

Код:

0001	10da	ffff	ffff

Описание: Логическая операция поразрядного исключающего ИЛИ регистров W и ‘f’. Если d=0, то результат сохраняется в регистре W, если d=1, то результат сохраняется в регистре ‘f’ (по умолчанию). Если a = 0, выбран банк быстрого доступа. Если a = 1, используется BSR.

Слов: 1

Циклов: 1

Выполнение команды по тактам

Q1	Q2	Q3	Q4
Декодирование команды	Чтение регистра ‘f’	Выполнение	Запись результата

Пример:

```
XORWF   REG, 0, 1
```

До выполнения команды

REG = 0xAF
W = 0xB5

После выполнения команды

REG = 0x1A
W = 0xB5

Уважаемые господа!

ООО «Микро-Чип» поставляет полную номенклатуру комплектующих фирмы **Microchip Technology Inc** и осуществляет качественную техническую поддержку на русском языке.

С техническими вопросами Вы можете обращаться по адресу support@microchip.ru

По вопросам поставок комплектующих Вы можете обращаться к нам по телефонам:
(095) 963-9601
(095) 737-7545
и адресу sales@microchip.ru

На сайте
www.microchip.ru
Вы можете узнать последние новости нашей фирмы, найти техническую документацию и информацию по наличию комплектующих на складе.