PIC12F629／675
アセンブリ言語

リファレンスの見方

PIC12F629／675の命令。

・[ ] は省略可能であることを表します。
・a,b,c,,, はリストであることを表します。
・f はF レジスタ(ファイルレジスタ)のアドレスが示す値を表します。
・b は整数値で指定したビットの位置(8ビットの数値)を表します。
・k はリテラル(整数数値)を表します。
・d は結果を保存する場所を表します。
・label　はラベルであることを表します。複数のラベルを使うときには label、label1、label2 などと表現します。
・expr は式であることを表します。式は一つの値でもかまいません。

構成

・プログラムメモリー: 0005～1FFFh(予約済み:0400～1FFFh は使用できず)
・データメモリー(バンク0): 00～7Fh
　INDF:00h、 TMR0:01h、 PCL:02h、 STATUS:03h、 FSR:04h、 GPIO:05h、 PCLATH:0Ah、 他制御:0B～1Fh、 GPR:20～5Fh。
・データメモリー(バンク1): 80～FFh
　INDF:80h、 PCL:82h、 STATUS:83h、 FSR:84h、 他制御:85～9Fh、 アクセス(20～5Fh):A0～DFh。
※STATUSにバンク切り替えのRP0ビットが含まれている。

・スタック: レベル1～8
・入出力:
　GP0:入出力
　GP1:入出力
　GP2:入出力
　GP3:入力
　GP4:入出力
　GP5:入出力

以下に命令を示す。

ADDLW	Add Literal and W
構文	[*label*] **ADDLW** k
オペランド	0≦k≦255
操作	(W)＋k→(W)
影響ステータス	C、DC、Z
説明	Wレジスタの内容が8ビットリテラル「k」に加算され、結果がWレジスタに配置されます。
ワード／サイクル	1／1

例:
`MOVLV B'0011' ;Wレジスタに値3を保存する`
`ADDLW B'1000' ;Wレジスタの値と8を加算してWレジスタに保存する`
`;Wレジスタの値はB'1011' (11)`	

ADDWF	Add W and f
構文	[*label*] **ADDWF** f, d
オペランド	0≦f≦127 d∈[0,1]
操作	(W)＋(f)→(宛先)
影響ステータス	C、DC、Z
説明	Wレジスタの内容をレジスタ「f」に加算します。「d」が0の場合、結果はWレジスタに保存されます。「d」が1の場合、結果はレジスタ「f」に保存されます。
ワード／サイクル	1／1

例:
`MOVLV B'0011' ;Wレジスタに値3を保存する`
`ADDLW VAR,F ;Wレジスタの値とVARを加算してVARに保存する`	

ANDLW	AND Literal with W
構文	[*label*] **ANDLW** k
オペランド	0≦k≦255
操作	(W) AND (k)→(W)
影響ステータス	Z
説明	Wレジスタの内容と、8ビットリテラル「k」とANDをとります。結果はWレジスタに格納されます。
ワード／サイクル	1／1
例:	

```
MOVLV    B'0011'  ;Wレジスタに値3を保存する。
ANDLW    VAR      ;Wレジスタの値とVARをANDしてWレジスタに保存する。
```

ANDWF	AND W with f
構文	[*label*] **ANDWF** f, d
オペランド	0≦f≦127 d∈[0,1]
操作	(W) AND (f)→(宛先)
影響ステータス	Z
説明	ANDのレジスタ「f」とWレジスタ。「d」が0の場合、結果はWレジスタに保存されます。「d」が1の場合、結果はレジスタ「f」に保存されます。
ワード／サイクル	1／1
例:	

```
MOVLV    B'0011'  ;Wレジスタに値3を保存する。
ANDWF    VAR,W    ;Wレジスタの値とVARをANDしてWレジスタに保存する。
```

BCF	Bit Clear f
構文	[*label*] **BCF** f, b
オペランド	0≦f≦127 0≦b≦7
操作	0→(f<b>)
影響ステータス	なし
説明	レジスタ「f」のビット「b」はクリアされます。
ワード／サイクル	1／1
例: `BCF STATUS,RP0 ;STATUSレジスタのRP0ビットを0にする`	

BSF	Bit Set f
構文	[*label*] **BSF** f, b
オペランド	0≦f≦127 0≦b≦7
操作	1→(f<b>)
影響ステータス	なし
説明	レジスタ「f」のビット「b」はセットされます。
ワード／サイクル	1／1
例: `BSF STATUS,RP0 ;バンク1に切り替える`	

BTFSS	Bit Test f、Skip if Set
構文	[*label*] **BTFSS** f, b
オペランド	0≦f≦127 0≦b≦7
操作	(f<b>)=1ならスキップ
影響ステータス	なし
説明	レジスタ「f」のビット「b」が「0」の場合、次の命令が実行されます。 ビット「b」が「1」の場合、次の命令は破棄され、代わりにNOPが実行され、これが2T_{CY}命令になります。
ワード／サイクル	1／1(スキップした場合は2)

例:
`BTFSS PORTA,0 ;RA0が1なら次の命令をスキップする`
`GOTO SW2 ;SW2を調べる命令にジャンプする`
`INCF COUNT ;カウンタをインクリメントする`	

BTFSC	Bit Test、Skip if Clear
構文	[*label*] **BTFSC** f, b
オペランド	0≦f≦127 0≦b≦7
操作	(f<b>)=0ならスキップ
影響ステータス	なし
説明	レジスタ「f」のビット「b」が「1」の場合、次の命令が実行されます。 レジスタ「f」のビット「b」が「0」の場合、次の命令は破棄され、代わりにNOPが実行され、これが2T_{CY}命令になります。
ワード／サイクル	1／1(スキップした場合は2)

例:
`BTFSC STATUS,Z ;直前の命令の結果はゼロか？`
`GOTO ZERO`
`GOTO NON_ZERO`	

CALL	Call Subroutine
構文	[*label*] **CALL** k
オペランド	0≦k≦2047
操作	(PC)＋1→TOS、 k→PC<10:0>、 (PCLATH<4:3>)→PC<12:11>
影響ステータス	なし
説明	サブルーチンを呼び出します。 最初に、リターン･アドレス(PC＋1)がスタックにプッシュされます。 11ビットの即値アドレスは、 PCビット<10:0>にロードされます。 PCの上位ビットはPCLATHからロードされます。 CALLは2サイクルの命令です。
ワード／サイクル	1／2
例: `CALL MYSUB ;サブルーチンMYSUBを呼び出す`	

CLRF	Clear f
構文	[*label*] **CLRF** f
オペランド	0≦f≦127
操作	00h→(f) 1→Z
影響ステータス	Z
説明	レジスタ「f」の内容がクリアされ、 Zビットが設定されます。
ワード／サイクル	1／1
例: `CLRF TRISB ;Bポートをすべて出力用に設定する`	

CLRW	Clear W
構文	[*label*] **CLRW**
オペランド	なし
操作	00h→(W) 1→Z
影響ステータス	Z
説明	Wレジスタがクリアされます。 ゼロビット(Z)が設定されます。
ワード／サイクル	1／1
例: `CLRW ;Wレジスタをクリアする`	

CLRWDT	Clear Watchdog Timer
構文	[*label*] **CLRWDT**
オペランド	なし
操作	00h→WDT 0→WDTプリスケーラー、 1→$\overline{TO}$ 1→$\overline{PD}$
影響ステータス	$\overline{TO}$、$\overline{PD}$
説明	CLRWDT命令は、 ウォッチドッグ・タイマーをリセットします。 また、 WDTのプリスケーラーもリセットします。 ステータスビット$\overline{TO}$および$\overline{PD}$が設定されます。
ワード／サイクル	1／1
例: `CLRWDT ;ウォッチドッグタイマー(WDT)をクリアする`	

COMF	Complement f
構文	[*label*] **COMF** f, d
オペランド	0≦f≦127 d∈[0,1]
操作	($\bar{f}$)→(宛先)
影響ステータス	Z
説明	レジスタ「f」の内容は補完されます。「d」が0の場合、結果はWに格納されます。「d」が1の場合、結果はレジスタ「f」に格納されます。
ワード／サイクル	1／1
例: `COMF DATA,F ;DATAのビットを反転してDATAに保存する`	

DECF	Decrement f
構文	[*label*] **DECF** f, d
オペランド	0≦f≦127 d∈[0,1]
操作	(f)－1→(宛先)
影響ステータス	Z
説明	レジスタ「f」をデクリメントします。「d」が0の場合、結果はWレジスタに保存されます。「d」が1の場合、結果はレジスタ「f」に保存されます。
ワード／サイクル	1／1
例: `DECF COUNT,F ;COUNTをデクリメントしてCOUNTに保存する`	

DECFSZ	Decrement f, Skip if 0
構文	[*label*] **DECFSZ** f, d
オペランド	0≦f≦127 d∈[0,1]
操作	(f)－1→(宛先); 結果=0の場合スキップ
影響ステータス	なし
説明	レジスタ「f」の内容を減少します。「d」が0の場合、結果はWレジスタに配置されます。「d」が1の場合、結果はレジスタ「f」に戻されます。 結果が ≠ 0以外の場合、次の命令が実行されます。結果が0の場合、代わりにNOPが実行、スキップされ、2T_{CY}命令になります。
ワード／サイクル	1／1(スキップした場合は2)

例:
`DECFSZ COUNT,F ;COUNTをデクリメントしてCOUNTに保存する`
`GOTO NOT_ZERO`
`GOTO COUNT_ZERO`	

GOTO	Unconditional Branch（無条件分岐）
構文	[*label*] GOTO k
オペランド	0≦k≦2047
操作	k→PC<10:0> PCLATH<4:3>→PC<12:11>
影響ステータス	なし
説明	GOTOは無条件分岐です。11ビットの即値がPCビット<10:0>にロードされます。PCの上位ビットは、PCLATH<4:3>からロードされます。GOTOは2サイクルの命令です。
ワード／サイクル	1／2
例: GOTO LOOP ;LOOPというラベルにジャンプする	

INCF	Increment f
構文	[*label*] **INCF** f, d
オペランド	0≦f≦127 d∈[0,1]
操作	(f)＋1→(宛先)
影響ステータス	Z
説明	レジスタ「f」の内容が増分されます。「d」が0の場合、結果はWレジスタに配置されます。「d」が1の場合、結果はレジスタ「f」に戻されます。
ワード／サイクル	1／1
例: `INCF COUNT,F ;COUNTをインクリメントしてCOUNTに保存する`	

INCFSZ	Increment f, Skip if 0
構文	[*label*] **INCFSZ** f, d
オペランド	0≦f≦127 d∈[0,1]
操作	(f)＋1→(宛先)、 結果=0の場合スキップ
影響ステータス	なし
説明	レジスタ「f」の内容が増分されます。「d」が0の場合、結果はWレジスタに配置されます。「d」が1の場合、結果はレジスタ「f」に戻されます。 結果が ≠0以外の場合、次の命令が実行されます。結果が0の場合、代わりにNOPが実行、スキップされ、2TCY命令になります。
ワード／サイクル	1／1(スキップした場合は2)

例:
`INCFSZ VAR,F ;VARをインクリメントする`
`GOTO NOT_ZERO`
`GOTO ZERO`	

IORLW	Inclusive OR Literal with W
構文	[*label*] **IORLW** k
オペランド	0≦k≦255
操作	(W) OR k→(W)
影響ステータス	Z
説明	Wレジスタの内容は、8ビットリテラル「k」とOR演算されます。結果はWレジスタに格納されます。
ワード／サイクル	／1
例: `IORLF  B'0010' ;Wレジスタの値と2をORしてWレジスタに保存する`	

IORWF	Inclusive OR W with f
構文	[*label*] **IORWF** f, d
オペランド	0≦f≦127 d∈[0,1]
操作	(W) OR (f)→(宛先)
影響ステータス	Z
説明	インクルーシブOR: Wレジスタとレジスタ「f」。「d」が0の場合、結果はWレジスタに配置されます。「d」が1の場合、結果はレジスタ「f」に戻されます。
ワード／サイクル	1／1

例:
`IORWF  VAR,W  ;Wレジスタの値とVARの値をORして結果を`
`;Wレジスタに保存する`	

MOVF	Move f
構文	[*label*] **MOVF** f, d
オペランド	0≦f≦127 d∈[0,1]
操作	(f)→(宛先)
影響ステータス	Z
説明	レジスタ「f」の内容は、「d」のステータスに応じて宛先に移動されます。 d=0の場合、 宛先はWレジスタです。 d=1の場合、 宛先はファイル･レジスタ「f」自体です。 ステータス･フラグZが影響を受けるため、 d=1はファイル･レジスタのテストに役立ちます。
ワード／サイクル	1／1
例: `MOVF FSR,0` 命令の後: W = value in FSR レジスター Z = 1	
例: `MOVF LEDS,W ;LEDSの値をWレジスタに保存する`	

MOVLW	Move literal to W
構文	[*label*] **MOVLW** k
オペランド	0≦k≦255
操作	k→(W)
影響ステータス	なし
説明	8ビットのリテラル「k」がWレジスタにロードされます。「do n't care」は「0」としてアセンブルされます。
ワード／サイクル	1／1

例:
`MOVLW B'0001' ;1をWレジスタに保存する`
`MOVWF LEDS ;変数LEDSに値を設定する`	

MOVWF	Move W to f
構文	[*label*] **MOVWF** f
オペランド	0≦f≦127
操作	(W)→(f)
影響ステータス	なし
説明	データをWレジスタからレジスタ「f」に移動します。
ワード／サイクル	1／1

例:

```
MOVWF    OPTION
```

命令の前:
OPTION= 0xFF
W = 0x4F

命令の後:
OPTION= 0x4F
W = 0x4F

例:

```
MOVLW    B'0001'  ;1をWレジスタに保存する
MOVWF    LEDS     ;変数LEDSに値を設定する
```

NOP	No Operation
構文	[*label*] **NOP**
オペランド	なし
操作	動作なし
影響ステータス	なし
説明	動作なし
ワード／サイクル	1／1
例:	NOP

例:

```
LOOP    NOP             :なにもしない
        GOTO    LOOP    ;LOOPにジャンプする
```

RETFIE	Return from Interrupt
構文	[*label*] **RETFIE**
オペランド	なし
操作	TOS→PC、 1→GIE
影響ステータス	なし
説明	割り込みから戻る。 スタックはPOPされ、 Top-of-Stack (TOS)がPCにロードされます。 割り込みは、グローバル割り込みイネーブル・ビットGIE(INTCON<7>)を設定することで有効になります。 これは2サイクルの命令です。
ワード／サイクル	1／2

例:

```
            ;退避したレジスタを復帰する
RETFIE      ;割り込み処理を終えて本来の処理を行う
```

RETLW	Return with literal in W
構文	[*label*] **RETLW** k
オペランド	0≦k≦255
操作	k→(W)、 TOS→PC
影響ステータス	なし
説明	Wレジスタには、 8ビットリテラル「k」がロードされます。 プログラム･カウンターは、 スタックの先頭(戻りアドレス)から読み込まれます。 これは2サイクルの命令です。
ワード／サイクル	1／2

例:

```
        CALL TABLE      ;W contains table
                        ;offset value
        ⋮               ;W now has table value
TABLE
        ADDWF PCL       ;W = offset
        RETLW k1 ;Begin table
        RETLW k2 ;
        ⋮
        RETLW kn ;End of table
```

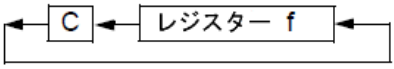
命令の前: 命令の後:

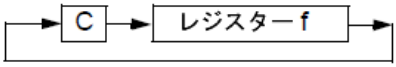
W = 0x07 W = value of k8

例:

```
                ;退避した変数を復帰する
        RETLW 0 ;Wレジスタに0を入れてリターンする
```

RETURN	Return from Subroutine
構文	[*label*] **RETURN**
オペランド	なし
操作	TOS→PC
影響ステータス	なし
説明	サブルーチンから戻ります。スタックがPOPされ、スタックの最上部(TOS)がプログラム･カウンターにロードされます。 これは2サイクルの命令です。
ワード／サイクル	1／2
例: ⋮ RETURN ;サブルーチンから戻る。	

RLF	Rotate Left f through Carry
構文	[*label*] **RLF** f, d
オペランド	0≦f≦127
	d∈[0,1]
操作	以下の説明を参照してください
影響ステータス	C
説明	レジスタ「f」の内容は、 キャリーフラグを介して左に1ビット回転します。「d」が0の場合、 結果はWレジスタに配置されます。「d」が1の場合、 結果はレジスタ「f」に保存されます。 
ワード／サイクル	1／1
例: MOVLW B'0101' ;0101をWレジスタに保存する MOVWF LEDS ;LEDSに0101を保存する RLF LEDS,w ;左にローテートしてWレジスタに保存する	

RRF	Rotate Right f through Carry
構文	[*label*] **RRF** f, d
オペランド	0≦f≦127 d∈[0,1]
操作	以下の説明を参照してください
影響ステータス	C
説明	レジスタ「f」の内容は、キャリー・フラグを介して1ビット右に回転します。「d」が0の場合、結果はWレジスタに配置されます。「d」が1の場合、結果はレジスタ「f」に戻されます。 
ワード／サイクル	1／1

例:

```
MOVLW   B'1010'  ;1010をWレジスタに保存する
MOVWF   LEDS     ;LEDSに1010を保存する
RRF     LEDS,w   ;右にローテートしてWレジスタに保存する
```

SLEEP	スリープ
構文	[*label*] **SLEEP**
オペランド	なし
操作	00h→WDT、 0→WDT プリスケーラー、 1→$\overline{TO}$、 0→$\overline{PD}$
影響ステータス	$\overline{TO}$、$\overline{PD}$
説明	パワーダウン･ステータス･ビット$\overline{PD}$がクリアされます。 タイムアウト･ステータス･ビット、 $\overline{TO}$が設定されます。 ウォッチドッグ･タイマーとそのプリスケーラーはクリアされます。 　発振器が停止した状態で、 プロセッサがスリープ･モードになります。
ワード／サイクル	1／1
例: `SLEEP　　;スリープする`	

SUBLW	Subtract W from Literal
構文	[*label*] **SUBLW** k
オペランド	0≦k≦255
操作	~~k－(W)→(W)~~　(W)－k→(W)
影響ステータス	C、DC、Z
説明	~~Wレジスタは、8ビット･リテラル「k」から減算されます(2の補数法)~~　Wレジスタから、 8ビット･リテラル「k」で減算されます(2の補数法)。 結果はWレジスタに格納されます。
ワード／サイクル	1／1

例:

```
LEDS    EQU     B'1111'
V       EQU     0xA
        ⋮
        MOVLW   V       ;VをWレジスタに保存する
        SUBLW   LEDS    ;~~LEDSからWレジスタの~~値を減算
                        ;Wレジスタの値からLEDSを減算
                        ;してWレジスタに保存する
```

SUBWF	Subtract W from f
構文	[*label*] **SUBWF** f, d
オペランド	0≦f≦127 d∈[0,1]
操作	~~(f) − (W) →(受け)~~ (W)−(f)→(受け)
影響ステータス	C、DC、Z
説明	~~レジスタ「f」からWレジスタを減算します(2の補数法)~~ レジスタ「W」からレジスタ「f」を減算します(2の補数法)。「d」が0の場合、結果はWレジスタに保存されます。「d」が1の場合、結果はレジスタ「f」に保存されます。
ワード／サイクル	1／1

例:
`SUBWF DATA,F ;~~DATAからWレジスタの値を引いてWレジスタに~~`
`;Wレジスタの値からDATAの値を引いてDATAに保存する`	

SWAPF	Swap Nibbles in f
構文	[*label*] **SWAPF** f, d
オペランド	0≦f≦127 d∈[0,1]
操作	(f<3:0>)→(受け<7:4>)、 (f<7:4>)→(受け<3:0>)
影響ステータス	なし
説明	レジスタ「f」の上位ニブルと下位ニブルが交換されます。「d」が0の場合、結果はWレジスタに配置されます。「d」が1の場合、結果はレジスタ「f」に配置されます。
ワード／サイクル	1／1

例:
`SWAPF STATUS,W ;STATUSの上下ビットを入れ替えて`
`;Wレジスタに保存する`	

XORLW	Exclusive OR Literal with W
構文	[*label*] **XORLW** k
オペランド	0→k→255
操作	(W) XOR k→(W)
影響ステータス	Z
説明	Wレジスタの内容は、8ビット･リテラル「k」とXORされます。結果はWレジスタに格納されます。
ワード／サイクル	1／1
例: `XORLW B'01010101'` Wレジスタの値を`B'01010101'`でXORする	

XORWF	Exclusive OR W with f
構文	[*label*] **XORWF** f, d
オペランド	0≦f≦127 d∈[0,1]
操作	(W) XOR (f)→(受け)
影響ステータス	Z
説明	Wレジスタの内容とレジスタ「f」の排他的論理和。「d」が0の場合、結果はWレジスタに保存されます。「d」が1の場合、結果はレジスタ「f」に保存されます。
ワード／サイクル	1／1
例: `XORWF DATA,W ;`Wレジスタの値と`DATA`を`XOR`して`W`レジスタに保存する	