HYCON 紘康科技

CPU CORE 應用說明書

H08A 與 H08B 指令集比較表與補充說明

目錄.

1 H08A與H08B指令集比較表

Instruction				Description	Cycles	Status Affected	Ref Page
H08A		H08B					
BYTE-ORIENTED FILE REGISTER OPERATIONS							
ADDC	f,d,a	ADDC	f,d,a	將 W 與 F 和 C 做相加，並將結果放至 W 或 F。	1	C,DC,N,OV,Z	
ADDF	f,d,a	ADDF	f,d,a	將 W 與 F 做相加，並將結果放至 W 或 F。	1	C,DC,N,OV,Z	
ADDL	k	ADDL	k	將常數 k 與 W 做相加，並將結果放至 W。	1	C,DC,N,OV,Z	
ANDF	f,d,a	ANDF	f,d,a	將 W 與 F 做 AND 運算，並將結果放至 W 或 F。	1	N,Z	
ANDL	k	ANDL	k	將常數 k 與 W 做 AND 運算，並將結果放至 W。	1	N,Z	
ARLC	f,d,a	-	-	將 F 內的值與 C 一起做左移動作，並將結果放至 W 或 F。	1	C,N,OV,Z	
ARRC	f,d,a	-	-	將 F 內的值做右移動作,MSB 保留不變,LSB 移至 C	1	C,N,Z	
CLRF	f,a	CLRF	f,a	將 F 內的值都清為 0。	1	None	
COMF	f,d,a	COMF	f,d,a	將 F 內的值取補數，並將結果放至 W 或 F。	1	N,Z	
CPSE	f,a	CPSE	f,a	若 F 與 W 的值相等，則跳過下一個指令。	1(2)(3)	None	
CPSG	f,a	CPSG	f,a	若 F 大於 W，則跳過下一個指令。	1(2)(3)	None	
CPSL	f,a	CPSL	f,a	若 F 小於 W，則跳過下一個指令。	1(2)(3)	None	
DCF	f,d,a	DCF	f,d,a	將 F 內的值減 1，並將結果放至 W 或 F。	1	C,DC,N,OV,Z	
DCSUZ	f,d,a	DCSUZ	f,d,a	將 F 內的值減 1，若不為 0 則跳過下一個指令，並將結果放至 W 或 F。	1(2)(3)	None	
DCSZ	f,d,a	DCSZ	f,d,a	將 F 內的值減 1，若為 0 則跳過下一個指令，並將結果放至 W 或 F。	1(2)(3)	None	
INF	f,d,a	INF	f,d,a	將 F 內的值加 1，並將結果放至 W 或 F。	1	C,DC,N,OV,Z	
INSUZ	f,d,a	INSUZ	f,d,a	將 F 內的值加 1，若不為 0 則跳過下一個指令，並將結果放至 W 或 F。	1(2)(3)	None	
INSZ	f,d,a	INSZ	f,d,a	將 F 內的值加 1，若為 0 則跳過下一個指令，並將結果放至 W 或 F。	1(2)(3)	None	

IORF	f,d,a	IORF	f,d,a	將 W 與 F 做 OR 運算，並將結果放至 W 或 F。	1	N,Z	
IORL	k	IORL	k	將常數 k 與 W 做 OR 運算，並將結果放至 W。	1	N,Z	
LBSR	k	-	-	將常數 k 搬到 BSR 暫存器去。	1	None	
LDPR	k,f	-	-	將常數 k (9-bit) 搬到第 f 個 FSR 暫存器去 (f = 0 ~ 1) 。	2	None	
MULF	f,a	-	-	將 W 與 F 做相乘。	2	None	
MULL	k	-	-	將常數 k 與 W 做乘法運算。	2	None	
MVF	f,d,a	MVF	f,d,a	將 W 內的值搬到 F 中(d=1)或 F 內的值搬到 W(d=0)。	1	None	
MVFF	fs,fd	-	-	將 Fs 內的資料搬到 Fd 中。	2	None	
MVL	k	MVL	k	將常數 k 搬到 W 去。	1	None	
RETL	k	RETL	k	將堆疊最上層的值取出來放至 PC 中，並將 W 的值設為 k ，而主程式由目前 PC 值開始執行	2	None	
RLF	f,d,a	RLF	f,d,a	將 F 內的值做左移動作，並將結果放至 W 或 F。	1	N,Z	
RLFC	f,d,a	RLFC	f,d,a	將 F 內的值與 C 一起做左移動作，並將結果放至 W 或 F。	1	C,N,Z	
RRF	f,d,a	RRF	f,d,a	將 F 內的值做右移動作，並將結果放至 W 或 F。	1	N,Z	
RRFC	f,d,a	RRFC	f,d,a	將 F 內的值與 C 一起做右移動作，並將結果放至 W 或 F。	1	C,N,Z	
SETF	f,a	SETF	f,a	將 F 內的值設為 0xFF。	1	None	
SUBC	f,d,a	SUBC	f,d,a	將 F 內的值減掉 W 及反向 C，並將結果放至 W 或 F。	1	C,DC,N,OV,Z	
SUBF	f,d,a	SUBF	f,d,a	將 F 內的值減掉 W，並將結果放至 W 或 F。	1	C,DC,N,OV,Z	
SUBL	k	SUBL	k	將常數 k 與 W 做減法，並將結果放至 W。	1	C,DC,N,OV,Z	
SWPF	f,d,a	SWPF	f,d,a	將 F 內的值高 4 位元與低 4 位元對調，並將結果放至 W 或 F。	1	None	
TFSZ	f,a	TFSZ	f,a	測試 F 內的值是否等於 0，若為 0 則跳過下一個指令。	1(2)(3)	None	
XORF	f,d,a	XORF	f,d,a	將 W 與 F 做 XOR 運算，並將結果放至 W 或 F。	1	N,Z	
XORL	k	XORL	k	將常數 k 與 W 做 XOR 運算，並將結果放至 W。	1	N,Z	
CONTROL OPERATIONS							
CALL	n,s	CALL	n	將下一個指令的 PC 值存到堆疊的最上層，並跳到位址 n 。	2	None	

CWDT		CWDT	0	將看門狗計時器清為 0。	1	TO	
IDLE		IDLE	0	進入等待模式	1	IdleB	
JC	n	-	-	若 C = 1 則跳到位址 n。	1(2)	None	
JMP	n	JMP	n	無條件跳到位址 n。	2	None	
JN	n	-	-	若 N = 1 則跳到位址 n。	1(2)	None	
JNC	n	-	-	若 C = 0 則跳到位址 n。	1(2)	None	
JNN	n	-	-	若 N = 0 則跳到位址 n。	1(2)	None	
JNO	n	-	-	若 OV = 0 則跳到位址 n。	1(2)	None	
JNZ	n	-	-	若 Z = 0 則跳到位址 n。	1(2)	None	
JO	n	-	-	若 OV = 1 則跳到位址 n。	1(2)	None	
JZ	n	-	-	若 Z = 1 則跳到位址 n。	1(2)	None	
NOP		NOP	0	空指令。	1	None	
POP		-	-	將堆疊指標暫存器減 1 後，取出該堆疊指標所指向堆疊層中堆疊的值放回 TOS 暫存器中。	1	None	
PUSH		-	-	在堆疊指標加 1 後，將取出的位址存入 TOS 暫存器中。	1	None	
RCALL	n	-	-	將下一個指令的 PC 值存到堆疊的最上層，並跳到位址 n,-1024≦n≦1023	2	None	
RET	s	RET	-	由副程式返回主程式，並將堆疊最上層的值取出來放至 PC 中，而主程式由目前 PC 值開始執行。	2	None	
RETI	s	RETI	-	由中斷副程式返回主程式，並將堆疊最上層的值取出來放至 PC 中，而主程式由目前 PC 值開始執行。	2	GIE	
RJ	n	-	-	無條件跳到位址 n,-1024≦n≦1023	2	None	
SLP	f,a	SLP	f,a	進入睡眠狀態。		PD	
BIT-ORIENTED FILE REGISTER OPERATIONS							
BCF	f,b,a	BCF	f,b,a	將 F 內某個位元 (Bit) 設定為 0。	1	None	
BSF	f,b,a	BSF	f,b,a	將 F 內某個位元 (Bit) 設定為 1。	1	None	

BTGF	f,b,a	BTGF	f,b,a	將 F 內某個位元 (Bit) 做 NOT 運算。	1	None	
BTSS	f,b,a	BTSS	f,b,a	測試 F 內某個位元 (Bit) 的值是否等於 1，若為 1 則跳過下一個指令。	1(2)(3)	None	
BTSZ	f,b,a	BTSZ	f,b,a	測試 F 內某個位元 (Bit) 的值是否等於 0，若為 0 則跳過下一個指令。	1(2)(3)	None	
PROGRAM MEMORY OPERATIONS							
MVLP	k	-	-	將常數 k(0≦k≦16384d)搬到 TABLE 指標器(TBLPTRU/TBLPTRH/TBLPTRL)	2	None	
TBLR	*	-	-	以 TBLPTR 記錄器之內容為位址指標，讀取程式記憶體之內容至 TBLDH/TBLDL 暫存器中。	2	None	
TBLR	*+	-	-	以 TBLPTR 記錄器之內容為位址指標，讀取程式記憶體之內容至 TBLDH/TBLDL 暫存器中。然後將位址指標自動+1。	2	None	

Remark

f	暫存器	b	暫存器第 b 個位元
n	記憶體位置	k	8 位元常數
d	資料存放地方; d = 0 表示存放在 W 累加器；d = 1 表示存放在 f 暫存器。		
a	資料存放在哪個記憶體位置,a=0 存放在目前記憶體位置；a=1 存放在 BSR 暫存器內所指定記憶體位置		
s	備份功能“wreg、status、bsr”暫存器,s = 1 表示 shadow register 備份 wreg、status、bsr 三個暫存器的資料；s = 0 不做備份動作		

注意！Shadow register 只有一組， 其保存資料永遠為最近一次推入的值

2 指令補充說明

2.1 PCLATL 的說明

一般程序的撰寫，按照排序來執行程序，如果想要跳到指定的程序區段執行，則一般使用 RJ；但是如果想要讓程序跳到一個變數的程序區間，可利用 PCLATL 的運算來達到目的。

一般 PCLATL 的運算可透過 MVF，ADDF，ANDF…等指令來完成，但是在下 PCLATL 的運算之前，需要先確定 PCLATH 與 PCLATU 的值。

注意!建議是使用 RJ，如需使用 JMP 須注意是否將組譯精簡功能(smart complier)開啟且跳躍範圍必須在-1024~1023 之間，不然透過 ADDF、SUBF 來運算 PCLATL 的值時，需接短指令不然會有跳錯行的情況

列出一些運算的注意事項：

如果想要透過 ADDF、SUBF 來運算 PCLATL 的值，如果 WREG 的值與 PCLATL 的值相加超過 0x100 則要小心 PCLATH，PCLATU 要事先改變，否則會跳錯程式。

例如：程式需要靠 WREG 的值來判斷跳躍的位址

```
        MVL     High  JPB             ; PCLATH  存放跳躍起始位址(bit 15~8)
        MVF     PCLATH,F,ACCE
        MVF     MainCount+1,W,ACCE      ; 讀取跳躍值
        ADDF    PCLATL,F,ACCE           ; 當前 PCLATL +  跳躍值
JPB:
        RJ      MainDaPro0
        RJ      MainDaPro1
        RJ      MainDaPro2
        RJ      MainDaPro3
        RJ      MainDaPro4
        RJ      MainDaPro5
        RJ      MainDaPro6
        rj      MainDaPro7
```

以上程式乍看之下是沒有問題的，但是如果當**JPB**的位置 **=0x00FF、0x01FF….**時，跳躍的位置就會出錯。

因此可以改成以下程式，保證**JPB**在任何位置下皆不會跳錯

```
        MVL     High   JPB              ; PCLATH 存放跳躍起始位址(bit 15~8)
        MVF     PCLATH,F,ACCE
        MVF     MainCount+1,W,ACCE
        ADDL    Low    JPB               ;測試跳躍值與 JPB 的位置相加是否超過 0x100, (Low    JPB)+( MainCount+1) ->W
        BTSZ    STAUTS,C,ACCE
        INF     PCLATH,F,ACCE            ;如果超過 0x100 則 [PCLATU,PCLATH] + 1
        MVF     PCLATL,F,ACCE            ; 自 W 載入 PCLATL
  JPB:
        RJ      MainDaPro0
        RJ      MainDaPro1
```

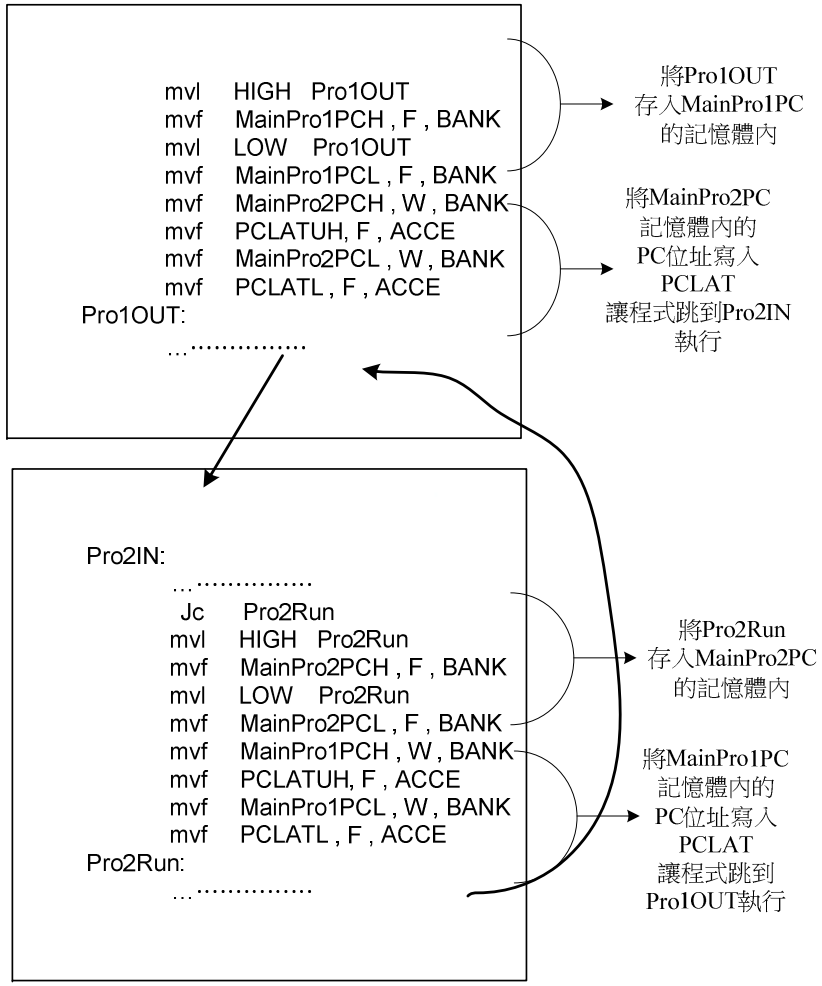
亦可以指定位址的方式，預知**PC**執行跳躍指令時不會更動**PTLATH**而使用更簡短的指令完成跳躍，例如：

```
        MVF     MainCount+1,W,ACCE
        JMP     JPB
        ……
        MVF     Table_Index,W,ACCE
        CALL        Table
  ORG 0700H     ;;  將 TABLE 放置於程式最後 256 行之內（以 2K word Rom size 為例, 最後 256 行為 0700H~07FFH）
  JPB:
        ADDF        PCLATL,F,ACCE
        RJ      MainDaPro0
        RJ      MainDaPro1
  Table:
        ADDF        PCLATL,F,ACCE
```

```
RETL    055H
RETL    0AAH
```

當一段程式在執行中如果想要跳至另外一個程式區段執行，等待另一個程式執行到一個階段，可以跳回原來的程式接著繼續執行。

例如：



註：請勿用 MVFF 來搬移 PCLATL 值

2.2 POP 的說明

POP 是對於堆疊的處理。POP 是將堆疊的值彈出到 TOSU、TOSH 與 TOSL 內，STKPTR 減 1。如果發現堆疊不夠用時，可以將 TOSU、TOSH 與 TOSL 存入記憶體內，再利用 POP 讓堆疊釋放一層，返回時再將先前儲存的堆疊值利用 PCLATL 返回。

這個功能尤其在 CALL 最後一層時，又要保留一層給中斷用，可以利用 POP 與 PCLATL 來完成。以下範例是 MCU 的 堆疊只有 6 層，當程式 CALL 到第 5 層時，要再往下 CALL 一層，又要保留一層給中斷用時的範例說明。

```
MainLoop:
        call    Tst1
        ……
        jmp     MainLoop

Tst1:
        ……
        call    Tst2
        ……
        ret
Tst2:
        ……
        call    Tst3
        ……
        ret
Tst3:
        ……
        call    Tst4
        ……
        ret
Tst4:
        ……
        call    Tst5
        ……
        ret
Tst5:
        ……
        mvff    TOSU,STKBufU
        mvff    TOSH,STKBufH
```

```
        mvff    TOSL,STKBufL
        POP
        call    Tst6
        mvff    STKBufU,PCLATU
        mvff    STKBufH,PCLATH
        mvf     STKBufL,W,BANK
        mvf     PCLATL,F,ACCE       → 返回到 Tst4
Tst6:
        ……
        ret
```

2.3 XOR 的運用

XOR 是異或值的運算，可用來做數值相等的判斷，例如：

```
        mvl     3
        xorf    Temp , W , BANK
        jz      ValEQU
        ……..
        ValEQU:
        ……..
```

XOR 也可做兩個記憶體的數值置換的功能，例如：記憶體 TEMP0 與記憶體 TEMP1 的數值對調

```
        mvf     TEMP0 , W , ACCE
        xorf    TEMP1 , W , ACCE
        xorf    TEMP0 , F , ACCE
        xorf    TEMP1 , F , ACCE
```

2.4 Object File

以二進制型態存放，透過 Global 提供給外部程序引用參數，參數可以是 Label、SRAM 的定義。

例如:

Subroutine 的 source code 為

```
Global      TEMP, MA, INDF0, POINC0, FSR0H, FSR0L
Global      ClearRAM
Global      F, W, ACCE, BANK
TEMP        EQU     080h
MA          EQU     090h
```

```
        INDF0       EQU     000h
        POINC0      EQU     001h
        FSR0H       EQU     00Fh
        FSR0L       EQU     010h
        F           EQU     1
        W           EQU     0
        ACCE        EQU     0
        BANK        EQU     1
ClearRAM:
        MVL         80h
        MVF         FSR0L, F, ACCE
        CLRF        FSR0H, ACCE
  ClearLoop:
        CLRF        POINC0, ACCE
        TFSZ        FSR0L, ACCE
        JMP         ClearLoop
        RET
```

經過 Compiler 後(組譯選項需勾選 obj)，產生 Subroutine.obj，其中釋放幾項參數:

SRAM 的參數 TEMP, MA, INDF0, POINC0, FSR0H, FSR0L

立即數定義參數 F, W, ACCE, BANK

Label(子程序) 參數 ClearRAM

主程序中可以引用 Subroutine.obj 所有的宣告參數

```
    ORG             0000h
    JMP         Begin
Begin:
       CALL         ClearRAM
Mainloop:
    MVL         0A0h
    MVF         TEMP,F,ACCE
    ….
    JMP         Mainloop
    INCLUDE Subroutine.obj
```

Object File 也可引用外部的參數，透過 EXTERN 呼叫

例如外部參數 MXX 定義為 SRAM 位址為 0A0h

```
EXTERN      MXX
CLRF        MXX, ACCE
```

3 修訂記錄

以下描述本文件差異較大的地方，而標點符號與字形的改變不在此描述範圍。

版本	頁次	變更摘要
V01	ALL	初版發行
V02	ALL	文件格式變更
		增加指令補充說明
V03	-	文件格式變更
V04	-	刪除 DAW 指令
V05	P7~9	更新 PCLAT 範例程式與描述
V06	P7~9	更新 PCLAT 範例程式與描述