HYCON 紘康科技

CPU CORE 应用说明书

H08A 与 H08B 指令集比较表与补充说明

©2008-2010 HYCON Technology Corp.

目录

1 H08A与H08B指令集比较表.....錯誤! 尚未定義書籤。

2 指令补充说明 .....錯誤! 尚未定義書籤。

2.1 PCLATL的说明 ..... 錯誤! 尚未定義書籤。

2.2 POP的说明 ..... 錯誤! 尚未定義書籤。

2.3 XOR的运用 ..... 錯誤! 尚未定義書籤。

2.4 Object File ..... 錯誤! 尚未定義書籤。

3 参考文献.....錯誤! 尚未定義書籤。

4 修订记录.....錯誤! 尚未定義書籤。

©2008-2010 HYCON Technology Corp

1 H08A 与 H08B 指令集比较表

Instruction				Description	Cycles	Status Affected		Ref Page
H08A		H08B				H08A	H08B	
BYTE-ORIENTED FILE REGISTER OPERATIONS								
MVL	k	MVL	k	将常数 k 搬到 W 去。	1	None	None	
MVF	f,d,a	MVF	f,d,a	将 W 内的值搬到 F 中(d=1)或 F 内的值搬到 W(d=0)。	1	None	None	
MVFF	fs,fd	-	-	将 Fs 内的资料搬到 Fd 中。	2	None		
LBSR	k	-	-	将常数 k 搬到 BSR 缓存器去。	1	None		
LDPR	k,f	-	-	将常数 k (9-bit) 搬到第 f 个 FSR 缓存器去 (f = 0 ~ 1) 。	2	None		
ADDC	f,d,a	ADDC	f,d,a	将 W 与 F 和 C 做相加，并将结果放至 W 或 F。	1	C,DC,N,OV,Z	C,Z	
ADDF	f,d,a	ADDF	f,d,a	将 W 与 F 做相加，并将结果放至 W 或 F。	1	C,DC,N,OV,Z	C,Z	
ADDL	k	ADDL	k	将常数 k 与 W 做相加，并将结果放至 W。	1	C,DC,N,OV,Z	C,Z	
INF	f,d,a	INF	f,d,a	将 F 内的值加 1，并将结果放至 W 或 F。	1	C,DC,N,OV,Z	C,Z	
SUBC	f,d,a	SUBC	f,d,a	将 F 内的值减掉 W 及反向 C，并将结果放至 W 或 F。	1	C,DC,N,OV,Z	C,Z	
SUBF	f,d,a	SUBF	f,d,a	将 F 内的值减掉 W，并将结果放至 W 或 F。	1	C,DC,N,OV,Z	C,Z	
SUBL	k	SUBL	k	将常数 k 与 W 做减法，并将结果放至 W。	1	C,DC,N,OV,Z	C,Z	
DCF	f,d,a	DCF	f,d,a	将 F 内的值减 1，并将结果放至 W 或 F。	1	C,DC,N,OV,Z	C,Z	
MULL	k	-	-	将常数 k 与 W 做乘法运算。	2	None		
MULF	f,a	-	-	将 W 与 F 做相乘。	2	None		
ANDL	k	ANDL	k	将常数 k 与 W 做 AND 运算，并将结果放至 W。	1	N,Z	Z	
ANDF	f,d,a	ANDF	f,d,a	将 W 与 F 做 AND 运算，并将结果放至 W 或 F。	1	N,Z	Z	
IORL	k	IORL	k	将常数 k 与 W 做 OR 运算，并将结果放至 W。	1	N,Z	Z	
IORF	f,d,a	IORF	f,d,a	将 W 与 F 做 OR 运算，并将结果放至 W 或 F。	1	N,Z	Z	
XORL	k	XORL	k	将常数 k 与 W 做 XOR 运算，并将结果放至 W。	1	N,Z	Z	

©2008-2010 HYCON Technology Corp

XORF	f,d,a	XORF	f,d,a	将 W 与 F 做 XOR 运算，并将结果放至 W 或 F。	1	N,Z	Z	
CLRF	f,a	CLRF	f,a	将 F 内的值都清为 0。	1	None	None	
COMF	f,d,a	COMF	f,d,a	将 F 内的值取补码，并将结果放至 W 或 F。	1	N,Z	Z	
SETF	f,a	SETF	f,a	将 F 内的值设为 0xFF。	1	None	None	
RLF	f,d,a	RLF	f,d,a	将 F 内的值做左移动作，并将结果放至 W 或 F。	1	N,Z	Z	
RLFC	f,d,a	RLFC	f,d,a	将 F 内的值与 C 一起做左移动作，并将结果放至 W 或 F。	1	C,N,Z	C,Z	
RRF	f,d,a	RRF	f,d,a	将 F 内的值做右移动作，并将结果放至 W 或 F。	1	N,Z	Z	
RRFC	f,d,a	RRFC	f,d,a	将 F 内的值与 C 一起做右移动作，并将结果放至 W 或 F。	1	C,N,Z	C,Z	
SWPF	f,d,a	SWPF	f,d,a	将 F 内的值高 4 位与低 4 位对调，并将结果放至 W 或 F。	1	None	None	
ARLC	f,d,a	-	-	将 F 内的值与 C 一起做左移动作，并将结果放至 W 或 F。	1	C,N,OV,Z		
ARRC	f,d,a	-	-	将 F 内的值做右移动作,MSB 保留不变,LSB 移至 C	1	C,N,Z		
INSZ	f,d,a	INSZ	f,d,a	将 F 内的值加 1，若为 0 则跳过下一个指令，并将结果放至 W 或 F。	1(2)(3)	None	None	
INSUZ	f,d,a	INSUZ	f,d,a	将 F 内的值加 1，若不为 0 则跳过下一个指令，并将结果放至 W 或 F。	1(2)(3)	None	None	
DCSZ	f,d,a	DCSZ	f,d,a	将 F 内的值减 1，若为 0 则跳过下一个指令，并将结果放至 W 或 F。	1(2)(3)	None	None	
DCSUZ	f,d,a	DCSUZ	f,d,a	将 F 内的值减 1，若不为 0 则跳过下一个指令，并将结果放至 W 或 F。	1(2)(3)	None	None	
CPSE	f,a	CPSE	f,a	若 F 与 W 的值相等，则跳过下一个指令。	1(2)(3)	None	None	
CPSG	f,a	CPSG	f,a	若 F 大于 W，则跳过下一个指令。	1(2)(3)	None	None	
CPSL	f,a	CPSL	f,a	若 F 小于 W，则跳过下一个指令。	1(2)(3)	None	None	
TFSZ	f,a	TFSZ	f,a	测试 F 内的值是否等于 0，若为 0 则跳过下一个指令。	1(2)(3)	None	None	
CONTROL OPERATIONS								
JMP	n	JMP	n	无条件跳到地址 n。	2	None	None	
RJ	n	-	-	无条件跳到地址 n,-1024≦n≦1023	2	None		
CALL	n,s	CALL	n	将下一个指令的 PC 值存到堆栈的最上层，并跳到地址 n 。	2	None	None	
RCALL	n	-	-	将下一个指令的 PC 值存到堆栈的最上层，并跳到地址 n,-1024≦n≦1023	2	None		

©2008-2010 HYCON Technology Corp

JC	n	-	-	若 C = 1 则跳到地址 n。	1(2)	None		
JNC	n	-	-	若 C = 0 则跳到地址 n。	1(2)	None		
JN	n	-	-	若 N = 1 则跳到地址 n。	1(2)	None		
JNN	n	-	-	若 N = 0 则跳到地址 n。	1(2)	None		
JZ	n	-	-	若 Z = 1 则跳到地址 n。	1(2)	None		
JNZ	n	-	-	若 Z = 0 则跳到地址 n。	1(2)	None		
JO	n	-	-	若 OV = 1 则跳到地址 n。	1(2)	None		
JNO	n	-	-	若 OV = 0 则跳到地址 n。	1(2)	None		
RET	s	RET	-	由子程序返回主程序，并将堆栈最上层的值取出来放至 PC 中，而主程序由目前 PC 值开始执行。	2	None	None	
RETI	s	RETI	-	由中断子程序返回主程序，并将堆栈最上层的值取出来放至 PC 中，而主程序由目前 PC 值开始执行。	2	GIE	GIE	
RETL	k	RETL	k	将堆栈最上层的值取出来放至 PC 中，并将 W 的值设为 k ，而主程序由目前 PC 值开始执行	2	None	None	
POP	-	-	-	将堆栈指针缓存器减 1 后，取出该堆栈指针所指向堆栈层中堆栈的值放回 TOS 缓存器中。	1	None		
CWDT	-	CWDT	-	将看门狗定时器清为 0。	1	TO	TO	
IDLE	-	IDLE	-	进入等待模式	1	IdleB	IdleB	
SLP	f,a	SLP	f,a	进入睡眠状态。		PD	PD	
NOP		NOP		空指令。	1	None	None	
BIT-ORIENTED FILE REGISTER OPERATIONS								
BCF	f,b,a	BCF	f,b,a	将 F 内某个位 (Bit) 设定为 0。	1	None	None	
BSF	f,b,a	BSF	f,b,a	将 F 内某个位 (Bit) 设定为 1。	1	None	None	
BTGF	f,b,a	BTGF	f,b,a	将 F 内某个位 (Bit) 做 NOT 运算。	1	None	None	
BTSS	f,b,a	BTSS	f,b,a	测试 F 内某个位 (Bit) 的值是否等于 1，若为 1 则跳过下一个指令。	1(2)(3)	None	None	

©2008-2010 HYCON Technology Corp

BTSZ	f,b,a	BTSZ	f,b,a	测试 F 内某个位 (Bit) 的值是否等于 0，若为 0 则跳过下一个指令。	1(2)(3)	None	None	
PROGRAM MEMORY OPERATIONS								
MVLP	k	-	-	将常数 k(0≦k≦16384d)搬到 TABLE 指标器 (TBLPTRU/TBLPTRH/TBLPTRL)	2	None		
TBLR	*	-	-	以 TBLPTR 记录器之内容为地址指针，读取程序内存之内容至 TBLDH/TBLDL 缓存器中。	2	None		
TBLR	*+	-	-	以 TBLPTR 记录器之内容为地址指针，读取程序内存之内容至 TBLDH/TBLDL 缓存器中。然后将地址指针自动+1。	2	None		

Remark

f	缓存器	b	缓存器第 b 个位
n	内存位置	k	8 位常数
d	数据存放地方; d = 0 表示存放在 W 累加器；d = 1 表示存放在 f 缓存器。		
a	数据存放在哪个内存位置,a=0 存放在目前内存位置；a=1 存放在 BSR 缓存器内所指定内存位置		
s	备份功能“wreg、status、bsr”缓存器,s = 1 表示 shadow register 备份 wreg、status、bsr 三个缓存器的数据；s = 0 不做备份动作		

注意！Shadow register 只有一组， 其保存资料永远为最近一次推入的值

©2008-2010 HYCON Technology Corp

2 指令补充说明

2.1 PCLATL 的说明

一般程序的撰写，按照排序来执行程序，如果想要跳到指定的程序区段执行，则一般使用 JMP 或 RJ；但是如果想要让程序跳到一个变量的程序区间，可利用 PCLATL 的运算来达到目的。

一般 PCLATL 的运算可透过 MVF，ADDF，ANDF…等指令来完成，但是在下 PCLATL 的运算之前，需要先确定 PCLATH 与 PCLATU 的值。

列出一些运算的注意事项：

如果想要透过 ADDF、SUBF 来运算 PCLATL 的值，如果 WREG 的值与 PCLATL 的值相加超过 0x100 则要小心 PCLATH，PCLATU 要事先改变，否则会跳错程序。

例如：程序需要靠 WREG 的值来判断跳跃的地址

```
        MVL     High  JPB             ; PCLATH 存放跳跃起始地址(bit 15~8)
        MVF     PCLATH,F,ACCE
        MVF     MainCount+1,W,ACCE      ; 读取跳跃值
        ADDF    PCLATL,F,ACCE           ; 当前 PCLATL + 跳跃值
JPB:
        JMP     MainDaPro0
        JMP     MainDaPro1
        JMP     MainDaPro2
        JMP     MainDaPro3
        JMP     MainDaPro4
        JMP     MainDaPro5
        JMP     MainDaPro6
        JMP     MainDaPro7
```

以上程序乍看之下是没有问题的，但是如果当JPB的位置 =0x00FF、0x01FF….时，跳跃的位置就会出错。

因此可以改成以下程序，保证JPB在任何位置下皆不会跳错：

```
        mvl     High  JPB             ; PCLATH 存放跳跃起始地址(bit 15~8)
```

©2008-2010 HYCON Technology Corp

```
        mvf     PCLATH,F,ACCE
        mvf     MainCount+1,W,ACCE
        addl    Low   JPB              ;测试跳跃值与 JPB 的位置相加是否超过 0x100, (Low   JPB)+( MainCount+1) ->W
        btsz    STAUTS,C,ACCE
        inf     PCLATH,F,ACCE          ;如果超过 0x100 则 [PCLATU,PCLATH] + 1
        mvf     PCLATL,F,ACCE          ; 自 W 载入 PCLATL
JPB:
        jmp     MainDaPro0
        jmp     MainDaPro1
```

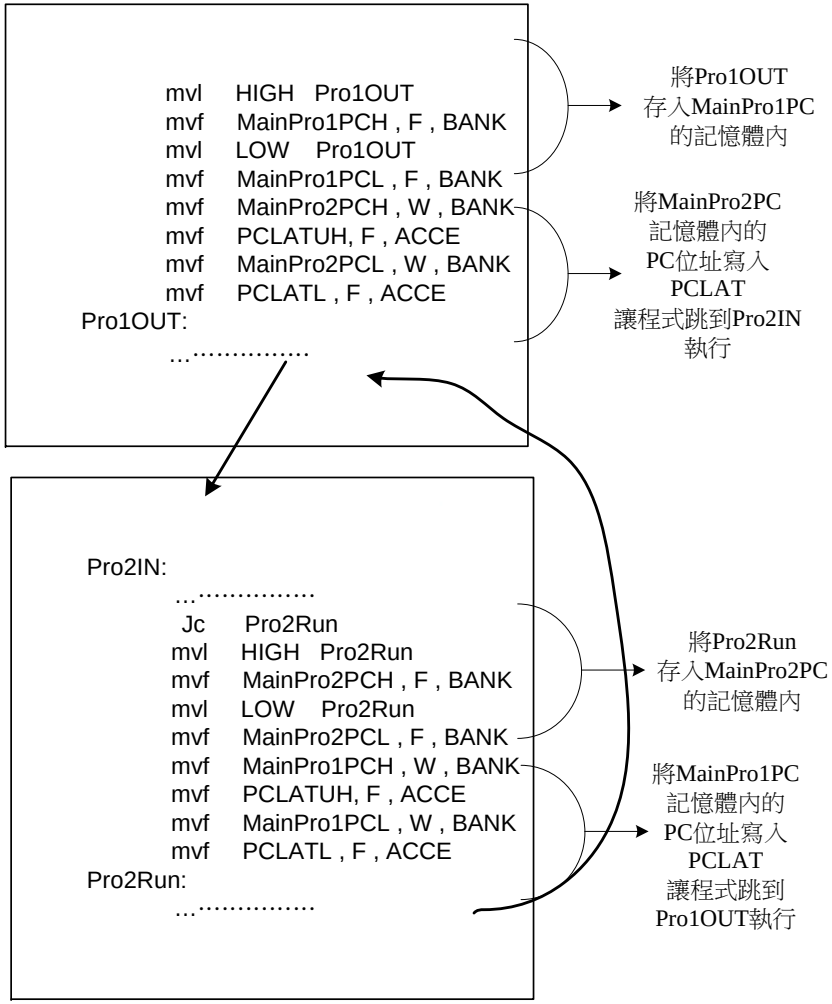
亦可以指定地址的方式，预知PC执行跳跃指令时不会更动PTLATH而使用更简短的指令完成跳跃，例如：

```
        mvf     MainCount+1,W,ACCE
        jmp     JPB
        ……
        mvf     Table_Index,W,ACCE
        call    Table
ORG 0700H   ;;  将 TABLE 放置于程序最后 256 行之内（以 2K word Rom size 为例，最后 256 行为 0700H~07FFH）
JPB:
        addf    PCLATL,F,ACCE
        jmp     MainDaPro0
        jmp     MainDaPro1
Table:
        addf    PCLATL,F,ACCE
        RETL    055H
        RETL    0AAH
```

©2008-2010 HYCON Technology Corp

当一段程序在执行中如果想要跳至另外一个程序区段执行，等待另一个程序执行到一个阶段，可以跳回原来的程序接着继续执行。

例如：



注：请勿用 MVFF 来搬移 PCLATL 值

©2008-2010 HYCON Technology Corp

2.2 POP 的说明

POP 是对于堆栈的处理。POP 是将堆栈的值弹出到 TOSU、TOSH 与 TOSL 内，STKPTR 减 1。如果发现堆栈不够用时，可以将 TOSU、TOSH 与 TOSL 存入内存内，再利用 POP 让堆栈释放一层，返回时再将先前储存的堆栈值利用 PCLATL 返回。

这个功能尤其在 CALL 最后一层时，又要保留一层给中断用，可以利用 POP 与 PCLATL 来完成。以下范例是 MCU 的 堆栈只有 6 层，当程序 CALL 到第 5 层时，要再往下 CALL 一层，又要保留一层给中断用时的范例说明。

```
MainLoop:
        CALL    Tst1
        ……
        JMP     MainLoop

Tst1:
        ……
        CALL    Tst2
        ……
        RET
Tst2:
        ……
        CALL    Tst3
        ……
        RET
Tst3:
        ……
        CALL    Tst4
        ……
        RET
```

©2008-2010 HYCON Technology Corp

```
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 的运用

XOR 是异或值的运算，可用来做数值相等的判断，例如：

```
        MVL     3
        XORF    Temp , W , BANK
        JZ      ValEQU
        ……..
```

©2008-2010 HYCON Technology Corp

```
        ValEQU:
        ……..
```

XOR 也可做两个内存的数值置换的功能，例如：内存 TEMP0 与内存 TEMP1 的数值对调

```
        MVF     TEMP0 , W , ACCE
        XORF    TEMP1 , W , ACCE
        XORF    TEMP0 , F , ACCE
        XORF    TEMP1 , F , ACCE
```

2.4 Object File

以二进制型态存放，透过 Global 提供给外部程序引用参数，参数可以是 Label、SRAM 的定义。

例如：

Subroutine 的 source code 为

```
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:
```

©2008-2010 HYCON Technology Corp

```
MVL     80h
MVF     FSR0L, F, ACCE
CLRF    FSR0H, ACCE
ClearLoop:
CLRF    POINC0, ACCE
TFSZ    FSR0L, ACCE
JMP     ClearLoop
RET
```

经过 Compiler 后(组译选项需勾选 obj)，产生 Subroutine.obj，其中释放几项参数:

SRAM 的参数 TEMP, MA, INDF0, POINC0, FSR0H, FSR0L

立即数定义参数 F, W, ACCE, BANK

Label(子程序) 参数 ClearRAM

主程序中可以引用 Subroutine.obj 所有的宣告参数

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

©2008-2010 HYCON Technology Corp

Object File 也可引用外部的参数，透过 EXTERN 呼叫

例如外部参数 MXX 定义为 SRAM 地址为 0A0h

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

©2008-2010 HYCON Technology Corp

3 参考文献

1、APD-CORE002-Vxx Instructions set H08A

2、APD-CORE003-Vxx Instructions set H08B

3、UG-HY11S14-Vxx HY11Pxx Family User’s Guide

©2008-2010 HYCON Technology Corp

4 修订记录

以下描述本文件差异较大的地方，而标点符号与字形的改变不在此描述范围。

版本	页次	变更摘要
V01	Page3	H08A 与 H08B 指令集比较表新增 H08B（ Instruction,Status Affected ）列。
	Page5	删除 PUSH 指令
	Page8	将 BTSS 改为 BTSZ
V02	Page15	新增参考文献
V03	All	文件编号由：APD-CORE001-V02变为：APD-CORE001-V03_SC
V04	-	删除 DAW 指令
V05	P7～9	更新 PCLAT 范例程序与描述

©2008-2010 HYCON Technology Corp