



十速科技

TenX Technology

TM8713 使用手冊

十速科技股份有限公司

TEL: 886-2-29728029

FAX: 886-2-29728774

網址: www.tenx.com.tw

目 錄

第1章 功能簡介		頁
1-1 前言		2
1-2 規格(Feature)		2
1-3 功能區塊圖(Function Block)		3
1-4 腳位定義(Pin Assignment)		3
1-5 腳位說明(Pin Description)		4
第2章 內部系統結構		
2-1 系統時鐘(System Clock)		5
2-2 程式記憶體(ROM)		5
2-3 資料暫存器(RAM)		6
2-4 堆疊器(Stack)		6
2-5 累加器(Accumulator)		6
2-6 索引暫存器(Index Register)		7
2-7 十進位運算(Decimal Operation)		7
2-8 計時器 1(Timer 1)		8
2-9 狀態暫存器(Status Register)		10
2-10 控制暫存器(Control Register)		12
2-11 蜂鳴器輸出腳(Buzzer Output Pin)		13
2-12 輸入/輸出埠(Input/Output Port)		13
2-13 冷光驅動線路(EL Plant Driver)		16
2-14 外部中斷線路(External Interrupt)		16
2-15 按鍵掃描(Key Board Scanning)		17
2-16 液晶驅動輸出腳(LCD Driver Output Pin)		28
2-17 電源線路的接法(The Connection of Power Circuit)		22
第3章 其他控制功能		
3-1 中斷功能(Interrupt Function)		25
3-2 重置功能(Reset Function)		25
3-3 頻率產生器(Frequency Generator)		26
3-4 預除器(Pre-divider)		27
3-5 Back-up 模式		27
第4章 指令說明		

第 1 章 功能簡介

1-1 前言

TM87 系列產品是一特別針對省電的電池應用而設計的四位元單晶片，晶片內部包含 ROM，RAM，Clock，I/O 及 LCD 驅動器，工作電壓可以從 1.5V 到 5V，內部 Data Bus 為 8 位元，每一個指令是 16 位元，是一精簡指令架構(RISC)，亦即每一行指令佔 2 個 Bytes(16 bits)，其效率相當之高。

其內部有兩種省電模式，一種是 Halt mode，即高速的 Clock 停住後，只剩下低速 Clock 在工作的模式，此時到耗電約為 3uA; 另一種是 Stop mode，即關掉所有的 Clock，此時幾乎是完全不耗電。

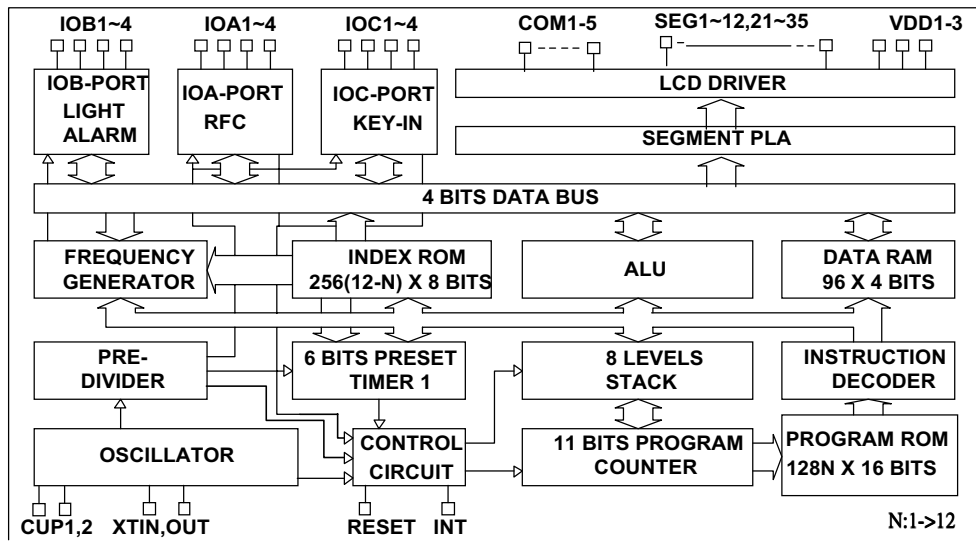
此系列內建有一個 PLA(Programmable Logic Array)架構的 LCD 驅動器可以讓寫程式的人任意編排他所要的 LCD 顯示圖案. 對於非矩陣式的 LCD 顯示來說, PLA 的架構遠比 Display RAM 的架構來的好用。

另外值得一提的優點是這一系列的 IC 的選擇相當的多與靈活，例如 Clock 可以選擇高低速同時使用(Dual Clock)，只有高速(Fast Only)或只有低速(Slow Only). I/O 腳，Buzzer，冷光板介面(EL light)均可與 LCD 驅動腳分享腳位，使能充分的運用到每一隻腳位而讓晶片發揮到更大的效用。

1-2 規格(Feature)

- 非常寬的工作電壓範圍: 1.2V~5.5V (經由 mask option 選擇)
- ROM 大小: 1536 x 16 bits
- RAM 大小: 96 x 4 bits
- 最大 LCD 驅動點數: 5 com x 27 segment -> 135 segments
- 每一隻 segment 都可以 mask option 選擇為 P open drain 或 DC output
- 最多可有 3 組(12 個 I/O port): I/OA1~4，IOB1~4，I/OC1~4
- 副程式(Subroutine)呼叫可有 8 層堆疊(Stack)
- 中斷功能
 - 外部中斷因子(Factor): 3 個，分別為 INT 腳，IOC 腳和 KI 腳
 - 內部中斷因子(Factor): 2 個，分別為預除器(Pre-divider)和 Timer1
- 內建冷光介面(EL Light)，鬧鈴(Alarm)，頻率(Frequency)及弦樂(Melody)產生器
- 內建一組 6 位元之可程式化之計時器(Programmable Timer)
- 內建看門狗計時器(Watchdog Timer)
- 雙時鐘(Dual Clock)操作
- 內建 LED 驅動掃描模式

1-3 功能區塊圖(Block Diagram)



1-4 腳位定義(Pin Assignment)

No	Name	No	Name
1	BAK	27	SEG21
2	XIN	28	SEG22
3	XOUT	29	SEG23
4	GND	30	SEG24/IOA1
5	VDD1	31	SEG25/IOA2
6	VDD2	32	SEG26/IOA3
7	VDD3	33	SEG27/IOA4
8	CUP1	34	SEG28/IOB1/ELC
9	CUP2	35	SEG29/IOB2/ELP
10	COM1	36	SEG30/IOB3/BZB
11	COM2	37	SEG31/IOB4/BZ
12	COM3	38	SEG32/IOC1/KI1
13	COM4	39	SEG33/IOC2/KI2
14	COM5	40	SEG34/IOC3/KI3
15	SEG1(K1)	41	SEG35/IOC4/KI4
16	SEG2(K2)	42	RESET
17	SEG3(K3)	43	INT
18	SEG4(K4)	44	TEST
19	SEG5(K5)		
20	SEG6(K6)		
21	SEG7(K7)		
22	SEG8(K8)		
23	SEG9(K9)		
24	SEG10(K10)		
25	SEG11(K11)		
26	SEG12(K12)		

1-5 腳位說明(Pin Description)

名稱	輸入/輸出	說明
BAK	電源	省電用，在 Li 電池模式下要接 0.1u 電容到地線。
VDD1,2,3	電源	供應 LCD 驅動器及 IC 的電源電壓。
RESET	輸入	系統重置信號。
INT	輸入	外部中斷要求信號，正緣或負緣觸發可由 mask option 選擇，內部電阻要 pull-up，pull-down 或者 open 也是經由 mask option 選擇。
TEST		測試用腳位，建議接到地線。
CUP1,2	輸出	產生電源用來供應 VDD1,2,3 的電壓，當 1/2 或 1/3 偏壓(Bias)時，必須跨接一個沒極性的電容，如果沒有選擇偏壓(Bias)則必須空接。
XIN XOUT	輸入 輸出	系統時鐘(System Clock)輸入。 系統時鐘(System Clock)輸出。 在雙時鐘模式或低速模式下外接 32.768KHz 的晶體振盪器(Crystal)或外接 RC 作為低速時鐘，若是選擇高速(Fast Only)模式，則可以外接電阻而形成 RC 震盪線路。
COM1~5	輸出	輸出信號去驅動 LCD 玻璃或 LED 的 common 腳。
SEG1~12 21~35	輸出	輸出信號去驅動 LCD 玻璃或 LED 的 segment 腳。 其中 SEG1~12 是與按鍵掃描輸出 K1~12 共用腳。
IOA1~4	輸入/輸出	輸入/輸出口 A，可用程式來設定內部 pull-low 電阻 這個腳位與 SEG24~27 共用腳位，可經由 mask option 來選擇。
IOB1~4	輸入/輸出	輸入/輸出口 B，可用程式來設定內部 pull-low 電阻 這個腳位與 SEG28~31/ELC,ELP,BZB, BZ 共用腳位，可經由 mask option 來選擇。
IOC1~4	輸入/輸出	輸入/輸出口 C，可用程式來設定內部 pull-low 電阻 這個腳位與 SEG32~35/KI1~4 共用腳位，可經由 mask option 來選擇。
ELC,ELP	輸出	冷光片顯示介面，用來驅動冷光片，與 SEG28~29 / IOB1~2 共用腳位。
BZB,BZ	輸出	鬧鐘輸出，可直接推 Buzzer，與 SEG30~31 / IOB3~4 共用腳位。
KI1~4	輸入	鍵盤掃描輸入腳。
GND	電源	接地腳。

第 2 章 內部系統結構

2-1 系統時鐘(System Clock)

本晶片共有四種不同的時鐘(Clock)模式，可經由 mask option 來選擇:

- a. 雙時鐘模式(Dual Clock)
內建高速 RC 振盪，約為 250KHz 或 500KHz(以 Mask Option 來選擇)，及外接 32.768KHz 的晶體或 RC 低速振盪(以 Mask Option 來選擇)。
- b. 高速時鐘模式且使用內部電阻(Fast Only and Use Internal Resistor)
用內部電阻產生高速 RC 振盪，約為 250KHz，XIN 及 XOUT 空接。
- c. 高速時鐘模式且使用外部電阻(Fast Only and Use External Resistor)
用外部電阻產生高速 RC 振盪，電阻跨接在 XIN 及 XOUT 之間。
- d. 低速時鐘模式(Slow Only)
外接 32.768KHz 的晶體低速振盪器或 RC 振盪跨接在 XIN 及 XOUT 之間。

2-2 程式記憶體(ROM)

TM8713 的 ROM 大小為 1536x16 bits，因為是精簡指令架構(RISC)所以每個指令只需要一個記憶空間，因而總共可寫 1536 行指令，ROM 可分為程式 ROM (Program ROM)及表格 ROM(Table ROM)，這兩種 ROM 其實是共同分享全部的 ROM 大小，而如何切分則由光罩選擇(Mask Option)來做，程式 ROM(Program ROM)的大小切分公式為: $(128*N) * 16 \text{ bits}$ ，而表格 ROM(Table ROM)的大小切分公式為: $256*(12-N)*8 \text{ bits}$ ， $N=1\sim 12$ ，兩者相加總和剛好是 $1536*16\text{bits}$ ，假設

$N=1$ 則程式 ROM(Program ROM)的大小: $(128*1)*16 \text{ bits}=128 * 16 \text{ bits}$

則表格 ROM(Table ROM)的大小: $256*(12-1)*8 \text{ bits}=2816 * 8 \text{ bits}$

$N=12$ 則程式 ROM(Program ROM)的大小: $(128*12)*16 \text{ bits}=1536 * 16 \text{ bits}$

則表格 ROM(Table ROM)的大小: $256*(12-12)*8 \text{ bits}=0 * 8 \text{ bits}$

由以上可以得知當程式 ROM 最大為 1536 時，表格 ROM 正值最小為 0; 而當表格 ROM 為最大 $2816 * 8$ 時，程式 ROM 正值最小為 $128 * 16$ 。

在 $1536 * 16 \text{ bits}$ 的 ROM 裡面，前面有 6 個中斷向量位址(Interrupt Vector Address)，當程式會用到該中斷時，其對應的中斷向量位址不能寫入一般程式。

名稱	位址
重置(Reset)	000H
外部中斷腳中斷(INT pin Interrupt)	010H
輸入口 C 中斷(IOC port Interrupt)	014H
計時器 1 中斷(Timer1 Interrupt)	018H
預除器中斷(Pre-divider Interrupt)	01CH
按鍵掃描中斷(Key Scan Interrupt)	024H

位址	
000H	Reset
010H	INT pin interrupt
014H	IOC or IOD interrupt
018H	Timer1 interrupt
01CH	Pre-divider interrupt
020H	No Use
024H	Key scan interrupt
028H	No Use
	16 bits

ROM 的位址對應圖

2-3 資料暫存器(RAM)

TM8713 的 RAM 大小為 $96 * 4$ bits，所有資料的存取都是以 4 位元(bits)為單位。程式對 RAM 的存取方式有兩種，一種是直接定指法(Direct Addressing Mode)，即直接對 RAM 做存取，另一種方法是索引定址法(Index Addressing Mode)，是利用索引暫存器(Index Register) HL 間接對 RAM 做存取的動作，為了共用模擬工具著想，存取的範圍也都是從 00H 到 7FH，但是中間 50H~6FH 是空著的，這是為了要能共用同一個 ICE(In Circuit Emulator)而設計。

特別要說明的是資料暫存器位址從 70H 到 7FH 的 16 個位址稱為工作暫存器(Working Register)，簡寫成 WR 或是指令集裡的 Ry，這是因為有一些指令是針對工作暫存器而設計的，例如 ADCI、SBCI、ANDI、EORI、LCT 等等，所以當程式設計師要使用這些指令做運算時，務必將資料先存在工作暫存器裡。

2-4 堆疊器(STACK)

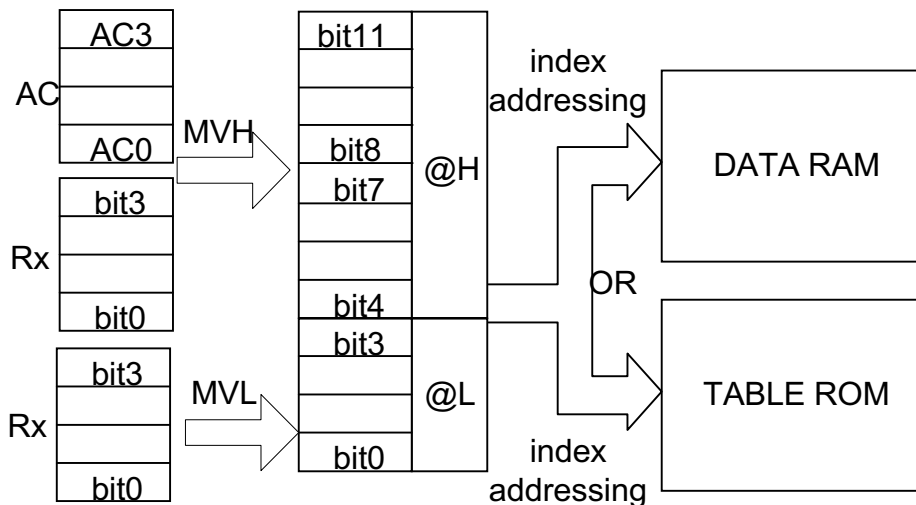
當程式執行 CALL 呼叫副程式(Subroutine)時，系統會先將現有位址，亦即程式計數器(Program Counter，簡稱 PC)先存到堆疊器內，等執行到 RTS 指令時才去堆疊器找回原有位址而存回程式計數器。此系列 IC 均有 8 層的堆疊器，亦即最多可連續呼叫 8 層副程式(Subroutine)。

2-5 累加器(Accumulator)

累加器是用來當作資料暫存器(RAM)及算術邏輯運算單元(簡寫為 ALU)中間運算時的緩衝(Buffer)。

2-6 索引暫存器(Index Register)

@HL 索引暫存器(@只是符號並無實際意義)是一個 12 位元的暫存器，其中@L 是 4 位元，而@H 是 8 位元，此暫存器是用於索引定址模式(Index Addressing Mode)的運算中，當執行 MVH 指令時，資料暫存器(RAM，指令表簡寫成 Rx)內的 4 位元數值會被搬入@L 暫存器中，當執行 MVH 指令時，累加器(Accumulator，簡寫 AC)的 4 位元數值會與資料暫存器(RAM，指令表簡寫成 Rx)內的 4 位元數值湊成 8 位元的數值而搬入@H 暫存器中，詳細請參照下圖:



2-7 十進位運算(Decimal Operation)

正常的情況下，IC 均是使用十六進位作為數值運算，然而經過十進制的轉換指令 DAA 之後，所有記憶體(RAM)、累加器(Accumulator)、表格 ROM 及立即運算數值(Immediate Data)都將變為十進位。下表是進位旗號(Carry Flag，簡寫為 CF)在加法運算前後對應累加器(AC)的變化:

在 DAA 運算之前的 AC 資料	在 DAA 運算之前的 CF 資料	在 DAA 運算之後的 AC 資料	在 DAA 運算之後的 CF 資料
$0 \leq AC \leq 9$	CF = 0	沒改變	沒改變
$A \leq AC \leq F$	CF = 0	AC = AC + 6	CF = 1
$0 \leq AC \leq 3$	CF = 1	AC = AC + 6	沒改變

<例 1>

```
LDS    10h,9      ;將立即運算數值 9 存入 RAM 10H 及 AC
LDS    11h,1      ;將立即運算數值 1 存入 RAM 11H 及 AC
RF     1h         ;將 CF 清除為 0
ADD*   10h        ;RAM 10H 的內容與 AC 作二進位相加，
                  ;(即 9+1=0Ah，CF=0) 結果存回 RAM 10H
                  ;及 AC
DAA*   10h        ;轉換內容成十進制
```

結果 RAM 10H 的內容會轉為”0”，而 CF 會變成”1”，也就是十進制的”10”。
 另外減法的運算也是一樣，下表是進位旗號(Carry Flag，簡寫為 CF)在減法運算前後對應累加器(AC)的變化:

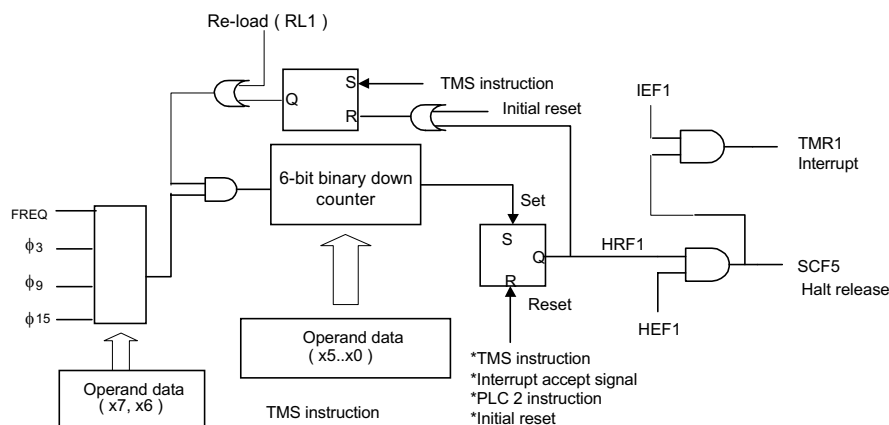
在 DAS 運算之前的 AC 資料	在 DAS 運算之前的 CF 資料	在 DAS 運算之後的 AC 資料	在 DAS 運算之後的 CF 資料
$0 \leq AC \leq 9$	CF = 1	沒改變	沒改變
$6 \leq AC \leq F$	CF = 0	AC= AC+ 0AH	沒改變

<例 2>

```
LDS    10h,1      ;將立即運算數值 1 存入 RAM 10H 及 AC
LDS    11h,2      ;將立即運算數值 2 存入 RAM 11H 及 AC
SF     1h         ;將 CF 設為 1 表示沒有借位
SUB*   10h        ;RAM 10H 的內容與 AC 作二進位相減，
                  ;(即 1-2=0FH，CF=0) 結果存回 RAM 10H
                  ;及 AC
DAS*   10h        ;轉換內容成十進制
```

結果 RAM 10H 的內容會轉為”9”，而 CF 會變成”0”，也就是十進制的”-1”。

2-8 計時器 1(TMR1)



2-8-1 一般動作

TMR1 包含了一個 6 位元的二進位倒數計數器，當計數到 3Fh 時將會產生 underflow 的信號而且會設 Halt 解除需求旗號 1(Halt Release Request Flag 1,HRF1)，這時如果 TMR1 中斷致能旗號 1(Interrupt Enable Flag1,IEF1)有設的話，中斷就會產生。

在電源打開的起始狀態，TMR1 預設的時鐘輸入(Clock input)為 $\phi 3$ (註 1)。當系統因看門狗時鐘(Watch Dog Timer)而產生重置(Reset)時，TMR1 的時鐘輸入(Clock input)仍會保留為上一次的設定。

(註 1) $\phi 3$: 預除器(Pre-divider)的第三級輸出，即(預除器頻率/ 2^3 Hz)。

2-8-2 重覆載入動作(Re-load Operation)

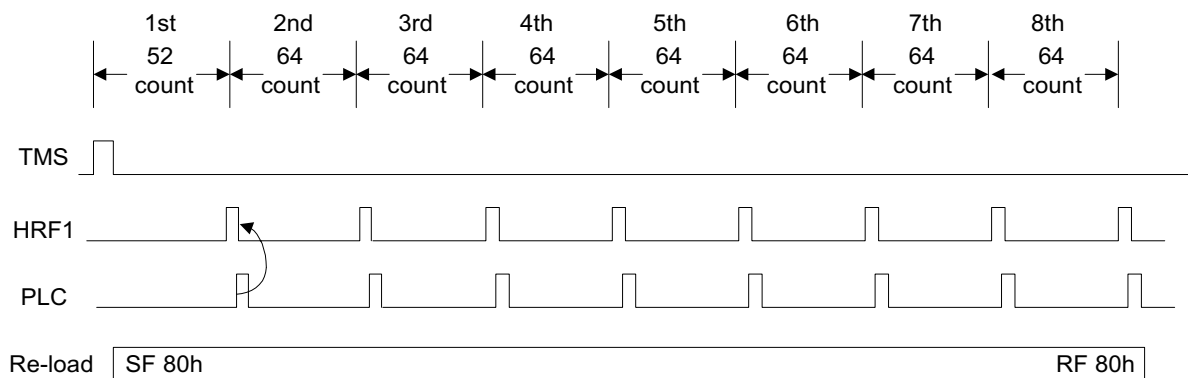
當計數的數字超過 3Fh 時就需要用重覆載入功能，SF 80h 指令啓動此功能，RF 80h 關掉此功能，當重覆載入功能啓動時，即使 TMR1 倒數到 3Fh 也不會產生 Underflow 而中止計數，在這期間，使用者必須利用 Halt 解除需求旗號(Halt Release Request Flag ,HRF)或中斷去查核計數數值是否為所希望的數值。

<注意>

在希望發生的最後一個 Halt Release 或中斷發生之前，請絕對不要關掉重覆載入功能，只要一關掉這功能，重覆計數的功能馬上停止。

<範例>

假設我們希望 TMR1 計數 500 個單位，亦即要計數 $64 * 7 + 52$ ，時序圖及程式的寫法如下:



```

Re-load SF 80h
LDS 0, 0 ;清除 underflow 計數暫存器
PLC 2
SHE 2 ;設定 HALT release 發生因子是 TMR1
TMSX 34h ;設定 TMR1 數值(52)及設定時鐘輸入是 $\phi 3$ 
SF 80h ;啓動重覆載入功能
    
```

RE_LOAD:

```

HALT
INC* 0 ;underflow 計數暫存器加一
PLC 2 ;清除 HRF1
    
```

JB3 END_TM1 ;如果 underflow 計數暫存器等於 8 則停止計數
 JMP RE_LOAD ;

END_TM1:

RF 80h ;關掉重覆載入功能

2-9 狀態暫存器(Status Register, STS)

TM8713 總共有 5 個狀態暫存器(Status Register)，每一個狀態暫存器是 4 bits：

2-9-1 狀態暫存器 1(Status Register1，STS1)

Bit 3	Bit 2	Bit 1	Bit 0
CF	ZF	X	X

CF	說 明
1	當加法有進位或減法有借位時
0	除了上述以外的情況

ZF	說 明
1	當累加器(Accumulator)等於 0
0	當累加器(Accumulator)不等於 0

2-9-2 狀態暫存器 2(Status Register2，STS2)

Bit 3	Bit 2	Bit 1	Bit 0
X	Start Condition Flag2 (SCF2)	Start Condition Flag1 (SCF1)	Back-up Flag (BCF)

SCF2	說 明
1	Halt Release 是因為 SCF4,5,7 的變化而產生
0	除了上述以外的情況

SCF1	說 明
1	Halt Release 是因為 IOC port 的變化而產生
0	除了上述以外的情況

BCF	說 明
1	更多的電流將供應給震盪器(Oscillator)以使 IC 更穩定
0	除了上述以外的情況

2-9-3 狀態暫存器 3(Status Register3，STS3)

Bit 3	Bit 2	Bit 1	Bit 0
Start Condition Flag7 (SCF7)	預除器第 15 級輸出 狀態 (PRED)	Start Condition Flag5 (SCF5)	Start Condition Flag4 (SCF4)

SCF7	說 明
1	Halt Release 是因為預除器的溢位而產生
0	除了上述以外的情況

PRED	說 明
1	預除器的第 15 級輸出為 1
0	除了上述以外的情況

SCF5	說 明
1	Halt Release 是因為 TMR1 underflow 而產生
0	除了上述以外的情況

SCF4	說 明
1	Halt Release 是因為 INT 腳而產生
0	除了上述以外的情況

2-9-4 狀態暫存器 3X(Status Register3X，STS3X)

Bit 3	Bit 2	Bit 1	Bit 0
X	X	X	Start Condition Flag8 (SCF8)

SCF8	說 明
1	Halt Release 是因為掃描鍵輸入腳 KI1~4 有被按而產生
0	除了上述以外的情況

2-9-5 狀態暫存器 4(Status Register4，STS4)

Bit 3	Bit 2	Bit 1	Bit 0
X	X	看門狗時鐘啓動旗 號(WDF)	系統時鐘選擇旗號 (CSF)

WDF	說 明
1	啓動看門狗時鐘(Watch Dog Timer)
0	關掉看門狗時鐘(Watch Dog Timer)

CSF	說 明
1	表示系統時鐘是在高速狀態(Fast Clock Mode)
0	表示系統時鐘是在低速狀態(Slow Clock Mode)

2-10 控制暫存器(Control Register, CTL)

總共有 4 個控制暫存器，分別為 CTL1~CTL4，每一個暫存器有的 bit 數多少不一定。

2-10-1 控制暫存器 1(Control Register1，CTL1)

Bit 4
啓動 Halt Release 是由於 IOC 信號的變化 (SEF4)

2-10-2 控制暫存器 2(Control Register2，CTL2)

Bit 6	Bit 5	Bit 4
X	啓動 Halt Release 是由於按鍵掃描 (HEF5)	X
Bit3	Bit 2	Bit 1
啓動 Halt Release 是由於預除器溢位 (HEF3)	啓動 Halt Release 是由於 INT 腳 (HEF2)	啓動 Halt Release 是由於 TMR1 Underflow(HEF1)

2-10-3 控制暫存器 3(Control Register3，CTL3)

Bit 6	Bit 5	Bit 4	Bit 3
X	啓動中斷會由按鍵掃描引起 (IEF5)	X	啓動中斷會由預除器溢位引起 (IEF3)
Bit3	Bit 2	Bit 1	
啓動中斷會由 INT 腳引起 (IEF2)	啓動中斷會由 TMR1 Underflow 引起 (IEF1)	啓動中斷會由 IOC 信號變化引起(IEF0)	

2-10-4 控制暫存器 4(Control Register4，CTL4)

Bit 7	Bit 5	Bit 4
啓動 Stop 解除會由 K1~4 信號改變引起 (SRF7)	啓動 Stop 解除會由 INT 腳信號改變引起 (SRF5)	啓動 Stop 解除會由 IOC 信號改變引起 (SRF4)

2-11 蜂鳴器輸出腳(Buzzer Output Pin)

TM8713 有兩支輸出腳，分別為 BZ 及 BZB，這兩支腳與 IOB3 和 IOB4 共用輸出腳，輸出的頻率有 1K、2K、4K 及 FREQ 信號頻率，另外也可以當作直流輸出，詳細請參照指令 ALM 的說明。

當蜂鳴器輸出腳配合 Timer 和頻率產生器(Frequency Generator)時也可以用來當作紅外線遙控器(IR Remote Controller)，此時頻率產生器(Frequency Generator)的設定值必須大於或等於 3，並且 ALM 指令必須緊跟在 FRQ 指令之後，範例程式如下：

```

SHE      1      ;啓動 TMR1 halt release 旗號.
TMSX    3Fh    ;設 TMR1 的數值為 3Fh 及時鐘來源為φ9.
SCC     40h    ;設頻率產生器的時鐘來源為 BCLK.
FRQX    2, 3   ;FREQ = BCLK / (4*2), 設頻率產生器的值為 3
          ;且 Duty cycle 是 1/2

ALM     1C0h   ;FREQ 信號輸出.

HALT                    ;等 halt release 因為 TMR1 而解除.
ALM     0      ;停止蜂鳴器輸出

```

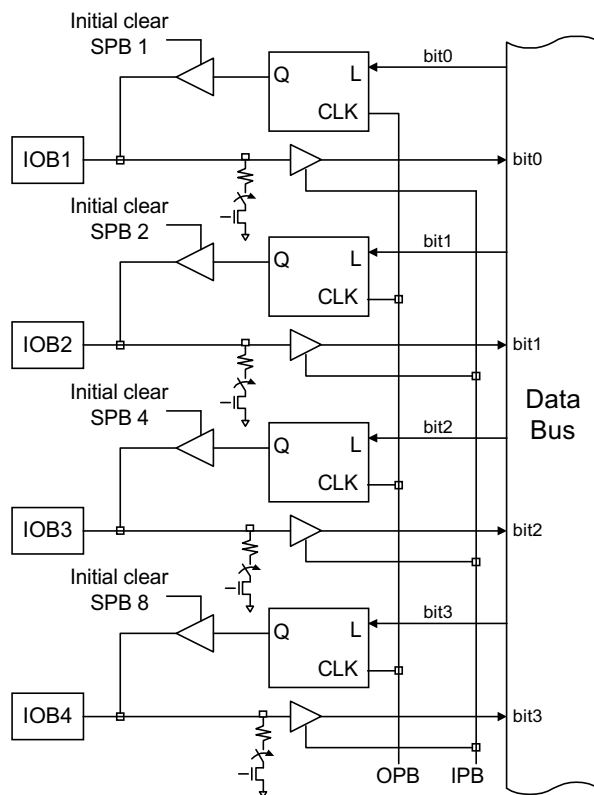
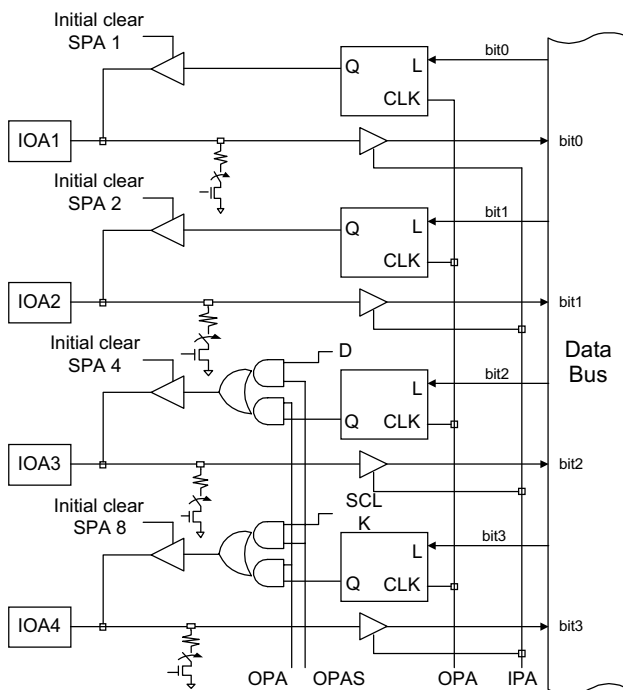
2-12 輸入/輸出埠(Input/Output Pin)

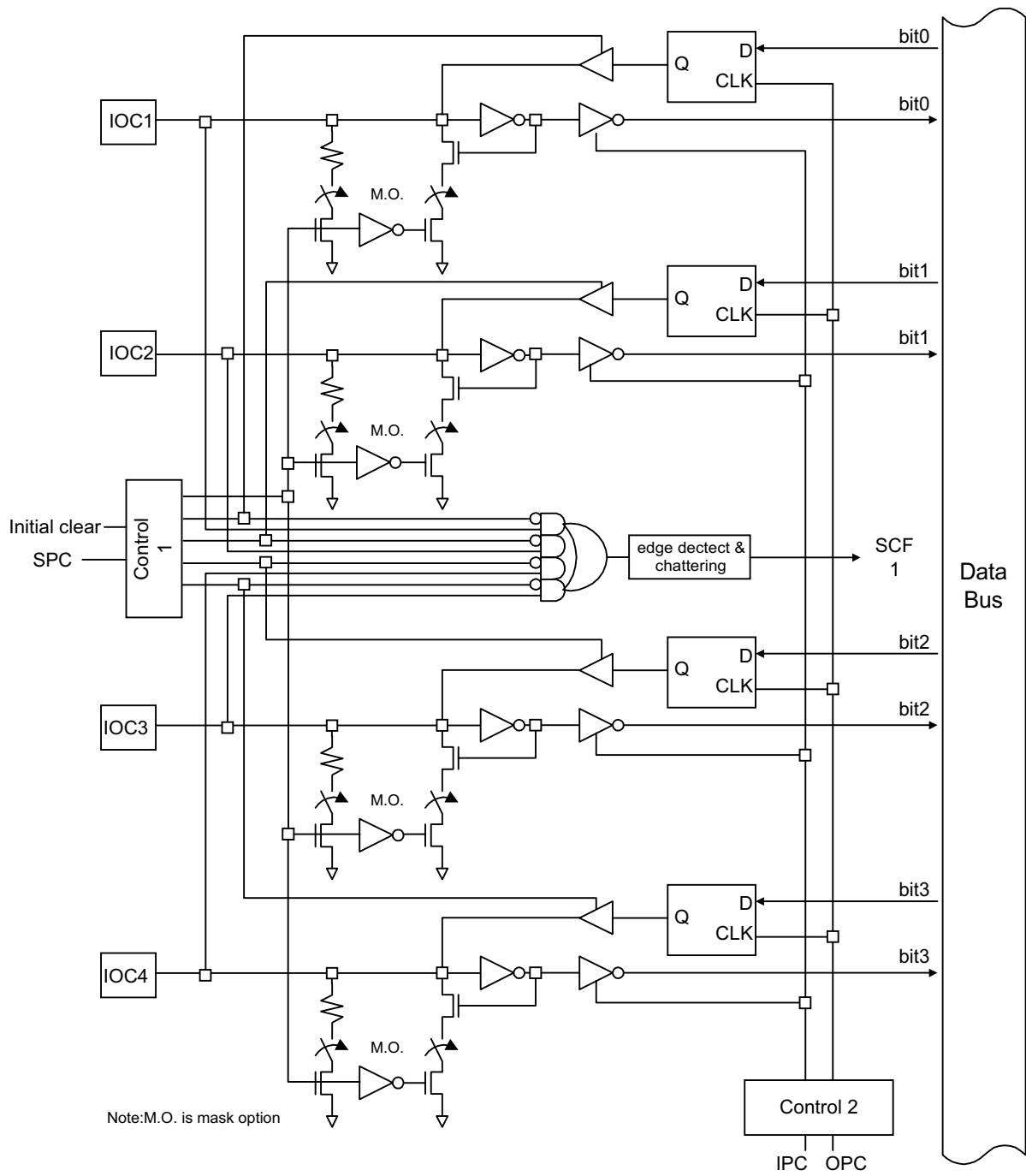
TM8713 總共有三組 12 個輸入/輸出埠，分別為 IOA、IOB 及 IOC，除了 IOC 埠當成輸入埠時有消除抖動(Chattering Cancel)的功能以外，其於兩組只能當一般的輸入/輸出埠，另外為了節省外接電阻，所有輸入/輸出埠當成輸入埠時都有下拉電阻(Pull-down Resistor)，此一架構對於按鍵的應用很有幫助，若是不需要下拉電阻(Pull-down Resistor)，也可以用 SF 指令將它關掉。

IOC 埠裡面還有一個 Low Level Hold 的功能，此一功能須透過光罩選擇(Mask Option)來選，當下拉電阻(Pull-down Resistor)及 Low Level Hold 功能都存在的情況，開機重置(Power-on Reset)時會自動啓動下拉電阻(Pull-down Resistor)而關掉 Low Level Hold 功能，執行 SPC 10h 指令可以獲得同樣的效果，如果執行 SPC 0h 則剛好相反，會關掉下拉電阻(Pull-down Resistor)而啓動 Low Level Hold 功能。這些功能都只有在 IOC 埠當成輸入埠時才有。當 IOC 埠為輸出埠時，下拉電阻(Pull-down Resistor)及 Low Level Hold 功能都自動會關掉。

IOC 埠消除抖動(Chattering Cancel)的頻率還可以有三種不同的選擇，可以用 SCC 指令來選。

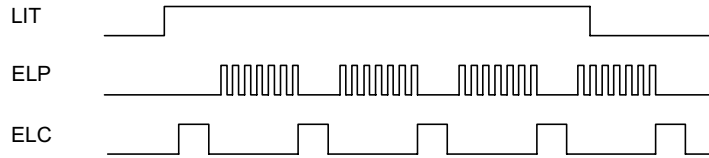
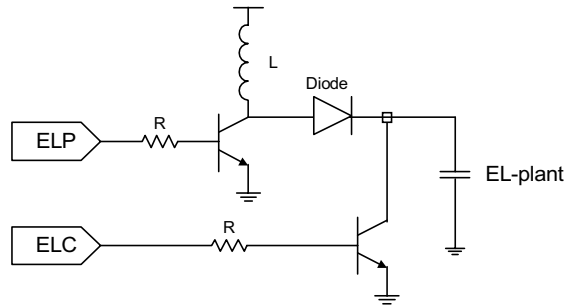
下圖為 IOA,IOB 及 IOC 埠的硬體架構:





2-13 冷光板驅動線路(EL-Plant Driver)

TM8713 提供冷光驅動線路，可以透過外接線路將電壓升至交流(AC) 100V 以上，參考的線路及時序圖如下：



<範例>

ELC 110h ;設 ELP 腳輸出 2/3Duty 的 BCLK 時鐘及 ELC 腳輸出
;1/4 duty 的 Φ 8 時鐘.

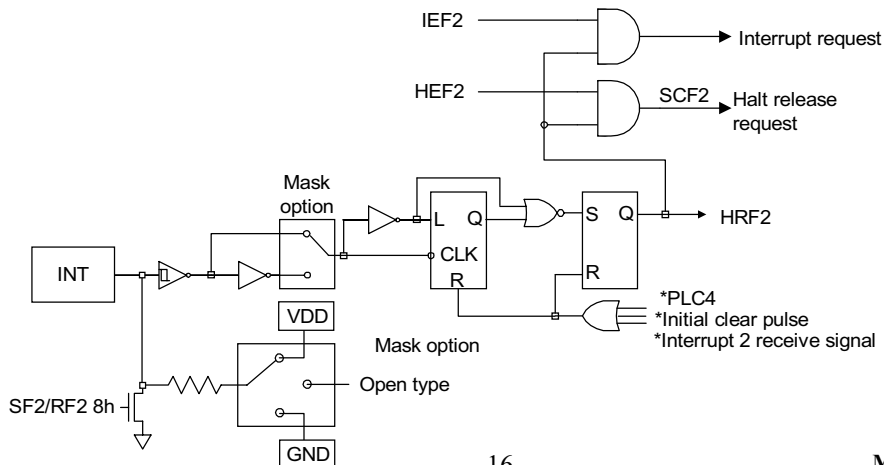
SF 4h ;啓動冷光驅動線路

.....

RF 4h ;關閉冷光驅動線路

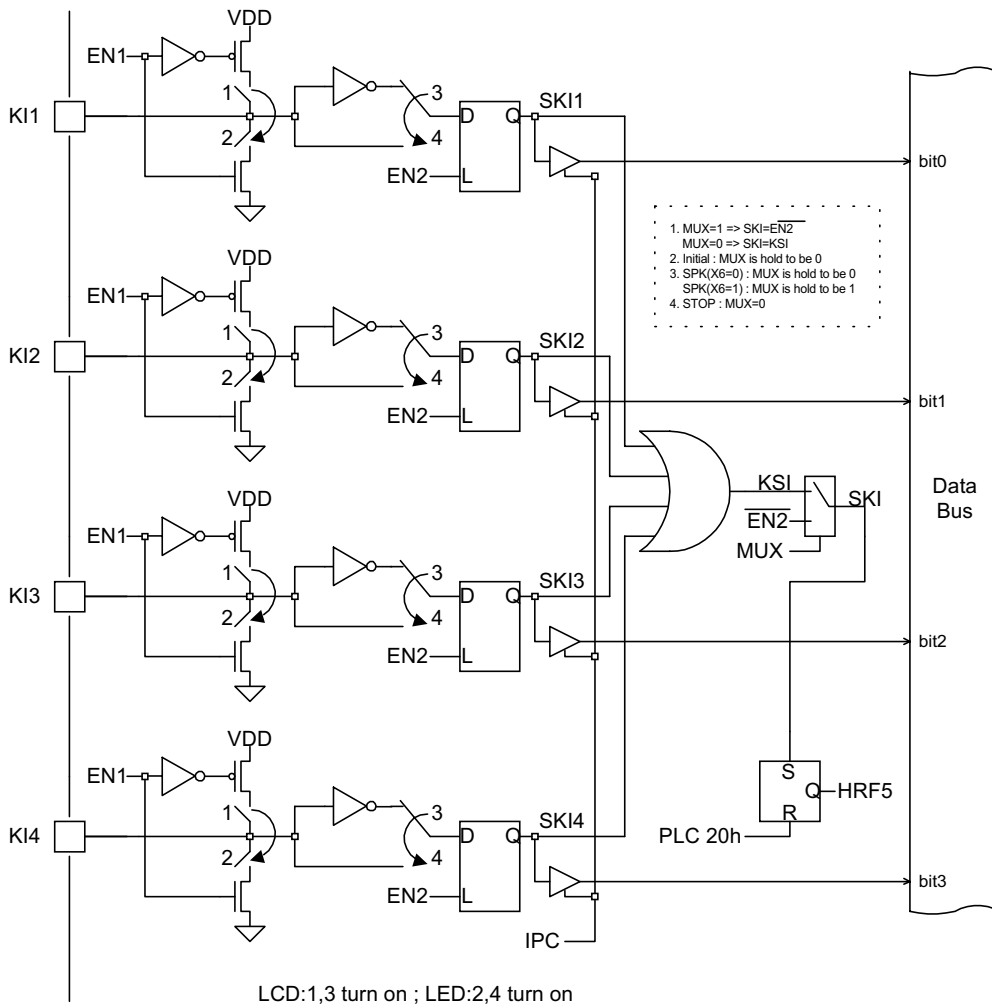
2-14 外部中斷線路(External Interrupt)

INT 腳總共有上拉電阻(Pull-up Resistor)、下拉電阻(Pull-down Resistor)及全開(Open)等三種模式，可經由光罩選擇來選取，中斷腳的內部結構線路如下圖所示：



2-15 按鍵掃描(Key Board Scanning)

TM8713 的按鍵掃描輸出腳 SK1~12 是與 LCD Segment 腳 SEG1~12 共用同一支腳位，而且是同時存在，按鍵掃描是例用 LCD 時序的一小段時間來做，而按鍵掃描的輸入腳 KI1~4 則與 LCD Segment 腳 SEG32~35 共用同一支腳位，但是必須經由光罩選擇(Mask Option)來選取一種用途，詳細硬體的架構及範例程式如下：



SPK	10h	;啓動所有的按鍵掃描輸出腳.
SHE	20h	;啓動 Halt 會因爲按鍵掃描而解除
HALT		
MCX	10h	;讀 SCF8 旗號(SKI).
JB0	ski_release	;清除 HRF5(SKI)

ski_release:

IPC	10h	;讀 KI1~4 輸入的擷取值.
-----	-----	------------------

```

JB0    ki1_release
JB1    ki2_release
JB2    ki3_release
JB3    ki4_release
.
ki1_release:
SPK    40h          ;只啓動 SK1 掃描輸出(第 1 排).
PLC    20h          ;清除 HRF5 以避免模式切換時引起錯誤的 Halt 解除
CALL   wait_scan_again ;等時間結束停止 LCD 時鐘以確定再掃描
IPC    10h          ;讀 KI1 輸入的擷取值
JB0    ki1_seg1
.
.
SPK    4fh          ; 只啓動 SKF 掃描輸出(第 16 排).
PLC    20h          ;清除 HRF5 以避免模式切換時引起錯誤的 Halt 解除
CALL   wait_scan_again ;等時間結束停止 LCD 時鐘以確定再掃描
IPC    10h          ;讀 KI1 輸入的擷取值
JB0    ki1_seg16
.
.
wait_scan_again:
HALT
PLC    20h
RTS
read_scf5:
MSC    10h          ;讀 SCF5(TMR1)旗號
JB1    exit_wait
JMP    read_scf5
exit_wait:
RTS

```

2-16 液晶驅動輸出腳(LCD Driver Output Pin)

TM8713 的 LCD 驅動輸出腳，也可以用光罩選擇(Mask Option)選成 LED mode，但驅動 LED 時需外加電晶體，當不是選 LED mode 時，Common 腳與 Segment 腳就用來做 LCD 的驅動或者直流輸出(DC Output)。LCD 的輸出方式可分為 Static，1/2 bias 1/2 duty，1/2 bias 1/3 duty，1/2 bias 1/4 duty，1/2 bias 1/5 duty，1/3 bias 1/3 duty，1/3 bias 1/4 duty，1/3 bias 1/5 duty，光罩選擇(Mask Option)在不同的輸出方式對應的頻率如下：

LCD duty cycle	Static			
Mask option	LCD not used	Slow	Typ.	Fast
LCD 掃描頻率	0Hz	32Hz	32Hz	64Hz

LCD duty cycle	1/2 duty			
Mask option	LCD not used	Slow	Typ.	Fast
LCD 掃描頻率	0Hz	16Hz	32Hz	64Hz

LCD duty cycle	1/3 duty			
Mask option	LCD not used	Slow	Typ.	Fast
LCD 掃描頻率	0Hz	21Hz	42Hz	85Hz

LCD duty cycle	1/4 duty			
Mask option	LCD not used	Slow	Typ.	Fast
LCD 掃描頻率	0Hz	16Hz	32Hz	64Hz

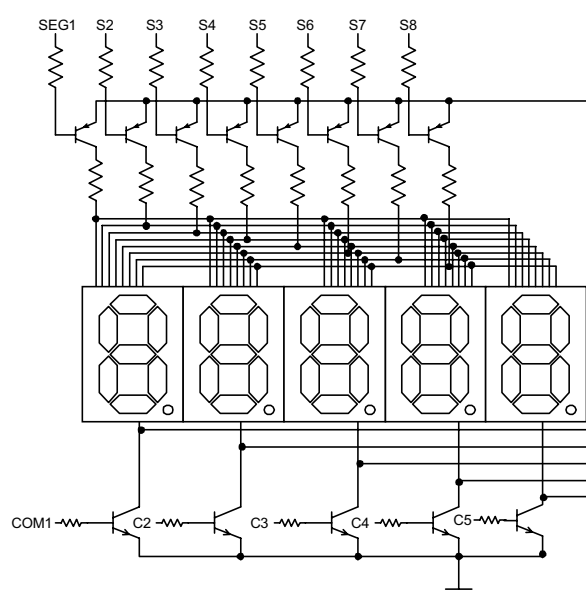
LCD duty cycle	1/5 duty			
Mask option	LCD not used	Slow	Typ.	Fast
LCD 掃描頻率	0Hz	25Hz	51Hz	102Hz

上述 Segment 腳可以光罩選擇(Mask Option)當做直流輸出(DC Output)，直流輸出又可分為 CMOS 的直流輸出及 P open-drain 的直流輸出，所以 Segment 腳有些拿來當 LCD 驅動，另一些拿來當輸出腳的現象是可能存在的。

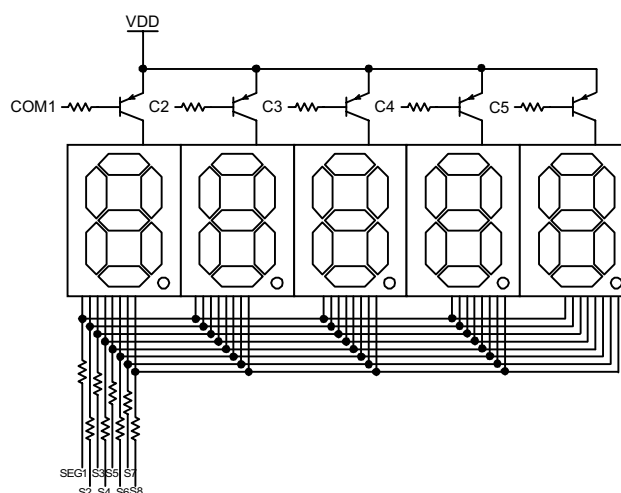
在定義 LCD 的*.cfg 檔裡，”COM”欄位填 0 表示是 CMOS 輸出，填 9 表示是 P open-drain 輸出。

當選成 LED mode 時，又可分為 High 工作模式及 Low 工作模式，其應用線路如下：

(1) High 工作模式



(2) Low 工作模式



<注意>請務必留意驅動電流大小避免將 IC 燒壞

同樣的當工作於 LED mode 時，不同的 Duty 對應的掃描頻率亦不相同，如下所示：

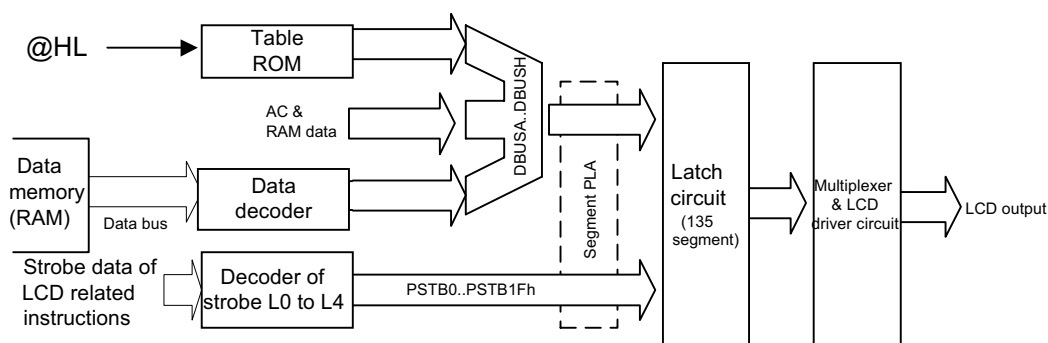
LED duty cycle	Static		
Mask option	Slow	Typ.	Fast
LED 掃描頻率	32Hz	32Hz	64Hz

LED duty cycle	1/2 duty		
Mask option	Slow	Typ.	Fast
LED alternating frequency	16Hz	32Hz	64Hz

LED duty cycle	1/3 duty		
Mask option	Slow	Typ.	Fast
LED alternating frequency	21Hz	42Hz	85Hz

LED duty cycle	1/4 duty		
Mask option	Slow	Typ.	Fast
LED alternating frequency	16Hz	32Hz	64Hz

TM87 系列 LCD 驅動器的結構是一個可程式邏輯陣列(Programmable Logic Array, PLA)的方式，不同於一般 Display RAM 的方式，本系列在使用 LCD 驅動器前需要事先定義 PLA 的內容，詳細區塊圖如下圖所示：



區塊圖含有幾個部分分別說明如下：

- (1) 資料解碼器(Data Decoder)用來解從資料暫存器(RAM)及表格 ROM(Table ROM)送過來的資料
- (2) 擷取(Latch)線路用來儲存 LCD 點亮的資料
- (3) L0~L4 解碼器用來解 LCD 相關的指令所指定的 strobe 資料(從 00h 到 1Fh)
- (4) 多工器(Multiplexer)用來選 1/2Duty，1/3Duty，1/4Duty 或 1/5Duty
- (5) LCD 驅動線路
- (6) 連接資料解碼器(Data Decoder)，L0~L4 解碼器和擷取(Latch)線路的 Segment 可程式邏輯陣列(Programmable Logic Array , PLA)

資料解碼器(Data Decoder)與解出來的 DBUSA~DBUSH 的對應表格如下：

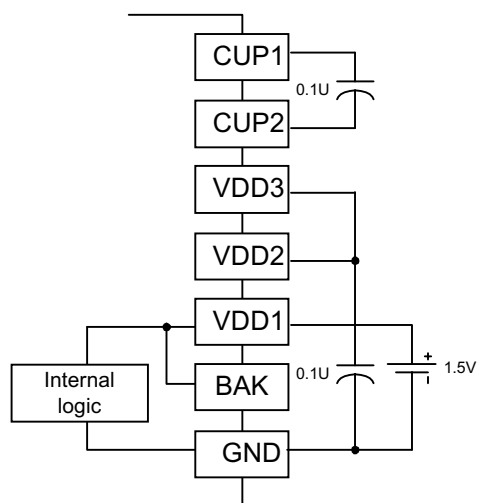
資料解碼器的內容	資料解碼器的輸出							
	DBUSA	DBUSB	DBUSC	DBUSD	DBUSE	DBUSF	DBUSG	DBUSH
0	1	1	1	1	1	1	0	1
1	0	1	1	0	0	0	0	1
2	1	1	0	1	1	0	1	1
3	1	1	1	1	0	0	1	1
4	0	1	1	0	0	1	1	1
5	1	0	1	1	0	1	1	1
6	1	0	1	1	1	1	1	1
7	1	1	1	0	0	*	0	1
8	1	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1	1
A-F	0	0	0	0	0	0	0	0

*<注意> 資料解碼器的內容”7”解碼出來的 DBUSF 可以光罩選擇(Mask Option)選為 0 或 1

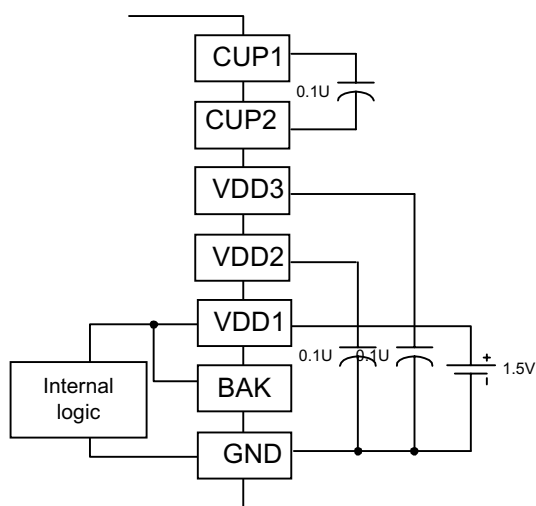
2-17 電源線路的接法(The Connection of Power Circuit)

本系列選擇不同的電源及偏壓時電源線路的接法也會不同，詳細如下：

2-17-1 Ag 電池模式及 1/2 Bias 或 Static

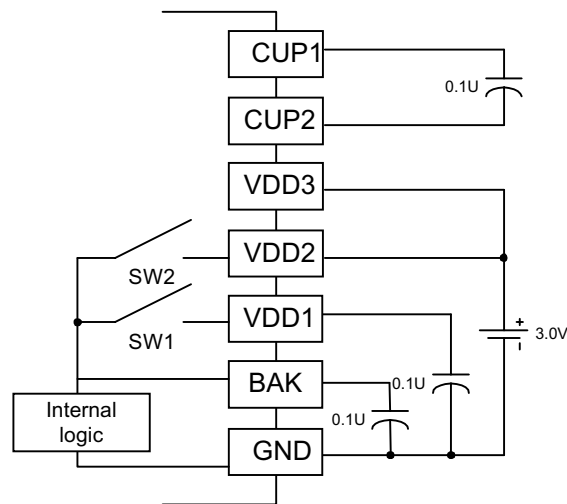


2-17-2 Ag 電池模式及 1/3 Bias



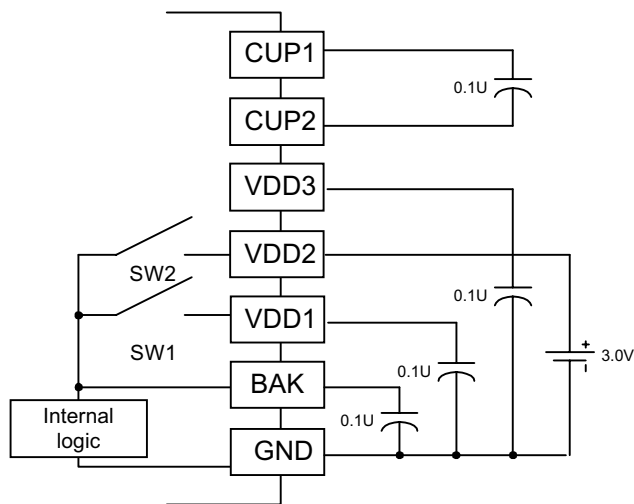
2-17-3 Li 電池模式及 1/2 Bias 或 Static

Backup flag(BCF)	SW1	SW2
BCF=0	ON	OFF
BCF=1	OFF	ON

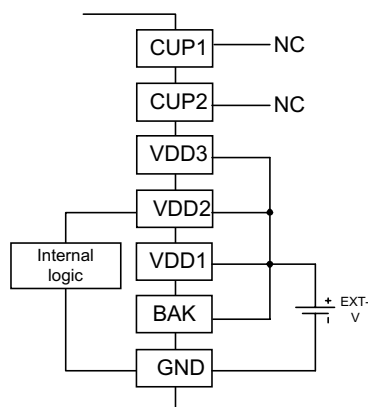


2-17-4 Li 電池模式及 1/3 Bias

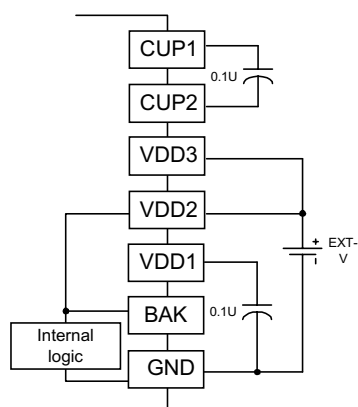
Backup flag(BCF)	SW1	SW2
BCF=0	ON	OFF
BCF=1	OFF	ON



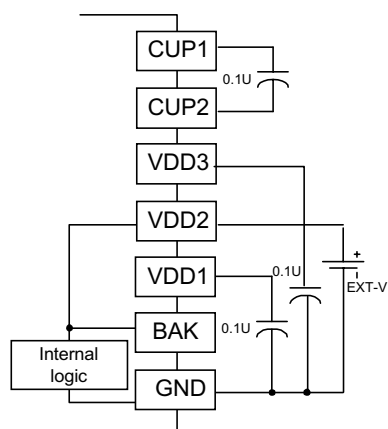
2-17-5 EXTV 電池模式及 Static



2-17-6 EXTV 電池模式及 1/2 Bias



2-17-7 EXTV 電池模式及 1/3 Bias



第 3 章 其它控制功能(Other Control Function)

3-1 中斷功能(Interrupt Function)

TM8713有7個中斷來源：3個外部中斷因子(Factor)及2個內部中斷因子(Factor)，這5個中斷的向量位址，權位高低及對應的中斷啓動旗號如下：

中斷因子	INT pin	IOC port	TMR1 underflow	Predivider overflow	Key_Board Scanning
中斷的向量位址	010H	014H	018H	01CH	024H
中斷啓動旗號	IEF2	IEF0	IEF1	IEF3	IEF5
權位高低	4 th	3 th	2 nd	1 st	5 th

3-2 重置功能(Reset Function)

重置(Reset)的方式共有開機重置(Power-on Reset)，RESET腳重置，看門狗重置(Watch dog Reset)及IOC埠重置，系統經過Reset後，所有信號均回復到起始狀態，這些信號的起始值如下：

程式計數器(Program counter)	(PC)	位址 000H
Start condition flags 1 to 7	(SCF1-7)	Reset mode
Backup flag	(BCF)	Set mode (Ag, Li 模式) Reset mode (EXTV 模式)
Stop release enable flags 4,5,7	(SRF4,5,7)	Reset mode
Switch enable flags 4	(SEF4)	Reset mode
Halt release request flag	(HRF 0~6)	Reset mode
Halt release enable flags 1 to 3	(HEF1-6)	Reset mode
Interrupt enable flags 0 to 3	(IEF0-6)	Reset mode
Alarm output	(ALARM)	直流輸出(DC output) 0
Pull-down flags in I/OC		Set mode
Input/output ports I/OA, I/OB, I/OC	(I/OA, I/OB, I/OC 埠)	輸入模式(Input mode)
I/OC port chattering clock	Cch	$\phi 10^*$
EL-light driver pumping clock source and duty cycle	Celp	$\phi 0$, duty cycle 爲 3/4
EL-light driver clearing clock source and duty cycle	Celc	$\phi 8$, duty cycle 爲 1/4
Frequency generator	Cfq	$\phi 0$, duty cycle 爲 1/4,沒有輸出的動作
LCD driver output		全亮或全滅(看 mask option)*
Timer 1		停止沒動作
Watchdog timer	(WDT)	Reset mode, WDF = 0
Reset Type	Level or Pulse	利用光罩(Mask)選擇
Reset Time	$\phi 15/2$ or $\phi 12/2$	利用光罩(Mask)選擇
Clock source	(BCLK)	低速時鐘(在雙時鐘模式下)

3-3 頻率產生器(Frequency Generator)

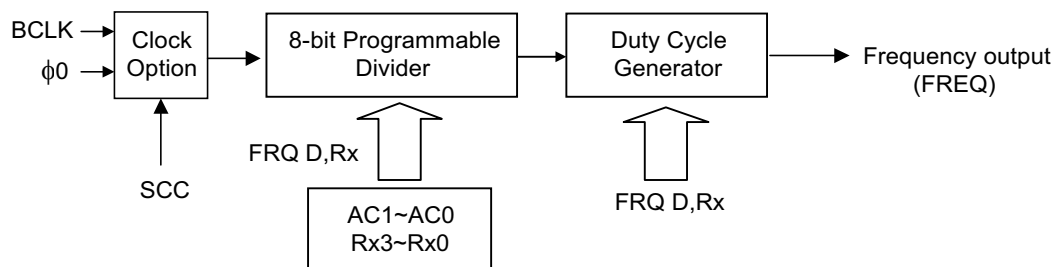
頻率產生器可用來產生鬧鈴(Alarm)及TMR1的時鐘來源，輸出時鐘可用下列公式計算：

$$\text{FREQ} = (\text{時鐘來源(Clock source)}) / ((N+1) * X) \text{Hz} \quad (X=1,2,3,4 \text{ 對應 } 1/1, 1/2, 1/3, 1/4 \text{ Duty})$$

(N為FRQ指令的運算子8位元資料)

料)

這裡的時鐘來源指的是BCLK或 $\phi 0$ ，選擇可用SCC指令來選，頻率產生器的區塊圖如下：



頻率產生器也可以用來產生單頻(Single Tone)Melody，它的音階如下面表格所示：

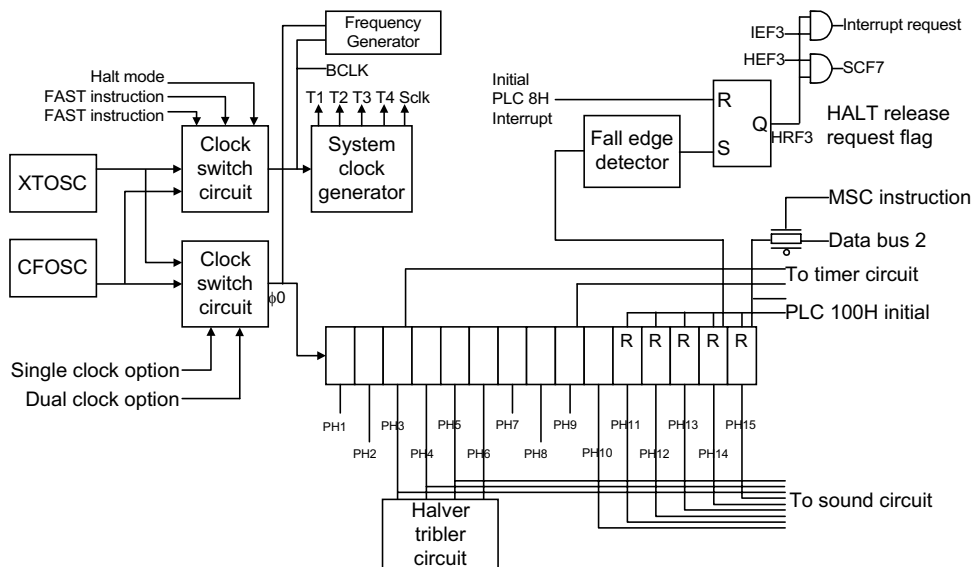
Tone	N 數值	FREQ	Ideal	%	Tone	N	FREQ	Ideal	%
C2	249	65.5360	65.4064	0.19	C4	62	260.063	261.626	-0.60
#C2	235	69.4237	69.2957	0.18	#C4	58	277.695	277.183	0.18
D2	222	73.4709	73.4162	0.07	D4	55	292.571	293.665	-0.37
#D2	210	77.6493	77.7817	-0.17	#D4	52	309.132	311.127	-0.64
E2	198	82.3317	82.4069	-0.09	E4	49	327.680	329.628	-0.59
F2	187	87.1489	87.3071	-0.18	F4	46	348.596	349.228	-0.18
#F2	176	92.5650	92.4986	0.07	#F4	43	372.364	369.994	0.64
G2	166	98.1078	97.9989	0.11	G4	41	390.095	391.995	-0.48
#G2	157	103.696	103.826	-0.13	#G4	38	420.103	415.305	1.16
A2	148	109.960	110.000	-0.04	A4	36	442.811	440.000	0.64
#A2	140	116.199	116.541	-0.29	#A4	34	468.114	466.164	0.42
B2	132	123.188	123.471	-0.23	B4	32	496.485	493.883	0.53
C3	124	131.072	130.813	0.20	C5	30	528.516	523.251	1.01
#C3	117	138.847	138.591	0.19	#C5	29	546.133	554.365	-1.48
D3	111	146.286	146.832	-0.37	D5	27	585.143	587.330	-0.37
#D3	104	156.038	155.563	0.31	#D5	25	630.154	622.254	1.27
E3	98	165.495	164.814	0.41	E5	24	655.360	659.255	-0.59
F3	93	174.298	174.614	-0.18	F5	22	712.348	698.456	1.99
#F3	88	184.090	184.997	-0.49	#F5	21	744.727	739.989	0.64
G3	83	195.048	195.998	-0.48	G5	20	780.190	783.991	-0.48
#G3	78	207.392	207.652	-0.13	#G5	19	819.200	830.609	-1.37
A3	73	221.405	220.000	0.64	A5	18	862.316	880.000	-2.01
#A3	69	234.057	233.082	0.42	#A5	17	910.222	932.328	-2.37
B3	65	248.242	246.942	0.53	B5	16	963.765	987.767	-2.43

<註>

- (1) 時鐘來源是 $\phi 0$ 如32,768Hz
- (2) Duty設為1/2Duty(設D=0)
- (3) FREQ代表輸出頻率
- (4) Ideal表示理想的音階頻率值
- (5) %表示誤差百分比

3-4 預除器(Pre-divider)

預除器是一個15階的計數器，計數的時鐘來源為 $\phi 0$ ，當 $\phi 0$ 從”H”準位(Level)變為”L”準位(Level)時，計數器的內容會改變，當執行PLC 100H的指令後 $\phi 11$ 到 $\phi 15$ 將會被清為0，由預除器產生的信號將供應給LCD驅動線路，系統時鐘，中止解除需求及I/O埠消除抖動使用，詳細區塊圖如下：



3-5 Back-up 模式

因為TM87系列的內部線路設計都是針對省電的觀念來設計，所以當有較大負載(Heavy Load)產生時，就必須進入Back-up模式才能防止IC產生誤動作，所謂較大負載(Heavy Load)如掃描按鍵、點亮LED及啟動鬧鈴(ALARM)等需要耗費較大電流去驅動的工作。在重置(Reset)的狀態下當電源選為Li或Ag時，BCF的旗號起始值為1，所以在Reset動作結束後，程式剛開始時，必須先把BCF關閉，否則會有大電流發生；如果電源是選EXT-V的話，BCF的旗號起始值為0，程式就不用把BCF關閉。

第 4 章 指令說明

- 在使用 RAM 工作前請記得必須先將 RAM 起始化(Initialize)，因為在電源打開時 RAM 的值是未知的。
- 工作記憶體(Working Register)也是記憶體(RAM)的一部份，他們的關係如下
工作記憶體(Working Register) $R_y =$ 記憶體(RAM) $R_x + 70h$

<Note> R_y : 工作記憶體(Working Register) ，位址範圍從 0~Fh，相對應的記憶體(RAM)

絕對位址為 70~7Fh。

工作記憶體(Working Register) R_y	對應的記憶體(RAM) R_x
0H	70H
1H	71H
2H	72H
.	.
.	.
.	.
DH	7DH
EH	7EH
FH	7FH

4-1 輸入/輸出指令(INPUT / OUTPUT INSTRUCTIONS)

指令	功能
LCT Lz, Ry	LCD latch [Lz] \leftarrow data decoder \leftarrow [Ry]

<說明>

將工作記憶體(Working Register) R_y 的內容值經由數值解碼器(Data Decoder)存入 Lz 所指的 LCD Latch。

指令	功能
LCB Lz, Ry	LCD latch [Lz] \leftarrow data decoder \leftarrow [Ry]

<說明>

將工作記憶體(Working Register) R_y 的內容值經由數值解碼器(Data Decoder)存入 Lz 所指的 LCD Latch，與 LCT 不同的是假如 R_y 的內容是 0，則數值解碼器(Data Decoder)輸出的值全為 0。

指令	功能
LCP Lz, Ry	LCD latch [Lz] ← [Ry] &AC

<說明>

將工作記憶體(Working Register)Ry 的內容值和累加器(Accumulator)的值存入 Lz 所指的 LCD Latch。

指令	功能
LCD Lz,@HL	LCD latch [Lz] ← [T@HL]

<說明>

將@HL 所指的表格 ROM(Table ROM)的內容值直接存入 Lz 所指的 LCD Latch。

指令	功能
LCT Lz,@HL	LCD latch [Lz] ← data decoder← [@HL]

<說明>

將索引暫存器@HL 所指 RAM 的內容值經由數值解碼器(Data Decoder)存入 Lz 所指的 LCD Latch。

指令	功能
LCB Lz,@HL	LCD latch [Lz] ← data decoder← [@HL]

<說明>

將索引暫存器@HL 所指 RAM 的內容值經由數值解碼器(Data Decoder)存入 Lz 所指的 LCD Latch。與 LCT 不同的是假如 Ry 的內容是 0，則數值解碼器(Data Decoder)輸出的值全為 0。

指令	功能
LCP Lz,@HL	LCD latch [Lz]← [@HL]&AC

<說明>

將索引暫存器@HL 所指 RAM 的內容值和累加器(Accumulator)的值存入 Lz 所指的 LCD Latch。

指令	功能
SPA X	定義 IOA 埠裡的每一支腳是輸入或輸出腳

<說明>

以直接數值(Direct Data)X(X4 X3 X2 X1 X0)來定義相對應的 IOA 腳是輸入或輸出腳，其對應表格如下：

X 數值	結果	X 數值	結果
X4=1	啓動 IOA1~4 下拉電阻(Pull low resistor)	X4=0	關掉 IOA1~4 下拉電阻(Pull low resistor)
X3=1	設 IOA4 是輸出模式	X3=0	設 IOA4 是輸入模式
X2=1	設 IOA3 是輸出模式	X2=0	設 IOA3 是輸入模式
X1=1	設 IOA2 是輸出模式	X1=0	設 IOA2 是輸入模式
X0=1	設 IOA1 是輸出模式	X0=0	設 IOA1 是輸入模式

指令	功能
OPA Rx	I/OA ← [Rx]

<說明>

將記憶體 Rx 的內容值輸出至 IOA 埠。

指令	功能
OPAS Rx,D	IOA1,2 ← [Rx], IOA3 ← D, IOA4 ← pulse

<說明>

將記憶體 Rx 的內容值輸出至 IOA 埠的 IOA1，IOA2，直接數值(Direct Data)輸出至 IOA3，脈波(Pulse)輸出至 IOA4。D 的值為 0 或 1。

指令	功能
IPA Rx	[Rx], AC ← [I/OA]

<說明>

將 IOA 埠讀入至記憶體 Rx 及累加器 AC 裡。

指令	功能
SPB X	定義 IOB 埠裡的每一支腳是輸入或輸出腳

<說明>

以直接數值(Direct Data)X(X4 X3 X2 X1 X0)來定義相對應的 IOB 腳是輸入或輸出腳，其對應表格如下：

X 數值	結果	X 數值	結果
X4=1	啓動 IOB1~4 下拉電阻(Pull low resistor)	X4=0	關掉 IOB1~4 下拉電阻(Pull low resistor)
X3=1	設 IOB4 是輸出模式	X3=0	設 IOB4 是輸入模式
X2=1	設 IOB3 是輸出模式	X2=0	設 IOB3 是輸入模式
X1=1	設 IOB2 是輸出模式	X1=0	設 IOB2 是輸入模式
X0=1	設 IOB1 是輸出模式	X0=0	設 IOB1 是輸入模式

指令	功能
OPB Rx	I/OB ← [Rx]

<說明>

將記憶體 Rx 的內容值輸出至 IOB 埠。

指令	功能
IPB Rx	[Rx], AC ← [I/OB]

<說明>

將 IOB 埠讀入至記憶體 Rx 及累加器 AC 裡。

指令	功能
SPC X	定義 IOC 埠裡的每一支腳是輸入或輸出腳

<說明>

以直接數值(Direct Data)X(X4 X3 X2 X1 X0)來定義相對應的 IOC 腳是輸入或輸出腳，其對應表格如下：

X 數值	結果	X 數值	結果
X4=1	啓動所有下拉(pull low)電阻及關掉 Low level hold 功能	X4=0	關掉所有下拉(pull low)電阻及啓動 Low level hold 功能
X3=1	設 IOC4 是輸出模式	X3=0	設 IOC4 是輸入模式
X2=1	設 IOC3 是輸出模式	X2=0	設 IOC3 是輸入模式
X1=1	設 IOC2 是輸出模式	X1=0	設 IOC2 是輸入模式
X0=1	設 IOC1 是輸出模式	X0=0	設 IOC1 是輸入模式

指令	功能
OPC Rx	I/OC ← [Rx]

<說明>

將記憶體 Rx 的內容值輸出至 IOC 埠。

指令	功能
IPC Rx	[Rx], AC ← [I/OC] or KI

<說明>

將 IOC 埠或 KI 的值讀入至記憶體 Rx 及累加器 AC 裡。

指令	功能
SPK X	設定掃描按鍵輸出的狀態

<說明>

當 SEG1~12 以光罩選擇(Mask Option)來選為 LCD 或 LED 用時，以 X(X5~0)來設定掃描按鍵輸出的狀態，詳細對應如下表：

X6	結果
1	不管有沒有按按鍵，只要在掃描周期過後都會產生 Release。
0	要按按鍵才會產生 Release。

X7,5,4	結果
000	設一次掃描 K1~16 中的一排按鍵。掃描那一排則用 X3~0 來選擇，例如 X3~0=0 表示掃描 K1 這一排，X3~0=15 則表示掃描 K16 這一排，其它依此類推。
001	K1~16=1 即同時掃描所有按鍵
010	K1~16=Hi-z 即不掃描
10X	設一次掃描 K1~16 中的 8 排按鍵。掃描那一組則用 X3 來選擇，例如 X3=0 表示掃描 K1~8 這一組，X3=1 則表示掃描 K9~16 這一組。
110	設一次掃描 K1~16 中的 4 排按鍵。掃描那一組則用 X3~2 來選擇： X3,2=00 -> 掃描 K1~4 X3,2=01 -> 掃描 K5~8 X3,2=10 -> 掃描 K9~12 X3,2=11 -> 掃描 K13~16
111	設一次掃描 K1~16 中的 2 排按鍵。掃描那一組則用 X3~1 來選擇： X3~1=000 -> 掃描 K1,2 X3~1=001 -> 掃描 K3,4 X3~1=010 -> 掃描 K5,6 X3~1=011 -> 掃描 K7,8 X3~1=100 -> 掃描 K9,10 X3~1=101 -> 掃描 K11,12 X3~1=110 -> 掃描 K13,14 X3~1=111 -> 掃描 K15,16

Notes: 1. 1=H/L (LED/LCD)

2. S1~12 = SEG1~12 輸出狀態

指令	功能
ALM X	設定蜂鳴器(Buzzer)的輸出頻率

<說明>

用直接數值(Direct Data)X(X8~X0)來設定蜂鳴器(Buzzer)的輸出頻率，X 值與對應的頻率如下：

X8	X7	X6	音頻高低的時鐘來源 (Clock source)
1	1	1	FREQ*
1	0	0	DC1
0	1	1	ϕ 3(4KHz)
0	1	0	ϕ 4(2KHz)
0	0	1	ϕ 5(1KHz)
0	0	0	DC0

Bit	音頻長短的時鐘來源 (Clock source)
X5	ϕ 15(1Hz)
X4	ϕ 14(2Hz)
X3	ϕ 13(4Hz)
X2	ϕ 12(8Hz)
X1	ϕ 11(16Hz)
X0	ϕ 10(32Hz)

- <注意>
1. FREQ 是頻率產生器的輸出信號
 2. 當蜂鳴器(Buzzer)輸出不需要封包(Envelop)時，X0~X5 需設為 0
 3. The frequency inside the () bases on the ϕ 0 is 32768Hz.

指令	功能
ELC X	設定冷光板(EL)的輸出頻率

<說明>

用直接數值(Direct Data)X(X8~X0)來設定冷光板(EL)的輸出頻率，X 值與對應的設定值如下：

ELP 腳的設定:

(X8,X7,X6)	昇壓的時鐘頻率	(X5,X4)	Duty cycle
000	ϕ 0	00	3/4 duty
100	BCLK	01	2/3 duty
101	BCLK/2	10	1/2 duty
110	BCLK/4	11	1/1 duty
111	BCLK/8		

ELC 腳的設定:

(X3,X2)	放電脈波的頻率	(X1,X0)	Duty cycle
00	$\phi 8$	00	1/4 duty
01	$\phi 7$	01	1/3 duty
10	$\phi 6$	10	1/2 duty
11	$\phi 5$	11	1/1 duty

4-2 累加器(Accumulator)及記憶體(RAM)的操作指令

指令	功能
MRW Ry,Rx	AC,[Ry] \leftarrow [Rx]

<說明>

將記憶體 Rx 的內容值搬到工作記憶體 Ry 及累加器(AC)裡

指令	功能
MRW @HL,Rx	AC,[@HL] \leftarrow [Rx]

<說明>

將記憶體 Rx 的內容值搬到索引暫存器(Index register)@HL 所指的記憶體及累加器(AC)裡

指令	功能
MWR Rx,Ry	AC,[Rx] \leftarrow [Ry]

<說明>

將記憶體 Ry 的內容值搬到工作記憶體 Rx 及累加器(AC)裡

指令	功能
MWR Rx,@HL	AC,[Rx] \leftarrow [@HL]

<說明>

將索引暫存器(Index register)@HL 所指的記憶體的內容值搬到記憶體 Rx 及累加器(AC)裡

指令	功能
SR0 Rx	[Rx]n, ACn \leftarrow [Rx](n+1),AC(n+1) [Rx]3, AC3 \leftarrow 0

<說明>

將記憶體 Rx 的內容向右移一個位元，並且把最高的位元填 0

指令	功能
SR1 Rx	$[Rx]n, ACn \leftarrow [Rx](n+1), AC(n+1)$ $[Rx]3, AC3 \leftarrow 1$

<說明>

將記憶體 **Rx** 的內容向右移一個位元，並且把最高的位元填 1

指令	功能
SL0 Rx	$[Rx]n, ACn \leftarrow [Rx](n-1), AC(n-1)$ $[Rx]0, AC0 \leftarrow 0$

<說明>

將記憶體 **Rx** 的內容向左移一個位元，並且把最低的位元填 0

指令	功能
SL1 Rx	$[Rx]n, ACn \leftarrow [Rx](n-1), AC(n-1)$ $[Rx]0, AC0 \leftarrow 1$

<說明>

將記憶體 **Rx** 的內容向左移一個位元，並且把最高的位元填 1

指令	功能
MRA Rx	$CF \leftarrow [Rx]3$

<說明>

將記憶體 **Rx** 內容的第三位元(Bit3)搬到溢位旗號(Carry flag, CF)

指令	功能
MAF Rx	$AC, [Rx] \leftarrow CF$

<說明>

將溢位旗號(Carry flag, CF)的內容值搬到記憶體 **Rx** 及累加器(Accumulator)，旗號的對應位元如下：

- Bit 3 CF
- Bit 2 AC=0 flag
- Bit 1 沒使用
- Bit 0 沒使用

4-3 運算指令(Operation Instruction)

指令	功能
INC* Rx	$[Rx], AC \leftarrow [Rx]+1$

<說明>

記憶體 Rx 的內容值加一後存入原來之記憶體 Rx 及累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
INC* @HL	$[@HL], AC \leftarrow [@HL]+1$

<說明>

索引暫存器@HL 所指記憶體的內容值加一後存入原來索引暫存器@HL 所指記憶體及累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
DEC* Rx	$[Rx], AC \leftarrow [Rx]-1$

<說明>

記憶體 Rx 的內容值減一後存入原來之記憶體 Rx 及累加器(AC)裡，運算結果會影響溢(借)位旗號(Carry flag, CF)。

指令	功能
DEC* @HL	$[@HL], AC \leftarrow [@HL]-1$

<說明>

索引暫存器@HL 所指記憶體的內容值減一後存入原來索引暫存器@HL 所指記憶體及累加器(AC)裡，運算結果會影響溢(借)位旗號(Carry flag, CF)。

指令	功能
ADC Rx	$AC \leftarrow [Rx]+AC+CF$

<說明>

二進位運算，記憶體 Rx 的內容值加累加器(AC)再加溢位旗號(Carry flag, CF)後存入累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
ADC @HL	$AC \leftarrow [@HL]+AC+CF$

<說明>

二進位運算，索引暫存器@HL 所指記憶體的內容值加累加器(AC)再加溢位旗號(Carry flag, CF)後存入累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
ADC* Rx	$AC, [Rx] \leftarrow [Rx] + AC + CF$

<說明>

二進位運算，記憶體 Rx 的內容值加累加器(AC)再加溢位旗號(Carry flag, CF)後存入原來之記憶體 Rx 及累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
ADC* @HL	$AC, [@HL] \leftarrow [@HL] + AC + CF$

<說明>

二進位運算，索引暫存器@HL 所指記憶體的內容值加累加器(AC)再加溢位旗號(Carry flag, CF)後存入原來索引暫存器@HL 所指記憶體及累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
SBC Rx	$AC \leftarrow [Rx] + \sim(AC) + CF$

<說明>

二進位運算，記憶體 Rx 的內容值減累加器(AC)再加溢位旗號(Carry flag, CF)後存入累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
SBC @HL	$AC \leftarrow [@HL] + \sim(AC) + CF$

<說明>

二進位運算，索引暫存器@HL 所指記憶體的內容值減累加器(AC)再加溢位旗號(Carry flag, CF)後存入累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
SBC* Rx	$AC, [Rx] \leftarrow [Rx] + \sim(AC) + CF$

<說明>

二進位運算，記憶體 Rx 的內容值減累加器(AC)再加溢位旗號(Carry flag, CF)後存入原來之記憶體 Rx 及累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
SBC* @HL	AC,[@HL] ← [@HL]+~(AC)+CF

<說明>

二進位運算，索引暫存器@HL 所指記憶體的內容值減累加器(AC)再加溢位旗號(Carry flag, CF)後存入索引暫存器@HL 所指記憶體及累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
ADD Rx	AC ← [Rx]+AC

<說明>

二進位運算，記憶體 Rx 的內容值加累加器(AC)後存入累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
ADD @HL	AC ← [@HL]+AC

<說明>

二進位運算，索引暫存器@HL 所指記憶體的內容值加累加器(AC)後存入累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
ADD* Rx	AC,[Rx] ← [Rx]+AC

<說明>

二進位運算，記憶體 Rx 的內容值加累加器(AC)後存入原來之記憶體 Rx 及累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
ADD* @HL	AC,[Rx] ← [@HL]+AC

<說明>

二進位運算，索引暫存器@HL 所指記憶體的內容值加累加器(AC)後存入原來索引暫存器@HL 所指記憶體及累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
SUB Rx	AC ← [Rx]+~(AC)+1

<說明>

二進位運算，記憶體 Rx 的內容值減累加器(AC)再加 1 後存入累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。
~表示補數。

指令	功能
SUB @HL	$AC \leftarrow [@HL] + \sim(AC) + 1$

<說明>

二進位運算，索引暫存器@HL 所指記憶體的內容值減累加器(AC)再加 1 後存入累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF) 。~表示補數。

指令	功能
SUB* Rx	$AC, [Rx] \leftarrow [Rx] + \sim(AC) + 1$

<說明>

二進位運算，記憶體 Rx 的內容值減累加器(AC)再加 1 後存入原來之記憶體 Rx 及累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF) 。~表示補數。

指令	功能
SUB* @HL	$AC, [@HL] \leftarrow [@HL] + \sim(AC) + 1$

<說明>

二進位運算，索引暫存器@HL 所指記憶體的內容值減累加器(AC)再加 1 後存入索引暫存器@HL 所指記憶體及累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF) 。~表示補數。

指令	功能
ADN Rx	$AC \leftarrow [Rx] + AC$

<說明>

二進位運算，記憶體 Rx 的內容值加累加器(AC)後存入累加器(AC)裡，運算結果不會影響溢位旗號(Carry flag, CF) 。

指令	功能
ADN @HL	$AC \leftarrow [@HL] + AC$

<說明>

二進位運算，索引暫存器@HL 所指記憶體的內容值加累加器(AC)後存入累加器(AC)裡，運算結果不會影響溢位旗號(Carry flag, CF) 。

指令	功能
ADN* Rx	$AC, [Rx] \leftarrow [Rx] + AC$

<說明>

二進位運算，記憶體 Rx 的內容值加累加器(AC)後存入原來之記憶體 Rx 及累加器(AC)裡，運算結果不會影響溢位旗號(Carry flag, CF) 。

指令	功能
ADN* @HL	$AC, [Rx] \leftarrow [@HL] + AC$

<說明>

二進位運算，索引暫存器@HL 所指記憶體的內容值加累加器(AC)後存入原來索引暫存器@HL 所指記憶體及累加器(AC)裡，運算結果不會影響溢位旗號(Carry flag, CF)。

指令	功能
AND Rx	$AC \leftarrow [Rx] \& AC$

<說明>

二進位運算，記憶體 Rx 的內容值與累加器(AC)作邏輯 AND 運算後存入累加器(AC)裡。

指令	功能
AND @HL	$AC \leftarrow [@HL] \& AC$

<說明>

二進位運算，索引暫存器@HL 所指記憶體的內容值與累加器(AC) 作邏輯 AND 運算後存入累加器(AC)裡。

指令	功能
AND* Rx	$AC, [Rx] \leftarrow [Rx] \& AC$

<說明>

二進位運算，記憶體 Rx 的內容值與累加器(AC) 作邏輯 AND 運算後存入原來之記憶體 Rx 及累加器(AC)裡。

指令	功能
AND* @HL	$AC, [Rx] \leftarrow [@HL] \& AC$

<說明>

二進位運算，索引暫存器@HL 所指記憶體的內容值與累加器(AC) 作邏輯 AND 運算後存入原來索引暫存器@HL 所指記憶體及累加器(AC)裡。

指令	功能
EOR Rx	$AC \leftarrow [Rx] \oplus AC$

<說明>

二進位運算，記憶體 Rx 的內容值與累加器(AC)作邏輯互斥或(Exclusive OR)運算後存入累加器(AC)裡。

指令	功能
EOR @HL	$AC \leftarrow [@HL] \oplus AC$

<說明>

二進位運算，索引暫存器@HL 所指記憶體的內容值與累加器(AC) 作邏輯互斥或(Exclusive OR)運算後存入累加器(AC)裡。

指令	功能
EOR* Rx	$AC, [Rx] \leftarrow [Rx] \oplus AC$

<說明>

二進位運算，記憶體 Rx 的內容值與累加器(AC) 作邏輯互斥或(Exclusive OR)運算後存入原來之記憶體 Rx 及累加器(AC)裡。

指令	功能
EOR* @HL	$AC, [Rx] \leftarrow [@HL] \oplus AC$

<說明>

二進位運算，索引暫存器@HL 所指記憶體的內容值與累加器(AC) 作邏輯互斥或(Exclusive OR)運算後存入原來索引暫存器@HL 所指記憶體及累加器(AC)裡。

指令	功能
OR Rx	$AC \leftarrow [Rx] AC$

<說明>

二進位運算，記憶體 Rx 的內容值與累加器(AC)作邏輯 OR 運算後存入累加器(AC)裡。

指令	功能
OR @HL	$AC \leftarrow [@HL] AC$

<說明>

二進位運算，索引暫存器@HL 所指記憶體的內容值與累加器(AC) 作邏輯 OR 運算後存入累加器(AC)裡。

指令	功能
OR* Rx	$AC, [Rx] \leftarrow [Rx] AC$

<說明>

二進位運算，記憶體 Rx 的內容值與累加器(AC) 作邏輯 OR 運算後存入原來之記憶體 Rx 及累加器(AC)裡。

指令	功能
OR* @HL	$AC, [Rx] \leftarrow [@HL] \mid AC$

<說明>

二進位運算，索引暫存器@HL 所指記憶體的內容值與累加器(AC) 作邏輯 OR 運算後存入原來索引暫存器@HL 所指記憶體及累加器(AC)裡。

指令	功能
ADCI Ry,D	$AC \leftarrow [Ry]+D+CF$

<說明>

二進位運算，工作記憶體 Ry 的內容值加直接數值(Direct data)D 再加溢位旗號(Carry flag, CF)後存入累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
ADCI* Ry,D	$AC, [Ry] \leftarrow [Ry]+D+CF$

<說明>

二進位運算，工作記憶體 Ry 的內容值加直接數值(Direct data)D 再加溢位旗號(Carry flag, CF)後存入原來之工作記憶體 Ry 及累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
SBCI Ry,D	$AC \leftarrow [Ry]+\sim(D)+CF$

<說明>

二進位運算，工作記憶體 Ry 的內容值減直接數值(Direct data)D 再加溢位旗號(Carry flag, CF)後存入累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。~表示補數。

指令	功能
SBCI* Ry,D	$AC, [Ry] \leftarrow [Ry]+\sim(D)+CF$

<說明>

二進位運算，工作記憶體 Ry 的內容值減直接數值(Direct data)D 再加溢位旗號(Carry flag, CF)後存入原來之工作記憶體 Ry 及累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。~表示補數。

指令	功能
ADDI Ry,D	$AC \leftarrow [Ry]+D$

<說明>

二進位運算，工作記憶體 Ry 的內容值加直接數值(Direct data)D 後存入累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
ADDI* Ry,D	$AC,[Ry] \leftarrow [Ry]+D$

<說明>

二進位運算，工作記憶體 Ry 的內容值加直接數值(Direct data)D 後存入原來之工作記憶體 Ry 及累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
SUBI Ry,D	$AC \leftarrow [Ry]+\sim(D)+1$

<說明>

二進位運算，工作記憶體 Ry 的內容值減直接數值(Direct data)D 再加 1 後存入累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。~表示補數。

指令	功能
SUBI* Ry,D	$AC,[Ry] \leftarrow [Ry]+\sim(D)+1$

<說明>

二進位運算，工作記憶體 Ry 的內容值減直接數值(Direct data)D 再加 1 後存入原來之工作記憶體 Ry 及累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。~表示補數。

指令	功能
ADNI Ry,D	$AC \leftarrow [Ry]+D$

<說明>

二進位運算，工作記憶體 Ry 的內容值加直接數值(Direct data)D 後存入累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
ADNI* Ry,D	$AC,[Ry] \leftarrow [Ry]+D$

<說明>

二進位運算，工作記憶體 Ry 的內容值加直接數值(Direct data)D 後存入原來之工作記憶體 Ry 及累加器(AC)裡，運算結果會影響溢位旗號(Carry flag, CF)。

指令	功能
ANDI Ry,D	$AC \leftarrow [Ry] \& D$

<說明>

二進位運算，工作記憶體 **Ry** 的內容值與直接數值(Direct data)**D** 作邏輯 AND 運算後存入累加器(AC)裡。

指令	功能
ANDI* Ry,D	$AC,[Ry] \leftarrow [Ry] \& D$

<說明>

二進位運算，工作記憶體 **Ry** 的內容值與直接數值(Direct data)**D** 作邏輯 AND 運算後存入原來之工作記憶體 **Ry** 及累加器(AC)裡。

指令	功能
EORI Ry,D	$AC \leftarrow [Ry] \oplus D$

<說明>

二進位運算，工作記憶體 **Ry** 的內容值與直接數值(Direct data)**D** 作邏輯互斥或 (Exclusive OR)運算後存入累加器(AC)裡。

指令	功能
EORI* Ry,D	$AC,[Ry] \leftarrow [Ry] \oplus D$

<說明>

二進位運算，工作記憶體 **Ry** 的內容值與直接數值(Direct data)**D** 作邏輯互斥或 (Exclusive OR)運算後存入原來工作記憶體 **Ry** 及累加器(AC)裡。

指令	功能
ORI Ry,D	$AC \leftarrow [Ry] \mid D$

<說明>

二進位運算，工作記憶體 **Ry** 的內容值與直接數值(Direct data)**D** 作邏輯 OR 運算後存入累加器(AC)裡。

指令	功能
ORI* Ry,D	$AC,[Ry] \leftarrow [Ry] \mid D$

<說明>

二進位運算，工作記憶體 **Ry** 的內容值與直接數值(Direct data)**D** 作邏輯 OR 運算後存入原來工作記憶體 **Ry** 及累加器(AC)裡。

4-4 載入(Load)/儲存(Store)指令

指令	功能
STA Rx	$[Rx] \leftarrow AC$

<說明>

將累加器(AC)的內容值儲存到記憶體 Rx 裡。

指令	功能
STA @HL	$[@HL] \leftarrow AC$

<說明>

將累加器(AC)的內容值儲存到索引暫存器@HL 所指的記憶體裡。

指令	功能
LDS Rx,D	$AC,[Rx] \leftarrow D$

<說明>

將直接數值(Direct data)D 載入記憶體 Rx 及累加器(AC)裡。

指令	功能
LDA Rx	$AC \leftarrow [Rx]$

<說明>

將記憶體 Rx 的內容值載入累加器(AC)裡。

指令	功能
LDA @HL	$AC \leftarrow [@HL]$

<說明>

將索引暫存器@HL 所指的記憶體的內容值載入累加器(AC)裡。

指令	功能
LDH Rx,@HL	$AC,[Rx] \leftarrow [@HL]$ 最高的 4 個位元

<說明>

將索引暫存器@HL 所指的記憶體的內容值中最高的 4 個位元(High nibble)載入記憶體 Rx 及累加器(AC)裡。

指令	功能
LDH* Rx,@HL	$AC,[Rx] \leftarrow [@HL]$ 最高的 4 個位元

<說明>

將索引暫存器@HL 所指的記憶體的內容值中最高的 4 個位元(High nibble)載入記憶體 Rx 及累加器(AC)裡。之後@HL 所指的位址自動加 1。

指令	功能
LDL Rx,@HL	AC,[Rx] ← [@HL]最低的 4 個位元

<說明>

將索引暫存器@HL 所指的記憶體的內容值中最低的 4 個位元(Low nibble)載入記憶體 Rx 及累加器(AC)裡。

指令	功能
LDL* Rx,@HL	AC,[Rx] ← [@HL]最低的 4 個位元

<說明>

將索引暫存器@HL 所指的記憶體的內容值中最低的 4 個位元(High nibble)載入記憶體 Rx 及累加器(AC)裡。之後@HL 所指的位址自動加 1。

4-5 CPU 控制指令(CPU Control Instructions)

指令	功能
NOP	無運算

<說明>

無任何運算。

指令	功能
HALT	CPU 進入中止狀態

<說明>

CPU 進入中止狀態，在雙時鐘工作模式下，只剩下低速時鐘在工作，能夠解除中止狀態(Halt release)有以下三種情況：

1. 中斷產生
2. IOC 埠產生信號變化
3. 符合 SHE 指令所設定的解除中止狀態的條件

指令	功能
STOP	CPU 進入停止狀態

<說明>

CPU 進入中止狀態，在雙時鐘工作模式下，兩個時鐘均停止不動，能夠解除停止狀態(STOP release)有以下三種情況：

1. KI1~4 中任何一支腳在掃描的周期裡產生信號變化
2. INT 腳有信號變化
3. IOC 埠的任何一支腳產生 Hi 的信號

指令	功能
SCA X	由直接數值 X 所設定的值可解除中止狀態(Halt release)

<說明>

當 X4=1 時表示 IOC 埠的信號改變可解除中止狀態(Halt release) 。X7~X5,X3~X0 保留沒用。

指令	功能
SIE* X	設定(Set)/重置(Reset)中斷致能旗號(Interrupt Enable Flag)

<說明>

X0=1	啓動 IOC 埠的信號改變會產生中斷
X1=1	啓動 TMR1 underflow 發生時會產生中斷
X2=1	啓動 INT 的信號改變會產生中斷
X3=1	啓動預除器(Pre-divider)溢位(overflow)發生時會產生中斷
X4=1	保留沒用
X5=1	啓動按鍵掃描期間按鍵被按時會產生中斷
X6=1	保留沒用

<註> X7 保留沒用

指令	功能
SHE X	設定(Set)/重置(Reset)中止解除旗號(Halt Release Flag)

<說明>

X1=1	啓動 TMR1 underflow 發生時可解除中止狀態
X2=1	啓動 INT 的信號改變可解除中止狀態
X3=1	啓動預除器(Pre-divider)溢位(overflow)發生時可解除中止狀態
X4=1	保留沒用
X5=1	啓動按鍵掃描期間按鍵被按時可解除中止狀態
X6=1	保留沒用

<註> X7 保留沒用

指令	功能
SRE X	設定(Set)/重置(Reset)停止解除旗號(Halt Release Flag)

<說明>

X4=1	啓動 IOC 埠的信號改變可解除停止狀態
X5=1	啓動 INT 的信號改變可解除停止狀態
X7=1	啓動按鍵掃描期間按鍵被按時可解除停止狀態

<註> X6, X3~X0 保留沒用

指令	功能
FAST	雙時鐘模式下，切換系統時鐘至高速時鐘模式

<說明>

以 TM8713 來說是將時鐘切換至高速的內建 RC 振盪器。

指令	功能
SLOW	雙時鐘模式下，切換系統時鐘至低速時鐘模式

<說明>

以 TM8713 來說是將時鐘切換至低速的外接 XIN/XOUT 振盪器。

指令	功能
MSB Rx	AC,[Rx] ← SCF1,SCF2,BCF

<說明>

將 SCF1，SCF2 及 BCF 旗號載入記憶體 Rx 及累加器(AC)以便判別中止解除的原因及 Backup 模式的狀態。相對應的位元意義如下：

Bit 3	Bit 2	Bit 1	Bit 0
No Use	Start condition flag 2 (SCF2)	Start condition flag 1 (SCF1)	Backup flag (BCF)
	中止解除是因 SCF4,5,7,8	中止解除是因 IOC port 信號變化	Backup 的狀態

指令	功能
MSC Rx	AC,[Rx] ← SCF4~SCF7

<說明>

將 SCF4 到 SCF7 旗號載入記憶體 Rx 及累加器(AC)以便判別中止解除的原因。相對應的位元意義如下：

Bit 3	Bit 2	Bit 1	Bit 0
Start condition flag 7 (SCF7)		Start condition flag 5 (SCF5)	Start condition flag 4 (SCF4)
中止解除是因預除器 (Pre-divider) overflow 而產生	第 15 階預除器的輸出內容	中止解除是因 TMR1 underflow 而產生	中止解除是因 INT 腳信號變化而產生

指令	功能
MCX Rx	$AC,[Rx] \leftarrow SCF8$

<說明>

將 SCF8 旗號載入記憶體 Rx 及累加器(AC)以便判別中止解除的原因。相對應的位元意義如下：

Bit 3	Bit 2	Bit 1	Bit 0
沒使用	沒使用	沒使用	Start condition flag 8 (SCF8)
沒使用	沒使用	沒使用	中止解除是因按鍵掃描期間按鍵被按而產生

指令	功能
MSD Rx	$AC,[Rx] \leftarrow WDF,CSF$

<說明>

將 WDF 及 CSF 旗號載入記憶體 Rx 及累加器(AC)。相對應的位元意義如下：

Bit 3	Bit 2	Bit 1	Bit 0
沒使用	沒使用	WDF	CSF
沒使用	沒使用	啓動看門狗時間旗號 (Watchdog timer enable flag)	選擇系統時鐘旗號 (System select flag)

4-6 索引位址指令(Index Address Instructions)

指令	功能
MVH Rx	$[@H] \leftarrow [Rx],AC$

<說明>

將記憶體 Rx 與累加器 AC 的內容合併後載入索引暫存器@HL 較高的 8 位元暫存器@H。

指令	功能
MVL Rx	$[@L] \leftarrow [Rx]$

<說明>

將記憶體 Rx 的內容載入索引暫存器@HL 較低的 4 位元暫存器@L。

4-7 十進制算術運算指令(Decimal Arithmetic Instructions)

指令	功能
DAA	$AC \leftarrow BCD[AC]$

<說明>

將累加器 AC 的內容值轉換成爲十進位並且重新存入累加器 AC 中，運算結果會影響溢位旗號(Carry flag, CF)。當這個指令被執行時，累加器 AC 必須是任何相加指令運算後的結果，如此才能配合溢位旗號(Carry flag, CF)作正確的轉換。

指令	功能
DAA* Rx	$AC, [Rx] \leftarrow BCD[AC]$

<說明>

將累加器 AC 的內容值轉換成爲十進位並且重新存入累加器 AC 和記憶體 Rx 中，運算結果會影響溢位旗號(Carry flag, CF)。當這個指令被執行時，累加器 AC 必須是任何相加指令運算後的結果，如此才能配合溢位旗號(Carry flag, CF)作正確的轉換。

指令	功能
DAA* @HL	$AC, [@HL] \leftarrow BCD[AC]$

<說明>

將累加器 AC 的內容值轉換成爲十進位並且重新存入累加器 AC 和索引暫存器 @HL 所指的記憶體中，運算結果會影響溢位旗號(Carry flag, CF)。當這個指令被執行時，累加器 AC 必須是任何相加指令運算後的結果，如此才能配合溢位旗號(Carry flag, CF)作正確的轉換。

指令	功能
DAS	$AC \leftarrow BCD[AC]$

<說明>

將累加器 AC 的內容值轉換成爲十進位並且重新存入累加器 AC 中，運算結果會影響溢位旗號(Carry flag, CF)。當這個指令被執行時，累加器 AC 必須是任何相減指令運算後的結果，如此才能配合溢位旗號(Carry flag, CF)作正確的轉換。

指令	功能
DAS* Rx	AC, [Rx] ← BCD[AC]

<說明>

將累加器 AC 的內容值轉換成爲十進位並且重新存入累加器 AC 和記憶體 Rx 中，運算結果會影響溢位旗號(Carry flag, CF)。當這個指令被執行時，累加器 AC 必須是任何相減指令運算後的結果，如此才能配合溢位旗號(Carry flag, CF)作正確的轉換。

指令	功能
DAS* @HL	AC, [@HL] ← BCD[AC]

<說明>

將累加器 AC 的內容值轉換成爲十進位並且重新存入累加器 AC 和索引暫存器 @HL 所指的記憶體中，運算結果會影響溢位旗號(Carry flag, CF)。當這個指令被執行時，累加器 AC 必須是任何相減指令運算後的結果，如此才能配合溢位旗號(Carry flag, CF)作正確的轉換。

4-8 跳躍指令(Jump Instructions)

指令	功能
JB0 X	當 AC bit0=1 時，程式會跳至 X

<說明>

當 AC bit0=1 時，程式會跳至 X，如果 bit0=0 時，程式會繼續往下執行。X 的範圍從 000h~5FFh。

指令	功能
JB1 X	當 AC bit1=1 時，程式會跳至 X

<說明>

當 AC bit1=1 時，程式會跳至 X，如果 bit1=0 時，程式會繼續往下執行。X 的範圍從 000h~5FFh。

指令	功能
JB2 X	當 AC bit2=1 時，程式會跳至 X

<說明>

當 AC bit2=1 時，程式會跳至 X，如果 bit2=0 時，程式會繼續往下執行。X 的範圍從 000h~5FFh。

指令	功能
JB3 X	當 AC bit3=1 時, 程式會跳至 X

<說明>

當 AC bit3=1 時, 程式會跳至 X, 如果 bit3=0 時, 程式會繼續往下執行。X 的範圍從 000h~5FFh。

指令	功能
JNZ X	當 AC !=0 時, 程式會跳至 X

<說明>

當 AC 不等於 0 時, 程式會跳至 X, 如果 AC=0 時, 程式會繼續往下執行。X 的範圍從 000h~5FFh。

指令	功能
JZ X	當 AC=0 時, 程式會跳至 X

<說明>

當 AC 等於 0 時, 程式會跳至 X, 如果 AC 不等於 0 時, 程式會繼續往下執行。X 的範圍從 000h~5FFh。

指令	功能
JNC X	當 CF=0 時, 程式會跳至 X

<說明>

當 CF 等於 0 時, 程式會跳至 X, 如果 CF=1 時, 程式會繼續往下執行。X 的範圍從 000h~5FFh。

指令	功能
JC X	當 CF=1 時, 程式會跳至 X

<說明>

當 CF 等於 1 時, 程式會跳至 X, 如果 CF=0 時, 程式會繼續往下執行。X 的範圍從 000h~5FFh。

指令	功能
JMP X	程式無條件跳至 X

<說明>

程式無條件跳至 X。X 的範圍從 000h~5FFh。

指令	功能
CALL X	STACK ← (PC)+1

<說明>

呼叫副程式(Subroutine) ，程式直接跳至 X 執行且會將原來之程式計數器(Program counter, PC)存入堆疊器(Stack)中。X 的範圍從 000h~5FFh。

指令	功能
RTS	PC ← (STACK)

<說明>

從副程式(Subroutine)中返回，程式計數器(Program counter, PC)從堆疊器(Stack)中回存。

4-9 其他的指令(Miscellaneous Instructions)

指令	功能
SCC X	設定頻率產生器(Frequency generator)及 IOC 消除抖動(Chattering cancel)的時鐘來源(Clock source)。

<說明>

相對應的位元定義如下：

位元	時鐘來源(Clock source)	位元	時鐘來源(Clock source)
X6=1	設頻率產生器的時鐘來源(Clock source)為系統時鐘 BCLK	X6=0	設頻率產生器的時鐘來源(Clock source)為系統時鐘為 φ0
(X2,X1,X0)=001	設消除抖動(Chattering cancel)的時鐘為 φ10	(X2,X1,X0)=010	設消除抖動(Chattering cancel)的時鐘為 φ8
(X2,X1,X0)=100	設消除抖動(Chattering cancel)的時鐘為 φ6		

<註>X7,X5,X4,X3 沒使用

指令	功能
FRQ D,Rx	頻率產生器(Frequency generator) ← D, [Rx], AC

<說明>

將累加器 AC 和記憶體 Rx 的內容值合併成 8 位元的數值來設頻率產生器(Frequency generator) 的起始值，D 則是設定頻率產生器的 Duty 值。

指令	Bit7	Bit6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FRQ D,Rx	AC3	AC2	AC1	AC0	Rx3	Rx2	Rx1	Rx0

D 數值		Duty 周期
D1	D0	
0	0	1/4 duty
0	1	1/3 duty
1	0	1/2 duty
1	1	1/1 duty

指令	功能
FRQ D,@HL	頻率產生器(Frequency generator) ← D, [@HL]

<說明>

將索引暫存器@HL 所指的表格 ROM(Table ROM)的 8 位元數值來設頻率產生器(Frequency generator) 的起始值，D 則是設定頻率產生器的 Duty 值。

指令	Bit7	Bit6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FRQ D,@HL	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0

<註> TD0 ~ TD7 表示索引暫存器@HL 所指的表格 ROM(Table ROM)的 8 位元內容值

D 數值		Duty 周期
D1	D0	
0	0	1/4 duty
0	1	1/3 duty
1	0	1/2 duty
1	1	1/1 duty

指令	功能
FRQX D, X	頻率產生器(Frequency generator) ← D, X

<說明>

將 8 位元的直接數值(Direct data)X 來設頻率產生器(Frequency generator) 的起始值，D 則是設定頻率產生器的 Duty 值。

指令	Bit7	Bit6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FRQX D, X	X7	X6	X5	X4	X3	X2	X1	X0

<註> X0 ~ X7 表示直接數值(Direct data)

D 數值		Duty 周期
D1	D0	
0	0	1/4 duty
0	1	1/3 duty
1	0	1/2 duty
1	1	1/1 duty

指令	功能
TMS Rx	選擇 TMR1 時鐘來源(Clock source)及預設 TMR1 數值

<說明>

將累加器 AC 和記憶體 Rx 的內容值合併成 8 位元的數值來設定 TMR1 時鐘來源 (Clock source)及預設 TMR1 數值。

		時鐘來源		TMR1 數值					
TMS	Rx	AC3	AC2	AC1	AC0	Rx3	Rx2	Rx1	Rx0

時鐘來源選擇

AC3	AC2	時鐘來源
0	0	$\phi 9$
0	1	$\phi 3$
1	0	$\phi 15$
1	1	FREQ

指令	功能
TMS @HL	選擇 TMR1 時鐘來源(Clock source)及預設 TMR1 數值

<說明>

將索引暫存器@HL 所指的表格 ROM(Table ROM)的 8 位元的數值來設定 TMR1 時鐘來源(Clock source)及預設 TMR1 數值。

		時鐘來源		TMR1 數值					
TMS	@HL	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0

時鐘來源選擇

TD7	TD6	時鐘來源
0	0	$\phi 9$
0	1	$\phi 3$
1	0	$\phi 15$
1	1	FREQ

指令	功能
TMSX X	選擇 TMR1 時鐘來源(Clock source)及預設 TMR1 數值

<說明>

用 8 位元的直接數值(Direct data)X 來設定 TMR1 時鐘來源(Clock source)及預設 TMR1 數值。

		時鐘來源		TMR1 數值					
TMSX	X	X7	X6	X5	X4	X3	X2	X1	X0

時鐘來源選擇

X7	X6	時鐘來源
0	0	$\phi 9$
0	1	$\phi 3$
1	0	$\phi 15$
1	1	FREQ

指令	功能
SF X	設旗號

<說明>

用 8 位元的直接數值(Direct data)X 來設定旗號。

- X0 : "1" 設溢位旗號(Carry flag, CF)為 1
- X1 : "1" 設 Back up flag(BCF)為 1 且進入 Back up 模式
- X2 : "1" 啓動冷光(EL light)驅動器輸出
- X3 : "1" 當 X2=1 且 X3 設為 1 時會啓動冷光(EL light)驅動器且解除中止的需求(Halt release request)會輸出，然後進入中止狀態(Halt mode)
- X4 : "1" 起始(Initial)並啓動(Enable)看門狗時鐘(Watch dog timer).
- X6,5 保留沒使用
- X7 : "1" 啓動 TMR1 重新載入(Re-load)功能

指令	功能
RF X	清除旗號

<說明>

用 8 位元的直接數值(Direct data)X 來清除旗號。

- X0 : "1" 設溢位旗號(Carry flag, CF)為 0
- X1 : "1" 設 Back up flag(BCF)為 0 並且跳出 Back up 模式
- X2 : "1" 關閉冷光(EL light)驅動器輸出
- X3 : 保留沒使用
- X4 : "1" 關掉看門狗時鐘(Watch dog timer).
- X6,5 保留沒使用
- X7 : "1" 關掉 TMR1 重新載入(Re-load)功能

指令	功能
SF2 X	設旗號

<說明>

用 8 位元的直接數值(Direct data)X 來設定旗號。

X2 : "1" 關掉 LCD/LED segment 輸出

X3 : "1" 啓動中斷腳(INT)下拉電阻(Pull low resistor)

X7~4,X1~0 保留沒使用

指令	功能
RF2 X	設旗號

<說明>

用 8 位元的直接數值(Direct data)X 來設定旗號。

X2 : "1" 啓動 LCD/LED segment 輸出

X3 : "1" 關掉中斷腳(INT)下拉電阻(Pull low resistor)

X7~4,X1~0 保留沒使用

指令	功能
PLC X	脈波控制(Pulse Control)

<說明>

用 9 位元的直接數值(Direct data)X 來控制。

X0 : "1" 清除中止解除需求旗號 0(HRF0)--IOC port 信號變化

X1 : "1" 清除中止解除需求旗號 1(HRF1)且停止 TMR1--TMR1
underflow

X2 : "1" 清除中止解除需求旗號 2(HRF2)—INT 腳信號變化

X3 : "1" 清除中止解除需求旗號 3(HRF3)—預除器 overflow

X5: "1" 清除中止解除需求旗號 5(HRF5)—K1~4 被按

X8: "1" 清除預除器(Pre-divider)的最後第 5 位元，當執行此指令
時 X3 必須爲 1

X7~6,X4 保留沒使用

附錄 TM8713 指令總表

Instruction	Machine Code	Function	Flag/Remark
NOP	0000 0000 0000 0000	No Operation	
LCT	Lz,Ry 0000 001Z ZZZZ YYYY	Lz ← (7SEG ← Ry)	
LCB	Lz,Ry 0000 010Z ZZZZ YYYY	Lz ← (7SEG ← Ry)	Blank Zero
LCP	Lz,Ry 0000 011Z ZZZZ YYYY	Lz ← Ry & AC	
LCD	Lz,@HL 0000 100Z ZZZZ 0000	Lz ← T@HL	
LCT	Lz,@HL 0000 100Z ZZZZ 0001	Lz ← (7SEG ← @HL)	
LCB	Lz,@HL 0000 100Z ZZZZ 0010	Lz ← (7SEG ← @HL)	Blank Zero
LCP	Lz,@HL 0000 100Z ZZZZ 0011	Lz ← @HL & AC	
OPA	Rx 0000 1010 0XXX XXXX	Port(A) ← Rx	
OPAS	Rx,D 0000 1011 DXXX XXXX	A1,2,3,4 ← Rx0,Rx1,D,Pulse	
OPB	Rx 0000 1100 0XXX XXXX	Port(B) ← Rx	
OPC	Rx 0000 1101 0XXX XXXX	Port(C) ← Rx	
FRQ	D,Rx 0001 00DD 0XXX XXXX	FREQ D=00 : 1/4 Duty D=01 : 1/3 Duty D=10 : 1/2 Duty D=11 : 1/1 Duty	
FRQ	D,@HL 0001 01DD 0000 0000	FREQ ← T@HL	
FRQX	D,X 0001 10DD XXXX XXXX	FREQ ← X	
MVL	Rx 0001 1100 0XXX XXXX	@L ← Rx	
MVH	Rx 0001 1101 0XXX XXXX	@H ← Rx & AC	
ADC	Rx 0010 0000 0XXX XXXX	AC ← Rx + AC + CF	CF
ADC	@HL 0010 0000 1000 0000	AC ← @HL + AC + CF	CF
ADC*	Rx 0010 0001 0XXX XXXX	AC,Rx ← Rx + AC + CF	CF
ADC*	@HL 0010 0001 1000 0000	AC,@HL ← @HL + AC + CF	CF
SBC	Rx 0010 0010 0XXX XXXX	AC ← Rx + ACB + CF	CF
SBC	@HL 0010 0010 1000 0000	AC ← @HL + ACB + CF	CF
SBC*	Rx 0010 0011 0XXX XXXX	AC,Rx ← Rx + ACB + CF	CF
SBC*	@HL 0010 0011 1000 0000	AC,@HL ← @HL + ACB + CF	CF
ADD	Rx 0010 0100 0XXX XXXX	AC ← Rx + AC	CF
ADD	@HL 0010 0100 1000 0000	AC ← @HL + AC	CF
ADD*	Rx 0010 0101 0XXX XXXX	AC,Rx ← Rx + AC	CF
ADD*	@HL 0010 0101 1000 0000	AC,@HL ← @HL + AC	CF
SUB	Rx 0010 0110 0XXX XXXX	AC ← Rx + ACB + 1	CF
SUB	@HL 0010 0110 1000 0000	AC ← @HL + ACB + 1	CF
SUB*	Rx 0010 0111 0XXX XXXX	AC,Rx ← Rx + ACB + 1	CF
SUB*	@HL 0010 0111 1000 0000	AC,@HL ← @HL + ACB + 1	CF
ADN	Rx 0010 1000 0XXX XXXX	AC ← Rx + AC	
ADN	@HL 0010 1000 1000 0000	AC ← @HL + AC	
ADN*	Rx 0010 1001 0XXX XXXX	AC,Rx ← Rx + AC	

ADN*	@HL	0010 1001 1000 0000	AC,@HL	← @HL + AC	
AND	Rx	0010 1010 0XXX XXXX	AC	← Rx AND AC	
AND	@HL	0010 1010 1000 0000	AC	← @HL AND AC	
AND*	Rx	0010 1011 0XXX XXXX	AC,Rx	← Rx AND AC	
AND*	@HL	0010 1011 1000 0000	AC,@HL	← @HL AND AC	
EOR	Rx	0010 1100 0XXX XXXX	AC	← Rx EOR AC	
EOR	@HL	0010 1100 1000 0000	AC	← @HL EOR AC	
EOR*	Rx	0010 1101 0XXX XXXX	AC,Rx	← Rx EOR AC	
EOR*	@HL	0010 1101 1000 0000	AC,@HL	← @HL EOR AC	
OR	Rx	0010 1110 0XXX XXXX	AC	← Rx OR AC	
OR	@HL	0010 1110 1000 0000	AC	← @HL OR AC	
OR*	Rx	0010 1111 0XXX XXXX	AC,Rx	← Rx OR AC	
OR*	@HL	0010 1111 1000 0000	AC,@HL	← @HL OR AC	
ADCI	Ry,D	0011 0000 DDDD YYYY	AC	← Ry + D + CF	CF
ADCI*	Ry,D	0011 0001 DDDD YYYY	AC,Ry	← Ry + D + CF	CF
SBCI	Ry,D	0011 0010 DDDD YYYY	AC	← Ry + DB + CF	CF
SBCI*	Ry,D	0011 0011 DDDD YYYY	AC,Ry	← Ry + DB + CF	CF
ADDI	Ry,D	0011 0100 DDDD YYYY	AC	← Ry + D	CF
ADDI*	Ry,D	0011 0101 DDDD YYYY	AC,Ry	← Ry + D	CF
SUBI	Ry,D	0011 0110 DDDD YYYY	AC	← Ry + DB + 1	CF
SUBI*	Ry,D	0011 0111 DDDD YYYY	AC,Ry	← Ry + DB + 1	CF
ADNI	Ry,D	0011 1000 DDDD YYYY	AC	← Ry + D	
ADNI*	Ry,D	0011 1001 DDDD YYYY	AC,Ry	← Ry + D	
ANDI	Ry,D	0011 1010 DDDD YYYY	AC	← Ry AND D	
ANDI*	Ry,D	0011 1011 DDDD YYYY	AC,Ry	← Ry AND D	
EORI	Ry,D	0011 1100 DDDD YYYY	AC	← Ry EOR D	
EORI*	Ry,D	0011 1101 DDDD YYYY	AC,Ry	← Ry EOR D	
ORI	Ry,D	0011 1110 DDDD YYYY	AC	← Ry OR D	
ORI*	Ry,D	0011 1111 DDDD YYYY	AC,Ry	← Ry OR D	
INC*	Rx	0100 0000 0XXX XXXX	AC,Rx	← Rx + 1	
INC*	@HL	0100 0000 1000 0000	AC,@HL	← @HL + 1	
DEC*	Rx	0100 0001 0XXX XXXX	AC,Rx	← Rx - 1	
DEC*	@HL	0100 0001 1000 0000	AC,@HL	← @HL - 1	
IPA	Rx	0100 0010 0XXX XXXX	AC,Rx	← Port(A)	
IPB	Rx	0100 0100 0XXX XXXX	AC,Rx	← Port(B)	
IPC	Rx	0100 0111 0XXX XXXX	AC,Rx	← Port(C)	
MAF	Rx	0100 1010 0XXX XXXX	AC,Rx	← STS1	B3 : CF B2 : ZERO B1 : (No use) B0 : (No use)
MSB	Rx	0100 1011 0XXX XXXX	AC,Rx	← STS2	B3 : (No use) B2 : SCF2(HRx) B1 : SCF1(CPT) B0 : BCF

MSC	Rx	0100 1100 0XXX XXXX	AC,Rx	← STS3	B3 : SCF7(PDV) B2 : PH15 B1 : SCF5(TM1) B0 : SCF4(INT)
MCX	Rx	0100 1101 0XXX XXXX	AC,Rx	← STS3X	B3 : (No use) B2 : (No use) B1 : (No use) B0 : SCF8(SKI)
MSD	Rx	0100 1110 0XXX XXXX	AC,Rx	← STS4	B3 : (No use) B2 : (No use) B1 : WDF B0 : CSF
SR0	Rx	0101 0000 0XXX XXXX	ACn, Rxn AC3, Rx3	← Rx(n+1) ← 0	
SR1	Rx	0101 0001 0XXX XXXX	ACn, Rxn AC3, Rx3	← Rx(n+1) ← 1	
SL0	Rx	0101 0010 0XXX XXXX	ACn, Rxn AC0, Rx0	← Rx(n-1) ← 0	
SL1	Rx	0101 0011 0XXX XXXX	ACn, Rxn AC0, Rx0	← Rx(n-1) ← 1	
DAA		0101 0100 0000 0000	AC	← BCD(AC)	
DAA*	Rx	0101 0101 0XXX XXXX	AC,Rx	← BCD(AC)	
DAA*	@HL	0101 0101 1000 0000	AC,@HL	← BCD(AC)	
DAS		0101 0110 0000 0000	AC	← BCD(AC)	
DAS*	Rx	0101 0111 0XXX XXXX	AC,Rx	← BCD(AC)	
DAS*	@HL	0101 0111 1000 0000	AC,@HL	← BCD(AC)	
LDS	Rx,D	0101 1DDD DXXX XXXX	AC,Rx	← D	
LDH	Rx,@HL	0110 0000 0XXX XXXX	AC,Rx	← H(T@HL)	
LDH*	Rx,@HL	0110 0001 0XXX XXXX	AC,Rx HL	← H(T@HL) ← HL + 1	
LDL	Rx,@HL	0110 0010 0XXX XXXX	AC,Rx	← L(T@HL)	
LDL*	Rx,@HL	0110 0011 0XXX XXXX	AC,Rx HL	← L(T@HL) ← HL + 1	
STA	Rx	0110 1000 0XXX XXXX	Rx	← AC	
STA	@HL	0110 1000 1000 0000	@HL	← AC	
LDA	Rx	0110 1100 0XXX XXXX	AC	← Rx	
LDA	@HL	0100 1100 1000 0000	AC	← @HL	
MRA	Rx	0110 1101 0XXX XXXX	CF	← Rx3	
MRW	@HL,Rx	0110 1110 0XXX XXXX	AC,@HL	← Rx	
MWR	Rx,@HL	0110 1111 0XXX XXXX	AC,Rx	← @HL	
MRW	Ry,Rx	0111 0YYY YXXX XXXX	AC,Ry	← Rx	
MWR	Rx,Ry	0111 1YYY YXXX XXXX	AC,Rx	← Ry	
JB0	X	1000 0XXX XXXX XXXX	PC	← X	if AC0 = 1
JB1	X	1000 1XXX XXXX XXXX	PC	← X	if AC1 = 1
JB2	X	1000 0XXX XXXX XXXX	PC	← X	if AC2 = 1

JB3	X	1000 1XXX XXXX XXXX	PC	← X	if AC3 = 1
JNZ	X	1010 0XXX XXXX XXXX	PC	← X	if AC ≠ 0
JNC	X	1010 1XXX XXXX XXXX	PC	← X	if CF = 0
JZ	X	1011 0XXX XXXX XXXX	PC	← X	if AC = 0
JC	X	1011 1XXX XXXX XXXX	PC	← X	if CF = 1
CALL	X	1100 0XXX XXXX XXXX	STACK PC	← PC + 1 ← X	
JMP	X	1101 0XXX XXXX XXXX	PC	← X	
RTS		1101 1000 0000 0000	PC	← STACK	CALL Return
SCC	X	1101 1001 0X00 0XXX	X6 = 1 X6 = 0 X2,1,0=001 X2,1,0=010 X2,1,0=100	: Cfq = BCLK : Cfq = PH0 : Cch = PH10 : Cch = PH8 : Cch = PH6	
SCA	X	1101 1010 000X 0000	X4	: Enable SEF4	C1-4
SPA	X	1101 1100 000X XXXX	X4 X3~0	: Set A4-1 Pull-Low : Set A4-1 I/O	
SPB	X	1101 1101 000X XXXX	X4 X3~0	: Set B4-1 Pull-Low : Set B4-1 I/O	
SPC	X	1101 1110 000X XXXX	X4 X3-0	: Set C4-1 Pull-Low / Low-Level-Hold : Set C4-1 I/O	
TMS	Rx	1110 0000 0XXX XXXX	Timer1	← Rx & AC	
TMS	@HL	1110 0001 0000 0000	Timer1	← T@HL	
TMSX	X	1110 0010 XXXX XXXX	X7,6 = 11 X7,6 = 10 X7,6 = 01 X7,6 = 00 X5~0	: Ctm = FREQ : Ctm = PH15 : Ctm = PH3 : Ctm = PH9 : Set Timer1 Value	
SPK	X	1110 0011 XXXX XXXX	X6=1 X6=0 X7,5,4=000 X7,5,4=001 X7,5,4=010 X7,5,4=10X X7,5,4=110 X7,5,4=111	: KEY_S release by scanning cycle : KEY_S release by normal key scanning : Set one of KO1~12 =1 by X3~0 : Set all = 1 : Set all Hi-z : Set eight of KO1~12 =1 by X3 X3=0 => KO1~8 X3=1 => KO9~12 : Set four of KO1~12 =1 by X3,2 X3,2=00 => KO1~4 X3,2=01 => KO5~8 X3,2=10 => KO9~12 : Set two of KO1~12 =1 by X3,2,1	IOC=normal IOC=KEY SCAN IOC=KEY SCAN

				X3~1=000=>KO1,2 X3~1=001=>KO3,4 X3~1=010=>KO5,6 X3~1=011=>KO7,8 X3~1=100=>KO9,10 X3~1=101=>KO11,12	
SHE	X	1110 1000 00X0 XXX0	X5 X3 X2 X1	: Enable HEF5 : Enable HEF3 : Enable HEF2 : Enable HEF1	KEY_S PDV INT TMR1
SIE*	X	1110 1001 00X0 XXXX	X5 X3 X2 X1 X0	: Enable IEF5 : Enable IEF3 : Enable IEF2 : Enable IEF1 : Enable IEF0	KEY_S PDV INT TMR1 CPT
PLC	X	1110 101X 00X0 XXXX	X8 X5,3-0	: Reset PH15~11 : Reset HRF5,3-0	
SRE	X	1110 1101 X0XX 0000	X7 X5 X4	: Enable SRF7 : Enable SRF5 : Enable SRF4	SRF7(KEY_S) SRF5 (INT) SRF4 (C Port)
FAST		1110 1110 0000 0000	SCLK	: High Speed Clock	
SLOW		1110 1111 0000 0000	SCLK	: Low Speed Clock	
SF	X	1111 0000 X00X XXXX	X7 X4 X3 X2 X1 X0	: Reload 1 Set : WDT Enable : HALT after EL : EL LIGHT On : BCF Set : CF Set	RL1 WDF BCF CF
RF	X	1111 0100 X00X 0XXX	X7 X4 X2 X1 X0	: Reload 1 Reset : WDT Reset : EL LIGHT Off : BCF Reset : CF Reset	RL1 WDF BCF CF
SF2	X	1111 1000 0000 XX00	X3 X2	: Enable INT powerful Pull-low : Close all Segments	INTPL RSOFF
RF2	X	1111 1001 0000 XX00	X3 X2	: Disable INT powerful Pull-low : Release Segments	INTPL RSOFF
ALM	X	1111 101X XXXX XXXX	X8,7,6=111 X8,7,6=100 X8,7,6=011 X8,7,6=010 X8,7,6=001 X8,7,6=000 X5~0	: FREQ : DC1 : PH3 : PH4 : PH5 : DC0 ← PH15~10	
ELC	X	1111 110X XXXX XXXX	X8=1 X8=0 X7,6=11	BCLKX PH0 BCLK/8	ELP - CLK BCLKX

