4BIT 系列 MCU

功能說明書

AMENDMENT HISTORY

Version	Date	Description
1.0	2021/03	新頒。

CONTENTS

第一章 4BIT Series Internal System Architecture

1-1. 電源供應

4BIT 系列 MCU 以 Mask Option & SWPWR 方式提供 1.5V, 3V & 5V 三種電源模式的操作電壓範圍，並提供 No Bias , 1/2~1/5 Bias 五種選項以供應 LCD driver 所需的不同驅動電壓。

1-1-1. 系統電源

VBAT 為電源正極的輸入腳位(若該 Body 無獨立 VBAT 腳位,則視應用接 VDD1,2 or 最高 LCD 電位腳位)，BAK 腳位提供時鐘震盪線路(包括外部腳位)以及 MCU 內部電路所需的操作電壓。

在選擇使用 3V (BCF=0：BAK=VL1/2/VDL/VREG)的電源模式下，BAK 腳位上的電壓位準會隨著電力備援模式的切換而有所不同，因此必須需外接接一個 0.1 uF 以上電容以穩定操作電壓。除此之外，選用其他的電源模式時皆需將 BAK 腳位直接與 VBAT 腳位連接在一起使用。

4BIT 系列 MCU 還提供 LCD driver 電壓可由內外(若有獨立 VBAT 腳位)部穩壓線路提供的選項，利用 MCU 的內外部線路產生一個 regulated voltage 直接提供給 VL1 或是 VL2 腳位使用。在使用這種選項時 只有 LCD driver 腳位的輸出電壓是由內外部穩壓線路所提供，其他的 IO 腳位仍在 VBAT 與 GND 之間的電位操作。

選用上述各種 Mask Option 時，其對應的 BAK，VL1，VL2 的接法如下：

Mask Option 1	Mask Option 2	Mask Option3	BAK 的接法	VL1 的接法	VL2 的接法
1.5V	BAK 無穩壓	-	VBAT	-	-
	BAK 內部穩壓	-	電容	-	-
	VL1 無穩壓	No Bias	-	VBAT	VBAT*
		1/2~1/5Bias	-	VBAT	電容
	VL1 內部穩壓	1/2~1/5Bias	-	電容	電容
	VL1 外部穩壓	1/2~1/5Bias	-	外部穩壓	電容
3/5V (BCF=0 : BAK=VL1/..)	VL 無穩壓(BAK 內部穩壓)	No Bias	電容	VBAT	VBAT
	VL 無穩壓	VBAT=VL3~5 (1/3~5Bias)	電容	電容	電容
		VBAT=VL2	電容	電容	VBAT
	VL1 內部穩壓	1/2~1/5Bias	電容	電容	電容
	VL2 內部穩壓	1/3~1/5Bias	電容	電容	電容
	VL1 外部穩壓	1/2~1/5Bias	電容	外部穩壓	電容
	VL2 外部穩壓	1/2~1/5Bias	電容	電容	外部穩壓
3/5V (BCF=0 : BAK=VBAT)	VL 無穩壓	No Bias	VBAT	VBAT	VBAT
		VBAT=VL3~5 (1/3~5Bias)	VBAT	電容	電容
		1/2~1/5Bias	VBAT	電容	VBAT
	VL1 內部穩壓	1/2~1/5Bias	VBAT	電容	電容
	VL2 內部穩壓	1/3~1/5Bias	VBAT	電容	電容
	VL1 外部穩壓	1/2~1/5Bias	VBAT	外部穩壓	電容
	VL2 外部穩壓	1/2~1/5Bias	VBAT	電容	外部穩壓

Notes:

1. 連接電容時請選用 0.1 uF 或是以上的電容值，並將電容的另一端接 GND。
2. 當選用 3/5V (BCF=0 : BAK=VL1)的 code option 而且又選擇由外部穩壓線路提供 LCD driver 輸出電壓的 code option 時，無論外部穩壓電源是提供給 VL1 或 VL2 使用，必須確保 VL1 的電壓值不可低於 BAK 的震盪&工作電壓，以免造成時鐘震盪線路以及 MCU 內部電路的不正常動作。
3. code option 1 : power source 的 code option

4. code option 2 : 穩壓 for BAK&LCD 的 code option
5. code option 3 : LCD Bias 的 code option
6. '*' : 若 OTP 1.5V 電源模式下需連接 VL2 高電位至 VPP 腳位, 因即使 option 選則擇"No Bias" 仍會產生 charge pump 提供 VL2 高電位, 故此時 OTP VL2 腳位仍須接電容。

1-1-2. LCD driver 的電源

4BIT 系列 MCU 以 Charge Pump 方式產生 LCD driver 所需的各個電壓位準(VDD1/VL1 ~ VDD5/VL5)，所產生的最高電壓則依 LCD Bias 的 code option 而定，以 1/3，1/4，1/5 Bias 而言，最高電壓分別為 VL3，VL4，VL5。
4BIT 系列 MCU 最多可提供使用者三種 Mask option 選擇 Charge Pump 的頻率，分別為 PH3，PH4，PH5 以及 PH6。使用者可依 LCD panel 的實際負載大小，LCD frame frequency(詳細說明請參閱 3-1)以及 LCD Bias 等實際情況調整 Charge Pump 頻率，原則上 Charge Pump 頻率最好設定成 LCD frame frequency 的兩倍以上才能獲得較佳的倍壓效率。但實際的需求仍須依據實際測試的顯示效果為準，以便在 MCU 耗電量與 LCD panel 的 負載之間取得適當的平衡。
如果 LCD panel 負載較大時可選用較快的 charge pump 頻率，使 LCD driver 可以輸出較為穩定的電壓；而若 LCD panel 負載較小時則可選擇 PH6 以降低 MCU 的耗電。
由於 Charge Pump 的架構在驅動大面積的 LCD panel 時，會因為 LCD panel 本身的耗電 較高而造成 LCD driver 所輸出的最高電壓不易達到穩定的電壓值，可以利用下面的幾種方法改善這個問題：

1. 在 charge pump 線路可將 VL1 ~ VL5 的電容充電至預期電壓值的條件下，增大 CUP 腳位之間所連接電容的電容值以及 VL1 ~ VL5 接地電容的電容值。
2. 在 charge pump 線路可將 VL1 ~ VL5 的電容充電至預期電壓值的條件下，選擇較快的 Charge Pump 頻率。
3. 有一些如 TM89 系列 MCU 針對 1/3 bias，1/4 Bias 的應用也提供了與 1/5 Bias 一樣在 CUPN、CUP0 之間以及 CUP1、CUP2 之間各接一個電容的 Mask Option，以有效提昇 Charge Pump 的效率。

由於製程的因素，當 LCD driver 的輸出電壓高於 6V 時，整體 MCU 的逆偏電流會明顯增加，會增加 MCU 無謂的耗電。故建議當 LCD driver 的最高輸出電壓高於 6V 時，請改用外部穩壓線路來提供較低的電壓給 LCD driver 使用，以便讓 LCD driver 的最高輸出電壓 維持低於 6V 的情況。

1-1-2-1. 利用 VL1 的電位作為 Charge Pump 線路的基礎電壓

當選用 1.5V 電源模式或是選用 3V 電源模式而且 VL1 電壓由內/外部穩壓線路提供時，Charge Pump 線路的基礎電壓就會由 VL1 供應。
此時 charge pump 線路所產生的 VL2 ~ VL5 電壓位準以 1/5Bias 為例如下：

VL2 = 2 * VL1

VL3 = 3 * VL1

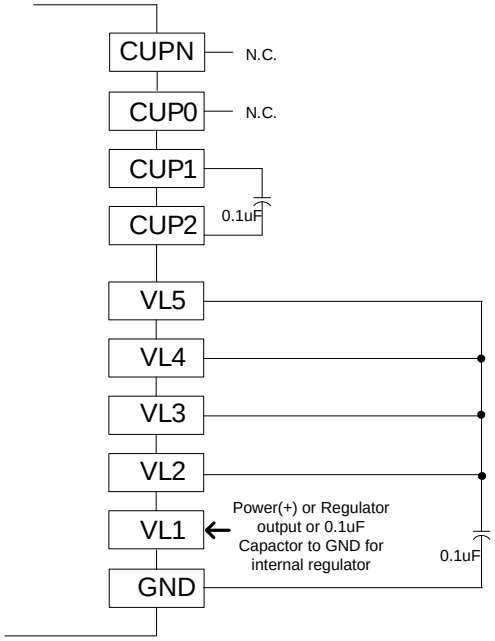
VL4 = 4 * VL1

VL5 = 5 * VL1

後面將說明利用 VL1 的電位作為 charge pump 線路的基礎電壓時，MCU 在不同的 LCD bias 的選項下所使用的系統電源應用線路。

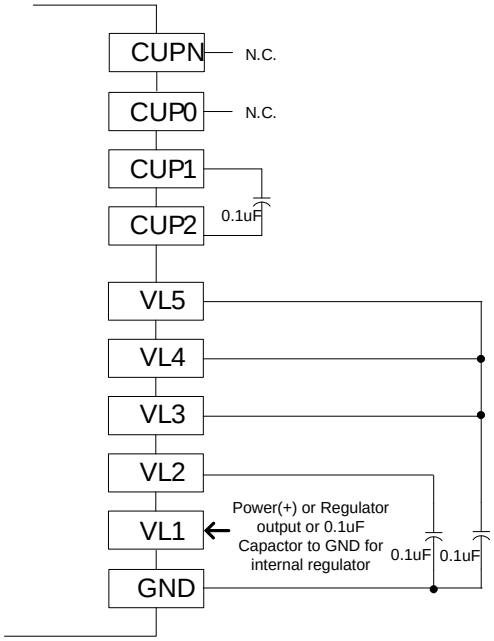
1-1-2-1-1. 由 VL1 產生 Charge Pump 模式下 選用 1/2 Bias 的 LCD panel

LCD panel 的 Vop = VL2 = 2 * VL1 (VL1 = 電源正極或是由內/外部穩壓線路提供)



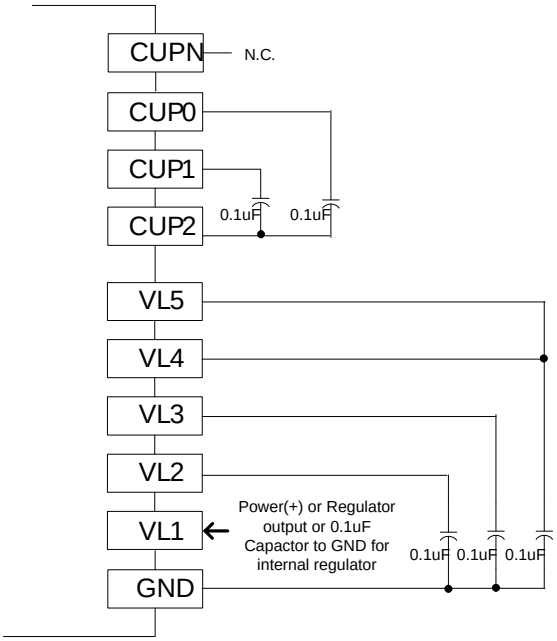
1-1-2-1-2. 由 VL1 產生 Charge Pump 模式下選用 1/3 Bias 的 LCD panel (一般尺寸的 LCD panel)

LCD panel 的 Vop = VL3 = 3 * VL1 (VL1 = 電源正極或是由內/外部穩壓線路提供)



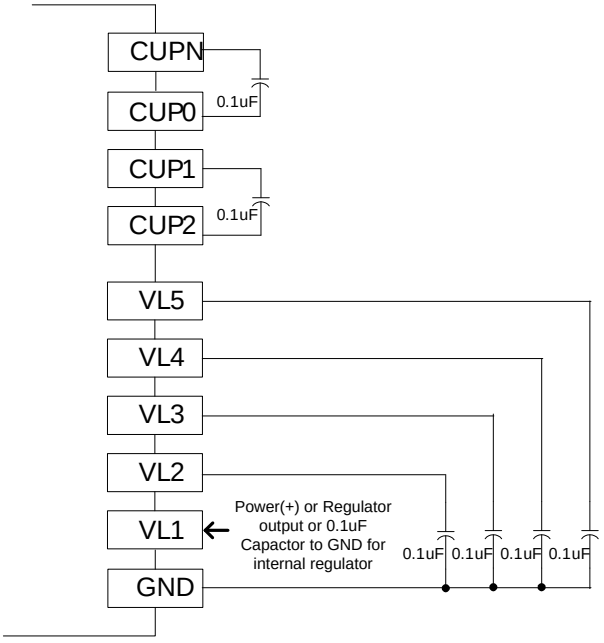
1-1-2-1-3. 由 VL1 產生 Charge Pump 模式下選用 1/4 Bias 的 LCD panel (一般尺寸的 LCD panel)

LCD panel 的 Vop = VL4 = 4 * VL1 (VL1 =電源正極或是由內/外部穩壓線路提供)



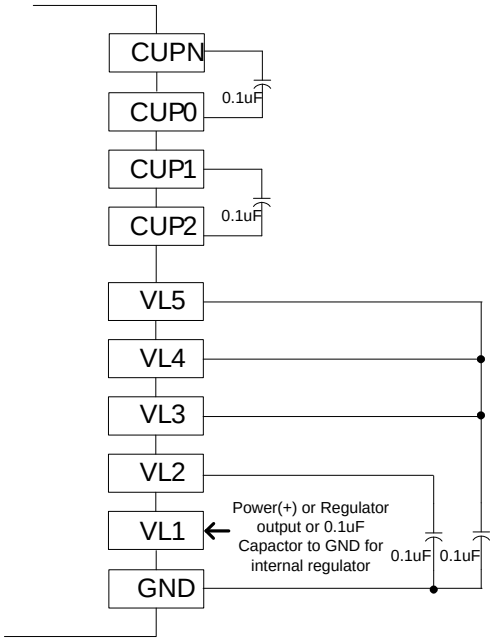
1-1-2-1-4. 由 VL1 產生 Charge Pump 模式下選用 1/5 Bias 的 LCD panel

LCD panel 的 Vop = VL5 = 5 * VL1 (VL1 =電源正極或是由內/外部穩壓線路提供)



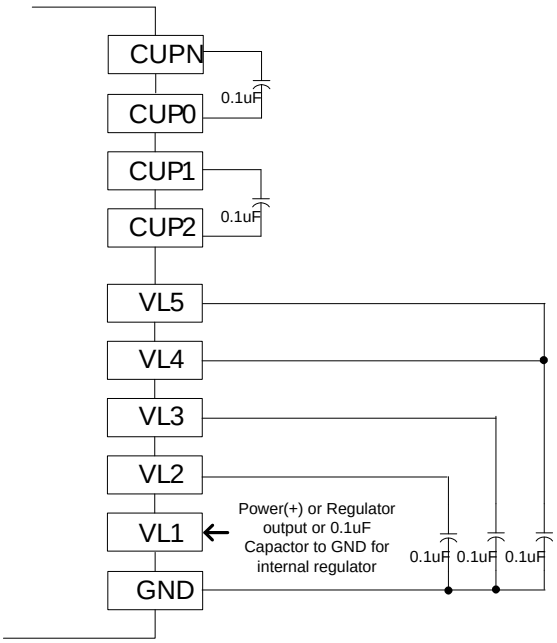
1-1-2-1-5. 由 VL1 產生 Charge Pump 模式下選用 1/3 Bias(TWO CAPACTOR FOR CUP1-2,CUPN-0)的 LCD panel (大尺寸的 LCD panel)

LCD panel 的 Vop = VL3 = 3 * VL1 (VL1 =電源正極或是由內/外部穩壓線路提供)



1-1-2-1-6. 由 VL1 產生 Charge Pump 模式下選用 1/4 Bias(TWO CAPACTOR FOR CUP1-2,CUPN-0)的 LCD panel (大尺寸的 LCD panel)

LCD panel 的 Vop = VL4 = 4 * VL1 (VL1 =電源正極或是由內/外部穩壓線路提供)



1-1-2-2. 利用 VL2 的電位作為 Charge Pump 線路的基礎電壓

在 3/5V 電源模式下，當 VL2 直接與電源正極連接或是選擇 VL2 電壓由內/外部穩壓線路提供時，Charge Pump 線路的基礎電壓就會由 VL2 供應。

此時 charge pump 線路所產生的 VL1，VL3 ~ VL5 電壓位準以 1/5Bias 為例如下：

VL1 = 1/2 * VL2

VL3 = 3/2 * VL2

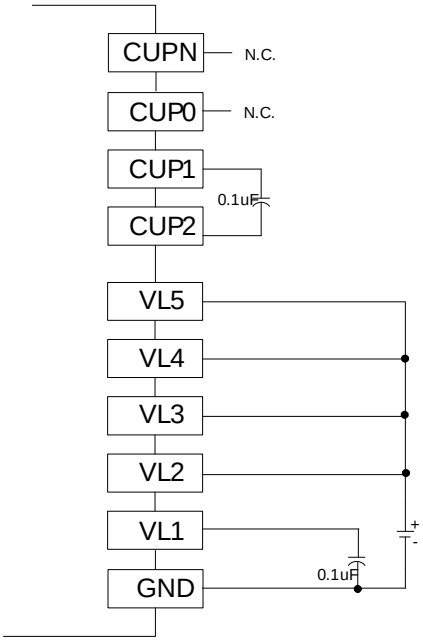
VL4 = 2 * VL2

VL5 = 5/2 * VL2

後面將說明利用 VL2 的電位作為 charge pump 線路的基礎電壓時，MCU 在不同的 LCD bias 的選項下所使用的系統電源應用線路。

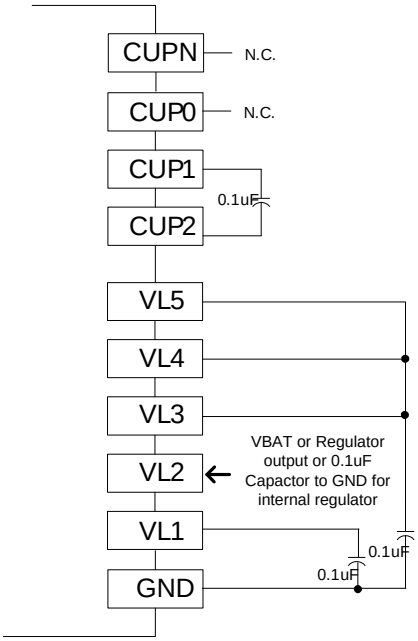
1-1-2-2-1. 由 VL2 產生 Charge Pump 模式下 選用 1/2 Bias 的 LCD panel

LCD panel 的 Vop = VL2 (VL2 = 電源正極, 因 VL 最高電位不可低於電源電壓故不支援內部穩壓)



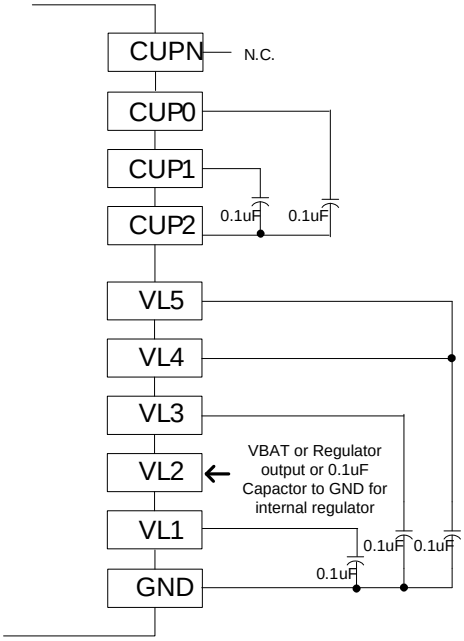
1-1-2-2-2. 由 VL2 產生 Charge Pump 模式下選用 1/3 Bias 的 LCD panel (一般尺寸的 LCD panel)

LCD panel 的 Vop = VL3 = 3/2 * VL2 (VL2 =電源正極或是由內/外部穩壓線路提供)



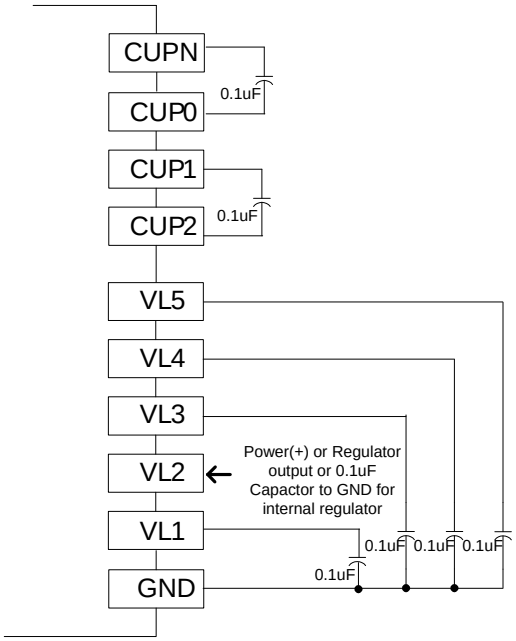
1-1-2-2-3. 由 VL2 產生 Charge Pump 模式下選用 1/4 Bias 的 LCD panel (一般尺寸的 LCD panel)

LCD panel 的 Vop = VL4 = 2 * VL2 (VL2 =電源正極或是由內/外部穩壓線路提供)



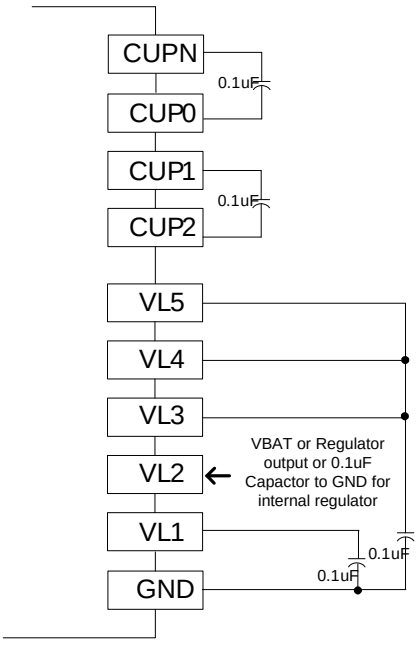
1-1-2-2-4. 由 VL2 產生 Charge Pump 模式下選用 1/5 Bias 的 LCD panel

LCD panel 的 Vop = VL5 = 5/2 * VL2 (VL2 =電源正極或是由內/外部穩壓線路提供)



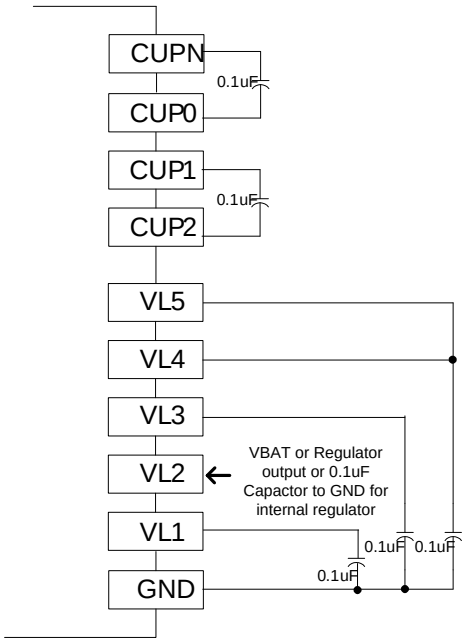
1-1-2-2-5. 由 VL2 產生 Charge Pump 模式下選用 1/3 Bias(TWO CAPACTOR FOR CUP1-2,CUPN-0)的 LCD panel (大尺寸的 LCD panel)

LCD panel 的 Vop = VL3 = 3/2 * VL2 (VL2 =電源正極或是由內/外部穩壓線路提供)



1-1-2-2-6. 由 VL2 產生 Charge Pump 模式下選用 1/4 Bias(TWO CAPACTOR FOR CUP1-2,CUPN-0)的 LCD panel (大尺寸的 LCD panel)

LCD panel 的 Vop = VL4 = 2 * VL2 (VL2 =電源正極或是由內/外部穩壓線路提供)



1-1-2-3. 利用最高 VL 的電位作為電容/阻分壓線路的基礎電壓

在 3/5V 電源模式下，當最高 VL 直接與電源正極連接時，電容/阻分壓線路的基礎電壓就會由最高 VL 供應。

此時電容分壓線路所產生的 VL1 ~ VL5 電壓位準以 1/5Bias 為例如下：

VL1 = 1/5 * VL5

VL2 = 2/5 * VL5

VL3 = 3/5 * VL5

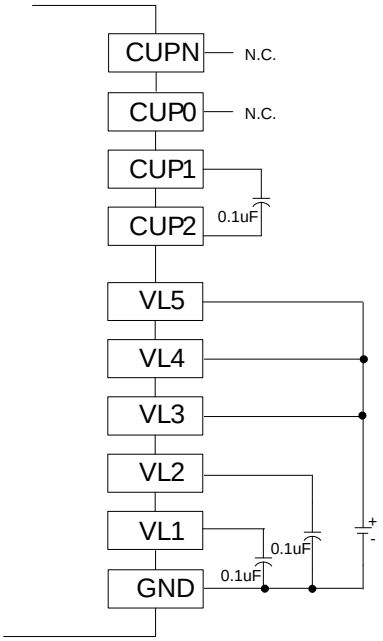
VL4 = 4/5 * VL5

VL5 = VL5

後面將說明利用最高 VL 的電位作為電容分壓線路的基礎電壓時，MCU 在不同的 LCD bias 的選項下所使用的系統電源應用線路(電阻分壓時則不需 CUP 電容，且若有提供 VL1~5 pins 則 VL1~5 接法同電容分壓)。

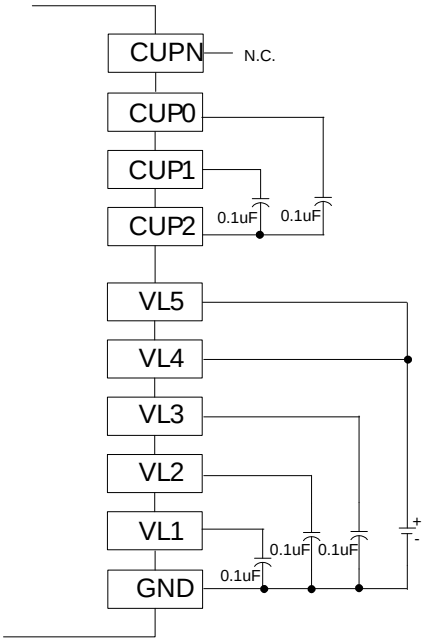
1-1-2-3-1. 由最高 VL 產生電容分壓模式下選用 1/3 Bias 的 LCD panel (一般尺寸的 LCD panel)

LCD panel 的 Vop = VL3 (VL3 =電源正極)



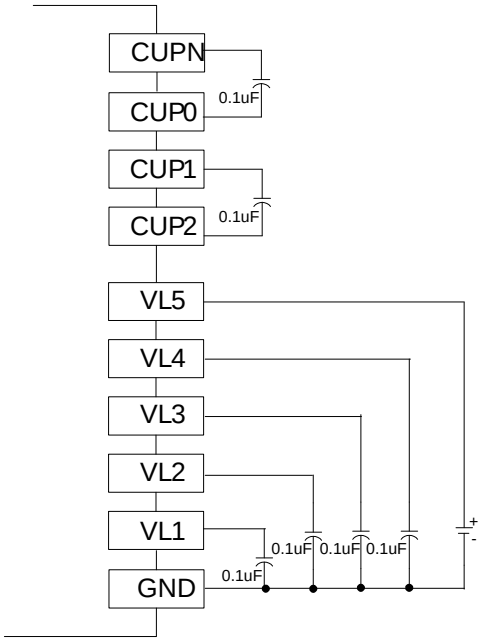
1-1-2-3-2. 由最高 VL 產生電容分壓模式下選用 1/4 Bias 的 LCD panel (一般尺寸的 LCD panel)

LCD panel 的 Vop = VL4 (VL4 =電源正極)



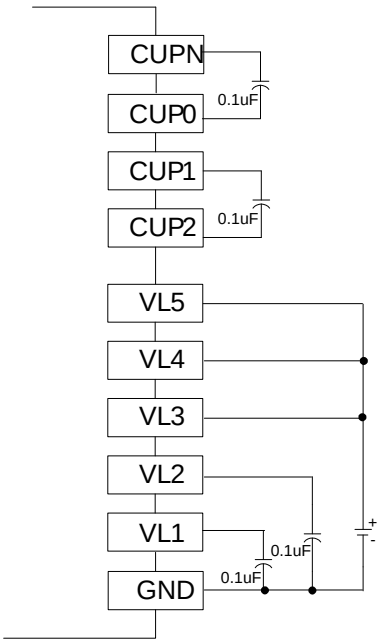
1-1-2-3-3. 由最高 VL 產生電容分壓模式下選用 1/5 Bias 的 LCD panel (一般尺寸的 LCD panel)

LCD panel 的 Vop = VL5 (VL5 =電源正極)



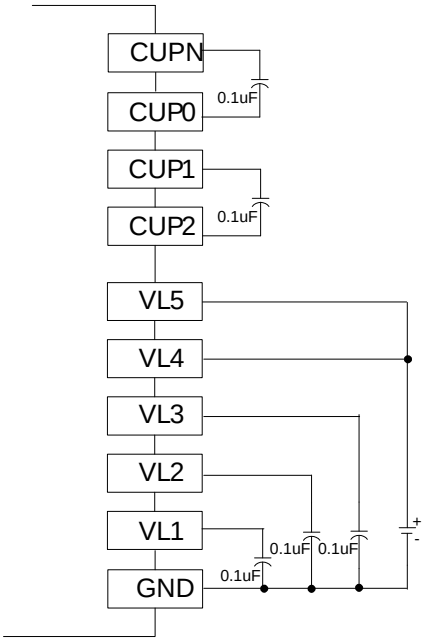
1-1-2-3-4. 由最高 VL 產生電容分壓模式下選用 1/3 Bias(TWO CAPACTOR FOR CUP1-2,CUPN-0) 的 LCD panel (大尺寸的 LCD panel)

LCD panel 的 Vop = VL3 (VL3 =電源正極)



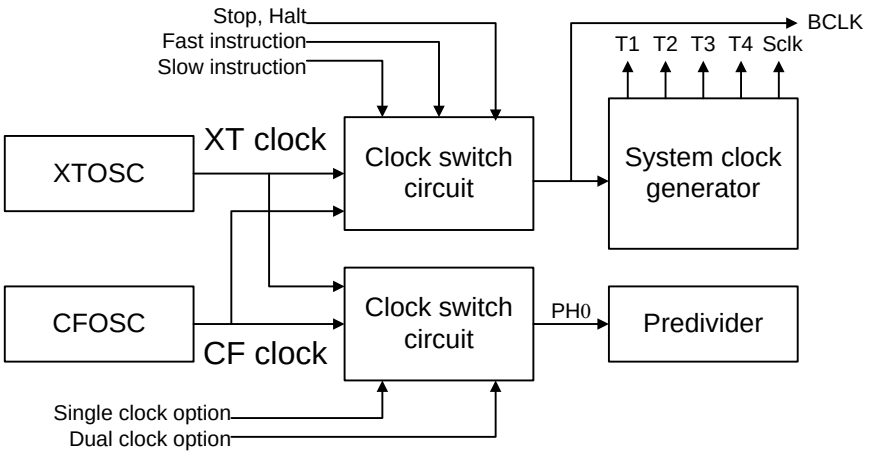
1-1-2-3-5. 由最高 VL 產生電容分壓模式下選用 1/4 Bias(TWO CAPACTOR FOR CUP1-2,CUPN-0) 的 LCD panel (大尺寸的 LCD panel)

LCD panel 的 Vop = VL4 (VL4 =電源正極)


CUPN
CUP0
CUP1
CUP2
VL5
VL4
VL3
VL2
VL1
GND
0.1uF
0.1uF
0.1uF
0.1uF
0.1uF
+
-

1-2. 系統時序控制單元

系統時序控制單元是由低速時鐘震盪器(XTOSC)，高速時鐘震盪器(CFOSC)，系統時鐘產生器(system clock generator)，預除器(pre-divider)以及時鐘切換線路組合而成。系統時序控制單元提供 MCU 執行指令以及其他週邊設備所需的各種不同的 clock 來源，其方塊圖如下所示：



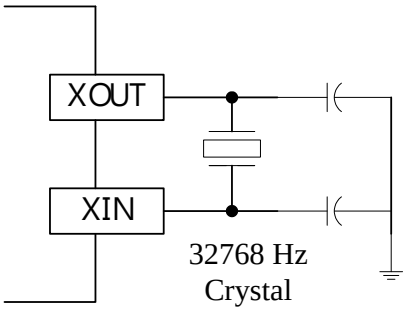
1-2-1. 低速時鐘震盪器(XTOSC)

低速時鐘震盪器可以提供低速的 clock 給 system clock generator，pre-divider，timer， IO port 的 chattering prevention 功能以及 LCD driver 等功能使用。當 MCU 的電源開啟之後或是 MCU 產生 STOP release 之後低速時鐘震盪器就會開始動作，只有在執行 STOP 指令之後才會停止動作。低速時鐘震盪器在 MCU 選用 “fast clock only option” 的 code option 時則永遠不會啟動。

4BIT 系列 MCU 只提供兩種不同種類的低速時鐘震盪器，external 32.768 KHz Crystal oscillator 以及 external RC oscillator，可以利用 code option 來選擇所需的種類。

1-2-1-1. External 32.768 KHz Crystal Oscillator

下圖是 External 32.768 KHz Crystal Oscillator 的應用線路：



(1) X'tal

在 PCB 上面，32.768 KHz 的 Crystal 元件位置必須儘量靠近 MCU IC 的位置，以減少外部的雜訊干擾 oscillator 的運作。

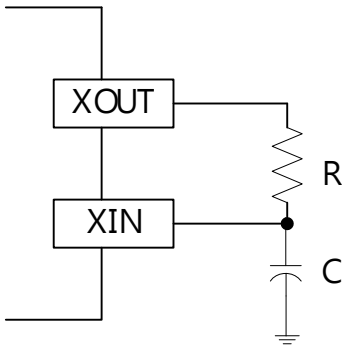
當 backup flag (BCF)設定為 1 時，可使得 Crystal oscillator 在電源雜訊較大的情況下仍能輸出較為穩定的 clock 信號，但是會增加 MCU 的耗電量。因此除非有特別的需求，否則請將 BCF flag 設定為 0。

下表說明 Crystal oscillator 在不同電源模式以及不同情況之下的耗電情況：

	1.5V 電源模式	3V 電源模式	EXTV 模式
BCF=1	耗電	耗電	耗電
BCF=0	省電	省電	耗電
Initial reset	耗電	耗電	耗電
After reset	省電	省電	耗電

1-2-1-2. External RC oscillator

下圖是 external RC oscillator 的應用線路：



(2) RC

1-2-2. 高速時鐘震盪器 (CFOSC)

高速時鐘震盪器提供一個高速的 clock(CF clock)給 MCU 使用，而且在不同的操作模式下會有不同使用方式。下面將說明高速時鐘震盪器的 3 種操作模式：

1. Fast Clock Only (MCU 只使用 CFOSC)操作模式：
在這個模式下 CF clock 將提供給 system clock generator，pre-divider，timer，I/O port 的 chattering prevention 功能以及 LCD driver 等功能使用。當 MCU 的電源開啟之後或是 MCU 產生 STOP release 之後，高速時鐘震盪器就會開始動作，只有在執行 STOP 指令之後才會停止動作。

2. Dual Clock 操作模式：
在這個模式下 CF clock 只提供給 system clock generator 使用，而且只有在執行 FAST 指令之後才會開始動作。執行 FAST 指令後，system clock generator 會將 clock source (BCLK)由 XT clock 切換成 CF clock，並開始提供高速的指令執行週期。
CFOSC 會在 MCU 進入 HALT mode，STOP mode 或是執行 SLOW 指令之後停止動作，而 MCU 在 CFOSC 停止動作之後會自動將 system clock generator 的 clock source 從 CF clock 切換到 XT clock (TM87ML28 可由 Mask Option 選擇繼續維持 BCLK = CF clock 使在離開 HALT/STOP mode 時直接起振 CFOSC 縮短等待時間(尤其是在離開 STOP mode 時可省去須先等 XTOSC 起振後才能再執行 FAST 指令等待 CFOSC 起振時間以爭取 SPI Slave mode 等應用反應時間)，甚至在 HALT mode 可由 Mask Option 選擇不停止 CFOSC 動作以維持 UART 運作)。

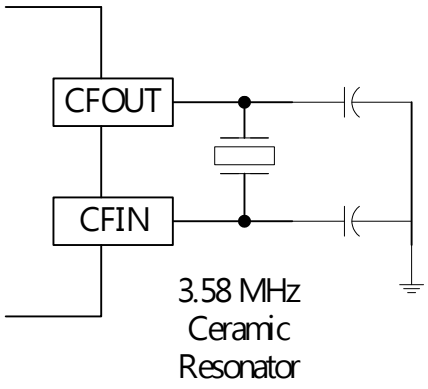
3. Slow Clock Only (MCU 只使用 XTOSC)操作模式：
在這個模式下 CFOSC 永遠不會動作。

為避免 MCU 的運作出現問題，在 3V 電源模式而且將 BCF flag 清除為 0 時，若 BAK 腳位的電壓會等於 VL1 腳位而且無穩壓的電壓的狀況下，因為 VL1 提供電流能力有限，建議此時不要以 CF Clock 的速度來執行程式指令，以免因為 BAK 電壓不穩定而使 MCU 的運作出現問題。

4BIT 系列 MCU 只提供三種不同種類的高速時鐘震盪器，external Ceramic Resonator oscillator，RC oscillator with external resistor 以及 RC oscillator with internal resistor，可以利用 code option 來選擇所需的種類。

1-2-2-1. External Ceramic Resonator oscillator

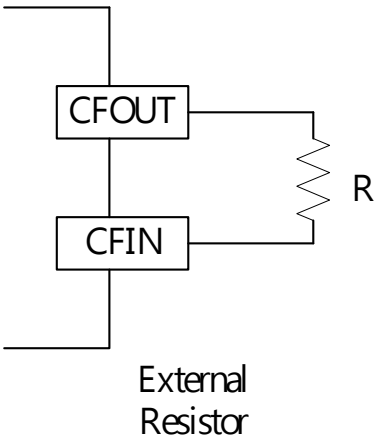
下圖是 external Ceramic Resonator oscillator 的使用方式：



在 PCB 上面，Ceramic Resonator 元件位置必須儘量靠近 MCU IC 的位置，以避免外部的雜訊干擾 oscillator 的運作。

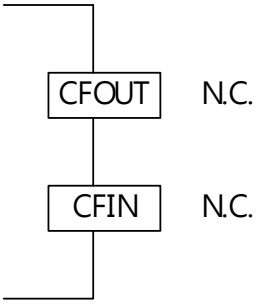
1-2-2-2. RC oscillator with External Resistor

下圖是 RC oscillator with external resistor 的使用方式：



1-2-2-3. RC oscillator with internal RC

下圖是 RC oscillator with internal resistor 的使用方式：



Internal RC

RC oscillator with internal resistor 可以提供數種不同的輸出頻率 (這兩種頻率是在 BAK 腳位上的電壓為 3V 時所測得)，使用者可以根據設計需求以 code option 作選擇。

1-2-3. 系統時序控制單元的操作模式

系統時序控制單元可以產生 MCU 運作時所需的各種時序的 clock，利用 code option 可以 選擇兩種操作模式，dual clock mode 以及 single clock mode。

每一種操作模式都有自己的 state machine，這些 state machine 包含了下面五個基本的狀態：

- RESET 狀態：當電源一旦開啟後或是 MCU 接收到 reset 的相關信號時 MCU 就會進入 RESET 狀態。
- FAST 狀態：XTOSC 以及 CFOSC 兩者同時啟動(dual clock mode)，或是只有 CFOSC 啟動 (fast clock only mode)。
- SLOW 狀態：只有 XTOSC 啟動。
- HALT 狀態：MCU 進入 HALT 模式。
- STOP 狀態：MCU 進入 STOP 模式。

下面會詳細描述各種操作模式的 state machine。

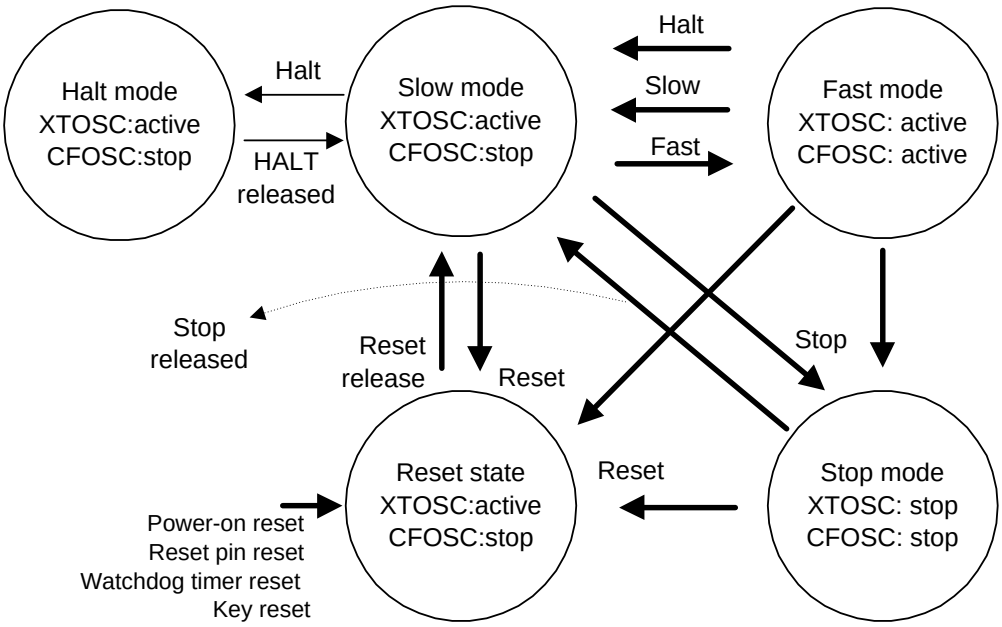
1-2-3-1. DUAL CLOCK 操作模式

在這個模式下 XTOSC 以及 CFOSC 都可以啟動或是停止使用，system clock generator 的 clock source 則可以透過 clock switch 的控制線路來選擇 XT clock 或是 CF clock。

XT clock 可以提供 system clock generator，pre-divider，timer，IO port 的 chattering prevention 功能以及 LCD driver 等所需的 clock 來源，而 CF clock 則只提供 system clock generator 所需的 clock 來源。

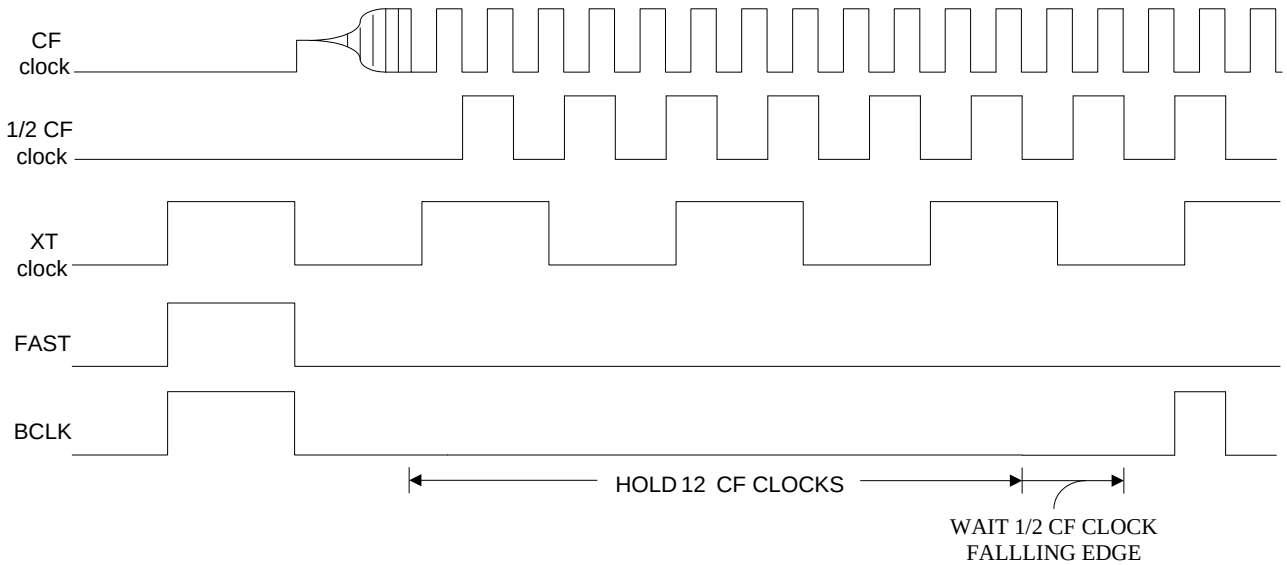
當 MCU 產生 HALT release 或是 STOP release 時, 除了如 TM87ML28 有提供 code option 可選擇維持在 Fast Mode 之外，system clock generator 的 clock 來源 (BCLK)會自動切換到 XT clock 並且停止 CFOSC 的動作。當 MCU 進入 STOP mode 之後， backup flag (BCF)會被自動設定為 1，以便 MCU 在產生 STOP release 的時候可以讓 XTOSC 順利啟動。

下圖說明 Dual Clock 操作模式的 state machine：



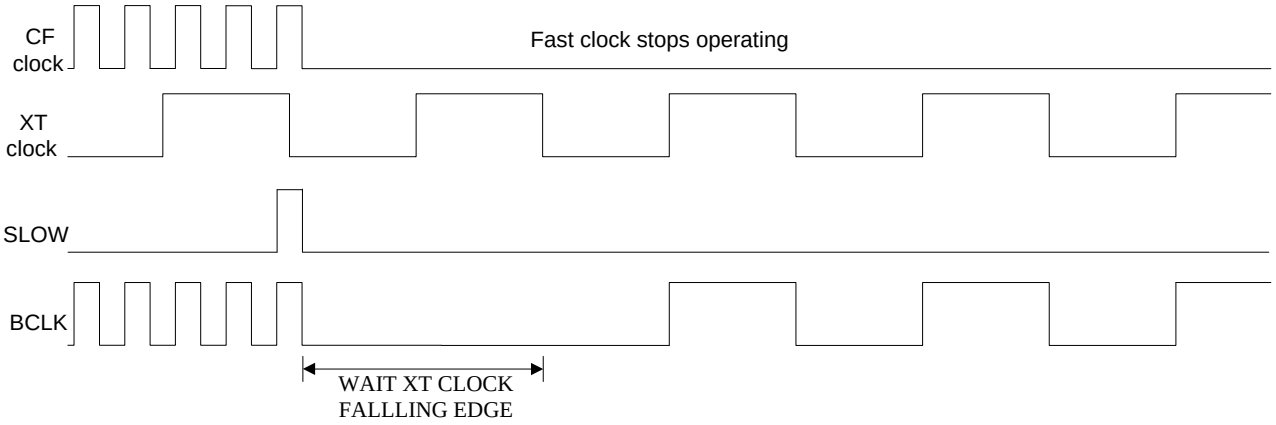
當程式在 SLOW Mode 下執行 FAST 指令之後 CFOSC 就會開始動作並強制 BCLK=0，但是 system clock generator 會等到 CFOSC 送出第 12 CF clock 完成起振程序之後，再等 1 CF 除頻 clock 下降緣觸發後(≦ 1 CF 除頻 clock 週期，CF clock 除頻值由執行 FAST 指令時設定值而定)才會將 clock source (BCLK)由 XT clock 切換到 CF 除頻 clock，這樣可以避免 MCU 在切換工作頻率時發生問題 。

下圖以執行 FAST $1 為例說明 BCLK 如何從 XT clock 切換到 CF 除頻 clock：



當程式執行 SLOW 指令之後 CFOSC 會立即停止動作並強制 BCLK=0，而 system clock generator 會等 XTOSC 送出第一個 XT clock 下降緣觸發(≦ 1 XT clock 週期)之後才會將 clock source (BCLK)由 CF clock 切換到 XT clock，這樣可以避免 MCU 在切換工作頻率時發生問題 。

下圖說明 BCLK 如何從 CF clock 切換到 XT clock：



受 DUAL CLOCK 操作模式架構影響，在 SLOW 模式下執行 SLOW 指令之後亦會延遲一個 XT clock 週期，而在 FAST 模式下執行 FAST 指令之後若未改變除頻值亦會延遲一個 CF 除頻 clock 週期，但若有改變除頻值則會因執行當時各除頻頻率狀態差異延遲≦ 一個新 CF 除頻 clock 週期

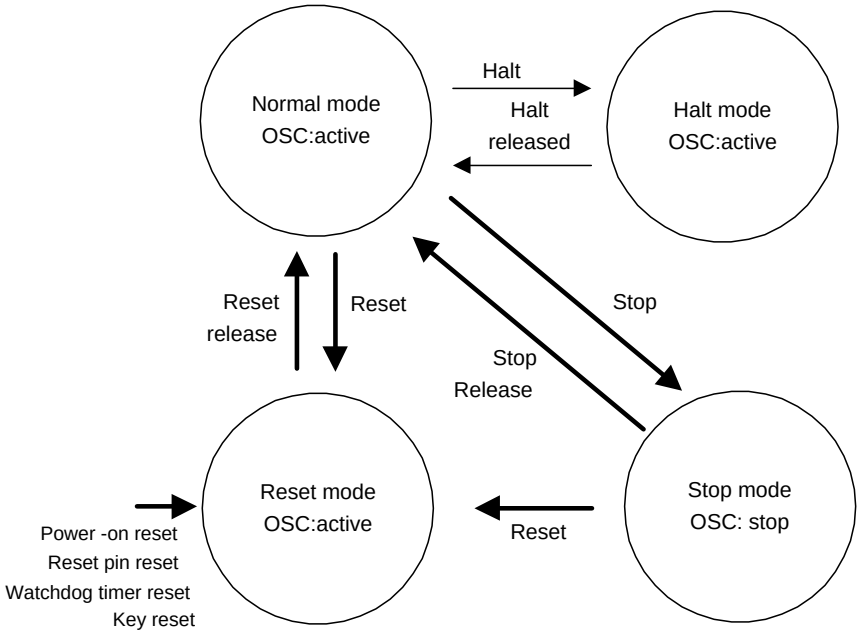
1-2-3-2. Single Clock 操作模式 (fast clock only or slow clock only)

Slow clock only (XT clock)以及 fast clock only (CF clock)兩者都屬於 Single clock 的操作模式，主要的差異在於 fast clock only (CF clock) 模式下 pre-divider 的 clock source (PH0) 以及 system clock generator 的 clock 來源 (BCLK)可以選擇的頻率範圍不同

在 Slow clock only (XT clock)下執行 SLOW 指令不會影響 BCLK，但 在 fast clock only (CF clock)下執行 FAST 指令若有切換除頻值則因執行當時各除頻頻率狀態差異，故切換後第一個週期會≦ 一個新 CF 除頻 clock 週期，但仍確保 BCLK=High or Low 至少有 0.5 CF clock 週期以確保 IC 可正常工作！

當 MCU 進入 STOP mode 之後， 除了部分產品有提供 BAK 內部穩壓可選擇 code option 維持 BCF=0 之外, backup flag (BCF)會被自動設定為 1，以便 MCU 在產生 STOP release 的時候可以讓 clock oscillator 順利啟動。

下圖說明 Single Clock 操作模式的 state machine：

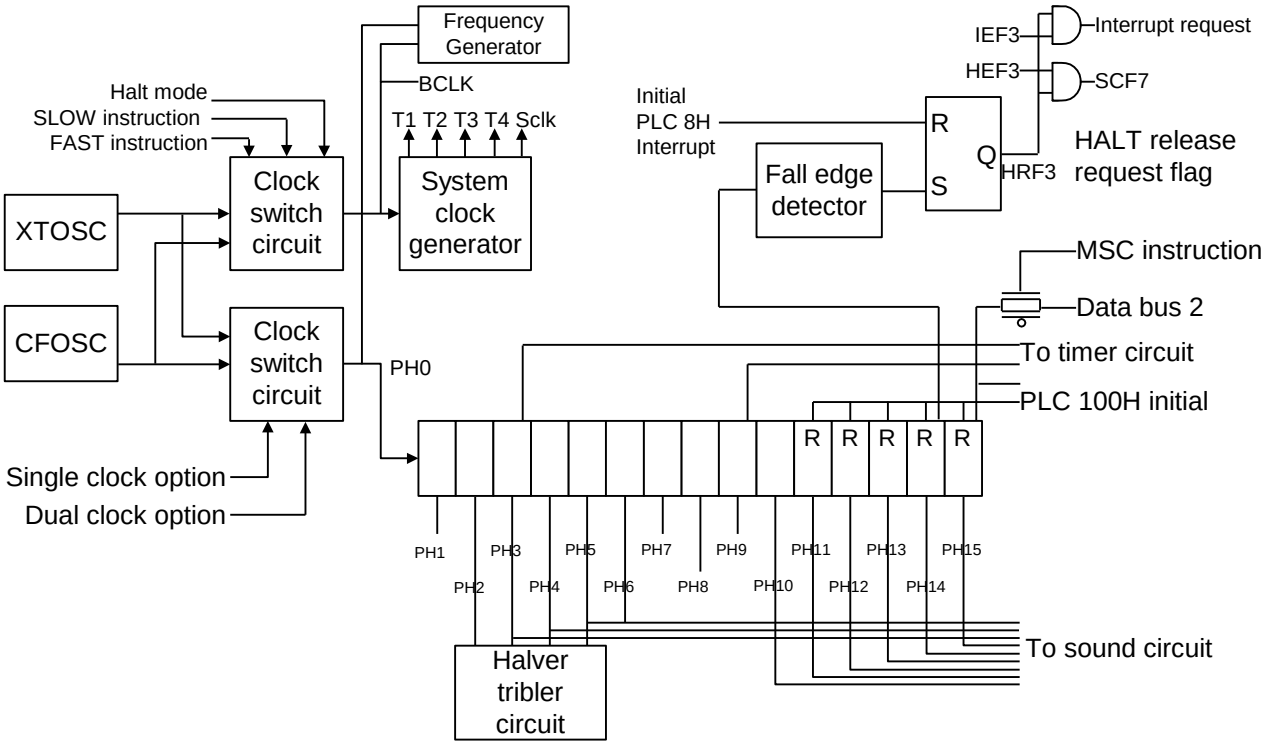


Transition Diagram of Single Clock Operation

1-2-4. PRE-DIVIDER

Pre-divider 是一個 15-bit 的 counter，如果從 clock source (PH0)輸入一個頻率為 N 的信號時，pre-divider 可以在輸出端產生 N/2(PH1) ~ N/2^15(PH15)等不同頻率的信號，而每一個輸出信號都會在 PH0 信號的下降緣改變信號位準。這些不同頻率的 clock 可提供給 charge-pump circuitry，LCD driver，frequency generator，timers，interrupt control circuit，HALT released control 以及 IO port 的 chattering prevention 功能等周邊設備使用。

下圖說明 pre-divider 以及其相關週邊設備的架構圖：



Pre-divider 的最後 5 級輸出信號(PH11 ~ PH15)會在 MCU 進入 RESET 狀態時或是執行 PLC 100H 指令之後歸零。

PH14 輸出信號的每一個下降緣都會將 halt release request flag 3 (HRF3)設定為 1，只有在 MCU 進入 RESET 狀態，執行 PLC $8 指令或是中斷發生時才會將 halt released request flag (HRF3)會清除為 0。

如果 pre-divider interrupt enable flag 3 (IEF3)已經預先設定為 1，當 halt released request flag 3 (HRF3)設定為 1 時， pre-divider 就會發出一個中斷的請求，MCU 接受中斷請求之後就會執行相關的中斷服務；如果 halt release enable flag 3 (HEF3) 已經設定為 1， pre-divider 就會向 MCU 送出一個 halt release request 信號讓 MCU 產生 HALT release， 同時也會將 status register 3 (STS3)中的 start condition flag 7 (SCF7) 設定為 1。

當 pre-divider 的 clock source (PH0)的頻率是 32768 Hz 時，PH14 就會在每隔 0.5 秒的時候將 HRF3 flag 設定為 1，因此可以做為計時器應用的基準信號。

如 TM87ML28 有提供 ADJ 指令功能則可藉由 Timer 每次產生 Oveflow 時自動增加或減少一個 PH0 週期來達到校正 pre-divider 頻率的作用，例如當使用 32768Hz X'tal 元件於時鐘等應用, 欲減少每日誤差, 可藉以下公式計算 Timer count 使時間校正至精準範圍內:

Timer 校正 Count 數(C) = (86400/S) / 2^N (S : 每日飄移秒數，N : 校正 Pre-divider 所選用 Ctm=PH 編號 ， C 為四捨五入整數)

注意 : 為了確保校正動作正確， 請選用 PH3/5/7/9/11/13/15 為 Ctm。

校正後每日飄移秒數(Sa)= (S (Ca-1)-86400)/Ca (Ca = 2^N x C)

範例:

該 32768Hz X'tal 元件每日飄移+1.1 秒, 校正 Pre-divider 計算如下:

Ctm=PH11 ：86400 / (1.1 x 2048) = 38.3.. => C=38 => Ca=77824 => Sa=- 0.01021150..秒

Ctm=PH9 ：86400 / (1.1 x 512) = 153.4.. => C=153 => Ca=78336 => Sa=- 0.00295521..秒

Ctm=PH7 : 86400 / (1.1 x 128) = 613.6.. => C=614 => Ca=78592 => Sa=+0.00063746..秒

Ctm=PH5 : 86400 / (1.1 x 32) = 2454.5.. => C=2455 => Ca=78560 => Sa=+0.00018966..秒

Ctm=PH3 : 86400 / (1.1 x 8) = 9818.8.. => C=9819 => Ca=78552 => Sa=+0.00007765..秒

Ctm 選擇越高頻能校正得越精準，但是就以此範例來說 Ctm=PH9/3 以下超過 64/4096 便需要使用 12/18bits Timer。故此架構可依精準要求&每日飄移範圍&可使用 Timer counter bits 彈性調整最高 PH 頻率進行校正。

以 Ctm3=PH5 執行上述校正指令流程範例如下:

```
ADJ    $7      ;X2=1 : 每次 timer overflow 減少一個 PH0 cycle , X1,0=3 : 設定由 Timer3 校正。
STE    $3      ;X2=0 : 正常 RTM 讀取 bits , X1,0=3 : 設定延伸 Timer3 counter 為 12bits。
SHLX           ;設定 HL=0000 ( TD15~0=4996 : Ctm=PH5 , Counter11~0=996H(=2454)。
setdat $0000
SRP    $4      ;X2=1 : 啟動 Timer3 re-load 校正值當每次 overflow & RL3=1)
T3TH   @HL     ;啟動 Timer3(Ctm3=PH5 , Timer3 overflow cycle=2454+1)
SF     $40     ;X6=1 : 啟動 RL3
```

TM87ML28 可使用 Table ROM 執行 Timer 可藉由執行 PTR 指令在生產時將個別 Ctm&校正值進行燒錄至 Table ROM。

注意：使用此校正功能時因 PH1~15 會受影響, 故例如 RFC control by TM2 等要求每次執行計數時間須相同時 Ctm2 須避免使用 PH1~15。

在選用 fast clock only 的選項時，pre-divider 可能會因為 CF clock 的頻率過高而無法產生 LCD driver 所需的低頻訊號，所以 MCU 應先將 BCLK 信號(CF clock)降頻之後才提供給 pre-divider 的 clock source (PH0)使用。而降頻之後的頻率則可以利用 code option 來做 選擇。

最多可供選擇的頻率如下表所示：

Mask Option name	Selected item
PH0 <-> FTOSC FOR FAST ONLY	(1) PH0 = FTOSC
PH0 <-> FTOSC FOR FAST ONLY	(2) PH0 = FTOSC/4
PH0 <-> FTOSC FOR FAST ONLY	(3) PH0 = FTOSC/8
PH0 <-> FTOSC FOR FAST ONLY	(4) PH0 = FTOSC/16
PH0 <-> FTOSC FOR FAST ONLY	(5) PH0 = FTOSC/32
PH0 <-> FTOSC FOR FAST ONLY	(6) PH0 = FTOSC/64
PH0 <-> FTOSC FOR FAST ONLY	(7) PH0 = FTOSC/128
PH0 <-> FTOSC FOR FAST ONLY	(8) PH0 = FTOSC/256

Notes:

1. 當該 IC 有新增提供 BCLK 除頻的功能時，因 PH0 並不受 BCLK 除頻影響，故 Mask Option 敘述將 “BCLK” 更改為 “FTOSC” (FTOSC : FAST oscillation clock output)。

下表說明 pre-divider 的 clock source (PH0)在不同狀態下所能使用的 clock 來源：

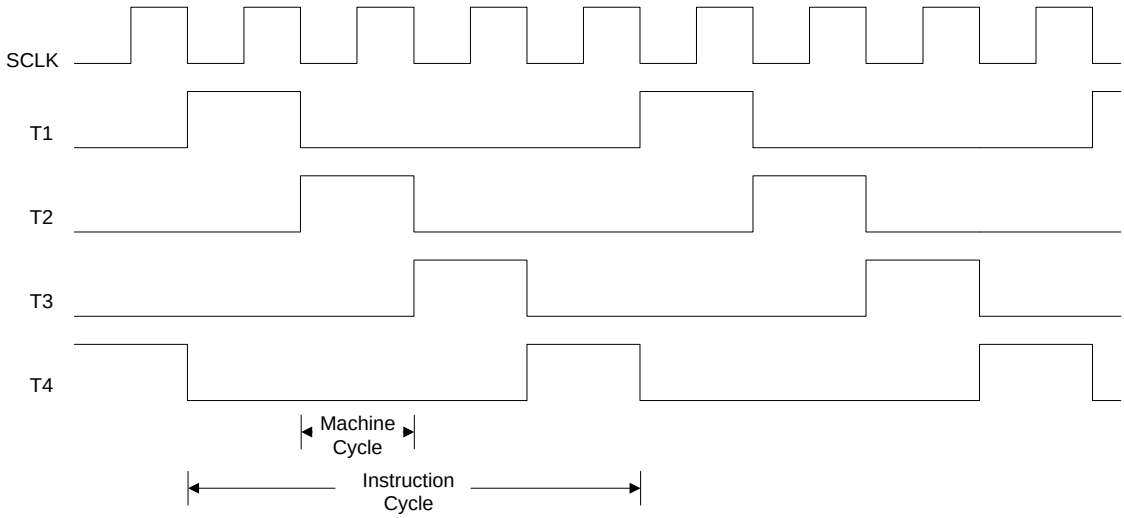
Condition	PH0
Slow clock only option	XT clock
Fast clock only option	CF clock
RESET state(dual clock option)	XT clock
Halt mode(dual clock option)	XT clock
Slow mode(dual clock option)	XT clock
Fast mode(dual clock option)	XT clock

1-2-5. System Clock Generator

System clock generator 可以產生 MCU 執行指令所需的各種時序。在 4BIT 系列 MCU 中，大部分的指令週期都可以在四個 machine cycle 內完成，但是也有一些指令週期需要八個 machine cycle 才能完成。

每一個 machine cycle 與 XT clock (slow clock mode)或是 CF clock (fast clock mode)的 clock 週期一致。

下圖說明基本的指令執行時序：



當程式執行 FAST 或是 SLOW 指令時可以讓 system clock generator 的 clock source 在 XT clock 以及 CF clock 之間做不同工作頻率的切換，下表說明 system clock generator 的 clock source 在不同狀態下的設定情況(TM87ML28 在 HALT mode 可由 Mask Option 選擇為 CF clock)：

Condition	BCLK
Slow clock only option	XT clock
Fast clock only option	CF clock
RESET state (dual clock option)	XT clock
Halt mode (dual clock option)	XT/CF clock
Slow mode (dual clock option)	XT clock
Fast mode (dual clock option)	CF clock

4BIT 系列 MCU 提供在 BCLK=CF clock 模式下，可藉由”FAST” 指令設定使 BCLK 頻率為 FTOSC 的除頻值, 使 FAST Mode 不用一直維持在最高頻率耗電狀態下, 可供選擇的頻率如下表所示：

指令	Selected item
FAST $0	(1) BCLK = FTOSC
FAST $1	(2) BCLK = FTOSC/2
FAST $2	(3) BCLK = FTOSC/4
FAST $3	(4) BCLK = FTOSC/8
FAST $4	(5) BCLK = FTOSC/16
FAST $5	(6) BCLK = FTOSC/32
FAST $6	(7) BCLK = FTOSC/64
FAST $7	(8) BCLK = FTOSC/128

1-3. PROGRAM COUNTER (PC)

Program counter 是由一個 16-bit 的 binary counter 所組成(不同規格的 MCU 可能會使用較小的 counter)，可作為 program memory (ROM)的定址之用，最大可定址到 62K 的位址數 (PC0 ~ PC15)。最高的五個位元 (PC15 ~ PC11)是 program memory 的 bank register，用來指示程式所在的 bank 位置，其餘較低的位元 (PC10 ~ PC0)則是用來指示 程式在 bank 中的位址 。

Program counter 的運作方式如下：

1. 當一個 word 長度的指令執行結束之後會將目前的 PC 值加 1 。

PC ← PC + 1

2. 當兩個 word 長度的指令執行結束之後會將目前的 PC 值加 2。

PC ← PC + 2

3. 當程式執行 JB0/JB1/JB2/JB3/JNZ/JNC/JZ/JC/JMP 跳躍指令，CALL 指令，中斷服務副程式或是 MCU 進入 RESET 狀態時，會將 table 1-1 中各個相關的特定位址載入 program counter。
如果這些指令載入的位址與目前的 bank 位置不同時，如 TM89 系列 & TM87ML28 的 compiler 程式會自動在該指令之前插入一個 SPBK 指令，將 bank 位置調整成載入位址的 bank 位置。一般 TM87 系列 JB0/JB1/JB2/JB3/JNZ/JNC/JZ/JC 指令只能在同樣 page 範圍內，CALL&指令則須由 compiler 經 BANK 轉換區切換(3 個指令週期)才能跳躍至不同 bank。

PC ← specified address shows in table 1-1

4. 當程式執行 JAC 指令時，會將 AC 值所指定的特定位址載入 program counter；如果 AC 值超出設定對應值 X 時(X 是 JAC 指令中的運算元)會將下一個指令的位址載入 program counter。

If(AC<=X) PC ← 指令中的特定位址(PC+AC+1)

else PC ← 下一個指令的位址(PC+X+2 ; X 為 AC 最大對應設定值)

5. 執行 RTS 指令後會將 stack pointer 所指向的 STACK 內容值載入 program counter。

(a). Stack pointer ← stack pointer – 1

(b). PC ← stack pointer 所指向的 STACK 內容值

6. 當程式執行 CALL 指令時, program counter 的內容值會先加 1 之後才儲存到 STACK 中。

(a).目前 Stack pointer 所指向的 STACK 位置 ← PC + 1

(b). Stack pointer ← stack pointer + 1

7. 當 MCU 接受中斷請求之後，目前的 program counter 內容值會直接儲存到 STACK 中。

(a). 目前 Stack pointer 所指向的 STACK 位置 ← PC

(b). Stack pointer ← stack pointer + 1

8. 當程式執行 CAC 指令時，program counter 的內容值會先加 X+2 (X 是 CAC 指令中的運算元) 之後才儲存到 STACK 中，然後會將 AC 值所指定的特定位址載入 program counter；如果 AC 值大於 X 時會將緊跟在 CAC 之後的下一個指令的位址載入 program counter。

If (AC <=X)

(a). 目前 Stack pointer 所指向的 STACK 位置 ← PC +X + 2

(b). Stack pointer ← stack pointer + 1

(c). PC ← 指令中的特定位址 (PC+AC+1)

Else if (AC >X)

PC ← 下一個指令的位址 (PC+X+2 ; X 是 CAC 指令中的運算元)

Table 1- 1

	PC15	PC14	PC13	PC12	PC11	PC	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
Initial reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Interrupt 2																
(INT pin)	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Interrupt 0																
(input port A, C or D)	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0
Interrupt 1																
(timer 1 interrupt)	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
Interrupt 3																
(pre-divider interrupt)	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0
Interrupt 4																
(timer 2 interrupt)	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Interrupt 5																
(Key Scanning interrupt)	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
Interrupt 6																
(RFC counter interrupt)	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
Interrupt 7																
(timer3 counter interrupt)	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
JAC,CAC instruction	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
CALL& others Jump instruction	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

Notes:

1. D15 ~ D0: 可以在指令中直接以“SETDAT”的方式來設定。
2. P15 ~ P0: 可以在指令運算元中直接設定。

1-3-1. JAC 指令的用法(由 TM89 系列&TM87ML28 提供)

JAC 指令可以提供最多 16 個不同的目的位址供程式選擇使用，目的位址的選擇是依據目前的 AC 內容值來決定。

JAC 指令是一個 multi-word 的指令，可能需要花費四個 (AC>X) 或是八個 (AC<=X)machine cycle 才能執行完成。

下面是 JAC 指令的語法。

```
Address     OPcode      Operand
PC          JAC          X          ; X=0~15
PC+1        SETDAT       label_0    ; if AC=0, label_0 is destination
PC+2        SETDAT       label_1    ; if AC=1, label_1 is destination
-----------------------------
```

```
PC+X+1      SETDAT      label_x      ; if AC=X, label_x is destination
PC+X+2      (Next Instruction)       ; if AC>X
```

指令運算元中的 X 表示 JAC 指令中共有 X+1 個目的位址可供程式選擇使用。

當 AC<=X 時，MCU 會將目前 AC 內容值所對應到的目的位址載入 program counter 中，在這個情況下，程式需要八個 machine cycle 才能完成 JAC 的指令，在這八個 machine cycle 中 MCU 會自動禁止所有的中斷請求。

當 AC>X 時，沒有與 AC 內容值相對應的目的位址可供選擇，MCU 會將下一個指令的位址(目前的 PC 值+X+2) 載入 program counter 中，在這個情況下，程式只需要四個 machine cycle 就能完成 JAC 的指令，在這四個 machine cycle 中 MCU 會自動禁止所有的中斷請求。

Example:

```
            LDS         $20,$0
loop:       JAC         5
            SETDAT      label_0     ; jump to label_0 if AC=0
            SETDAT      label_1     ; jump to label_1 if AC=1
            SETDAT      label_2     ; jump to label_2 if AC=2
            SETDAT      label_3     ; jump to label_3 if AC=3
            SETDAT      label_4     ; jump to label_4 if AC=4
            SETDAT      label_5     ; jump to label_5 if AC=5
            INC*        $20         ; if AC > 5, AC = AC+1
            JMP         loop        ;
```

1-3-2. CAC 指令的用法(由 TM89 系列&TM87ML28 提供)

CAC 指令可以提供最多 16 個不同的副程式位址供程式呼叫使用，副程式位址的呼叫是依據目前的 AC 內容值來決定。

CAC 指令是一個 multi-word 的指令，可能需要花費四個 (AC>X) 或是八個 (AC<=X) machine cycle 才能執行完成。下面是 CAC 指令的語法。

```
Address     OPcode      Operand
PC          CAC          X          ; X=0~15
PC+1        SETDAT      label_0     ; if AC=0, label_0 subroutine is called
PC+2        SETDAT      label_1     ; if AC=1, label_1 subroutine is called
-------------------------------
PC+X+1      SETDAT      label_x     ; if AC=X, label_x subroutine is called
PC+X+2      (Next Instruction)      ; return or if AC>X
```

指令運算元中的 X 表示 CAC 指令中共有 X+1 個副程式位址可供程式呼叫使用。

當 AC<=X 時，MCU 會先將下一個指令的位址(目前的 PC 值+X+2)儲存到 stack 中，然後再將目前 AC 內容值所對應到的副程式位址載入 program counter 中，在這個情況下，程式需要八個 machine cycle 才能完成 CAC 的指令，在這八個 machine cycle 中 MCU 會自動 禁止所有的中斷請求。

當 AC>X 時，沒有與 AC 內容值相對應的副程式位址可供選擇，MCU 會將下一個指令的位址(目前的 PC 值+X+2) 載入 program counter 中，在這個情況下，程式只需要四個 machine cycle 就能完成 CAC 的指令，在這四個 machine cycle 中 MCU 會自動禁止所有的中斷請求。

Example:

```
            LDS     $20,$0 ; initial AC=0
loop:       CAC     5
            SETDAT  label_0; jump to label_0 if AC=0, and return to label_x
            SETDAT  label_1; jump to label_1 if AC=1, and return to label_x
            SETDAT  label_2; jump to label_2 if AC=2, and return to label_x
            SETDAT  label_3; jump to label_3 if AC=3, and return to label_x
            SETDAT  label_4; jump to label_4 if AC=4, and return to label_x
            SETDAT  label_5; jump to label_5 if AC=5, and return to label_x
label_x:    INC*    $20         ; return or if AC > 5, AC = AC+1
            JMP     loop        ;
```

1-4. PROGRAM MEMORY AND TABLE ROM

4BIT 系列 MCU 的 Program memory 最大可達 64K x 16 bits，是用來存放程式指令以及 look-up table 的唯讀記憶體(ROM)。64K 的 program memory 被分割成 32 個 bank，每一個 bank 包含了 2048 個位址。

4BIT 系列 MCU 也提供了 look up table 的功能，稱之為 table ROM。這個 table ROM 在 實體上是與 program memory 共用一個記憶體，最大可以定義成 60K x 8 bits。一旦 MCU 將部份的 program memory 劃分給 table ROM 使用後，MCU 可使用的 program memory 就會相對減少。

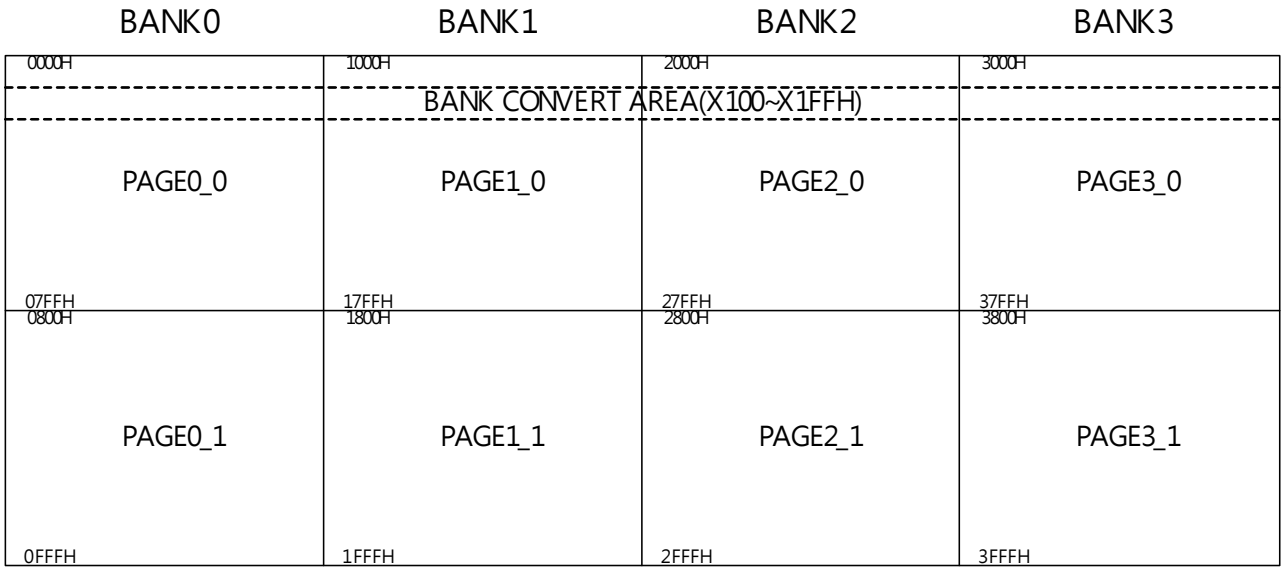
Table ROM 與 program memory 之間的記憶體大小的相對關係會依不同的 MCU 規格而有所不同，請參照每一個 MCU 的 data sheet 來決定所能使用的記憶體容量。

1-4-1. PROGRAM MEMORY

4BIT 系列 program memory 分割 bank 方式因新舊架構不同而有 2k&4Kword 兩種格式:

1-4-1-1. 4K word bank

如一般 TM87 架構以 4K word 為一個 bank，圖示如下說明 program memory 各個 bank 的分頁方式：



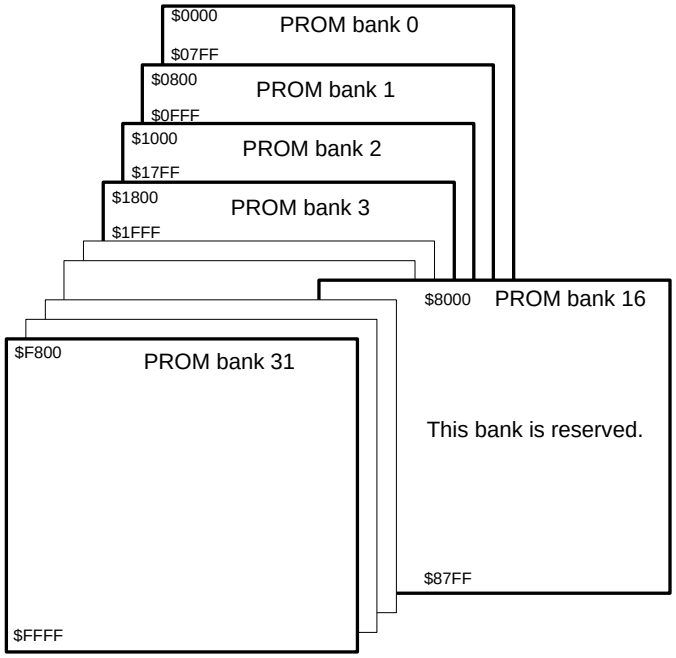
每個 bank 內會分割成兩個 page，每個 page 佔 2K word，JB0~3，JNZ，JNC，JZ，JC 指令只能在同一個 page 內跳躍，只有 CALL & JMP 指令可在 page 間跳躍。

當該 Body 大於 4Kword 時，Compiler 會佔用各 bank 位址：100~1FFH 以進行 bank 轉換，當以 CALL & JMP 指令跳躍至其他 bank 時會花費三個指令週期(亦即 12 系統 clock 週期) ，若為同一 bank 則僅 花費一個指令週期。

如果程式指令在 program memory 的位址是在某一個 page/bank 的最後一個位址時，當指令執行結束之後，program counter 會自動指向下一個 page/bank 的第一個位址，使用者不需考慮程式跨越 page/bank 之間的問題。

1-4-1-2. 2K word bank

如 TM89 系列&TM87ML28 架構以每 2K 位址為一個 bank，下圖說明 program memory 各個 bank 的分頁方式：



第 16 個 bank 的 program memory 有特殊用途，無法供程式使用。

如果程式指令在 program memory 的位址是在某一個 bank 的最後一個位址時，當指令執行結束之後，program counter 會自動指向下一個 bank 的第一個位址，使用者不需考慮程式跨越 bank 之間的問題。

跳躍或是副程式呼叫等指令中，只有 JAC 以及 CAC 指令能夠對全部的程式記憶體空間做直接定址，因此不需要做 bank 的調整。除此之外，其他的跳躍或是呼叫指令包含 CALL 以及 JMP 指令都因為 OP code 的位元數限制，只能在同一個 bank 的程式記憶體位址範圍內做直接定址。

如果這些指令也要能對全部的程式記憶體空間做直接定址的話，就必須以絕對位址或是 label 的方式來描述的目位址或是副程式位址，這樣 compiler 程式在編譯 source code 的 時候就會檢查目的位址或是副程式位址與該指令位址是否在同一個 bank 中，如果不是就會在該指令之前插入一個 SPBK 指令來調整 program counter 的 bank 值。

MCU 在執行 SPBK 指令以及下一個位址的指令時會暫停處理所有的中斷請求，以免造成 bank 設定上的錯誤。

舉例說明：

The source file is as below:

```
        ORG     $0130           ; bank 0
        CALL    label1          ; bank 0
        …
        …
        ORG     $0F45           ; bank 1
Label1:
```

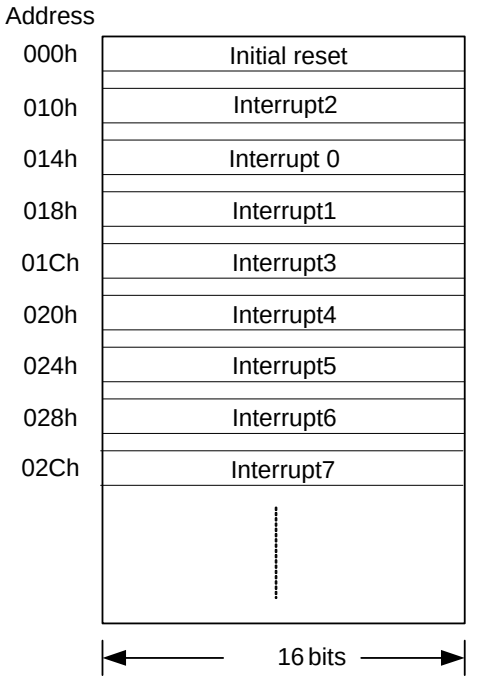
```
        DAA                     ; bank 1
```

The source file after compiling:

```
        SPBK    1               ; set bank 1 by compiler
CALL    0F45h                   ; in bank 0
        …
        …
        DAA                     ; bank 1, PC= 0F45h
```

在 page/bank 0 的 program memory 中，有一些特定的位址已經保留作為 MCU 的中斷服務副程式位址使用，例如 reset address (000H), interrupt 0 address (014H), interrupt 1 address (018H), interrupt 2 address (010H), interrupt 3 address (01CH), interrupt 4 address (020H), interrupt 5 address (024H), interrupt 6 address (028H), and interrupt 7 address (02CH)。

下圖說明這些中斷服務副程式位址的關係：



Instruction ROM (PROM) organization

1-4-2. TABLE MEMORY (TROM)

Table ROM 可以用來存放一些程式中所使用的常數(constant)或是 look up table，而所有的 Table ROM 的資料都只能利用 HL index register(@HL)以索引定址的方式來讀取。

Table ROM 最多共有三種資料長度的讀取格式：4 bits、8 bits 及 16 bits(TM89 系列&TM87ML28 提供)，而且都能在四個 machine cycle 內完成讀取的動作。

- LDL(*), LDH(*)等相關指令可執行 4 bits table ROM 資料的讀取
- LCD 等相關指令可執行 8 bits table ROM 資料的讀取
- LCDH 等相關指令可執行 16 bits table ROM 資料的讀取

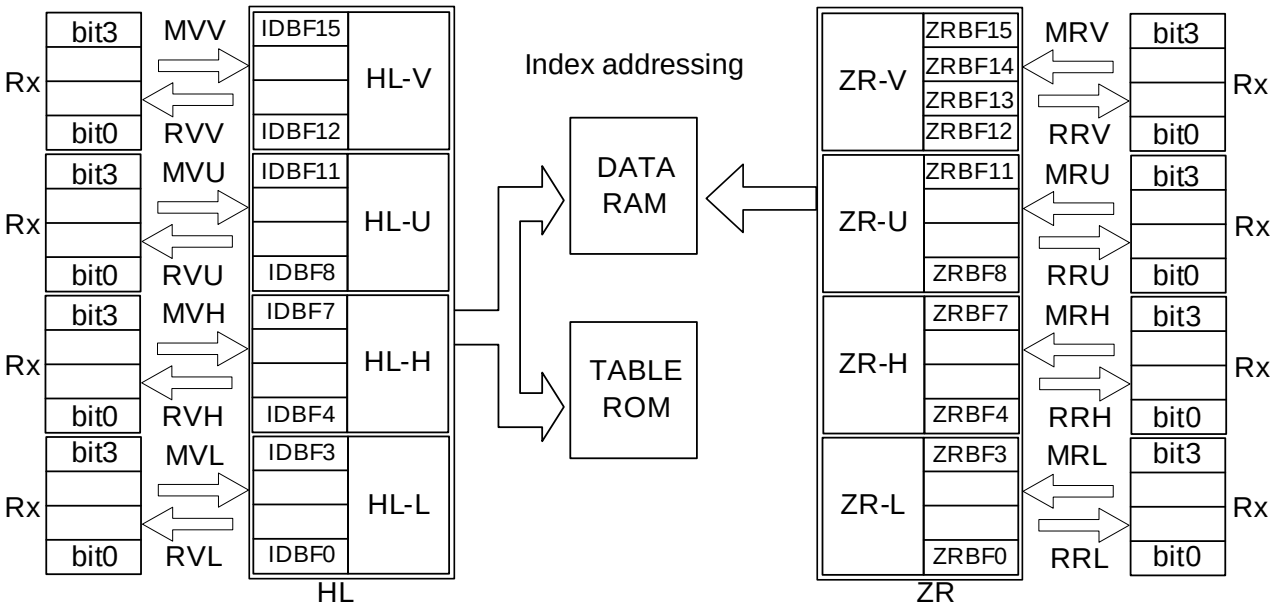
1-5. INDEX REGISTER (HL and ZR)

4BIT 系列 MCU 提供兩種 index register (HL and ZR)作為 data RAM 以及 table ROM 的索引定址之用(目前 ZR 僅由 TM89 系列提供)。

HL index register 是最多可支援 16-bit 的 register，可作為 data RAM 以及 table ROM 的索引定址之用(@HL)。HL index register 又可細分為 HL-V, HL-U, HL-H and HL-L 等四個 sub- register，IDBF15 ~ IDBF0 則代表 HL index register 中的各個位元。

ZR index register 是最多可支援 16-bit 的 register，而且只能作為 data RAM 的索引定址之用(@ZR)。ZR index register 又可細分為 ZR-V, ZR-U, ZR-H and ZR-L 等四個 sub-register，ZRBF15 ~ ZRBF0 則代表 ZR index register 中的各個位元。

下圖是這兩種 index register 與 sub-register 的關係圖：



1-5-1. HL INDEX REGISTER

HL index register 是由 HL-V, HL-U, HL-H and HL-L 等四個 sub-register 所組成的一個 16-bit 的 register(部分 MCU 規格中可能會小於 16-bit)。

HL index register 的內容值可以透過三種不同的存取指令來做資料的更新以及備份，分別是以 immediate data 直接寫入的指令，與 data RAM 內容值做資料存取的指令以及在指令結束後自動加 1/2/4 的指令。

1-5-1-1. 以 IMMEDIATE DATA 來更新 HL 內容值

TM89 系列&TM87ML28 HL index register 的內容值可以利用 SHLX 指令運算元中的 immediate data 同時更新 16 個 bit 的資料。

1-5-1-2. 透過 DATA RAM 來做 HL 內容值的更新或是備份(TM89 系列提供)

HL index register 的內容值可以一次 4 bits 或是 16 bits 的方式與 data RAM 的內容值做資料的更新或是備份。

在使用 4 bits 資料更新或是備份的方式時，執行 MVV, MVU, MVH and MVL 等指令可將 data RAM 的內容值寫入相對應的 HL-V, HL-U, HL-H and HL-L sub-register 中。

執行 RVV, RVU, RVH and RVL 等指令可將 HL-V, HL-U, HL-H and HL-L 等 sub-register 的內容值回存到指定的 data RAM 位址中。

下表說明 sub-register 與 data RAM 各個位元的對應關係：

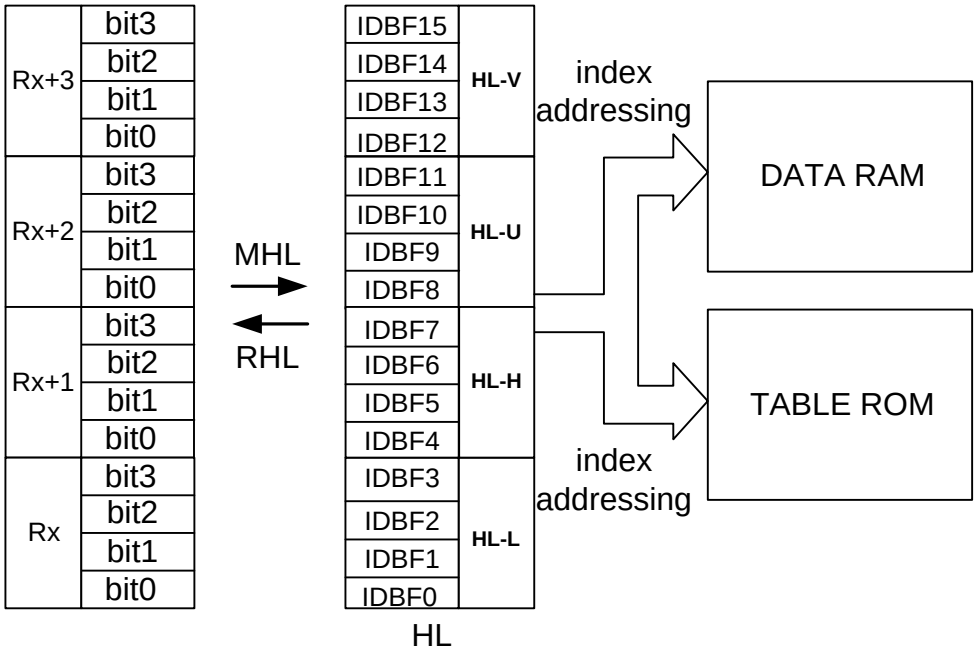
MVV Rx or RVV Rx				MVU Rx or RVU Rx				MVH Rx or RVH Rx				MVL Rx or RVL Rx			
(Rx)3	(Rx)2	(Rx)1	(Rx)0	(Rx)3	(Rx)2	(Rx)1	(Rx)0	(Rx)3	(Rx)2	(Rx)1	(Rx)0	(Rx)3	(Rx)2	(Rx)1	(Rx)0
@HL-V register				@HL-U register				@HL-H register				@HL-L register			
IDBF15	IDBF14	IDBF13	IDBF12	IDBF11	IDBF10	IDBF9	IDBF8	IDBF7	IDBF6	IDBF5	IDBF4	IDBF3	IDBF2	IDBF1	IDBF0

在使用 16 bits 的資料更新或是備份方式時，如 TM89 系列 MCU 會將特定的 64 個 data RAM 位址(0080 ~ 00BFH)區域規劃成 HL index register 的 16-bit 資料備份區，然後以每 4 個連續位址一組的方式區分成 16 組 資料備份單元。可以利用 MHL X 或是 RHL X 指令(X=0~FH)在四個 machine cycle 內完成一組資料備分單元與整個 HL index register 之間 16 bits 的資料傳輸動作 。

執行 MHL 指令可將四個連續位址的 data RAM 內容值同時寫入 16-bit 的 HL index register。執行 RHL 指令 可將 HL index register 的 16 bits 內容值同時儲存到四個連續位址的 data RAM 中。

在使用 MHL 以及 RHL 指令時，指令運算元 X 會指向四個連續的 data RAM 位址，其對應關係請參閱 1-7-2。

下表說明 HL index 與 data RAM 在以 16-bit 方式傳送資料時各個位元的對應關係：



1-5-1-3. 指令結束後自動增加 HL 內容值

4BIT 系列 MCU 的索引定址模式指令依各個 MCU 規格可以從 data RAM 以及 table ROM 中同時存取 1 個，2 個或是 4 個連續位址的內容值，所以在這些指令結束之後，有一些指令會自動將 HL index register 的內容值分別加 1、2 或是 4 的值。

1-5-1-4. HL index register 用於比較後遮蔽指令

HL index register 除了可以作為索引定址之用外，還可以做為 “比較後遮蔽” 指令的條件判斷式之用，相關的指令有 CPHL 以及 CPHLH(TM89 系列&TM87ML28 提供)。

CPHL X 指令可以設定一個 8-bit 的 immediate data (X)，當指令執行後 MCU 會比較@HL-H 和 @HL-L 的內容值與 X 的值是否相同。如果比較結果相同，MCU 在下一個指令週期的 時候無論原來的指令為何都會改成執行 NOP 指令(註解)；如果比較結果不相同，MCU 在下一個 指令週期的時候就會正常執行原來的指令。

註解: 不同的的指令會產生不同數目的 NOP 指令週期，

1. 單一指令週期的指令會產生一個 NOP 指令週期
2. 雙指令週期的 指令會產生兩個 NOP 指令週期
3. CAC X、JAC X 指令會產生一個 NOP 指令週期，NOP 指令執行結束後 PC 值會直接跳到 PC+X+2 的位址
4. ERX、ERY、ELZ 以及 CLPG 指令則不會產生 NOP 週期，仍會執行原指令

MCU 在執行 CPHL 以及下一個指令的指令週期中會自動禁止所有的中斷請求，直到這兩個指令週期結束為止。

下表說明 CPHL 指令中各個位元之間的比較關係：

CPHL X	X7	X6	X5	X4	X3	X2	X1	X0
Content of @HL	IDBF7	IDBF6	IDBF5	IDBF4	IDBF3	IDBF2	IDBF1	IDBF0
Sub-register	@HL-H				@HL-L			

Example:

```
        ... ... ... ... ... ...          ; @HL = 30h
        CPHL        $30h
        JMP         lable1       ; NOP will instead of this jump instruction
        JMP         lable2       ; this instruction will be executed and then jump to lable2
        ... ... ...
lable1:
        ... ... ... ..
lable2:
```

CPHLH 是一個兩個字元組長度的指令，同時也需要八個 machine cycle 才能完成。這個指令可以設定一個 16-bit 的 immediate data (X)，當指令執行後 MCU 會比較 HL index register 的內容值與 X 的值是否相同。如果比較結果相同，MCU 在下一個指令週期的時候無論原來的指令為何都會改成執行 NOP 指令(請參閱 1-5-1-4 的註解)；如果比較結果不相同，MCU 在下一個指令週期的時候就會正常執行原來的指令。

MCU 在執行 CPHLH 以及下一個指令的指令週期中會自動禁止所有的中斷請求，直到這兩個指令的指令週期皆結束為止。

下表說明 CPHLH 指令中各個位元之間的比較關係：

CPHLH	X15 ~ X12	X11 ~ X8	X7 ~ X4	X3 ~ X0
@HL	IDBF15 ~ IDBF12	IDBF11 ~ IDBF8	IDBF7 ~ IDBF4	IDBF3 ~IDBF0

Example:

```
        ... ... ... ... ... ...            ; @HL = 1234h
        CPHLH
        SETDAT      $1234h
        JMP         lable1          ; NOP will instead of this jump instruction
        JMP         lable2          ; this instruction will be executed and then jump to lable2
        ... ... ...
lable1:
        ... ... ...
lable2:
```

1-5-2. ZR INDEX REGISTER(TM89 系列提供)

ZR index register 是由@ ZR-V, @ ZR-U, @ ZR-H and @ ZR-L 等四個 sub-register 所組成的一個依 MCU 規格最大可支援 16-bit 的 register。

ZR index register 的內容值可以透過三種不同的存取指令來做資料的更新以及備份，分別是以 immediate data 直接寫入的指令，與 data RAM 內容值做資料存取的指令以及在指令結束後自動加 1/2/4 的指令。

1-5-2-1. 以 IMMEDIATE DATA 來更新 ZR 內容值

ZR index register 的內容值可以利用 SZRX 指令運算元中的 immediate data 同時更新 16 bits 的資料。

1-5-2-2. 透過 DATA RAM 來做 ZR 內容值的更新或是備份

ZR index register 的內容值可以一次 4 bits 或是 16 bits 的方式與 data RAM 的內容值做資料的更新或是備份。

在使用 4 bits 的資料更新或是備份方式時，執行 MRV, MRU, MRH 和 MRL 等指令可將 data RAM 的內容值寫入相對應的@ ZR -V, @ ZR -U, @ ZR –H 和@ ZR -L sub-register 中。

執行 RRV, RRU, RRH 和 RRL 等指令可將@ ZR -V, @ ZR -U, @ ZR –H 和@ ZR -L 等 sub-register 的內容值回存到指定的 data RAM 位址中。

下表說明 sub-register 與 data RAM 各個位元的對應關係：

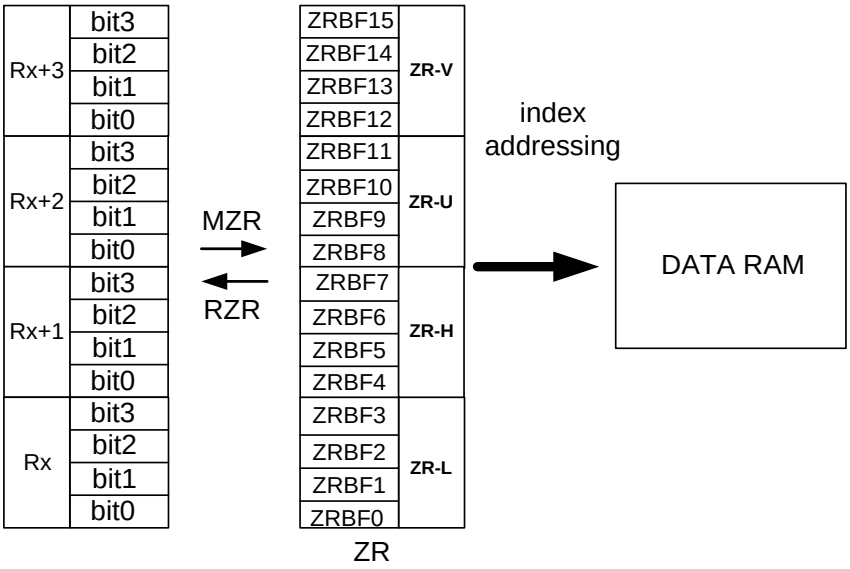
MRV Rx or RRV Rx				MRU Rx or RRU Rx				MRH Rx or RRH Rx				MRL Rx or RRL Rx			
Rx0	(Rx)3	(Rx)2	(Rx)1	(Rx)0	(Rx)3	(Rx)2	(Rx)1	(Rx)0	(Rx)3	(Rx)2	(Rx)1	(Rx)3	(Rx)2	(Rx)1	(Rx)0
@ZR-V register				@ZR-U register				@ZR-H register				@ZR-L register			
ZRBF12	ZRBF11	ZRBF10	ZRBF9	ZRBF8	ZRBF7	ZRBF6	ZRBF5	ZRBF4	ZRBF3	ZRBF2	ZRBF1	ZRBF3	ZRBF2	ZRBF1	ZRBF0

在使用 16 bits 的資料更新或是備份方式時，如 TM89 系列 MCU 會將特定的 64 個 data RAM 位址(00C0 ~ 00FFH)區域規劃成 ZR index register 的 16-bit 資料備份區，然後以每 4 個連續位址一組的方式區分成 16 組 資料備份單元。可以利用 MZR X 或是 RZR X 指令 (X=0~FH)在四個 machine cycle 內完成一組資料備分單元與整個 ZR index register 之間 16 bits 的資料傳輸動作 。

執行 MZR 指令可將四個連續位址的 data RAM 內容值同時寫入 16-bit 的 ZR index register。執行 RZR 指令可將 ZR index register 的 16 bits 內容值同時儲存到四個連續位址的 data RAM 中。

在使用 MZR 以及 RZR 指令時，指令運算元 X 會指向四個連續的 data RAM 位址，其對應關係*請參閱第 1-7-2 小節詳細說明*。

下表說明 ZR index 與 data RAM 在以 16-bit 方式傳送時各個位元的對應關係：



1-5-2-3. 指令結束後自動增加 ZR 內容值

4BIT 系列 MCU 的索引定址模式指令可以從 data RAM 中同時存取 1 個，2 個或是 4 個連續位址的內容值，所以在這些指令結束之後，有一些指令會自動將 ZR index register 的內容值分別加 1，2 或是 4 的值。

1-5-2-4. ZR index register 用於比較後遮蔽指令

ZR index register 除了可以作為索引定址之用外，還可以做為 “比較後遮蔽” 指令的條件判斷式之用，相關的指令有 CPZR 以及 CPZRH。

CPZR X 指令可以在設定一個 8-bit 的 immediate data (X)，當指令執行後 MCU 會比較@ZR-H 和@ZR-L 的內容值與 X 的值是否相同。如果比較結果相同，MCU 在下一個指令週期的時候無論原來的指令為何都會改成執行 NOP 指令(請參閱 1-5-1-4 的註解)；如果比較結果不相同，MCU 在下一個指令週期的時候就會正常執行原來的指令。

MCU 在執行 CPZR 以及下一個指令的指令週期中會自動禁止所有的中斷請求，直到這兩個指令週期結束為止。

下表說明 CPZR 指令中各個位元之間的比較關係：

CPZR X	X7	X6	X5	X4	X3	X2	X1	X0
Content of @ZR	ZRBF7	ZRBF6	ZRBF5	ZRBF4	ZRBF3	ZRBF2	ZRBF1	ZRBF0
Sub-register	@ZR-H				@ZR-L			

Example:

```
        ... ... ... ... ... ...          ; @ZR = 30h
        CPZR        $30h
        JMP         lable1       ; NOP will instead of this jump instruction
        JMP         lable2       ; this instruction will be executed and then jump to lable2
        ... ... ...
lable1:
        ... ... ... ..
lable2:
```

CPZRH 是一個兩個字元組長度的指令，同時也需要八個 machine cycle 才能完成。這個指令可以設定一個 13-bit 的 immediate data (X)，當指令執行後 MCU 會比較 ZR index register 的內容值與 X 的值是否相同。如果比較結果相同，MCU 在下一個指令週期的時候無論原來的指令為何都會改成執行 NOP 指令(請參閱 1-5-1-4 的註解)；如果比較結果不相同，MCU 在下一個指令 週期的時候就會正常執行原來的指令。

MCU 在執行 CPZRH 以及下一個指令的指令週期中會自動禁止所有的中斷請求，直到這兩個指令週期結束為止。

下表說明 CPZRH 指令中各個位元之間的比較關係：

CPZRH	X15 ~ 12	X11 ~ X8	X7 ~ X4	X3 ~ X0
@ZR	ZRBF15~12	ZRBF11~ZRBF8	ZRBF7~ZRBF4	ZRBF3~ZRBF0

Example:

```
        ... ... ... ... ... ...          ; @ZR = 1234h
        CPZRH
        SETDAT      $1234h
        JMP         lable1       ; NOP will instead of this jump instruction
        JMP         lable2       ; this instruction will be executed and then jump to lable2
        ... ... ...
lable1:
        ... ... ...
lable2:
```

1-6. STACK REGISTER (STACK)以及STACK POINTER (SP)

Stack register 會依據先進後出的規則將副程式呼叫指令的下一個程式位址或是進入中斷服務副程式之前的所執行的程式位址儲存起來，以便程式執行 RTS 指令之後可以返回原來的程式位址。

4BIT 系列 MCU 規劃了最多 16 層 * 16 bits 的 STACK register 來儲存程式位址，而且如 TM89 系列這些 stack register 與 data RAM 共用 64 個記憶體的位址(0200h~023Fh)。每一層的 stack register 需使用四個連續位址的 data RAM，可以儲存 16 bits 的程式位址。所有的 stack register 內容值也都可以利用 data RAM 的相關指令來做存取的動作。

Stack pointer 是一個 4-bit up/down counter，它是用來指示目前 stack register 的使用況。Stack pointer 的值如 TM89 系列可以利用 LSP 指令儲存到 data RAM 以及 AC 中。以下以 16 層 STACK 規格說明 Stack pointer 動作。

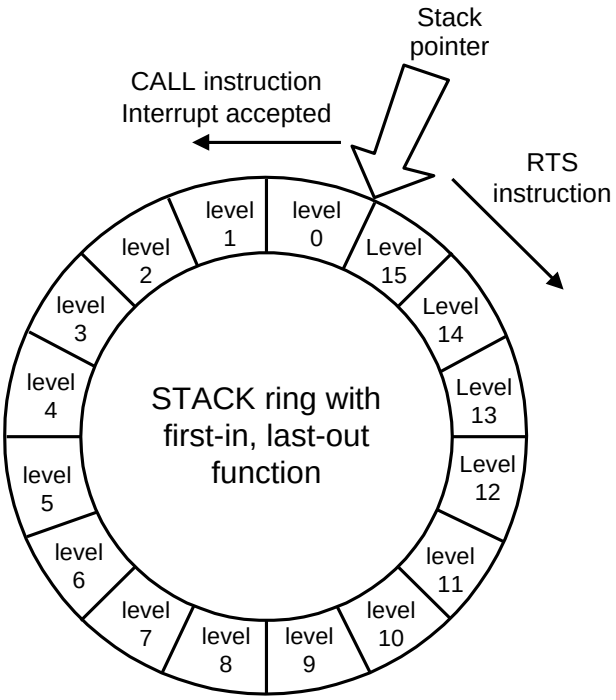
當 MCU 進入 RESET 狀態之後會將 stack pointer 的內容值清除為 0。

當 MCU 執行副程式呼叫指令或是中斷服務副程式的時候，會先將副程式呼叫指令的下一個程式位址或是進入中斷服務副程式之前的所執行的程式位址儲存到目前 stack pointer 所指向的 stack register，然後再將 stack pointer 加 1。如果目前的 stack pointer 已經是 0 (由 15 變為 0)，stack pointer 在加 1 之後就會發生 overflow 而且 stack pointer 會變成 1。

當 MCU 執行 RTS 指令後，會先將目前的 stack pointer 減 1，然後才將新的 stack pointer 所指向的 stack register 內容值回存到 program counter。如果目前的 stack pointer 已經是 0 (由 1 變為 0)，stack pointer 在減 1 之後就會發生 underflow 而且 stack pointer 會變成 15。

MCU 並沒有提供 stack pointer 發生 overlfow 或是 underflow 的 flag 可供辨識，因此當發生 stack overflow 時，原先存放在 stack register 0 的位址資料將會被新的程式位址覆蓋過去。如果發生 stack underflow 的時候，原先存放在 stack register 15 的位址資料則會被存回到 program counter 中。

下圖說明 stack register 的結構：



1-7. DATA RAM

4BIT 系列 MCU 的 Data RAM 是一個最大可定址到 64K addresses x 4 bits 的隨機存取記憶體(RAM)，作為程式存放資料之用。Data RAM 是一個多功能的記憶體，這些功能可以區分為，Stack register, LCD display memory, working register (Ry), data memory (Rx)以及 index register 的 16-bit 資料備份區。

Data RAM 可以使用兩種定址方式來存取資料，一是直接定址方式，另一個是索引定址方式。為了提升 MCU 的資料存取速度，如 TM89 系列 MCU 另外提供了三種資料長度的存取模式，分別是 4-bit、8-bit 以及 16-bit ，*詳細資料請參閱指令使用說明書。*

而為使直接定址的範圍可涵蓋 data RAM 全部的位址，如 TM89 系列 MCU 針對 LCD display memory、working register (Ry)、data memory (Rx)三種記憶體功能都能提供分頁的存取模式。

當 data RAM 以直接定址的方式做資料存取時，每個指令週期只能一次存取 4-bit 的資料。如果使用索引定址方式存取資料，每個指令週期則可以同時存取 4-bit、8-bit 或是 16-bit 的資料。

下面將詳述 data RAM 各種功能的使用方式以及分頁模式。

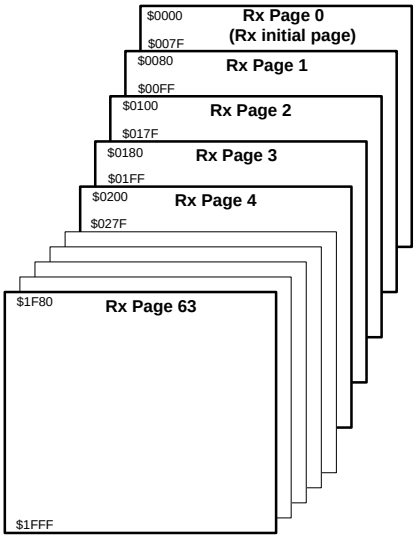
1-7-1. DATA MEMORY (Rx)

Data RAM 全部的位址都可以作為 data memory 使用時，可做為程式中一般資料的儲存。程式可利用兩種方式存取 data memory 的資料，一是直接定址，另一個是 索引定址。

在使用直接定址存取 Data memory 資料時一般 MCU Rx 只能涵蓋 data RAM 0000~007Fh 的位址，但是如 TM89 系列&TM87ML28 則可使用 Rx 的分頁模式涵蓋所有 data RAM 的位址，而在使用索引定址存取資料時則可以直接涵蓋所有 data RAM 的位址。

1-7-1-1. DATA MEMORY 的直接定址方式與 Rx page 的分頁模式

Rx page 的分頁模式以 8K x 4 bits data memory 為例是分割成 64 個 Rx page (page 0 ~ page 63)，每一個 Rx page 包含 128 addresses x 4 bits，其中 Rx page 0 稱為 Rx initial page。
右下圖說明 Rx page 分頁模式與 data memory 的關係：



當程式針對 Rx initial page 做直接定址時不需另外設定 Rx page 值就可以正確的存取資料，但是針對其他 Rx page 做直接定址時則必須先執行設定 Rx page 的指令(SRX 或是 ERX 指令)之後才能存取到正確的位址資料。

如 TM89 系列&TM87ML28 MCU 提供兩種設定 Rx page 的方式，一是由 compiler 程式自動設定，另一種是由使用者自行設定。

1. 由 compiler 程式自動設定 Rx page

程式指令中 data memory 的位址必須以絕對位址的方式來描述，compiler 程式在編譯過程中會自動檢查該位址是否落在 Rx initial page 的位址範圍內。如果不在 Rx initial page 位址範圍內時，compiler 程式會自動在該指令之前插入一個 SRX 指令來調整 Rx page 。

MCU 在執行 SRX 指令以及下一個位址指令時會自動禁止所有的中斷請求，以免造成 page 設定的錯誤。

Example:

Source file before compiling:

```
... ...
LDS     $0100     ; Rx memory in page 2
LDS     $0181     ; Rx memory in page 3
……
```

After compiling:

```
……
SRX     $2        ; set Rx memory page 2, inserted by compiler
LDS     $0100
SRX     $3        ; set Rx memory page 3, inserted by compiler
LDS     $0181
……
```

2. 由使用者自行設定 Rx page

由於程式在對 Rx initial page 以外的 data RAM 做直接定址存取前都要加入 SRX 指令，當存取次數頻繁時不僅會佔用程式的空間，也會影響程式的執行速度。除了建議改用索引定址方式之外，還可以利用 ERX 指令來自行設定 Rx Page 的方式來改善。

一旦 ERX 指令執行之後，程式中所有直接定址的指令都只能在該設定的 Rx page 中存取資料，而且 compiler 程式也不會在指令之前自動插入 SRX 指令。如果需要變更 Rx page 時只需重新執行 ERX 指令即可。

程式如要解除 ERX 的設定時，只要執行 CLPG 指令(X0=1)即可。因此 Compiler 程式編譯時遇到 CLPG(X0=1)指令之後就會恢復自動插入 SRX 指令的功能。

當程式執行任何一個 ERY，ERX 或是 ELZ 指令後，MCU 就會禁止所有的中斷服務，必須等到所有 Ry，Rx 以及 Lz page 的設定全部解除之後才會恢復中斷服務。

Example:

```
ERX   $2        ; set Rx memory in page 2
LDS   $0100     ; SRX didn't insert
LDS   $0101     ; SRX didn't insert
ERX   $3        ; update Rx memory to page 3
LDS   $0180     ; SRX didn't insert
LDS   $0181     ; SRX didn't insert
CLPG  $1
```

當使用者自行設定 Rx page 時，在程式的編寫上必須遵守下面兩條規範：

規範 1：程式中凡是被任何一個設定 memory page 的指令(ERX，ERY，ELZ)所涵蓋的 程式位址區間都被定義為 memory page 限制區間(memory page constrained segment)。
程式編寫中儘量不要讓任何 memory page 限制區間之外的 JMP/CALL 相關指令的目的位址落在 任何一個 memory page 限制區間內，而且 memory page 限制區間內的 JMP/CALL 相關指令的目的位址也不能超過與該指令前後位址相鄰最近的 ERY，ERX，ELZ 或是 CLPG 指令的位址範圍。
這樣可以避免程式因為 JMP/CALL 相關指令的執行而進入不同的 memory page 限制區間，導致 指令在存取 data RAM 資料時發生錯誤。

Example:

```
        NOP             ; label1, 2 can't set here!!
        ERX     $4
        JMP     label1
        NOP
label1: NOP
        NOP             ; label2 can't set here!!
        ERX     $5
        NOP             ; label1, 2 can't set here!!
        ERY     $27
        NOP             ; label1 can't set here!!
label2: NOP
        NOP
        JMP     label2
        ELZ     $5
        NOP             ; label1, 2 can't set here!!
        CLPG    $7
        NOP             ; label1, 2 can't set here!!
```

規範 2：在程式中 ERX 指令以及 CLPG 1 指令不可以直接編寫在 CPHL, CPZR, CPHLH 以及 CPZRH 等指令之後，因為在 HL/ZR 與設定值比較結果相同的情況下，這些指令會將下一個指令遮蔽，改以 NOP 來執行。

1-7-1-2. DATA MEMORY 的索引定址方式

Data memory 可以利用 HL 以及 ZR 兩種 index register 來進行索引定址的資料存取，這兩種 index register 的索引定址範圍都可以涵蓋整個 data memory 的位址。

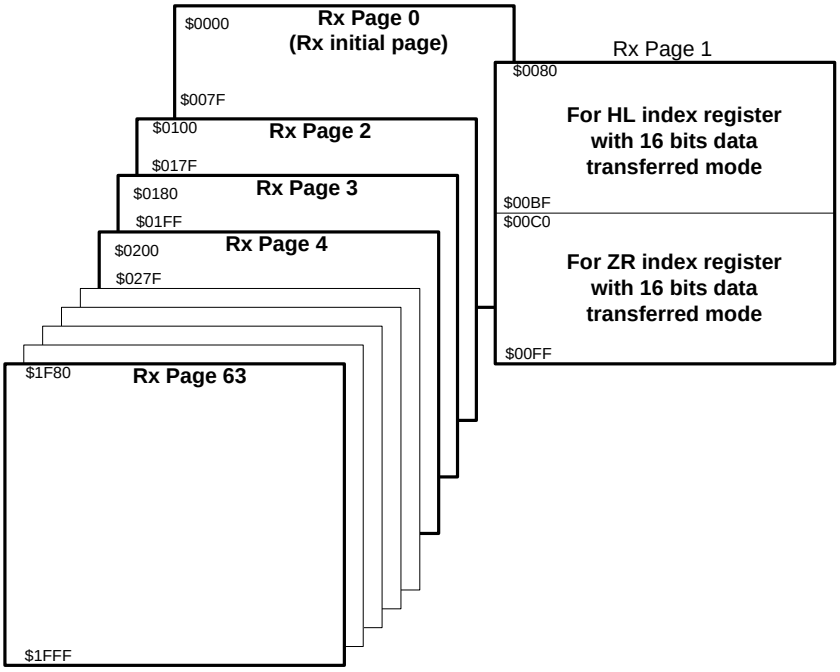
而在使用索引定址方式時，有些指令可在四個 machine cycle 中就能完成 4-bit, 8-bit 或是 16-bit 的資料存取，以提昇程式的執行效率。詳細的用法請參閱指令說明。

1-7-2. INDEX REGISTER 的 16-bit 資料備份區(TM89 系列提供)

當 MCU 提供將 Data RAM 的 128 個位址(0080h ~ 00FFh)規劃給 HL 以及 ZR index register 做為 16-bit 資料備份區之用，其中 0080h ~ 00BFh 的位址只提供給 HL index register 的資料存取之用

(MHL 以及 RHL 指令)，00C0h ~ 00FFh 的位址只提供給 ZR index register 的資料存取之用(MZR 以及 RZR 指令)。

下圖說明 index register 的 16-bit 資料備份區在 data RAM 位址中的對應關係。



當程式使用 index register 的 16-bit 資料備份區時，在四個 machine cycle 內便可將 HL 或是 ZR index register 的內容值同時寫入四個連續的 data RAM 位址上，或是將四個連續的 data RAM 位址的內容值存回 HL 或是 ZR index register。

下表說明 index register 與四個連續的 data RAM 位址中各個位元的相對關係：

HL backup unit	HL index register (IDBF15 ~ IDBF0)			
Operand X	IDBF15~12	IDBF11~8	IDBF7~4	IDBF3~0
0	$0083	$0082	$0081	$0080
1	$0087	$0086	$0085	$0084
2	$008B	$008A	$0089	$0088
3	$008F	$008E	$008D	$008C
4	$0093	$0092	$0091	$0090
5	$0097	$0096	$0095	$0094
6	$009B	$009A	$0099	$0098
7	$009F	$009E	$009D	$009C
8	$00A3	$00A2	$00A1	$00A0
9	$00A7	$00A6	$00A5	$00A4
A	$00AB	$00AA	$00A9	$00A8
B	$00AF	$00AE	$00AD	$00AC
C	$00B3	$00B2	$00B1	$00B0
D	$00B7	$00B6	$00B5	$00B4

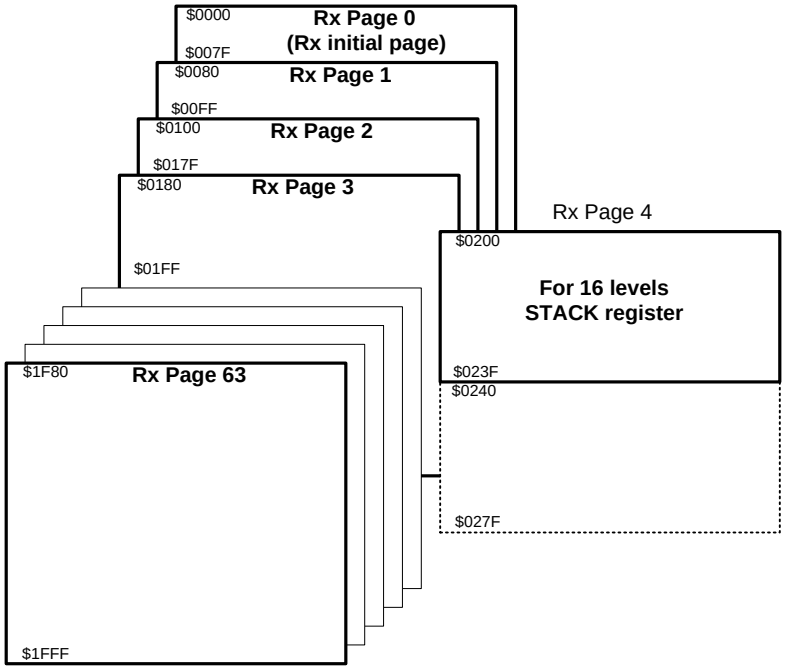
HL backup unit	HL index register (IDBF15 ~ IDBF0)			
Operand X	IDBF15~12	IDBF11~8	IDBF7~4	IDBF3~0
E	$00BB	$00BA	$00B9	$00B8
F	$00BF	$00BE	$00BD	$00BC

ZR backup unit	ZR index register(ZRBF15 ~ ZRBF0)			
Opernad X	ZRBF15~12	ZRBF11~8	ZRBF7~4	ZRBF3~0
0	$00C3	$00C2	$00C1	$00C0
1	$00C7	$00C6	$00C5	$00C4
2	$00CB	$00CA	$00C9	$00C8
3	$00CF	$00CE	$00CD	$00CC
4	$00D3	$00D2	$00D1	$00D0
5	$00D7	$00D6	$00D5	$00D4
6	$00DB	$00DA	$00D9	$00D8
7	$00DF	$00DE	$00DD	$00DC
8	$00E3	$00E2	$00E1	$00E0
9	$00E7	$00E6	$00E5	$00E4
A	$00EB	$00EA	$00E9	$00E8
B	$00EF	$00EE	$00ED	$00EC
C	$00F3	$00F2	$00F1	$00F0
D	$00F7	$00F6	$00F5	$00F4
E	$00FB	$00FA	$00F9	$00F8
F	$00FF	$00FE	$00FD	$00FC

1-7-3. STACK REGISTER

TM89 系列提供將 Data RAM 的 64 個位址(0200h ~ 023Fh)規劃給 stack register 使用，每一層的 stack register 都會佔用四個連續的 data RAM 位址。*stack register 的使用方式請參考 1-6 小節詳細說明。*

下圖說明 stack register 在 data RAM 位址中的對應關係：

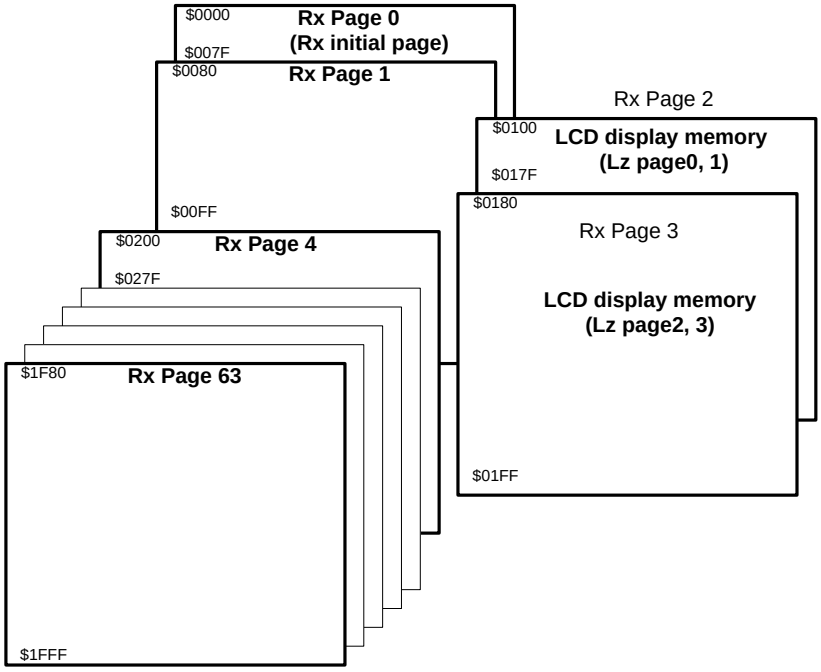


下表說明每一層 stack register 中所儲存的 PC 值與四個連續位址的 data RAM 內容值的對應關係：

STACK pointer	PC15~PC12	PC11~PC9	PC8~PC4	PC3~PC0
0	$0203	$0202	$0201	$0200
1	$0207	$0206	$0205	$0204
2	$020B	$020A	$0209	$0208
3	$020F	$021E	$021D	$020C
4	$0213	$0212	$0211	$0210
5	$0217	$0216	$0215	$0214
6	$021B	$021A	$0219	$0218
7	$021F	$021E	$021D	$021C
8	$0223	$0222	$0221	$0220
9	$0227	$0226	$0225	$0224
A	$022B	$022A	$0229	$0228
B	$022F	$022E	$022D	$022C
C	$0233	$0232	$0231	$0230
D	$0237	$0236	$0235	$0234
E	$023B	$023A	$0239	$0238
F	$023F	$023E	$023D	$023C

1-7-4. LCD DISPLAY MEMORY 與 Lz PAGE 的分頁方式

TM89 系列提供將 Data RAM 的 256 個位址(0100h ~ 01FFh)規劃給 LCD display memory 使用，下圖說明 LCD display memory 在 data RAM 位址中的對應關係：



1-7-4-1. 與 data memory(Rx)相同的存取方式

TM89 系列 LCD display memory 與其他 data RAM 一樣，可以視為 data memory (Rx)或是 working register (Ry)的一部分來做資料的存取，因此同樣可使用任何存取 data RAM 指令對 LCD display memory 做存取。

1-7-4-2. 以 Lz 直接定址方式存取 LCD display memory

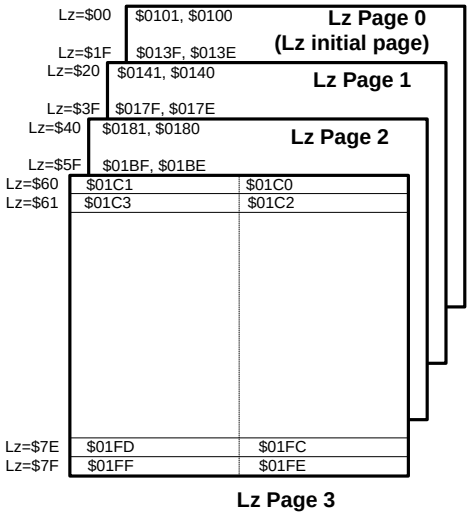
4BIT 系列 MCU 提供 Lz 直接定址方式可以藉由 DBUSA~H 對應至七節碼的 8bits 格式寫入 LCD display memory，若 MCU 提供 LCD display memory 為彈性的 PLA 架構，便可藉由任意設定每個 SEG 每個 duty 的 Lz & DBUS 值以便於符合 LCD 玻璃對七節碼的設定，而如 TM89 系列 MCU 雖然為固定 LCD RAM 架構，但為了提供與 TM87 系列 MCU 相容的 LCD 相關指令，所以在 LCD display memory 的存取功能上仍保留了 Lz 直接定址 (PSTB)以及可將八個 bit 的資料 DBUS) 同時寫入 LCD display memory 的特性。因此 Lz 直接定址方式將 LCD display memory 定義為 128 個 Lz addresses x 8 bits，每一個 LCD display memory 的 Lz address 是由兩個連續的 data RAM 位址所組成。

但是 Lz 的直接定址範圍只限於 LCD display memory 的 256 個位址，無法定址到其他 data RAM 的位址。LCT，LCB，LCP，LCD，LCDH 等相關指令可以利用@ZR 的索引定址方式寫入所有 data RAM @ZR 的索引位址。此外當 LCD 相關指令在使用 Lz 直接定址時只能將資料寫入 LCD display memory，無法直接以 Lz 直接定址的方式讀取 LCD display memory 的資料。

1-7-4-3. Lz PAGE 的分頁方式

Lz page 的分頁模式是將 LCD display memory 分割成 4 個 Lz page(page 0 ~ page 3)，每一個 page 包含 32 addresses x 8 bits，Lz page 0 是 Lz initial page。

下圖說明 LCD display memory 的架構與 Lz page 的分頁方式，圖中在方格外面的 Lz 代表 LCD display memory 的 Lz 位址，而在方格內部的位址代表該 Lz 位址所對應的 data RAM 實際位址。



下表說明 LCD display memory 與兩個連續位址的 data RAM 之間每個位元的對應關係：

Content in Lz address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Content in Data memory	Bit3	Bit2	Bit1	Bit0	Bit3	Bit2	Bit1	Bit0
Address corresponding to data memory	(Lz * 2) + 101h				(Lz * 2) + 100h			

當程式針對 Lz initial page 做直接定址時不需另外設定 Lz page 值，但是針對其他 Lz page 做直接定址時則必須先執行設定 Lz page 的指令(SLZ 或是 ELZ 指令)之後才能存取到正確的位址資料。

如 TM89 系列 MCU 提供兩種設定 Lz page 的方式，一是由 TM89 ICE 的 compiler program 自動設定，另一種 是由使用者自行設定。

1. 由 compiler 程式自動設定 Lz page

程式指令中 LCD display memory 的 Lz 位址必須以絕對位址的方式來描述，compiler 程式在編譯過程中會自動檢查該位址是否落在 Lz initial page 的位址範圍內。如果不在 Lz initial page 位址範圍時，compiler 程式會自動在該指令之前插入一個 SLZ 指令來調整 Lz page。

MCU 在執行 SLZ 指令以及下一個位址指令時會自動禁止所有的中斷請求，以免造成 page 設定的錯誤。

For example,

Source file before compiling:

```
... ...
LCB     $23, @HL        ; Rx memory in page 1
LCB     $51, @HL        ; Rx memory in page 2
……
```

After compiling:

```
……
```

```
SLZ     $1              ; set Rx memory page 1,
                        ; inserted by compiler.
LCB     $23, @HL
SLZ     $2              ; set Rx memory page 2, insert by compiler.
LCB     $51, @HL
……
```

2. 由使用者自行設定 Lz page

由於程式在對 Lz initial page 以外的 LCD display memory 做直接定址存取前都要加入 SLZ 指令，當存取次數頻繁時不僅會佔用程式的空間，也會影響程式的執行速度。除了 建議改用索引定址方式之外，還可以利用 ELZ 指令來自行設定 Lz Page 的方式來改善。

一旦 ELZ 指令執行之後，程式中所有 Lz 直接定址的指令都只能在該設定的 Lz page 中 存取資料，而且 compiler 程式也不會在指令之前自動插入 SLZ 指令。如果需要變更 Lz page 時只需重新執行 ELZ 指令即可。

程式如要解除 ELZ 的設定時，只要執行 CLPG 指令(X2=1)即可。因此 Compiler 程式編譯時遇到 CLPG(X2=1)指令之後就會恢復自動插入 SLZ 指令的功能。

為確保指令不會定址到錯誤的 Lz page，Compilier 在編譯過程中會檢查程式中每一個 ELZ 指令到下一個 ELZ 指令或是 CLPG(X2=1)指令位址之間的所有指令所使用的 data RAM 位址是否都在同一個 Lz Page 內，否則就會送出錯誤訊息。

當程式執行任何一個 ERY，ERX 或是 ELZ 指令後，MCU 就會禁止所有的中斷服務，必須等到所有 Ry，Rx 以及 Lz page 的設定全部解除之後才會恢復中斷服務。

For example:

```
ELZ     $1              ; set Lz memory in page 1
LCB     $23, @HL        ; SLZ didn't insert
LCB     $31, @HL        ; SLZ didn't insert
ELZ     $2              ; update Lz memory to page 2
LCB     $41, @HL        ; SLZ didn't insert
LCB     $52, @HL        ; SLZ didn't insert
CLPG    $4
```

使用者自行設定 Lz page 時，在程式的編寫上必須遵守下面兩條規範：

規範 1：程式中凡是被任何一個設定 memory page 的指令(ERX，ERY，ELZ)所涵蓋的 程式位址區間都被定義為 memory page 限制區間(memory page constrained segment)。

程式編寫中儘量不要讓任何 memory page 限制區間之外的 JMP/CALL 相關指令的目的位址落在任何一個 memory page 限制區間內，而且 memory page 限制 區間內的 JMP/CALL 相關指令的目的位址也不能超過與該指令前後位址相鄰最近的 ERY，ERX，ELZ 或是 CLPG 指令的位址範圍。

這樣可以避免程式因為 JMP/CALL 相關指令的執行而進入不同的 memory page 限制區間，導致 指令在存取 data RAM 資料時發生錯誤。

For example,

```
NOP             ; label1, 2 can't set here!!
ERX     $4
JMP     label1
```

```
            NOP
label1:     NOP
            NOP                     ; label2 can't set here!!
            ERX         $5
            NOP                     ; label1, 2 can't set here!!
            ERY         $27
            NOP                     ; label1 can't set here!!

label2:     NOP
            NOP
            JMP         label2
            ELZ         $5
            NOP                     ; label1, 2 can't set here!!
            CLPG        $7
            NOP                     ; label1, 2 can't set here!!
```

規範 2：在程式中 ELZ 指令以及 CLPG 4 指令不可以直接編寫在 CPHL，CPZR，CPHLH 以及 CPZRH 等指令之後，因為在 HL/ZR 與設定值比較結果相同的情況下，這些指令會將下一個指令遮蔽，改以 NOP 來執行。

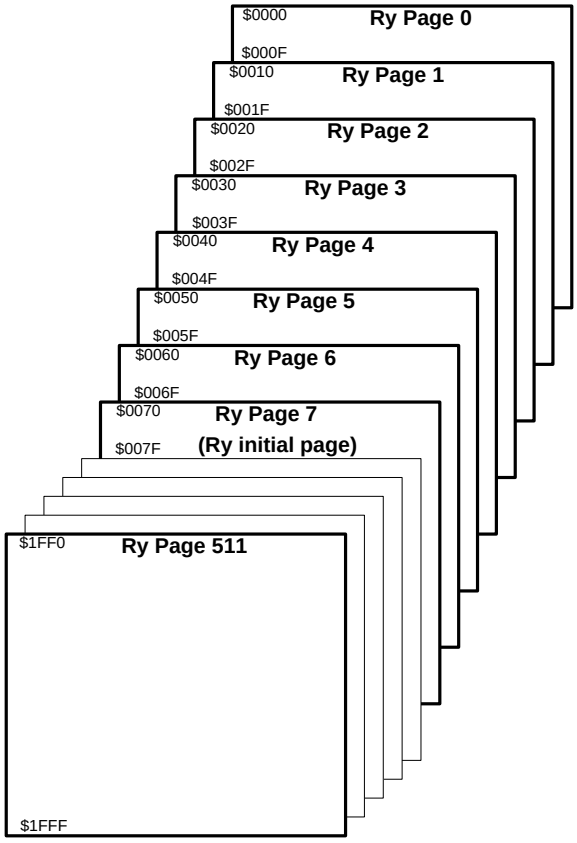
1-7-5. WORKING REGISTER (Ry)以及 Ry page 分頁模式

當程式需要同時在兩個 data RAM 位址之間使用直接定址方式做資料的存取時，或是需要將 immediate data 使用直接定址方式存入 data RAM 時都必須使用 working register(Ry)的定址方式來指定其中的一個記憶體位址。

一般 MCU Ry 只能涵蓋 data RAM 0070~007Fh 的位址，但是如 TM89 系列&TM87ML28 則可使用 Ry 的分頁模式涵蓋所有 data RAM 的位址都可以作為 working register 使用。

Ry page 的分頁模式將 working register 劃分成 512 個 Ry page(page 0 ~ page 511)，每一個 page 包含 16 addresses x 4 bits，Py Page 7 是 Ry initial page。

下圖說明 Ry page 分頁模式與 working register 的關係：



當程式針對 Ry initial page 做直接定址時不需另外設定 Ry page 值即可存取正確的位址資料，但是針對其他 Ry page 做直接定址時則必須先執行設定 Ry page 的指令(SRY 或是 ERY 指令)之後才能存取到正確的位址資料。

如 TM89 系列&TM87ML28 提供兩種設定 Ry page 的方式，一是由 compiler 程式自動設定，另一種是由使用者自行設定。

1. 由 compiler 程式自動設定 Ry page

程式指令中 working register 的位址必須以絕對位址的方式來描述，Compiler 程式在編譯過程中會自動檢查該位址是否落在 Ry initial page 的位址範圍內。如果不在 Ry initial page 位址範圍時，compiler 程式會自動在該指令之前插入一個 SRY 指令來調整 Ry page。

MCU 在執行 SRY 指令以及下一個位址指令時會自動禁止所有的中斷請求，以免造成 page 設定的錯誤。

For example,

Source file before compiling:

```
... ...
LCT     @ZR, $0070      ; Rx memory in page 7
LCT     @ZR, $0020      ; Rx memory in page 2
......
```

After compiling:

```
......
LCT     @ZR, $0070      ; Ry initial page
SRY     $2              ; set Ry memory page 2, insert by compiler
LCT     @ZR, $0020
```

……

2. 由使用者自行設定 Ry page

由於程式在對 Ry initial page 以外的 working register 做直接定址存取前都要加入 SRY 指令，當存取次數頻繁時不僅會佔用程式的空間，也會影響程式的執行速度。除了建議改用索引定址方式之外，還可以利用 ERY 指令來自行設定 Ry Page 的方式來改善。

一旦 ERY 指令執行之後，程式中所有 Ry 直接定址的指令都只能在該設定的 Ry page 中存取資料，而且 compiler 程式也不會在指令之前自動插入 SRY 指令。如果需要變更 Ry page 時只需重新執行 ERY 指令即可。

程式如要解除 ERY 的設定時，只要執行 CLPG 指令(X1=1)即可。因此 Compiler 程式編譯時遇到 CLPG(X1=1)指令之後就會恢復自動插入 SLZ 指令的功能。

為確保指令不會定址到錯誤的 Ry page，Compilier 在編譯過程中會檢查程式中每一個 ERY 指令到下一個 ERY 指令或是 CLPG (X1=1)指令位址之間的所有指令所使用的 data RAM 位址是否都在同一個 Ry Page 內，否則就會送出錯誤訊息。

當程式執行任何一個 ERY，ERX 或是 ELZ 指令後，MCU 就會禁止所有的中斷服務，必須等到所有 Ry，Rx 以及 Lz page 的設定全部解除之後才會恢復中斷服務。

For example,

```
ERY     $2          ; set Ry memory in page 2
LCT     @ZR, $0020  ; SRY didn't insert
LCT     @ZR, $002A  ; SRY didn't insert
ERY     $3          ; update Ry memory to page 3
LCT     @ZR, $0039  ; SRY didn't insert
LCT     @ZR, $003D  ; SRY didn't insert
CLPG    $2
```

使用者自行設定 Ry page 時，在程式的編寫上必須遵守下面兩條規範：

規範 1：程式中凡是被任何一個設定 memory page 的指令(ERX，ERY，ELZ)所涵蓋的 程式位址區間都被定義為 memory page 限制區間(memory page constrained segment)。

程式編寫中儘量不要讓任何 memory page 限制區間之外的 JMP/CALL 相關指令的目的位址落在任何一個 memory page 限制區間內，而且 memory page 限制區間內的 JMP/CALL 相關指令的目的位址也不能超過與該指令前後位址相鄰最近的 ERY、ERX、ELZ 或是 CLPG 指令的位址範圍。

這樣可以避免程式因為 JMP/CALL 相關指令的執行而進入不同的 memory page 限制區間，導致指令在存取 data RAM 資料時發生錯誤。

For example,

```
        NOP                 ; label1, 2 can't set here!!
        ERX     $4
        JMP     label1
        NOP
label1:NOP
        NOP                 ; label2 can't set here!!
        ERX     $5
        NOP                 ; label1, 2 can't set here!!
        ERY     $27
        NOP                 ; label1 can't set here!!
```

```
label2: NOP
        NOP
        JMP     label2
        ELZ     $5
        NOP                 ; label1, 2 can't set here!!
        CLPG    $7
        NOP                 ; label1, 2 can't set here!!
```

規範 2：在程式中 ERY 指令以及 CLPG(X1=1)指令不可以直接編寫在 CPHL，CPZR，CPHLH 以及 CPZRH 等指令之後，因為在 HL/ZR 與設定值比較結果相同的情況下，這些指令會將下一個指令遮蔽，改以 NOP 來執行。

1-8. ACCUMULATOR (AC)

Accumulator (AC)是將 ALU 與其他 register 或是 data RAM 連接起來的一個重要 register，也是資料傳輸時必經的一個 register。

1-9. Arithmetic and Logic Unit (ALU)

ALU 負責邏輯與算術運算。TM89 系列 MCU 的 ALU 可以執行下面這些運算全部功能：

二進位/十進位加減法	(INC, DEC, ADC(M), SBC(M), ADD(M), SUB(M), ADN, ADCI, SBCI, ADDI, SUBI, ADNI)
邏輯運算	(AND, EOR, OR, ANDI, EORI, ORI)
位移運算	(SR0, SR1, SL0, SL1)
旋轉運算	(RRC, RLC)
條件判斷運算	(JB0, JB1, JB2, JB3, JC, JNC, JZ, JNZ, CAC, and JAC)
二進位轉十進位換算	(DAA, DAS)
二進位/十進位乘法	(MULH, MULD)

在程式中使用二進位或是十進位的算術運算時必須特別注意，所有的運算數值都必須先轉換成一致的進位格式，否則會得到錯誤的運算結果。

1-9-1. 二進位加減法運算結果換算成十進位的格式(BCD)

如果程式中需要使用二進位的加減法運算指令來進行十進位的數值運算時，其二進位的算術運算結果必須在換算成十進位的格式才能得到正確的結果。

1-9-1-1.二進位加法運算結果的二進位轉十進位換算

DAA、DAA*、DAA#等指令可以將二進位加法運算的結果換算成十進位的格式。

下表說明二進位加法的運算結果換算成十進位的方法：

AC data before DAA execution	CF data before DAA execution	AC data after DAA execution	CF data after DAA execution
0 ≤ AC ≤ 9	CF = 0	no change	no change
A ≤ AC ≤ F	CF = 0	AC= AC+ 6	CF = 1
0 ≤ AC ≤ 3	CF = 1	AC= AC+ 6	no change

範例：利用 DAA*指令將二進位加法運算的結果換算十進位的格式，

```
LDS     $0010, 9    ; 將 “9” 寫入 data memory 位址 10h。
LDS     $0011, 1    ; 將 “1” 寫入 data memory 位址 11h 以及 AC。
ADD*    $0010       ; 將 data memory 位址 10h 的內容值與 AC 的內容值
                    ; 作加法運算，運算結果 “A” 寫回 data memory
                    ; 位址 10h 以及 AC。
DAA*    $0010       ; 將 data memory 位址 10h 的內容值換算成十進位，
                    ; 換算的結果 “0” 寫回 data memory 位址 10h，
                    ; CF 進位成 “1” 。
```

1-9-1-2. 二進位減法運算結果的二進位轉十進位換算

DAS、DAS*、DAS#等指令可以將二進位減法運算的結果換算成十進位的格式。

下表說明二進位減法的運算結果換算成十進位的方法：

AC data before DAS execution	CF data before DAS execution	AC data after DAS execution	CF data after DAS execution
0 ≤ AC ≤ 9	CF = 1	no change	no change
6 ≤ AC ≤ F	CF = 0	AC= AC+ A	no change

範例：利用 DAS*指令將二進位減法運算的結果換算十進位的格式，

```
LDS     $0010, 1    ; 將 "1" 寫入 data memory 位址 10h。
LDS     $0011, 2    ; 將 "2" 寫入 data memory 位址 11h 以及 AC。
SF      1           ; 將 CF 設為 1，表示沒有減法借位。
SUB*    $0010       ; 將 data memory 位址 10h 的內容值與 AC 的內容值作
                    ; 減法運算，運算結果 "F" 寫回 data memory
                    ; 位址 10h 以及 AC，CF 被設為 0 (發生借位)。
DAS*    $0010       ; 將 data memory 位址 10h 的內容值換算成十進位，
                    ; 換算的結果 "9" 寫回 data memory 位址 10h，
                    ; CF 進位成 "0" 。
```

1-9-2. 十進位的加減法運算指令

如 TM89 系列 MCU 在索引定址模式下也提供十進位的加減法運算指令來進行十進位的數值運算，其運算結果直接就是十進位的格式。

1-9-2-1. 十進位的加法運算指令

ADC(M,*,#) @HL/ZR,DA、ADD(M,*,#) @HL/ZR,DA 等十進位加法指令可以直接產生十進位的運算結果。

範例：十進位加法指令直接產生十進位的結果，

```
SHLX
setdat  $0010       ; 將 10h 寫入 HL index register。
LDS     @HL, 9      ; 將 "9" 寫入 data memory 位址 10h。
LDS     $0011, 1    ; 將 "1" 寫入 data memory 位址 11h 以及 AC。
ADD*    @HL,DA      ; 將 data memory 位址 10h 的內容值與 AC 的內容值作
                    ; 加法運算，運算結果 "0" 寫回 data memory
                    ; 位址 10h，CF 進位成 "1" 。
```

1-9-2-2. 十進位的減法運算指令

SBC(M,*,#) @HL/ZR,DA 、 SUB(M,*,#) @HL/ZR,DA 等十進位減法指令可以直接產生十進位的運算結果。

範例：十進位減法指令直接產生十進位的結果

```
SZRX
setdat  $0010       ; 將 10h 寫入 ZR index register。
```

```
LDS     @ZR, 1      ; 將 "1" 寫入 data memory 位址 10h。
LDS     $0011, 2    ; 將 "2" 寫入 data memory 位址 11h 以及 AC。
SF      1           ; 將 CF 設為 1，表示沒有減法借位。
SUB*    @ZR,DA      ; 將 data memory 位址 10h 的內容值與 AC 的內容值作
                    ; 減法運算，運算結果 "9" 寫回 data memory
                    ; 位址 10h，CF 進位成 "0" 。
```

1-9-3. 乘法運算

TM89 系列 MCU 提供一個 4 bits * 4 bits = 8 bits 的乘法運算功能，所有的乘法指令都可以在一個指令周期內完成。

下面這些乘法指令可以直接處理十進位的乘法運算：

MULD Rx, MULD @HL, MULD# @HL, MULD @ZR and MULD# @ZR

下面這些乘法指令可以處理二進位的乘法運算：

MULH Rx, MULH @HL, MULH# @HL, MULH @ZR and MULH# @ZR

在乘法運算中，被乘數必須存放在 data RAM 中，乘數必須放在 MUI register 中。而乘法的運算結果的高位元的 4-bit 資料會被存放到 MU register，低位元的 4-bit 資料則會存放到 AC。

(data memory) x (MUI) = (MU, AC)

程式中可以利用 SMUI 指令將 data RAM 的資料儲存到 MUI register(乘數)；而 MMH 指令則可將 MU register 的內容值(乘積)存到 data RAM 以及 AC。

為提升程式的乘法運算效率，有些加減法指令可以直接針對乘法運算的結果(MU register 的內容值)做運算，例如 ADCM, ADDM, SBCM, 以及 SUBM 等。

Example 1: (9 x 6 二進位乘法運算)

```
LDS    $0010, $9    ; 將 "9" 寫入 data memory 位址 10h。
LDS    $0011, $6    ; 將 "6" 寫入 data memory 位址 11h。
SMUI   $0010        ; 將 data memory 位址 10h 的內容值寫入 MUI。
MULH   $0011        ; 執行二進位乘法運算，運算結果存到 MU=3h 以及 AC=6h。
MMH    $0012        ; 將 MU 的內容值 "3" 寫入 data memory 位址 12h。
```

Example 2: (9x6 十進位乘法運算)

```
LDS    $0010, $9    ; 將 "9" 寫入 data memory 位址 10h。
LDS    $0011, $6    ; 將 "6" 寫入 data memory 位址 11h。
SMUI   $0010        ; 將 data memory 位址 10h 的內容值寫入 MUI。
MULD   $0011        ; 執行十進位乘法運算，運算結果存到 MU=5 以及 AC=4。
MMH    $0012        ; 將 MU 的內容值 "5" 寫入 data memory 位址 12h。
```

Example 3: (89 x 67 = 5963 的十進位乘法運算)

在這個範例中 Rx 代表 data memory 的位址。下面先以手算的乘法式子說明乘法運算的程序。

```
        89          ; 被乘數      -> 存放在 Rx = 11h,10h
x       67          ; 乘數        -> 存放在 Rx = 21h,20h
---------------
      0000          ; Step0: 乘法運算的結果放在 Rx = 33h~30h，
```

```
                                ; 並將預設值設為 0。
        +      63
        ------------
               63               ; Step1
        +     56
        ------------
              623               ; Step2
        +     54
        ------------
             1163               ; Step3
        +    48
        ------------
             5963               ; Step4: 此為乘法運算的最後結果。
```

下面是乘法運算的範例程式：

```
        SHLX
        setdat  $0010           ; HL = 10h
        RHL             $0      ; 將 HL 內容值(10h)存至 HL 備份區 0 = 83h~80h，
                                ; HL 備份單元 0 代表正在運算中的被乘數的存放位址。
        SZRX
        setdat  $0020           ; ZR =  20h
        LDS8    @HL,    $89     ; 將被乘數 89 存至 Rx = 11h, 10h
        LDS8    @ZR,    $67     ; 將乘數 67 存至 Rx = 21h, 20h
        SMUI# @ZR               ; 將乘數的個位數(7)寫入 MUI, ZR = 21h
        RZR             $0      ; 將 ZR 內容值(21h) 存至 ZR 備份單元 0=C3h~C0h，
                                ; ZR 備份單元 0 代表下一個準備運算的乘數的存放位址。
        SZRX
        setdat  $0030           ; ZR = 30h，將 Rx:33h~30h 指定為乘法運算結果。
        LDSH    @ZR
        setdat  $0000           ; 將 Rx:33h~30h 寫入 0 (對應至手算式子的 Step0)
        MULD# @HL               ; Rx:10h(9) x MUI(7) = MU=6 , AC=3; HL=11h.
        RZR             $1      ; 將 ZR 內容值(30h) 存至 ZR 備份單元 1=C7h~C4h，
                                ; ZR 備份區 1 表示乘法運算中已經完成運算的乘積的
                                ; 存放位址。
        ADD*# @ZR,DA            ; Rx:30h(0) + AC(3) --> CF=0, Rx:30h=3; ZR=31h.
        RZR             $2      ; 將 ZR 內容值(31h) 存至 ZR 備份單元 2=CBh~C8h，
                                ; ZR 備份單元 2 表示乘法運算中下一個準備計算的乘積的
                                ; 存放位址。
        ADCM*# @ZR,DA           ; Rx:31h(0) + MU(6) + CF(0) --> CF=0, Rx:31h=6;
                                ; ZR =32h
        JNC     lab1a           ; Jump to “lab1a” (對應至手算式子的 Step1)
        INC*#   @ZR
        …..

lab1a:  MULD# @HL               ; Rx:11h(8) x MUI(7) --> MU=5 , AC=6; HL=12h.
        MZR             $2      ; 將 ZR 備份單元 2=CBh~C8h(0031H)的值載入 ZR
        ADD*# @ZR,DA            ; Rx:31h(6) + AC(6) --> CF=1,Rx:31h=2; ZR =32h.
        RZR             $2      ; 將 ZR 內容值(32h) 存至 ZR 備份單元 2=CBh~C8h
```

```
        ADCM*# @ZR,DA     ; Rx:32h(0)+MU(5)+CF(1) --> CF=0, Rx:32h=6;
                          ; ZR =33h.
        JNC    lab2a      ; Jump to “lab2a”. (對應至手算式子的 Step2)
        INC*#  @ZR
        …..

lab2a: MHL        $0      ; 將 HL 備份單元 0=83h~80h(10H)的值載入 HL
        MZR       $0      ; 將 ZR 備份單元 0=C3h~C0h(0021H)的值載入 ZR
        SMUI# @ZR         ; 將乘數的十位數(6)寫入 MUI, ZR = 22h
        RZR       $0      ; 將 ZR 內容值(22h) 存至 ZR 備份單元 0=C3h~C0h
        MULD# @HL         ; MUI(6) x Rx:10h(9) --> MU=5,AC=4; HL=11h.
        MZR       $1      ; 將 ZR 備份單元 1= C7h~C4h (30h)的值載入 ZR
        IDC%              ; ZR+1, ZR=31h.
        RZR       $1      ; 將 ZR 內容值(31h)存至 ZR 備份單元 1=C7h~C4h
        ADD*# @ZR,DA      ; Rx:31h(2) + AC(4) --> CF=0, Rx:31h=6, ZR=32h.
        RZR       $2      ; 將 ZR 內容值(32h) 存至 ZR 備份單元 2=CBh~C8h
        ADCM*# @ZR,DA     ; Rx:32h(6) + MU(5) + CF(0) --> CF=1,Rx:32h=1;
                          ; ZR=33h.
        JNC    lab1b      ; No jump occur.
        INC*#  @ZR        ; Rx:33h(0)+1 --> Rx:33h=1.
        JNC    lab1b      ; Jump to lab1b. (對應至手算式子的 Step3)
        INC*#  @ZR
        …..

lab1b: MULD# @HL          ; Rx:11h(8) x MUI(6) --> MU=4, AC=8; HL=12h
        MZR       $2      ; 將 ZR 備份單元 2 = CBh~C8h (32h)的值載入 ZR
        ADD*# @ZR,DA      ; Rx:32h(1) + AC(8) --> CF=0, Rx:32h=9; ZR=33h.
        RZR       $2      ; 將 ZR 內容值(33h) 存至 ZR 備份單元 2=CBh~C8h
        ADCM*# @ZR,DA     ; Rx:33h(1)+ MU(4)+CF(0) -->CF=0, Rx:33h=5;
                          ; ZR=34h.
        JNC    lab2b      ; Jump to “lab2b”. (對應至手算式子的 Step4)
        INC*#  @ZR
        …..

lab2b:  LDA    $0033      ; AC=5.
        LDA    $0032      ; AC=9.
        LDA    $0031      ; AC=6.
        LDA    $0030      ; AC=3
```

1-10. TIMERS

MCU 最多內建三組 6-bit timer (TMR1、TMR2、TMR3(TM89 系列&TM87ML28 提供))，每一組 timer 都可以先預設一個起始值然後開始倒數到 0，並且在任何時候都可將 timer 的內容值讀出來並儲存到 data RAM 中。

此外這三個 timer 可以透過互相串接的設定方式組合成 12-bit 或是 18-bit 長度的 timer，無論是 6-bit、12-bit 或是 18-bit 的 timer 都具有相同的操作功能。其中 TMR1 只能單獨作為 6-bit timer 使用；TMR2 可以單獨作為 6-bit timer 使用，或是與 TMR1 組合成 12-bit timer 之用，或是與 TMR1，TMR3 組合成 18-bit timer 之用；TMR3 可以單獨作為 6-bit timer 使用，或是與 TMR1 組合成 12-bit timer 之用。

此外這三個 timer 亦可以由 TM87ML28 所提供 STE 指令選擇其中一個 timer 透過串接一個額外 6-bit counter 的設定方式獨立延伸成 12-bit timer。而且這個額外的 6-bit counter 亦可藉由設定將內容值讀出來並儲存到 data RAM 中。

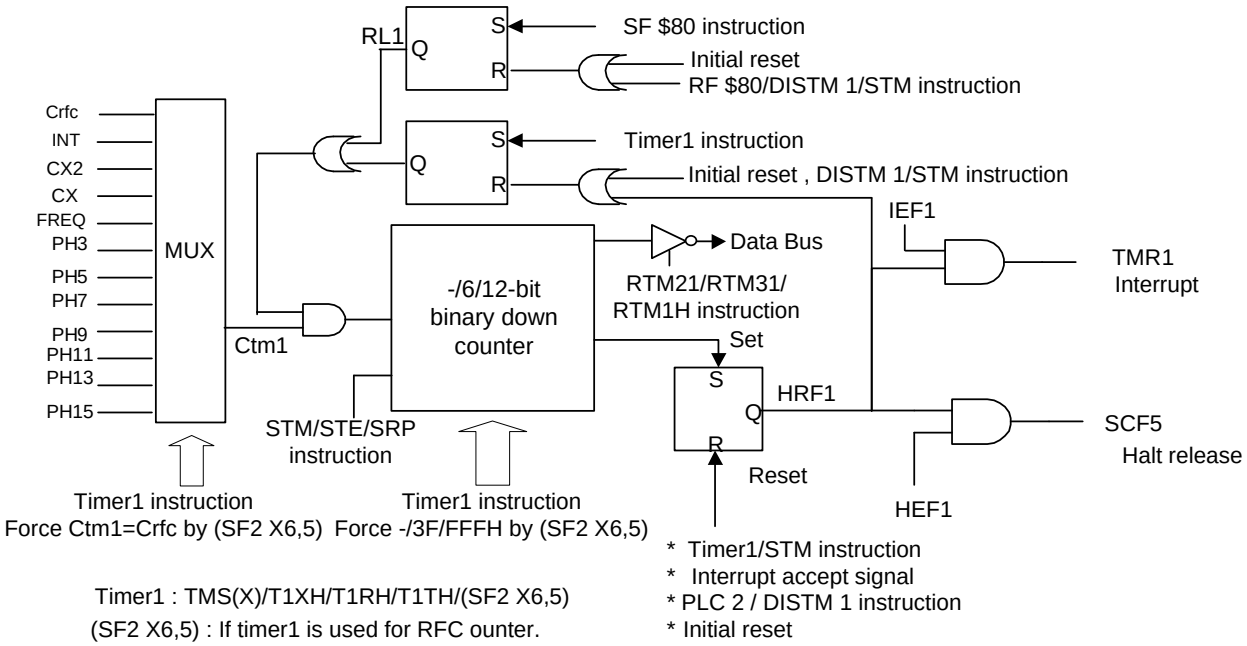
Timer 有以下幾種操作方式，基本的單次倒數計時功能(normal)，自動重新倒數計時功能(re-load)，作為 RFC 功能中的 counter 之用。此外 TMR2 還可以用來控制 RFC counter 的開啟或是關閉。在 TM87ML28 可提供 ADJ 指令選擇由 TMR1~3 其中一個校正 PDV(*請參閱第 1-2-4 小節詳細說明*)，也可選擇將 TMR2,3 underflow 輸出頻率(SRP=1)當作 SIO clock source，也提供 ST3OV 指令可選擇由 TMR3 在每次 underflow 時停止 FREQ 或者 TMR2 一個 Clock Source 週期。

1-10-1. 6-BIT TIMER

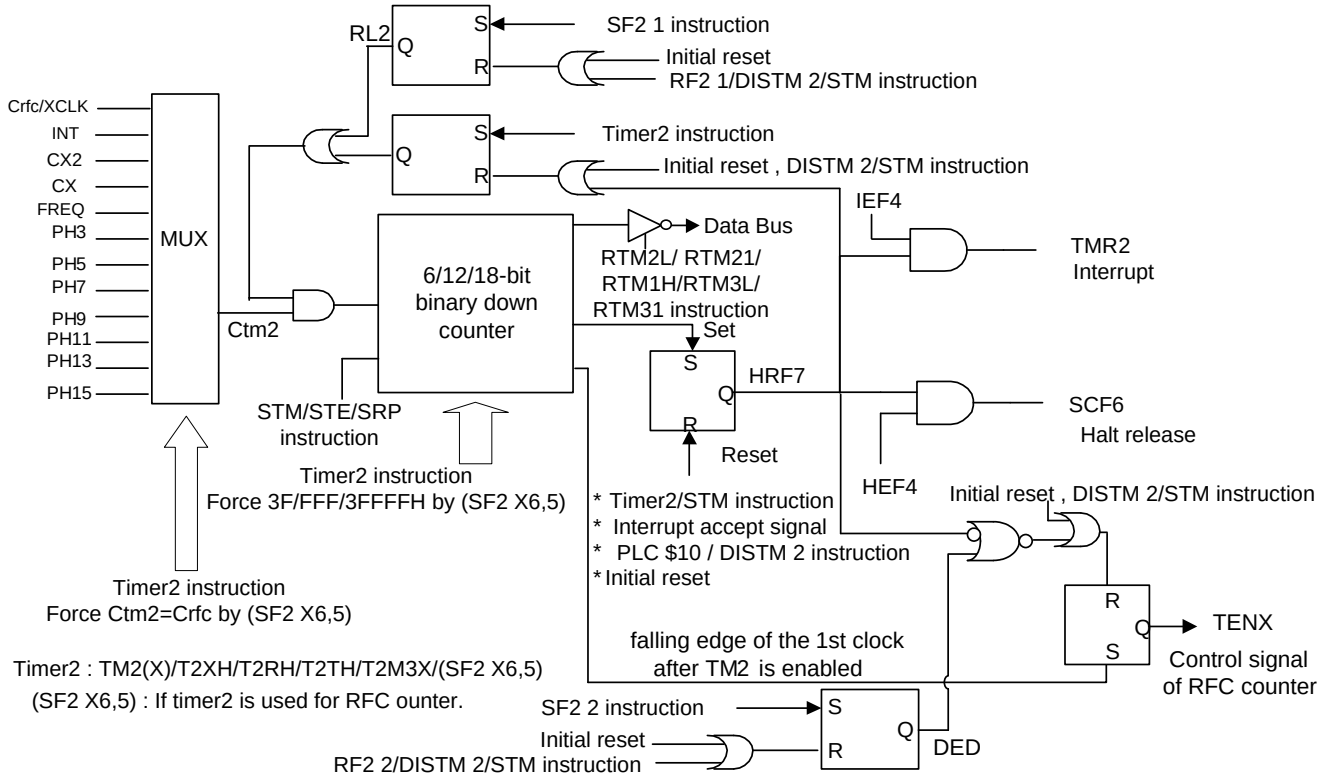
TMR1、TMR2、TMR3(TM89 系列&TM87ML28 提供)都可以單獨作為 6-bit timer 之用，而且都有相同的功能。

下面三個圖示說明 TMR1, TMR2, TMR3 的線路架構：

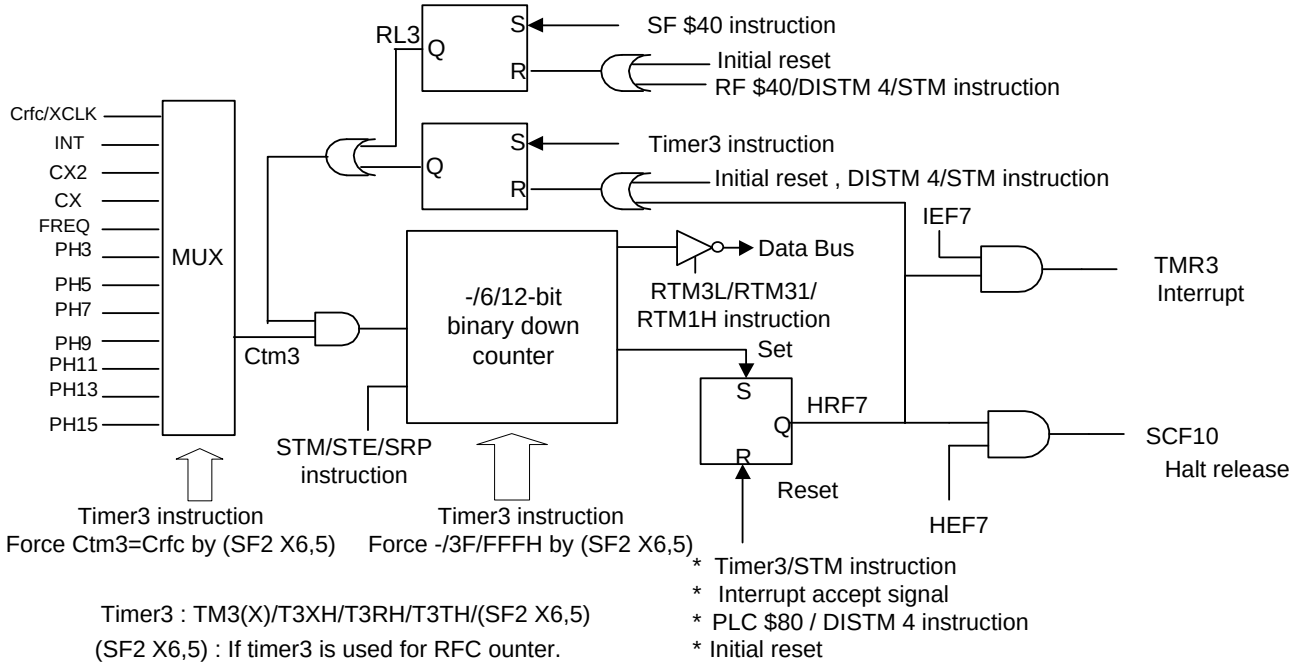
TMR1 的線路架構如下：



TMR2 的線路架構如下：



TMR3 的線路架構如下：



1-10-1-1. 單次倒數計時功能

這是 timer 的基本功能，timer 起始值的設定，clock source 的選擇以及 timer 的啟動控制都可以在同一個指令中完成設定，而且 timer 會在發生 underflow 之後或是在程式執行 DISTM 指令後停止運作。

當程式執行 DISTM 指令停止 timer 動作之後，timer 的內容值會維持在停止動作前的狀態。

下表說明 6-bit timer 的起始值的設定，clock source 的選擇以及啟動控制的相關指令：

TMRn	OPCODE	Select clock (CS3~0)				Pre-set data of timer (CT5~0)					
TMR1	T1XH SETDAT SD	SD15	SD14	SD13	SD12	SD11	SD10	SD9	SD8	SD7	SD6
	T1RH SETDAT Rx	(Rx)15	(Rx)14	(Rx)13	(Rx)12	(Rx)11	(Rx)10	(Rx)9	(Rx)8	(Rx)7	(Rx)6
	T1TH @HL	TD15	TD14	TD13	TD12	TD11	TD10	TD9	TD8	TD7	TD6
	TMSX X	0	X8	X7	X6	X5	X4	X3	X2	X1	X0
	TMS Rx	0	0	AC3	AC2	AC1	AC0	(Rx)3	(Rx)2	(Rx)1	(Rx)0
	TMS @HL	0	0	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0
TMR2	T2XH SETDAT SD	SD15	SD14	SD13	SD12	SD5	SD4	SD3	SD2	SD1	SD0
	T2RH SETDAT Rx	(Rx)15	(Rx)14	(Rx)13	(Rx)12	(Rx)5	(Rx)4	(Rx)3	(Rx)2	(Rx)1	(Rx)0
	T2TH @HL	TD15	TD14	TD13	TD12	TD5	TD4	TD3	TD2	TD1	TD0
	TM2X X	0	X8	X7	X6	X5	X4	X3	X2	X1	X0
	TM2 Rx	0	0	AC3	AC2	AC1	AC0	(Rx)3	(Rx)2	(Rx)1	(Rx)0
	TM2 @HL	0	0	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0
TMR3	T3XH SETDAT SD	SD15	SD14	SD13	SD12	SD5	SD4	SD3	SD2	SD1	SD0
	T3RH SETDAT Rx	(Rx)15	(Rx)14	(Rx)13	(Rx)12	(Rx)5	(Rx)4	(Rx)3	(Rx)2	(Rx)1	(Rx)0
	T3TH @HL	TD15	TD14	TD13	TD12	TD5	TD4	TD3	TD2	TD1	TD0
	TM3X X	0	X8	X7	X6	X5	X4	X3	X2	X1	X0
	TM3 Rx	0	0	AC3	AC2	AC1	AC0	(Rx)3	(Rx)2	(Rx)1	(Rx)0
	TM3 @HL	0	0	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0

當 TMR2,3 為 6-bit timer 時，執行 T2XH/T2RH/T2TH,T3XH/T3RH/T3TH 指令時 CT5~0 是由 SD/(RX)/TD5~0 所設定 & SD/(RX)/TD11~6 會被忽略。

當 TMR1 為 6-bit timer 時，執行 T1XH/T1RH/T1TH 指令時 CT5~0 與 TMR2,3 不同則是改由 SD/(Rx)/TD11~6 所設定 & SD/(RX)/TD5~0 會被忽略。

當 timer 從起始值開始倒數至 0 之後會產生一個 timer underflow 的信號，這個信號會將 timer halt release request flag (HRFn)設定為 1 並停止 timer 的動作。執行 PLC 指令可以將 HRFn 的 flag 清除為 0。

如果這時 timer interrupt enable flag (IEFn)已經設定為 1 時，在 HRFn flag 被設定為 1 之後 MCU 就會接受 timer 提出的中斷請求。

如果 timer halt release enable flag (HEFn)已經設定為 1 時，一旦 HRFn flag 被設定為 1，MCU 就會產生 HALT release 並且將 status register 3 (STS3)的 start condition flag (SCFn)設定為 1。

下表說明每一個 timer 所對應的 flag 以及相關的設定指令：

Timer	可設定的 timer halt	可清除 HRFnflag	對應的 timer	對應的 timer halt	對應的 start

	release request flag (HRFn)	的 PLC 指令設定	interrupt enable flag (IEFn)	release enable flag (HEFn)	condition flag (SCFn)
TMR1	HRF1	PLC $02	IEF1	HEF1	SCF5
TMR2	HRF4	PLC $10	IEF4	HEF4	SCF6
TMR3	HRF7	PLC $80	IEF7	HEF7	SCF10

請注意，當 timer 在進行倒數作業而且並未啟動 re-load 功能的時候，執行 PLC 指令指令會停止 timer 的動作；但是 timer 在進行倒數作業而且已經啟動 re-load 功能的時候，執行 PLC 指令指令不會停止 timer 的動作。

下表說明 clock source 的設定方式：

Bit setting in instruction				clock source of each timer
CS3	CS2	CS1	CS0	
0	0	0	0	~PH9
0	0	0	1	~PH3
0	0	1	0	~PH15
0	0	1	1	FREQ
0	1	0	0	~PH5
0	1	0	1	~PH7
0	1	1	0	~PH11
0	1	1	1	~PH13
1	0	0	0	CX
1	0	0	1	CX2
1	0	1	0	INT
1	0	1	1	~XCLK

Notes:

1. When the TIMER clock is PH3
TIMER set time = (Set value + error) * 8 * 1/fosc.
2. When the TIMER clock is PH9
TIMER set time = (Set value + error) * 512 * 1/fosc.
3. When the TIMER clock is PH15
TIMER set time = (Set value + error) * 32768 * 1/fosc .
4. When the TIMER clock is PH5
TIMER set time = (Set value + error) * 32 * 1/fosc.
5. When the timer clock is PH7
TIMER set time = (Set value + error) * 128 * 1/fosc.
6. When the TIMER clock is PH11
TIMER set time = (Set value + error) * 2048 * 1/fosc.
7. When the TIMER clock is PH13
TIMER set time = (Set value + error) * 8192 * 1/fosc.
8. When the TIMER clock is FREQ
TIMER set time = (Set value + error) * 1/FREQ. *Refer to section 2-3-4 for the detail of FREQ.*
9. When the TIMER clock is CX
TIMER set time = (Set value + error) * 1/CX.
10. When the TIMER clock is CX2

TIMER set time = (Set value + error) * 1/CX2.

11. When the TIMER clock is INT

TIMER set time = (Set value + error) * 1/INT.

12. When the TIMER clock is XCLK

TIMER set time = (Set value + 1) * 1/XCLK & TIMER is controlled by SIO.

Set value: Decimal number of timer set value

error: the tolerance of set value, 0 < error <1.

fosc: clock source of the predivider

PH3~PH15: The 3rd~15th stage output of the predivider

1-10-1-2. 自動重新倒數計時功能(re-load 功能)

如果 timer 開啟自動重新倒數計時功能之後，當 timer 發生 underflow 的時候不會停止倒數的動作，反而會依 SRP=0/1 繼續從 3Fh/設定值開始重新執行倒數功能(一般 MCU SRP 固定為 0，若 MCU 有提供 SRP 設定功能則 SRP 起始值=0，須由 SRP 指令設定變更為 1)，直到程式將 re-load 功能關閉後才會停止 timer 的動作。

下表說明設定開啟或是關閉 timer re-load 功能的相關指令：

Instruction	TMR1 re-load function	TMR2 re-load function	TMR3 re-load function
SF $80	Enabled	NA	NA
SF2 $01	NA	enabled	NA
SF $40	NA	NA	enabled
RF $80	Disabled	NA	NA
DISTM 1	Disabled	NA	NA
RF2 $01	NA	Disabled	NA
DISTM 2	NA	Disabled	NA
RF $40	NA	NA	Disabled
DISTM 4	NA	NA	Disabled
STM	Disabled	Disabled	Disabled

一旦程式啟動 Timer 的 re-load 功能之後 timer 就會開始動作，所以程式只能在 timer 完成 初始值設定並且開始動作之後才能啟動 re-load 功能，否則 timer 會在一個未知初始值的情況下開始倒數作業。

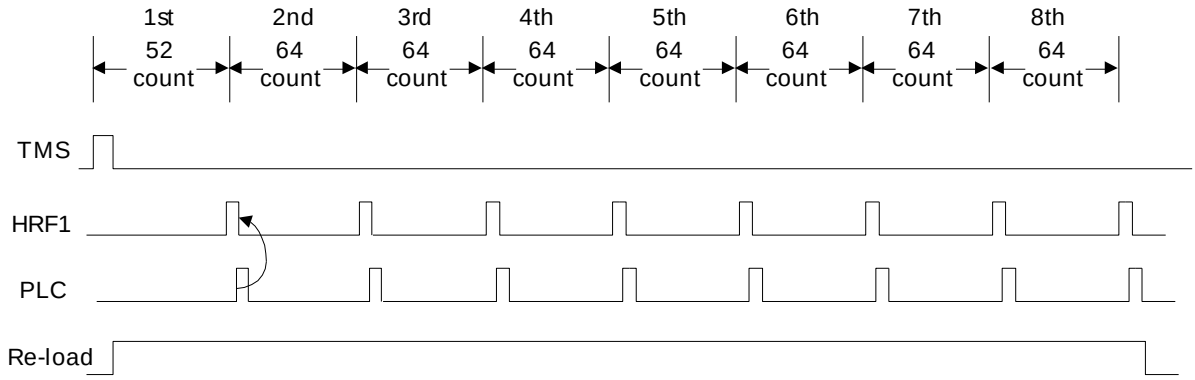
在 timer 的 re-load 功能啟動後，每當倒數至 0 之後就會發生 timer underflow，此時 timer 會從 3Fh 開始重新倒數，但是 timer underflow 仍會將 timer halt release request flag (HRFn)設定為 1。因此程式中必須利用 timer halt release request flag (HRFn)來判斷已經發生了幾次 timer underflow，這樣才能正確的計算時間長度。

在 re-load 功能關閉之前隨時都可以重新設 timer 的起始值，一旦重新設定 timer 起始值之後，timer 就會立刻從新設定的起始值開始倒數計時。

當 re-load 功能關閉的時候 timer 就會馬上停止動作，所以使用者應在最後一次 timer underflow 發生之後才關閉 timer 的 re-load 功能，這樣才不會影響 timer 的正常動作。

下面的範例說明如何使用 timer 的 re-load 功能：

如果程式希望 timer 的倒數計數值為 500 時，可以將 500 的倒數值規劃成一次 52 的倒數值以及 7 次 64 的倒數值(64*7 + 52)。在這個範例中程式會進入 HALT mode 來等待 TMR1 產生 underflow。




```
                LDS     0, 0            ; 將 data memory 位址 0 的內容值作為 TMR1
                                        ; underflow 次數的計數器，並將其歸 0。
                PLC     2               ; 將 HRF1 清除為 0
                SHE     2               ; 設定 TMR1 underflow 可以讓 MCU 產生 HALT release
                TMSX    $34             ; 設定 TMR1 的起始值(52)， clock source 選 PH9，
                                        ; 然後啟動 TMR1
                SF      $80             ; 開啟 TMR1 的 re-load 功能
RE_LOAD:
                HALT
                INC*    0               ; 將 TMR1 underflow 計數器加 1
                PLC     2               ; 將 HRF1 清除為 0
                JB3     END_TM1         ; 檢查 TMR1 underflow 次數是否等於 8 次
                JMP     RE_LOAD
END_TM1:
                RF      $80             ; 關閉 TMR1 的 re-load 功能
```

TM87ML28 提供 SRP 指令可將 re-load 功能切換成在每次 underflow 時皆會重新載入起始值，如上範列若最晚在第一次 underflow 之前增加執行 SRP 指令設定 X0=1 則每次 underflow 時皆會重新載入起始值(52)，故 timer 的倒數計數值將變成 52 + 7 x (52 + 1) =423(第一次 underflow 之前會有最大一個 Ctm 週期誤差)。

1-10-1-3. TIMER 作為 RFC COUNTER 之用

請參閱第 2-8-2-2 小節詳細說明。

1-10-1-4. 利用 TMR2 控制 RFC COUNTER

請參閱第 2-8-2-4 小節詳細說明。

1-10-1-5. 讀取 TIMER 目前的內容值

有兩種方式可以將三種 timer 目前的內容值讀出並儲存到 data memory 中，一種是直接以每次四個 bit 的方式分別讀取三個 timer 的內容值；另一種是先同步將三個 timer 內容值(18-bit)讀出並儲存在特定的暫存器中，然後再以每次四個 bit 的方式分別讀取 18-bit 暫存器的內容值。

這些讀取的動作並不會影響 timer 的正常動作，但是因為讀取的時間無法與 timer 的轉態時間取得同步，所以 timer 正在動作的情況下有可能會讀取到正在轉態中的 timer 內容值。

因為所有的 timer 都是倒數計時的架構，而設計者當初考慮到讀取 timer 內容值的使用時機大部分都是將 timer 當作計數器使用，因此將 timer 讀出來的內容值設計成是與目前 timer 的實際值呈補數的關係。

例如 timer 是從 3F 開始倒數，當 timer 已經倒數 6 個 clock 之後變成 39h，讀出來的 timer 內容值是 6，表示 timer 已經計數了 6 個 clock。

下表說明 timer 內容值與讀出值之間的補數關係：

Content of timer	Read out data
3Fh	00h
3Eh	01h
…	…
01h	3Eh
00h	3Fh

下面說明兩種讀取方式的詳細動作：

1. 直接以每次四個 bit 的方式分別讀取三個 6bit-timer 的內容值(這是 MCU 的原始設定)：

下表說明以四個 bit 的方式直接讀取三個 6bit-timer 的內容值的相關指令(STM X1,0=00，STE X2~0=000)：

inst.	Content of 6-bit TMR1						Content of 6-bit TMR2						Content of 6-bit TMR3					
	ct5	ct4	ct3	ct2	ct1	ct0	ct5	ct4	ct3	ct2	ct1	ct0	ct5	ct4	ct3	ct2	ct1	ct0
RTM2L									b3	b2	b1	b0						
RTM21					b3	b2	b1	b0										
RTM1H	b3	b2	b1	b0														
RTM3L															b3	b2	b1	b0
RTM31					b3	b2							b1	b0				

上表中的這些指令在執行之後會直接將所讀取到的 4-bit 的 timer 內容值儲存到 data memory 以及 AC 中。

Note: b3 ~ b0 表示 data memory 內容值的 bit3~bit0。

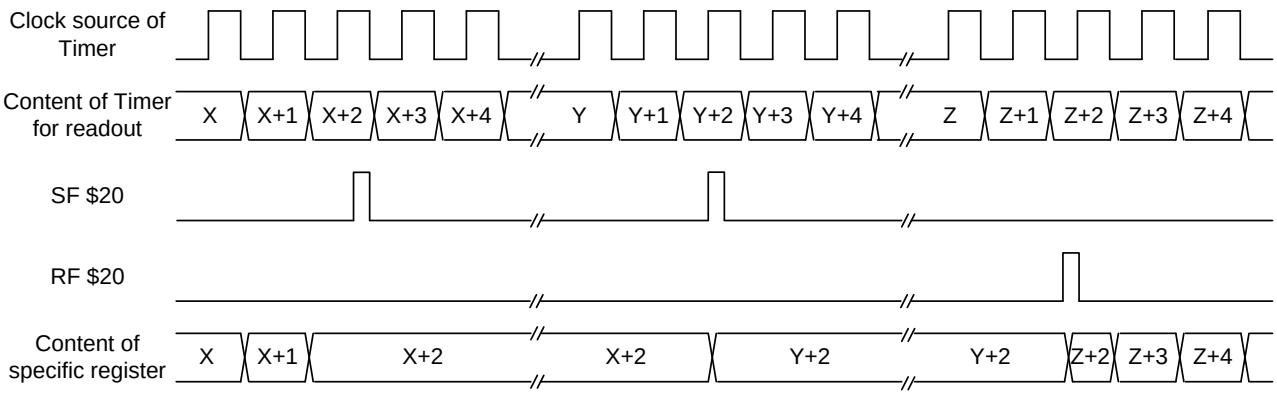
2. 同步將三個 timer 內容值(18-bit)讀出的方式：

執行 SF $20 可以同時將三個 timer 內容值(18-bit)讀出並儲存在特定的暫存器(specific register)中，然後再利用上表中的指令以每次 4-bit 的方式將 18-bit 的資料逐一儲存到 data memory 以及 AC 中。

在執行 SF $20 指令後，特定的暫存器中的資料將不會因為 timer 的持續動作而改變，必須等到再一次執行 SF $20 指令後才會更新暫存器的資料。

當程式執行 RF $20 指令後，timer 的讀取方式就會回復到每次四個 bit 的方式分別讀取三個 timer 的不同步內容值的模式。

下圖說明同時將三個 timer 內容值(18-bit)讀出的 timing 圖：



1-10-1-6. 利用 TMR3 微調 FREQ&TMR2 頻率

TM87ML28 提供 ST3OV 指令可由設定 X1,0=1 選擇由 TMR3 來調整 TMR2(設定 SRP X1=1), FREQ 頻率值。當 TMR3 (設定 Ctm3=FREQ/TMR2 & SRP X2=1)每次 underflow 時會遮蔽 TMR2/FREQ 一個 Ctm2/Cfq 週期已使長期等效頻率維持在趨近目標值。推導公式如下：

A / B = Q + R => 1/B x 1/R = Q/A x 1/R + 1/A

A : Ctm2 /Cfq 頻率

B : TMR2-underflow/FREQ-output 微調後目標頻率

Q : TMR2 /FREQ count number(整數)

1/R : TMR3 count number(四捨五入之整數)

由公式可得知 TMR2/FREQ 輸出週期(Q/A)每經 1/R 次計數會比目標頻率(B)少一個 Ctm2/Cfq 週期(1/A)，故 TMR3 須設定 Count 數為 1/R(四捨五入之整數)使每次 underflow 時須遮蔽一個 Ctm2/Cfq 週期可使 TMR2/ FREQ 等效輸出頻率趨近於目標值。

範例：以 Cfq=4MHz 產生 FREQ 輸出 153.6KHz UART 等效頻率。

4MHz/153.6KHz = 26.0416666.. => Q=26 , 1/R= 1/0.0416666… = 24

故範例程式內容如下：

```
SCC $40         ;set Cfq=XCLK(4MHz)
FRQX $3,$19     ;set FRQ : 1/1duty & Value=25(Count Number=25+1)
ST3OV $1        ;set mask one Cfq cycle by TMR3 underflow.
TM3X  $0d7      ;set Ctm3=FREQ & Value=23(Count Number=23+1)
SRP    $4       ;set TMR3 SRP=1 to make all underflow time by same count number.
SF $40          ;set RL3=1
```

當設定 FREQ 為 UART 的 TX/RX 的 16 倍頻時鐘來源時，TXCK/RXCK=1 或者執行 MWM$3,Ry / RXD(UART1W=1 : TXD)接收到下降緣則會自動啟動 FREQ。

1-10-2. 12-BIT TIMER(TM89 系列&TM87ML28 提供)

12-bit timer 可由兩個 6-bit timer 以 STM 指令所組合而成，可以將 TMR1 與 TMR2 組合成 12-bit 的 TMR2，或是將 TMR1 與 TMR3 組合成 12-bit 的 TMR3。在 12-bit 的 TMR2 或是 TMR3 架構下，6-bit 的 TMR1 功能將會失效。此外 TM87ML28 亦提供 STE 指令可以單獨選擇其中一個 Timer 延展成 12-bit timer，TMR1 也只有在設定 STE X1,0=01 時才會是 12-bit timer。

下表說明組合 12-bit timer 的相關指令：

Instruction to merge timers	TMR1	TMR2	TMR3
STM X1,0=1	●	● (dominate)	
STM X1,0=2	●		● (dominate)
STE X1,0=1	●		
STE X1,0=2		●	
STE X1,0=3			●

12-bit 的 TMR2 與 6-bit 的 TMR2 有相同的功能，而 12-bit 的 TMR3 與 6-bit 的 TMR3 有相同的功能。

如果 12-bit timer 開啟 re-load 功能之後，當 timer 發生 underflow 的時候不會停止倒數的動作，反而會依照 SRP=0/1 繼續從 FFFh/設定值開始重新執行倒數功能，直到程式將 re-load 功能關閉後才會停止 timer 的動作。

下表說明設定與啟動 12-bit timer 的相關指令：

OPCODE	Select clock(CS3~0)				Pre-set data of timer(CT11~0)											
T1XH SETDAT SD	SD15	SD14	SD13	SD12	SD11	SD10	SD9	SD8	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
T1RH SETDAT RX	(Rx)15	(Rx)14	(Rx)13	(Rx)12	(Rx)11	(Rx)10	(Rx)9	(Rx)8	(Rx)7	(Rx)6	(Rx)5	(Rx)4	(Rx)3	(Rx)2	(Rx)1	(Rx)0
T1TH @HL	TD15	TD14	TD13	TD12	TD11	TD10	TD9	TD8	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0
T2XH SETDAT SD	SD15	SD14	SD13	SD12	SD11	SD10	SD9	SD8	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
T2RH SETDAT RX	(Rx)15	(Rx)14	(Rx)13	(Rx)12	(Rx)11	(Rx)10	(Rx)9	(Rx)8	(Rx)7	(Rx)6	(Rx)5	(Rx)4	(Rx)3	(Rx)2	(Rx)1	(Rx)0
T2TH @HL	TD15	TD14	TD13	TD12	TD11	TD10	TD9	TD8	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0
T3XH SETDAT SD	SD15	SD14	SD13	SD12	SD11	SD10	SD9	SD8	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
T3RH SETDAT RX	(Rx)15	(Rx)14	(Rx)13	(Rx)12	(Rx)11	(Rx)10	(Rx)9	(Rx)8	(Rx)7	(Rx)6	(Rx)5	(Rx)4	(Rx)3	(Rx)2	(Rx)1	(Rx)0
T3TH @HL	TD15	TD14	TD13	TD12	TD11	TD10	TD9	TD8	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0

請注意，當 timer 設定成 12-bit timer 時, 不可使用原先 6-bit TMR2 或是 TMR3 的相關指令來做設定，否則會有設定值錯誤的情形發生。

TMR1 由 STE 延長成 12-bit timer 時執行 T1XH/T1RH/T1TH 會變成與 Timer2,3 一樣由 SD/(Rx)/TD11~0 設定 CT11~0，與 TMR1 為 6-bit timer 由 SD/(Rx)/TD11~6 設定 CT5~0 不同。

下列表格說明如何讀取 12-bit Timer 的內容值：

inst.	Content of 12-bit TMR2 by execute STM X1,0=01												Content of 12-bit TMR3 by execute STE X1,0=11											
	ct 11	ct 10	ct 9	ct 8	ct 7	ct 6	ct 5	ct 4	ct 3	ct 2	ct 1	ct 0	ct 11	ct 10	ct 9	ct 8	ct 7	ct 6	ct 5	ct 4	ct 3	ct 2	ct 1	ct 0
STE	X2=0																							
RTM2L									b3	b2	b1	b0												
RTM21					b3	b2	b1	b0																
RTM1H	b3	b2	b1	b0																				
RTM3L																					b3	b2	b1	b0
RTM31					b3	b2													b1	b0				
STE	X2=1																							
RTM2L									b3	b2	b1	b0												
RTM21							b1	b0									b3	b2						
RTM1H													b3	b2	b1	b0								
RTM3L																					b3	b2	b1	b0
RTM31																	b3	b2	b1	b0				

inst.	Content of 12-bit TMR2 by execute STE X1,0=10												Content of 12-bit TMR3 by execute STM X1,0=10											
	ct 11	ct 10	ct 9	ct 8	ct 7	ct 6	ct 5	ct 4	ct 3	ct 2	ct 1	ct 0	ct 11	ct 10	ct 9	ct 8	ct 7	ct 6	ct 5	ct 4	ct 3	ct 2	ct 1	ct 0
STE	X2=0																							
RTM2L									b3	b2	b1	b0												
RTM21							b1	b0									b3	b2						
RTM1H													b3	b2	b1	b0								
RTM3L																					b3	b2	b1	b0
RTM31																	b3	b2	b1	b0				
STE	X2=1																							
RTM2L									b3	b2	b1	b0												
RTM21					b3	b2	b1	b0																

RTM1H	b3	b2	b1	b0																			
RTM3L																			b3	b2	b1	b0	
RTM31					b3	b2													b1	b0			

inst.	Content of 12-bit TMR1 by execute STE X1,0=01												Content of 6-bit TMR2						Content of 6-bit TMR3					
	ct 11	ct 10	ct 9	ct 8	ct 7	ct 6	ct 5	ct 4	ct 3	ct 2	ct 1	ct 0	ct 5	ct 4	ct 3	ct 2	ct 1	ct 0	ct 5	ct 4	ct 3	ct 2	ct 1	ct 0
STE	X2=0																							
RTM2L															b3	b2	b1	b0						
RTM21											b3	b2	b1	b0										
RTM1H							b3	b2	b1	b0														
RTM3L																					b3	b2	b1	b0
RTM31											b3	b2							b1	b0				
STE	X2=1																							
RTM2L									b3	b2	b1	b0												
RTM21					b3	b2	b1	b0																
RTM1H	b3	b2	b1	b0																				
RTM3L																					b3	b2	b1	b0
RTM31					b3	b2													b1	b0				

inst.	Content of 12-bit TMR2 by execute STE X1,0=10												Content of 6-bit TMR1						Content of 6-bit TMR3					
	ct 11	ct 10	ct 9	ct 8	ct 7	ct 6	ct 5	ct 4	ct 3	ct 2	ct 1	ct 0	ct 5	ct 4	ct 3	ct 2	ct 1	ct 0	ct 5	ct 4	ct 3	ct 2	ct 1	ct 0
STE	X2=0																							
RTM2L									b3	b2	b1	b0												
RTM21							b1	b0									b3	b2						
RTM1H													b3	b2	b1	b0								
RTM3L																					b3	b2	b1	b0
RTM31																	b3	b2	b1	b0				
STE	X2=1																							
RTM2L									b3	b2	b1	b0												
RTM21					b3	b2	b1	b0																

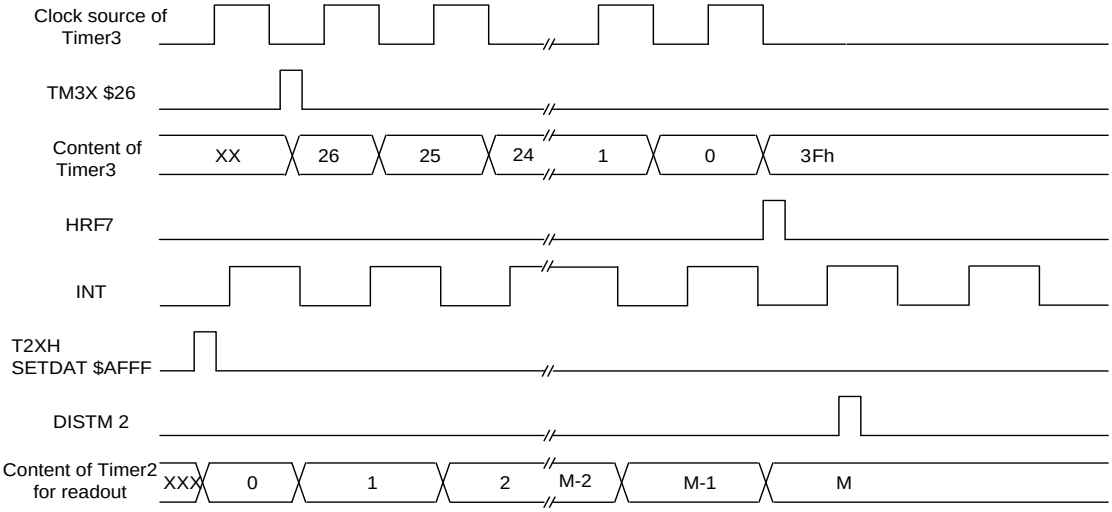
RTM1H	b3	b2	b1	b0																			
RTM3L																			b3	b2	b1	b0	
RTM31					b3	b2													b1	b0			

inst.	Content of 12-bit TMR3 by execute STE X1,0=11												Content of 6-bit TMR1						Content of 6-bit TMR2					
	ct 11	ct 10	ct 9	ct 8	ct 7	ct 6	ct 5	ct 4	ct 3	ct 2	ct 1	ct 0	ct 5	ct 4	ct 3	ct 2	ct 1	ct 0	ct 5	ct 4	ct 3	ct 2	ct 1	ct 0
STE	X2=0																							
RTM2L																					b3	b2	b1	b0
RTM21																	b3	b2	b1	b0				
RTM1H													b3	b2	b1	b0								
RTM3L									b3	b2	b1	b0												
RTM31							b1	b0									b3	b2						
STE	X2=1																							
RTM2L																					b3	b2	b1	b0
RTM21					b3	b2													b1	b0				
RTM1H	b3	b2	b1	b0																				
RTM3L									b3	b2	b1	b0												
RTM31					b3	b2	b1	b0																

下面的範例說明如何使用 12-bit timer 2 (TMR2)與 6-bit timer 3 (TMR3)來完成計算 INT pin 上所接收到的 clock 的頻率：

```
        STM    $1          ; 將 TMR2 設定成 12-bit timer
        T2XH               ; 設定 INT 為 TMR2 的 clock source，起始值為 FFFh
        setdat $AFFF ;
        TM3X $026          ; 設定 PH9 為 TMR3 的 clock source，起始值為 26h
        SHE    $90         ; 設定 TMR2, TMR3 可以產生 HALT release
        HALT
        DISTM$2            ; 關閉 TMR2
        MCX    $6f         ; 檢查是否 TMR2 underflow
        JB1    tm2_ov
        PLC    $070        ; 清除 HRF7(TMR3).
        RTM2L $0010        ; 讀取 12-bit TMR2 的內容值，
        RTM21 $0011        ; 代表 INT pin 的輸入 clock 數目
        RTM1H $0012;
        ……..
tm2_ov: NOP                ; TMR2 已經發生 underflow
```

下圖說明這個範例中各個信號之間的 timing：



1-10-3. 18-BIT TIMER(TM89 系列&TM87ML28 提供)

18-bit timer 是由三個 6-bit timer 所組合而成，而且只能組合成一個 18-bit 的 TMR2。在 18-bit 的 timer 架構下，6-bit 的 TMR1 以及 TMR3 功能將會失效。

下表說明組合 18-bit timer 的相關指令：

instruction to merge timers	TMR1	TMR2	TMR3
STM 3	●	● (dominate)	●

18-bit 的 TMR2 與 6-bit 的 TMR2 有相同的功能。

如果 18-bit timer 開啟 re-load 功能之後，當 timer 發生 underflow 的時候不會停止倒數的動作，反而會繼續從 3FFFFh 開始重新執行倒數功能，直到程式將 re-load 功能關閉後才會停止 timer 的動作。

下表說明設定與啟動 18-bit timer 的相關指令：

OPCODE	Select clock (CS3~0)				Pre-set data of timer (CT17~0)						
T2M3X X SETDAT SD	SD15	SD14	SD13	SD12	X5	X4	X3	X2	X1	X0	CT17~12
					SD11	SD10	SD9	SD8	SD7	SD6	CT11~6
					SD5	SD4	SD3	SD2	SD1	SD0	CT5~0

請注意，當 TMR2 設定成 18-bit timer 時，不可使用原先 6-bit 或是 12-bit TMR2 的相關指令來做設定，否則會有設定值錯誤的情形發生。

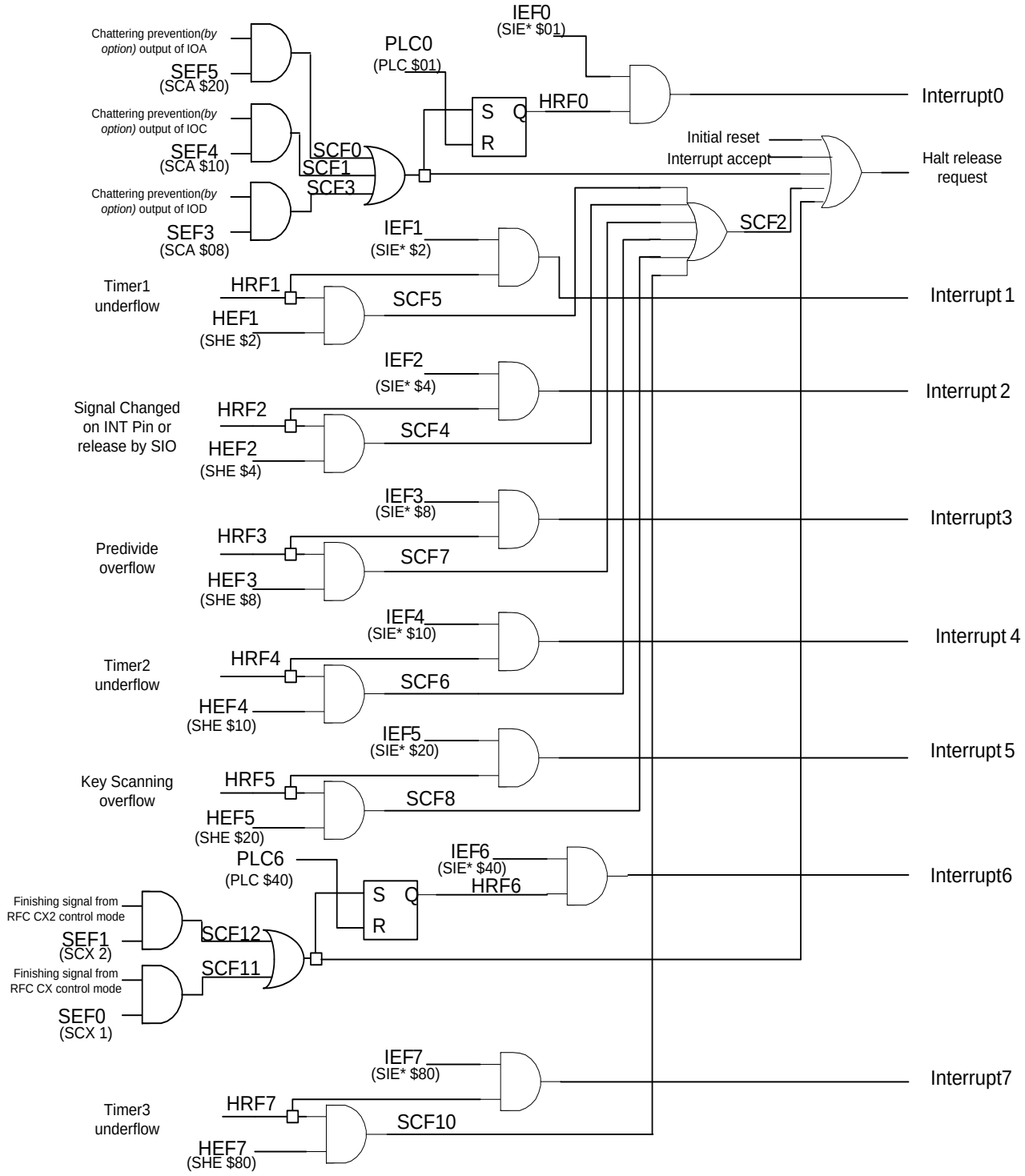
下表說明如何讀取 18-bit TMR2 的內容值(STM X1,0=11 , STE X2~0=000)：

Timer read-out inst.	Content of 18-bit TMR2																	
	Ct-17	Ct-16	Ct-15	Ct-14	Ct-13	Ct-12	Ct-11	Ct-10	Ct-9	Ct-8	Ct-7	Ct-6	Ct-5	Ct-4	Ct-3	Ct-2	Ct-1	Ct-0
RTM2L															b3	b2	b1	b0
RTM21											b3	b2	b1	b0				
RTM1H							b3	b2	b1	b0								
RTM3L			b3	b2	b1	b0												
RTM31	b1	b0									b3	b2						

1-11. STATUS REGISTER (STS)

4BIT 系列 MCU 共有 7 種不同的 status register (STS)，分別是 STS1, STS2, STS3, STS3X, STS4, STS4X 以及 STS5，主要用來記錄各種可以使得 MCU 產生 HALT release 的 start condition flag 以及其他重要的信號 flag。每一個 status register 包含 4 個 bit 的資料，都可以利用相關的指令做讀取並儲存到 data memory 中。

下圖說明 4BIT 系列 MCU 中各個 start condition flags 的架構圖：



1-11-1. STATUS REGISTER 1 (STS1)

STS1 記錄了 start condition flag 11, 12 (SCF11, 12) 兩個可以使 MCU 產生 HALT release 的因子。

STS1 是由四個 flag 所組成：

1. Carry flag (CF)
Carry flag 會將算術運算中的進位以及借位的結果記錄下來。

2. Zero flag (ZF)
Zero flag 可以指示 accumulator (AC)內容值的狀態。
當 AC 內容值為 0 時，zero flag 會設定為 1；
當 AC 內容值不等於 0 時，zero flag 會清除為 0。

3. Start condition flag 12 (SCF12)
當 RFC = TYPE B 時提供 SCF12 功能，如果 RFC counter 是由 CX2 pin 的輸入信號來控制啟動或是關閉，當程式執行 SCX 指令將 SEF1 flag 設定為 1 之後，在 CX2 pin 的控制信號結束時就會將 SCF12 flag 設定 為 1。
當程式重新執行 SCX 指令後會自動將 SCF12 清除為 0。

4. Start condition flag 11 (SCF11)
當 RFC = TYPE B 時提供 SCF11 功能，如果 RFC counter 是由 CX pin 的輸入信號來控制啟動或是關閉時，當程式執行 SCX 指令將 SEF0 flag 設定為 1 之後，在 CX pin 的控制信號結束時就會將 SCF11 flag 設定為 1。
當程式重新執行 SCX 指令後會自動將 SCF11 清除為 0。

執行 MAF 指令可將 STS1 的內容值儲存到 AC 以及 data memory。

執行 MRA 指令可將 data memory 的內容值寫入 STS1 中的 carry flag。

下表說明 status register 1 (STS1)各 flag 與 data memory 內容值之間的關係：

Bit 3	Bit 2	Bit 1	Bit 0
Carry flag (CF)	Zero flag(Z)	SCF12	SCF11
Read/write	Read only	Read only	Read only

1-11-2. STATUS REGISTER 2 (STS2)

STS2 記錄了 start condition flag 1, 2, 3 (SCF1, 2, 3) 三個可以使 MCU 產生 HALT release 的因子。

STS2 是由 start condition flag 1, 2, 3 (SCF1, 2, 3)以及 backup flag(BCF)等四個 flag 所組成：

1. Start condition flag 3 (SCF3)

如果 SCA 指令已經設定 IOD port 的輸入腳位上的信號變化可以使 MCU 產生 HALT release 之後，當這些輸入信號發生變化時就會將 SCF3 設定為 1。
當程式重新執行 SCA 指令後會自動將 SCF3 清除為 0。

2. Start condition flag 1 (SCF1)
如果 SCA 指令已經設定 IOC port 的輸入腳位上的信號變化可以使 MCU 產生 HALT release 之後，當這些輸入信號發生變化時就會將 SCF1 設定為 1。
當程式重新執行 SCA 指令後會自動將 SCF1 清除為 0。

3. Start condition flag 2 (SCF2)
當 SCF4、5、6 、7 、9 、10 等 flag 所代表的任何一個可以使 MCU 產生 HALT release 的因子發生時，SCF2 就會設定為 1。只有在這些 flag 全部都清除為 0 之後 SCF2 才會清除為 0。
只要有任何一個 start condition flag 設定為 1 時，MCU 將無法進入 HALT mode。

4. Backup flag (BCF)

BCF flag 指示目前 MCU 是否進入電力備援模式。
執行 SF 2h 指令可以使得 MCU 進入電力備援模式並將 BCF flag 設定為 1；執行 RF 2h 指令可以使得 MCU 離開電力備援模式並將 BCF flag 清除為 0。

執行 MSB 指令可將 STS2 的內容值儲存到 AC 以及 data memory，而且 STS2 是一個唯讀的暫存器。

下表說明 status register 2 (STS2)各 flag 與 data memory 內容值之間的關係：

Bit 3	Bit 2	Bit 1	Bit 0
Start condition flag 3 (SCF3)	Start condition flag 2 (SCF2)	Start condition flag 1 (SCF1)	Backup flag (BCF)
Halt release caused by the IOD port	Halt release caused by SCF4,5,6,7,9,10	Halt release caused by the IOC port	The back up mode status
Read only	Read only	Read only	Read only

1-11-3. STATUS REGISTER 3 (STS3)

STS3 記錄了 start condition flag 4、5 、7 (SCF4、5 、7)三個可以使 MCU 產生 HALT release 的因子。

STS3 是由四個 start condition flag 所組成：

1. Start condition flag 4 (SCF4)
在 halt release enable flag 2 (HEF2)設定為 1 之後，INT 腳位上的上昇緣或是下降緣 (由 code option 決定)信號所觸發產生或是由 SIO 所產生的 halt release request flag 2 (HRF2)會將 start condition flag 4 (SCF4)設定為 1。
執行 PLC 指令將 halt release request flag 2 (HRF2)清除為 0，或是執行 SHE 指令將 halt release enable flag 2 (HEF2)清除為 0，同樣都會將 start condition flag 4 (SCF4)清除為 0。

2. Start condition flag 5 (SCF5)
在 halt release enable flag 1 (HEF1)設定為 1 之後，TMR1 在 underflow 之後所產生的 halt release request flag 1 (HRF1)會將 start condition flag 5 (SCF5)設定為 1。
執行 PLC 指令將 halt release request flag 1 (HRF1)清除為 0，或是執行 SHE 指令將 halt release enable flag 1 (HEF1)清除為 0，同樣都會將 start condition flag 5 (SCF5)清除為 0。

3. Start condition flag 7 (SCF7)
在 halt release enable flag 3 (HEF3)設定為 1 之後，pre-divider 在 overflow 之後所產生的 halt release request flag 3 (HRF3)會將 start condition flag 7 (SCF7)設定為 1。
執行 PLC 指令將 halt release request flag 3 (HRF3)清除為 0，或是執行 SHE 指令將 halt release enable flag 3 (HEF3)清除為 0，同樣都會將 start condition flag 7 (SCF7)清除為 0。

4. Pre-divider 第 15 級的輸出信號(PH15)

執行 MSC 指令可將 STS3 的內容值儲存到 AC 以及 data memory，而且 STS3 是一個唯讀的暫存器。

下表說明 status register 3 (STS3)各 flag 與 data memory 內容值之間的關係：

Bit 3	Bit 2	Bit 1	Bit 0
Start condition flag 7 (SCF7)	Content of 15th stage of the pre-divider	Start condition flag 5 (SCF5)	Start condition flag 4 (SCF4)

Halt release caused by pre-divider overflow		Halt release caused by TMR1 underflow	Halt release caused by INT pin/SIO
Read only	Read only	Read only	Read only

1-11-4. STATUS REGISTER 3X (STS3X)

STS3X 記錄了 start condition flag 0、6、8 (SCF0、6、8)三個可以使 MCU 產生 HALT release 的因子。

STS3X 是依 RFC TYPE=A/B(TM89 系列為 B TYPE，TM87ML28 則可由 Mask Option 選擇)來決定由四/三個 start condition flag 所組成：

1. Start condition flag 8 (SCF8)
在 halt release enable flag 5 (HEF5)設定為 1 之後，當每個掃描週期結束時或是在 KI1~4 腳位上經掃描電路偵測到高電位的輸入信號時(由 key matrix 掃描功能的相關指令所設定)所產生的 halt release request flag 5 (HRF5)會將 start condition flag 8 (SCF8)設定為 1。
執行 PLC 指令將 halt release request flag 5 (HRF5)清除為 0，或是執行 SHE 指令將 halt release enable flag 5 (HEF5)清除為 0，同樣都會將 start condition flag 8 (SCF8)清除為 0。

2. Start condition flag 6 (SCF6)
在 halt release enable flag 4 (HEF4)設定為 1 之後，TMR2 在 underflow 之後所產生的 halt release request flag 4 (HRF4)會將 start condition flag 6 (SCF6)設定為 1。
執行 PLC 指令將 halt release request flag 4 (HRF4)清除為 0，或是執行 SHE 指令將 halt release enable flag 4 (HEF4)清除為 0，同樣都會將 start condition flag 6 (SCF6)清除為 0。

3. Start condition flag 0 (SCF0)
如果 SCA 指令已經設定 IOA port 的輸入腳位上的信號變化可以使 MCU 產生 HALT release 之後，當這些輸入信號發生變化時就會將 SCF0 設定為 1。
當程式重新執行 SCA 指令後會自動將 SCF0 清除為 0。

4. Start condition flag 9 (SCF9)
當 RFC TYPE=A 時提供 SCF9 功能，在 halt release enable flag 6 (HEF6)設定為 1 之後，RFC 在 CX control mode 結束動作之後所產生的 halt release request flag 6 (HRF6)會將 start condition flag 9 (SCF9)設定為 1。
執行 PLC 指令將 halt release request flag 6 (HRF6)清除為 0，或是執行 SHE 指令將 halt release enable flag 6 (HEF6)清除為 0，同樣都會將 start condition flag 9 (SCF9)清除為 0。

執行 MCX 指令可將 STS3X 的內容值儲存到 AC 以及 data memory，而且 STS3X 是一個唯讀的暫存器。

下表說明 status register 3X (STS3X)各 flag 與 data memory 內容值之間的關係：

Bit 3	Bit 2	Bit 1	Bit 0
Reserved	Start condition flag 0 (SCF0)	Start condition flag 6 (SCF6)	Start condition flag 8 (SCF8)
	Halt release caused by the IOA port	Halt release caused by TMR2 underflow	Halt release caused by key matrix scanning function
Read only	Read only	Read only	Read only

1-11-5. STATUS REGISTER 4 (STS4)

STS4 是由四個 flag 所組成：

1. System clock selection flag (CSF)
CSF flag 可以指示目前 system clock generator 所使用的 clock 來源。
當 CSF flag 設定為 1 時表示目前正在使用高速 clock (CF clock)，當 CSF flag 清除為 0 時表示目前正在使用低速 clock (XT clock)。

2. Watch dog timer enable flag (WDF)
WDF flag 可以指示目前 watch dog timer 的功能是否已經開啟。
當 WDF flag 設定為 1 時表示目前 watch dog timer 功能已經開啟，當 WDF flag 清除為 0 時，表示目前 watch dog timer 功能已經關閉。

3. Overflow flag of 16-bit RFC counter (RFOVF)
當 16-bit RFC counter 發生 overflow 的時候會將 RFOVF flag 設定為 1，執行 SF2(X6,X5)指令可以將 RFOVF flag 清除為 0。

4. Table ROM In Application Program Flag(PGMF) (TM87ML28 提供)

當執行 PTR 指令(TM87ML28 提供)將 RILH15~0 內的資料燒錄至 HL15~1 所指定 Table ROM Word Address 時會將 PGMF flag 設定為 1 直到燒錄結束(若是燒錄失敗則維持為 1 直到下次執行 PTR 燒錄成功。

執行 MSD 指令可將 STS4 的內容值儲存到 AC 以及 data memory，而且 STS4 是一個唯讀的暫存器。

下表說明 status register 4 (STS4)各 flag 與 data memory 內容值之間的關係：

Bit 3	Bit 2	Bit 1	Bit 0
Table ROM In Application Program Flag (PGMF)	The overflow flag of 16-bit RFC counter (RFVOF)	Watch dog timer Enable flag (WDF)	System clock selection flag (CSF)
Read only	Read only	Read only	Read only

1-11-6. STATUS REGISTER 4X (STS4X)

STS4X 記錄了 start condition flag 10 (SCF10) 一個可以使 MCU 產生 HALT release 的因子。

STS4X 是由 start condition flag 10 (SCF10)以及三個輸入腳位(CX, CX2, INT)的輸入信號狀態所組成：

1. CX 輸入腳位的輸入信號狀態
CX pin 是 RFC 功能中的一個輸入腳位，MCU 可以透過 STS4X 暫存器讀取它的邏輯位準。

2. CX2 輸入腳位的輸入信號狀態
CX2 pin 是 RFC 功能中的另一個輸入腳位，MCU 可以透過 STS4X 暫存器讀取它的邏輯位準。

3. INT 輸入腳位的輸入信號狀態
INT pin 是一個外部中斷的信號來源，MCU 可以透過 STS4X 暫存器讀取它的邏輯位準。

4. Start condition flag 10 (SCF10)
在 halt release enable flag 7 (HEF7)設定為 1 之後，TMR3 在 underflow 之後所產生的 halt release request flag 7 (HRF7)會將 start condition flag 10 (SCF10)設定為 1。
執行 PLC 指令將 halt release request flag 7 (HRF7)清除為 0，或是執行 SHE 指令將 halt release enable flag 7 (HEF7)清除為 0，同樣都會將 start condition flag 10 (SCF10)清除為 0。

執行 MDX 指令可將 STS4X 的內容值儲存到 AC 以及 data memory，而且 STS4X 是一個 唯讀的暫存器。

下表說明 status register 4X (STS4X)各 flag 與 data memory 內容值之間的關係：

Bit 3	Bit 2	Bit 1	Bit 0
Start condition flag 10 (SCF10)	Logic state on INT pin	Logic state on CX2 pin	Logic state on CX pin
Halt release caused by TMR3 underflow			
Read only	Read only	Read only	Read only

1-11-7. STATUS REGISTER 5 (STS5)

STS5 是由四個 key matrix 掃描功能輸入腳位(KI1 ~ KI4)上的邏輯狀態所組成：

執行 MKI 指令可將 STS5 的內容值儲存到 AC 以及 data memory，而且 STS5 是一個唯讀的暫存器。

下表說明 status register 5 (STS5)各位元與 data memory 之間的關係：

Bit 3	Bit 2	Bit 1	Bit 0
Logic state on KI4 key-scanning input pin	Logic state on KI3 key-scanning input pin	Logic state on KI2 key-scanning input pin	Logic state on KI1 key-scanning input pin
Read only	Read only	Read only	Read only
Bit 3	Bit 2	Bit 1	Bit 0

1-12. CONTROL REGISTER (CTL)

4BIT 系列 MCU 共有 5 種不同的 control register (CTL)，分別是 CTL1、CTL2、CTL3、CTL4 以及 CTL5。MCU 必須先將對應的 control register 設定為 1，這樣在這些 start condition flag 發生時才能夠產生 HALT release。

1-12-1. CONTROL REGISTER 1 (CTL1)

CTL1 是由 SEF3、SEF4、SEF5 所組成，執行 SCA 指令可以設定或是清除這些 flag，control register 1 (CTL1)是唯寫的暫存器。

1. Switch enable flag 5 (SEF5)
當 switch enable flag 5 (SEF5)設定為 1 之後，IOA port 輸入腳位上的任何信號產生有效改變就能讓 MCU 產生 HALT release。

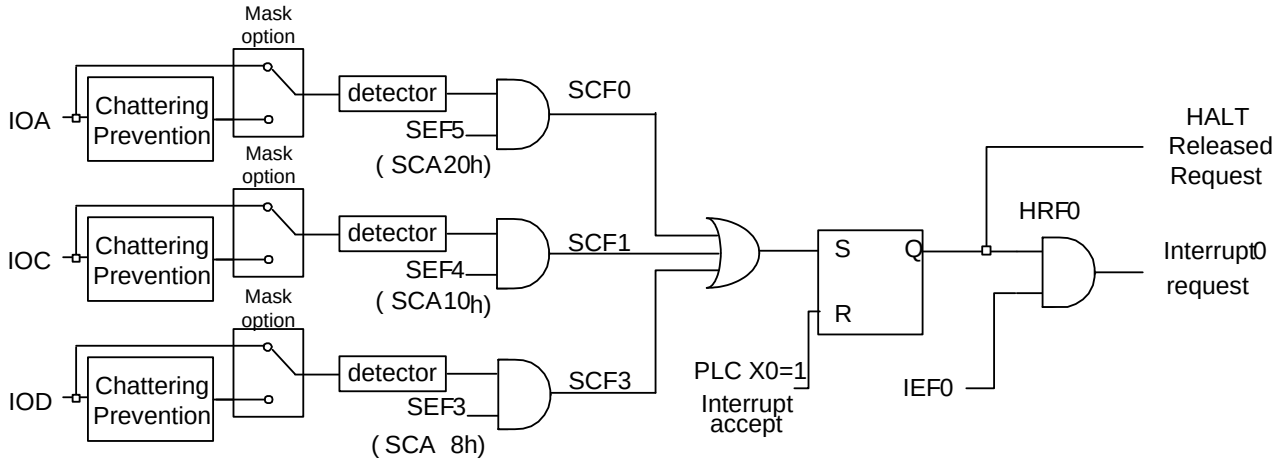
2. Switch enable flag 4 (SEF4)
當 switch enable flag 4 (SEF4)設定為 1 之後，IOC port 輸入腳位上的任何信號產生有效改變就能讓 MCU 產生 HALT release。

3. Switch enable flag 3 (SEF3)
當 switch enable flag 3 (SEF3)設定為 1 之後，IOD port 輸入腳位上的任何信號產生有效改變就能讓 MCU 產生 HALT release。

下表說明 control register 1 (CTL1)中每一個 flag 的功能：

Bit5	Bit 4	Bit3
Switch enable flag 5 (SEF5)	Switch enable flag 4 (SEF4)	Switch enable flag 3 (SEF3)
Enables the halt release caused by the signal change on IOA port (HRF0)	Enables the halt release caused by the signal change on IOC port (HRF0)	Enables the halt release caused by the signal change on IOD port (HRF0)
Write only	Write only	Write only

下圖說明 control register 1 (CTL1)與其他信號之間的關係：



上圖中的 detector 代表高電位偵測功能(若 TM87ML28H 提供一組 IOA,E code option 選擇為 P-H 模式則 detector 內會先作反相)，接下來會詳細說明 CTL1 在不同狀態下的使用方式。

1-12-1-1. 產生 HALT release 的設定方式

在 control register 1 (CTL1) flags 設定為 1 之後，IOA, IOC, IOD 等各個 IO port 的所有輸入腳位信號經過 OR 邏輯閘之後的信號發生變化(選用 chattering prevention 功能)，或是有任何輸入信號變成高電位(選用高電位偵測功能)時 MCU 就會產生 HALT release，並將 start condition flag 0, 1, 3 (SCF0, SCF1, SCF3)設定為 1。

當 IO port 是選用高電位偵測功能來設定 start condition flag 時，只有在 IO port 的所有輸入腳位的信號都在低電位(若 I/O option 選擇為 P-H 模式則為高電位)時 MCU 才能進入 HALT mode。只要任何一個輸入腳位的信號變成高電位(若 I/O option 選擇為 P-H 模式則為低電位)就能將對應的 start condition flag 設定為 1。

當 IO port 是選用 chattering prevention 功能來設定 start condition flag 時，無論 IO port 的輸入腳位的信號如何都能讓 MCU 進入 HALT mode。只要所有輸入腳位信號的 "OR" 結果有變化時就能將對應的 start condition flag 設定為 1。*請參閱 2-5-1、2-5-3、2-5-4 的 IO port 結構圖。*

1-12-1-2. 產生 STOP release 的設定方式

當 IO port 選用不同的信號偵測功能來設定 start condition flag 時，MCU 產生 STOP release 的設定方式也有所不同。MCU 必需在 IO port 所有輸入信號都是低電位(若 I/O option 選擇為 P-H 模式則為高電位)的情況下才能進入 STOP mode。

當 IO port 是選用高電位偵測功能來設定 start condition flag 時，在將 control register 1 (SEF5, SEF4, SEF3)設定為 1 之後，IOA, IOC, IOD port 輸入腳位上有任何輸入信號變成 高電位(若 I/O option 選擇為 P-H 模式則為低電位)時就可以讓 MCU 產生 STOP release 並設定對應的 start condition flag。

當 IO port 是選用 chattering prevention 功能來設定 start condition flag 時，在 control register 1 (SEF5, SEF4, SEF3)以及 control register 4 (SRF6, SRF4, SRF3)同時都設定為 1 之後，IOA, IOC, IOD port 輸入腳位上有任何輸入信號變成高電位(若 I/O option 選擇為 P-H 模式則為低電位)時就可以讓 MCU 產生 STOP release。產生 STOP release 之後 MCU 會先進入 HALT mode，IO port 輸入腳位上的高電位信號必須持續維持一段時間(震盪器起振時間加上 Chattering Prevention 的時間　週期)，直到 chattering prevention 功能確認這個高電位信號並設定對應的 start condition flag 為止。

如果這個高電位信號在 start condition flag 尚未設定之前就變回低電位(若 I/O option 選擇為 P-H 模式則為高電位)，MCU 就會立即返回 STOP mode。

1-12-1-3. 中斷服務的設定方式

在 start condition flag 0, 1, 3 (SCF0, SCF1, SCF3)設定為 1 之後就會將 halt release request flag (HRF0) 設定為 1。

在這個情況下，如果 SIE*指令已經先將 interrupt enable flag 0 (IEF0) 設定為 1，那麼 MCU 就會接受這個中斷請求並且將 interrupt request flag 0 (interrupt 0)設定為 1，程式就會進入 IO port 的中斷程序。

請注意，MCU 在接受 IOA, C, D port 的中斷請求之後會自動將 control register 1 (SEF5, SEF4, SEF3)清除為 0，所以在下次使用之前必須再執行一次 SCA 指令來設定 CTL1 暫存器。

1-12-2. CONTROL REGISTER 2 (CTL2)

Control register 2 (CTL2)是由 halt release enable flags 1~7 (HEF1~7)等 flag 所組成，而執行 SHE 指令可以設定或是清除這些 flag。

下表說明 control register 2 (CTL2)中每一個 flag 的功能：

Halt release enable flag	HEF7	HEF6	HEF5	HEF4
Halt release condition	Enable the halt release caused by TMR3 underflow (HRF7)	Enable the halt release caused by RFC counter is controlled by CX pin if A-Type for RFC (HRF6)	Enable the halt release caused by key matrx scanning function(HRF5)	Enable the halt release caused by TMR2 underflow (HRF4)
Halt release enable flag	HEF3	HEF2	HEF1	
Halt release condition	Enable the halt release caused by pre-divider overflow (HRF3)	Enable the halt release caused by INT pin / SIO (HRF2)	Enable the halt release caused by TM1 underflow (HRF1)	

1-12-2-1. 產生 HALT release 的設定方式

在 halt release enable flag 1,4,7 (HEF1,4,7)設定為 1 之後，當 TMR1, 2, 3 發生 underflow 的時候就可以讓 MCU 產生 HALT release。

同樣的，當 halt release enable flag 2,3 (HEF2,3)設定為 1 之後，只要在 INT pin 的輸入信號產生上昇緣或是下降緣的變化或是 pre-divider 發生 overflow 的時候就可以讓 MCU 產生 HALT release。

在 halt release enable flag 5 (HEF5)設定為 1 之後，當每個掃描週期結束時或是在 KI1~4 腳位上經掃描電路偵測到高電位的輸入信號時(由 key matrix 掃描功能的相關指令所設定)就可以讓 MCU 產生 HALT release。
若 RFC 為 Type A 方提供 HEF6 使用在 CX control mode，在 halt release enable flag 6 (HEF6)設定為 1 之後，當 RFC 執行結束的時候就可以讓 MCU 產生 HALT release。

1-12-2-2. 產生 STOP release 的設定方式

MCU 進入 STOP mode 之後，CTL2 中只有 HEF2, HEF5 兩個 flag 才可以讓 MCU 產生 STOP release：

1. 在 halt release enable flag 2 (HEF2)設定為 1 之後，INT 輸入腳位上的任何信號變化或是由 SIO 所產生的 halt release request flag 2 (HRF2)會將 start condition flag 4 (SCF4)設定為 1。 此時 MCU 會產生 STOP release。
2. 在 halt release enable flag 5 (HEF5)設定為 1 之後，key matrix 掃描功能在 KI1~4 腳位上經掃描電路偵測到高電位的輸入信號後所產生的 halt release request flag 5 (HRF5)會將 start condition flag 8(SCF8)設定為 1。此時 MCU 會產生 STOP release。

   因為 LCD driver 在 STOP mode 不會送出掃描信號，所以 key matrix 掃描功能無法利用與 LCD 波形同步的掃描週期信號來產生 STOP release，在這情況下必須設定為 Normal key scanning mode 才能產生 STOP release。

1-12-3. CONTROL REGISTER 3 (CTL3)

Control register 3 (CTL3)是由八個可以啟動或是關閉程式中斷功能的 interrupt enable flags 0 ~ 7 (IEF0~7)所組成，執行 SIE*指令可以設定或是清除這些 interrupt enable flag。

下表說明 control register 3 (CTL3)中每一個 flag 的功能：

Interrupt enable flag	IEF7	IEF6	IEF5	IEF4
Interrupt request flag	Enable the interrupt request caused by TMR3 underflow (HRF7)	Enable the interrupt request caused by RFC counter is controlled by CX/CX2 pin (HRF6)	Enable the interrupt request caused by key matrix scanning function (HRF5)	Enable the interrupt request caused by TMR2 underflow (HRF4)
Interrupt flag		Interrupt 6	Interrupt 4	Interrupt 4
Interrupt enable flag	IEF3	IEF2	IEF1	IEF0
Interrupt request flag	Enable the interrupt request caused by predivider overflow (HRF3)	Enable the interrupt request caused by INT pin / SIO (HRF2)	Enable the interrupt request caused by TM1 underflow (HRF1)	Enable the interrupt request caused by IOA, IOC or IOD port signal changed (HRF0)
Interrupt flag	Interrupt 3	Interrupt 2	Interrupt 1	Interrupt 0

在程式設定 interrupt enable flag 之後，當上表中對應的中斷因子(HRFn)產生的時候就會向 MCU 提出一個中斷請求。

當 MCU 接受中斷請求並且進入中斷程序之後，會將相關的中斷因子(HRFn)以及 interrupt enable flag (IEFn)自動清除為 0。因此程式在結束中斷副程式之前必需重新設定 interrupt enable flag (IEFn)，以便等待下一次中斷因子的發生。

就算沒設定任何的 halt release enable flag (HEFn)，MCU 在接受中斷請求之後都可以產生 HALT release 並進入中斷程序，但是還是有一些因子需要另外設定對應的 Switch enable flag (SEFn),SIO release enable flag(SSREn), INT interrupt/release enable flag(INTFEN)(若 INT 與 SIO 共用 HRF)才能產生 Halt Release 並進入中斷程序。

只有下面三種中斷因子發生時可以讓 MCU 產生 STOP release 並進入中斷程序。

1. 在 interrupt enable flag 0 (IEF0) 設定為 1 之後，只要 IOA, IOC, IOD port 輸入腳位上的信號變成高電位(若 I/O option 選擇為 P-H 模式則為低電位)而且可以將 start condition flag 0, 1, 3 (SCF0, SCF1, SCF3)設定為 1， MCU 就會產生 STOP release。
2. 在 interrupt enable flag 2(IEF2) 設定為 1 之後，當 INT pin 的輸入信號改變就可以讓 MCU 產生 STOP release。
3. 在 interrupt enable flag 5(IEF5) 設定為 1 之後，當 key matrix 掃描功能在 KI1~4 腳位上經掃描電路偵測到高電位的輸入信號後就可以讓 MCU 產生 STOP release。

1-12-4. CONTROL REGISTER 4 (CTL4)

Control register 4 (CTL4)是由三個 stop release enable flag 3, 4, 6 (SRF3, 4, 6)所組成， 執行 SRE 指令可以設定或是清除這些 stop release enable flag。

下表說明 control register 4 (CTL4)中每一個 flag 的功能：

Stop release enable flag	SRF6	SRF4	SRF3
Stop release request	Enable the stop release caused by signal change on IOA when chattering prevention option is enabled	Enable the stop release caused by signal change on IOC when chattering prevention option is enabled	Enable the stop release caused by signal change on IOD when chattering prevention option is enabled

當 IOA, IOC, IOD port 選用 chattering prevention 功能來設定 start condition flag 時，一定要設定 Control register 4 (CTL4)之後才會產生 STOP release。

在 control register 1 (SEF5, SEF4, SEF3)以及 control register 4 (SRF6, SRF4, SRF3)同時都設定為 1 之後， IOA, IOC, IOD 等 port 輸入腳位上有任何輸入信號變成高電位時就可以 讓 MCU 產生 STOP release。

Example:

這個範例是將 IO port, KI1~4 and INT pin 設定成 MCU 可以產生 STOP release 的因子，IOC 以及 IOD 都設定為輸入模式，IOC 選用 Chattering prevention 功能，IOD 沒有選用 Chattering prevention 功能。

```
PLC     $25     ; 清除 HRF0, HRF2, HRF5.
SHE     $24     ; 設定 HEF2, HEF5
SCA     $18     ; 設定 SEF4, SEF3
SRE     $10     ; 設定 SRF4，因為 IOC 使用 Chattering prevention 功能
STOP            ; Enters the stop mode.

        ……      ; STOP release

MSC     $10     ; 檢查是否因為 INT pin 產生 STOP release
MSB     $11     ; 檢查是否因為 IOC 或是 IOD port 產生 STOP release
MCX     $12     ; 檢查是否因為 key matrix 掃描功能產生 STOP release
```

1-12-5. CONTROL REGISTER 5 (CTL5)

當 RFC 為 TYPE B 須設定 Control register 5 (CTL5)兩個 switch enable flag 0, 1 (SEF0, 1)方可啟動 CX,CX2 halt release 功能，執行 SCX 指令可以設定或是清除這些 switch enable flag。

下表說明 control register 5 (CTL5)中每一個 flag 的功能：

Bit 1	Bit 0
Switch enable flag 1 (SEF1)	Switch enable flag 0 (SEF0)
Enables the halt release caused by the end of control signal when RFC counter is controlled by CX2 (SCF12)	Enables the halt release caused by the end of control signal when RFC counter is controlled by CX (SCF11)
Write only	Write only

如果 RFC counter 是由 CX2 pin 的輸入信號來控制啟動或是關閉時，當程式執行 SCX 指令將 switch enable flag 1 (SEF1)設定為 1 之後，在 CX2 pin 的控制信號結束時就會將 start condition flag 12 (SCF12)設定為 1 並使得 MCU 產生 HALT release。

如果 RFC counter 是由 CX pin 的輸入信號來控制啟動或是關閉時，當程式執行 SCX 指令將 switch enable flag 0 (SEF0)設定為 1 之後，在 CX pin 的控制週期結束時就會將 start condition flag 11 (SCF11)設定為 1 並使得 MCU 產生 HALT release。

1-13. HALT MODE

當 MCU 進入 HALT mode 時，除了程式停止執行之外，其他的功能都會繼續正常的運作。在這個模式下可以降低 MCU 一部分的耗電量。

在程式執行 HALT 指令之後，如果沒有任何可以使得 MCU 產生 HALT release 的因子存在時(SCF0, SCF1, SCF3, SCF11, SCF12, HRF1 ~ 7(依 RFC TYPE A/B 提供 HRF6/SCF11,12))，MCU 就會進入 HALT mode。

下面四種情況可以讓 MCU 產生 HALT release：

1. 發生中斷

當中斷發生後 MCU 會自動產生 HALT release，在程式執行完中斷服務副程式並執行 RTS 指令之後，MCU 會重新進入 HALT mode。

如果 MCU 是因為發生中斷而產生 HALT release 時，相對應的中斷因子(halt release signal, HRFn)會自動清除為 0。

2. 執行 SCA 指令將 SEF5, 4, 3 flag 設定為 1 之後，IOA, IOC, IOD 等各個 IO port 的所有 輸入腳位信號經過 OR 邏輯閘之後的信號發生變化(選用 chattering prevention 功能)，或是有任何輸入信號變成高電位(選用高電位偵測功能)時 MCU 就會產生 HALT release。

當 IO port 是選用高電位偵測功能來設定 start condition flag 時，只有在 IO port 的所有 輸入腳位的信號都在低電位時 MCU 才能進入 HALT mode。只要任何一個輸入腳位的信號變成高電位時 MCU 就會產生 HALT release。

當 IO port 是選用 chattering prevention 功能來設定 start condition flag 時，無論 IO port 的輸入腳位的信號如何都能讓 MCU 進入 HALT mode。只要各個 IO port 的所有輸入腳位信號經過 OR 邏輯閘處理之後的輸出信號發生變化時 MCU 就會產生 HALT release。

3. 執行 SCX 指令將 SEF1, 0 flag 設定為 1 之後，而 RFC counter 是由 CX2 或是 CX pin 的輸入信號來控制啟動或是關閉時，在 CX2 或是 CX pin 的控制信號結束時 MCU 就會產生 HALT release。

4. 當 SHE 指令所設定可以使得 MCU 產生 HALT release 的因素發生時 (HRF1~ 7)， MCU 就會產生 HALT release。

如果 MCU 不是因為發生中斷而產生 HALT release 時，程式必須利用 MAF, MSB, MSC, MCX, MDX 以及 MMW Ry,$7(TM87ML28 讀取 SSCF3~1&INTF)等指令來找出使得 MCU 產生 HALT release 的因子(halt release flag, HRFn)，然後執行 PLC 指令將這些因素(halt release flag, HRFn)清除，以便下一次進入 HALT mode。

在程式尚未將這些因子(halt release flag, HRFn)清除之前，MCU 將無法進入 HALT mode。

1-14. STOP MODE

當 MCU 進入 STOP mode 時，全部的功能都會停止運作。在這個模式下 MCU 本身將不會有任何的耗電。

在 STOP 模式下，所有的 SEGn 以及 COMn 等輸出腳位都會輸出 GND 位準的信號，直到 MCU 產生 STOP release 之後才會恢復正常的輸出波形。

在 MCU 進入 STOP mode 之後會自動將 BCF flag 設定為 1 並且啟動電力備援模式，以便 讓 MCU 離開 STOP mode 的時候可以順利啟動震盪器。因此建議程式在 MCU 產生 STOP release 之後視情況關閉電力備援模式，以減少 MCU 的耗電。

在 MCU 準備進入 STOP mode 之前，程式必須先完成下面幾項工作(視實際需要才做設定):

1. 如果 IOA, IOC 以及 IOD 已經選用 chattering prevention 功能時，執行 SRE 指令來設定相關的 SRF6, 4, 3 flags。

2. 執行 SCA 指令設定 SEF5, 4, 3 flags，以便 IOA, IOC, IOD port 可以產生 STOP release。

3. 將 INT/SIO, Key matrix 掃描功能，以及 IOA, IOC, IOD 等中斷請求(IEF0, 2, 5)設定為產生 STOP release 的因子，而且只有外部中斷因子才能產生 STOP release。

4. 確認所有可以使 MCU 產生 STOP release 的 IOA, IOC, IOD 等輸入腳位上的信號都是低電位(若 I/O option 選擇為 P-H 模式則為高電位)。

5. 將 stop release signal (HRF0, HRF5 以及 HRF2)清除為 0。

下面幾個情況發生時可以使 MCU 產生 STOP release：

A. 當 IOA, IOC 或是 IOD 選用高電位偵測功能時不需設定 SRE，port 上的任何一個輸入腳位上的信號變成高電位(若 I/O option 選擇為 P-H 模式則為低電位)時即可產生 STOP release。

當 IOA, IOC 或是 IOD port 選用 chattering prevention function 時，MCU 會在 SRE 指令所設定的 IO port 上的信號變成高電位(若 I/O option 選擇為 P-H 模式則為低電位)時起振產生 clock 偵測，而且維持這個電位直到 MCU 可將 SCF0, 1, 3 flag 設定為 1 的為止(除受原本 SCC 指令設定 Cch 週期長短影響外，此時尚需加上震盪電路的起振時間)才會產生 STOP release，否則 MCU 就會在輸入腳位皆變成低電位(若 I/O option 選擇為 P-H 模式則為高電位)信號的時候重新進入 STOP mode。

B. 當 INT pin 上的輸入信號依 code option 產生 falling/rising edge 時。

C. 當 key matrix 掃描功能在 KI1~4 腳位上透過掃描電路偵測到高電位的輸入信號時。

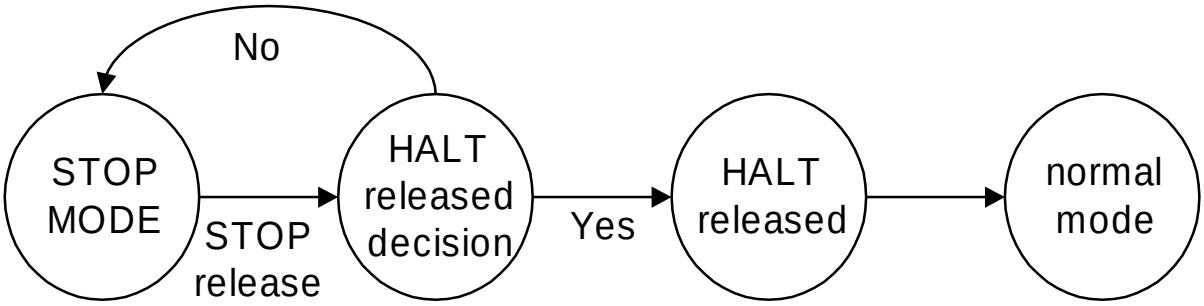
D. 當 I2C standby 時 SDA 產生 low pulse。

E. 當 SPI Slave Mode SSB or SCK(若 SSB no use)產生轉態 。

當 MCU 進入正常操作模式之後，程式可以利用 MSB, MSC 或是 MCX 指令來找出使得 MCU 產生 STOP release 的因子(halt release flags, HRF0, 2, 5)，然後執行 PLC 指令將這些因子(halt release flag, HRF0, 2, 5)清除為 0，以便下一次可以重新進入 STOP mode。在尚未將這些因子(halt release flag, HRF0, 2, 5)清除之前，MCU 將無法進入 STOP mode。

當外部中斷發生時，在 MCU 接受中斷請求之後就會產生 STOP release 然後直接執行中斷服務副程式，在 MCU 產生 STOP release 的時候會將所有的中斷因子 halt release signals, HRFn)清除為 0。程式在執行中斷服務副程式的 RTS 指令後，MCU 會重新進入 STOP mode。

下圖說明 MCU 如何在產生 STOP release 後進入正常操作模式：


No
STOP
MODE
STOP
release
HALT
released
decision
Yes
HALT
released
normal
mode

1-15. 電力備援模式(BACK UP MODE)

當系統上的高耗電量功能啟動時會使得系統上的電源電壓產生很大的變動，4BIT 系列 MCU 的電力備援模式可以讓 MCU 在這個情況下仍然能維持 MCU 正常的動作。但是開啟 電力備援模式之後會增加 MCU 的耗電量，因此建議使用者除非必要，請儘量關閉電力備援模式以達到省電的效能。

Back up flag (BCF)可以指示目前 MCU 是否已經啟動電力備援模式，當 BCF flag 設定為 1 時，表示 MCU 已經啟動動電力備援模式。執行 SF 指令可以啟動電力備援模式，執行 RF 指令可以關閉電力備援模式。

在啟動電力備援模式後 MCU 的部份功能會做自動的調整或是改變其特性，其說明如下：

1. 如果 MCU 選用 32.768 KHz Crystal oscillator 作為低速時鐘振盪器時，啟動電力備援模式後振盪器會加大驅動能力以避免輸出頻率被電源電壓干擾甚至使得時鐘振盪器停振，但是會增加 MCU 的耗電量。

2. 當 MCU 進入 STOP mode 之後除了部分有內部 BAK 穩壓產品可以提供 Mask Option 選擇維持 BCF=0 繼續啟動 BAK 穩壓之外，一般會自動開啟電力備援模式並將 BCF flag 設定為 1。

3. 在使用 3V 電源模式下，啟動電力備援模式後 MCU 內部功能會在 VBAT 的電源電壓下工作；關閉電力備援模式之後，若 code option 是選用 “BCF=0 時 BAK=VL1” 等非”EXTV” 或” BCF=0 時 BAK=VBAT”的功能時，MCU 的內部功能會在較低的操作電壓下(VL1/VL2/…)工作，以達到省電的目的。

此時需注意，RC 震盪器輸出頻率會因為 BAK 電壓的改變而產生變動，此外若 BAK 的電壓是由內外部穩壓線路提供時，其提供的電壓值必須高於 MCU 的最低工作電壓，以免造成 MCU 無法正常工作。

因為 FAST CLOCK 更須注意 MCU 工作頻率&耗電限制，故必須在足夠的工作電壓和電流驅動能力下才能執行 FAST 模式，因此若在關閉電力備援模式時無足夠工作電壓和電流驅動能力請勿仍停留在 FAST 模式或是進入 FAST 模式，以免造成 MCU 的誤動作。

在不同的電源模式下，MCU 在不同的狀態下會自動設定不同的電力備援模式，一般在起始(除了 EXTVMode)或是 STOP Mode 下皆是設定 BCF=1，但對有提供 BAK 穩壓的部分產品若有提供 Mask Option 則可自行選擇 BCF=1/0 當 BCF=0 : BAK=穩壓時，如下表所示：

1.5V 電源模式或是 3V 以上電源模式(非”EXTV” & BCF=0 => BAK=VBAT)：

4BIT Series status	BCF flag status
Initial reset cycle	BCF = 1 (hardware controlled)
After initial reset cycle	BCF = 1 (hardware controlled)
Executing SF 2h instruction	BCF = 1
Executing RF 2h instruction	BCF = 0
HALT mode	Previous state
STOP mode	BCF = 1 (hardware controlled)

MCU 的功能	BCF = 0	BCF = 1
32.768 KHz Crystal Oscillator	Small Driver	Large Driver
BAK 腳位的電壓(提供給時鐘振盪器以及內部功能使用)	VBAT	VBAT

1.5V 電源模式或是 3.0V 以上電源模式(BCF=0 時 BAK<VBAT”)：

4BIT Series status	BCF flag status
Initial reset cycle	BCF = 1 by hardware controlled or 1/0 by Mask Option
After initial reset cycle	BCF = 1 by hardware controlled or 1/0 by Mask Option
Executing SF 2h instruction	BCF = 1
Executing RF 2h instruction	BCF = 0
HALT mode	Previous state
STOP mode	BCF = 1 by hardware controlled or 1/0 by Mask Option

MCU 的功能	BCF = 0	BCF = 1
32.768 KHz Crystal Oscillator	Small Driver	Large Driver
BAK 腳位的電壓(提供給時鐘振盪器以及內部功能使用)	VL1/VL2/VREG…	VBAT

EXTV

4BIT Series status	BCF flag status
Initial reset cycle	BCF = 0 (hardware controlled)
After initial reset cycle	BCF = 1 (hardware controlled)
Executing SF 2h instruction	BCF = 1
Executing RF 2h instruction	BCF = 0
HALT mode	Previous state
STOP mode	BCF = 1 (hardware controlled)

MCU 的功能	BCF = 0	BCF = 1
32.768 KHz Crystal Oscillator	Large Driver	Large Driver
BAK 腳位的電壓(提供給時鐘振盪器以及內部功能使用)	VBAT	VBAT

因 EXTV 為外接電源而且無須考量省電的應用，故 BCF 值在此模式並無任何意義!

第二章 Control Function

2-1. INTERRUPT FUNCTION

4BIT 系列 MCU 總共有八個中斷因子：三個外部中斷因子以及五個內部中斷因子。

當 MCU 接受中斷請求之後會立刻停止目前執行的程式，然後跳到對應的中斷位址去執行中斷服務副程式。

下表說明各個中斷因子的來源，中斷位址以及相關的 flag：

Interrupt source	INT pin/SIO	IOA,C,D port	TMR1 underflow	Pre-divider overflow	TMR2 underflow	Key matrix Scanning	Een of RFC counter controlled signal	TMR3 underflow
Interrupt vector	010h	014h	018h	01Ch	020h	024h	028h	02Ch
Interrupt enable flag	IEF2	IEF0	IEF1	IEF3	IEF4	IEF5	IEF6	IEF7
Interrupt request flag	HRF 2	HRF 0	HRF 1	HRF 3	HRF 4	HRF 5	HRF 6	HRF 7

請注意，MCU 在下列幾種情況下會暫時停止處理所有的中斷請求：

1. 執行所有八個 machine cycle 的指令的時候。

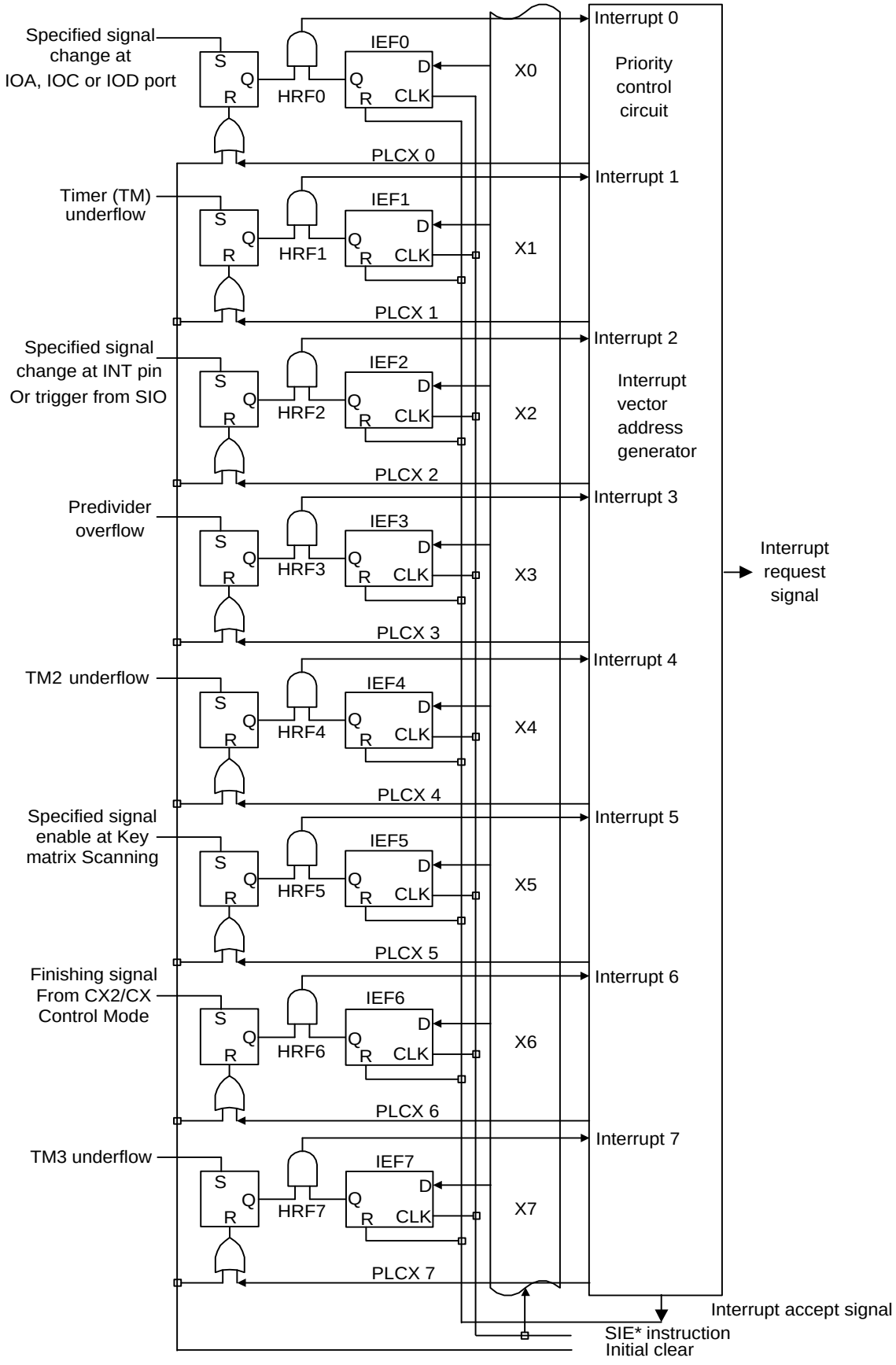
2. 執行 SRX、SRY、SLZ、SPBK 指令以及這些指令的下一個位址的指令時。

3. 執行 8 個 machine cycle 的 CAC、JAC 指令時。

4. 執行 CPHL、CPHLH、CPZR、CPZRH 指令以及這些指令的下一個位址的指令時(目前 TM8726 仍須由軟體暫時停止中斷請求)。

5. 執行 SIE，RTS 指令時。

下圖是中斷功能的控制線路：



2-1-1. 中斷請求以及中斷服務

2-1-1-1. 外部中斷因子

外部中斷因子有：INT pin,SIO, IOA, IOC or IOD ports 以及 Key matrix 掃描功能的輸入腳位。

1. INT 輸入腳位/SIO 的中斷請求
使用者可以利用光罩選擇的方式選擇 INT 腳位上的輸入信號是以上昇緣或是下降緣來對 MCU 產生中斷。
在 interrupt enable flag 2 (IEF2)設定為 1 之後(若產品有提供 SIO 與 INT 共用 HRF2，則需再設定 SSRE3~1&INTFEN 啟動)，INT 輸入腳位上的信號改變或是 SIO 產生觸發條件(詳見 SIO 功能說明)就會將 HRF2 flag 設定為 1 並向 MCU 提出中斷請求(interrupt 2)。MCU 在接受這中斷請求後 就會跳到 10h 的程式位址去執行 interrupt 2 的中斷副程式。
INT 輸入腳位上的信號改變後還需維持至少一個 machine cycle 以上的時間才能確保產生中斷信號。

2. IOA, C, D port 的中斷請求
在執行 SCA 指令設定 control register 1 (SEF5, SEF4, SEF3)之後，IOA, IOC, IOD 等 各個 IO port 的所有輸入腳位信號經過 OR 邏輯閘之後的信號發生變化時(選用 chattering prevention 功能)，或是有任何輸入信號變成高電位(選用高電位偵測功能)時就會將 interrupt request signal (HRF0)設定為 1。
如果 SIE*指令已經先將 interrupt enable flag 0 (IEF0) 設定為 1，那麼 MCU 就會接受這個中斷請求(interrupt 0)。MCU 接受這中斷請求後就會跳到 14h 的程式位址去執行 interrupt 0 的中斷副程式。

3. Key matrix 掃描功能的中斷請求
在 halt release enable flag 5 (HEF5)設定為 1 之後，當每個掃描週期結束時或是在 KI1~4 腳位上經掃描電路偵測到高電位的輸入信號時(由 key matrix 掃描功能的相關指令所設定)就會將 interrupt request signal (HRF5)設定為 1。
如果 SIE*指令已經先將 interrupt enable flag 5 (IEF5) 設定為 1，那麼 MCU 就會接受這個中斷請求(interrupt 5)。MCU 接受這中斷請求後就會跳到 24h 的程式位址去執行 interrupt 5 的中斷副程式。

2-1-1-2. 內部中斷因子

內部中斷因子有：timer 1 (TMR1), timer 2 (TMR2), timer 3 (TMR3), RFC counter 以及 pre-divider。

1. Timers (TMR1, 2, 3) 的中斷請求
當 TMR1, 2, 3 發生 underflow 的時候會將 interrupt request signal (HRF1, 4, 7)設定為 1。
如果 SIE*指令已經先將 interrupt enable flag 1, 4, 7 (IEF1, 4, 7) 設定為 1，那麼 MCU 就會接受這個中斷請求(interrupt 1, 4, 7)。MCU 接受這中斷請求後就會跳到 18h, 20h, 2Ch 的程式位址去執行 interrupt 1, 4, 7 的中斷副程式。

2. Pre-divider 的中斷請求
當 pre-divider 發生 overflow 的時候會將 interrupt request signal (HRF3)設定為 1。
如果 SIE*指令已經先將 interrupt enable flag 3 (IEF3) 設定為 1，那麼 MCU 就會接受這個中斷請求(interrupt 3)。MCU 接受這中斷請求後就會跳到 1Ch 的程式位址去執行 interrupt 3 的中斷副程式。

3. RFC counter (由 CX/CX2 控制的模式) 的中斷請求
當 RFC counter 是由 CX 或是 CX2 來控制啟動或是關閉時，當 CX 以及 CX2 的控制信號結束的時候會將 interrupt request signal (HRF6)設定為 1。

如果 SIE*指令已經先將 interrupt enable flag 6 (IEF6) 設定為 1，那麼 MCU 就會接受這個中斷請求(interrupt 6)。MCU 接受這中斷請求後就會跳到 28h 的程式位址去執行 interrupt 6 的中斷副程式。

2-1-2. 中斷優先次序

當所有的中斷請求同時發生時，MCU 會根據中斷優先次序先處理優先權最高的中斷請求(pre-divider)，其他的中斷請求則會暫時不做處理。

各個中斷因子的優先次序如下表所示：

Interrupt factor	INT pin	IOA,C,D port	TMR1 underflow	Pre-divider Overflow	TMR2 Underflow	Key matrix Scanning	End of RFC counter controlled signal	TMR3 underflow
Interrupt priority	7th	6th	2nd	1st	3rd	8th	5th	4th

下面的範例中假設所有的中斷請求都同時發生，而且所有的 IEF flag 也都已設定為 1，所有的 IOC port 都設定為輸入模式。

```
PLC     $FF         ; 清除所有的 HRFn flag
SCA     $10         ; 啟動 IOC port 的 switch enable flag
SIE*    $FF         ; 設定所有的 interrupt enable flag

;... ... ... ... ...    ; 所有的中斷請求同時發生

                    ; MCU 接受 predivider overflow 的中斷請求，
                    ; 然後完成中斷服務。

SIE*    $F7         ; 設定所有的 interrupt enable flag (predivider 除外)

                    ; MCU 接受 TMR1 underflow 的中斷請求，
                    ; 然後完成中斷服務。

SIE*    $F5         ; 設定所有的 interrupt enable flag (predivider. TMR1 除外)

                    ; MCU 接受 TMR2 underflow 的中斷請求，
                    ; 然後完成中斷服務。

SIE*    $E5         ; 設定所有的 interrupt enable flag (predivider. TMR1, TMR2
                    ; 除外)

                    ; MCU 接受 TMR3 underflow 的中斷請求，
                    ; 然後完成中斷服務。

SIE*    $65         ; 設定所有的 interrupt enable flag (predivider. TMR1, TMR2,
                    ; TMR3 除外)
                    ; MCU 接受 CX/CX2 控制 RFC counter 的中斷請求，
                    ; 然後完成中斷服務。

SIE*    $25         ; 設定所有的 interrupt enable flag (predivider. TMR1, TMR2,
                    ; TMR3, RFC counter 除外)
```

```
                        ; MCU 接受 IOC port 的中斷請求，然後完成中斷服務。

SIE*    $24             ; 設定所有的 interrupt enable flag (predivider. TMR1, TMR2,
                        ; TMR3, RFC counter, IOC port 除外)

                        ; MCU 接受 INT pin 的中斷請求，然後完成中斷服務。

SIE*    $20             ; 設定所有的 interrupt enable flag (predivider. TMR1, TMR2,
                        ; TMR3, RFC counter, IOC port, INT pin 除外)

                        ; MCU 接受 Key matrix 掃描功能的中斷請求，
                        ; 然後完成中斷服務。

                        ; 至此，所有的中斷請求都已經處理完畢。
```

2-1-3. 中斷服務

當 MCU 接受中斷請求之後，會自動暫停目前正在執行的程式並跳到中斷位址去執行中斷副程式，此時 MCU 會進行下面這些動作：

1. 將中斷請求發生時正在執行的指令位址儲存到 stack register (STACK)。
2. 將中斷請求對應的中斷位址載入 program counter (PC)。
3. 將中斷請求對應的 interrupt request flag (HRFn)清除為 0，並將所有的 interrupt enable flags (IEFn)也清除為 0。

當 MCU 接受中斷請求之後，程式會依照下面的程序執行：

```
Instruction 1       ; 當程式執行這個指令時 MCU 接受中斷請求，MCU 將目前的程式位址
                    ; 存入 STACK。
NOP                 ; 將對應的中斷位址載入 PC 並執行一個 NOP 指令。
Instruction A       ; 程式跳到中斷服務副程式執行。
Instruction B
Instruction C
 .............
RTS                 ; 中斷服務副程式執行完畢。
Instruction 1*      ; MCU 重新執行中斷請求發生時正在執行的指令。
Instruction 2
```

當中斷服務副程式開始執行的時候，所有的 interrupt enable flags (IEF0 ~ IEF7)都會被清除為 0，程式如需繼續處理其他的中斷請求時，必須在中斷服務副程式中重新執行 SIE*指令來設定這些 flag。

2-2. RESET FUNCTION

MCU 在接收到下面四種方式所產生的 reset 信號後就會進入 RESET 狀態：MCU 的 power on reset，RESET(B) pin，external key reset，low voltage reset，watch dog timer 發生 overflow。

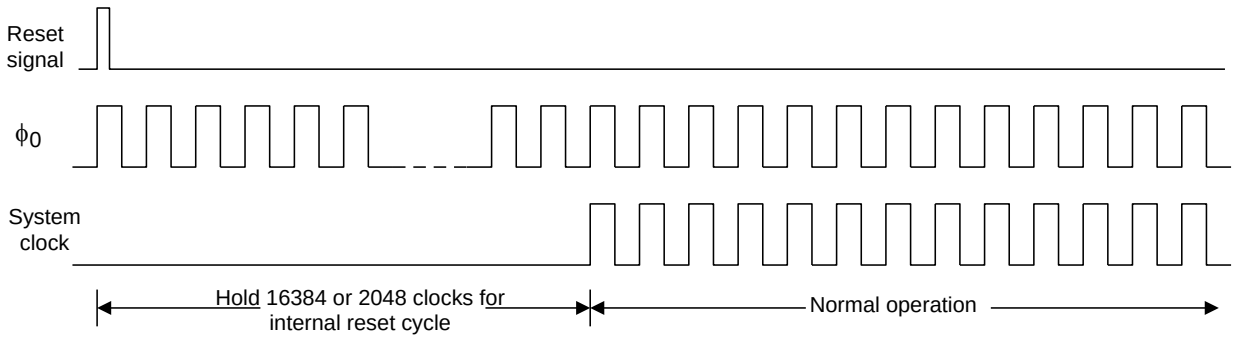
下表說明 MCU 進入 RESET 狀態之後，各個 flag 以及功能的起始狀態：

Program counter	(PC)	Address 0000H
Start condition flags	(SCF1~12)	0
Backup flag	(BCF)	1/0(0:EXTV or by Mask Option)
Stop release enable flags	(SRF3,4,6)	0
Switch enable flags	(SEF0,1,3,4,5)	0
Halt release request flags	(HRF 0~7)	0
Halt release enable flags	(HEF1~7)	0
Interrupt enable flags	(IEF0~7)	0
BZ,BZB pin output		DC 0
I/OB~D,F,G pins	pull-low 電阻	Turn On (if use by Mask Option)
I/OA,E pins	pull-high/low 電阻	Turn On (if use by Mask Option)
Input/output ports	(PORT I/OA~G)	Input mode
I/OA,C,D port chattering clock	Cch	PH10*
EL panel driver pumping clock source and duty cycle	Celp	PH0, duty cycle is 1/4
EL panel driver clearing clock source and duty cycle	Celc	PH8, duty cycle is 1/4
EL plant driver output pins	ELC, ELP	Output 0
Frequency generator clock source and duty cycle	Cfq	PH0, duty cycle is 1/4, 停止動作
Resistor frequency converter	(RFC)	停止動作, RFC0~5 output 0
LCD driver output waveform		全部都是 OFF 波形
Key matrix scanning function		停止動作, KO1~16=1
Timer 1/2/3		停止動作
Watch dog timer	(WDT)	停止動作
Watch dog timer enable flag	(WDF)	WDF = 0
系統時鐘的 clock source (Dual clock 模式)	(BCLK)	SLOW clock
SDA,SCL,RXD,TXD,SIOX,SIOY,SSB pins	pull-high 電阻	Turn On
SCK pin	pull-high/low 電阻	Turn On(if use by Mask Option)

Note: PH10: pre-divider 的第 10 級輸出信號。

MCU 接收到 reset 信號之後就會進入 RESET 狀態，然後開始進行 RESET 時間長度的計數，等到 RESET 時間長度的計數結束後 MCU 就會進入正常操作狀態。

利用 code option 可以選擇兩種 RESET 時間長度的計數，一種是 16384 個 PH0 的 clock cycle (PH15/2)，另一種是 2048 個 PH0 的 clock cycle (PH12/2)。



下面將詳細說明各種產生 reset 的方式：

2-2-1. POWER ON RESET

當外部的電源送到 MCU 的 VBAT 電源腳位上的時候，或是電源電壓降到 0.6V 或 10%VBAT(若有支援 10ms Power On Time Spec)以下之後再重新回升到正常工作電壓時 MCU 就會產生一個 power on reset 的信號，使得 MCU 進入 RESET 狀態並開始進行 RESET 時間長度的計數。

等到 RESET 時間長度的計數結束之後 MCU 就會進入正常操作狀態。

可以利用 code option 決定是否選用 power-on reset 功能。

2-2-2. RESET PIN

TM87,85/89 系列當一個高/低電位的輸入信號送到 RESET/RESETB 輸入腳位後就會使得 MCU 進入 RESET 狀態。

RESET/RESETB 輸入腳位上有一個內建的 pull-down/up 電阻。如果在 RESET/RESETB 腳位以及電源 VDD/GND 之間接一個 0.1 uF 的電容，這樣可以利用這個 RC 線路在 RESET/RESETB 腳位上產生 power on reset 信號。

MCU 對於 RESET/RESETB 腳位上的輸入信號有兩種處理方式，一種是 level reset 方式，另一種是 pulse reset 方式。這兩種方式可以利用 code option 來選擇。說明如下：

2-2-2-1. LEVEL RESET

在選用這種方式時，當一個高/低電位的信號送到 RESET/RESETB 腳位後，MCU 會進入 RESET 狀態，但是不會立刻進行 RESET 時間長度的計數，要一直等到 RESET/RESETB 腳位上的信號變成低/高電位之後才會開始。

等到 RESET 時間長度的計數結束之後 MCU 就會進入正常操作狀態。

2-2-2-2. PULSE RESET

在選用這種方式時，當一個高/低電位的信號送到 RESET/RESETB 腳位後，MCU 會進入 RESET 狀態，而且會立刻進行 RESET 時間長度的計數。

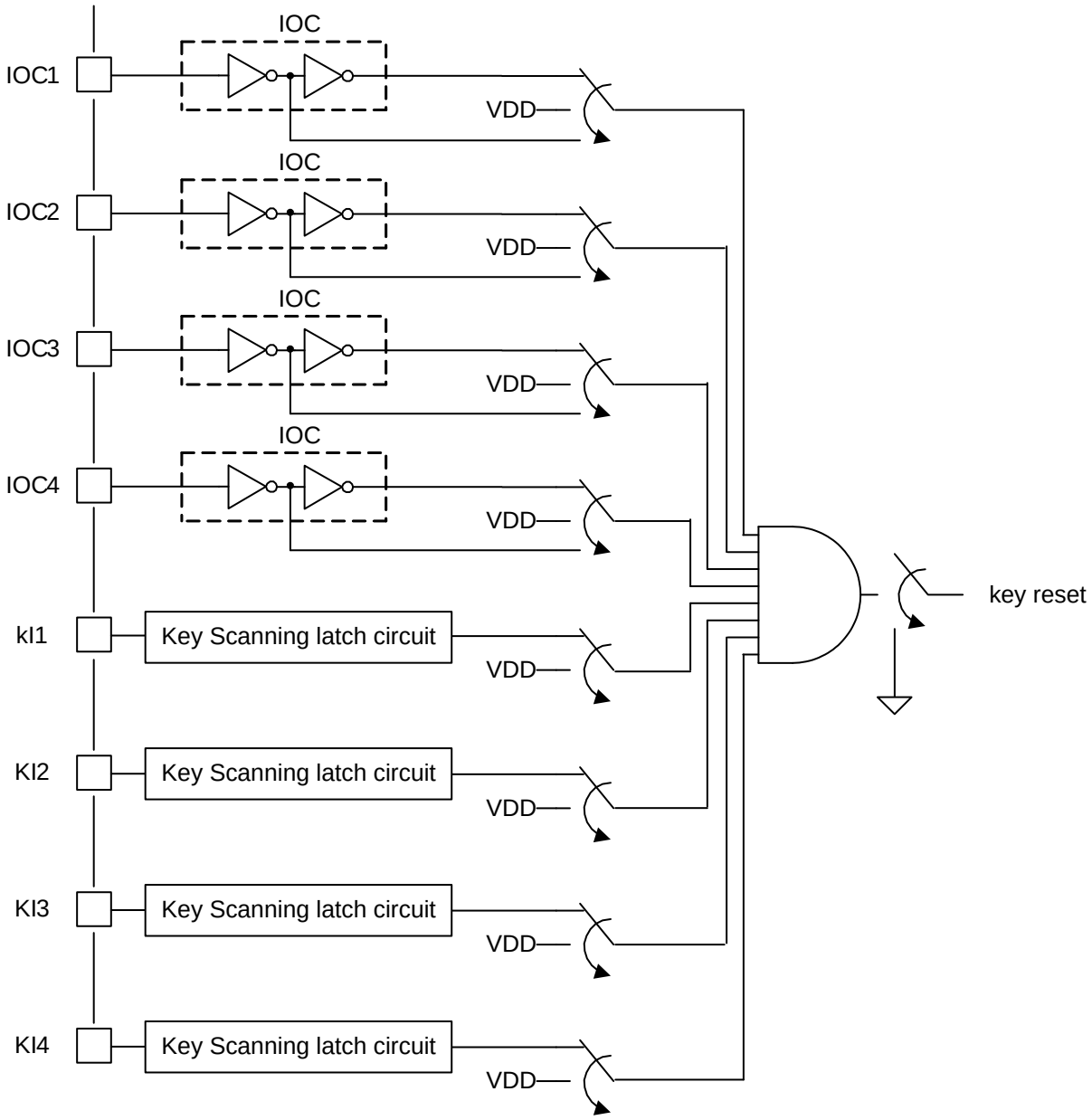
不管 RESET/RESETB 腳位上的信號是否已經變成低/高電位，等到 RESET 時間長度的計數結束之後 MCU 都會進入正常操作狀態。

2-2-3. EXTERNAL KEY RESET

在選用這種方式時，當幾個特定的輸入腳位上同時同時輸入高電位(若 I/O option 選擇為上拉電阻模式則為低電位)信號時可以產生 MCU 的 reset 信號，這些特定的輸入腳位是利用 code option 來選擇那幾個 IOC port 的輸入腳位或是 key matrix 掃描功能的輸入腳位(KI1 ~ KI4)組合而成。

當這些特定的輸入腳位(已選用的 IOC 輸入腳位以及 KI 的輸入腳位)上同時輸入高電位(若 I/O option 選擇為上拉電阻模式則為低電位)的信號時，MCU 就會進入 RESET 狀態。等到有任何一個輸入信號變成低電位之後 MCU 就會開始進行 RESET 時間長度的計數，等到 RESET 時間長度的計數結束之後 MCU 就會進入正常操作狀態。

下圖說明這個功能的線路架構：

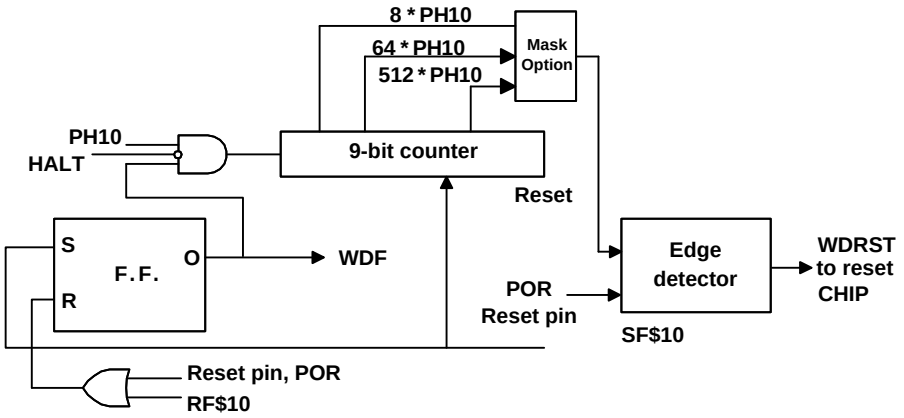


2-2-4. WATCH DOG TIMER 發生 OVERFLOW 時

Watch dog timer 是由一個 9-bit 的 counter 所組成，而 timer 的 clock source 是來自 pre-divider 的第十級輸出信號(PH10)。

當 watch dog timer 發生 overflow 之後，MCU 會馬上進入 RESET 狀態並開始 RESET 時間長度計數，在 RESET 時間長度計數結束之後就會進入正常操作模式。

下圖說明 watch dog timer 的線路架構：



除了選擇 WDT option 為"ALWAYS ENABLE"時 watch dog timer 會固定開啟&watch dog timer enable flag (WDF) 固定為 1 外， 如果 MCU 是因為 power on reset，reset pin，external key reset 等方式而進入 RESET 狀態時，MCU 會將 watch dog timer 關閉，watch dog timer enable flag (WDF)清除為 0。

如果 MCU 是因為 watch dog timer 發生 overflow 而進入 RESET 狀態時，watch dog timer 仍然會持續動作(但計數器會被維持清除直到系統 RESET 結束)，watch dog timer enable flag (WDF) 不會被清除為 0，而且 pre-divider 的前十級輸出信號(PH0 ~ PH10)不會被歸零。

執行 SF $10 指令後會先將 watch dog timer 歸零後啟動，並將 watch dog timer enable flag (WDF) 設定為 1。在 watch dog timer 啟動之後，程式必須在 timer 發生 overflow 之前先執行 RF $10(若未先執行 RF $10 則可能會造成重新執行 SF $10 時 timer 歸零不良，但有提供 WDT "ALWAYS ENABLE" option 的 MCU 則不用)再重新執行 SF $10 指令，以便將 timer 歸零。

在 watch dog timer 啟動後，只要 MCU 進入 HALT 或是 STOP mode，watch dog timer 就會暫時停止計數。等到 MCU 產生 HALT release 或是 STOP release 之後 timer 才會繼續動作。但若在 PH10=1 時進入 HALT 或是 STOP mode 或是執行 RF $10 時間點則會有多計數一次的問題，故建議使用者在每次 MCU 進入 HALT 或是 STOP mode 之後加計一次 timer 值或是直接在每次進入 HALT 或是 STOP mode 前執行 RF $10(若有提供 WDT "ALWAYS ENABLE" option 的 MCU 則不用)並於離開後再執行 SF $10 重新啟動 watch dog timer 程序以免因 HALT 或是 STOP mode 一直累積而發生 overflow。

執行 RF $10 指令可將 watch dog timer 關閉並將 watch dog timer enable flag (WDF)清除為 0。但選擇 WDT option 為"ALWAYS ENABLE"時 RF $10 指令就會無效，只能一直以 SF $10h 指令避免 overflow。

Watch dog timer 可以設定三種發生 overflow 的時間長度，八個 PH10 的 clock cycle，64 個 PH10 的 clock cycle 以及 512 個 PH10 的 clock cycle，這些設定都可以由 code option 選擇。

2-2-1. LOW VOLTAGE RESET(LVR)

當 VBAT 電壓低於 LVR 電位會使 MCU 進入 RESET 狀態直到電壓高於 LVR release 電位才會離開 RESET Mode。

4-BIT 系列提供兩種 LVR 架構，可以自行選擇所需要的 low voltage reset 功能。說明如下：

1. LVR(1):
   a. 3V Power Mode Only(非精準型&無溫度補償)。
   b. STOP 模式 VBAT=3.0V 電流<0.5uA。

c. 適用於比較在乎耗電，用於搭配”POWER ON RESET：(1) USE”確保可正常使電壓上升有效產生 POWER ON RESET。

2. LVR2
   a. 精準度高&有溫度補償。
   b. 進入&離開 RESET 電位可分開選擇設定，除了可藉此避免 RESET 的不穩定，也可確保離開 RESET Mode 有足夠的起始電位。
   c. STOP 模式 VBAT=3.0V 電流<1uA。
   d. 由於受穩壓架構限制 LVR2 本身依應用需求再分成兩個 code option，說明如下：
      (1) LVR2(VBAT > 1.2V for "LVR2 RELEASE VOLTAGE" < 2.20V)
      RELEASE 電壓小於 2.20V option 因受工作電壓限制 VBAT 進入 reset 後須維持>1.2V 以避免可能產生誤動作風險，適用於非斷電&VBAT 上升時間緩慢超過"POWER ON RESET" 規格要求應用，但是 2.2V 以上 option 則沒有工作電壓下限限制。此外唯有此 Mask Option 可使 RESET Mode 耗電小於 1uA。
      (2) LVR2((Power-Up > "LVR2 RELEASE VOLTAGE" follow Spec. of "POWER ON RESET")
      適用於重新上電應用，全部電壓 option 皆可適用，須符合"POWER ON RESET" 規格要求。此 Mask Option 建議選擇”POWER ON RESET：(2) NO USE”取代針對在雙電源切換應用時避免因電壓差而誤產生 power on reset，而在真正重新上電時仍可產生同樣 Power on reset 作用以及確保升至夠高的電位才會離開 RESET Mode。

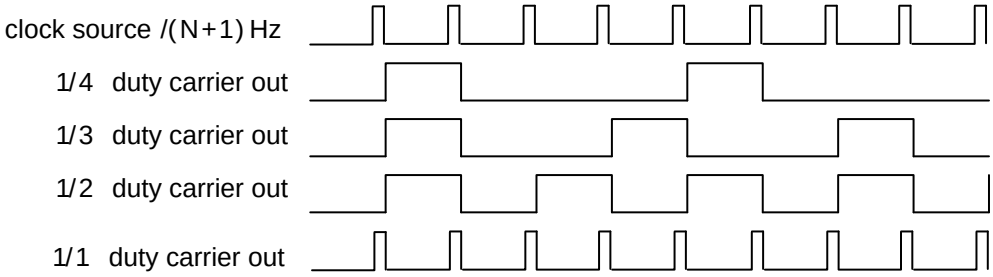
2-3. 頻率產生器(FREQUENCY GENERATOR)

頻率產生器會根據指令運算元的設定而產生不同頻率以及 duty cycle 的輸出信號，可以作為 buzzer output，EL plant driver，TMR1 ，TMR2 ，TMR3，SIO(TM87ML28)以及 RFC counter 等功能的 clock source 之用。

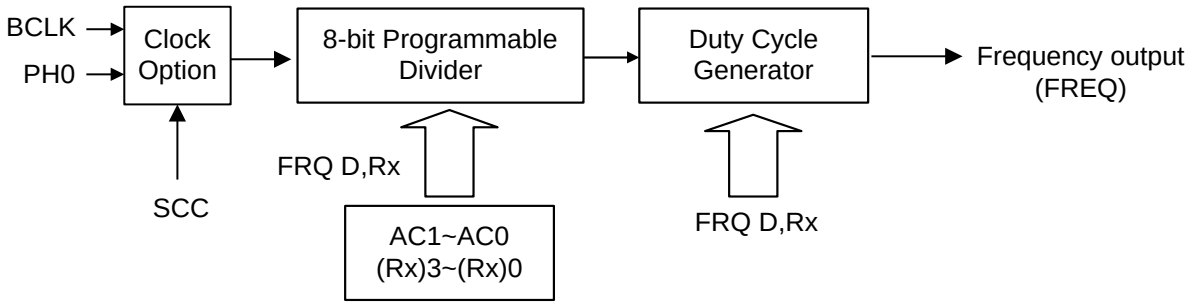
頻率產生器可以從 BCLK 或是 PH0 選擇信號作為 clock source，clock source 的信號先經過預除器將信號頻率除以 N+1 (N 是預除值)，所產生的波形會是一個 clock source 信號週期的高電位位準以及 N 個 clock source 信號週期的低電位位準。當預除值(N)設定為 0 的時候，預除器所產生的信號波形會與 clock source 的波形相同。

預除器所產生的信號再經過一個 duty cycle 產生器之後就可以產生所需的信號 (FREQ)。

下圖說明頻率產生器所產生的各種 duty cycle 的波形：



下圖是頻率產生器的功能方塊圖：



執行 SCC 指令可以選擇 BCLK 或是 PH0 做為頻率產生器的 clock source(在 TM87ML28 則以 XCLK 取代 BCLK，如此可藉由 SXCLK 指令設定 X0=1 在 FAST Mode 時切換 BCLK 變成 FTOSC 而不受 FAST 指令選擇 BCLK 頻率=FTOSC 除頻輸出切換影響)。

執行 FRQ 等相關指令可以設定頻率產生器的預除值(N)以及 duty cycle(D)，以便產生所期望的輸出信號，輸出信號的頻率可藉由下面的公式計算出來：

頻率產生器的輸出頻率 = (clock source 的頻率) / ((N+1) * X) Hz.

N : 是 FRQ 相關指令中所設定的預除值

X : 1~ 4，分別代表 1/1, 1/2, 1/3, 1/4 duty

在指令運算元中，預除值(N)可以是 immediate data, table ROM 的內容值或是 AC 以及 data memory 兩者的內容值。

下表說明運算元與預除值(N)之間的位元關係：

	The bit pattern of preset data N							
FRQ 的相關指令	bit7	Bit6	bit5	bit4	bit3	bit2	bit1	bit0
FRQ D,Rx	AC3	AC2	AC1	AC0	(Rx)3	(Rx)2	(Rx)1	(Rx)0
FRQ D,@HL	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0
FRQX D,X	X7	X6	X5	X4	X3	X2	X1	X0

Notes:

1. TD0 ~ TD7 represents the data of table ROM.
2. X0 ~ X7 運算元中的 immediate data.

下表說明運算元與 duty cycle 設定值(D)之間的位元關係：

Preset Letter D		Duty Cycle
D1	D0	
0	0	1/4 duty
0	1	1/3 duty
1	0	1/2 duty
1	1	1/1 duty

在完成頻率產生器的設定之後並不會馬上啟動，而且平時也是不動作的。只有在 Timer、RFC、EL panel driver、ALM、SIO(TM87ML28)等功能啟動後而且使用 FREQ 作為信號輸入時，或是執行 SBZ 指令(X1=1 或是 X0=1)後才會啟動頻率產生器的 Clock Source (Cfq)讓 FREQ 開始輸出信號。當這些功能關閉或是停止時，FREQ 也會停止輸出信號。

因此如果要將 FREQ 的輸出信號作為計時器的基準頻率時須特別留意，當 timer 將 FREQ 設定成 clock source，在 timer 發生 underflow 之後如果沒有其他功能持續啟動以便讓頻率產生器繼續工作，FREQ 將會停止輸出信號，必須等到下一次重新啟動 timer 時頻率產生器才會開始輸出信號。這樣每次 timer 發生 underflow 的時間周期就會受到影響而無法得到一個準確的計時週期。

如果真有此應用的必要時，則需藉由啟動其它相關功能使頻率產生器一直維持動作。

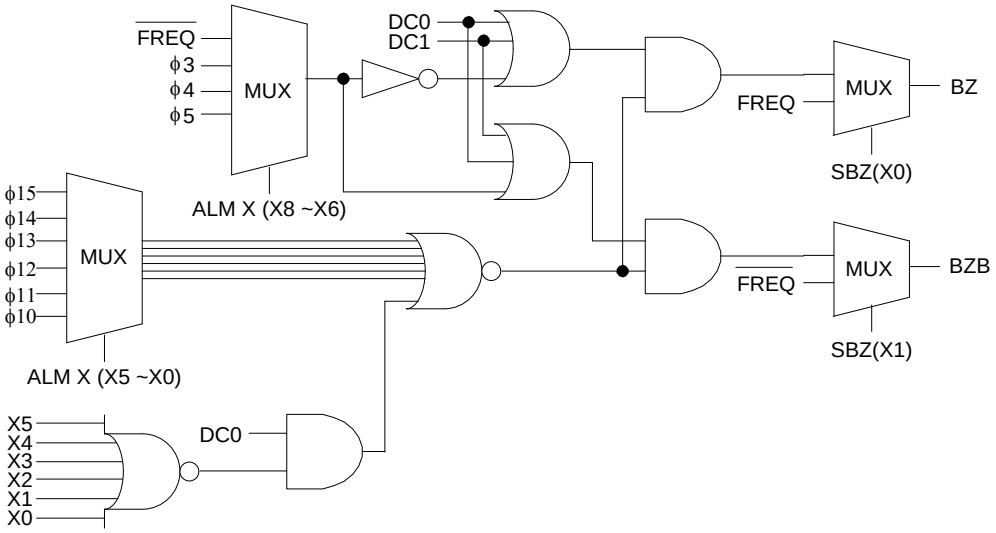
2-4. BUZZER輸出功能

Buzzer 輸出功能透過 BZB 以及 BZ 兩個輸出腳位推動 buzzer 等元件就可以達到發出聲音的功能，BZB 以及 BZ 的輸出信號是以反相的方式輸出。

執行 SBZ 指令可以將頻率產生器的輸出信號或是由 ALM 指令所設定的調變信號輸出至 BZ 以及 BZB 的腳位上。

如果 buzzer 輸出功能的信號來源是來自頻率產生器的話，就可以產生遙控器所使用的 carrier outputs 或是 single tone melodies 等不同的功能。如果 buzzer 功能的信號來源是 來自 ALM 指令所設定的調變信號的話，就可以產生 sound effect 的功能。

後面將會詳細說明這些功能的用法。



This figure shows the organization of the buzzer output.

2-4-1.SOUND EFFECT 功能

Sound effect 功能可以產生一個由 FREQ(頻率產生器的輸出信號), PH3 (4096 Hz), PH4 (2048 Hz), PH5 (1024 Hz)等高頻信號(carrier 信號)，以及 PH10 (32 Hz), PH11 (16 Hz), PH12 (8 Hz), PH13 (4 Hz), PH14 (2 Hz), PH15 (1 Hz)等低頻信號(envelope 信號)所合成的調變信號。

執行 ALM 指令可以設定這些信號的組合，設定合成的調變信號時請注意下列事項：

1. 高頻信號只能選擇 PH3, PH4, PH5 或是 FREQ 中的一個信號；低頻信號可以任選 PH10 ~ PH15 中的信號來組合(可以同時選擇這些信號)。

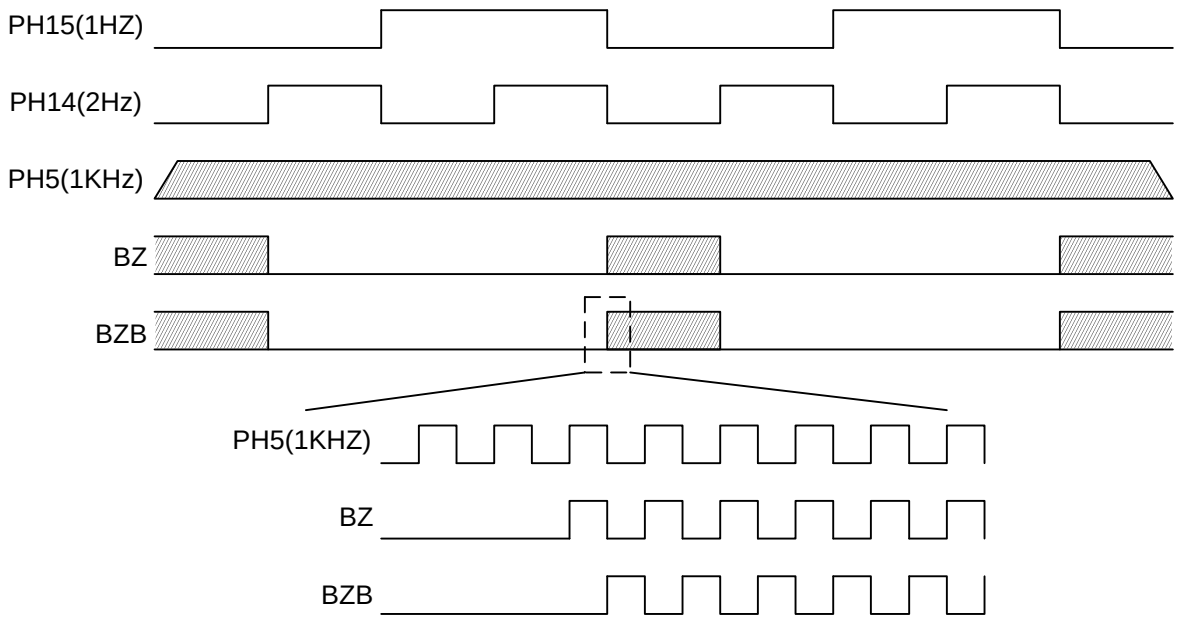
2. 上述中括弧內的頻率是以 PH0 = 32768 Hz 為基準所計算出來的頻率。

3. MCU 進入 RESET 狀態後，BZ 以及 BZB 會同時輸出低位準的輸出信號(DC0)。

下面範例說明 buzzer 功能輸出一個由 PH5 (carrier)以及 PH15, PH14 (envelope)所組成的調變信號：

```
.....
ALM    $070            ; Output the waveform.
.....
```

下圖說明這個範例所輸出的調變信號：



2-4-2. 遙控器所使用的 CARRIER OUTPUT

Buzzer 輸出功能搭配 timer 以及頻率產生器使用之後就可以產生遙控器所需的 carrier 信號。

在紅外線遙控器的應用中，頻率產生器的預除值必須大於 3，因為如果頻率產生器的輸出信號頻率太高將會使得 timer 在產生 underflow 之後無法及時將 BZ 或是 BZB 的輸出信號　改設定成 DC0，而使得 BZ 或是 BZB 多送出幾個額外的 carrier 信號。

此外，ALM 指令必須緊接著在 FRQ 等相關指令後執行，頻率產生器的輸出信號(FREQ)就可由 BZ 或是 BZB pin 輸出，作為紅外線遙控器的 carrier 信號之用。

範例：

```
SHE         1       ; 設定 TMR1 的 halt release enable flag.
TMSX        $3F     ; TMR1 的起始值為 3Fh，clock source 選擇 PH9，啟動 TMR1
SCC         $40     ; 選擇 BCLK 作為頻率產生器的 clock source
FRQX        2, 3    ; 頻率產生器的預除值為 3，duty cycle 是 1/2
ALM         $1C0    ; 將頻率產生器的輸出信號(FREQ)輸出到 BZ pin
HALT                ; 等待 TMR1 產生 HALT release request
……

ALM         0       ; BZ pin 輸出 DC0 信號
```

2-4-3. SINGLE TONE MELODY

如果頻率產生器設定特定的預除值(N)以及 1/2 duty cycle 之後，配合 buzzer 的輸出功能就可以產生特定音階的 single tone。

下表說明各個音階所需設定的預除值(N)：

Tone	N	FREQ	Ideal	%	Tone	N	FREQ	Ideal	%

C2	249	65.536	65.4064	0.19	C4	62	260.063	261.626	-0.6
#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	327.68	329.628	-0.59
F2	187	87.1489	87.3071	-0.18	F4	46	348.596	349.228	-0.18
#F2	176	92.565	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	109.96	110	-0.04	A4	36	442.811	440	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	0.2	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	587.33	-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	655.36	659.255	-0.59
F3	93	174.298	174.614	-0.18	F5	22	712.348	698.456	1.99
#F3	88	184.09	184.997	-0.49	#F5	21	744.727	739.989	0.64
G3	83	195.048	195.998	-0.48	G5	20	780.19	783.991	-0.48
#G3	78	207.392	207.652	-0.13	#G5	19	819.2	830.609	-1.37
A3	73	221.405	220	0.64	A5	18	862.316	880	-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

Notes:

1. Above variation does not include X'tal variation.

2. If PH0 = 65536 Hz, C3 - B5 may have more accurate frequency.

3. The clock source is PH0, i.e. 32,768 Hz.

4. The duty cycle is 1/2 Duty (D=2).

5. “FREQ” is the output frequency.

6. “ideal” is the ideal tone frequency.

7. “%” is the frequency deviation.

2-5. IO PORTS

4BIT 系列 MCU 提供五個 IO port : IOA, IOB, IOC, IOD 以及 IOE 有 IPA~E,OPA~E,SPA~E 等指令可控制 I/O port，另外由 MWM&MMW 指令以 Rm 定址方式更可擴充最多 IOF, IOG, IOH 以及 IOI 四個 I/O port，並由 data RAM 以 Ry 定址方式進行間接設定與資料存取的動作，每一個 IO port 都有四個 IO 腳位。

如果 IO port 與其他功能共用輸出腳位時，當 code option 將共用腳位設定給其他功能使用之後，該 IO port 腳位的相關功能將自動關閉。

2-5-1. IOA PORT

IOA port 的四個腳位都可以利用 SPA 指令各自設定成 input 或是 output mode，但是在 MCU 離開 RESET 狀態之後，所有 IOA port 的腳位都會自動設定成 input mode。

執行 OPA 指令會將 data memory 的內容值寫入 IOA port 的 output register，如果 IOA port 已經設定成 output mode，output register 的內容值就會輸出到 IOA 的腳位上。

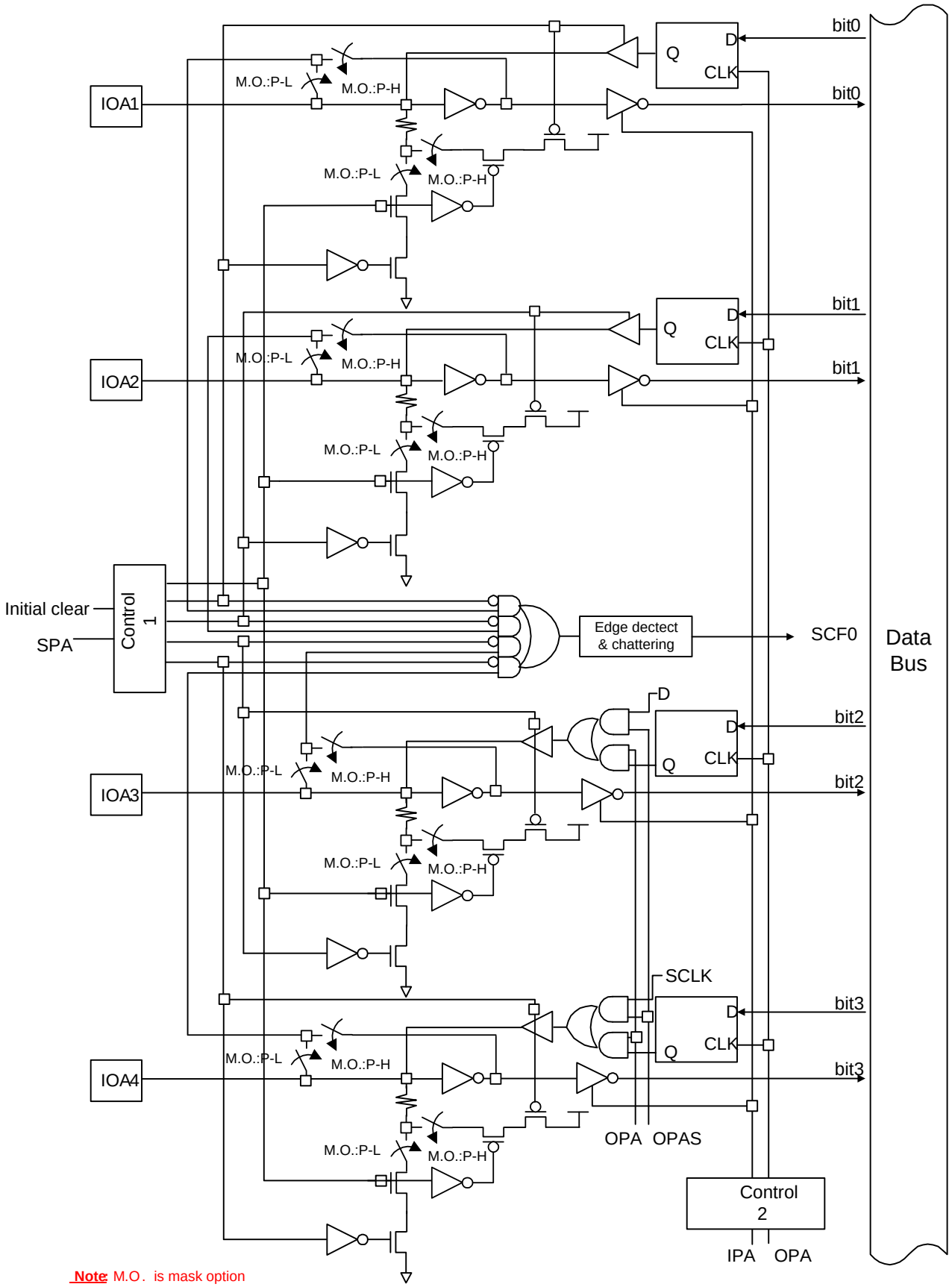
執行 IPA 指令可以將 IOA port 腳位上的信號儲存到 data memory。如果 IOA port 已經設定成 output mode，執行 IPA 指令就會將 output register 的內容值儲存到 data memory。

當程式要將 IOA port 切換成 output mode 之前必須先執行 OPA 指令將準備輸出的信號寫入 output register 中，這樣 IOA port 的腳位才不會在切換到 output mode 之後輸出一些不需要的信號。

IOA port 的各個腳位在 input mode 下都內建一個防止輸入信號空接的下拉電阻，這個下拉電阻可以利用 code option 來決定是否要使用。在 input mode 下，如果輸入腳位上的信號空接時將會造成大電流流過 input buffer，當 code option 選擇使用下拉電阻時起始狀態為啟動而且可藉由執行 SPA 指令統一設定四個腳位啟動或是關閉，但若該腳位設定為 output mode 則下拉電阻會自動關閉。

另外如 TM87ML28 其中一組 IOA code option 除了下拉電阻亦有提供上拉電阻。選擇上拉電阻時不會改變 input & output data 的相位，只有對 input data 送至 Halt Release 功能前會先進行反相。

下圖說明 IOA port 的結構：



Note: M.O. is mask option

2-5-1-1. PSEUDO SERIAL OUTPUT

執行 OPAS 指令可以將 IOA port 的輸出功能模擬成 serial output 的模式(pseudo serial output mode)，在執行這個指令之前必須先將 IOA port 設定成 output 的模式。

在使用 IOA port 的 pseudo serial output 功能時，各個輸出腳位的輸出信號如下所示：

1). IOA1 以及 IOA2 會輸出 data memory 的 bit0 以及 bit1 的資料。

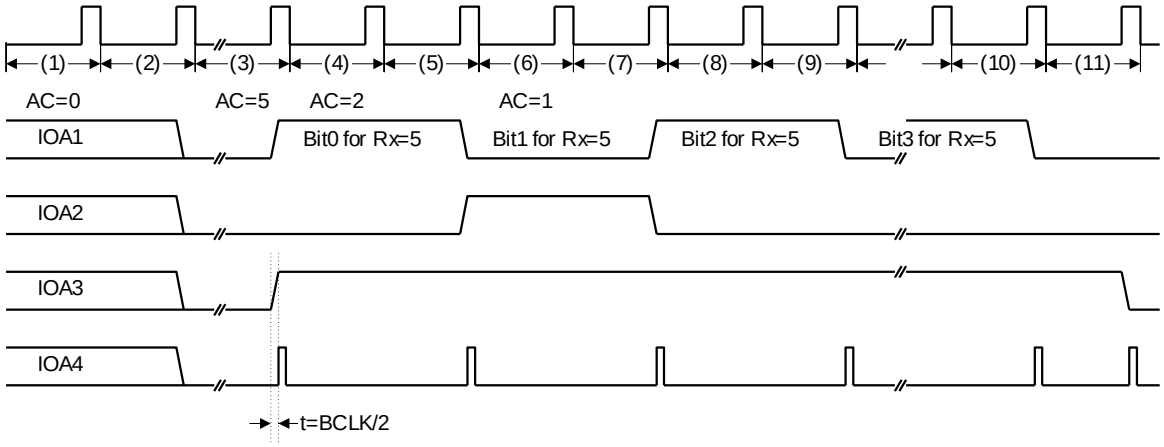
2). IOA3 會輸出 OPAS 指令運算元中 immediate data (D) 的資料。

3). IOA4 會在 OPAS 指令週期結束之前送出一個寬度是系統時鐘週期一半(BCLK/2)的脈衝信號。

下面是一個 OPAS 指令範例：

```
(1)     LDS   $0A, 0
(2)     OPA   $0A
        SPA   $0F
         :
         :
        LDS   1,5     ; output 0101b to IOA1, IOA2, IOA3 as the enabled signal of shift
                      ; operation
(3)     OPAS  1,1     ; Bit 0 output, shift operation is enabled
(4)     SR0   1       ; Shifts bit 1 to bit 0
(5)     OPAS  1,1     ; Bit 1 output
(6)     SR0   1       ; Shifts bit2 to bit 0
(7)     OPAS  1,1     ; Bit 2 output
(8)     SR0   1       ; Shifts bit 3 to bit 0
(9)     OPAS  1,1     ; Bit 3 output
         :
         :
(10)    OPAS  1,1     ; Last data
(11)    OPAS  1,0     ; Shift operation is disabled
```

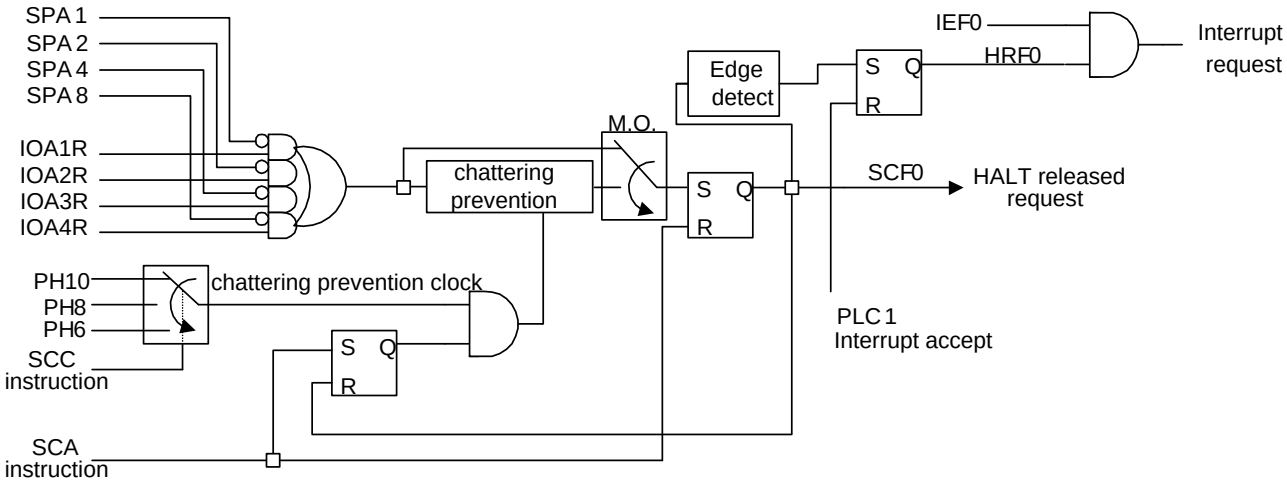
下圖說明上面範例中各個 IOA 輸出腳位上輸出信號的關係：



2-5-1-2. START CONDITION FLAG 0 (SCF0)的設定

IOA port 設定 start condition flag 0 (SCF0)的方式有兩種，一種是外部信號直接設定(或稱為高電位偵測)方式，另一種是外部信號經過雜訊消除線路(或稱為 chattering prevention)的方式。這兩種方式可以利用 code option 的方式來選擇。

下圖說明 IOA port 的相關控制線路以及 flag：



Note: The default prevention clock is PH10

依 code option 選擇 pull-low/high 電阻，IOA1~4R 與 IOA1~4 為同/反相關係。因為 IOA1~4R 在經過 OR 邏輯閘之後才會送到高電位偵測功能或是 chattering prevention 功能做判斷，因此必須要 IOA1~4R 其中三個同時為低電位，只有一個輸入腳位的信號改變的時候，這樣 OR 邏輯閘的輸出信號才會產生變化。

2-5-1-2-1. 外部信號直接設定方式

當 code option 選擇使用高電位偵測功能時，如果 SCA 指令已經將 switch enable flag 5 (SEF5) 設定為 1，IOA1~4R 只要有從低電位變成高電位的改變就能將 start condition flag 0 (SCF0)設定為 1。

一旦 SCF0 設定為 1 之後，MCU 就無法再次偵測 IOA port 上的信號變化，必須先將 IOA port 的所有輸入腳位的信號設定為 0 之後再重新執行 SCA 指令將 SCF0 清除為 0，以便下一次 IOA port 的信號改變可以重新設定 SCF0。

2-5-1-2-2. 外部信號經過雜訊消除線路的方式

Chattering prevention 線路有兩種特性，可以消除輸入信號的雜訊以及偵測輸入信號的改變。

1. 消除輸入信號的雜訊

如果輸入信號上的雜訊小於設定的時間長度， chattering prevention 線路會將這個雜訊消除。(但若雜訊週期剛好與 chattering prevention clock 週期同步則例外) 。

消除信號雜訊所使用的 chattering prevention clock 頻率共有三種，PH10 (32 ms), PH8 (8 ms)或是 PH6 (2 ms)，可以利用 SCC 指令來設定，而 PH10 是 MCU 的原始設定。

2. 偵測輸入信號的改變

Chattering prevention 線路在消除雜訊之後的輸入信號的上昇緣或是下降緣發生時都會送出一個信號將 start condition flag 0 (SCF0)設定為 1。

當 code option 選擇使用 chattering prevention 功能時，如果 SCA 指令已經將 switch enable flag 5 (SEF5)設定為 1，IOA port 的輸入信號無論是從低電位變成高電位的信號或是從高電位變成低電位，chattering prevention 功能都能夠判斷出信號的改變並將 SCF0 設定為 1。

一旦 SCF0 設定為 1 之後，MCU 就無法再次偵測 IOA port 上的信號變化，必須重新執行 SCA 指令將 SCF0 清除為 0，以便下一次 IOA port 的信號改變可以重新設定 SCF0。

2-5-1-2-3. 執行 SCA 指令的注意事項

當程式執行 SCA 指令之後 MCU 會將 SCF0, SCF1 以及 SCF3 同時清除為 0，因此在執行 SCA 指令之前必需先確認 SCF1 以及 SCF3 是否也已經被設定為 1，以免遺漏了 IOC 以及 IOD port 上的信號變化。

2-5-1-3. HALT RELEASE，STOP RELEASE 以及中斷服務

2-5-1-3-1. HALT RELEASE

Start condition flag 0 (SCF0)是 IOA port 的 halt release request signal，當 SCF0 設定為 1 之後就會將 halt release request flag 0 (HRF0)也設定為 1，MCU 就會產生 HALT release。

當 IOA port 選用不同的方式來設定 SCF0 時，MCU 產生 HALT release 的條件也有所不同。

1. 當 IOA port 是選用高電位偵測方式來設定 SCF0 時，在將 SEF5 設定為 1 之後，IOA1~4R 任何一個變成高電位時就可以讓 MCU 產生 HALT release。
   但是 MCU 必需在 IOA1~4R 都是低電位的情況下才能進入 HALT mode。
2. 當 IOA port 是選用 chattering prevention 方式來設定 SCF0 時，在 SEF5 設定為 1 之後， IOA port 輸入腳位上所有輸入信號經 OR 邏輯閘後有任何的變化就可以讓 MCU 產生 HALT release。
   在選用 chattering prevention 功能時，MCU 可以在 IOA port 的輸入信號是任何的情況下進入 HALT mode。

2-5-1-3-2. 中斷服務

Start condition flag 0 (SCF0)也是 IOA port 的 interrupt request signal，如果 IOA port 的 interrupt enable flag 0 (IEF0)已經設定為 1 時，當 SCF0 設定為 1 之後就會將 halt release request flag 0 (HRF0)也設定為 1，MCU 就會接受 IOA port 的中斷請求。

2-5-1-3-3. STOP RELEASE

當 IOA port 選用不同的方式來設定 start condition flag 0 時，MCU 產生 STOP release 的條件也有所不同。但是都只有在 IOA port 所有輸入信號都是低電位的情況下 MCU 才能進入 STOP mode。

1. 當 IOA port 是選用高電位偵測方式來設定 SCF0 時，在將 SEF5 設定為 1 之後，IOA port 輸入腳位上有任何輸入信號變成高電位時就可以讓 MCU 產生 STOP release。

2. 當 IOA port 是選用 chattering prevention 方式來設定 start condition flag 時，只有在 SEF5 以及 SRF6 同時設定為 1 之後， IOA port 輸入腳位上有任何輸入信號變成高電位時才可以讓 MCU 產生 STOP release。

在 MCU 產生 STOP release 之後會先進入 HALT mode，IOA port 輸入腳位上的高電位信號必須持續維持一段時間，直到 chattering prevention 功能確認這個高電位信號並將 start condition flag 0 設定為 1 之後才能回到低電位。

如果這個高電位信號在 SCF0 尚未設定之前就回到低電位，MCU 會立即返回 STOP mode。

2-5-2. IOB PORT

IOB port 的四個腳位都可以利用 SPB 指令各自設定成 input 或是 output mode，但是在 MCU 離開 RESET 狀態之後，所有 IOB port 的腳位都會自動設定成 input mode。

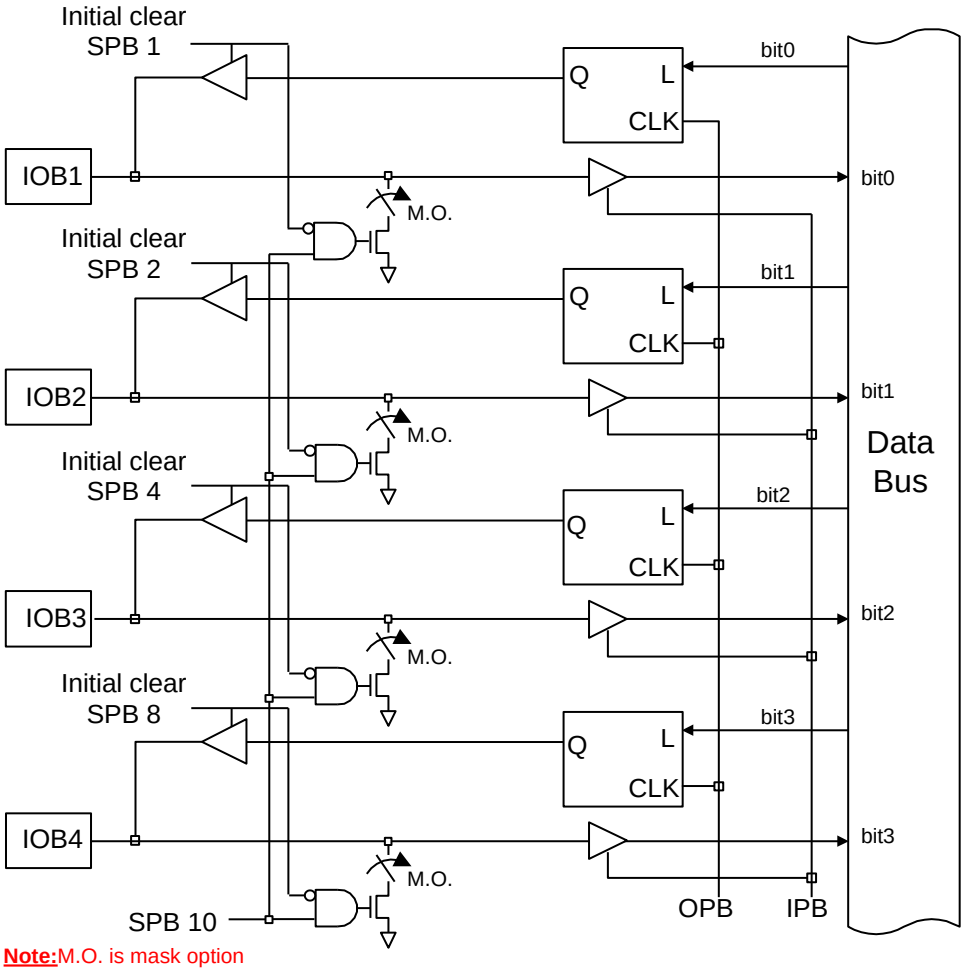
執行 OPB 指令會將 data memory 的內容值寫入 IOB port 的 output register，如果 IOB port 已經設定成 output mode，output register 的內容值就會輸出到 IOB 的腳位上。

執行 IPB 指令可以將 IOB port 腳位上的信號儲存到 data memory。如果 IOB port 已經設定成 output mode，執行 IPB 指令就會將 output register 的內容值儲存到 data memory。

當程式要將 IOB port 切換成 output mode 之前必須先執行 OPB 指令將準備輸出的信號寫入 output register 中，這樣 IOB port 的腳位才不會在切換到 output mode 之後輸出一些不需要的信號。

IOB port 的各個腳位在 input mode 下都內建一個防止輸入信號空接的下拉電阻，這個下拉電阻可以利用 code option 來決定是否要使用。在 input mode 下，如果輸入腳位上的信號空接時將會造成大電流流過 input buffer，下拉電阻可藉由執行 SPB 指令統一設定四個腳位啟動或是關閉，但若該腳位設定為 output mode 則下拉電阻會自動關閉。

下圖說明 IOB port 的結構：



Note:M.O. is mask option

2-5-3. IOC PORT

IOC port 的四個腳位都可以利用 SPC 指令各自設定成 input 或是 output mode，但是在 MCU 離開 RESET 狀態之後，所有 IOC port 的腳位都會自動設定成 input mode。

執行 OPC 指令會將 data memory 的內容值寫入 IOC port 的 output register，如果 IOC port 已經設定成 output mode，output register 的內容值就會輸出到 IOC 的腳位上。

執行 IPC 指令可以將 IOC port 腳位上的信號儲存到 data memory。如果 IOC port 已經設定成 output mode，執行 IPC 指令就會將 output register 的內容值儲存到 data memory。

當程式要將 IOC port 切換成 output mode 之前必須先執行 OPC 指令將準備輸出的信號寫入 output register 中，這樣 IOC port 的腳位才不會在切換到 output mode 之後輸出一些不需要的信號。

IOC port 的各個腳位在 input mode 下都內建一個防止輸入信號空接的下拉電阻以及 low-level hold 元件，這兩個元件可以利用 code option 來決定是否要使用。下拉電阻可以 單獨選擇是否使用，但是 low-level hold 元件必須與下拉電阻同時使用，不可單獨使用。 在 input mode 下，如果輸腳位上的信號空接時將會造成大電流流過 input buffer，下拉電阻可藉由執行 SPC 指令統一設定四個腳位啟動或是關閉，但若該腳位設定為 output mode 則下拉電阻會自動關閉。

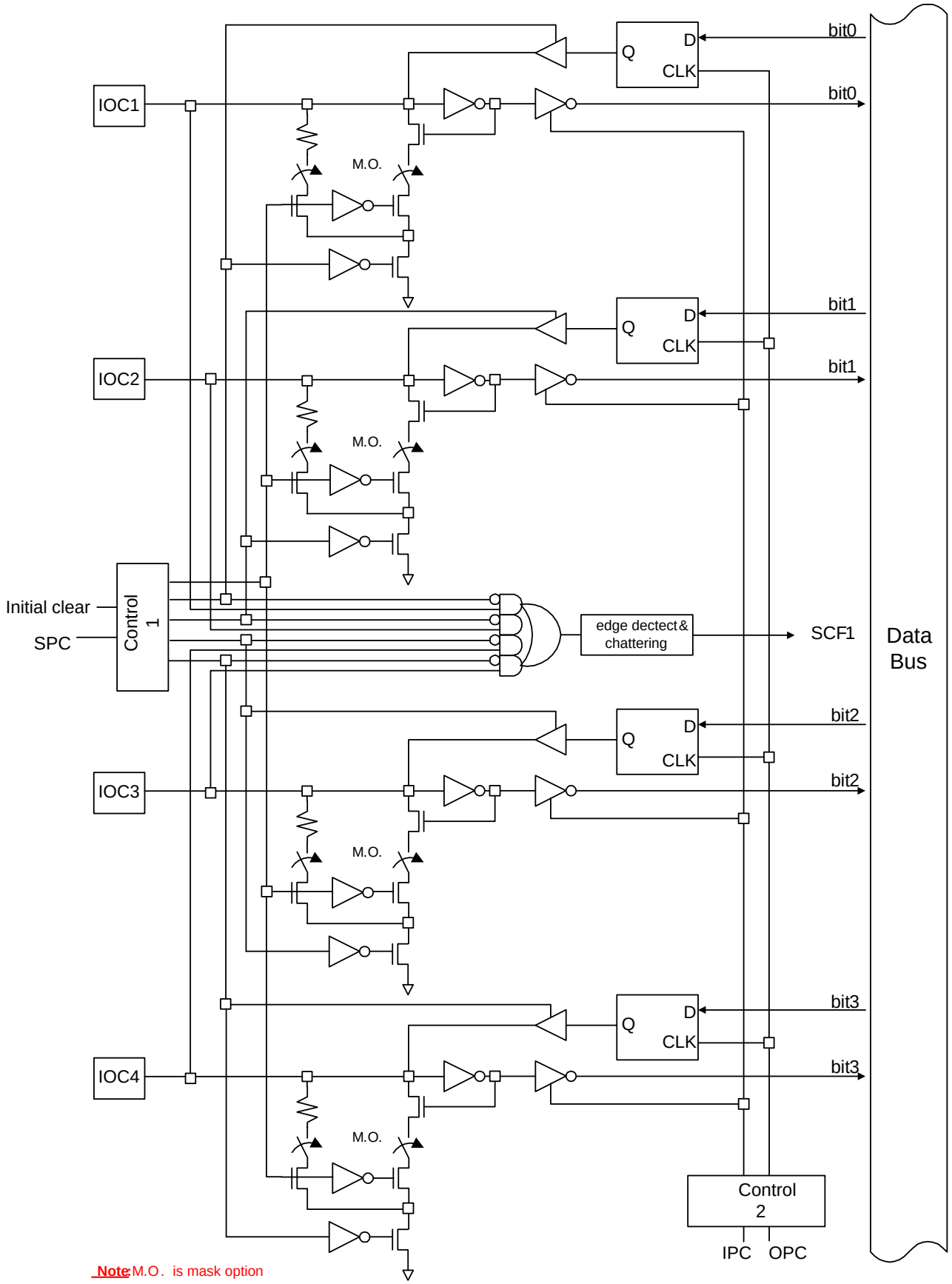
當 code option 同時選擇使用下拉電阻以及 low-level hold 元件時，執行 SPC 10h 指令可以開啟下拉電阻並關閉 low-level hold 元件，執行 SPC 0h 指令可以關閉下拉電阻並開啟 low-level hold 元件。

Low-level hold 的功能主要是在應用電路上利用 bonding option 來做為程式應用的選擇，其目的一方面可使 PCB 上只需預留 VBAT 電位的 bonding option 接點而省去 GND 的 bonding option 接點，另一方面藉此功能仍可達到 bonding option 不會耗電的目的。

Bonding option 的使用方法如下：

1. MCU 離開 reset 狀態之後先開啟 bonding option pad 的下拉電阻，若 pad 是空接時，則藉由下拉電阻將 pad 拉至 GND 電位，若 pad 已經藉由 bonding option 接到 VBAT 電位，則 pad 仍可維持在 VBAT 電位，但是此時會有電流流經下拉電阻。
2. 接著執行 SPC 0h 指令(X4=0)將下拉電阻關閉並將 Low-level hold 功能啟動，如果 pad 空接時則 Low-level hold 功能會將 pad 維持在 GND 電位，若 pad 已經藉由 bonding option 接到 VBAT 電位時則因下拉電阻關閉而不再耗電。

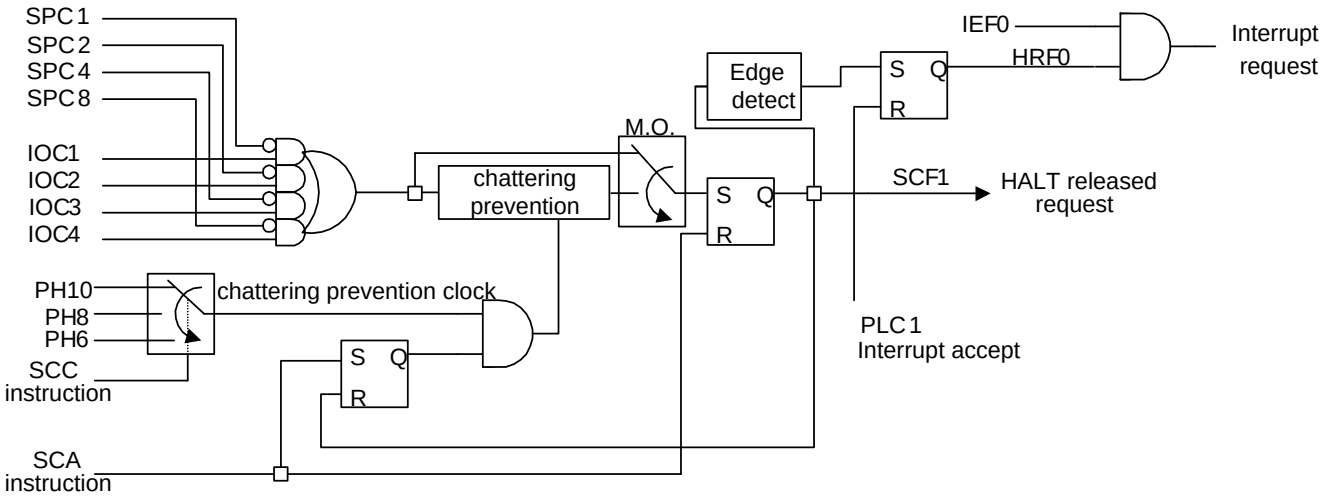
下圖說明 IOC port 的結構：



2-5-3-1. START CONDITION FLAG 1 (SCF1)的設定

IOC port 設定 start condition flag 1 (SCF1)的方式有兩種，一種是外部信號直接設定(或稱為高電位偵測)方式，另一種是外部信號經過雜訊消除線路(或稱為 chattering prevention)的方式。這兩種方式可以利用 code option 的方式來選擇。

下圖說明 IOC port 的相關控制線路以及 flag：



Note: The default prevention clock is PH10

因為 IOC port 的四個輸入信號在經過 OR 邏輯閘之後才會送到高電位偵測功能或是 chattering prevention 功能做判斷，因此必須要三個 IOC port 的輸入信號同時為低電位，只有一個輸入腳位的信號改變的時候，這樣 OR 邏輯閘的輸出信號才會產生變化。

2-5-3-1-1. 外部信號直接設定方式

當 code option 選擇使用高電位偵測功能時，如果 SCA 指令已經將 switch enable flag 4 (SEF4) 設定為 1，IOC port 的輸入信號只要有從低電位變成高電位的改變就能將 start condition flag 1 (SCF1) 設定為 1。

一旦 SCF1 設定為 1 之後，MCU 就無法再次偵測 IOC port 上的信號變化，必須先將 IOC port 的所有輸入腳位的信號設定為 0 之後再重新執行 SCA 指令將 SCF1 清除為 0，以便下一次 IOC port 的信號改變可以重新設定 SCF1。

2-5-3-1-2. 外部信號經過雜訊消除線路的方式

Chattering prevention 線路有兩種特性，可以消除輸入信號的雜訊以及偵測輸入信號的改變。

1. 消除輸入信號的雜訊

如果輸入信號上的雜訊小於設定的時間長度，chattering prevention 線路會將這個雜訊消除。(但若雜訊週期剛好與 chattering prevention clock 週期同步則例外)。

消除信號雜訊所使用的 chattering prevention clock 頻率共有三種，PH10 (32 ms), PH8 (8 ms)或是 PH6 (2ms)，可以利用 SCC 指令來設定，而 PH10 是 MCU 的原始設定。

2. 偵測輸入信號的改變

Chattering prevention 線路在消除雜訊之後的輸入信號的上昇緣或是下降緣發生時將 start condition flag 1 (SCF1)設定為 1。

當 code option 選擇使用 chattering prevention 功能時，如果 SCA 指令已經將 switch enable flag 4 (SEF4) 設定為 1，IOC port 的輸入信號無論是從低電位變成高電位的信號或是從高電位變成低電位，chattering prevention 功能都能夠判斷出信號的改變並將 SCF1 設定為 1。

一旦 SCF1 設定為 1 之後，MCU 就無法再次偵測 IOC port 上的信號變化，必須重新執行 SCA 指令將 SCF1 清除為 0，以便下一次 IOC port 的信號改變可以重新設定 SCF1。

2-5-3-1-3. 執行 SCA 指令的注意事項

當程式執行 SCA 指令之後 MCU 會將 SCF0, SCF1 以及 SCF3 同時清除為 0，因此在執行 SCA 指令之前必需先確認 SCF0 以及 SCF3 是否也已經被設定為 1，以免遺漏了 IOA 以及 IOD port 上的信號變化。

2-5-3-2. HALT RELEASE，STOP RELEASE 以及中斷服務

2-5-3-2-1. HALT release

Start condition flag 1 (SCF1)是 IOC port 的 halt release request signal，當 SCF1 設定為 1 之後就會將 halt release request flag 0 (HRF0)也設定為 1，MCU 就會產生 HALT release。

當 IOC port 選用不同的方式來設定 start condition flag 1 時，MCU 產生 HALT release 的條件也有所不同。

1. 當 IOC port 是選用高電位偵測方式來設定 start condition flag 1 時，在將 SEF4 設定為 1 之後，IOC port 輸入腳位上有任何輸入信號變成高電位時就可以讓 MCU 產生 HALT release。
   但是 MCU 必需在 IOC port 所有輸入信號都是低電位的情況下才能進入 HALT mode。
2. 當 IOC port 是選用 chattering prevention 方式來設定 start condition flag 1 時，在 SEF4 設定為 1 之後， IOC port 輸入腳位上所有輸入信號經 OR 邏輯閘輸出的信號有變化時就可以讓 MCU 產生 HALT release。
   在選用 chattering prevention 功能時，MCU 可以在 IOC port 的輸入信號是任何的情況下進入 HALT mode。

2-5-3-2-2. 中斷服務

Start condition flag 1 (SCF1)也是 IOC port 的 interrupt request signal，如果 IOC port 的 interrupt enable flag 0 (IEF0)已經設定為 1 時，當 SCF1 設定為 1 之後就會將 halt release request flag 0 (HRF0)也設定為 1，MCU 就會接受 IOC port 的中斷請求。

2-5-3-2-3. STOP release

當 IOC port 選用不同的方式來設定 start condition flag 1 時，MCU 產生 STOP release 的設定方式也有所不同。但是都只有在 IOC port 所有輸入信號都是低電位的情況下 MCU 才能進入 STOP mode。

1. 當 IOC port 是選用高電位偵測方式來設定 start condition flag 1 時，在 SEF4 設定為 1 之後，如果 IOC port 輸入腳位上有任何輸入信號變成高電位時就可以讓 MCU 產生 STOP release。
2. 當 IOC port 是選用 chattering prevention 方式來設定 start condition flag 1 時，只有在 SEF4 以及 SRF4 同時設定為 1 的時候， IOC port 輸入腳位上有任何輸入信號變成高電位時才可以讓 MCU 產生 STOP release。

   若是選用 chattering prevention 方式，在 MCU 產生 STOP release 之後會先進入 HALT mode，而且 IOC port 輸入腳位上的高電位信號必須持續維持一段時間，直到 chattering prevention 功能確認這個高電位信號並將 start condition flag 1 設定為 1 之後才能回到低電位。

   如果這個高電位信號在 start condition flag 1 尚未設定之前就回到低電位，MCU 會立即返回 STOP mode。

2-5-4. IOD PORT

IOD port 的四個腳位都可以利用 SPD 指令各自設定成 input 或是 output mode，但是在 MCU 離開 RESET 狀態之後，所有 IOD port 的腳位都會自動設定成 input mode。

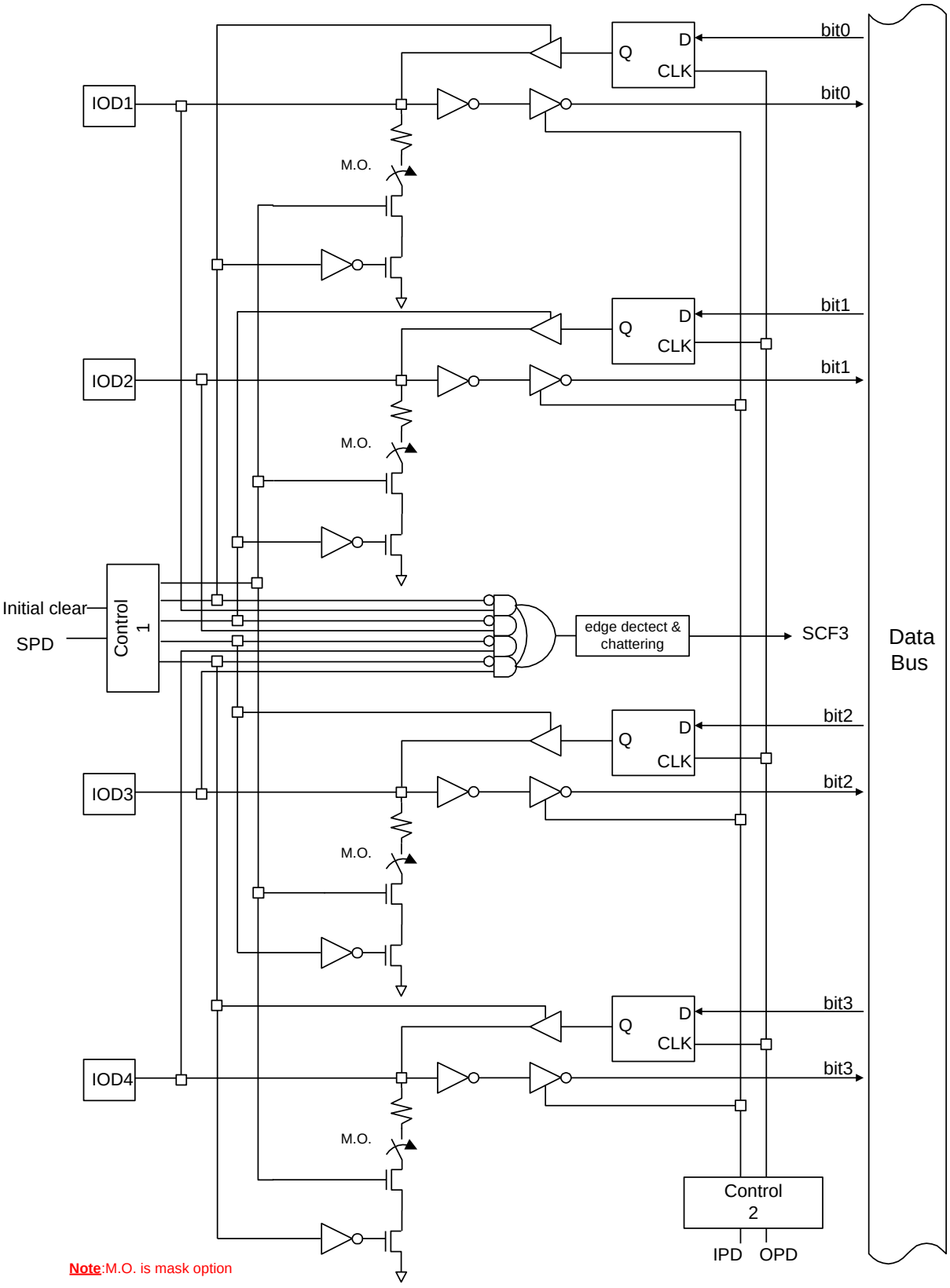
執行 OPD 指令會將 data memory 的內容值寫入 IOD port 的 output register，如果 IOD port 已經設定成 output mode，output register 的內容值就會輸出到 IOD 的腳位上。

執行 IPD 指令可以將 IOD port 腳位上的信號儲存到 data memory。如果 IOD port 已經設定成 output mode，執行 IPD 指令就會將 output register 的內容值儲存到 data memory。

當程式要將 IOD port 切換成 output mode 之前必須先執行 OPD 指令將準備輸出的信號寫入 output register 中，這樣 IOD port 的腳位才不會在切換到 output mode 之後輸出一些不需要的信號。

IOD port 的各個腳位在 input mode 下都內建一個防止輸入信號空接的下拉電阻，這個下拉電阻可以利用 code option 來決定是否要使用。在 input mode 下，如果輸腳位上的信號空接時將會造成大電流流過 input buffer，下拉電阻可藉由執行 SPD 指令統一設定四個腳位啟動或是關閉，但若該腳位設定為 output mode 則下拉電阻會自動關閉。

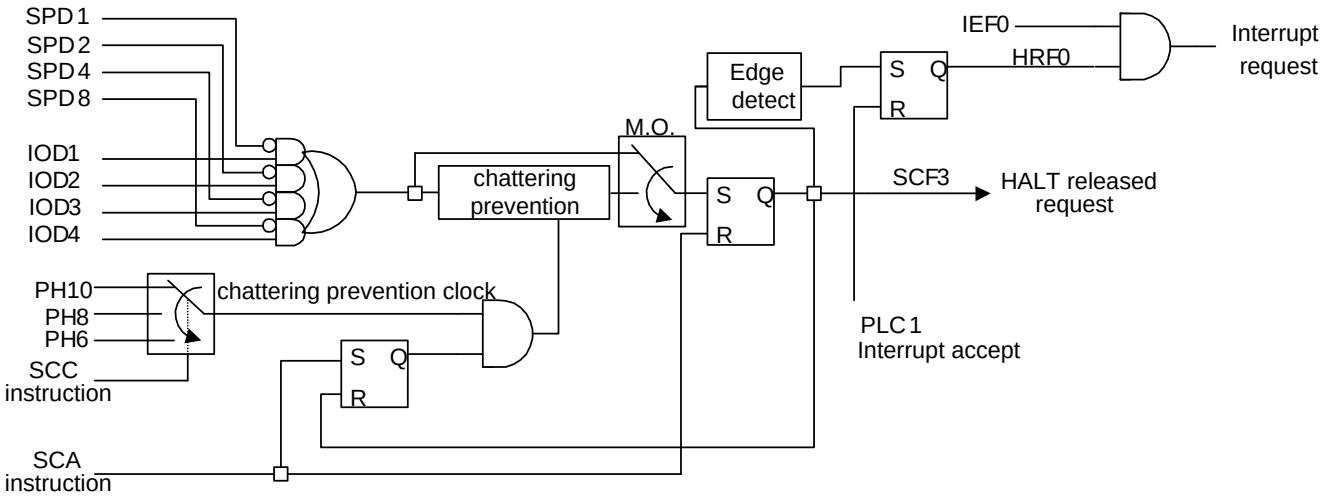
下圖說明 IOD port 的結構：



2-5-4-1. START CONDITION FLAG 3 (SCF3)的設定

IOD port 設定 start condition flag 3 (SCF3)的方式有兩種，一種是外部信號直接設定(或稱為高電位偵測)方式，另一種是外部信號經過雜訊消除線路(或稱為 chattering prevention)的方式。這兩種方式可以利用 code option 的方式來選擇。

下圖說明 IOD port 的相關控制線路以及 flag：



Note: The default prevention clock is PH10

因為 IOD port 的四個輸入信號在經過 OR 邏輯閘之後才會送到高電位偵測功能或是 chattering prevention 功能做判斷，因此必須要三個 IOD port 的輸入信號同時為低電位，只有一個輸入腳位的信號改變的時候，這樣 OR 邏輯閘的輸出信號才會產生變化。

2-5-4-1-1. 外部信號直接設定方式

當 code option 選擇使用高電位偵測功能時，如果 SCA 指令已經將 switch enable flag 3 (SEF3) 設定為 1，IOD port 的輸入信號只要有從低電位變成高電位的改變就能將 start condition flag 3 (SCF3) 設定為 1。

一旦 SCF3 設定為 1 之後，MCU 就無法再次偵測 IOD port 上的信號變化，必須先將 IOD port 的所有輸入腳位的信號設定為 0 之後再重新執行 SCA 指令將 SCF3 清除為 0，以便下一次 IOD port 的信號改變時可以重新 設定 SCF3。

2-5-4-1-2. 外部信號經過雜訊消除線路的方式

Chattering prevention 線路有兩種特性，可以消除輸入信號的雜訊以及偵測輸入信號的改變。

1. 消除輸入信號的雜訊

如果輸入信號上的雜訊小於設定的時間長度， chattering prevention 線路會將這個雜訊消除。(但若雜訊週期剛好與 chattering prevention clock 週期同步則例外) 。

消除信號雜訊所使用的 chattering prevention clock 頻率共有三種，PH10 (32 ms), PH8 (8 ms)或是 PH6 (2 ms)，可以利用 SCC 指令來設定，而 PH10 是 MCU 的原始設定。

2. 偵測輸入信號的改變

Chattering prevention 線路在消除雜訊之後的輸入信號的上昇緣或是下降緣發生時將 start condition flag 3 (SCF3)設定為 1。

當 code option 選擇使用 chattering prevention 功能時，如果 SCA 指令已經將 switch enable flag 3 (SEF3) 設定為 1，IOD port 的輸入信號無論是從低電位變成高電位的信號或是從高電位變成低電位，chattering prevention 功能都能夠判斷出信號的改變並將 SCF3 設定為 1。

一旦 SCF3 設定為 1 之後，MCU 就無法再次偵測 IOD port 上的信號變化，必須重新執行 SCA 指令將 SCF3 清除為 0，以便下一次 IOD port 的信號改變可以重新設定 SCF3。

2-5-3-1-3. 執行 SCA 指令的注意事項

當程式執行 SCA 指令之後 MCU 會將 SCF0, SCF1 以及 SCF3 同時清除為 0，因此在執行 SCA 指令之前必需先確認 SCF0 以及 SCF1 是否也已經被設定為 1，以免遺漏了 IOA 以及 IOC port 上的信號變化。

2-5-4-2. HALT RELEASE，STOP RELEASE 以及中斷服務

2-5-4-2-1. HALT release

Start condition flag 3 (SCF3)是 IOD port 的 halt release request signal，當 SCF3 設定為 1 之後就會將 halt release request flag 0 (HRF0)也設定為 1，MCU 就會產生 HALT release。

當 IOD port 選用不同的方式來設定 start condition flag 3 時，MCU 產生 HALT release 的條件也有所不同。

1. 當 IOD port 是選用高電位偵測方式來設定 start condition flag 3 時，在 SEF3 設定為 1 之後，如果 IOD port 輸入腳位上有任何輸入信號變成高電位時就可以讓 MCU 產生 HALT release。

   但是 MCU 必需在 IOD port 所有輸入信號都是低電位的情況下才能進入 HALT mode。

2. 當 IOD port 是選用 chattering prevention 方式來設定 start condition flag 3 時，在 SEF3 設定為 1 之後，IOD port 輸入腳位上所有輸入信號經 OR 邏輯閘輸出的信號有變化時就可以讓 MCU 產生 HALT release。

   在選用 chattering prevention 功能時，MCU 可以在 IOD port 的輸入信號是任何的情況下進入 HALT mode。

2-5-4-2-2. 中斷服務

Start condition flag 3 (SCF3)也是 IOD port 的 interrupt request signal，如果 IOD port 的 interrupt enable flag 0 (IEF0)已經設定為 1 時，當 SCF3 設定為 1 之後就會將 halt release request flag 0 (HRF0)也設定為 1，MCU 就會接受 IOD port 的中斷請求。

2-5-4-2-3. STOP release

當 IOD port 選用不同的方式來設定 start condition flag 3 時，MCU 產生 STOP release 的設定方式也有所不同。但是都只有在 IOD port 所有輸入信號都是低電位的情況下 MCU 才能進入 STOP mode。

1. 當 IOD port 是選用高電位偵測方式來設定 start condition flag 3 時，在 SEF3 設定為 1 之後，如果 IOD port 輸入腳位上有任何輸入信號變成高電位時就可以讓 MCU 產生 STOP release。

2. 當 IOD port 是選用 chattering prevention 方式來設定 start condition flag 3 時，只有在 SEF3 以及 SRF3 同時設定為 1 的時候， IOD port 輸入腳位上有任何輸入信號變成高電位時才可以讓 MCU 產生 STOP release。

在 MCU 產生 STOP release 之後會先進入 HALT mode，IOD port 輸入腳位上的高電位信號必須持續維持一段時間，直到 chattering prevention 功能確認這個高電位信號並將 start condition flag 3 設定為 1 之後才能回到低電位。

如果這個高電位信號在 start condition flag 3 尚未設定之前就回到低電位，MCU 會立即返回 STOP mode。

2-5-5. IOE PORT

IOE port 的四個腳位都可以利用 SPE 指令各自設定成 input 或是 output mode，但是在 MCU 離開 RESET 狀態之後，所有 IOE port 的腳位都會自動設定成 input mode。

執行 OPE 指令會將 data memory 的內容值寫入 IOE port 的 output register，如果 IOE port 已經設定成 output mode，output register 的內容值就會輸出到 IOE 的腳位上。

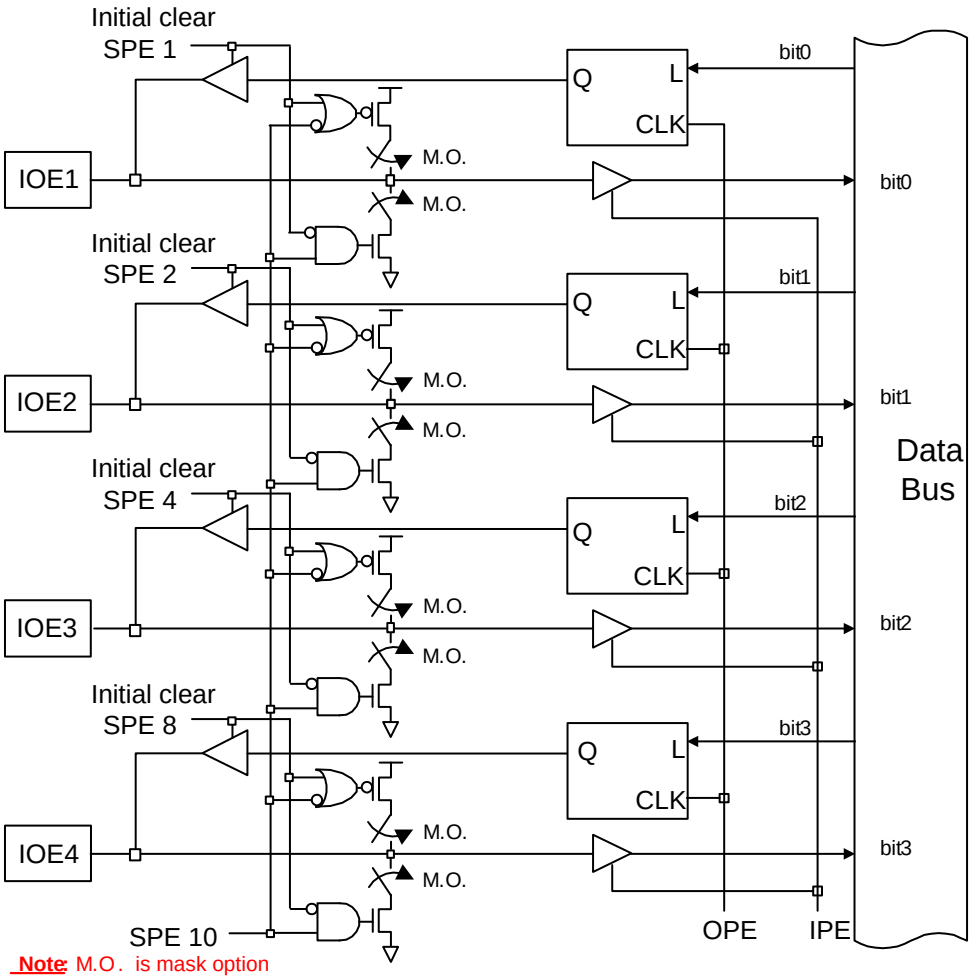
執行 IPE 指令可以將 IOE port 腳位上的信號儲存到 data memory。如果 IOE port 已經設定成 output mode，執行 IPE 指令就會將 output register 的內容值儲存到 data memory。

當程式要將 IOE port 切換成 output mode 之前必須先執行 OPE 指令將準備輸出的信號寫入 output register 中，這樣 IOE port 的腳位才不會在切換到 output mode 之後輸出一些不需要的信號。

IOE port 的各個腳位在 input mode 下都內建一個防止輸入信號空接的下拉電阻，這個下拉電阻可以利用 code option 來決定是否要使用。在 input mode 下，如果輸入腳位上的信號空接時將會造成大電流流過 input buffer，下拉電阻可藉由執行 SPE 指令統一設定四個腳位啟動或是關閉，但若該腳位設定為 output mode 則下拉電阻會自動關閉。

如 TM87ML28 其中一組 IOE code option 除了下拉電阻亦有提供上拉電阻。選擇上拉電阻時不會改變 input & output data 的相位。

下圖說明 IOE port 的結構：



Note: M.O. is mask option

2-5-6. IOF/G/H/I PORT

4BIT 系列可藉由 Rm 方式提供擴充最多 4 組 I/O，但與一般 I/O 指令不同在於對 data RAM 存取是以 working register(Ry)的定址方式，SPF/G/H/I(MWM 1/3/5/7,Ry)指令只單純設定 input 或是 output mode，下拉電阻的開啟或是關閉則皆由 SPLX(MWM 8,Ry)指令執行。

IOF/G/H/I port 的四個腳位都可以利用 SPF/G/H/I(MWM 1/3/5/7,Ry)指令各自設定成 input 或是 output mode，但是在 MCU 離開 RESET 狀態之後，所有 IOF/G/H/I port 的腳位都會自動設定成 input mode。

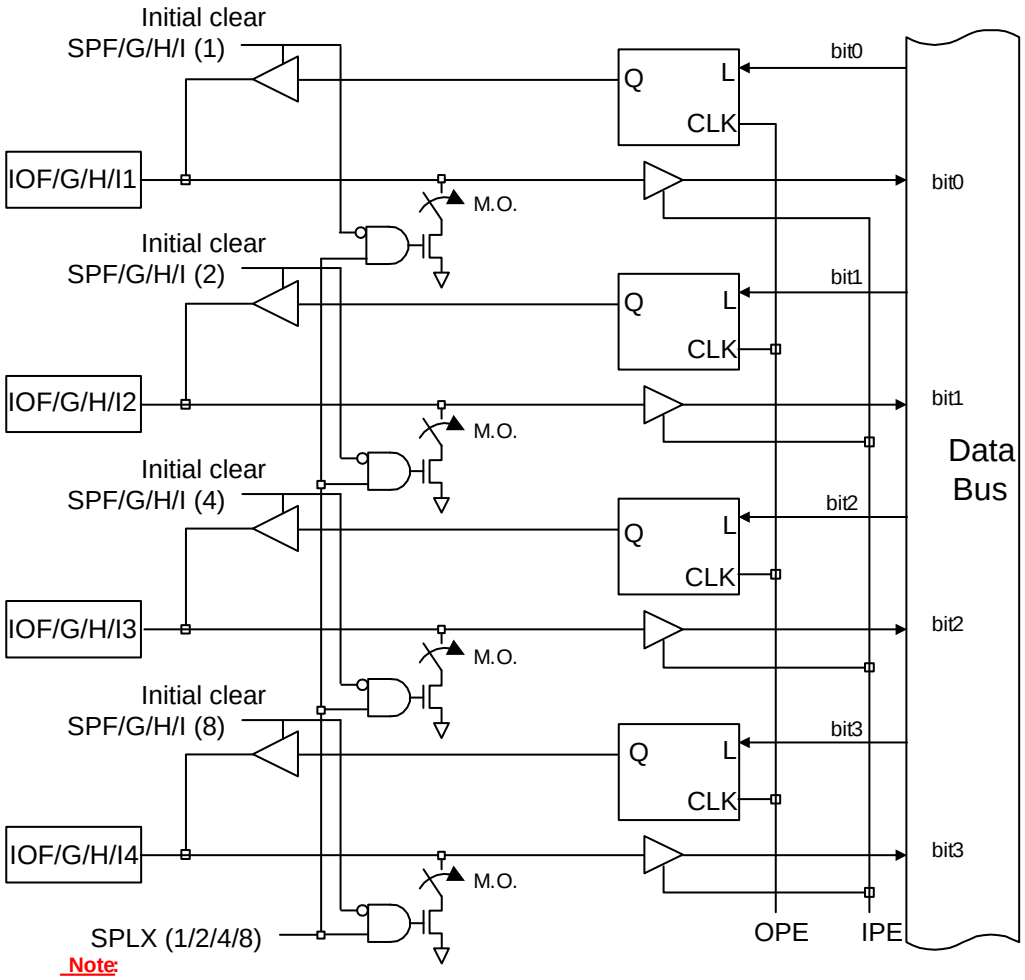
執行 OPF/G/H/I(MWM 0/2/4/6,Ry)指令會將 data memory 的內容值寫入 IOF/G/H/I port 的 output register，如果 IOF/G/H/I port 已經設定成 output mode，output register 的內容值就會輸出到 IOF/G/H/I 的腳位上。

執行 IPF/G/H/I(MMW Ry,0/2/4/6)指令可以將 IOF/G/H/I port 腳位上的信號儲存到 data memory。如果 IOF/G/H/I port 已經設定成 output mode，執行 IPF/G/H/I 指令就會將 output register 的內容值儲存到 data memory。

當程式要將 IOF/G/H/I port 切換成 output mode 之前必須先執行 OPF/G/H/I 指令將準備輸出的信號寫入 output register 中，這樣 IOF/G/H/I port 的腳位才不會在切換到 output mode 之後輸出一些不需要的信號。

IOF/G/H/I port 的各個腳位在 input mode 下都內建一個防止輸入信號空接的下拉電阻，這個下拉電阻可以利用 code option 來決定是否要使用。在 input mode 下，如果輸入腳位上的信號 空接時將會造成大電流流過 input buffer，下拉電阻可藉由執行 SPLX(MWM 8,Ry)指令 bit3~0 分別對 IOI~F 統一設定四個腳位啟動或是關閉，但若該腳位設定為 output mode 則下拉電阻會自動關閉。

下圖說明 IOF/G/H/I port 的結構：



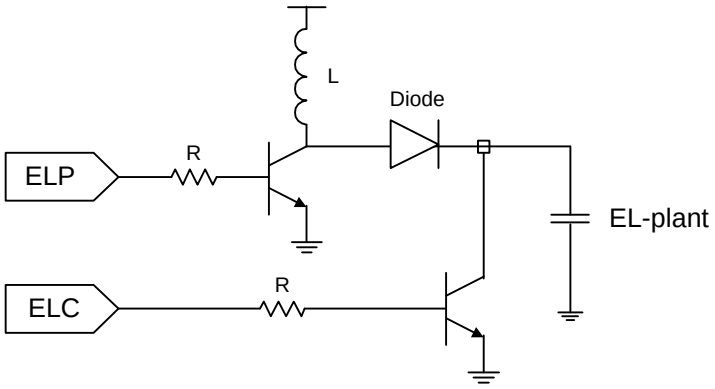
Note:

1. M.O. is mask option
2. () : data in address of Ry

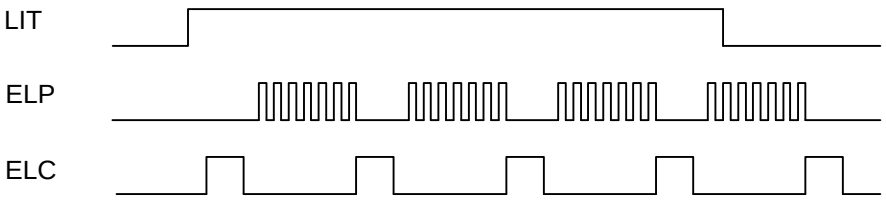
2-6. EL-PLANT DRIVER

4BIT 系列 MCU 提供一個 EL-plant driver (ELP 以及 ELC 輸出腳位)，只需加上少數的外部零件就可以直接驅動 EL plant 以產生系統所需的光源。ELP 腳位可輸出倍壓電路所需的控制信號，ELC 腳位則可輸出放電線路所需的控制信號。

下圖是 EL-plant driver 的應用線路：



下圖是 EL-plant driver 的控制信號：(LIT: 內部控制信號)



執行 SF 指令可以啟動 EL-plant driver 的功能(LIT 信號為高電位)，執行 RF 指令可以關閉 EL-plant driver 的功能(LIT 信號為低電位)。但是在開啟 EL-plant driver 的功能之前必須先執行 ELC 指令設定 ELP 以及 ELC 腳位的輸出信號頻率以及 duty cycle。

當 EL-plant driver 的功能啟動之後，ELC 腳位會在 ELP 腳位開始輸出倍壓信號之前送出一個 pulse 信號將 EL-plant 元件上的殘餘電荷清除，以免在後續倍壓程序中損壞 EL-plant 元件。

同樣在關閉 EL-plant driver 的功能之後，ELC 腳位會在 ELP 腳位輸出最後一組倍壓信號之後接著送出一個 pulse 信號，以便將 EL-plant 元件上的殘餘電荷清除。

下表說明 ELC 指令中設定 ELP 輸出信號的頻率以及 duty cycle 的方式：

(X8,X7,X6)	Pumping clock frequency	(X9,X5,X4)	Duty cycle
000	PH0	000	1/4 duty*
011	FREQB	001	1/3 duty*
100	BCLK	X10	1/2 duty
101	BCLK/2	X11	1/1 duty
110	BCLK/4	100	3/4 duty
111	BCLK/8	101	2/3 duty

* : 如 TM8722 等無'X9' 的 MCU 並無提供 1/4 & 1/3 duty 而只有 3/4 & 2/3 duty。

下表說明 ELC 指令中設定 ELC 輸出信號的頻率以及 duty cycle 的方式：

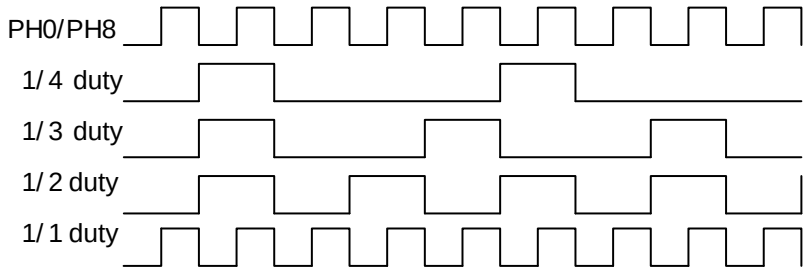
(X3,X2)	Discharge pulse frequency	(X1,X0)	Duty cycle
00	PH8	00	1/4 duty
01	PH7	01	1/3 duty
10	PH6	10	1/2 duty
11	PH5	11	1/1 duty

ELP 以及 ELC 的輸出信號在 MCU 進入 RESET 狀態之後的原始設定如下所示：

ELP 腳位的輸出信號是：PH0 的頻率以及 1/4 duty cycle

ELC 腳位的輸出信號是：PH8 的頻率以及 1/4 duty cycle

輸出信號的頻率與 duty cycle 之間的關係圖如下：



下面的範例說明如何設定以及啟動 EL-plant driver

```
ELC     $110        ; 設定 ELP 的輸出信號頻率為 BCLK 以及 1/3 duty cycle，
                    ; ELC 的輸出信號頻率為 PH8 以及 1/4 duty cycle。
SF      $C          ; 啟動 EL- plant driver 然後使得 MCU 進入 HALT mode。
……
RF      $4          ; 關閉 EL-plant driver
```

2-7. EXTERNAL INT PIN

INT 是 MCU 接收外部中斷信號的輸入腳位，腳位上的邏輯位準可以利用執行 MDX 指令儲存到 data memory。

INT 輸入腳位有三種特性可供選擇使用，內建 pull-low 電阻，內建 pull-high 電阻以及高阻抗輸入。這三種輸入特性可以利用 code option 的方式來選擇。

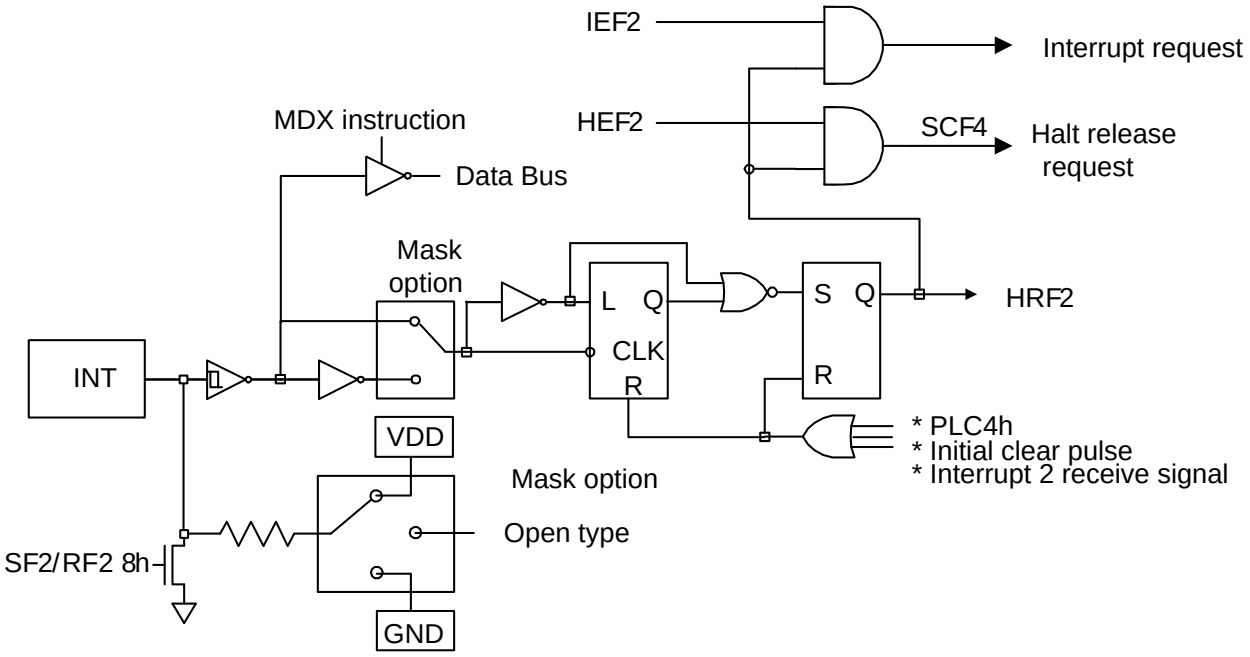
此外 INT 輸入腳位還內建一個可由 SF2 以及 RF2 指令開啟或是關閉的低阻值 pull-low 元件，這個元件的阻值遠低於 pull-high 或是 pull-low 的電阻值。

INT 腳位上的輸入信號只有在改變的時候才會觸發中斷信號，使用者可以利用 code option 來選擇輸入信號的上昇緣或是下降緣來向 MCU 提出中斷請求。

當 INT pin 偵測到輸入信號發生變化後會將 halt release request flag 2 (HRF2)設定為 1。在這個情況下，如果 halt release enable flag (HEF2)已經設定為 1，MCU 就會產生 HALT release 或是 STOP release 並將 start condition flag 2 (SCF2)設定 1。

如果 interrupt enable mode (IEF2)已經設定為 1，當 halt release request flag 2 (HRF2)設定為 1 之後 MCU 就會接受這個中斷請求。

下圖說明 INT 腳位的線路圖：

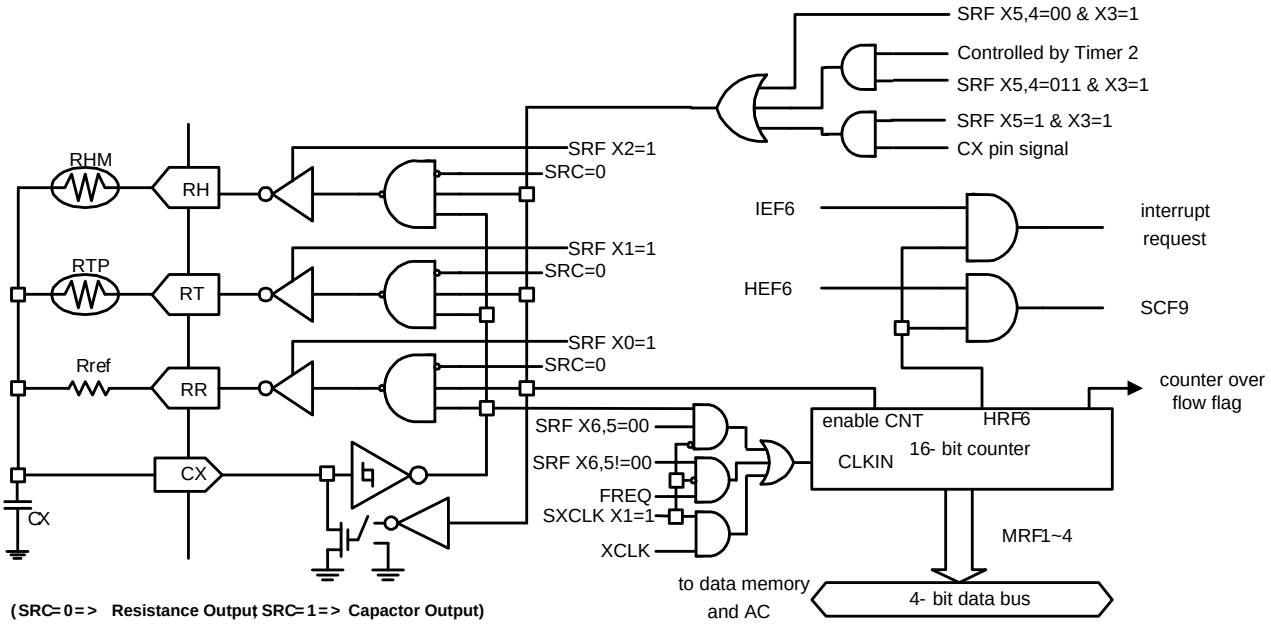


但 INT 若與 SIO 共用 HRF2 則須先以 MWM 7,Ry 設定 INTFEN=1 方可啟動 interrupt 或 halt release。

2-8. RESISTANCE TO FREQUENCY CONVERTER (RFC)

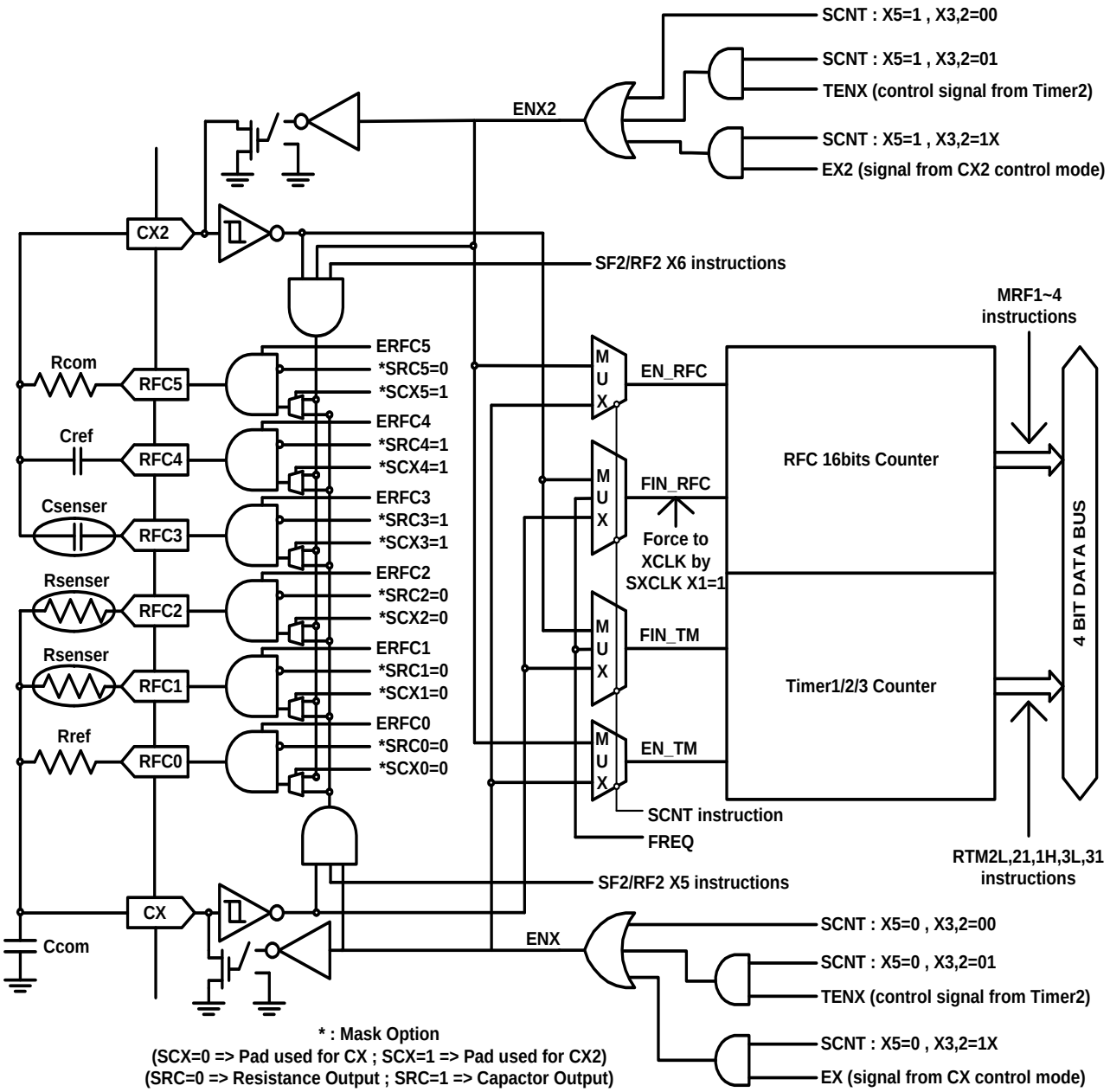
Resistance to frequency converter (RFC)是由 RC 震盪器與 RFC counter 所組成， RC 震盪器中的電阻以及電容是應用線路中的外部零件。RC 震盪器可以將外部元件的電阻值或是電容值轉換成對應的頻率，RFC counter 則將這個頻率轉變成數值供程式運算　使用。

TM87(TM87ML28 也可以由 Mask Option 選擇架構 B)&TM85 系列 MCU 採用架構 A 僅能提供一組 RFC 的功能，整個 RFC 的功能由執行 SRF X3=1 指令之後便能啟動，執行 SRF X3=0 指令則關閉 RFC 的功能。下圖說明 RFC 功能的基本架構與應用線路：



此架構完全由 SRF 一個指令完成所有控制動作。

另如 TM89 系列 MCU 採用架構 B 可提供兩組 RFC 的功能，下圖說明 RFC 功能的基本架構與應用線路：



此架構由 SCNT 負責設定操作模式，由 SF2&RF2 控制 RFC 啟動與結束，而 SRF 僅設定 RFC5~0 的開關。

架構 A 圖中 RR,RT,RH pin 等同架構 B 圖中 RFC0,1,2。

2-8-1. RC 震盪器的設定與啟動

4BIT 系列 MCU 提供兩種不同架構的 RFC 功能，一種架構是用來測試電阻性感測器，如 2-8 小節中架構 B "RFC 基本架構與應用線路" 圖中的 RFC0~RFC2，而另一種架構是用來測試電容性感測器，如 2-8 小節中 "RFC 基本架構與應用線路" 圖中的 RFC3~RFC5。

在用來測試電阻性感測器的 RFC 中，使用一個輸入腳位(CX 或是 CX2)以及一個設定成連接電阻元件的輸出腳位(RFC0~RFC5)，搭配外部的電阻與電容就可組成一個 RC 震盪器。在這個架構下，enable 任何一個 RFC0~5 的 output 腳位之後就可以透過該腳位所連接的電阻對電容進行充電與放電，此充/放電的信號可以經由 CX/CX2 的 input 腳位輸出到 RFC0~5 的 output 腳位而形成 RC 震盪器。

在用來測試電容性感測器的 RFC 中，需要使用一個輸入腳位(CX 或是 CX2)以及設定成連接電阻元件的輸出腳位 RFCn (RFC0~RFC5)以及另一個設定成連接電容元件的輸出腳位 RFCm (RFC0~RFC5)，並搭配外部的電阻與電容就可組成一個 RC 震盪器。在這個架構下，必須先將其中一個連接電容的 RFCm 的 output 腳位設定為 GND，然後 enable RFCn 的 output 腳位。這樣 RFCn 的輸出信號就可以透過該腳位所連接的電阻對 RFCm 所連接的電容進行充電與放電，此充/放電的信號可以經由 CX/CX2 的 input 腳位輸出到 RFCn 的 output 腳位而形成 RC 震盪器。

在使用 RFC 功能的 RC 震盪器之前必須先以 code option 來設定 RFC0~RFC5 在 MCU 硬體上的功能，說明如下：

1. 利用 code option 設定每一個 RFC0~RFC5 的輸出腳位是搭配 CX 或是 CX2 的腳位來組成 RC 震盪器。
2. 利用 code option 來設定每一個 RFC0~RFC5 的輸出腳位在應用線路中是連接電阻元件或是電容元件。
   如果將 RFCn 輸出腳位設定成連接電阻元件，RFCn 腳位可以輸出 0V 或是 VBAT 的電位。
   如果將 RFCn 輸出腳位設定成連接電容元件，RFCn 只能輸出 0V 的電壓。
3. 利用 Mask Option 選擇 CX 或是 CX2 的輸入腳位在停止振盪時是否是在高阻抗輸入模式或是低阻抗輸入模式(在 CX 或是 CX2 輸入腳位上開啟一個 pull low 元件)。
   選擇低阻抗輸入模式這個功能時可以將 CX 或是 CX2 腳位上的電壓位準在 RC 震盪器停止振盪後快速放電至 0V，避免因電容放電速度的不同而影響 RFC counter 的計數結果。
   如果在應用上 CX 或是 CX2 的輸入信號是外部的控制信號時則不能選擇低阻抗輸入模式的功能，否則會發生輸入信號衝突的情況。

依架構 A/B 執行 SRF 指令可以將運算元中所指定的 RR,RT,RH/RFC0 ~ RFC5 腳位設定成輸出模式或是高阻抗模式 (tri-state)，當 RR,RT,RH/RFC0~RFC5 腳位設定成輸出模式之後 RC 震盪器並不會立刻啟動，必須執行 SRF(X3=1)/SF2 指令來開啟對應的 CX 或是 CX2 輸入功能之後才會開始動作。

當程式執行 SRF(X3=0)/RF2 指令關閉對應的 CX 或是 CX2 輸入功能或是將對應的 RFC0~RFC5 腳位設定成高阻抗模式之後 RC 震盪器就會停止動作。

建議在啟動 RC 震盪器的時候，讓程式先設定模式&輸出腳位然後再執行 SRF(X3=1)/SF2 指令，這樣可以獲得比較準確的計數數值。

在同一個 CX 或是 CX2 的輸入腳位上每次只能啟動一組 RC 震盪器，否則會造成震盪頻率的誤差。另外，雖然兩個輸入腳位上(CX 以及 CX2)所分別對應的兩組不同的 RC 震盪器可以同時啟動，但是考慮到同時啟動所產生的電源雜訊會影響 RC 震盪器的輸出頻率，所以不建議同時啟動兩組 RC 震盪器。

當 RC 震盪器啟動後所產生的 clock 會透過 CX 或是 CX2 腳位傳送到 RFC counter 做計數的動作。

2-8-2. RFC COUNTER 的計數功能與控制方式

在 RFC 功能中，RFC counter 是透過 CX 或是 CX2 腳位來接收 RC 震盪器所產生的不同頻率的 clock 信號，並在一段固定時間內計數所接收到的 clock 數。

架構 A 中只有 16-bit RFC counter 而在架構 B 中 RFC counter 有四種不同的型態以及四種不同的控制方式，這些都可以利用執行 SCNT 指令來做設定。

RFC counter 的四種型態分別是：16-bit RFC counter，TMR1，TMR2 以及 TMR3。

RFC counter 的四種控制方式分別是：由程式指令來控制，由 TMR2 來控制，由 CX 或 CX2 輸入腳位的單週期信號來控制，由 CX 或 CX2 輸入腳位的高電位信號來控制以及由 CX 輸入腳位的上升緣信號來控制(目前僅架構 A 提供，如 TM87ML26)。

雖然架構 B 中 SCNT 指令可以將一組 RFC counter 設定成同時由 CX 以及 CX2 pin 控制，或是將 RFC counter 的 clock source 設定成可以同時接收來自 CX 以及 CX2 pin 的 clock 信號，但是當這兩組控制信號或是 clock source 同時啟動後，RFC counter 的動作模式以及接收的信號來源會變得很複雜，因此建議使用這種設定方式時不要同時啟動兩種的控制模式。

當程式完成 RC 震盪器的設定之後，必須先將 RFC counter 設定完成才能執行 SF2 指令(架構 A 為 SRF(X3=1))來啟動 RFC 的功能。在 RFC counter 完成計數動作之後，程式可以利用 MRF1~4 指令來讀取 16-bit RFC counter 的計數值，或是利用讀取 timer 內容值的相關指令來讀取 TMR1~3 的計數值。

無論是選擇何種方式來控制 RFC counter，當程式執行 SF2 指令(架構 A 為 SRF(X3=1)啟動 RFC 功能的時候就會自動將 RFC counter 歸零。

2-8-2-1. 16-BIT RFC COUNTER

這個是專門提供給 RFC 功能所使用的計數器，16-bit RFC counter 在計數過程中發生 overflow 時會將 16-bit RFC counter overflow flag (RFOVF)設定為 1。執行 MSD 指令可以將 RFOVF flag 儲存到 data memory，當程式執行 SF2 指令(架構 A 為 SRF(X3=1))的時候會同時將 RFOVF flag 清除為 0。

16-bit RFC counter 可以利用 code option 的方式選擇發生 overflow 之後是否需要自動重新計數，或是立刻停止動作。若是選擇立刻停止動作，counter 的內容會停在 0000h 的數值。若是選擇自動重新計數，RFOVF 將仍然維持為 1，直到下一次 overflow 發生時才會清除為 0。因此可以將 RFOVF 看成是 16-bit RFC counter 中第 17 級的輸出信號。

程式可以利用 MRF1~4 指令來讀取 16-bit RFC counter 的計數值，並將計數值儲存到 data memory。

2-8-2-2. TIMER 作為 RFC COUNTER

在架構 B 中執行 SCNT 指令可以將 TMR1 或 TMR2 或 TMR3 設定成為 RFC 功能的 counter 之用，並且還可以選擇 CX 或是 CX2 pin 作為 RFC counter 的信號輸入腳位，詳細的設定方式請參閱 SCNT 的指令使用說明。

無論是 6-bit, 12-bit 或是 18-bit 的 timer 都可以作為 RFC counter 之用，而且程式中也可以同時啟動兩組的 RFC 功能，其中一組使用標準的 16-bit RFC counter，另一組則可以使用 timer 做為 RFC counter。

當 timer 作為 RFC counter 使用時，timer 的 clock source 的信號來源以及 preset data 是由 RFC 功能的硬體線路自動設定，而且只能從最大值 (3Fh, FFFh, 3FFFFh) 開始倒數至 0。

除了設定 timer 的 clock source 的信號來源 以及起始值與一般 timer 的設定方式不同之外，其他動作都與一般的 timer 相同，例如 underflow 發生時會停止動作並設定 timer halt release request flag (HRFn)，可以設定 re-load 功能以及讀取 timer 的內容值等等。

請注意，當 timer 作為 RFC counter 使用時，切勿再以一般 timer 的相關指令對該 timer 進行設定及控制，否則將會影響 RFC counter 的動作。此外也不可在尚未將 RFC 功能的 counter 內容值讀取完畢之前執行一般 timer 的相關指令，否則會毀損原先的 counter 內容值。

2-8-2-3. 由程式指令來控制 RFC COUNTER

在這個控制方式下，RFC counter 的 clock source 必須是選擇接收 CX 或是 CX2 腳位上的信號然後利用 SF2(架構 A 為 SRF(X3=1)或是 RF2(架構 A 為 SRF(X3=0)指令來控制 counter 的啟動或是停止。

當執行 SF2 20h (X5=1) (架構 A 為 SRF(X3=1)或是 SF2 40h (X6=1) (若為架構 B)指令時，任何型態的 RFC counter 的內容值都會在歸零之後開始計數來自 CX 或是 CX2 的信號。在執行 RF2 20h (X5=1(架構 A 為 SRF(X3=0))或是 RF2 40h (X6=1) (若為架構 B)指令之後，RFC counter 就會停止計數。

範例：

本範例將在一段固定時間內計數來自 CX pin 的 clock 數目，利用 SF2(架構 A 為 SRF(X3=1)指令啟動計數功能，計數的時間長度由 TMR1 來設定。

程式中會在計數的時間長度結束後檢查 16-bit RFC counter 是否發生 overflow，如果尚未發生 overflow 則讀取 16-bit RFC counter 的內容值；如果已經發生 overflow 則會將計數的時間長度縮短。本範例會使用 RFC0 以及 CX 腳位來組成 RC 震盪器。

架構 A:

```
                                    ; TMR1 設定計數的時間長度

  LDS       0, 0                    ; TMR1 的 clock source 是 PH9
    LDS     1, 3                    ; TMR1 的起始值是 3F
    LDS     2, 0Fh
    SHE     2                       ; 設定 TMR1 的 underflow 產生 halt release
RE_CNT:
    LDA     0
    OR*     1                       ; 製作 TMR1 的起始值
    TMS     2                       ; 啟動 TMR1
    SRF     9                       ; 設定 RR 成輸出模式並啟動 CX pin 的 RFC 功能
    HALT
    SRF     1                       ; TMR1 underflow 時停止 CX pin 的 RFC 功能
    MRF1    10h                     ; 讀取 16-bit RFC counter
    MRF2    11h
    MRF3    12h
    MRF4    13h
    MSD     20h
    JB2     CNT1_OF                 ; 檢查 RFC counter 是否 overflow
    JMP     DATA_ACCEPT
CNT1_OF:
    DEC*    2                       ; 縮短 TMR1 設定的時間長度
    LDS     20h, 0
    SBC*    1
    JZ      CHG_CLK_RANGE           ; 改變 TMR1 的 clock source
    PLC     1                       ; 清除 TMR1 的 halt release request flag
```

```
    JMP         RE_CNT
```

架構 B:

```
                                ; TMR1 設定計數的時間長度
    LDS     0, 0                ; TMR1 的 clock source 是 PH9
    LDS     1, 3                ; TMR1 的起始值是 3F
    LDS     2, $0F
    SHE 2                       ; 設定 TMR1 的 underflow 產生 halt release
    SCNT    $00                 ; 設定 CX pin， 程式控制方式，16-bit RFC counter
    SRF $01                     ; 設定 RFC0 成輸出模式
RE_CNT:
    LDA     0
    OR*     1                   ; 製作 TMR1 的起始值
    TMS 2                       ; 啟動 TMR1
    SF2     $20                 ; 啟動 CX pin 的 RFC 功能
    HALT
    RF2     $20                 ; TMR1 underflow 時停止 CX pin 的 RFC 功能
    MRF1    $10                 ; 讀取 16-bit RFC counter
    MRF2    $11
    MRF3    $12
    MRF4    $13
    MSD     $20
    JB2     CNT1_OF             ; 檢查 RFC counter 是否 overflow
    JMP     DATA_ACCEPT
CNT1_OF:
    DEC*    2                   ; 縮短 TMR1 設定的時間長度
    LDS     $20, 0              ; 設定 AC=0
    SBC*    1
    JZ   CHG_CLK_RANGE          ; 改變 TMR1 的 clock source
    PLC     1                   ; 清除 TMR1 的 halt release request flag
    JMP RE_CNT
```

2-8-2-4. TMR2 來控制 RFC COUNTER

TMR2 所能控制的 RFC counter 種類有 16-bit RFC counter(架構 A 固定 counter)，TMR1 或是 TMR3，但不可以是 TMR2 本身。

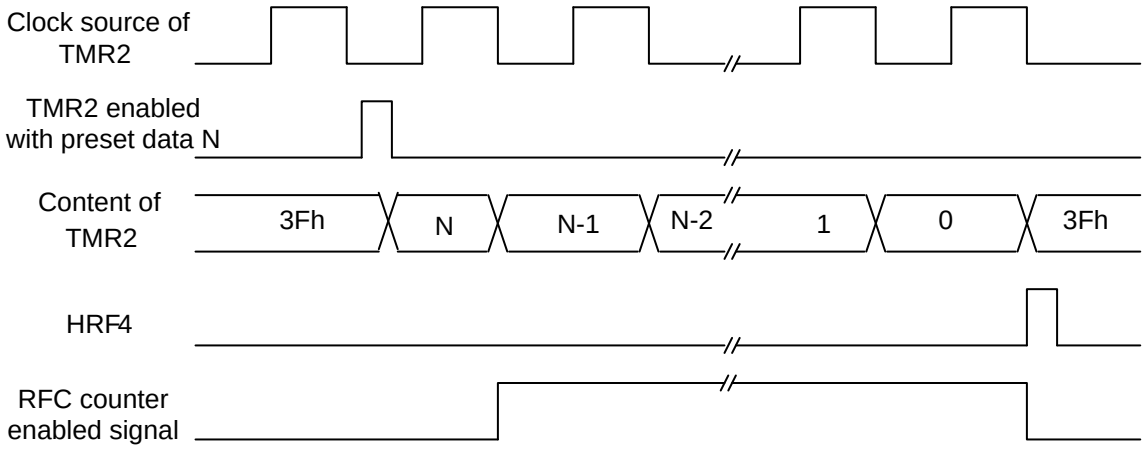
所有的 timer 中只有 TMR2 (6-bit，12-bit 或是 18-bit)可以用來控制 RFC counter 的啟動或是關閉，程式中可以利用 SCNT(架構 A 為 SRF(X5,4=01))指令來設定這項控制功能。

在 RFC 功能選用這種控制方式時，RFC counter 不會在程式執行 SF2(架構 A 為 SRF(X3=1)指令啟動 RFC 的功能之後馬上啟動計數，必須等到 TMR2 啟動以後並等到 TMR2 的 clock source 出現第一個下降緣信號才會啟動 RFC counter 的計數動作。等到 TMR2 發生 underflow 並且停止動作的時候才會將 RFC counter 關閉。這樣 TMR2 就可以提供一個很精確的計數時間來控制 RFC counter，而不會出現一般 timer 在計數時會產生一個小於 clock source 信號週期的誤差。

在此控制方式下，若 RFC counter 的 clock source 設定為 CX 或是 CX2 pin 時，由 CX 或是 CX2 pin 所組成的 RC 振盪器會與 RFC counter 一樣會在程式執行 SF2(架構 A 為 SRF(X3=1))指令啟動 RFC 的功能之後同時由 TMR2 來控制啟動與關閉。

但若將 RFC counter 的 clock source 設定為頻率產生器的輸出(FREQ)時，架構 A 因為應用設定目的在於藉由設定 TMR2 clock source 為 PH3/5/7/9/11/13/15 等來自穩定 32KHz X'tal SLOW CLOCK 以及 FREQ=BCLK 進入 FAST Mode 達成由 MCU 內部自行計算 FAST RC 頻率，為避免干擾風險故不會啟動 CX 震盪，而架構 B 因當初應用設計用途為計算 CX 或是 CX2 多週期應用搭配設定 TMR2 clock source 為 CX 或是 CX2 因此須先動作才可啟動 RFC counter，另外 TM87M28 亦可執行 SXCLK 指令設定 X1=1 直接切換 Crfc=XCLK 藉此可在不佔用 FREQ 條件下同樣由 TMR2 control RFC 模式由 MCU 內部自行計算 FAST RC 頻率，故由 CX 或是 CX2 pin 所組成的 RC 振盪器會直接由 SF2，RF2 指令來啟動與關閉，不會等到 TMR2 的 clock source 出現第一個下降緣信號才啟動，但是當 TMR2 clock source 不是 CX 以及 CX2 時，若需要仍然可藉由關閉輸出腳位以及由 IO 輸出 GND 電位避免 CX 以及 CX2 產生震盪。

下圖說明 TMR2 控制 RFC counter 的 timing 圖：



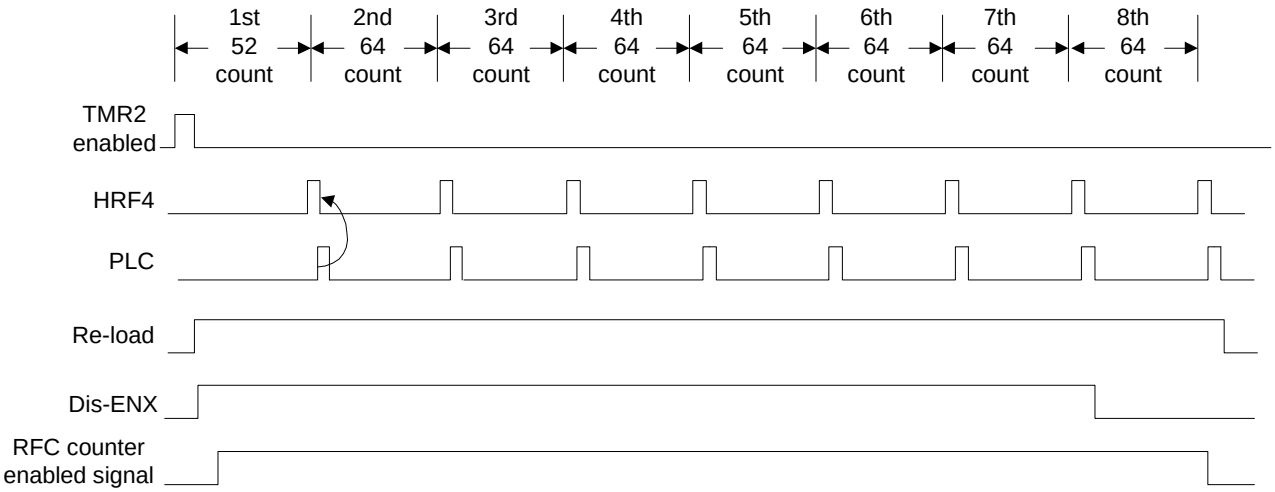
TMR2 同樣可以使用 re-load 功能來控制 RFC counter，因為控制 RFC counter 的信號(上圖中的 "RFC counter enable signal")會在 TMR2 發生 underflow 時被清除為 0，因此必須另外設定 Dis-ENX flag 來避免 RFC counter enable signal 被 HRF4 flag 清除為 0，以便 RFC counter 能持續動作。

程式在啟動 TMR2 的 re-load 功能之後必須緊接著執行 SF2 2h 指令，以便將 Dis-ENX flag 設定為 1，這樣 TMR2 在 underflow 時所產生的 halt release request flag 4 (HRF4)就不會 影響 RFC counter 的動作。

當程式要關閉 TMR2 的 re-load 功能時，必須在 TMR2 產生最後一次 underflow 之前先執行 RF2 2h 指令將 Dis-ENX flag 清除為 0。這樣控制 RFC counter 的信號就會在 TMR2 產生 最後一次 halt release request flag 4 (HRF4)時被清除為 0，而 RFC counter 也會同步停止動作。

下面的範例說明如何利用 TMR2 的 re-load 功能來控制 16-bit RFC counter：

如果程式希望 TMR2 的倒數計數值為 500 時，可以將 500 的倒數值規劃成一次 52 的倒數值以及 7 次 64 的倒數值(64*7 + 52)。在這個範例中程式會進入 HALT mode 來等待 TMR2 產生 underflow。



架構 A:

```
                LDS     0,0             ; 將 data memory 位址 0 的內容值作為 TMR2 underflow
                                        ; 次數的計數器，並將其歸零。
                PLC     $10             ; 將 HRF4 清除為 0
                SHE     $10             ; 設定 TMR2 underflow 可以讓 MCU 產生 HALT release
                SRF     $19             ; 設定 TMR2 來控制 16-bit RFC counter，設定 RR 成輸出模式
                                        ; 並啟動 CX 腳位上的 RFC 功能
                TM2X    $34             ; 設定 TMR2 的起始值(52)， clock source 選 PH9，然後
                                        ; 啟動 TMR2
                SF2     3               ; 開啟 TMR2 的 re-load 功能並將 Dis-ENX flag 設定為 1
RE_LOAD:
                HALT
                INC*    0               ; 將 TMR2 underflow 計數器加 1
                PLC     $10             ; 將 HRF4 清除為 0
                LDS     $20, 7
                SUB     0               ; 當第七次的 TMR2 underflow 發生後將 Dis-ENX flag
                                        ; 清除為 0
                JNZ     NOT_RESET_DED
                RF2     2               ; 將 Dis-ENX flag 清除為 0
NOT_RESET_DED:
                LDA     0               ; 將 TMR2 underflow 計數器除內容值寫入 AC
                JB3     END_TM1         ; i 檢查 TMR2 underflow 次數是否等於 8 次
                JMP     RE_LOAD
END_TM1:
                RF2     1               ; 關閉 TMR2 的 re-load 功能
```

架構 B:

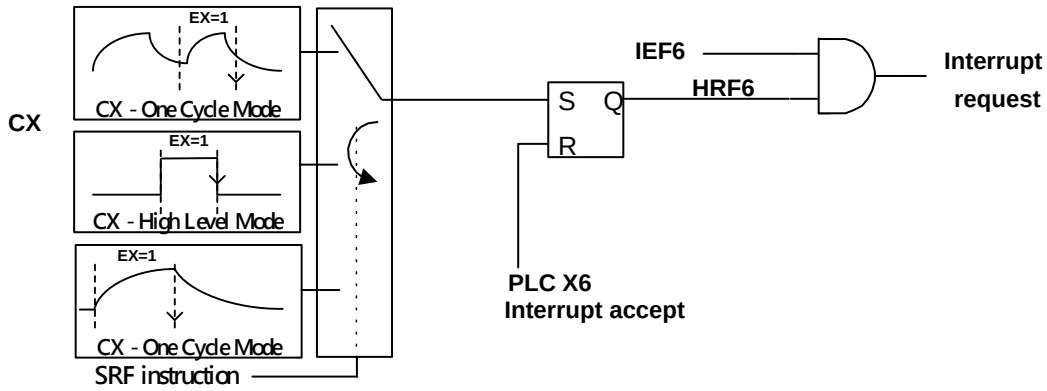
```
                LDS     0,0             ; 將 data memory 位址 0 的內容值作為 TMR2 underflow
                                        ; 次數的計數器，並將其歸零。
                PLC     $10             ; 將 HRF4 清除為 0
                SHE     $10             ; 設定 TMR2 underflow 可以讓 MCU 產生 HALT release
                SCNT    $04             ; 設定 TMR2 來控制 16-bit RFC counter，
                                        ; 且 RFC counter clock source 為 CX。
                SRF     $01             ; 設定 RFC0 成輸出模式
                SF2     $20             ; 啟動 CX 腳位上的 RFC 功能
```

```
                TM2X  $34          ; 設定 TMR2 的起始值(52)， clock source 選 PH9，然後
                                   ; 啟動 TMR2
                SF2   3            ; 開啟 TMR2 的 re-load 功能並將 Dis-ENX flag 設定為 1
RE_LOAD:
                HALT
                INC*  0            ; 將 TMR2 underflow 計數器加 1
                PLC   $10          ; 將 HRF4 清除為 0
                LDS   $20, 7
                SUB   0            ; 當第七次的 TMR2 underflow 發生後將 Dis-ENX flag
                                   ; 清除為 0
                JNZ   NOT_RESET_DED
                RF2   2            ; 將 Dis-ENX flag 清除為 0
NOT_RESET_DED:
                LDA   0            ; 將 TMR2 underflow 計數器除內容值寫入 AC
                JB3   END_TM1      ; i 檢查 TMR2 underflow 次數是否等於 8 次
                JMP   RE_LOAD
END_TM1:
                RF2   1            ; 關閉 TMR2 的 re-load 功能
```

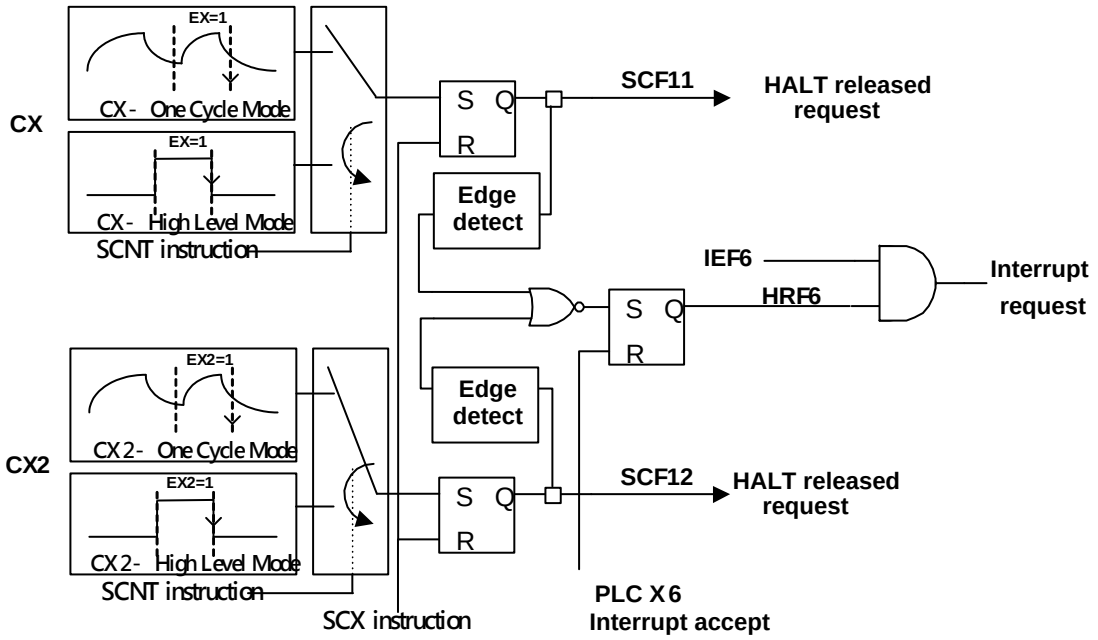
2-8-2-5. 由 CX 或是 CX2 輸入腳位的外部信號直接來控制 RFC COUNTER

在這個控制方式下，送到 CX 或是 CX2 的外部信號變成是直接控制 RFC counter 開始或是停止計數的信號，而且 counter 會固定計數來自頻率產生器的輸出信號(FREQ)。

下圖說明架構 A 利用 CX/CX2 輸入腳位的單一週期信號或是高電位信號或是上升緣信號來控制 RFC counter 的架構及相關 flag：



下圖說明架構 B 利用 CX/CX2 輸入腳位的單一週期信號或是高電位信號來控制 RFC counter 的架構及相關 flag：



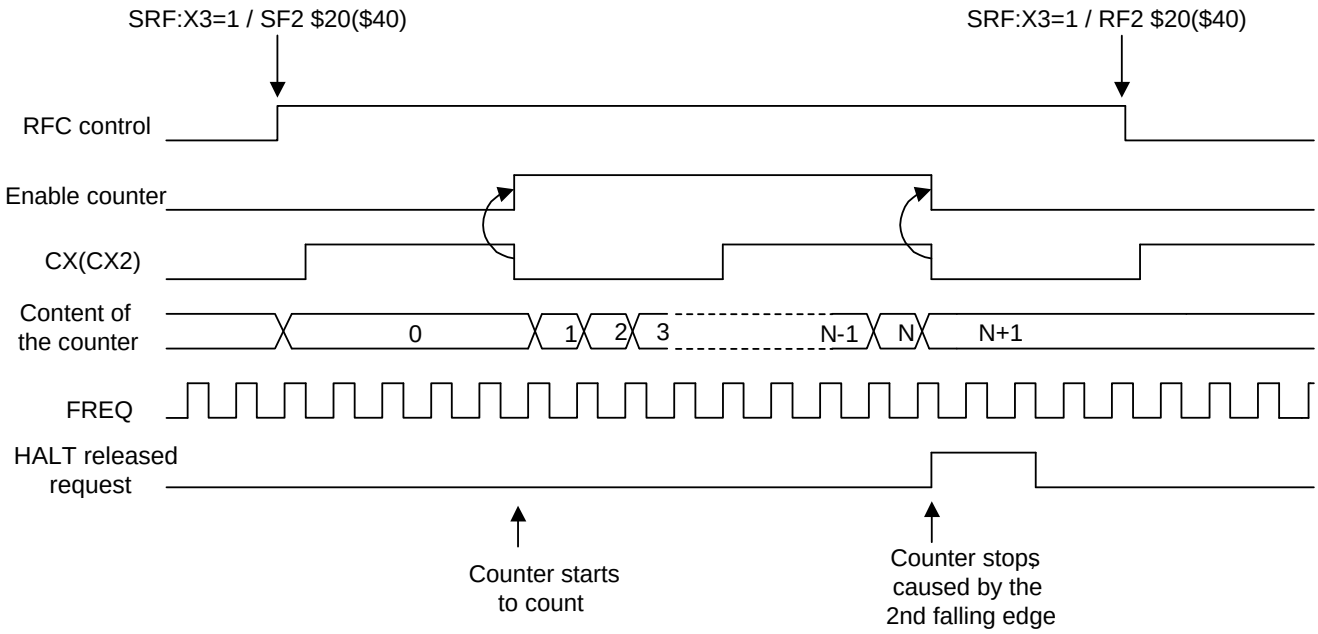
2-8-2-5-1. 由 CX 或是 CX2 輸入腳位的單一週期信號直接來控制 RFC COUNTER

在執行 SF2(架構 A 為 SRF(X3=1))指令啟動 RFC 功能之後，RFC counter 並不會立刻開始計數，必須等到第一個下降緣的信號送到 CX 或是 CX2 腳位之後才會開始計數。當第二個下降緣的信號送到 CX 或是 CX2 腳位之後 RFC counter 就會停止計數的動作。程式必須執行 RF2(架構 A 為 SRF(X3=0))指令才能停止　整個 RFC 的功能。

架構 B 因提供 2 組 RFC 故需要執行 SCX 指令提供 SCF11,12 分辨，當程式執行 SCX 指令將 switch enable flag 0, 1 (SEF0, 1)設定為 1 之後，在第二個下降緣信號送到 CX 或是 CX2 腳位的時候會將 start condition flag 11, 12 (SCF11, 12)設定為 1，而且同時也會將 halt release request flag 6(HRF6)設定為 1。執行 SCX 指令時會同時將 SCF11 以及 SCF12 清除為 0。

在上述情況下，如果 interrupt enable flag 6 (IEF6)已經設定為 1，在 halt release request flag 6 (HRF6)設定為 1 之後 MCU 就會接受這個中斷請求。

下圖說明利用 CX/CX2 的下降緣信號來控制 RFC counter 的 timing：



範例：

架構 A:

```
        SCC     0h      ; 設定頻率產生器的 clock source 是 PH0
        FRQX    1, 5    ; 設定頻率產生器輸出頻率 FREQ = (PH0/6) / 3
                        ; 頻率產生器設定值為 5 & FREQ 為 1/3 duty 波形.
        SHE     40h     ; 設定 RFC 動作結束產生 halt release
        SRF     2ch     ; 設定用 CX 上的單週期信號控制 RFC counter，
                        ;設定 RH 成輸出模式，並啟動 CX 震盪
        HALT
        PLC     40h     ;在 CX 第二個下降緣時產生 halt release, 執行 PLC 指令清除
HRF6
        MRF1    10h     ; 讀取 RFC 16bits counter 內容
        MRF2    11h
        MRF3    12h
        MRF4    13h
```

架構 B:

```
        SCC     $0      ; 設定頻率產生器的 clock source 是 PH0
        FRQX    1, 5    ; 設定頻率產生器輸出頻率 FREQ = (PH0 / 6)/3
                        ; 頻率產生器設定值為 5 & FREQ 為 1/3 duty 波形.
        STM     3       ; 將 TMR2 設定成 18-bit
        SHE     $10     ; 設定 TMR2 underflow 產生 halt release
        SCX     1       ; 清除 SCF11,12 並且設定 SEF0 flag
        SCNT    $0a     ; 設定用 CX 上的單週期信號控制 RFC counter 啟動 ,
                        ; TMR2=RFC counter,
                        ; clock source=FREQ
        SRF     $04     ; 設定 RFC2 為輸出模式
        SF2     $20     ; 啟動 CX 的 RFC 功能
        HALT
```

```
        MCX     $6f     ; 讀取 SCF6
        JB1     TM2_UF  ; 檢查 TMR2 是否 underflow
        MAF     $7f     ; 讀取 SCF11
        ANDI    $7f,1   ;
        JZ      NOT_CX  ; 檢查 CX 控制信號是否結束
        SCX     1       ; 清除 SCF11,12 並且設定 SEF0 flag
        PLC     50h     ; 清除 HRF6
        RTM2L   $10     ; 讀取 TMR2 內容
        RTM21   $11
        RTM1H   $12
        RTM3L   $13
        RTM31   $14
.....
TM2_UF:                 ; Timer2 已經發生 underflow
```

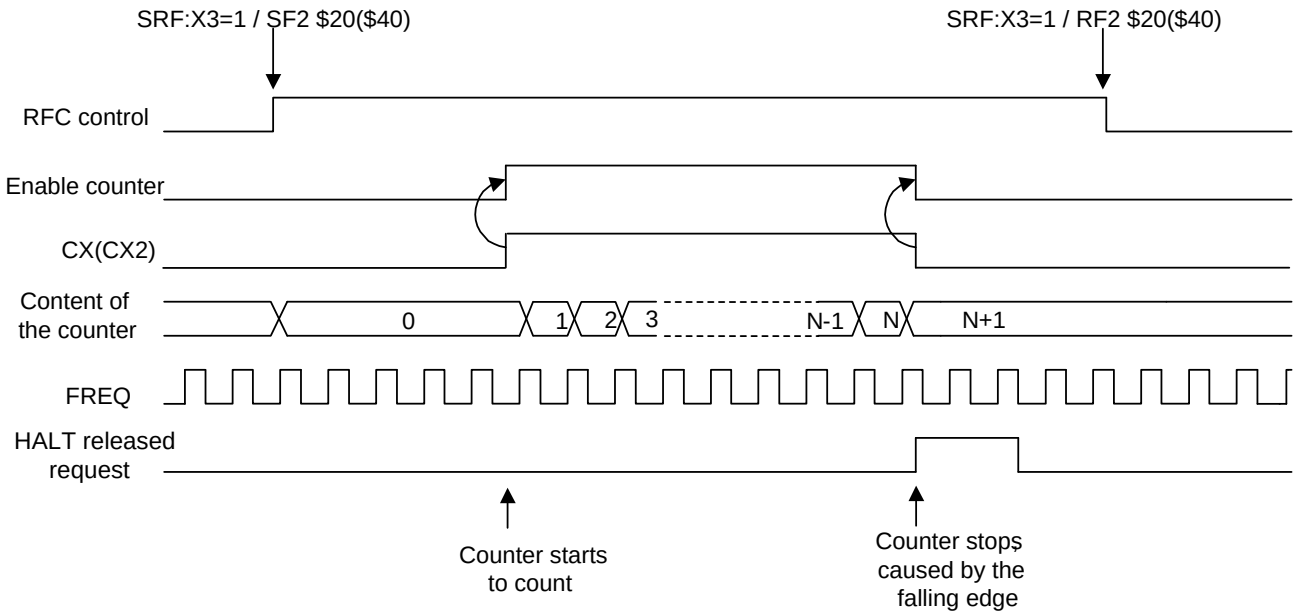
2-8-2-5-2. 由 CX 或是 CX2 輸入腳位的高電位信號直接來控制 RFC COUNTER

在執行 SF2(架構 A 為 SRF(X3=1))指令啟動 RFC 功能之後，RFC counter 並不會立刻開始計數，必須等到第一個高電位的信號送到 CX 或是 CX2 腳位時才會開始計數。當 CX 或是 CX2 腳位上的信號變成低電位時 RFC counter 就會停止計數的動作。

架構 B 因提供 2 組 RFC 故需要執行 SCX 指令提供 SCF11,12 分辨，當程式執行 SCX 指令將 switch enable flag 0, 1 (SEF0, 1)設定為 1 之後，在下降緣的信號送到 CX 或是 CX2 腳位時候會將 start condition flag 11, 12 (SCF11, 12)設定為 1，而且同時也會將 halt release request flag 6(HRF6)設定為 1。執行 SCX 指令時會同時將 SCF11 以及 SCF12 清除為 0。

在上述情況下，如果 interrupt enable flag 6 (IEF6)已經設定為 1，在 halt release request flag 6 (HRF6)設定為 1 之後 MCU 就會接受這個中斷請求。

下圖說明利用 CX/CX2 的高電位信號來控制 RFC counter 的 timing：



範例：

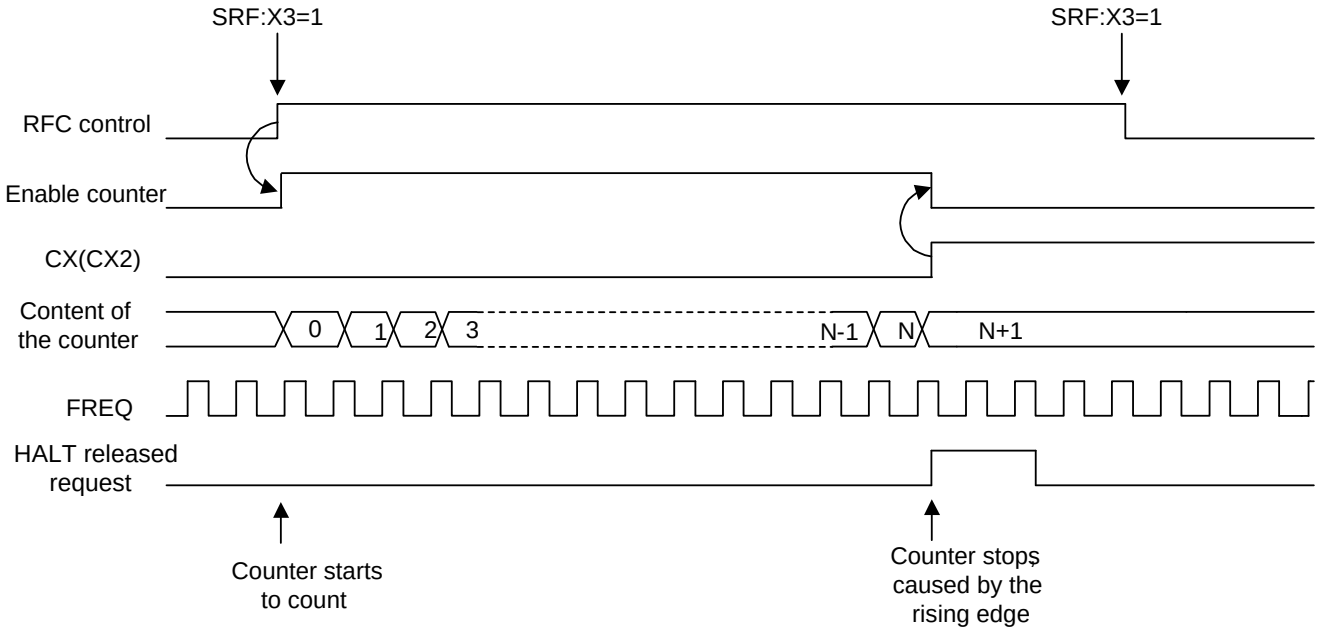
```
        SCC         $0          ; 設定頻率產生器的 clock source 是 PH0
        FRQX        3, 0        ; 設定頻率產生器輸出頻率 FREQ = PH0
        STM         2           ; 將 TMR3 設定成 12-bit
        SHE         $80         ; 設定 TMR3 underflow 產生 halt release
        SCX         2           ; 清除 SCF11,12 並且設定 SEF1 flag
        SCNT        $2f         ; CX2 高電位控制 RFC counter, TMR3=RFC counter,
                                ; clock source=FREQ
        SRF         $08         ; 設定 RFC3 為輸出模式
        SF2         $40         ; 啟動 CX2 的 RFC 功能
        HALT
        MDX         $6f         ; 讀取 SCF10
        JB3         TM3_UF      ; 檢查 TMR3 是否 underflow
        MAF         $7f         ; 讀取 SCF12
        ANDI        $7f,2       ;
        JZ          NOT_CX      ; 檢查 CX2 控制信號是否結束
        SCX         2           ; 清除 SCF11,12 並且設定 SEF1 flag
        PLC         50h         ; 清除 HRF6
        RTM3L       $10         ; 讀取 TMR3 內容
        RTM31       $11
        RTM1H       $12
.....
TM3_UF:                         ; Timer3 已經發生 underflow
```

2-8-2-5-3. 由 CX 輸入腳位的上升緣信號直接來控制 RFC COUNTER

目前僅架構 A 有提供，在執行 SRF(X3=1)指令啟動 RFC 功能之後，RFC counter 會立刻開始計數，當 CX 由 GND 開始上升至使 input 轉態之後 RFC counter 就會停止計數的動作。程式必須執行 SRF(X3=0)指令才能停止 整個 RFC 的功能。為確保下次由 GND 開始上升必須先確保執行完全下降至 GND 後才可以繼續執行 SRF(X3=1)。可由 code option 選擇 CX 的輸入腳位在停止時為低阻抗輸入模式以加速放電。

在上述情況下，如果 interrupt enable flag 6 (IEF6)已經設定為 1，在 halt release request flag 6 (HRF6)設定為 1 之後 MCU 就會接受這個中斷請求。

下圖說明利用 CX/CX2 的上升緣信號來控制 RFC counter 的 timing：



範例(架構 A)：

```
SCC     0h      ; 設定頻率產生器的 clock source 是 PH0
FRQX    1, 5    ; 設定頻率產生器輸出頻率 FREQ = (PH0/6) / 3
                ; 頻率產生器設定值為 5 & FREQ 為 1/3 duty 波形.
SHE     40h     ; 設定 RFC 動作結束產生 halt release
SRF     7ch     ; 設定用 CX 上的單週期信號控制 RFC counter，
                ;設定 RH 成輸出模式，並啟動 CX 震盪
HALT
PLC     40h     ;在 CX 上升緣時產生 halt release, 執行 PLC 指令清除 HRF6
MRF1    10h     ; 讀取 RFC 16bits counter 內容
MRF2    11h
MRF3    12h
MRF4    13h
```

2-8-2-6. 經 TMR2 模式實現由 CX 或是 CX2 輸入腳位的多週期信號來控制 RFC COUNTER

在架構 B 下，當 SCNT 指令將 RFC counter 設定是由 TMR2 控制以及將 RFC counter 的 clock source 設定成頻率產生器的輸出信號(FREQ)時，如果再將 TMR2 的 clock source 設定成 CX 或是 CX2 pin，這樣就可以利用 CX 或是 CX2 pin 透過 TMR2 間接控制 RFC counter。這樣便可將 CX/CX2 單週期控制模式衍生成為多週期的控制模式。

範例：(計算 CX 輸入信號的頻率：以 PH0 做為計時基準去計算 CX 腳位上 100 個 input clock 的時間長度)

```
SCC     $0      ; 設定頻率產生器的 clock source 是 PH0
FRQX    3, 0    ; 設定頻率產生器輸出頻率 FREQ = PH0
SCNT    $14     ; sets CX：信號源=FREQ , 控制方式=TMR2 ,
```

```
                                ; counter =16-bit RFC counter
    SRF         $20             ; 將 RFC5 設定成輸出模式.
    STM         1               ; TMR2=12 bit timer
    SF2         $20             ; 啟動 CX 的 RFC 功能(CX 開始產生振盪)
    T2XH
    setdat      $8064           ; TMR2 的 clock source=CX , 起始值=100，啟動 TMR2
    SHE         $10             ; 設定 TMR2 可以產生 HALT release
    HALT
    RF2         $20             ; 關閉 CX 的 RFC 功能(CX 此時才停止振盪)
    MSD         $6F
    JB2         CNT1_OF         ; 檢查 16-bit RFC counter 是否 overflow
    PLC         $10             ; 清除 TMR2 的 halt release request flag
    MRF1        $10             ; 讀取 16-bit RFC counter 的內容值
    MRF2        $11
    MRF3        $12
    MRF4        $13
.....
CNT1_OF:                        ; 16-bit RFC counter 已經發生 overflow
```

2-8-2-7. RFC 的應用範例

架構 B 範例一：(TMR2 同時控制兩組 RFC counter)

註：當程式同時啟動兩組 RFC 功能中的 RC 震盪器的時候會產生較大的電源雜訊而影響 RC 震盪器的輸出頻率，所以不建議同時啟動兩組 RC 震盪器。

```
    SCNT        $04             ; sets CX：信號源=CX，控制方式=TM2，
                                ; counter=16-bit RFC counter。
    SCNT        $24             ; sets CX2：信號源=CX2，控制方式=TM2，
                                ; counter=TMR3。
    STM         2               ; TMR3=12 bit timer。
    SRF         $11             ; 將 RFC0 以及 RFC4 設定成輸出模式。
    SF2         $60             ; 同時啟動 CX 以及 CX2 兩組 RFC 功能。
    SHE         $90             ; 設定 TMR2, TMR3 可以產生 HALT release。
    TM2X        $1df            ; 設定 TMR2 起始值=1Fh ，clock source=PH13，
                                ; 啟動 TMR2。
    HALT
    MDX         $6f             ; 讀取 SCF10 (TM3)。
    JB3         TM3_UF          ; 檢查 TMR3 是否 underflow。
    MSD         $6f
    JB2         CNT1_OF         ; 檢查 16-bit RFC counter 是否 overflow。
    PLC         $10             ; 清除 TMR2 的 halt release request flag。
    MRF1        $10             ; 讀取 16-bit RFC counter 的內容值。
    MRF2        $11
    MRF3        $12
    MRF4        $13
    RTM3L       $20             ; 讀取 12-bit TMR3 的內容值。
    RTM31       $21
    RTM1H       $22
.....
```

```
CNT1_OF:                    ; 16-bit RFC counter 已經發生 overflow
……
TM3_UF:                     ; Timer3 已經發生 underflow
```

架構 B 範例二：(CX 與 CX2 的輸入信號分別控制兩個 RFC counter)

```
        SCC         $0      ; 設定頻率產生器的 clock source 是 PH0。
        FRQX        3, 0    ; 設定頻率產生器輸出頻率 FREQ = PH0。
        SCNT        $1d     ; sets CX：信號源=FREQ，控制方式=CX High-Level，
                            ; counter =TMR1。
        SCNT        $28     ; sets CX2：信號源=FREQ，
                            ; 控制方式=CX2 One-Cycle，
                            ; counter =16-bit RFC counter。
        SRF         $21     ; 將 RFC0 以及 RFC5 設定成輸出模式。
        SCX         3       ; 清除 SCF11,12 以及設定 SEF1,0。
        SF2         $60     ; 同時啟動 CX 以及 CX2 兩組 RFC 功能。
        SIE*        $40     ; 設定 RFC 功能的 interrupt release request flag。
        HALT

... ... ... ... ...

.org $0028                  ; RFC interrupt address。
        MAF         $60     ; 讀取 SCF11, SCF12。
        JB0         CX_RLS  ; 檢查 CX 是否產生中斷。
        JB1         CX2_RLS ; 檢查 CX2 是否產生中斷。
... ... ... ... ...

CX_RLS:
        SCX         3       ; 清除 SCF11,12 以及設定 SEF1,0。
... ... ... ... ...

        SRF         $22     ; 將 RFC1 以及 RFC5 設定成輸出模式。
        SF2         $60     ; 同時啟動 CX 以及 CX2 兩組 RFC 功能。
        LDA         $60     ; 檢查 SCF12。
        JB1         CX2_RLS ; 檢查 CX2 是否產生中斷。
        SIE*        $40     ; 設定 RFC 功能的 interrupt release request flag。
        RTS
```

架構 B 範例三：(RFC 功能中使用電容性感測器)

本範例採用的應用線路圖如 2-8 小節中 “RFC 基本架構與應用線路” 所示，控制方式採用的是由程式指令來控制 RFC counter。

```
        SHE         $2      ; 設定 TMR1 underflow 產生 halt release。
        SCNT        $20     ; 設定 CX2 pin， 程式控制方式，16-bit RFC counter。
        SRF         $30     ; 設定 RFC4，RFC5 成輸出模式，爲測試電容性感測器
                            ; 而搭建 RC 震盪器。
                            ; 在 code option 中，這兩個輸出腳位必須將 RFC5
                            ; 設定成連接電阻元件的模式，而將 RFC4 設定成連
                            ; 接電容元件的模式。
```

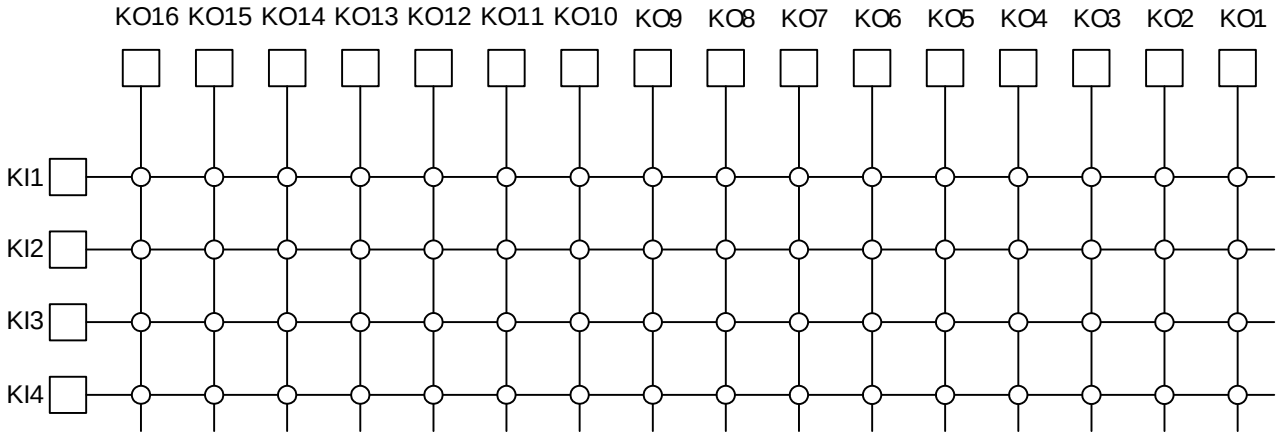
```
        SF2             $40             ; 啟動 CX2 pin 的 RFC 功能。
        TMSX            $7F             ; TMR1 的 clock source 是 PH3，計數值爲$3F。
        HALT
        RF2             $40             ; TMR1 underflow 時停止 CX2 pin 的 RFC 功能。
        MRF1    $10                     ; 讀取 16-bit RFC counter。
        MRF2    $11
        MRF3            $12
        MRF4    $13
        MSD             $14
        JB2             CNT1_OF         ; 檢查 RFC counter 是否 overflow。
        JMP             DATA_ACCEPT
CNT1_OF:
...
...
DATA_ACCEPT:
...
...
```

2-9. KEY MATRIX掃描功能

Key matrix 掃描功能是由 KI1 ~ KI4 四個輸入腳位，16 個輸出腳位(與 LCD 輸出腳位 SEG1 ~ SEG16 共用，為方便說明，後面文件中改以 KO1~KO16 稱呼)以及外部的矩陣式鍵盤所組成。

KO1 ~ KO16 上的掃瞄信號如 TM87 系列會隨 LCD Alternating Frequency 轉態週期而變或如 TM89 系列會以固定的 PH6 週期從 SEG1~SEG16 的輸出波形中擷取一小段時間送出(請參閱 3-4-1)，KI~K4 的輸入腳位則會在同樣的時間週期去讀取這些掃描信號，以判斷是否有鍵盤的按鍵被壓下。

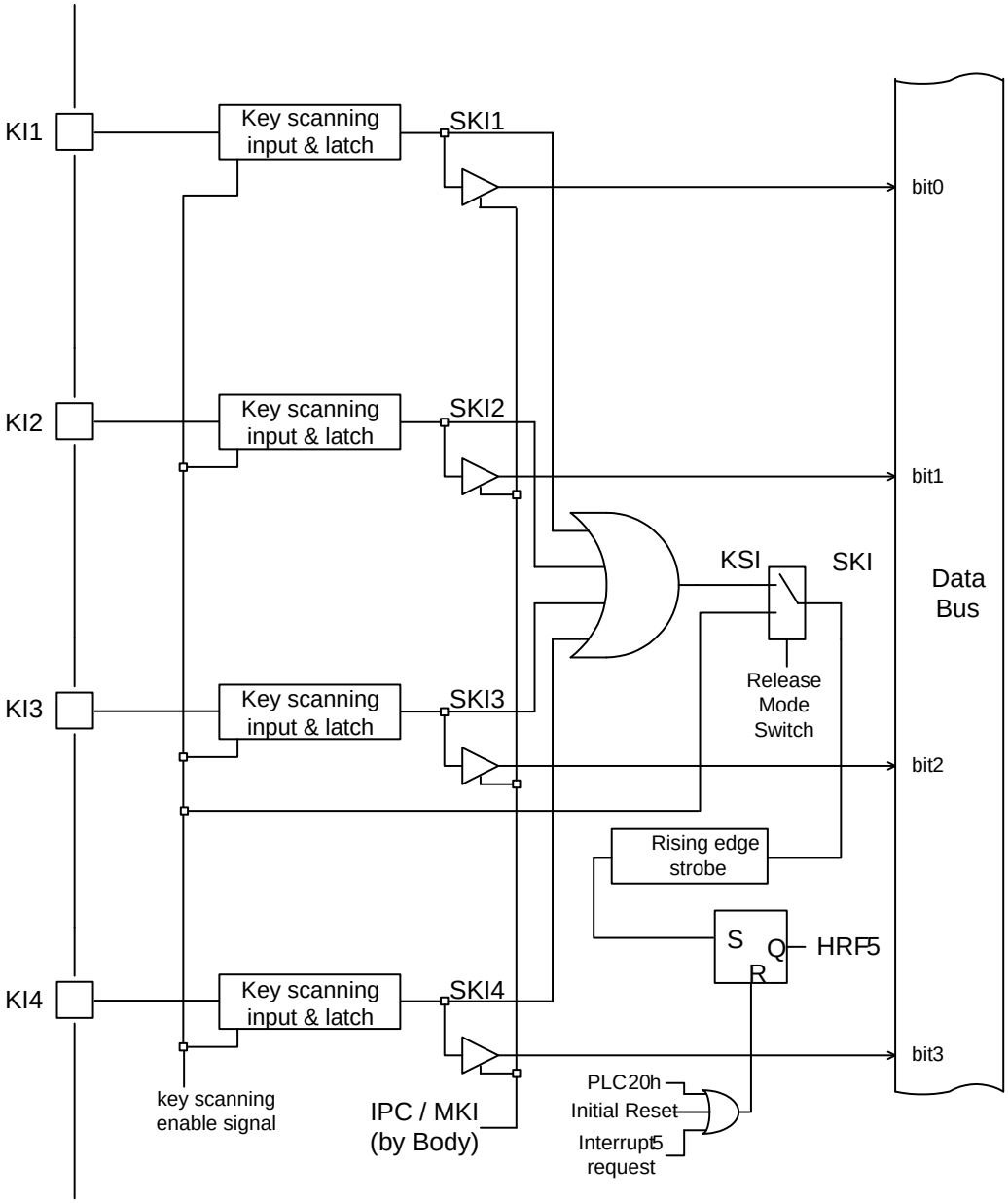
下圖說明 Key matrix 掃描功能典型應用線路：



Note: KO16 = SEG16，KO15 = SEG15，依此類推。

執行 SPKXH D (2 words instruction), SPKRH D (2 words instruction), SPKTH(#) D, SPKX X, SPKRx, and SPK(#) @HL 等指令可以設定不同的掃描模式以及選擇設定產生 halt release request flag 5 (HRF5)的方式。*詳細的用法請參閱指令說明書。*

下圖說明 key matrix 掃描功能中輸入部分的控制線路與相關 flag：



如 TM89 系列 MCU 都是以 PH6 的信號作為 key matrix 掃描功能的掃描頻率，使用者無法自行選擇。

另外如 TM89 系列 MCU 針對 KO1 ~ KO16 所輸出的掃瞄信號的時間長度也提供兩種 code option 的選擇，一種是 PH0 的週期長度，另一個是 PH1 的週期長度(請參閱 3-4-2 的 non-overlap 功能)。在選用較短的掃描時間長度時必須特別注意，掃描信號是否會因為 key matrix 掃描應用線路上的 RC 寄生負載而產生 delay，造成 MCU 無法正確的讀取掃描信號。

2-9-1. KI1 ~ KI4 輸入信號的讀取方式

Key matrix 掃描功能會在與掃瞄信號同步的時間抓取 KI1~KI4 腳位上的輸入信號，並將抓取到的輸入信號寫入 SKI1~SKI4 的暫存器中。此時程式可以執行 MKI 指令(若如 TM87 系列與 IOC port 共用則執行 IPC 指令，但對腳位 code option 為 KI，部份 MCU 仍須執行 SPC 指令將該腳位設定為 output 以免誤產生 IOC HALT release)來讀取 SKI1~SKI4 的內容值並儲存到 data memory，用以判斷是哪一個按鍵被壓下了。

2-9-2. KEY MATRIX 掃描功能的 HALT RELEASE REQUEST 以及中斷

當 SKI1~SKI4 的暫存器中至少有一個以上的高電位信號出現時，或是與 LCD 波形同步的掃瞄輸出信號產生時都可以將 halt release request flag 5 (HRF5)設定為 1，這兩種方式可以利用 key matrix 掃描功能的相關指令來選擇。執行 PLC 20h 指令可以將 HRF5 清除為 0。

當 halt release request flag 5 (HRF5)被設定為 1 時，如果 halt release enable flag 5 (HEF5)已經設定為 1，MCU 就會產生 HALT release 並且將 start condition flag 8 (SCF8)設定為 1。

相同情況下，如果 interrupt enable flag 5 (IEF5)已經設定為 1，MCU 就會接受 key matrix 掃描功能的中斷請求。

2-9-3. KI1 ~ KI4 作為 MCU 的 RESET 腳位

如果 MCU 已經利用 code option 選擇了這項 reset 功能後，只要 KI1~KI4 (已經設定作為 KI 的輸入腳位)同時偵測到掃描信號之後，MCU 就會進入 RESET 狀態並且開始 RESET 時間長度的計數。

2-9-4. 範例程式

範例 1：

```
        SPKXH       0               ; 設定有按鍵被壓下時才會設定 HRF5
        setdat      $FFFF           ; 在一個掃描週期內同時掃瞄所有的鍵盤
        PLC         $20             ; 清除 HRF5
        SHE         $20             ; 設定 HEF5
        HALT                        ; 等待 HRF5 使 MCU 產生 HALT release
        MCX         $10             ; 檢查 SCF8 (SKI)
        JB0         ski_release
        ……..
        ……
ski_release:
        MKI         $10             ; 讀取 SKI1~SKI4 的內容值
        JB0         ki1_release
        JB1         ki2_release
        JB2         ki3_release
        JB3         ki4_release
        .
        .
ki1_release:
        SPKXH       1               ; 檢查 KI1 腳位上的按鍵，並切換為掃描週期
                                    ; Release 模式。
        setdat      $0001
        PLC         $20             ; 清除 HRF5 以避免不正常的 HALT release
        CALL        wait_scan_again
                                    ; 等待下一個掃描信號，等待時間必須大於
                                    ; 掃描週期。
        MKI         $10             ; 讀取 SKI1 的內容值
        JB0         ki1_seg1
        ……..
        ……..
        SPKXH       1               ; 只送出 KO16 的掃描信號
        setdat      $8000
```

```
        PLC     $20             ; 清除 HRF5 以避免不正常的 HALT release
        CALL    wait_scan_again ; 等待下一個掃描信號，等待時間必須大於
                                ; 掃描週期。
        MKI     $10             ; 讀取 SKI1 的內容值。
        JB0     kil_seg16
        ……
        ……
wait_scan_again:
        HALT
        PLC    20h
        RTS
```

範例 2：(如 TM87 系列與 IOC port 共用)

```
        SPC     0fh             ; 設定 IOC1~4 為 output 模式，避免誤產生 IOC release
        SPKX    10h             ; 設定有按鍵被壓下時才會設定 HRF5，
                                ; 啟動在一個掃描週期內同時掃瞄所有的鍵盤
        PLC     20h             ; 清除 HRF5
        SHE     20h             ;啟動 HEF5.
        HALT                    ; 等待 HRF5 使 MCU 產生 HALT release。
        MCX     10h             ;檢查 SCF8 (SKI) 。
        JB0     ski_release
        ……
        ……
ski_release:
        IPC     10h             ; 讀取 SKI1~SKI4 的內容值
        JB0     ki1_release
        JB1     ki2_release
        JB2     ki3_release
        JB3     ki4_release
        .
        .
ki1_release:
        SPKX    40h             ; 檢查 KI1 腳位上的按鍵，並切換為掃描週期
                                ; Release 模式。
        PLC     20h             ; 清除HRF5以避免不正常的HALT release
        CALL    wait_scan_again; 等待下一個掃描信號，等待時間必須大於掃描週期。

        IPC     10h             ; 讀取 SKI1 的內容值
        JB0     ki1_seg1
        ……
        ……
        SPK     4fh             ; 只送出 KO16 的掃描信號
        PLC     20h             ; 清除 HRF5 以避免不正常的 HALT release
        CALL    wait_scan_again     ; 等待下一個掃描信號，等待時間必須大於掃描週期。
        IPC     10h             ; 讀取SKI1的內容值
        JB0     kil_seg16
        ……
        ……
```

```
wait_scan_again:
        HALT
        PLC    20h
        RTS
```

2-10. LOW VOLTAGE DETECT(LBD)

4-BIT 穩壓系列提供每階 50mV 的多階 LBD 功能。除了一般以指令執行 LBD 偵測功能之外，部分 MCU 更可提供省電型 auto-LBD 硬體自行偵測功能以達成即時反應需求。

2-10-1. LBD 指令偵測模式

依不同 MCU 規格提供高/低壓模式：2.75~2.00/2.10~1.35V 各 16 階 option 選擇(例如 TM87ML25)或是單一模式 2.90~1.35V 共 32 階 option 選擇(例如 TM87ML26)。但不論何種模式皆依例如 TM87ML25H/L 分別針對 2.4/1.35V option 由測試在室溫下調整至±25mV 誤差範圍內，故此階 option 會是最精準的電壓，而其他階 option 電壓則基本上因參考電壓受 VBAT 影響故差距越大的 option 電壓誤差也會越大。

程式可由 Rm 位址：B(32 階則須加上 Rm 位址：9 bit2 設定 LBD4)選擇所需要的 LBD 電壓規格。當 code option 選擇 LBD USE 時 MCU 會因啟動參考電壓約 0.5uA 耗電，可由執行 MMM 9,Ry 指令設定 bit0=1 啟動偵測，依 MCU 規格等待偵測完畢後再關閉 LBD，然後藉由執行 MMM Ry,9 指令讀取 bit0：LBF=0/1 判斷 VBAT >/< LBD 電壓 option。

2-10-2. 省電型 auto-LBD 即時偵測模式

例如 TM87ML26，除了原本 LBD 指令偵測功能之外，因穩壓有提供 LOW VOLTAGE RESET(LVR2)功能，故當應用不需要使用 LVR2 reset 功能時可由 code option 選擇將 LVR2 當作 auto-LBD 功能取代，由於耗電低故可以一直進行偵測。而一旦 code option 選擇 auto-LBD 功能便在 reset 後一直保持啟動偵測模式。所有 MCU 提供 Auto-LBD 皆是 2.90~1.35V 共 32 階 option 單一模式。

Auto-LBD 功能可以藉由執行 MMM Ry,9 讀取 bit3(LVR2F)=0/1 判斷 VBAT >/< auto-LBD 電壓 option，此外亦可藉由執行 MMM 9,Ry 指令設定 bit3(LVR2RLS)=1 & 啟動 INT release，當 LVR2F 由 0->1 或是 1->0 皆會產生 INT release 然後由 LVR2F 確認 release 由 auto-LBD 所產生(由 LVR2F 有無轉態排除是否因電源不穩定而誤產生 release) 或是 INT 本身(TM87ML26 提供 MDX 指令可讀取 INT 狀態判斷)。因此藉由 auto-LBD 硬體自行一直偵測，程式不但可以不用像 LBD 每隔一段時間就要執行指令偵測，更適用於更換電池&1.5/3.0V 雙電源等 VBAT 電壓變化較快須即時反應的應用。但是與 LBD 同樣因受穩壓工作電壓限制故仍須避免 VBAT 低於 1.2V 而有誤產生 LVR2F:1->0 release 的風險。

Auto-LBD 此時偵測電壓 option 會切換到與 LBD 一起由 Rm 控制，故與 LBD 同樣可以藉由相同 Rm 指令調整 auto-LBD 電壓 option(LBD4~0) ，起始值亦與 LBD 同樣為 2.4/1.35V option，但須注意差別的是 TM87ML26H 生產時是以 1.80Voption 調整 LVR2 電壓，故 auto-LBD 最精準電壓 option 是 1.80V option 而非 2.40Voption，也因如此對於應用會涵蓋 2.90V & 1.35V 兩個極端 option 時使用 auto-LBD 功能電壓誤差會較能兼顧(但仍需注意 VREG & VDL 受 VBAT 電壓大幅度變化影響幅度)。另外須注意由於以 Rm 指令切換 auto-LBD 偵測電壓時，因 auto-LBD 一直在進行偵測動作，故切換範圍若有需要切換 LBD4 時為了避免在中間過程執行指令調整 LBD4&3~0 順序錯誤而產生錯誤中斷依目前 LVR2F=0/1 依調高/低應該先切換 LBD4/3~0 之後再切換 LBD3~0/4 & 依調低/高應該先切換 LBD3~0/4 之後再切換 LBD4/3~0。

2-11. POWER MODE SWITCH(SWPWR)

如 TM87ML26 可提供 1.5/3.0V Power Mode 切換功能以針對 VBAT 目前電壓範圍調整 I/O Pull-low/high 阻值等亦受 VBAT 電壓影響特性。由於使用 SWPWR 功能時除了 LCD 電壓必須選擇穩壓以及若使用內建 VDL code option 建議選擇 "VL3 = 1.5 x VDL for 1/3Bias" 以避免 VDL 穩壓值變動之外，建議 code option 選擇 BAK=VREG 以避免在 BCF=0(3V Power Mode & BCF=0：BAK<VBAT)切換時造成內部電壓的變動 & 32KHz X' tal driver 變動風險，，故經由切換至 3.0V Power Mode 時限制為 BCF=0：BAK<VBAT 模式。除了以執行 SWPWR 指令切換之外，4-BIT 穩壓系列亦提供 code option 選擇省電型 auto-SWPWR 硬體自行偵測功能以達成即時反應需求。

2-11-1. SWPWR 指令偵測模式

如 TM87ML26 可藉由 SWPWR 指令設定 X1,0=10/11 直接切換系統為 3.0/1.5V Power Mode，設定 X1=0 則系統將回到原來模式(尤其是 3.0V Power Mode 選擇 code option 為 BCF=0 ：BAK=VBAT 模式)。

一般應用方式為根據 LBD 偵測結果執行 SWPWR 指令切換，當 code option 選擇 BAK=VREG 並且維持在 BCF=0 模式 & "VL3 = 1.5 x VDL for 1/3Bias" 則系統差異只在 I/O Pull-Low/High 阻值變化以避免因 VBAT 電壓降至過低而造成阻值過大產生干擾等問題或是因 VBAT 電壓升至過高而造成阻值過低產生耗電過大等問題。

當 3V Power Mode 應用只是單純希望降低 Pull-Low/High 阻值以進一步降低干擾或是放/充電速度則可直接藉由執行 SWPWR 指令設定 X1,0=11 完成。因為整個 Power Mode 切換故同樣建議使用上述穩壓建議以避免困擾，但若因故無法使用 BAK=VREG 穩壓則須注意避免使用 32KHz X' al 造成因為在 VBAT 電壓仍在 3V 高壓時因 1.5V Power Mode driver 過大造成停振問題。

2-11-2. 省電型 auto-SWPWR 即時偵測模式

如 TM87ML26 穩壓有提供 LOW VOLTAGE RESET(LVR2)功能，當應用不需要使用 LVR2 reset 功能時可由 code option 選擇將 LVR2 當作 auto-SWPWR 功能取代，由於耗電低故可以一直進行偵測。而一旦 code option 選擇 auto-SWPWR 功能便在 reset 後一直保持啟動偵測模式。Auto-SWPWR 提供 2.90~1.35V 共 32 階 option 單一模式，與 LVR2 一樣可由 code option ："LVR2 RESET/RELEASE VOLTAGE" 分開設定 3.0->1.5V/1.5->3.0V Power Mode。

Auto-SWPWR 功能可以藉由執行 MMW Ry,9 讀取 bit3(LVR2F)=0/1 判斷 VBAT >/< auto-SWPWR 電壓 option ，此外亦可藉由執行 MWM 9,Ry 指令設定 bit3(LVR2RLS)=1 & 啟動 INT release，當 LVR2F 由 0->1 或是 1->0 皆會產生 INT release 然後由 LVR2F 確認 release 由 auto-SWPWR 所產生(由 LVR2F 有無轉態排除是否因電源不穩定而誤產生 release) 或是 INT 本身(TM87ML26 提供 MDX 指令可讀取 INT 狀態判斷)。因此藉由 auto-SWPWR 硬體自行一直偵測，更適用於 1.5/3.0V 雙電源等 VBAT 電壓變化較快須即時反應的應用。但是與 auto-LBD 同樣因受穩壓工作電壓限制故仍須避免 VBAT 低於 1.2V 而有誤產生 LVR2F:1->0 release 的風險。

2-12. In Application Programming (IAP) for Table ROM

TM87ML28 提供 Table TOM word 燒錄指令功能以節省外接 EEPROM 等成本。執行程序如下：

1. 直接外灌 VBAT&BAK=2.2V 燒錄電壓，或是設定 BAK=VREG：2.20V option 燒錄電壓(建議 BAK 外接 1uF 以上電容以避免因燒錄電流使 BAK 電壓掉落過大)。
2. 執行 SHLX 或是 VL/H/U/V 指令設定 RAM 欲燒錄至 Table ROM 的四個位址。
3. 分別執行 MRI $0/1/2/3 指令將 HL 所指定 RAM 位址的 4bits data 傳送至 16bits MRI 暫存器。

4. 執行 SHLX or MVL/H/U/V 指令設定 Table ROM word 燒錄中的 low byte 位址。
5. 暫時先關閉所有會產生 HALT release Flag 以避免燒錄尚未完成就被迫離開。
6. 執行 PTR 指令進行對 MRI 所 latch 16bits data 燒錄至 HL 所設定 Table ROM word 位址，燒錄完成後系統 將自動產生 HALT release。
7. 執行 MSD 只指令讀取 bit3(PGMF)， 若 PGMF = 0 則表示燒錄正常, 若 PGMF=1 則表示燒錄失敗，需再重新執行 PTR 指令燒錄。

2-13. Serial IO(SIO)

TM87ML28 提供 I2C(Master Only)/UART/SPI 三種序列資料傳輸功能(SIO)，由於這三種模式同時只能選擇其中一種模式執行，因此相關設定亦共用同樣的 Rm 位址，說明基本執行順序如下：

1. UART &I2C 可依應用需求選擇與 IOA1~4’ code option 的獨立腳位或是選擇與 SPI 共用 SIOX,Y 腳位，也可選擇一個腳位與 SPI 共用 SIOX,Y 腳位(但須注意因 SIOX/SIOY 有三種 SIO 模式切換的考量，故無論 DSIOPHL=1 或是 DC 模式 I2CEN/UARTEN/SPIEN=0 時 pull-high 仍會啟動，因此建議選擇共用 SIOX/SIOY 時設定為 Open-Drain 或是維持在 SIO 啟動狀態)。
2. TM87ML28 內建 SIO 相關腳位 pull-high/low 電阻，可由 code option 選擇 SIO 相關 pull-high/low 阻值，建議選擇 VBAT=3V 約 40KΩ 阻值以符合一般傳輸速度，但若傳輸速度夠慢或是預計設定所有相關腳位皆是 DC 模式或是預計改用外接電阻則可選擇一般 3V Power Mode 阻值以節省耗電。
3. 執行 MWM $c,Ry 指令選擇進入其中一個模式，UART/SPI 更同時設定部分模式選擇。
4. 執行 MWM $4,Ry 指令可設定 bit3=1 & 由 bit1,0 選擇 SIO 的 Clock Source，TM87ML28 提供 SXCLK,FREQ,Timer2,Timer3 四種 Clock Source 選擇：
   a. 若 Clock Source 欲選擇 SXCLK 則須先執行 MWM $4,Ry 指令設定 bit3=0 & 由 bit2~0 選擇 XCLK 的除頻值為 SXCLK 的輸出頻率後再執行 MWM $4,Ry 指令設定 bit3=1 選擇 clock source=SXCLK。
   b. 當設定 FREQ 為 SIO Clock Source 時須先自行執行 SCC 指令設定 Cfq=XCLK 與指令頻率同步，並且須自行注意在 SIO 啟動期間勿使用 FREQ 在其他功能以免影響 SIO 正常頻率動作。
   c. 當設定 Timer2/3 為 SIO Clock Source 時則須先自行設定 Ctm2/3=XCLK 與頻率指令同步，此外一旦選擇 Ctm2/3=XCLK 之後 Timer2/3 啟動與關閉將由 SIO 完全控制&硬體會自動固定在 RL2/3=1 & SRP2/3=1 & HRF4/7=0，輸出頻率=F(Ctm2/3)÷(Timer2/3 set value + 1)。
   d. TM87ML28 提供 ST3OV 指令可由 Timer3 在每次 underflow 時選擇對 FREQ/Timer2 mask 一個 Cfq/Ctm2 週期方式調整至較精準的頻率值，尤其是 UART 並無同步時鐘故需要能提供足夠精準的頻率值，欲使用 Int-RC 亦可先由設定 Crfc=FREQ(Cfq=XCLK)或是直接以執行 SXCLK 指令設定 Crfc=XCLK 由 TMR2(Ctm2=X’tal 穩定頻率來源) Control RFC 或是由 CX/2 Control 模式外灌精準穩定時間來源方式計算追蹤 Int-RC 頻率後再藉此方式針對每顆 IC 調整至同樣頻率以達到節省 X’tal 元件成本(*請參閱第 1-10-1-6 小節詳細說明*)。
5. 執行 MWM $8,Ry 指令可由 bit3,2 設定 SIOTYP2,1(起始值=00) 以提高傳輸速度或是節省耗電考量選擇 SIO 部分或是全部 I/O pins 為 DC 或是 Open-Drain 模式 & 若所有相關腳位選擇由外部提供 Pull-High/Low 電阻則可由 bit1 設定 DSIOPHL=1 (起始值=0)關掉相關 pins 內建的 Pull-High/Low 電阻 (DC 模式會自動關閉該腳位的 Pull-High/Low 電阻)。以上模式設定後除非產生系統 reset 才會回到起始值，故中間切換至其他 SIO 模式回來後並不需要再進行重設。

6. 執行 MWM $6,Ry 可設定各項模式選擇(依各項性質&實際應用決定在啟動 SIO 前後設定)，執行 MMW Ry,$6 可讀取相關 Flag。
7. 執行 MWM $5,Ry 可啟動 SIO 或者設定部分模式選擇。
8. 執行 MWM $2,Ry 可設定輸出資料的 low-nibble，然後執行 MWM $3,Ry 則可設定輸出資料的 high-nibble & 開始進行傳輸。
9. 執行 MWM $7,Ry 可設定 SSRE3~1&INTFEN 選擇啟動 SIO & INT release flag，由於 SIO 與 INT 共用 HRF2 flag 故需執行 MMW Ry,$7 讀取 SSCF3~1&INTF 以判斷 release 來源。
10. 執行 MMW Ry,$2 指令可讀取輸入資料的 low-nibble， 然後執行 MMW Ry,$3 指令可讀取輸入資料的 high-nibble & 清除相關接收的 Flag。

2-13-1. I2C(Master Only)

TM87ML28 只提供 I2C Master Mode 功能，以下說明 I2C 執行流程&相關細節：

1. 執行 MWM $C,Ry 指令&設定(Ry)=0001b 進入 I2C 模式。
2. 執行 MWM $4,Ry 指令選擇 I2C SCL 頻率，與 UART&SPI 差異如下：
   a. SCR3=0 ：更換 SCR2~-0=000~111 設定為 SXCLK=XCLK/4/8/16/32/64/128/256/2。
   b. SCR3=10：SCR1,0=00/01/10/11 選擇 SXCLK/FREQ/Timer2/Timer3，而除了 SXCLK 直接更換頻率之外，其餘輸出頻率皆除以 2 才是 SCL 頻率。

   由於 I2C 必須確保 SDA pin 電位在 SCL pin 為電位'1'時是穩定的，故頻率會除以 2 以使 SDA 與 SCL 轉態點分開，因此 SCL pin H:L 寬度皆為 1:1 比例，故選擇 FREQ/Timer 當 I2C Clock Source 並不會增加 Pull-High 穩定時間，FREQ/Timer 使用時機應只在避開因與系統時鐘頻率為 SCL 的 2 的 N 次方等關係而產生干擾時。
3. 執行 MWM $8,Ry 指令由 bit3,2 設定 SIOTYP2,1 並維持 bit2(DSIOPHL)=0 啟動內建 pull-high 電阻時各模式特性如下表所示：

SIOTYP2,1		00(initial)	01	10	11
I2C	SDA(I/O) ,SCL(O)	NOD(I/O) , NOD(O) *I2C-1 (standard & for Multi-Master)	NOD(I/O) ,NOD(O) *I2C-2 (for Multi-Master)	DC(O)/NOD(I) ,DC(O) (For Single-Master)	DC(I/O) ,DC(O) (For Single-Master)

NOD : N-type Normal Open Drain.
DC : CMOS output.
*I2C-1 : SDA(I) disable Pull-High when SCL=High to reduce current.
*I2C-2 : SDA(I) always enable Pull-High to avoid ACKI:Low->High at SCL=HIGH.

   a. SIOTYP2,1=00 模式：為起始設定值，SDA&SCL 皆為標準 Open Drain 模式故適用於 Multi-Master 應用。但為節省耗電考量，當 SDA 為輸出模式時若輸出電位為'0'時會關閉 pull-high 電阻，當 SDA 為輸入模式時若輸入信號為電位'0' 則在 SCL 在電位'1'時會關閉 pull-high 電阻。
   b. SIOTYP2,1=01 模式：與 SIOTYP=00 模式同樣 SDA&SCL 皆為標準 Open Drain 模式故適用於 Multi-Master 應用。但考量若 ACK 輸入信號在 SCL 在電位'1'時會有電位'0'至'1'應用需求，故保留持續啟動 pull-high 電阻的標準模式。
   c. SIOTYP2,1=10 模式：SCL 為 DC 輸出模式 可去除 SCL pull-high 耗電故適用於 Single-Master 應用。SDA 在輸出模式時亦為 DC 模式，但在輸入模式時仍為與 SIOTYP2,1=00 模式一樣提供內建 pull-high 控制模式，故適用於對外接 Multi-Slave 應用(所有 Slave 皆須為 Open Drain 輸出模式，故傳輸速度仍會受 pull-high 限制)。

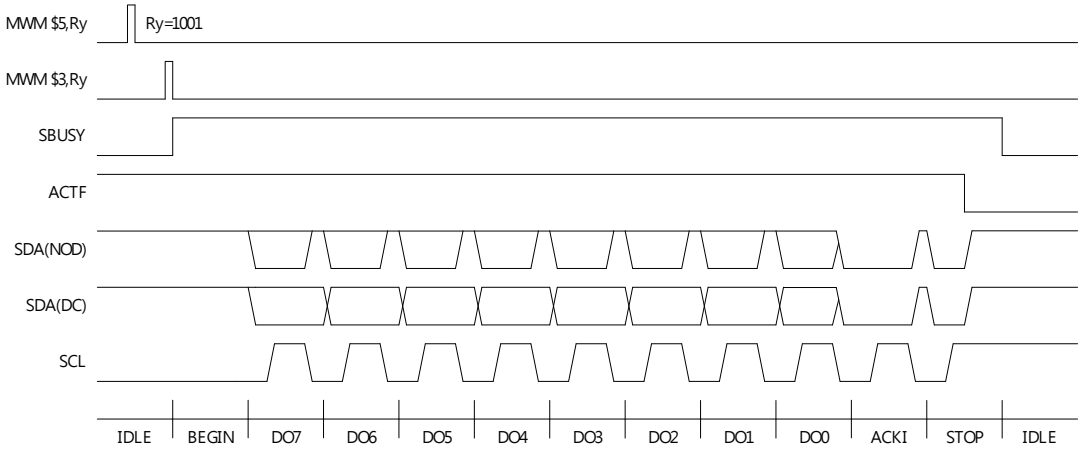
d. SIOTYP2,1=11 模式：SCL 為 DC 輸出模式 & SDA 無論輸出或輸入皆是 DC 模式(只有在 ACKI 位元仍會啟動 pull-high)，故不僅可節省耗電也可提升傳輸速度，但僅適用於外接 Single-SIve 應用。雖然在送出 STOP bit 完成後到執行 MWM $5,Ry 指令設定 START=1 之前亦會停止輸出&啟動 pull-high，但在 BUSY:1->0 之前仍是維持輸出模式，故除非確定對方 IRQ 會在 SDA:0->1 時延遲至少 1/2 cycle of SCL 之後才送出，否則不建議使用在 IRQ 應用。

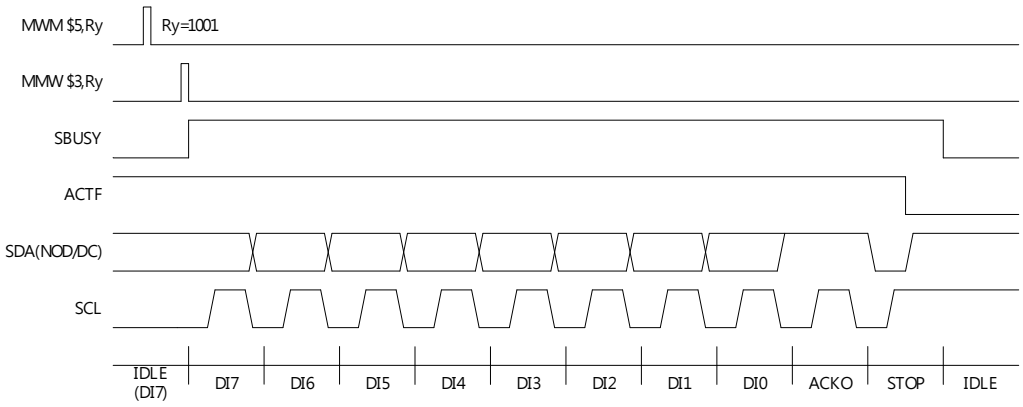
4. 執行 MWM $5,Ry 指令可由 bit3 設定 I2CEN = 1/0 啟動/關閉 I2C 並由 bit2~0 依 STRSTP,START,STOP 組合設定不同的傳輸模式如下表所示：

STRSTP	START	STOP	Description
X	0	0	Nothing
0	0	1	Sent STOP bit after ACKI/ACKO bit by WDATH/RDATH for write/read mode.
0	1	0	Sent START bit before MSB of data by WDATH
0	1	1	Sent START bit before MSB of data by WDATH & sent STOP bit after ACKI/ACKO bit for LSB of data =0/1.
1	0	1	Direct sent STOP bit
1	1	0	Direct sent START bit & force to Write Mode(don't care LSB=1) after write SDAT7~4.
1	1	1	Direct sent START & STOP bits

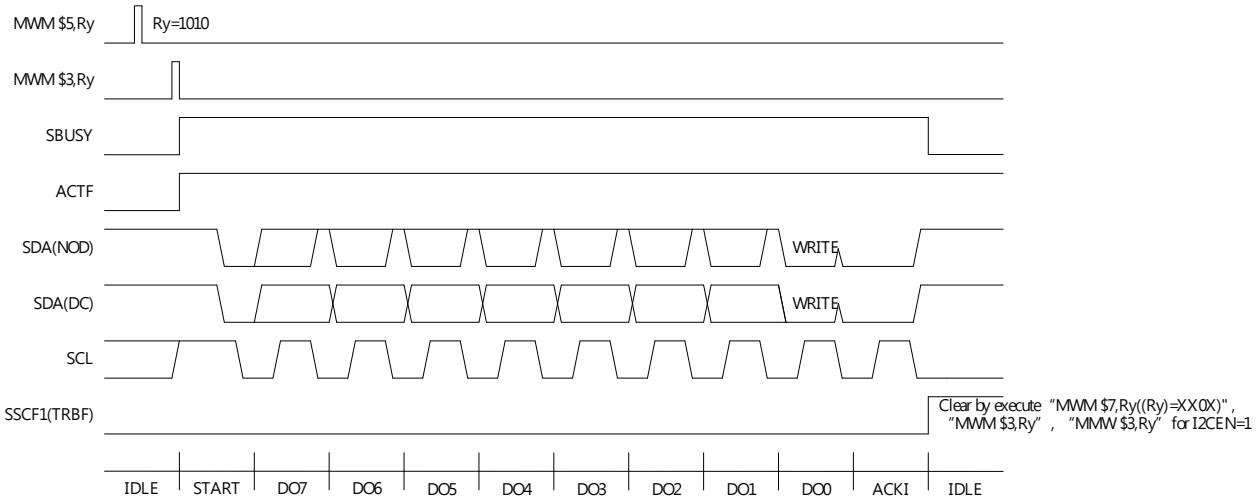
a. START&STOP=00：不會直接執行任何輸出，也不會間接跟著執行 MWM $3,Ry 或是 MMWRy,$3 組合輸出。

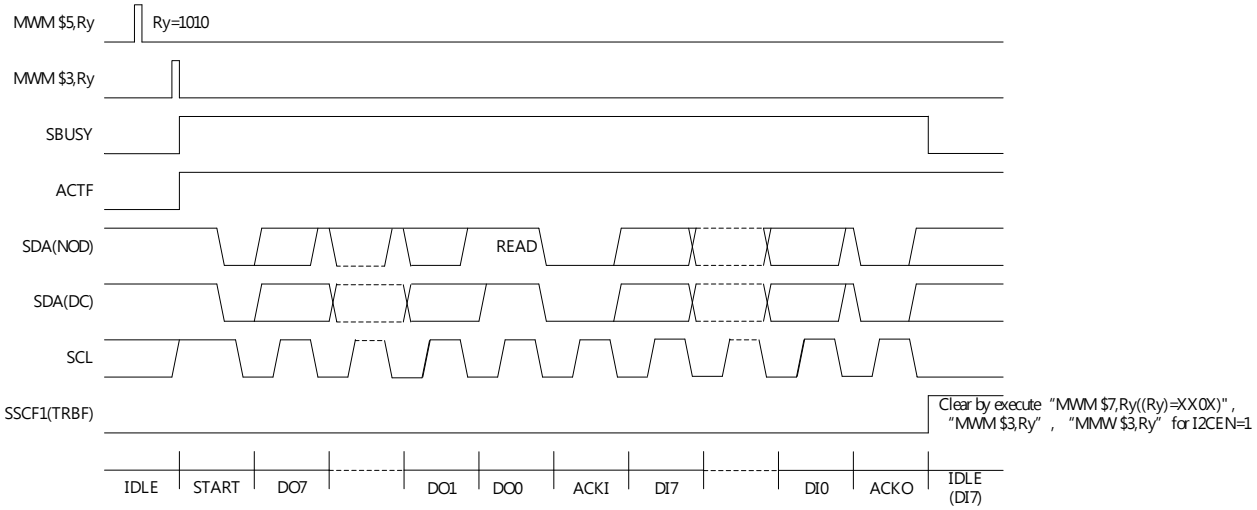
b. STRSTP,START&STOP=001：執行後不會有立即動作，在傳送/接收模式會一直等到執行 MWM $3,Ry / MMW Ry,$3 指令並且當資料傳輸完成後才會傳送 STOP bit 離開，在接收模式時當接收資料完成後會改傳送 ACK 電位'1' 使對方(SLAVE)之後停止輸出模式以便之後能傳送 STOP bit(ACKOSTP=0)。傳送/接收模式時序分別如下列圖所示：



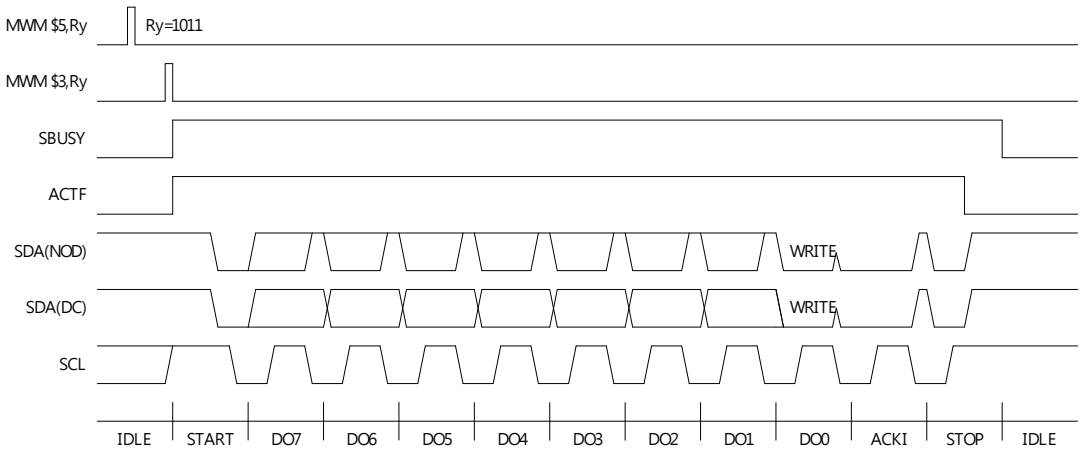


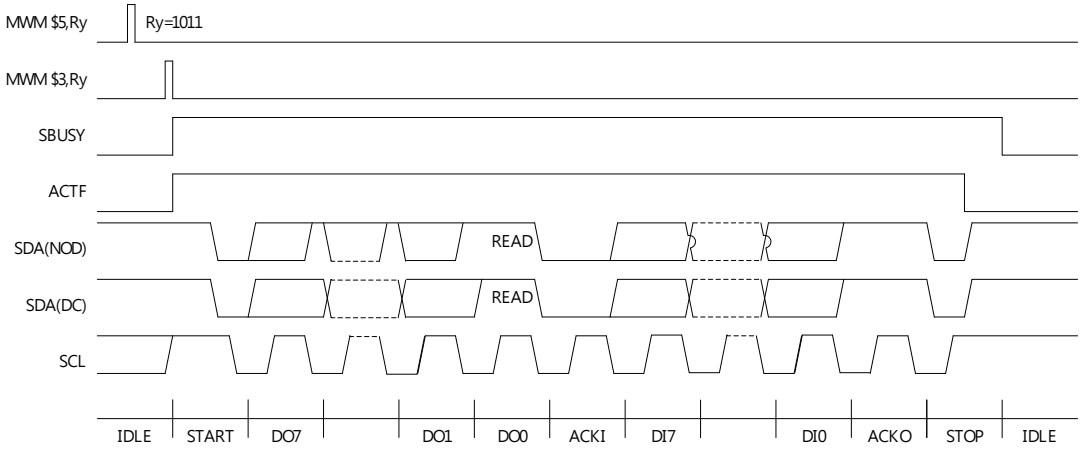
c. STRSTP,START&STOP=010：執行後不會有立即動作，一直等到後面執行 MWM $3,Ry 指令時才會先傳送 START bit 然後才開始傳送一個 Byte 資料，若 LSB=0 則停下來等待下一個動作，但是若 LSB=1 則進入接收模式自動接收一個 Byte 資料後再停下來等待下一個動作。進入傳送/接收模式時序分別如下列圖所示：



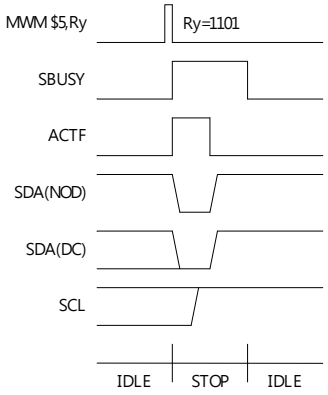


d. STRSTP,START&STOP=011：執行後不會有立即動作，一直等到後面執行 MWM $3,Ry 指令時才會先送 START bit 然後開始傳送資料，傳送完成後若 LSB=0 則直接傳送 STOP bit 離開，若 LSB=1 則進入接收模式，當接受一個 Byte 資料完成後再直接傳送 STOP bit 離開。進入傳送/接收模式時序分別如下列圖所示：

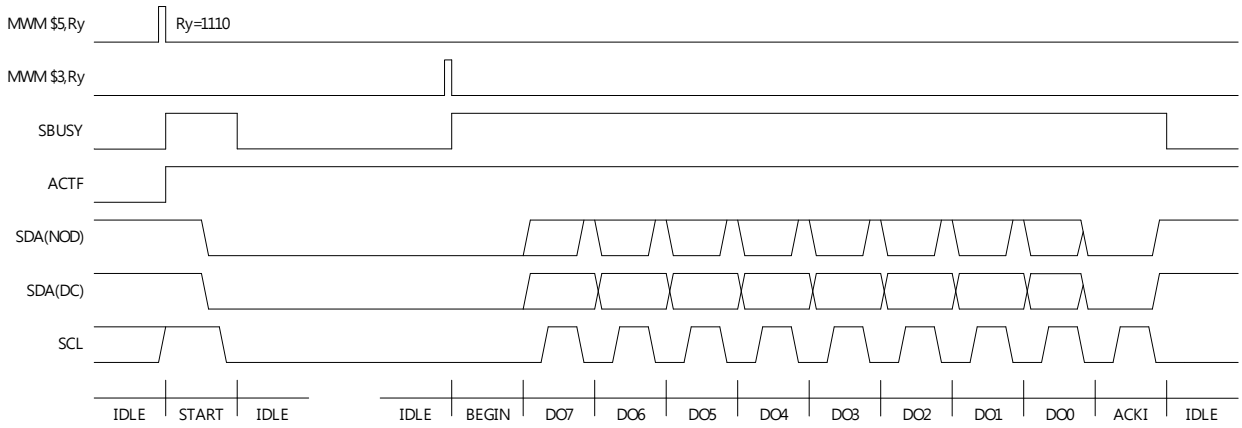




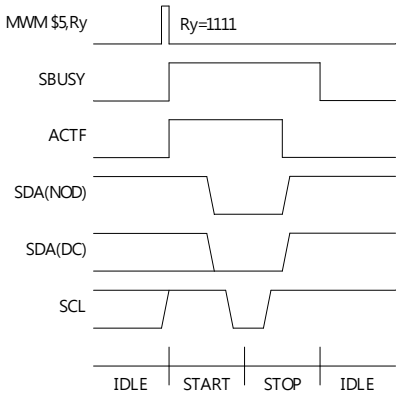
e. STRSTP,START&STOP=101：執行後會直接傳送一個 STOP bit 離開。時序如下圖所示：



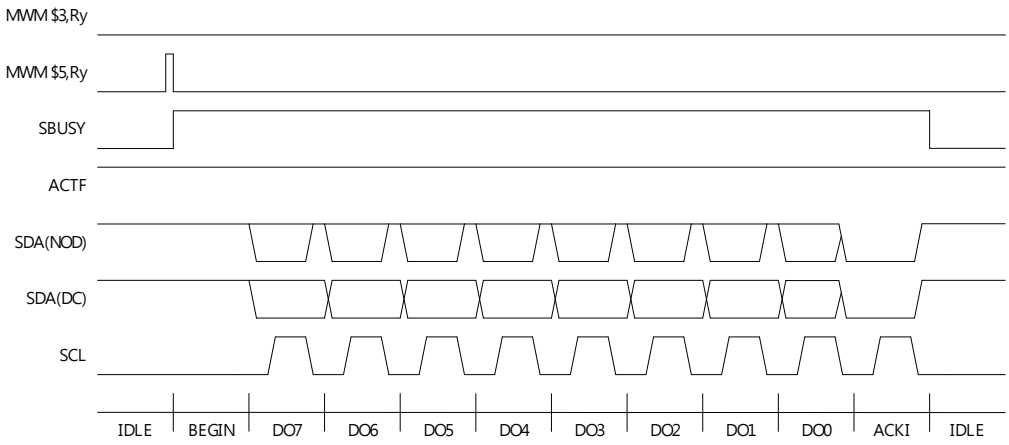
f. STRSTP,START&STOP=110：執行後會直接傳送一個 START bit，但是後面執行 MWM $3,Ry 指令後不論 LSB=0/1 皆會維持在傳送模式以提供搭配 Software Reset 等功能組合需求。時序如下圖所示：

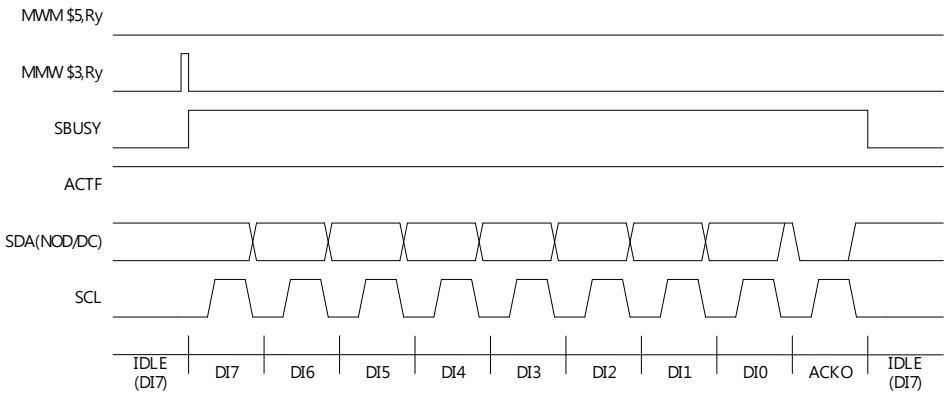


g. STRSTP,START&STOP=111：執行後會直接傳送一個 START bit 並且緊接著傳送一個 STOP bit 以提供搭配 Software Reset 等功能組合需求。時序如下圖所示：



h. 以上模式設定後最多只會對後面一個執行 MWM $3,Ry 或是 MMW Ry,$3 指令動作產生影響，之後再執行 MWM $3,Ry 或是 MMW Ry,$3 指令只會單純進行資料傳輸動作，故不需再執行 MWM $5,Ry 重設 STRSTP,START,STOP 模式。傳送/接收模式時序分別如下列圖所示：

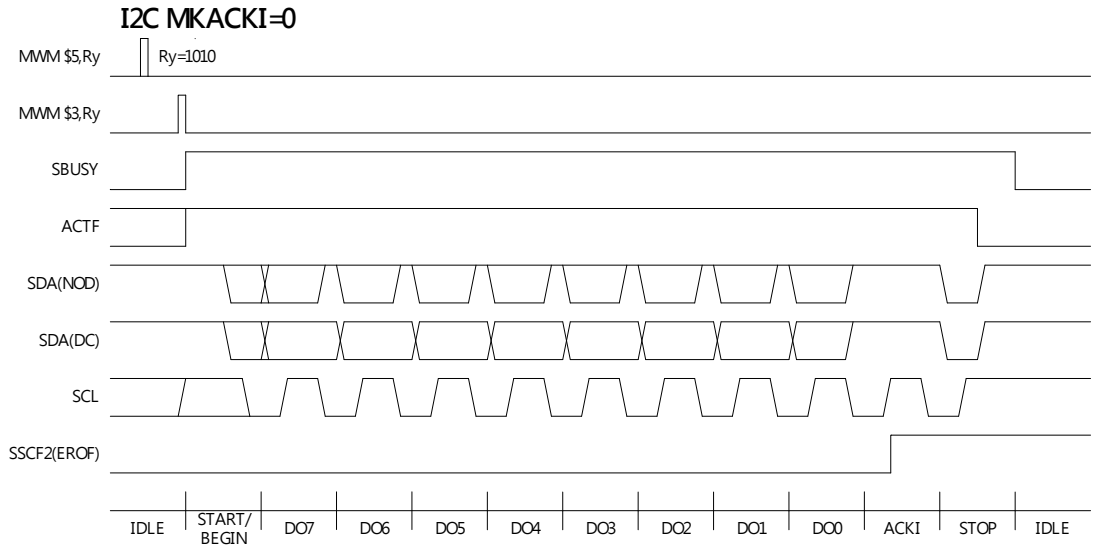


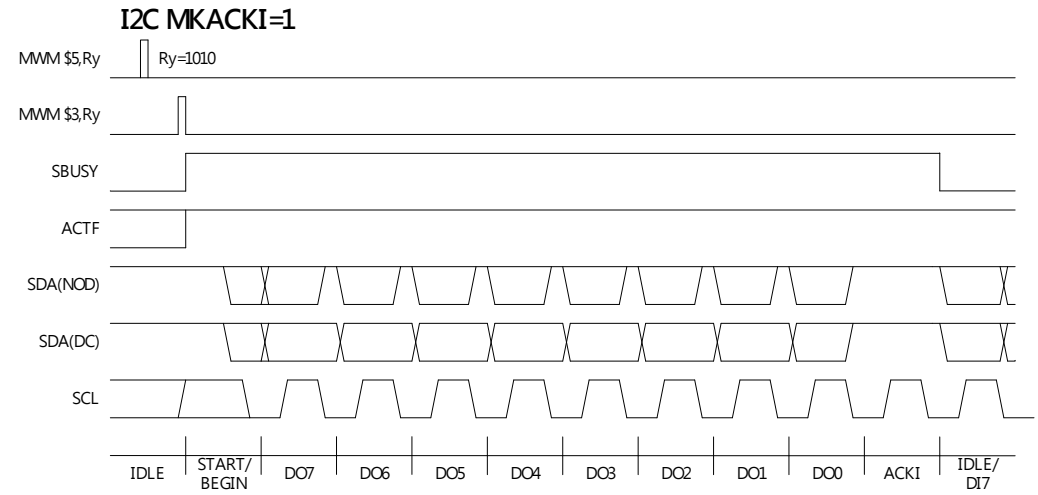


i. 以上需特別注意在尚未傳送 STOP bit 之前 SCL 在 IDLE 模式會停留在電位'0'，故若 SCL 為 Open Drain 輸出模式但選擇外接 pull-high 電阻(DSIOPHL=1)則須注意停留時間對耗電的影響。

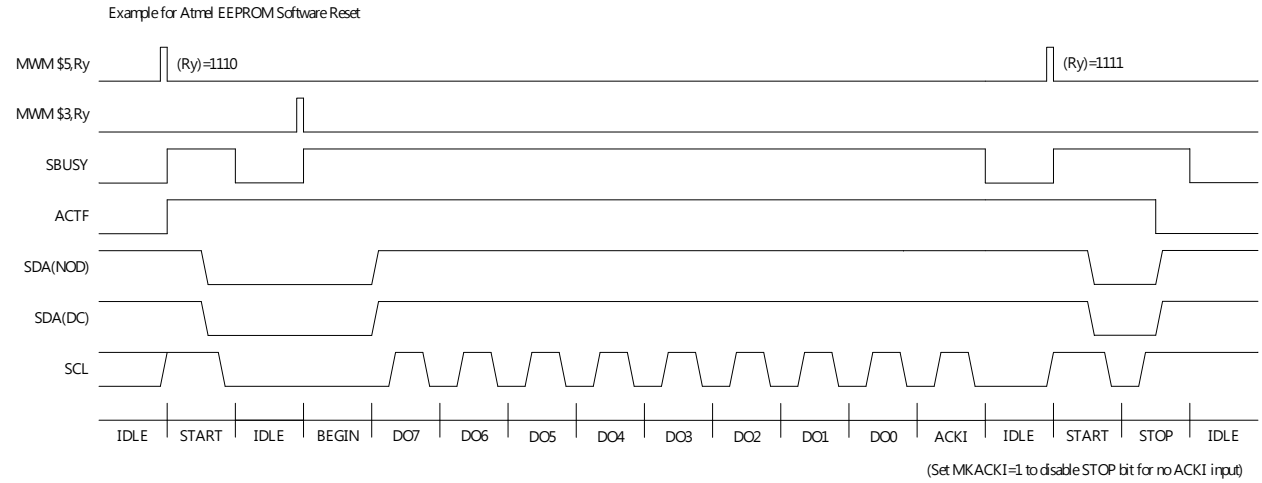
5. 執行 MWM $6,Ry 指令可由 bit3~0 分別設定 SDOCHK,ACKOSTP,WTACKI0,MKACKI 功能。分別說明如下：

a. MKACKI：當 MKACKI=0(起始值)時，若 ACK 輸入信號為電位'1'則會緊接著送 STOP bit 離開並且當有設定 EROFEN=1 時會使 EROF=1。當設定 MKACKI=1 時則會忽略 ACK 輸入信號以提供搭配 Software Reset 等功能組合需求，但若 ACKI 為電位'1'仍會使 ACKF=>1 以供檢查需要。MKACKI=0/1 時序分別如下列圖所示：

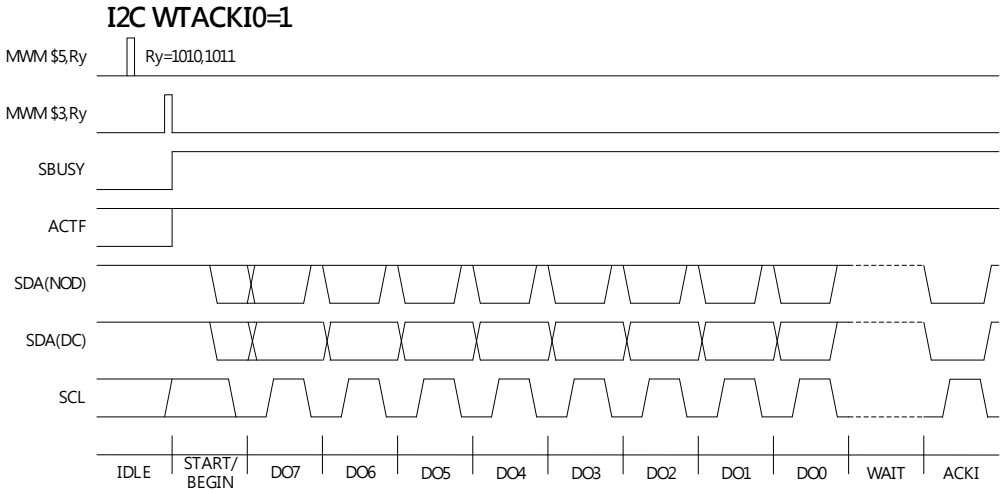




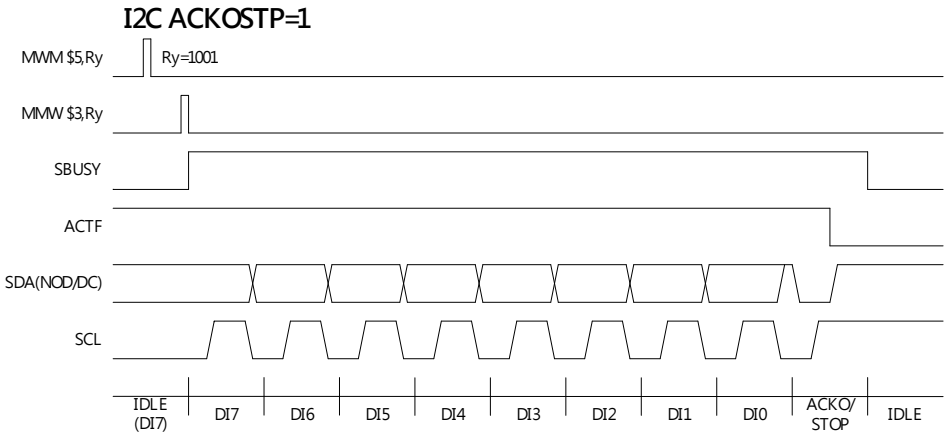
如下圖範例所示，搭配 STRSTP,START,STO=011 & 111 完成 Atmel EEPROM Software Reset 的要求：



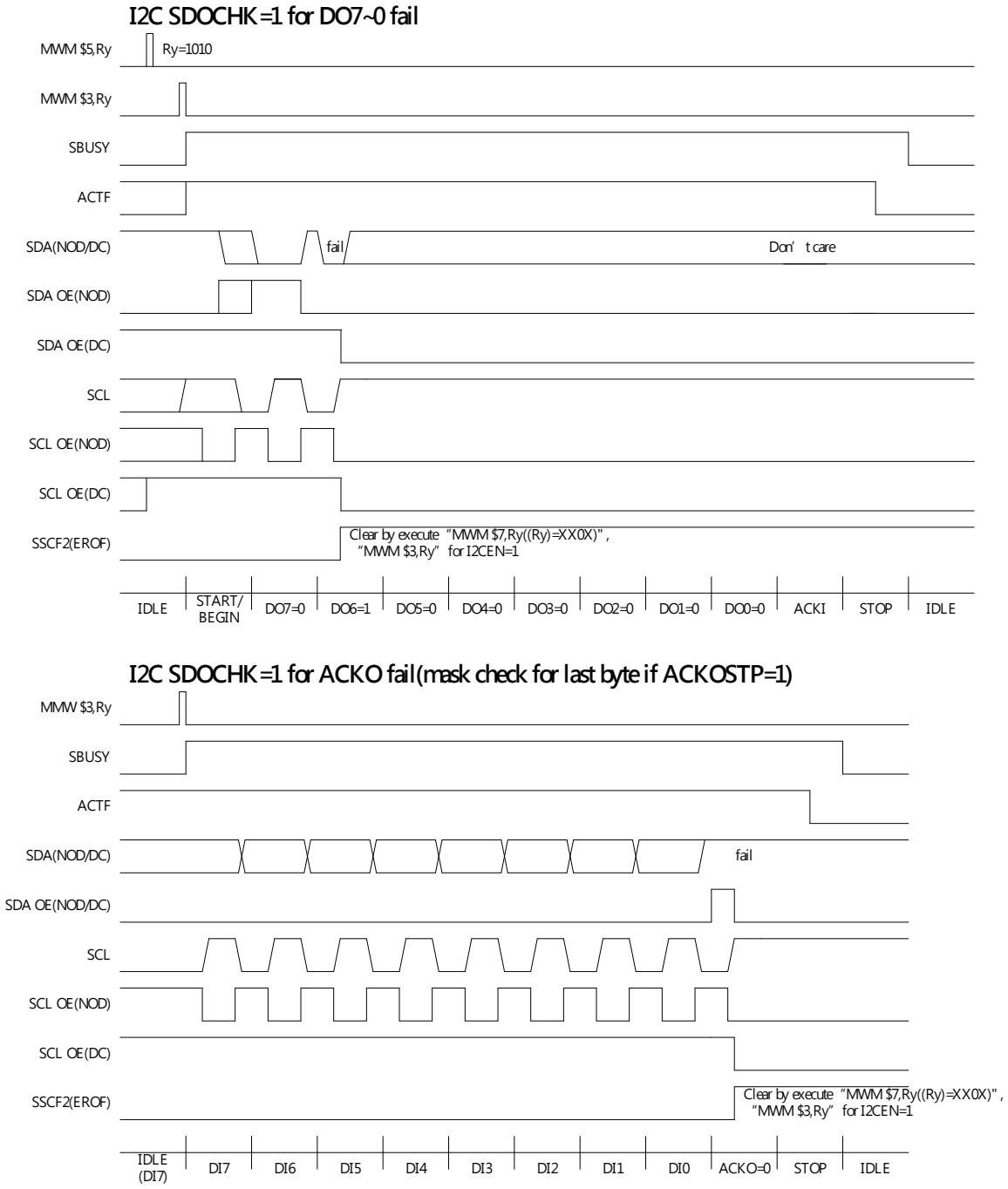
b. WTACKI0：起始值=0，當設定 WTACKI0=1 時在依各位元組資料傳送完成後會進入 WAIT 模式等待 SDA 轉為電位'0'之後才會對 ACKI bit 傳送 SCL 以針對若對方(SLAVE)架構是在接收資料後執行動作結束後才送電位'0'回覆 ACK 方式。時序如下圖所示：



c. ACKOSTP：當 ACKOSTP=0(起始值)時系統會以標準 I2C 方式在傳送 ACKO 電位'1'後再傳送 STOP bit，而當設定 ACKOSTP=1 之後則會以 STOP bit 取代 ACKO bit 以縮短一 個 SCL 週期時間或是避免對方(SLAVE)沒有因 ACKO=1 而會停止轉換輸出模式的機制。時序如下圖所示：



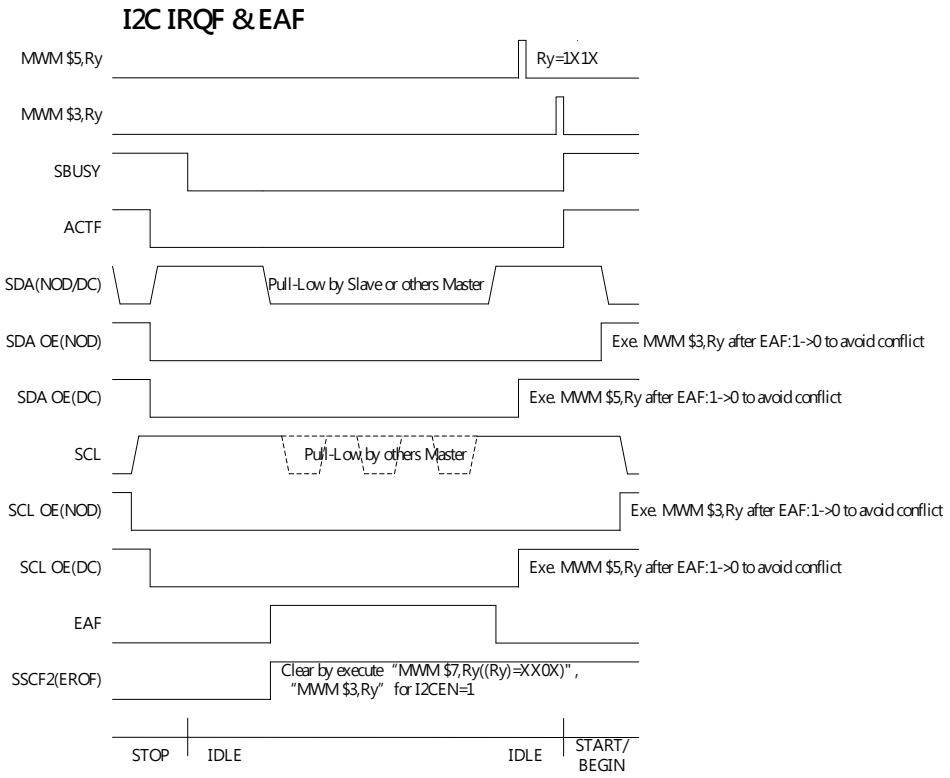
d. SDOCHK：起始值=0，當設定 SDOCHK=1 時系統會檢查所有輸出訊號與 SDA 腳位電位是否一致(包含位元組資料&ACKO)，一旦不一致發生後 SDA&SCL 會立即關閉輸出模式，雖然仍會跑完整個傳輸時序才會設定 SBUSY=>0 產生 TRBF(若 TRBFEN=1)但會忽略 ACKI，若有設定 EROFEN=1 時若是則會產生 EROF=1 旗號。時序如下圖所示：



6. 執行 MMW Ry,$6 指令可由 bit3~0 分別讀取 ACTF,ACKF,EAF,SBUSY 旗號。

   a. SBUSY：起始值=0，當執行每次動作時皆會設定 SBUSY=1 直到動作結束後才會回到 SBUSY=0 狀態。

   b. EAF：起始值=0，當 ACTF=0 時若 SDA:1=>0 @SCL=1 則會設定 EAF=1 直到 SDA:0=>1 @SCL=1，此時若有設定 IRQFEN=1 則會產生 IRQF=1 旗號，主要功用在於對方(Slave)執行內部動作完成後會發出 IRQ 信號通知 Master 讀取資料應用或是 Multi-Master 應用時當其他 Master 執行動作時可確認 EAF：1->0 後再開始執行動作以避免衝突。時序如下圖所示：



如圖所示，當 SDA 為 DC output 模式時，執行 MWM $5 Ry 指令 設定 START=1 時便會啟動 SDA 為輸出模式，故須在確認 EAF=>0 後方可設定 START=1。

c. ACKF：起始值=0，在每次傳送一個位元組資料後會由對方(Slave)回覆一個 ACKI=0 已表示對方有正常完成接收動作，可由讀取 ACKF 確認 ACKI 電位值，若 ACKF=1 則表示對方並未回覆 ACK 信號，此時若有設定 EROFEN=1& MKACKI=0 則會產生 EROF=1 旗號。

d. ACTF：起始值=0，當內部電路從 START bit 開始會設定 ACTF=1 直到 SDA:0=>1@SCL=1 才會回到 ACTF=0 以提供確認整個 I2C 動作是否完成。

7. 執行 MWM $7,Ry 指令可由 bit3~0 分別設定 IRQFEN,EROFEN,TRBFEN,INTFEN=1(起始值=0)以產生 IRQF,EROF,TRBF,INTF 旗號(起始值=0)由執行 MMW Ry,$7 指令讀取，若設定 IEF2=1 或是 HEF2=1 當 IRQF,EROF,TRBF,INTF 其中一個旗號=1 則可啟動 HRF2 Release。

a. INTFEN&INTF：起始值=0，由於 SIO 與 INT 共用 HRF2 Release，故欲啟動 INT 本身 release 必須增加設定 INTFEN=1 方可在 INT pin 符合轉態條件時產生 INTF=1，設定 INTFEN=0 可清除 INTF 旗號。

b. TRBFEN&TRBF：起始值=0，設定 TRBFEN=1 可在每次 SBUSY:1=>0 時產生 TRBF=1，執行 MMW $3,Ry 指令或者執行 MWM Ry,$3 指令或者設定 I2CEN=0 或者設定 TRBFEN=0 皆可清除 TRBF 旗號。

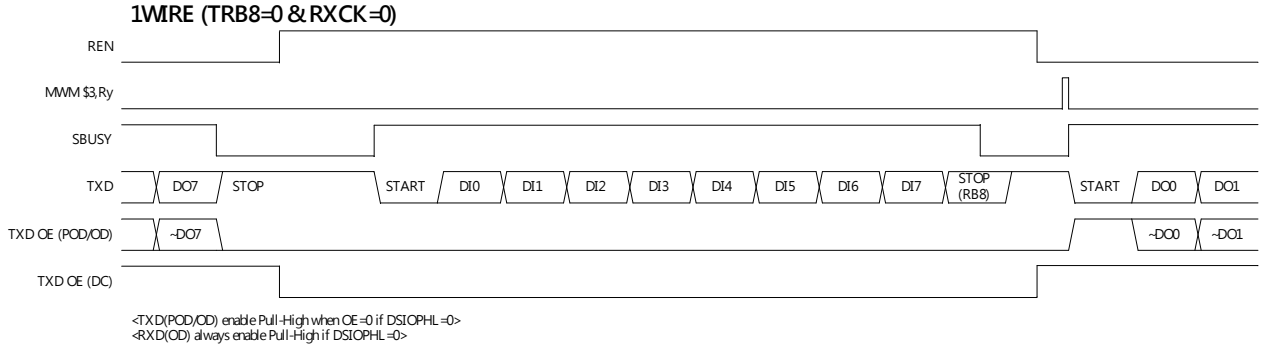
c. EROFEN&EROF：起始值=0，設定 EROFEN=1 可在設定 SDOCHK=1 時當發生 SDA 電位與傳送資料& ACKO=1(ACKOSTP=0)不一致或是 ACKF=1 時產生 EROF=1 ，執行 MMW $3,Ry 指令或者設定 I2CEN=0 或者設定 EROFEN=0 皆可清除 EROF 旗號。

d. IRQFEN&IRQF：起始值=0，設定 IRQFEN=1 可在 ACTF=0 時當發生 SDA:1=>0 @ SCL=1 產生 IRQF=1，建議在確認 EAF=0 之後可由執行 MMW $3,Ry 指令或者設定 I2CEN=0 或者設定 IRQFEN=0 皆可清除 IRQF 旗號。

2-13-2. UART

TM87ML28 提供 UART Master & Slave 完整功能，以下說明 UART 執行流程&相關細節：

1. 執行 MWM $C,Ry 指令&設定(Ry) = 01XXb 進入 UART 模式 並且 同時由 bit1,0 設定 TRB8, UART1W 模式：
   a. TRB8=0/1 設定 UART 為 8/9 bits 資料傳送模式，當 TRB8=1 時 bit 9=TB8(由執行 MWM$5,Ry 指令 bit0 設定，起始值=0)。
   b. UART1W=0 設定 UART 傳輸&接收分別由 TX&RX pins 獨立進行。UART1W=1 則設定 UART 傳輸&接收皆只由 TXD pin 共用分開進行，TXD pin 傳送/接收模式切換由 REN=0/1 控制(若欲使用在 SIOTYP2,1=11 模式以節省耗電，因 TXD pin 輸出/輸入模式由 REN=0/1 控制切換& DC 模式會關閉 Pull-High 電阻，故需在傳送出 STOP 位元使 TXD 輸出電位'1'之後確認對方已輸出電位'1' 再設定 REN=1 & 確認對方送出 STOP 位元(若對方為 TRB8=1 模式則須在 RXBF=>1 後再 delay 大於 1/2 個鮑率)後再設定 REN=0 以避免 floating 發生)。時序如下圖所示：



2. 執行 MWM $4,Ry 指令選擇 UART16 倍頻頻率，與 I2C&SPI 差異如下：
   a. SCR3=0 ：由 SCR2~-0=000~111 設定為 SXCLK=XCLK/2~256。
   b. SCR3,2=10/11：SCR1,0=00/01/10/11 選擇 SXCLK/FREQ/Timer2/Timer3 輸出為傳送/接收資料模式(TXD/RXD)每個傳輸 bit(鮑率：Baud Rates)的 16 倍頻頻率。

   由於 UART 並無同步時鐘輸出，因此除非 XCLK 本身即為 UART 鮑率的 2 的 N 次方頻率則可選擇 SXCLK 為 UART 鮑率的 16 倍頻(但須注意有無干擾的問題), 否則往往須另外產生特定頻率，故可使用 FREQ & Timer 產生 UART 鮑率的 16 倍頻 & 選擇 FREQ & Timer2 更可藉由執行 ST3OV 指令由 Timer3 微調至更加精準的等效頻率。

3. 執行 MWM $8,Ry 指令由 bit3,2 設定 SIOTYP2,1 並維持 bit2(DSIOPHL)=0 啟動內建 pull-high 電阻時各模式特性如下表所示：

SIOTYP2,1		00(initial)	01	10	11
UART	TXD(I/O) ,RXD(I)	NPOD(O)/NOD(I) ,NOD(I) (standard)	NOD(I/O) ,NOD(I) (For Multi-Master)	DC(O)/NOD(I) ,NOD(I) (For Single-Master & Multi-Slave)	DC(I/O) ,DC(I) (For One By One)

POD : P-type Normal Open Drain.
NOD : N-type Normal Open Drain.
NPOD : N-type Pseudo Open Drain.
DC : CMOS output.

a. SIOTYP2,1=00 模式：為起始設定值，TXD pin 輸出為 NPOD 模式(每次由 0=>1 時一開始 1/2 XCLK 時間會先輸出電位'1'以加速電位上升，之後才由 Pull-High 維持，故若使用在 Multi-Master 須能完全避開重疊。TXD(UART1W=1 & 非傳送模式時)/RXD pin 為 NOD 模式。

b. SIOTYP2,1=01 模式：TXD & RXD pins 皆為一直啟動 Pull-High 電阻的 NOD 模式，適用於 Multi-Master 應用。

c. SIOTYP2,1=10 模式：TXD pins 輸出時為 DC 模式以節省耗電，TXD 接收模式(UART1W=1)或是 RXD pins 則仍為 NOD 模式，適用於 Single-Master & Multi-Slave 應用。

d. SIOTYP2,1=11 模式：TXD & RXD pins 皆為 DC 模式以完全去除 Pull-High 電阻耗電&去除干擾等影響接收判斷問題，適用於一對一應用。

e. TXD pin 在所有模式輸出電位'0'時皆會關閉 pull-high 電阻以節省耗電。

4. 執行 MWM $5,Ry 指令可由 bit3 設定 UARTEN = 1/0 啟動/關閉 UART 並由 bit2~0 設定 REN,RENRLS,TB8：

a. TB8：當 TRB8=1 時可用於設定傳送時的第九個資料位元。

b. RENRLS：依 TRB8=0/1 模式，當 RENRLS=0 時接收到 STOP/RB8 位元時若 RXBFEN=1 便會直接設定 RXBF=1。RENRLS=1 時則當接收到 STOP-bit/RB8=0 時則不會設定 RXBF=1。RENRLS=1 應用目的如下說明：

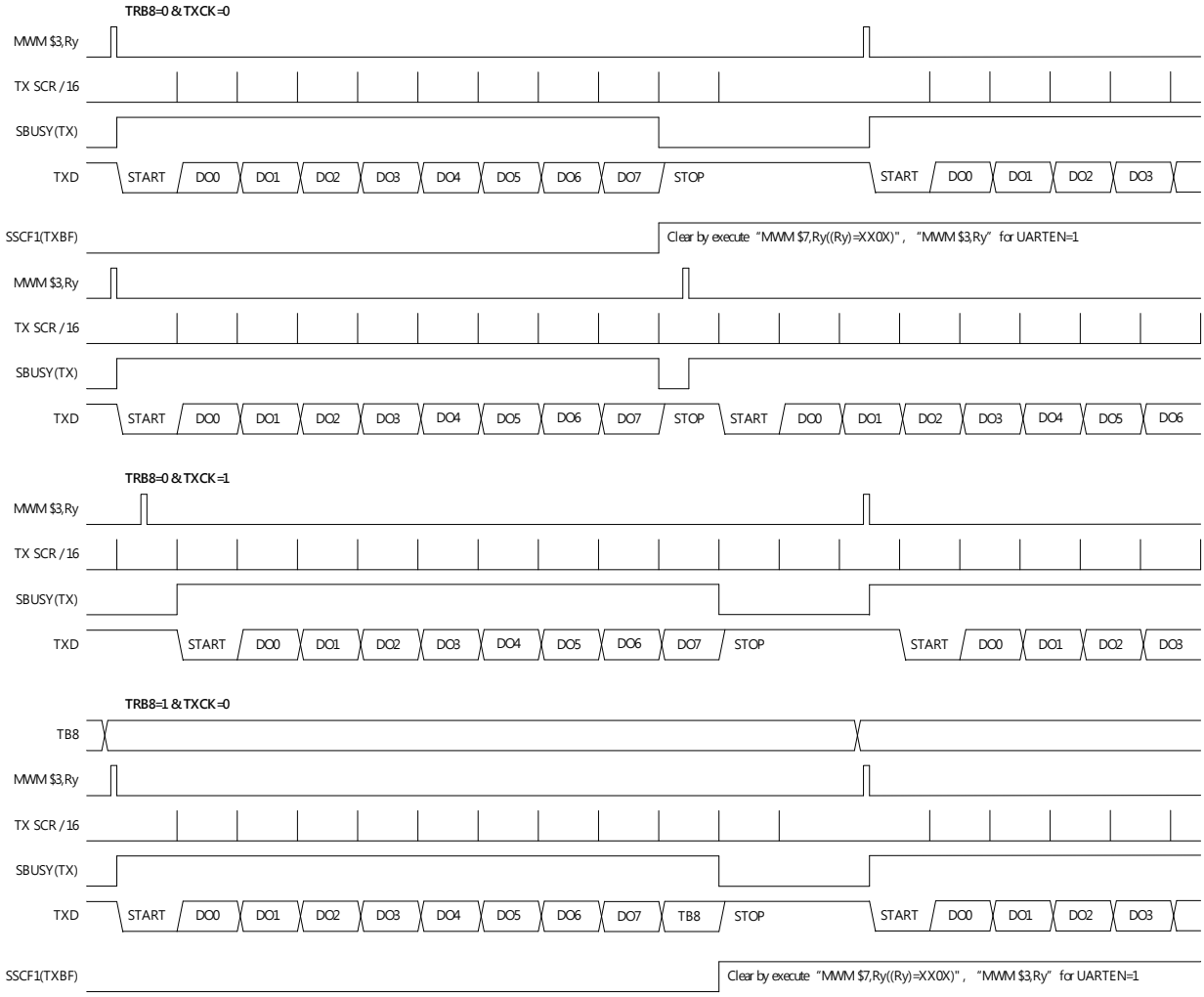
(1) TRB8=0：STOP 位元正常應為電位'1'，但若接收到電位'0'則表示 STOP 位元不正常故可因此忽略不處理。

(2) TRB8=1：當 Multi-Slave 應用時所接收到的 RB8 位元=0/1 代表資料/位址位元組，故當 RB8=1 須能產生中斷來檢查接收的位址是否是定址到自己，若不是則維持 RENRLS=1 以忽略後面 Master 所傳出的資料位元組，若是則設定 RENRLS=0 以接收後面 Master 所傳出的資料位元組。

c. REN：起始值 =0，設定 REN=1 將依 UART1W=0/1 啟動 RXD/TXD 接收模式，若 UART1W=1 & TXD 為 DC 輸出模式則當 REN=1 才會關閉輸出。

5. 執行 MWM $6,Ry 指令可由 bit3~0 分別設定 TXDCHK,CLRXF,RXCK,TXCK 功能。分別說明如下：

a. TXCK：起始值 =0，TXCK=0 當傳送開始前&結束後(若 RX 16 倍頻 Clock Source 與 TX 不同或是無 RX 接收進行中)便會停止 16 倍頻 Clock 以節省耗電。設定 TXCK=1 會使 TX Clock 一直處於啟動狀態使每次傳輸可維持在同樣鮑率軌跡上以針對傳輸對方無自行偵測 START 位元歸零調整能力。當設定 TXDCHK=1 & TX & RX 設定同樣 Clock Source 時須設定 TXCK=1 以避免 TX START 位元與 Clock 因產生 race 問題 而誤產生 TXOF。比較 TXCK=0/1 TX-SCR/16 時序如下圖所示：

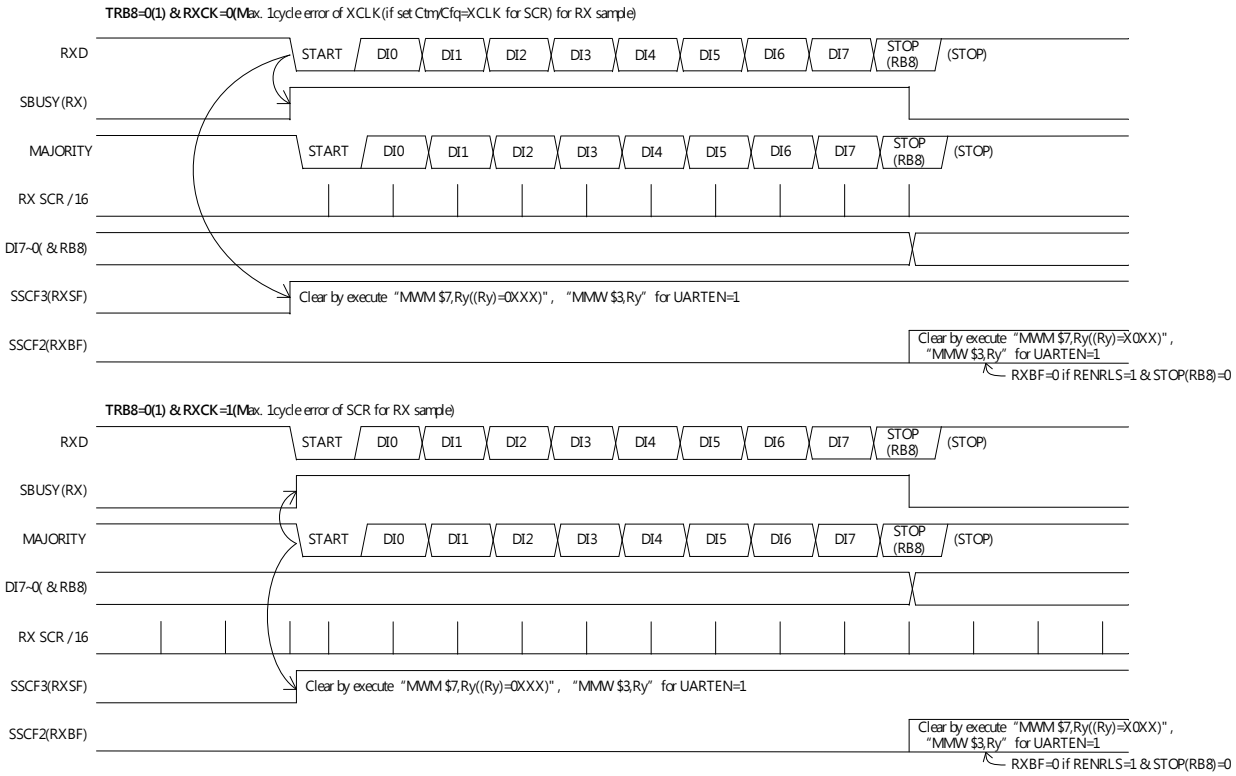


當 TXCK=0 時如上圖所示若傳送 STOP 位元完成時仍無執行 MWM $3,Ry 指令傳送下一筆資料則會停止 TX clock 以節省耗電直到執行 MWM $3,Ry 指令再啟動 TX clock，此時會因重新計數而使鮑率軌跡與之前不同。若在 STOP 位元結束前執行 MWM $3,Ry 指令傳送下一筆資料則 TX clock 不會停止而使 START 位元等到與 TX SCR/16 同步後才輸出電位'0'，此時鮑率軌跡仍維持不變。

當 TCXK=1 則如上圖所示即使在傳送 STOP 位元完成之後仍繼續維持 TX clock 動作，START 位元會等到與 TX-SCR/16 同步後才輸出電位'0'，因此 鮑率軌跡仍維持不變。

TRB8=1 時仍會等 STOP 位元傳送結束後再停止 TX clock。

b. RXCK：起始值 =0，RXCK=0 時偵測到 RXD/TXD pin(UART1W=0/1)下降緣會立即設定 RXSF=1(若 RXSFEN=1)&此時才會啟動 RX clock，等到接收到 STOP/RB8(TRB8=0/1)之後便會停止 RX clock 以節省耗電。RXCK=1 時則會一直啟動 RX clock ，此時會經過 Majority Detect 過濾判斷為 RXD/TXD pin(UART1W=0/1)電位：1=>0 才會設定 RXSF=1(若 RXSFEN=1) 以降低誤判 START 機率& 啟動接收程序並調整 RX SCR/16 軌跡。比較 RXCK=0/1 RX-SCR/16 時序如下圖所示：



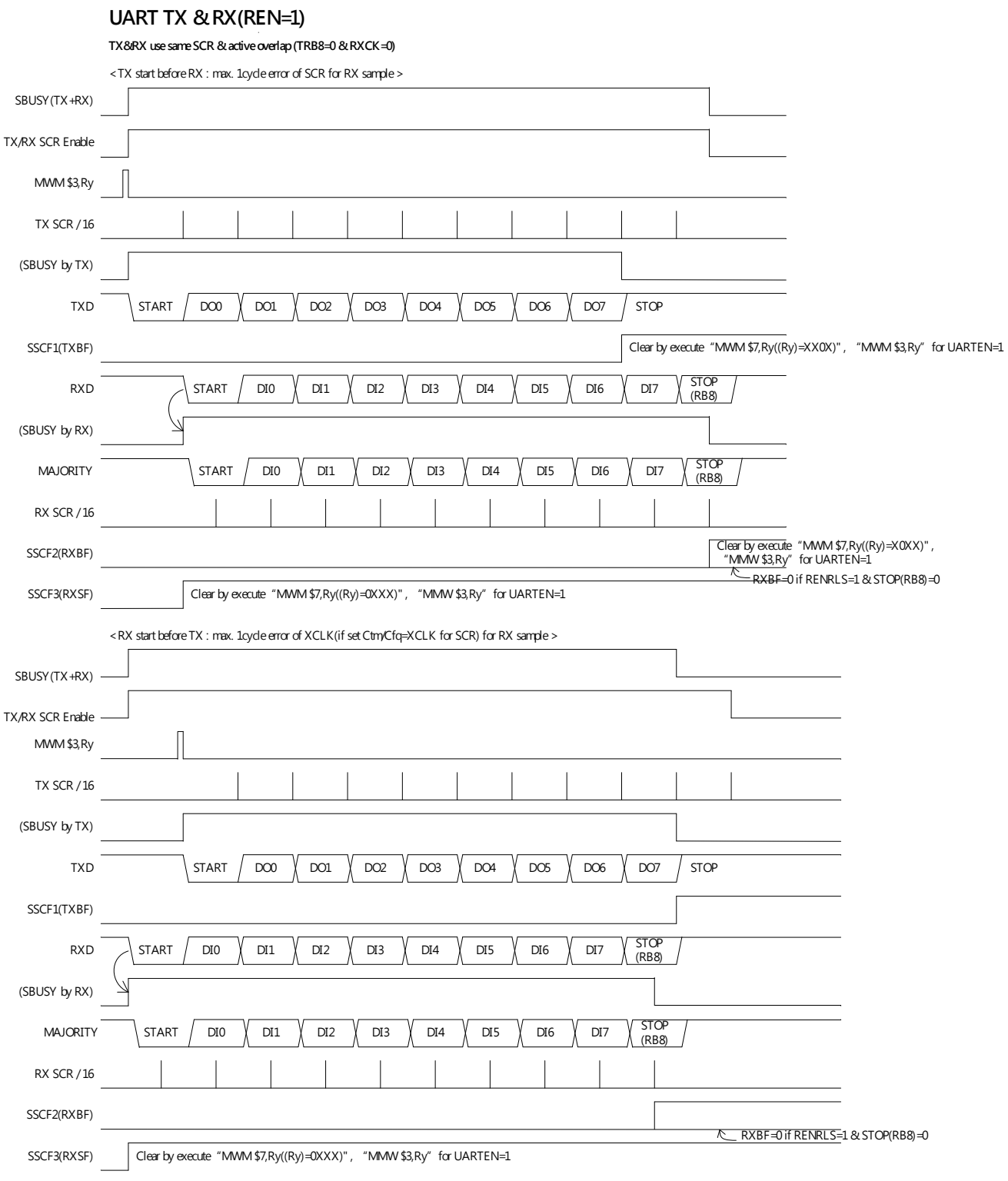
由上圖所示，雖然 RXCK=0/1 會造成 RXSF=>1 時間不同，但後續皆由經過 Majority Detect 過濾後抓取資料以降低接收資料錯誤機率，藉由調整抓取資料時間點使兩者 RXBF=>1 誤差小於一個 RX-SCR cycle。

c. CLRXF：寫入 1 可清除 RCVBF/TXSF & RCVOVF/TXOF。

d. TXDCHK：設定 1 可啟動 TXD pin check 功能並且切換 RCVBF/RCVOVF 變成 TXSF/TXOF 以供判斷。當 TXDCHK=1 時若 RX&TX 設定同樣的 clock source 時，若無法確保 clock source 是由 TX START 所啟動則須設定 TXCK=1 以避免因 race 問題而有誤產生 TXOF=1 風險。

6. 執行 MMW Ry,$6 指令依 TXDCHK=0/1 可由 bit3~0 分別讀取 RCVOVF/TXOF,RCVBF/TXSF,RB8F, SBUSY 旗號。

a. SBUSY：起始值=0，當每次 TX 或是 RX 動作時皆會設定 SBUSY=1 直到 TX&RX 動作皆結束後才會回到 SBUSY=0 狀態。時序如下圖所示：



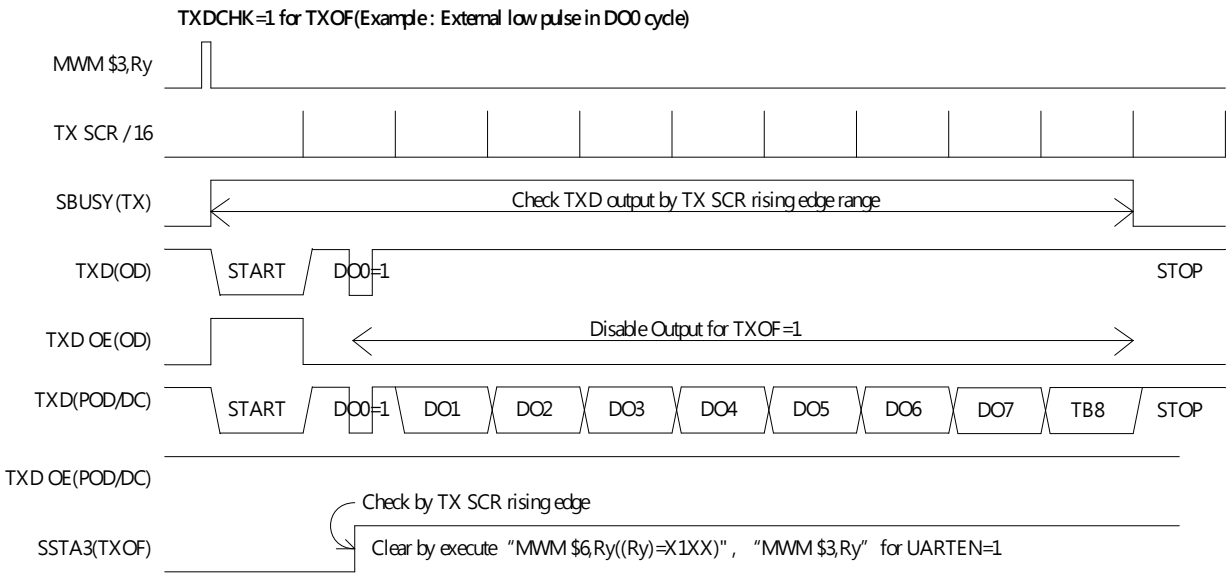
b. RB8F：起始值=0，在每次接收動作完成後可由讀取 RB8F 得知 STOP/RB8 位元(TRB8=0/1)電位。

c. RCVBF：起始值=0，當接收資料完成後硬體會自動設定 RCVBF=1 直到執行 MMW Ry,$3 指令讀取資料後會自動清除，此外亦可由寫入 CLRXF=1 或是設定 UARTEN=0 強迫清除。

d. RCVOVF：起始值=0，當 RCVBF=1 之後若未清除則在接收下一筆資料完成後不僅 RCVBF 仍=1 亦會設定 RCVOVF=1，同樣可由執行 MMW Ry,$3 指令或是寫入 CLRXF=1 或是設定 UARTEN=0 清除。

e. TXSF：起始值=0，當設定 TXDCHK=1 後若在無進行 TX 傳送動作時(UART1W=1 時須增加 REN=0 條件)卻在 TXD pin 上有下降緣發生則硬體會自動設定 TXSF=1 以供判斷 TXDpin 上是否會有雜訊發生而需進行干擾改善或者在 Multi-Master 應用時判斷是否已有其他 Master 進行傳送動作，可由執行 MWM $3,Ry 指令或是寫入 CLRXF=1 或是設定 UARTEN=0 清除，只要當 TXSF=1 時不會產生 TXBF =1。

f. TXOF：起始值=0，當進行 TX 傳送動作時所傳送資料包含(START&TB8 位元)與 TXD pin 上電位不同時，硬體除了會自動設定 TXOF=1 之外若 TXD 為 NOD 輸出模式(SIOTYP2,1=01)亦會立即停止輸出，但 SBUSY 仍會繼續維持=1 直到整個傳送流程結束，只要當 TXOF=1 時不會產生 TXBF =1)。範例時序如下圖所示：



7. 執行 MWM $7,Ry 指令可由 bit3~0 分別設定 RXSFEN,RXBFEN,TXBFEN,INTFEN=1(起始值=0)以產生 RXSF,RXBF,TXBF,INTF 旗號(起始值=0)由執行 MMW Ry,$7 指令讀取，若設定 IEF2=1 或是 HEF2=1 當 RXSF,RXBF,TXBF,INTF 其中一個旗號=1 則可啟動 HRF2 Release。

a. INTFEN&INTF：起始值=0，由於 SIO 與 INT 共用 HRF2 Release，故欲啟動 INT 本身 release 必須增加設定 INTFEN=1 方可在 INT pin 符合轉態條件時產生 INTF=1，設定 INTFEN=0 可清除 INTF 旗號。

b. TXBFEN&TXBF：起始值=0，設定 TXBFEN=1 可在每次開始傳送 STOP 位元時產生 TXBF=1(若產生 TXSF=1 或者 TXOF=1 則不會產生 TXBF=1)，執行 MWM $3,Ry 指令或者設定 UARTEN=0 或者設定 TXBFEN=0 皆可清除 TXBF 旗號。

c. RXBFEN&RXBF：起始值=0，設定 RXBFEN=1 可在每次接收到 STOP/RB8 位元(TRB8=0/1)時產生 RXBF=1，執行 MMW $Ry,$3 指令或者設定 UARTEN=0 或者設定 RXBFEN=0 皆可清除 RXBF 旗號。

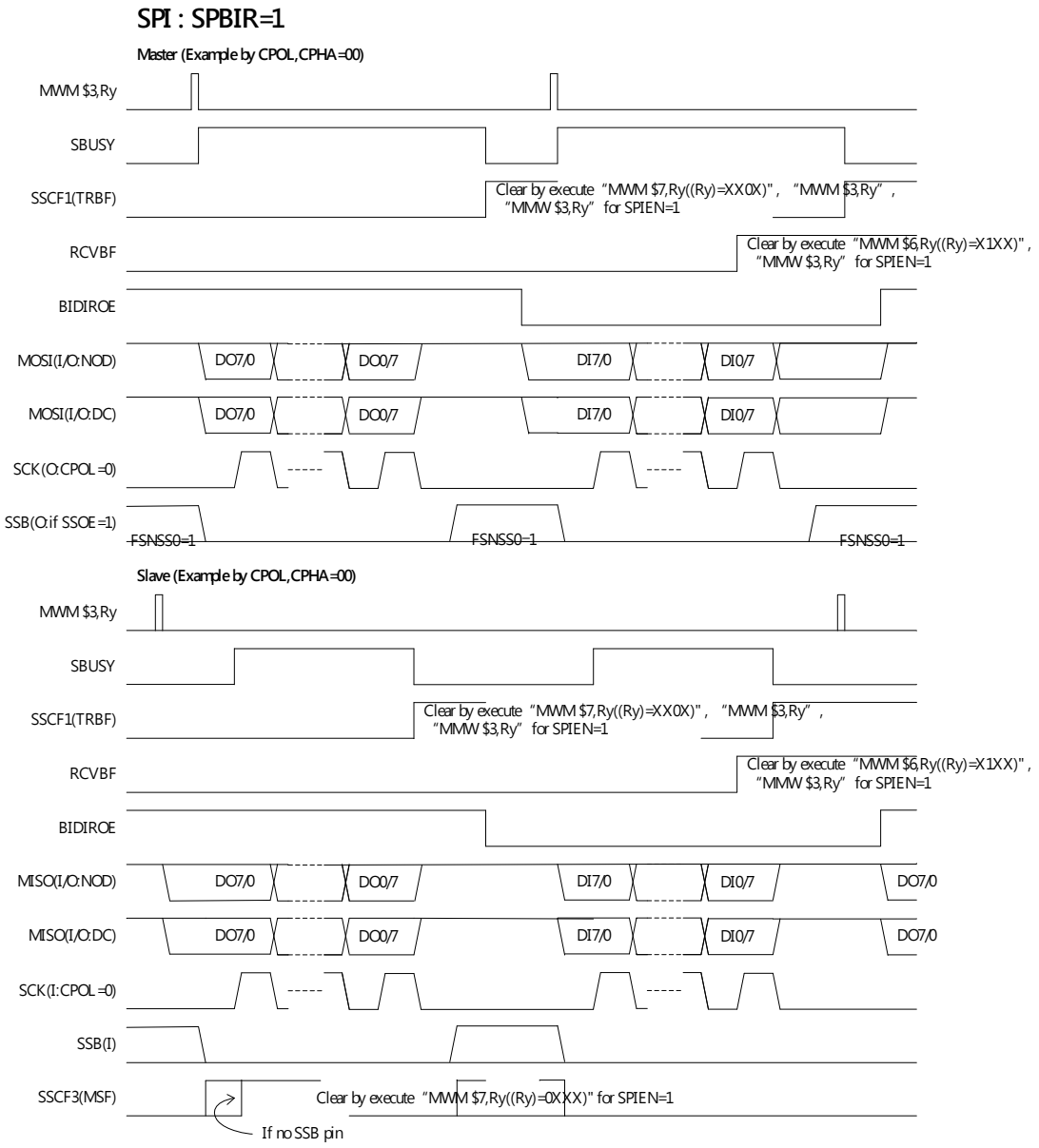
d. RXSFEN&RXSF：起始值=0，設定 RXSFEN=1 & REN=1 可在偵測到 RXD/TXD (UART1W=0/1) pin 有下降緣(RXCK=1 時會先經 Majority Detect 過濾降低錯誤機率) 時產生 RXSF=1 以便程式可以在 RXBF=>1 之前先完成準備工作(TM87ML28 Mask Option 提供 HALT Mode 仍可留在 Fast Mode & 維持 FAST O.S.C. 震盪以提供 UART 在 HALT Mode 仍

可在 TXCK=1 或是 REN=1 時繼續維持運作) ，但是若在 latch START 位元之前就回到電位'1' 仍會被判定為 Noise 而結束接收動作，若此時無傳送動作則 SBUSY 會回到 0 但不會設定 RXBF=1，執行 MMW $Ry,$3 指令或者設定 UARTEN=0 或者設定 RXSFEN=0 皆可清除 RXSF 旗號。

2-13-3. SPI

TM87ML28 提供 SPI Master & Slave 完整功能，以下說明 I2C 執行流程&相關細節：

1. 執行 MWM $C,Ry 指令&設定(Ry) = 1XXXb 進入 SPI 模式。並且 同時由 bit2,1,0 設定 CPOL,CPHA,SPBIR 模式：
   a. SPBIR=0 時由 MOSI/MISO pin 為 Master/Slave 輸出 pin & Slae/Master 輸入 pin。SBIR=1 時由 MOSI/MISO(=>MOMI/SISO)pin 為 Master/Slave 單一輸出&輸入 pin，此時由 BIDIROE=1 啟動輸出模式。時序如下圖所示：



b. CPHA=0/1 設定由 SCK 第一/二轉態點開始以 SCK 週期取樣。

c. CPOL=0/1 設定 SCK pin 在 SPI 無動作時停在電位'0/1', 須與 SCK Mask Option ："PULL-LOW/HIGH"一致，若 Mask Option 選擇"NO USE" 則須與外接電阻為 pull-low/high 一致(若無外接電阻，對方須為 Master & DC output 以避免 SCK pin 產生 floating 問題，此時 CPOL 單純設定與對方一致)。

2. 執行 MWM $4,Ry 指令選擇 SPI 頻率，與 I2C&SPI 差異如下：

a. SCR3=0 ：由 SCR2~-0=000~111 設定為 SXCLK=XCLK/2~256。

b. SCR3=10：SCR1,0=00/01/10/11 選擇 SXCLK/FREQ/Timer2/Timer3 輸出為 SCK 傳輸頻率。

3. 執行 MWM $8,Ry 指令由 bit3,2 設定 SIOTYP2,1 並維持 bit2(DSIOPHL)=0 啟動內建 pull-high 電阻時各模式特性如下表所示：

SIOTYP2,1		00(initial)	01	10	11
SPI		(standard & for Sub-Master)	(For Multi-Master)	(For Single-Master & Multi-Slave)	(For Single-Master & Single-Slave)
(Master)	MISO(I) ,MOSI(I/O) ,SCK(O) SSB(I/O)	NOD(I) ,NOD(I/O) ,NOD/POD(O) ,DC(I/O)	NOD(I) NOD(I/O) , NOD/POD(O) ,NOD(I/O)	NOD(I) ,NOD(I/O) ,DC(O) ,DC(I/O)	DC(I) ,DC(I/O) ,DC(O) ,DC(I/O)*SPI-1
(Slave)	MISO(I/O) ,MOSI(I) ,SCK(I) ,SSB(I)	NOD(I/O) NOD(I) ,NOD/POD(I) ,DC(I)	NOD(I/O) NOD(I) , NOD/POD(I) ,NOD(I)	NOD(I/O) ,NOD(I) ,DC(I) DC(I)	DC(I/O) ,DC(I) ,DC(I) ,DC(I)*SPI-1

POD : P-type Normal Open Drain.
NOD : N-type Normal Open Drain.
DC : CMOS output.
*SPI-1 : If SSB pin use & DSIOPHL(SPI)=0, MISO/MOSI(I) enable Pull-High when SSB=1, set DSIOPHL=1 to save current if MISO/MOSI(I)=0 when SSB=1.

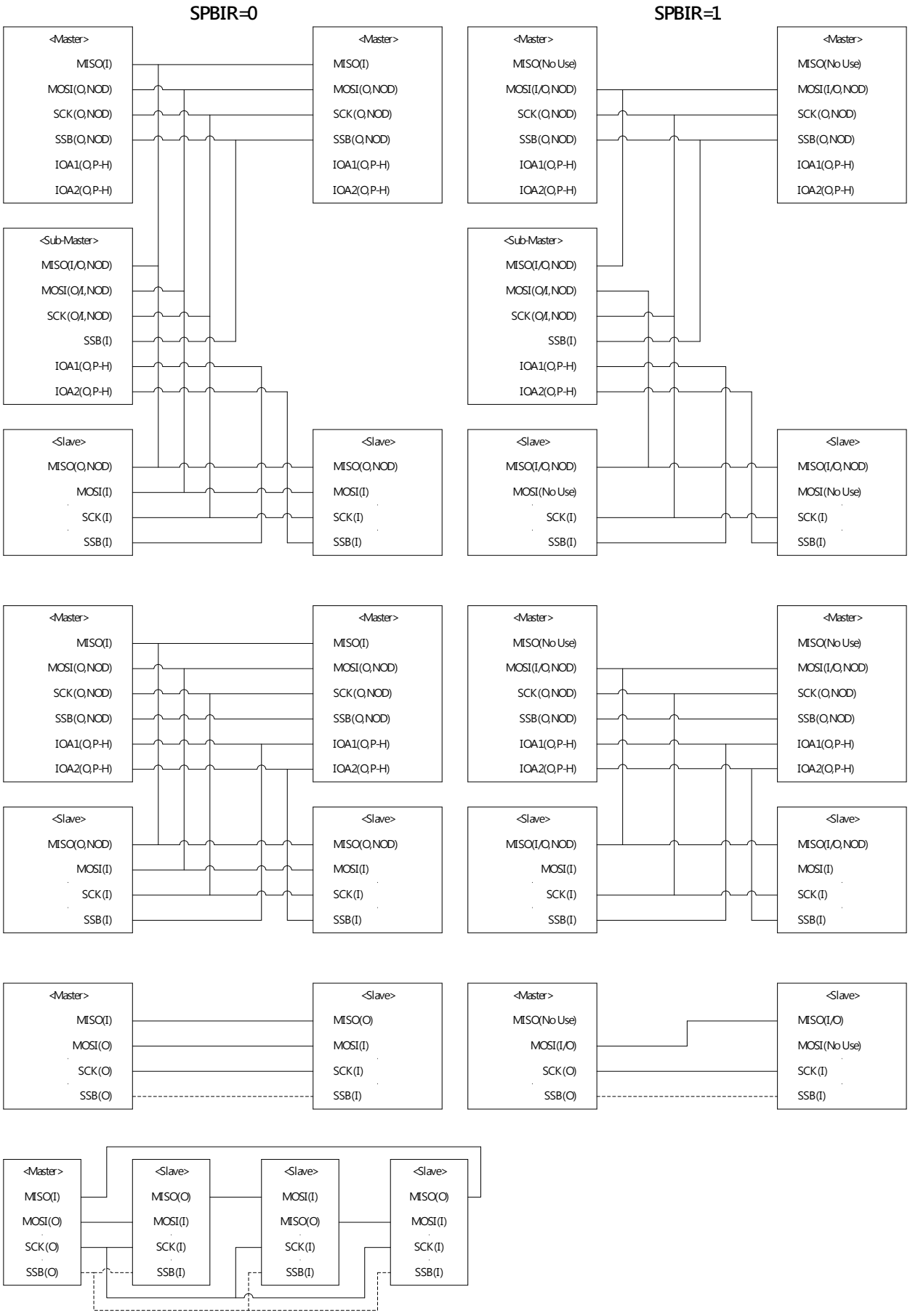
a. SIOTYP2,1=00 模式：為起始設定值，MISO & MOSI & SCK pins 皆為 Open Drain 模式，只有 SSB 為 DC pin，適用於 Sub-Master 應用(當在 Master 模式時維持 SSOE=0，一旦 SSB pin 被上一級 Master 拉至電位'0'則會立即切換成 Slave 模式&會產生 MSF(若設定 MSFEN=1))。

b. SIOTYP2,1=01 模式：所有 SPI pins 皆為 Open Drain 模式，適用於 Multi-Master 應用(此時須設定 SSOE=1，一旦 SSB pin 被其它 Master 切換電位由'1'->'0'或是'0'->'1'皆會產生 MSF(若設定 MSFEN=1)以得知目前有其它 Master 佔用中)

c. SIOTYP2,1=10 模式：MOSI & MISO pins 為 Open Drain 模式，SCK & SSB 則為 DC 模式，適用於 Single-Master & Multi-Slave 應用以節省 SCK & SSB 耗電。

d. SIOTYP2,1=11 模式：所有 SPI pins 皆為為 DC 模式，適用於 Single-Master & Singlei-Slave 應用以節省所有 SPI pins 耗電 & 提升傳輸速度。

下圖為各種 SPI 應用電路參考：



圖中 Sub-Master 應用，上一級的 Master 若亦要拉出 IOA1,2 控制選擇底下兩個 Slave 單元直接

進行傳輸則 SSB pin 須維持在電位'1'以避免 Sub-Master 誤被選擇轉換成 Slave 模式&進行傳輸，此時須改由 IOA1,2 產生 IOA 中斷告知另一個 Master & Sub-Master 以避免產生衝突。

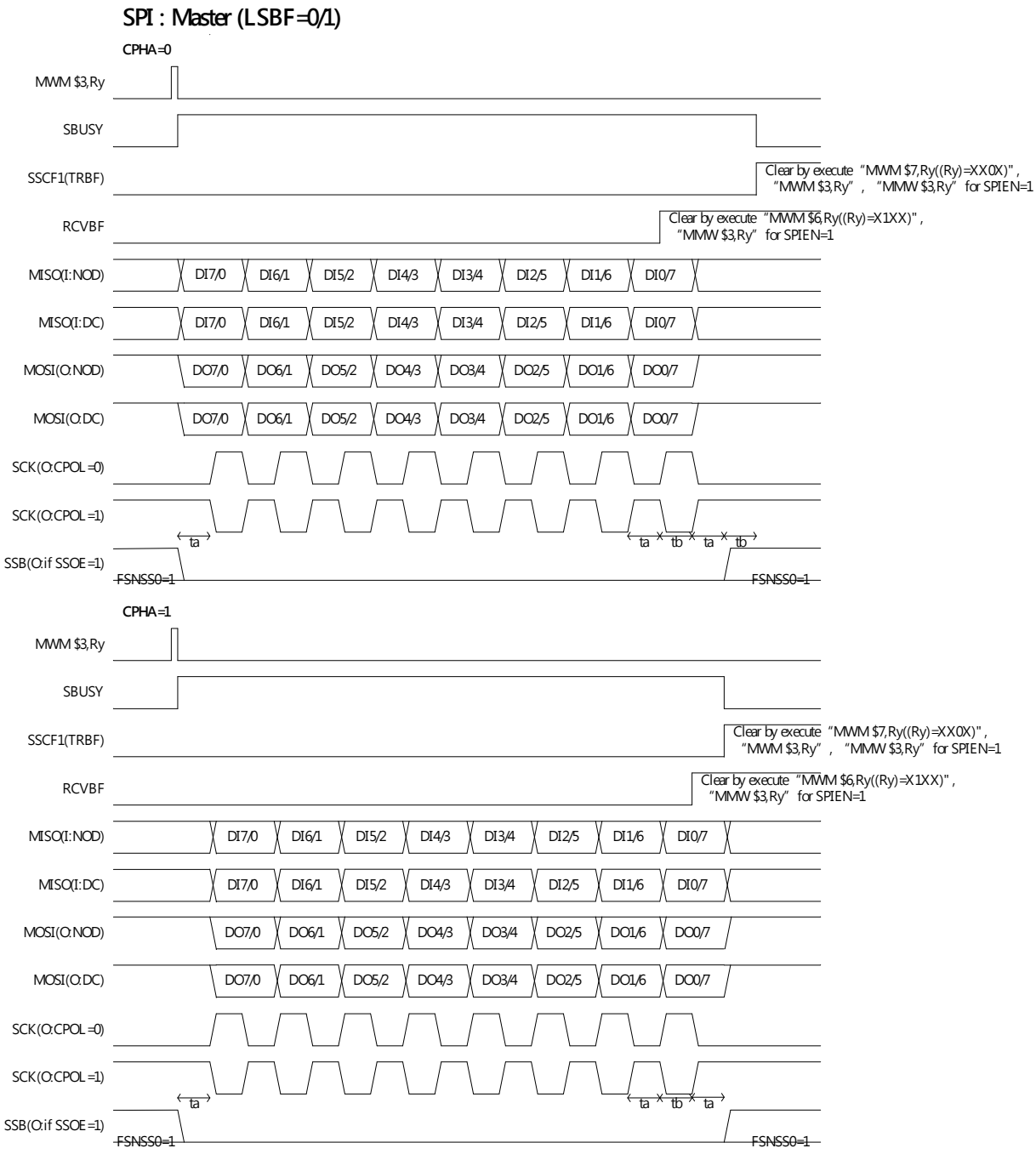
4. 執行 MWM $5,Ry 指令可由 bit3 設定 SPIEN = 1/0 啟動/關閉 UART 並由 bit2~0 設定 MSTR,BIDIROE,LSBF：

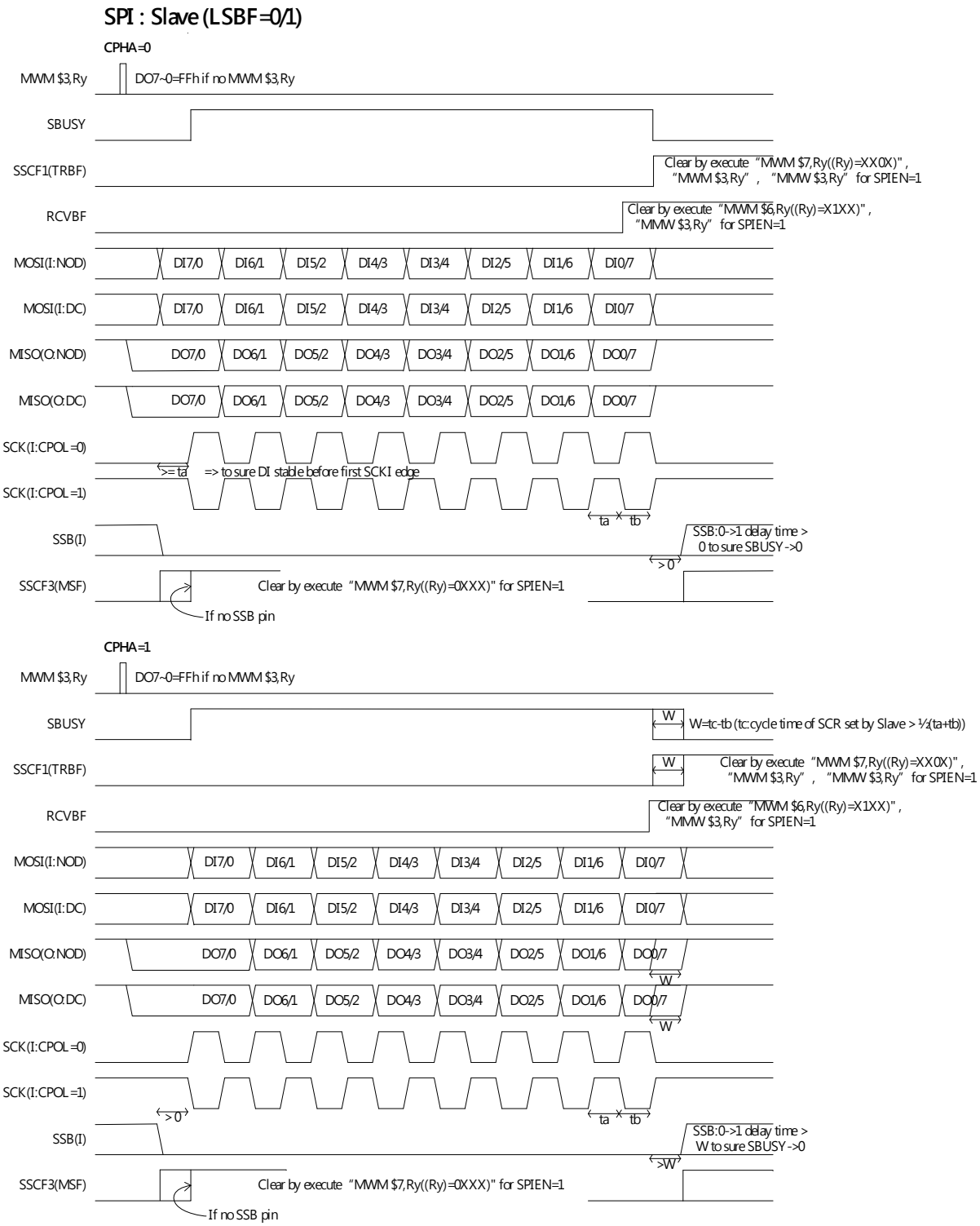
   a. LSBF：起始值=0，設定 LSBF=0/1 時資料會先從位元 7/0 傳輸至位元 0/7 結束。

   b. BIDIROE：起始值=0，當 SPBIR=1 時設定 BIDIROE=1 可在 Master/Slave 模式下啟動 MOSI/MISO pin 為輸出模式。

   c. MSTR：起始值=0 為 Slave 模式，設定 MSTR=1 可切換成 Master 模式。在 Slave 模式若未執行 MWM $2,Ry & MWM $3,Ry 指令設定輸出位元組則輸出值固定為 FFh。

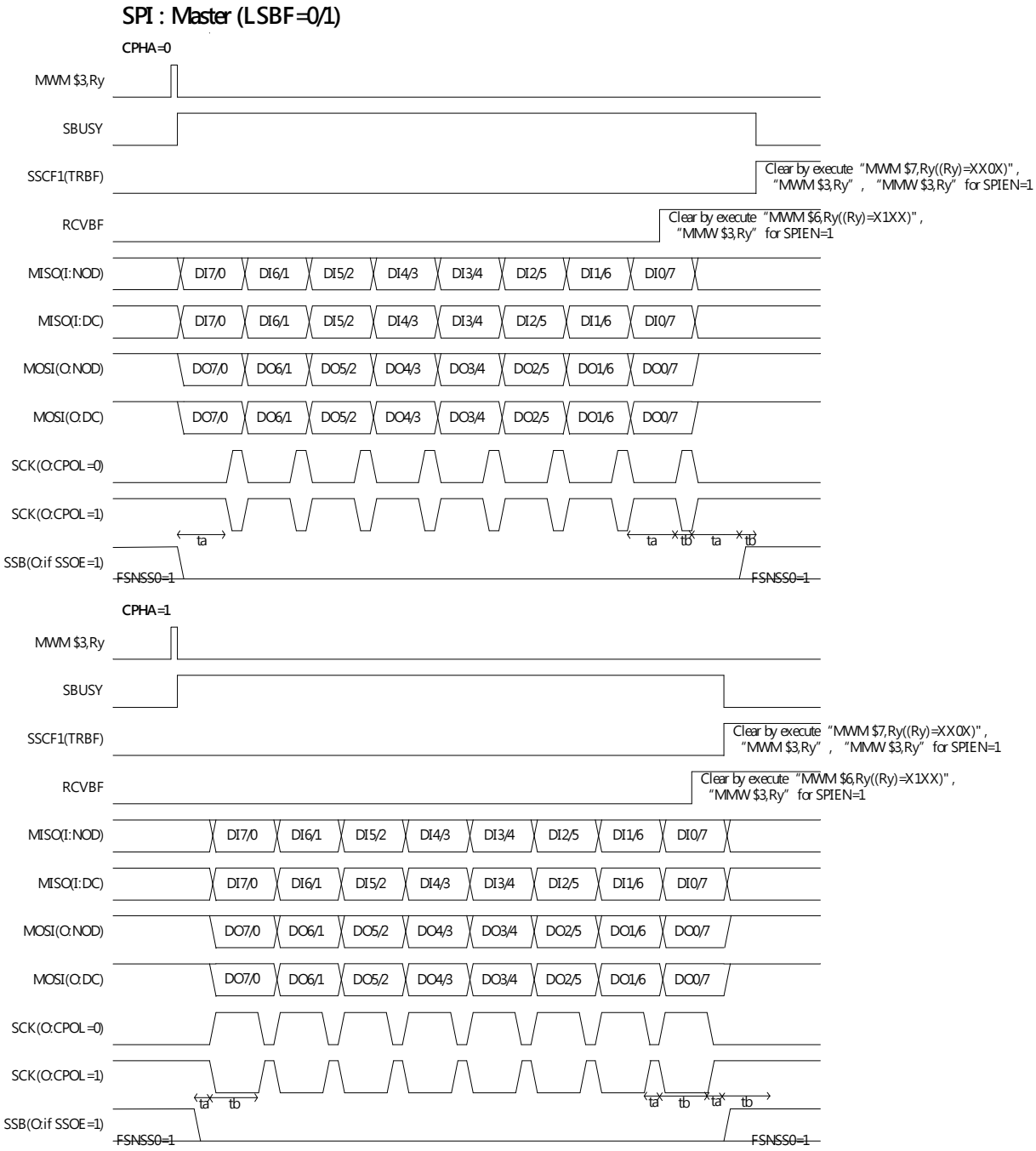
   Master & Slave 模式等時序如下列圖示：


SPI : Master (LSBF=0/1)
CPHA=0
MWM $3,Ry
SBUSY
SSCF1(TRBF)
Clear by execute "MWM $7,Ry((Ry)=XX0X)", "MWM $3,Ry", "MMW $3,Ry" for SPIEN=1
RCVBF
Clear by execute "MWM $6,Ry((Ry)=X1XX)", "MMW $3,Ry" for SPIEN=1
MISO(I:NOD)
MISO(I:DC)
MOSI(O:NOD)
MOSI(O:DC)
DI7/0
DI6/1
DI5/2
DI4/3
DI3/4
DI2/5
DI1/6
DI0/7
DO7/0
DO6/1
DO5/2
DO4/3
DO3/4
DO2/5
DO1/6
DO0/7
SCK(O:CPOL=0)
SCK(O:CPOL=1)
ta
tb
SSB(O:if SSOE=1)
FSNSS0=1
CPHA=1



由圖所示，Slave & CPHA=1 模式時由於在接收 SCK 最後一個 clock 之後要維持最後輸出位元以確認對方可正確接收，內部會依所選擇 SCR 頻率在開始送最後一個位元時啟動 SCR 以產生延遲使 MISO pin 輸出資料可在 SCK 結束後繼續維持輸出，但須注意選擇 SCR 頻率週期必須至少大於所接收 SCK 週期的一半 & 若 code option 有選擇使用 SSB pin 則所接收 SSB 建議在 MISO 輸出結束使 SBUSY->0 之後在轉換電位'0'->'1'以避免誤判為異常 MSF=1。

若選擇 Clock Source 為 FREQ(1/1duty),Timer 則輸出 SCK 波形因 H/L 寬度不同而對取樣時間有所差別，Master 模式時序如下圖所示：



由上圖可知，當 CPHA=0 可增加取樣前 MISO & MOSI & SCK pins Pull-High/Low 穩定時間，故在無干擾問題下可選擇 Pull-High/Low 較大阻值以減少耗電。但是 CPHA=1 時若 SCK pin 為 Open Drain 模式則會增加 SCK pin Pull-High/Low 耗電時間 & 反而造成 SCK pinPull-High/Low 穩定時間不足，但若 SCK 為 DC 模式則無此問題。故欲使用 FREQ/Timer 當 SPI clock source 以增加取樣前 Pull-High/Low 穩定時間時建議選擇 CPHA=0 模式， 若須選擇 CPHA=1 模式則需選擇 SCK pin 為 DC 模式，但因距離 SSB:0->1 時間縮短故若 Slave 端亦是使用 TM87ML28 搭配須更注意避免在 SSB pin 電位：'0'->'1' 時因仍在 BUSY=1 而產生異常 MSF 旗號。

5. 執行 MWM $6,Ry 指令可由 bit3~0 分別設定 TXOCHK,CLRXF,SSOE,FSNSS0 功能。分別說明如下：

a. FSNSS0：起始值=0，在 Master 模式設定 SSOE=1 時 FSNSS0=0 只有在執行 MWM $3,Ry 指令啟動傳輸時才會輸出 SSB 電位'0'，位元組傳輸結束後便會自動回到電位'1'。設定 FSNSS0=1 會使 SSB pin 立即輸出電位'0'直到設定回 FSNSS0=0，適用於在傳輸前提供更多時間給對方反應或是多位元組傳輸等應用或是在 Slave & CPHA=1 模式延遲 SSB pin 電位：'0'->'1'以避開仍在 BUSY=1 問題。

b. SSOE：起始值=0，在 Master 模式時設定 SSOE=1 會使 SSB 切換成輸出模式 & 依 FSNSS0=0/1 設定 SSB pin 輸出為自動/強制電位'0'模式。

c. CLRXF：寫入 1 可清除 RCVBF & RCVOVF。

d. TXOCHK：起始值=0，設定 TXOCHK=0/1 決定 SSRE2=WCOFEN/EROFEN & 所產生 SSCF2= WCOF/EROF 功能。

6. 執行 MMW Ry,$6 指令可由 bit3~0 分別讀取 RCVOVF, RCVBF, SSI, SBUSY 旗號。

a. SBUSY：起始值=0，當執行每次傳輸動作時皆會設定 SBUSY=1 直到傳輸動作結束後才會回到 SBUSY=0 狀態。

b. SSI：讀取 SSB pin 電位反向旗號，當 SSB pin 電位為'0/1'時 SSI=1/0，可由 SSI 確認 SSB pin 狀態。

c. RCVBF：起始值=0，當接收資料完成後硬體會自動設定 RCVBF=1 直到執行 MMW Ry,$3 指令讀取資料後會自動清除，此外亦可由寫入 CLRXF=1 或是設定 SPIEN=0 強迫清除。

d. RCVOVF：起始值=0，當 RCVBF=1 之後若未清除則在接收下一筆資料完成後不僅 RCVBF 仍=1 亦會設定 RCVOVF=1，同樣可由執行 MMW Ry,$3 指令或是寫入 CLRXF=1 或是設定 SPIEN=0 清除。

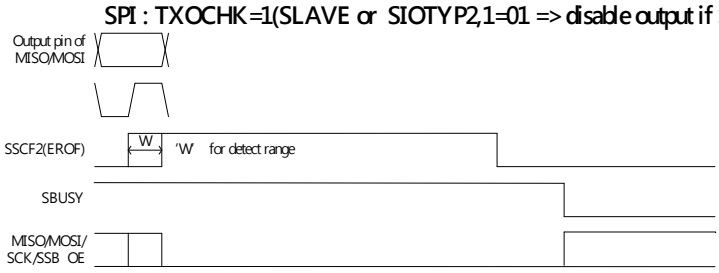
7. 執行 MWM $7,Ry 指令可由 bit3~0 分別設定 MSFEN,WCOFEN/EROFEN,TRBFEN,INTFEN=1(起始值=0)以產生 MSF,WCOF/EROF,TRBF,INTF 旗號(起始值=0)由執行 MMW Ry,$7 指令讀取，若設定 IEF2=1 或是 HEF2=1 當 MSF,WCOF/EROF,TRBF,INTF 其中一個旗號=1 則可啟動 HRF2 Release。

a. INTFEN&INTF：起始值=0，由於 SIO 與 INT 共用 HRF2 Release，故欲啟動 INT 本身 release 必須增加設定 INTFEN=1 方可在 INT pin 符合轉態條件時產生 INTF=1，設定 INTFEN=0 可清除 INTF 旗號。

b. TRBFEN&TRBF：起始值=0，設定 TRBFEN=1 可在每次 SBUSY:1=>0 時產生 TRBF=1，執行 MMW $3,Ry 指令或者執行 MWM Ry,$3 指令或者設定 SPIEN=0 或者設定 TRBFEN=0 皆可清除 TRBF 旗號。

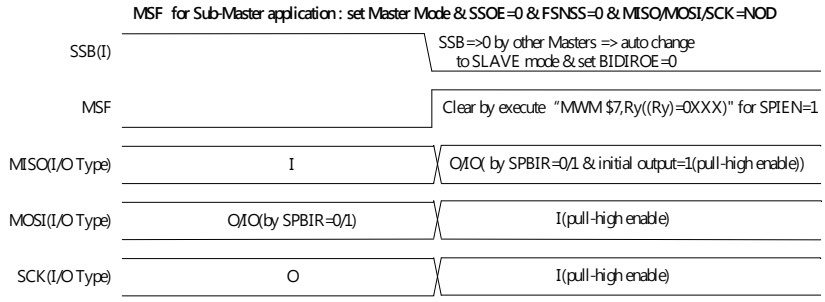
c. WCOFEN&WCOF：起始值=0，設定 TXOCHK=0 時 SSRE2/SSCF2 當 WCOFEN&WCOF 使用。當設定 WCOFEN=1 時，若在 SBUSY=1 時誤執行 MWM$3,Ry 指令時傳送資料&動作不會受到影響但會產生 WCOF=1 提醒，執行 MMW $3,Ry 指令或者設定 SPIEN=0 或者設定 WCOFEN=0 皆可清除 WCOF 旗號。

d. EROFEN&EROF。起始值=0，設定 TXOCHK=1 時 SSRE2/SSCF2 當 EROFEN&EROF 使用。當設定 EROFEN=1 時，若 MOSI&MISO pins 其中一個在輸出時電位與資料不一致會產生 EROF=1，執行 MMW $3,Ry 指令或者設定 SPIEN=0 或者設定 EROFEN=0 皆可清除 EROF 旗號。當 EROF=1 時若為 SIOTYP2,1=01 或者 Slave 模式則硬體會立即停止輸出，時序如下圖所示：

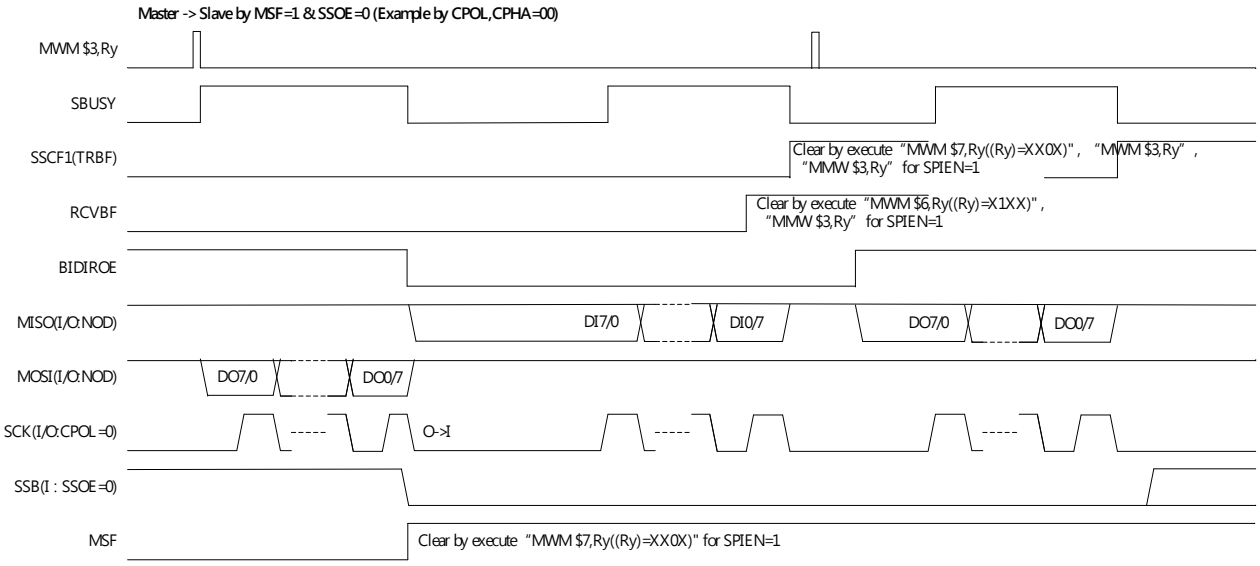


e. MSFEN&MSF：起始值=0，當設定 MSFEN=1 時，若有以下條件發生時皆會產生 MAF=1，設定 SPIEN=0 或者設定 MSFEN=0 皆可清除 MSF 旗號：

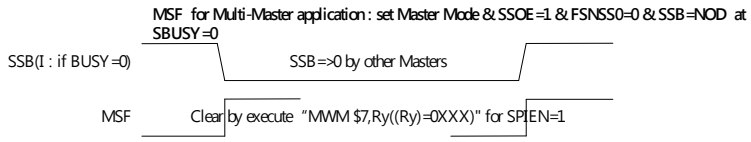
(1) Sub-Master 應用(SIOTYP2,1=00 or 01)：若在 Master 模式&SSOE=0 時 SSB 產生下降緣則會由硬體直接切換成 Slave 模式並產生 MSF=1。對 MISO,MOSI,SCK pin 影響如下圖以 SIOTYP2,1=01 模式：



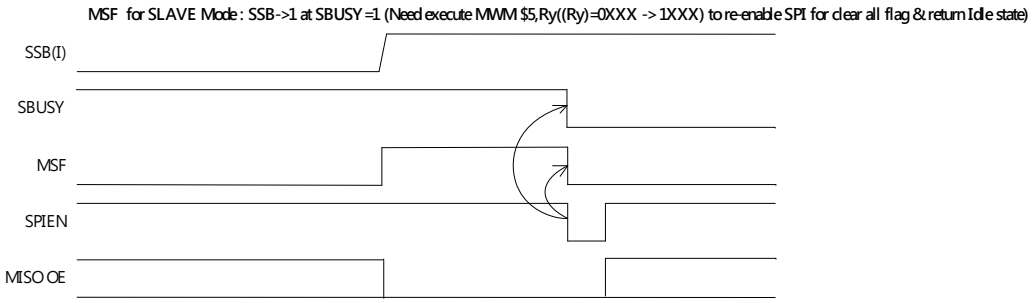
時序如下圖以 SPBIR=1 為例所示：



(2) Multi-Master 應用(SIOTYP2,1=01)：若在 Master 模式&SSOE=1&FSNSS=0&SBUSY=0 時 SSB 電位由'0'->'1'/'1'->'0'皆會產生 MSF=1 以提醒有其它 Master 有啟動/結束傳輸動作。時序如下圖所示：



(3) Slave 應用時 SSB pin 電位由'1'->'0' & '0'->'1' 皆會產生 MSF=1，然後由 SSI 旗號是否=1 確認目前是否仍在被選擇狀態中。但是若 SSB 在 SBUSY=1 時就轉為電位'1'則硬體會自動停止輸出，若對方 Master 是在尚未傳送第 8 個 SCK 轉態前就結束則 SBUSY 會仍維持在 1 的狀態，此時須先設定 SPIEN=0 再啟動方可重新工作。時序如下圖所示：



(4) Slave 應用時若無使用 SSB pin 時(尤其是一對一應用欲省略 SSB pin 時)則當 SBUSY=0 時 SCK 第一次轉態時會產生 MSF=1 以提醒系統有傳輸動作開始進行。

第三章 LCD DRIVER

4BIT 系列 MCU 內建一個最多可顯示 1024 個 dot 的 dot matrix LCD driver (64 SEGs x 16 COMs)。LCD panel 所顯示的畫面內容資料全部儲存在 LCD display memory 中，而且 每一個 dot 的 ON/OFF 資料都會儲存在特定的 LCD display memroy 位元上，SEG pin 的 輸出波形則會根據 LCD display memory 位元的資料而改變。

LCD driver 的部份 COM&SEG 輸出腳位可以 code option 的方式設定成 DC 或是 P_open Drain 輸出腳位，而且 DC 或是 P_open Drain 輸出的資料也一樣儲存在特定的 LCD display memroy 位元上。部分 MCU 也可將部分 COM 輸出腳位可以 code option 的方式設定成 SEG 輸出腳位。部分 MCU 也可以 code option 的方式設定將 COM 輸出腳位對調以配合單面跑線 PCB 需求，例如 TM87ML26 除了維持相容於既有 TM8726 COM 輸出腳位順序之外亦可將 COM1~9 順序對調為 COM9~1，故當 duty<9 時因為須由 COM9 開始以 code option 的方式設定成 DC 或是 P_open Drain 輸出腳位時則 COM1 可與 SEG1 保時相鄰位置而去除中間穿插 DC 或是 P_open Drain 的困擾，使 LCD 跑線可集中縮小 PCB 面積以及減少干擾影響。

當 MCU 進入 RESET 狀態的時候，LCD driver 的輸出波形會使得全部的 LCD dot 依 code option 決定都點亮或是都不會點亮。這個在 RESET 狀態就設定的 LCD 波形會持續輸出到第一個 LCD 的相關指令執行後才會開始改變。

在 MCU 進入正常操作模式時，無論目前 LCD display memory 所存放的資料為何，執行 SF2 4h 指令後可以暫時關閉 LCD 全部的畫面，執行 RF2 4h 可以恢復顯示 LCD 全部的 畫面。

3-1. LCD LIGHTING SYSTEM

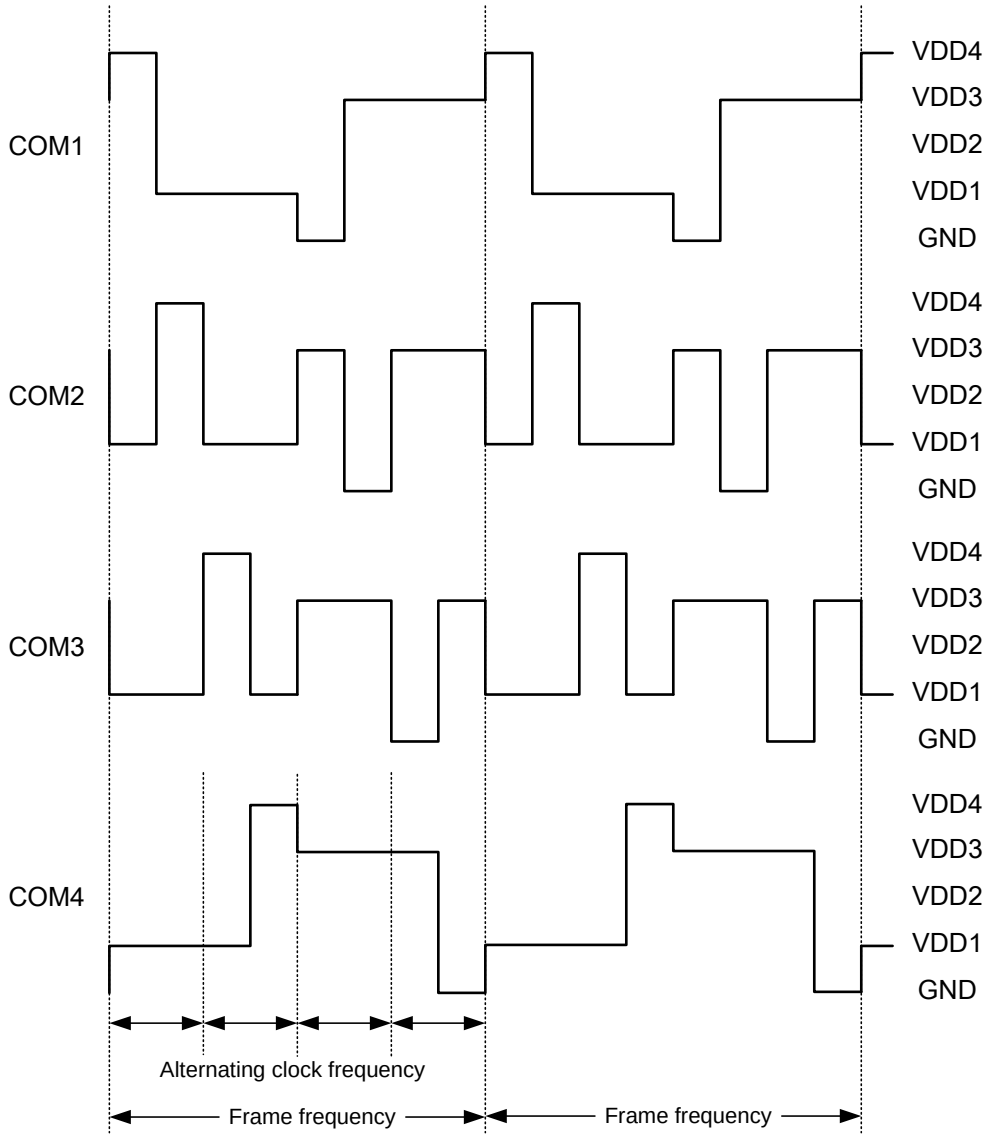
LCD lighting systems 是由 LCD bias 以及 LCD duty cycle 所組合而成，而 LCD bias 以及 LCD duty cycle 都是以 code option 的方式選擇。

LCD duty cycle 可以選擇的項目有：1/1 duty, 1/2duty, 1/3duty, 1/4 duty, 1/5 duty, 1/6 duty, 1/8 duty, 1/9 duty, 1/10 duty, 1/11 duty, 1/12 duty, 1/13 duty, 1/14 duty, 1/15 duty, 1/16 duty。

LCD bias 可以選擇的項目有：no bias, 1/2bias, 1/3 bias, 1/4 bias, 1/5 bias。

任何一個 COM 腳位上所送出的重複波形的頻率都可以代表 LCD frame frequency，每一種 LCD lighting system 都會有不同的 LCD frame frequency，這個頻率可以利用 LCD 的 alternating clock frequency 以及 duty cycle 計算出來，使用者可以利用 code option 來選擇 pre-divider 的輸出信號(PH3~PH10)作為 LCD alternating clock source ，另可利用 code option 來選擇是否將 LCD alternating clock source 先除 3(此模式下 LCD alternating clock source 最高可擴充至 PH1)之後再輸出 LCD alternating clock。

下圖說明 frame frequency 與 alternating clock frequency 之間的波型關係：



所有的 LCD frame frequency 都是在 PH0 的頻率是 32768 Hz 的基準下所計算出來的，計算公式如下：

LCD frame frequency = LCD alternating clock frequency (x 1 or 3 by code option) x LCD duty cycle

舉例而言，如果期望的 LCD frame frequency 是 51 Hz，而且是使用 1/5 duty cycle，LCD alternating clock =LCD alternating clock source，這樣 LCD alternating clock frequency 的計算公式如下所示：

51 Hz (frame frequency)=alternating clock frequency x 1 x 1/5 duty

所以 alternating clock frequency=255 Hz，這個頻率與 pre-divider 的輸出信號中最接近的是 PH7 (256 Hz)。

下列表說明 LCD frame frequency(中間的數字(Hz))與 LCD duty cycle 以及 predivider 的輸出信號(alternating clock)之間的關係(PH0=32768Hz)

1. LCD alternating clock = LCD alternating clock source：

DUTY	PH3	PH4	PH5	PH6	PH7	PH8	PH9	PH10
1/1	4096	2048	1024	512	256	128	64 f	32 t,s
1/2	2048	1024	512	256	128	64 f	32 t	16 s
1/3	1365	683	341	171	85 f	43 t	21 s	11
1/4	1024	512	256	128	64 f	32 t	16 s	8
1/5	819	410	205	102 f	51 t	26 s	13	6
1/6	683	341	171	85 f	43 t	21 s	11	5
1/7	585	293	146	73 f	37 t	18 s	9	5
1/8	512	256	128 f	64 t	32 s	16	8	4
1/9	455	228	114 f	57 t	28 s	14	7	4
1/10	410	205	102	51	26	13	6	3
1/11	372	186	93	47	23	12	6	3
1/12	341	171	85	43	21	11	5	3
1/13	315	158	79	39	20	10	5	2
1/14	293	146	73	37	18	9	5	2
1/15	273	137	68	34	17	9	4	2
1/16	256	128	64	32	16	8	4	2

對應 code option “LCD FRAME FREQUENCY”:

(1) SLOW : s

(2) TYPICAL : t

(3) FAST : f

(4) O/P : 0Hz(LCD not used)

2. LCD alternating clock = LCD alternating clock source x 2/3：

DUTY	PH3	PH4	PH5	PH6	PH7	PH8	PH9	PH10
1/1							43 f	21 t,s
1/2						43 f	21 t	11 s
1/3					57 f	28 t	14 s	
1/4					43 f	21 t	11 s	

1/5				68 f	34 t	17 s		
1/6				57 f	28 t	14 s		
1/7				49 f	24 t	12 s		
1/8			85 f	43 t	21 s			
1/9			76 f	38 t	19 s			

3. LCD alternating clock = LCD alternating clock source x 4/3：

DUTY	PH3	PH4	PH5	PH6	PH7	PH8	PH9	PH10
1/1							85 f	43 t,s
1/2						85 f	43 t	21 s
1/3					114 f	57 t	28 s	
1/4					85 f	43 t	21 s	
1/5				137 f	68 t	34 s		
1/6				114 f	57 t	28 s		
1/7				98 f	49 t	24 s		
1/8			171 f	85 t	43 s			
1/9			152 f	76 t	38 s			

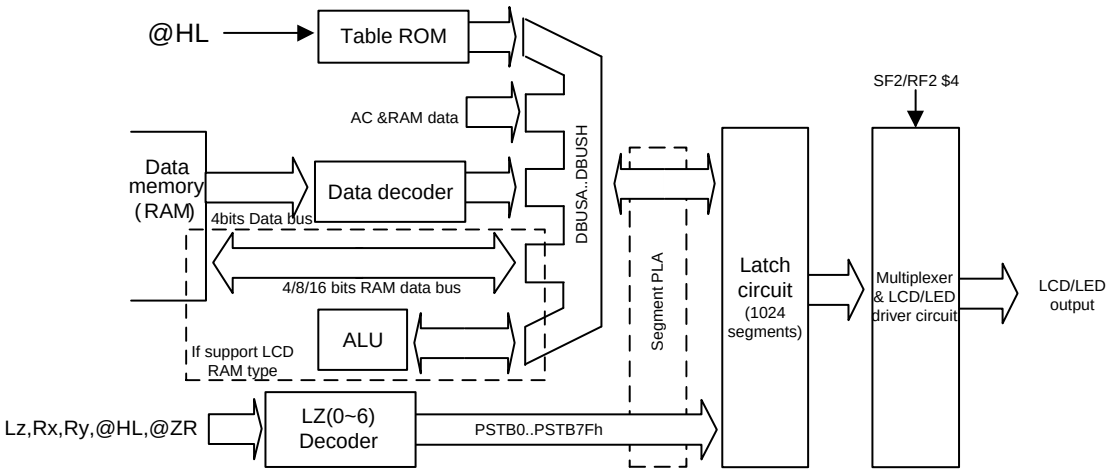
4. LCD alternating clock = LCD alternating clock source x 1/3 for TM87ML28 ：

DUTY	PH3	PH4	PH5	PH6	PH7	PH8	PH9	PH10
1/1	1365	683	341	171	85	43	21	11
1/2	683	341	171	85	43	21	11	5
1/3	455	228	114	57	28	14	7	4
1/4	341	171	85	43	21	11	5	3
1/5	273	137	68	34	17	9	4	2
1/6	228	114	57	28	14	7	4	2
1/7	195	98	49	24	12	6	3	2
1/8	171	85	43	21	11	5	3	1
1/9	152	76	38	19	9	5	2	1

選擇 LCD frame frequency 時請儘量選擇高於 24 Hz 的頻率，否則很容易發現 LCD 的螢幕顯示會有閃爍的現象。

3-2. LCD DISPLAY MEMORY

4BIT 系列 LCD display memory 資料流程說明如下:



1. Segnment PLA 為設定每個 COM(DCoutput 或是 P open-drain output)以及 SEG 在每個 duty 的輸出 latch 所對應的 Lz 位址(PSTB0~7Fh)以及七節碼資料(DBUSA~H) ，若 MCU 如一般 TM87 系列提供 code option 彈性設定的功能，則使用者可在編輯器上依顯示玻璃規劃自行設定。若 MCU 如一般 TM89 系列為固定 PLA 則在 Data Sheet 上會列表出與 COM 以及 SEG 的對應關係。

   當 latch circuit 中的 latchs 經 PLA 所設定 PSTB 與指令送出的 Lz 位址吻合時，便會開啟這些 latchs 並依 PLA 所設定 DBUS 更改輸出資料，再依最新 latch 輸出更改 LCD 輸出波形。

2. LCD display memory 定址可藉由 Lz 直接定址，若該 MCU 提供其可當 RAM 使用，則在 RAM 中規劃了 256 x 4 bit (0100h ~ 01FFh)做為 LCD display memory 使用，裡面儲存了 1024 個 LCD dot 的輸出信號的資料，故 MCU 亦可藉由 Rx,Ry,@HL,@ZR 等一般 RAM 定址方式。Lz&DBUS 與 RAM 位址&資料位元的對應關係如下表所示:

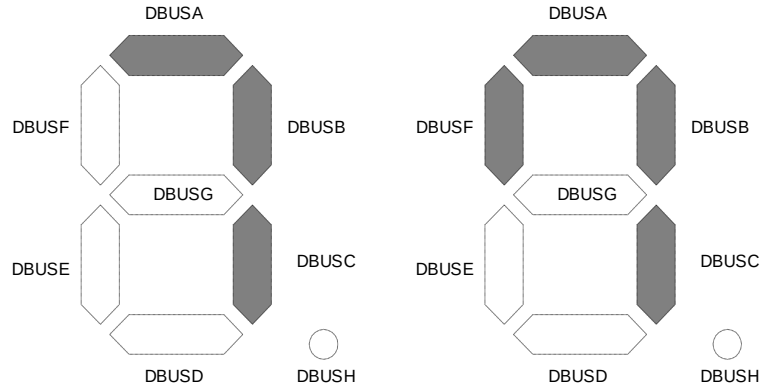
Lz	DBUS	Rx(BIT3~0)
00h	D~A	0100h
	H~E	0101h
01h	D~A	0102h
	H~E	0103h
7Eh	D~A	01FCh
	H~E	01FDh
7Fh	D~A	01FEh
	H~E	01FFh

3. 4BIT 系列提供 LCD 七節碼轉換功能，RAM data 經轉換後送至 DBUSA~H 輸出如下表所示:

Content of data memory	Output of data decoder							
	DBUSA	DBUSB	DBUSC	DBUSD	DBUSE	DBUSF	DBUSG	DBUSH
0(LCT(X))	1	1	1	1	1	1	0	1
0(LCB(X))	0	0	0	0	0	0	0	0
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	*note	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

* Note：DBUSF 輸出值可由 code option 選擇為 0 或是 1，玻璃顯示七節碼差別如下圖所示:



DBUSF=0 DBUSF=1

如表所示當執行 LCT(X)指令時，若輸入值為 0，則轉換後結果僅 DBUSG=0，故會顯示 0 的圖形，但若執行執行 LCB(X)指令時，則轉換後結果 DBUSA~H 皆為 0，故會如輸入值為 A~F 時同樣無任何顯示。

在直接 Lz 定址模式下，尚提供其他未經七節碼轉換的指令模式，輸出資料位元對應至 DBUSA~H 如下表所示:

	DBUSA	**DBUSB**	**DBUSC**	**DBUSD**	**DBUSE**	**DBUSF**	**DBUSG**	**DBUSH**
LCP(X)	Ry0	Ry1	Ry2	Ry3	AC0	AC1	AC2	AC3
LCD(X)	T@HL0	T@HL1	T@HL2	T@HL3	T@HL4	T@HL5	T@HL6	T@HL7
LCE(#) Lz,@HL	@HL(8)0	@HL(8)1	@HL(8)2	@HL(8)3	@HL(8)4	@HL(8)5	@HL(8)6	@HL(8)7
LCE(#) Lz,@ZR	@ZR(8)0	@ZR(8)1	@ZR(8)2	@ZR(8)3	@ZR(8)4	@ZR(8)5	@ZR(8)6	@ZR(8)7

(1) LCP(X)指令：將 AC & Ry 輸出合併送至 DBUSA~H。

(2) LCD(X)指令：將 Table ROM 8 bits 輸出送至 DBUSA~H。

(3) 若 MCU 提供 RAM 8bits data type ， 則可提供 LCE 指令，藉由@HL 或者@ZR 定址將 RAM 8bits 輸出送至 DBUSA~H。

若 LCD 有一致輸出需求，MCU 可藉由 LCTX,LCBX,LCPX,LCDX 指令在一個指令週期更改 16 Lz 位址(X0~XFh)資料狀態，以藉此加速更改速度。

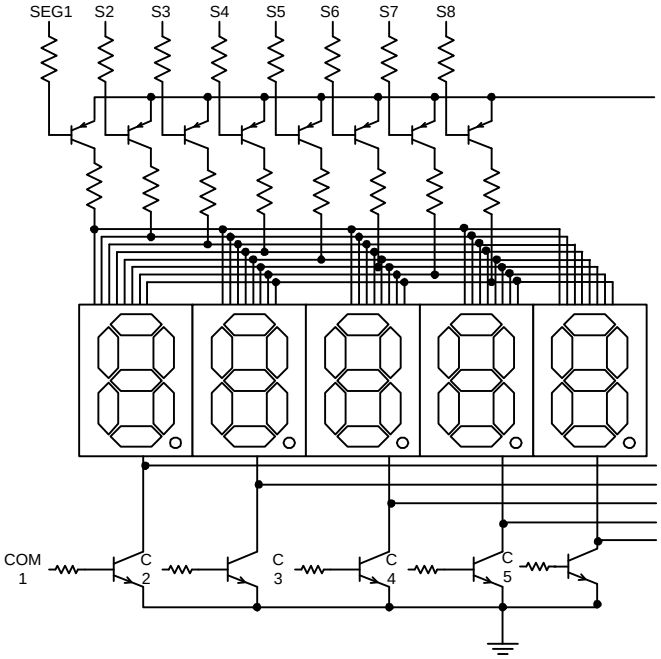
一般 LCD 指令只能單向將資料輸出至 LCD Driver，但若 MCU 提供將 LCD display memory 當 RAM 使用，則可藉由一般 RAM 存取指令直接對 LCD Driver 輸出資料直接處理，不需另外佔用其他一般 RAM 的位置間接處理完後再經由 LCD 指令更改 LCD Driver 的輸出。

如圖所示，執行 SF2 $4 指令時並不會直接更動 latch circuit 輸出值，只是將所有 LCD drvier 顯示關閉而已。有些 TM87 系列早期 MCU 只是單純輸出 OFF 波形，後期 MCU 則是已將波型完全停止以大幅節省耗電。

5. 4BIT 系列部份 MCU 亦提供兩種 LED Driver 模式，外接電路如下所示：

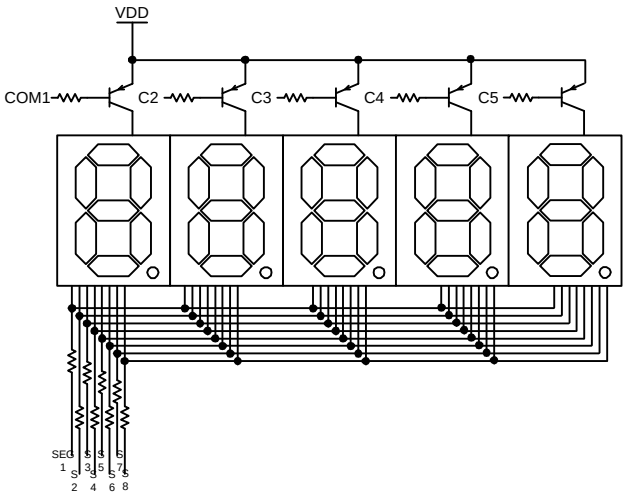
(1) High Active Mode

Mask Option name	Selected item
LCD/LED ACTIVE TYPE	(2) LED HIGH ACTIVE



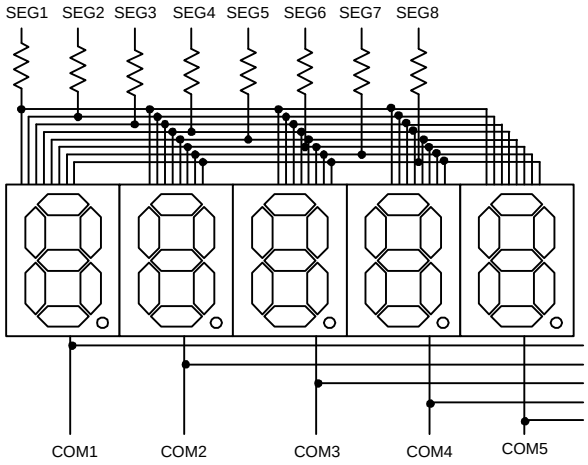
(2) Low Active Mode

Mask Option name	Selected item
LCD/LED ACTIVE TYPE	(3) LED LOW ACTIVE



Note：此模式雖然可以省去 SEG 的電晶體元件，但是因 SEG 直接承受 LED turn on 電流，故建議對應同一個 COM 的 SEG 數不要超過 8 個。

6. 4BIT 系列部份 MCU 亦為共陰極 LED 面板提供直接 LED Driver 模式以節省電晶體元件成本(實際仍視應用對 LED 亮度要求是否需更大電流驅動能力而定，但整體耗電請勿超過 40mA 以免使 IC 有燒毀風險)，外接電路如下所示：



3-3. COM以及SEG pin作為一般的OUTPUT PIN

有一些 COM 輸出腳位因 LCD duty cycle 的緣故而不需當成 LCD driver 使用的情形下，可以利用 code option 設定成 CMOS type DC output 或是 P open-drain output 輸出腳位，在這個情況下並不會影響其他作為 LCD driver 的 COM 輸出腳位的功能。

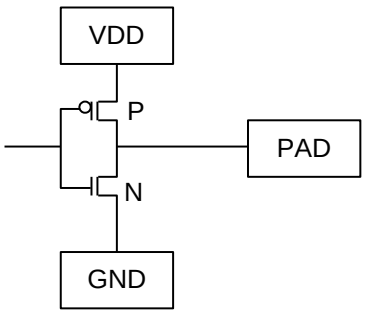
SEG 輸出腳位則只要 MCU 該腳位有提供 SEG typ DC output 或是 P open-drain output 輸出腳位 code option 便可依需求自由設定，在這個情況下並不會影響其他作為 LCD driver 的 SEG 輸出腳位的功能。

這些 output pin 的輸出資料都儲存在 LCD display memory 中幾個相對應的位元中，如果要改變這些 output pin 的輸出信號時，只需將對應的 LCD display memory 的位元資料改變即可。若如 TM87 系列為 PLA write only type MCU 需依照 LCD configuration file(.cfg)中自行所設定 Lz 位址以相關指令改變之外, 另外如 TM89 系列為固定的 LCD RAM read/write type 則依 Data Sheet 所定義對應 RAM 位址進行存取，此外仍保留 Lz 相關指令可依對應 LCD RAM 位址進行更改。

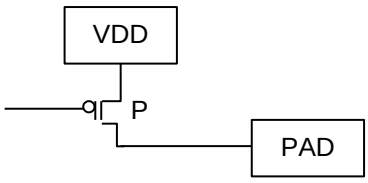
在 3-2 小節的以 TM8959 為例對應表中說明：COM5 ~ COM8 的 output pin 對應資料儲存在 data RAM 的 017Fh 位址上；COM9 ~ COM12 的 output pin 對應資料儲存在 data RAM 的 01BFh 位址上；COM13 ~ COM16 的 output pin 對應資料儲存在 data RAM 的 01FFh 位址上。

當 COM 以及 SEG 輸出腳位被設定成為 output 的功能之後，其輸出信號不會因為程式進入 STOP mode 而有所改變，也不會被執行 SF2 4h 指令後關閉 LCD 畫面顯示的功能所影響。

CMOS output type 以及 P open-drain output type 的結構如下所示：



CMOS Output Type



P Open-Drain Output Type

只能將沒有使用的 COM 腳位設定成 output pin，而且在選擇 COM 腳位作為 output pin 時 必須維持 LCD COM 輸出腳位的順序，不可打斷其順序。

例如，在 LCD 使用 1/5 duty cycle 的時候必須保留 COM1~COM5 作為 LCD driver，COM6~COM16 才能做為 DC output 使用。不可將 COM1~COM4 以及 COM6 作為 LCD driver，而將 COM5 設定成 DC output 。

3-4. COM以及SEG腳位上輸出Non-Overlap信號的功能

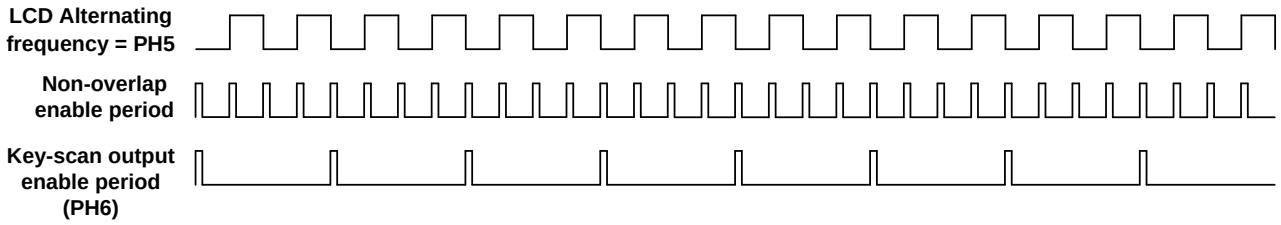
當 MCU 由 code option 選擇使用 Key-scan 功能時，COM 以及 SEG 所輸出的 LCD 波形中會產生 non-overlap 信號以供掃描進行，但未選擇使用 Key-scan 功能的 code option 時有些 MCU 仍可以依 MCU&應用差異對耗電，干擾等實際影響優劣，利用 code option 選擇"ALL HI-Z" 或是"ALL NO USE" 決定是否要加入 non-overlap 的信號。除 TM872X 系列基本上選擇"ALL HI-Z"會比較省電之外，其餘 MCU 基本上選擇"ALL NO USE"會比較省電。此外也可以依應用需求利用 code option 設定 non-overlap 信號的寬度調整對 LCD 顯示&耗電等影響。

Non-overlap 的信號會在特定的時間周其伴隨著 COM 以及 SEG 腳位上的 LCD 波形一起輸出，原則上 non-overlap 的信號會在 LCD alternating clock 的上升緣以及下降緣的時間　輸出，但是在幾個特定的情況下會有些不同，*請參考 3-4-1* 小節詳細說明。

3-4-1. Non-overlap 信號作為 key-matrix 的掃描信號

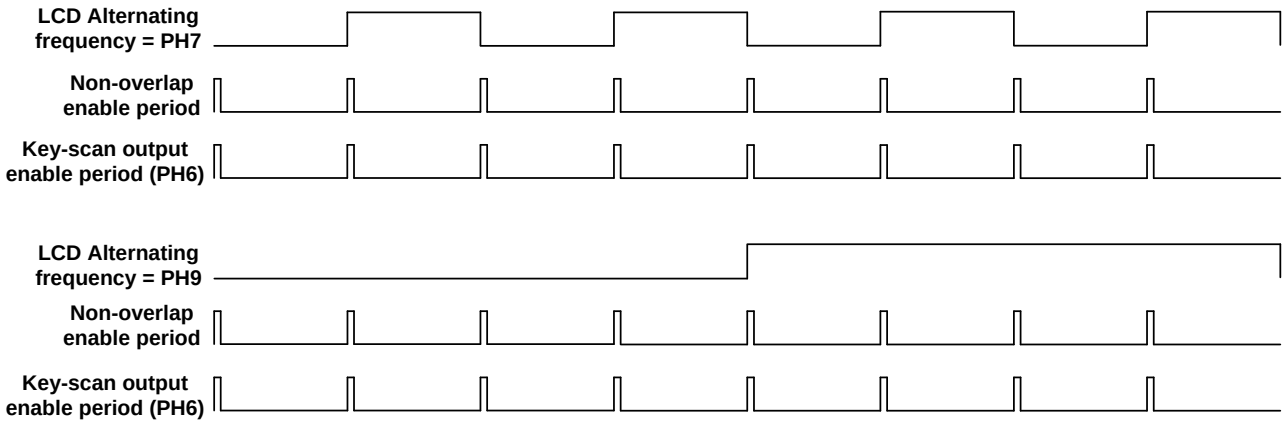
當 MCU 在使用 key matrix 掃描功能的時候必須利用 code option 將 non-overlap 信號的輸出功能選擇成 "all use" 或是"only COM,SEG1~16 use"。

4BIT 系列目前掃描週期分兩種模式，一種如 TM87 系列 non-overlap 信號與掃描信號的輸出皆隨 LCD alternating clock 的頻率而改變，另一種則如 TM89 系列因掃描信號的輸出週期固定為 PH6,故如果 LCD alternating clock 的頻率高於 PH6 時，所有的 COM 以及 SEG 腳位上的 non-overlap 信號則會在 LCD alternating clock 的上升緣以及下降緣的時候將 LCD 的輸出 信號改成 Hi-Z 的輸出，但是 SEG1~SEG16 會在 PH6 的下降緣將 non-overlap 信號改成　掃描信號的輸出。請參考下圖之說明：



如果 LCD alternating clock 的頻率低於 PH6 時，所有的 COM 以及 SEG 腳位上的 non-overlap 信號都會固定以 PH6 的週期將 LCD 的輸出信號改成 Hi-Z 的輸出，而 SEG1~SEG16 會在 PH6 的週期時將 non-overlap 信號改成掃描信號的輸出。

因此，一旦 LCD alternating clock 頻率大於等於 PH6 的頻率時，non-overlap 的信號就會 固定在 PH6 的週期輸出，而不會跟著 LCD alternating clock 的頻率而改變。請參考下圖之說明：



在 MCU 進入 RESET 狀態之後，non-overlap 信號的輸出功能就會開始在 SEG1 ~ SEG16 腳位上輸出有效位準(LCD/LED 為 GND/VBAT)的掃描信號。一旦程式執行 SPKTH、SPKXH、SPKRH、PK 等指令之後，只有這些指令所指定的 SEG 腳位上的 non-overlap 信號才會輸出有效位準的掃描信號，其餘未被指定的 SEG 腳位上的 non-overlap 信號就只會輸出 Hi-Z 的狀態。

3-4-2. Non-Overlap 信號長度的設定

當 non-overlap 信號作為 key-matrix 掃描功能使用時，MCU 可以利用另一個 code option 來設定 non-overlap 信號的時間長度，一種是 PH0 的信號週期長度，另一種是 PH1 的信號週期長度。

請注意，當 non-overlap 信號是作為 key matrix 掃描信號使用時，選擇 PH0 的時間長度可減輕 LCD 在顯示畫面時 SEG1~16 與其它 segment pin 在亮度上的差異，但是如果因為外部元件的 RC 負載過大，而導致 MCU 無法正確讀取到掃描信號時，建議改用 PH1 的時間長度。