SN8P1603 系列 8-BIT 微處理器

Note : The content of this document is subject to be changed without notice. Please confirm that is the latest version before using this document.

第一章. 概論

1. 緒論：

SN8P1603 是一系列 8 位元微處理器，利用 CMOS 技術製程，擁有獨特的電子結構，使其有低耗電及高效能的特色。此晶片採用一流的 IC 結構所設計，程式記憶體高達 1024*16 bits OTP ROM，48*8 bits 資料記憶體，一個 8-bit 計時/事件計數器 (TC0) ，看門狗計時器(watchdog timer)，2 個中斷源(TC0，INT0) ，14 個輸入/輸出(I/O)腳位，4 層堆疊緩衝區(stack buffer)，除此之外，使用者可自行選擇微處理器的振盪器型式，SN8P1603 有 4 種振盪器型式供使用者選擇，產生系統時脈，3 種外部振盪器型式，包含高效能的石英振盪器、陶瓷震盪器、廉價的 RC 振盪器，及 1 組內部 RC 振盪器。

2. 特性：

- ***記憶體結構***
  OTP ROM 容量：1024 * 16 bits
  RAM 容量：48 * 8 bits

- ***I/O 端點型態(總共 14 個腳位)***
  1 個輸入腳位，有中斷功能
  雙向輸入/輸出埠：13 個腳位
  5 個腳位具有系統喚醒功能

- ***內建電壓檢測器***
  永遠致能，保護電源緩慢上升、電源快速上升以及電源電壓不足等情況之下系統重置成功。

- ***56 個功能強大指令***
  所有的指令皆爲 16 bits 格式，執行時間爲 1 或 2 個執行週期。
  執行時間：1 個指令週期爲 4 個振盪時脈。
  JMP 指令(無 ROM 範圍限制)。
  CALL 指令(無 ROM 範圍限制)。
  表格查詢功能，**MOVC** 指令(無 ROM 範圍限制)。

- ***2 個中斷源***
  1 個內部中斷：TC0
  1 個外部中斷：INT0

- ***4 層堆疊緩衝區***
- ***1 個 8-bit 計時/事件計數器***

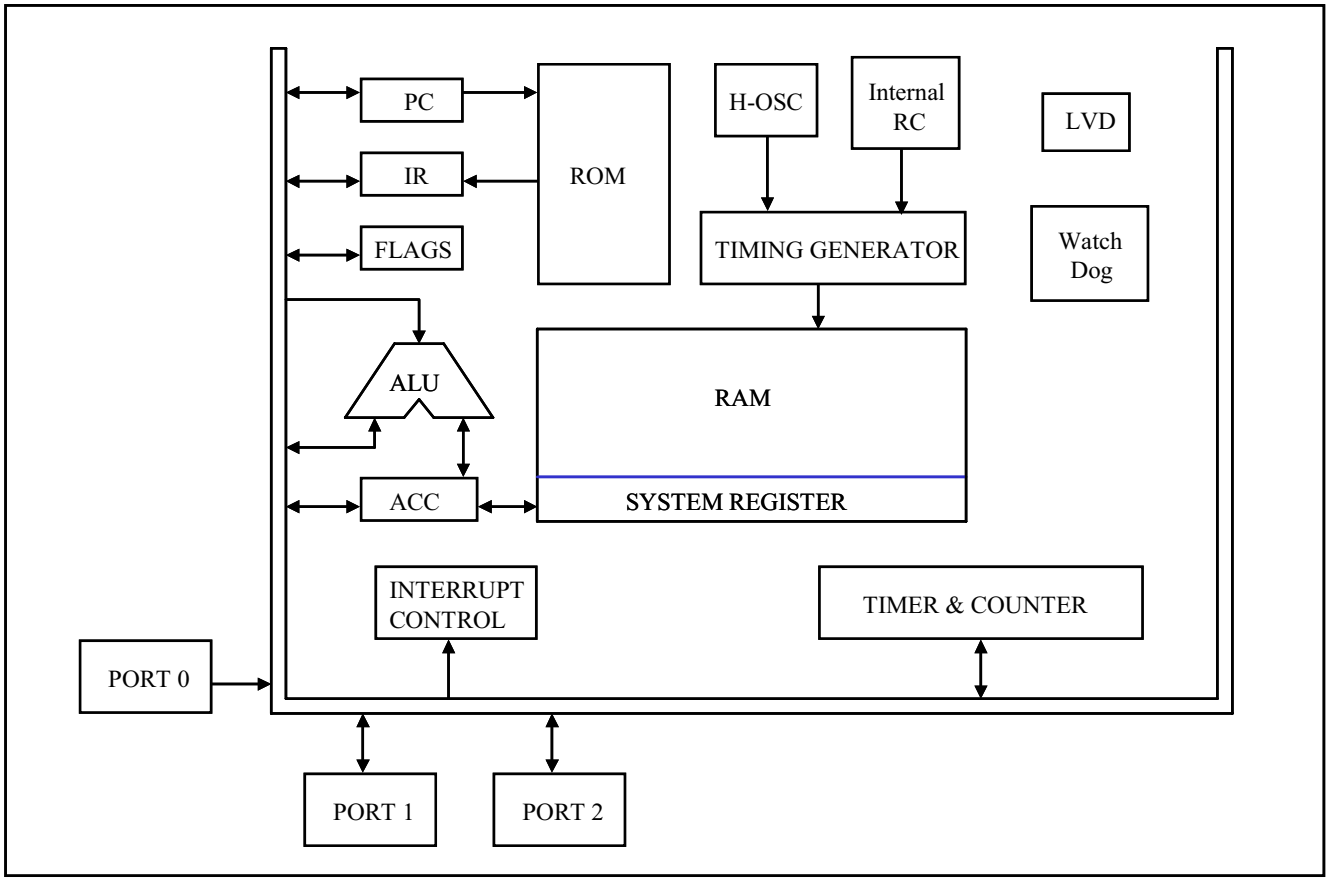
- ***1 個看門狗計時器***

- ***可接受的振盪器型式***
  石英/陶瓷振盪器：速度最高爲 20MHz
  RC 振盪器：速度最高爲 10MHz
  內部 RC 振盪器：16KHz

- ***包裝型式***

  PDIP：18 pin
  SOP：18 pin
  SSOP：18 pin

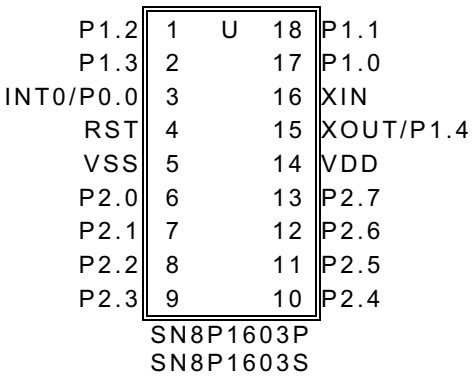
3. 系統方塊圖：



4. 腳位配置：

編號格式：SN8P1603Y

Y = P > PDIP，S > SOP



5. 腳位說明：

腳 位 名 稱	形態	說明
VDD，VSS	P	電源輸入腳位。
RST	I	系統重置電路輸入腳位，史密特觸發結構，動作 ‘0’ ，一般狀態爲 ‘1’ 。
XIN	I	振盪器輸入腳位。
XOUT/P1.4	I/O	振盪器輸出腳位，RC 模式爲 P1.4 輸入/輸出腳位。
P0.0/INT0	I	P0.0 與 INT0 觸發腳位，有系統喚醒功能(史密特觸發結構)。
P1.0~P1.4	I/O	P1.0~P1.4 雙向腳位，具有系統喚醒功能。
P2.0~P2.7	I/O	P2.0~P2.7 雙向腳位。

第二章. 位址 (ADDRESS)

1. 程式記憶體 (ROM)：

SN8P1603 提供可定址的程式記憶體高達 1024*16bits，及透過 10-bit 程式計數器(PC)取回指令，也可利用 ROM CODE 暫存器(Y、Z)去查閱 ROM 資料，儲存於 R、A 暫存器。所有的程式記憶體被分割爲 2 個編碼區，從 000H 至 00F 用來執行中斷引導(特別區)。第二區，從 010H 至 3FFH 用來儲存指令程式碼及查尋表格資料，OTP ROM 最後一個位址 ‘3FFH’ 已被保留，作爲選項使用，無法由程式控制使用。

	OTP ROM
000h	系統重置引導
001h	
002h	
003h	
004h	保留區
005h	‘’
006h	‘’
007h	‘’
008h	中斷引導
009h	.
00Ah	.
.	.
.	.
010h	一般用途區
.	.
.	.
.	.
.	.
.	.
.	.
3FEh	.
3FFh	保留區

2. 資料記憶體 (RAM)：

SN8P1603 內建 48*8 bits 的記憶體空間，作爲儲存一般用途資料，以及內建特殊用途記憶體，作爲系統暫存器，這些記憶體位於 RAM bank0，前端 48*8 bits(00H~2FH)分配給一般資料記憶體，最後的 128*8 bits(80H~FFH)分配爲系統暫存器。

2.1. RAM BANK 位置：

	RAM 位置
00H	一般用途區
2FH	RAM 位址結束
80H	系統暫存器
	‘’
	‘’
	‘’
FFH	‘’

2.2. 系統暫存器配置表 (BANK 0)：

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
8	-	-	R	Z	Y	-	PFLAG	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
C	P1W	P1M	P2M	-	-	-	-	-	INTRQ	INTEN	OSCM	-	-	-	PCL	PCH
D	P0	P1	P2	-	-	-	-	-	-	-	TC0M	TC0C	-	-	-	STKP
E	-	-	-	-	-	-	-	@YZ	-	-	-	-	-	-	-	-
F	-	-	-	-	-	-	-	-	STK3	STK3	STK2	STK2	STK1	STK1	STK0	STK0

PFLAG= ROM page、特殊旗標暫存器。

R= 工作暫存器及 ROM 資料查詢緩衝區。

Y，Z= 工作、@YZ 及 ROM 定址暫存器。

P0~P2= P0 至 P2 資料緩衝區。

P1M~P2M= P1、P2 輸入/ 輸出模式暫存器。

INTRQ= 中斷需求暫存器。

INTEN= 中斷致能暫存器。

OSCM= 振盪器模式暫存器。

PCH，PCL= 程式計數器。

TC0M= 計時/事件 計數器 0 模式暫存器。

TC0C= 計時/事件 計數器 0 計數暫存器。

STKP= 堆疊指標緩衝區。

STK0~STK3= 堆疊 0~堆疊 3 緩衝區。

@YZ= RAM YZ 間接定址索引指標。

2.3. 系統暫存器表格：

位址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	備註
080H	-	-	-	-	-	-	-	-	-	-
081H	-	-	-	-	-	-	-	-	-	-
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
085H										
086H	-	-	-	-	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	-	-	-	-	-	-
0C0H	-	-	-	P14W	P13W	P12W	P11W	P10W	R/W	P1W 喚醒暫存器
0C1H	-	-	-	P14M	P13M	P12M	P11M	P10M	R/W	P1M I/O 指示
0C2H	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M	R/W	P2M I/O 指示
0C3H	-	-	-	-	-	-	-	-	-	-
0C4H	-	-	-	-	-	-	-	-	-	-
0C5H	-	-	-	-	-	-	-	-	-	-
0C6H	-	-	-	-	-	-	-	-	-	-

0C7H	-	-	-	-	-	-	-	-	-	-
0C8H	-	-	TC0IRQ	-	-	-	-	P00IRQ	R/W	INTRQ
0C9H	-	-	TC0IEN	-	-	-	-	P00IEN	R/W	INTEN
0CAH	WTCKS	WDRST	-	-	CPUM0	CLKMD	STPHX	-	R/W	OSCM
0CBH	-	-	-	-	-	-	-	-	-	-
0CCH	-	-	-	-	-	-	-	-	-	-
0CDH	-	-	-	-	-	-	-	-	-	-
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH	-	-	-	-	-	-	PC9	PC8	R/W	PCH
0D0H	-	-	-	-	-	-	-	P00	R/W	P0 資料緩衝
0D1H	-	-	-	P14	P13	P12	P11	P10	R/W	P1 資料緩衝
0D2H	P27	P26	P25	P24	P23	P22	P21	P20	R/W	P2 資料緩衝
0D3H	-	-	-	-	-	-	-	-	-	-
0D4H	-	-	-	-	-	-	-	-	-	-
0D5H	-	-	-	-	-	-	-	-	-	-
0D6H	-	-	-	-	-	-	-	-	-	-
0D7H	-	-	-	-	-	-	-	-	-	-
0D8H	-	-	-	-	-	-	-	-	-	-
0D9H	-	-	-	-	-	-	-	-	-	-
0DAH	TC0ENB	TC0RATE2	TC0RATE1	TC0RATE0	TC0CKS	-	-	-	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	-	-	-	-	-	-	-	-	-	-
0DDH	-	-	-	-	-	-	-	-	-	-
0DEH	-	-	-	-	-	-	-	-	-	-
0DFH	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0	R/W	STKP 堆疊指示
0E0H	-	-	-	-	-	-	-	-	-	-
0E1H	-	-	-	-	-	-	-	-	-	-
0E2H	-	-	-	-	-	-	-	-	-	-
0E3H	-	-	-	-	-	-	-	-	-	-
0E4H	-	-	-	-	-	-	-	-	-	-
0E5H	-	-	-	-	-	-	-	-	-	-
0E6H	-	-	-	-	-	-	-	-	-	-
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ 索引指示
0F0H										
0F1H										
0F2H										
0F3H										
0F4H										
0F5H										
0F6H										
0F7H										
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R	STK3L
0F9H	-	-	-	-	-	-	S3PC9	S3PC8	R	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R	STK2L
0FBH	-	-	-	-	-	-	S2PC9	S2PC8	R	STK2H

0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R	STK1L
0FDH	-	-	-	-	-	-	S1PC9	S1PC8	R	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R	STK0L
0FFH	-	-	-	-	-	-	S0PC9	S0PC8	R	STK0H

注意：
a) 所有暫存器名稱都已經在 SN8ASM 組合語言中宣告。
b) 暫存器的單一位元名稱在 SN8ASM 組合語言中已經以 ‘F’ 字首宣告，所以程式編寫時，須在名稱之前加 ‘f’ 宣告之。

例如：PFLAG 中的 C 宣告爲 FC，DC 宣告爲 FDC，Z 宣告爲 FZ。
OSCM 中的 WTCKS 宣告爲 FWTCKS，WDRST 宣告爲 FWDRST，CPUM0 宣告爲 FCPUM0...。

3. 累積器 (ACC)：

ACC(程式中稱爲 A)是一個 8-bit 的資料暫存器，負責在 ALU 及資料記憶體間傳輸及運用資料，若運算結果 ACC 爲 0(Z)、溢位(C)或借位(DC)發生，這些旗標將會被設定在 PFLAG 暫存器中。

PFLAG 初值 = xxxx xxxx

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	-	-	-	-	-	C	DC	Z

3.1. 溢位旗標：

C = 1：若執行算術加法產生溢位信號，或執行算術減法沒有產生借位信號，或執行旋轉指令後，搬移出的邏輯是 ‘1’ 。

C = 0：若執行算術加法沒有發生溢位信號，或執行算術減法產生借位信號，或執行旋轉指令後，搬移出的邏輯是 ‘0’ 。

3.2. 小數溢位旗標：

DC = 1：若執行算術加法時，從低半字節產生信號，或執行算術減法時，沒有從高半字節產生借位號。

DC = 0：若執行算術加法時，沒有從低半字節產生信號，或執行算術減法時，從高半字節產生借位信號。

3.3. 零旗標：

Z = 1：運算後，ACC 等於零。
Z = 0：運算後，ACC 不等於零。

3.4. 範例：

3.4.1. 加法運算無溢位：

```
MOV     A,#00001100b
B0MOV   BUF0,A              ; BUF0=00001100b
MOV     A,#00001111b        ; A=00001111b
ADD     A,BUF0              ; A=A+BUF0

結果 :  A=00011011b         ; 無溢位產生
        C=0
        DC=1
        Z=0                 ; A 不爲 0,Z=0
```

3.4.2. 加法運算有溢位：

```
MOV     A,#11111110b
B0MOV   BUF0,A              ; BUF0=11111110b
MOV     A,#00001111b        ; A=00001111b
ADD     A,BUF0              ; A=A+BUF0

結果 :  A=00001101b         ; 溢位產生
        C=1
        DC=1
        Z=0                 ; A 不爲 0,Z=0
```

3.4.3. 加法運算結果 0：

```
MOV     A,#00000000b
B0MOV   BUF0,A              ; BUF0=00000000b
MOV     A,#00000000b        ; A=00000000b
ADD     A,BUF0              ; A=A+BUF0

結果 :  A=00000000b         ; 無溢位產生,結果爲 0
        C=0
        DC=0
        Z=1                 ; A=0,Z=1
```

3.4.4. 減法運算無借位：

```
MOV     A,#00001100b
B0MOV   BUF0,A              ; BUF0=00001100b
MOV     A,#00001111b        ; A=00001111b
SUB     A,BUF0              ; A=A-BUF0

結果 :  A=00000011b         ; 無借位產生
        C=1
        DC=1
        Z=0                 ; A 不爲 0,Z=0
```

3.4.5. 減法運算有借位：

```
MOV     A,#00001111
B0MOV   BUF0,A              ; BUF0=00001111b
MOV     A,#00001100b        ; A=00001100b
SUB     A,BUF0              ; A=A-BUF0

結果 :  A=11111101b         ; 借位產生
        C=0
        DC=0
        Z=0                 ; A 不爲 0,Z=0
```

3.4.6. 減法運算結果 0：

```
MOV     A,#00001100b
B0MOV   BUF0,A              ; BUF0=00001100b
MOV     A,#00001100b        ; A=00001100b
SUB     A,BUF0              ; A=A-BUF0

結果 :  A=00000000b         ; 無借位產生,結果爲 0
        C=1
        DC=1
        Z=1                 ; A=0,Z=1
```

3.4.7. 旋轉指令與 C 旗標關係：

以左旋轉爲範例說明：

```
B0BCLR   FC              ; 設 C 爲 0
或
B0BSET   FC              ; 設 C 爲 1

MOV      A,#10101010b
B0MOV    BUF0,A          ; BUF0=10101010b
RLC      BUF0            ; A=BUF0 左旋轉，C 初值設爲 0，A=01010100b
                         ; C 初值設爲 1, A=01010101b
                         ; C=1
B0MOV    BUF0,A          ; BUF0=A
RLC      BUF0            ; A=BUF0 左旋轉，C 初值設爲 0，A=10101001b
                         ; C 初值設爲 1，A=10101011b
                         ; C=0
```

3.4.8. 利用 C,DC,Z 作程序跳躍的巨集程式：

運算結束，C、DC、Z 數值隨運算結果變化，可以利用其數值作爲判斷條件，引導程式進入不同程序，執行不同功能，提供下列巨集，應用於此。

C=1，跳躍到 ADRS 位址：			C=0，跳躍到 ADRS 位址：		
JC	MACRO	ADRS	JNC	MACRO	ADRS
	B0BTS0	FC		B0BTS1	FC
	JMP	<ADRS>		JMP	<ADRS>
	ENDM			ENDM	

DC=1，跳躍到 ADRS 位址：			DC=0，跳躍到 ADRS 位址：		
JDC	MACRO	ADRS	JNDC	MACRO	ADRS
	B0BTS0	FDC		B0BTS1	FDC
	JMP	<ADRS>		JMP	<ADRS>
	ENDM			ENDM	

Z=1，跳躍到 ADRS 位址：			Z=0，跳躍到 ADRS 位址：		
JZ	MACRO	ADRS	JNZ	MACRO	ADRS
	B0BTS0	FZ		B0BTS1	FZ
	JMP	<ADRS>		JMP	<ADRS>
	ENDM			ENDM	

範例：

TJC：

```
        B0MOV   BUF0,A          ; BUF0=A
        RLC     BUF0            ; A=BUF0 左旋轉結果
        JC      TJC1            ; C=1，跳躍至 TJC1
        .       .               ; C=0
        .       .
TJC1：
        .       .
        .       .
```

TJNC：

```
        B0MOV   BUF0,A          ; BUF0=A
        RLC     BUF0            ; A=BUF0 左旋轉結果
        JNC     TJNC1           ; C=0，跳躍至 TJNC1
        .       .               ; C=1
        .       .
TJNC1：
        .       .
        .       .
```

TJZ：

```
        B0MOV   A,BUF0          ; BUF0=A
        JZ      TJZ1            ; A=0，跳躍至 TJZ1
        .       .               ; A 不爲 0
        .       .
TJZ1：
        .       .
        .       .
```

TJNZ：

```
        B0MOV   A,BUF0          ; BUF0=A
        JNZ     TJNZ1           ; A 不爲 0，跳躍至 TJNZ1
        .       .               ; A=0
        .       .
TJNZ1：
        .       .
        .       .
```

4. 工作暫存器：

資料記憶體中 RAM BANK0 82H 至 86H 位址，儲存特殊定義的暫存器(R、Y、Z 及 PFLAG 暫存器)，如下表所示。這些暫存器可當作一般用途的工作緩衝區，也可以用在存取 ROM 及 RAM 的資料，例如：所有的 ROM 表格皆可利用 R、Y 及 Z 暫存器，執行查詢表格的工作，而 RAM 記憶體資料也可利用 Y、Z 暫存器作間接存取。

RAM	80H	81H	82H	83H	84H	85H	86H	87H
	-	-	R	Z	Y	-	PFLAG	-

4.1. Y,Z 暫存器：

Y、Z 暫存器皆爲 8-bit 緩衝區，有三個主要功能，第一，Y、Z 暫存器可當做工作暫存器。第二，Y、Z 暫存器可當作@YZ 暫存器的資料指標。第三，ROM 位址的定址工作，此功能可應用於表格查詢。

例如：若欲讀取 RAM BANK0 20H 位址中的資料，可使用間接定址法存取資料，如下所示：

```
B0MOV   Y,#00H      ; 設定 RAM BANK0 至 Y 暫存器
B0MOV   Z,#20H      ; 設定位址 20H 至 Z 暫存器
B0MOV   A, @YZ      ; 利用資料指標@YZ 從 RAM 的 020H 位址讀一筆資
                    ; 料，儲存至 ACC 中
```

4.2. 查詢表格說明：

在查詢 ROM 資料功能中，Y 暫存器指向 ROM 位址中間的 8-bit 資料，而 Z 暫存器指向最低的 8-bit，在執行 MOVC 指令後，ROM 的低位元組資料儲存在 ACC，而高位元組資料儲存在 R 暫存器。

例如：從 Table_1 查詢 ROM 資料。

```
        B0MOV   Y, #TABLE1$M    ; 設定查詢表格的中間位址
        B0MOV   Z, #TABLE1$L    ; 設定查詢表格的低位址
        MOVC                    ; 查詢資料， R = 00H，ACC = 35H
        .                       ;
        INCMS   Z               ; 查詢下一個 ROM 的資料
        NOP                     ;
        MOVC                    ; 查詢資料，R = 51H，ACC = 05H
        .       .               ;
        .       .               ;
TABLE1: DW      0035H           ; 定義一個字元(16 bits)資料
        DW      5105H           ;
        DW      2012H           ;
        .       .
```

上述方式存取 ROM 資料時，無法跨越 Y 範圍(boundary)，需對 Y、Z 加以處理，利用以下巨集便可解決此問題：

```
INC_YZ    MACRO
          INCMS     Z           ; Z=Z+1
          JMP       @F          ; Z 無溢位產生

          INCMS     Y           ; Y=Y+1
          NOP                   ; 不理會 Y 有無溢位產生，直接離開
@@:
          ENDM
```

利用 INC_XYZ 巨集修正上述程式：

```
          B0MOV     Y, #TABLE1$M    ; 設定查詢表格的中間位址
          B0MOV     Z, #TABLE1$L    ; 設定查詢表格的低位址
          MOVC                      ; 查詢資料，  R = 00H，ACC = 35H
          .                         ;
          INC_YZ                    ; 查詢下一個  ROM 的資料
          MOVC                      ; 查詢資料，R = 51H，ACC = 05H
          .         .               ;
          .         .               ;
TABLE1：  DW        0035H           ; 定義一個字元(16 bits)資料
          DW        5105H           ;
          DW        2012H           ;
          .         .
```

4.3. 定址種類：

SN8P1603 提供三種定址模式存取 RAM 資料，包含立即定址模式、直接定址模式及間接定址模式，這三種不同定址模式的主要目的如下所述。立即定址模式是在 ACC 或特定 RAM 中，利用一個立即資料來設定 RAM 位址(MOV A,# I，B0MOV M,# I)；直接定址模式利用位址編碼存取記憶體位置(MOV A,12H，MOV 12H,A)；間接定址模式是在資料指標工作暫存器(Y/Z)設定一個位址，利用 MOV 指令在 ACC 與@YZ 暫存器之間讀/寫資料(B0MOV A,@YZ，B0MOV @YZ,A)。

4.3.1. 立即定址模式：

```
          MOV       A,#12H          ; 設置一個立即資料 12H 到 ACC
          B0MOV     R,#28H          ; 設置一個立即資料 28H 到 R 暫存器
```

4.3.2. 直接定址模式：

```
          MOV       A,12H           ; 取得一個在 BANK0 12H 位址的內容，儲存在 ACC
```

4.3.3. 間接定址模式含@YZ 暫存器：

```
CLR     Y           ；清除 Y 暫存器
B0MOV   Z,#16H      ；設置一個立即資料 16H 至 Z 暫存器
B0MOV   A, @YZ      ；利用資料指標@YZ 從 RAM 的 016H 位置讀一筆資
                    ；料，儲存至 ACC 中
```

5. 程式計數器：

程式計數器(PC)是一個 10-bit 的二進制計數器，被分割爲高位元組的 2-bit 於 PCH 及低位元組的 8-bit 於 PCL，如下表所示。PC 負責由核心電路取回指令時作位置指示的工作，在一般程式執行中，每個指令被執行後，程式計數器會自動加 1，除此之外，當執行 CALL 和 JMP 指令時，PC 同時會被特定位址所取代，CALL 及 JMP 指令執行時，目的位址被放置在 bit0 ~bit9。

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PCH	-	-	-	-	-	-	PC9	PC8

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PCL	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

PC 初值 = xxxx xx00 0000 0000

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0

5.1. 單一位址跳躍：

如果位元測試指令結果是成立的，PC 將加 2 階，跳躍至下一個指令。

```
        B0BTS1      FC          ；若溢位旗標(C)=1，跳過下一個指令
        JMP         C0STEP      ；否則跳至 C0STEP
        .
        .
C0STEP：NOP
        .
        .
```

5.2. 多數位址跳躍：

使用者要做多數位址跳躍動作，可以用 JMP 指令及 ADD M,A 指令(M = PCL)，使多數位址跳躍功能運作，若 ADD PCL,A 執行後有溢位信號發生，溢位信號將不會影響 PCH 暫存器，位址跳躍也不會到達下一個 PAGE 的 RAM 位址，所以必須對 PCH 及 PCL 加以運算處理，才能夠得到跨越 PAGE 的功能。

例：若 PC = 0323H (PCH = 03H，PCL = 23H)

```
; PC = 0323H
                    MOV         A,#28H
                    B0MOV       PCL,A       ; 跳躍至位址 0328H
                    .           .
                    .           .
; PC = 0328H        .           .
                    MOV         A,#00H
                    B0MOV       PCL,A       ; 跳躍至位址 0300H
```

例：若 PC = 0323H (PCH = 03H，PCL = 23H) ，此方式稱之爲 JUMP TABLE。

```
; PC = 0323H
                    B0ADD       PCL,A       ; PCL = PCL + ACC，PCH 不會被
                                            改變
                    JMP         A0POINT     ; 若 ACC = 0，跳躍至 A0POINT
                    JMP         A1POINT     ; 若 ACC = 1，跳躍至 A1POINT
                    JMP         A2POINT     ; 若 ACC = 2，跳躍至 A2POINT
                    JMP         A3POINT     ; 若 ACC = 3，跳躍至 A3POINT
                    .           .
                    .           .
```

如果有 8 個條件可供 JMP，而 ACC 數值並非一定介於 0~7 之間，必須將 ACC 的數值限制在 0~7 之間，若未限制 ACC 範圍，當 ACC 大於 7 時，程序便會跳至條件式跳躍以外的位址繼續執行，容易導致程序錯誤。修正上述程序如下：

```
AND         A,#7            ; 限制 ACC 數值於 0 至 7 之間
B0ADD       PCL,A           ; PCL = PCL + ACC，PCH 不會被改變

JMP         A0POINT         ; 若 ACC = 0，跳躍至 A0POINT
JMP         A1POINT         ; 若 ACC = 1，跳躍至 A1POINT
JMP         A2POINT         ; 若 ACC = 2，跳躍至 A2POINT
JMP         A3POINT         ; 若 ACC = 3，跳躍至 A3POINT
JMP         A4POINT         ; 若 ACC = 4，跳躍至 A4POINT
JMP         A5POINT         ; 若 ACC = 5，跳躍至 A5POINT
JMP         A6POINT         ; 若 ACC = 6，跳躍至 A6POINT
JMP         A7POINT         ; 若 ACC = 7，跳躍至 A7POINT
.           .               ; 非條件式跳躍的程序位址
.           .
```

由於 PCH 不會因 PCL 的溢位而自動加 1，所以不能跨 PAGE，請注意下述程序中的 PCH 數值：

```
2031    0000FA    2D16      MOV       A,#16H
2032    0000FB    2A07      AND       A,#7
2033    0000FC    03CE      B0ADD     PCL,A
2034    0000FD    8105      JMP       A0POINT     ; PCH=00H，PCL=FDH
2035    0000FE    8106      JMP       A1POINT     ; PCH=00H，PCL=FEH
2036    0000FF    8107      JMP       A2POINT     ; PCH=00H，PCL=FFH
2037    000100    8108      JMP       A3POINT     ; PCH=00H，PCL=00H
2038    000101    8109      JMP       A4POINT     ; PCH=00H，PCL=01H
2039    000102    810A      JMP       A5POINT     ; PCH=00H，PCL=02H
2040    000103    810B      JMP       A6POINT     ; PCH=00H，PCL=03H
2041    000104    810C      JMP       A7POINT     ; PCH=00H，PCL=04H
```

上述範例中，當 **A=3 時，PCL 由 FFH 加 1 成爲 00H，但 PCH 不會因爲 PCL 加 1 溢位成爲 00H 而加 1，所以 PCH 仍然爲 00H**。所以當在此應用時，必須判斷 PCL 是否即將溢位，若是，必須對 PCH 做加 1 動作，提供一段巨集程序做修正：

```
@JMP_A    MACRO   VAL                                          ; VAL 爲 JMP 階數
          IF      (($+1) !& 0XFF00) !!= (($+(VAL)) !& 0XFF00)  ; 判斷 PCL 是否溢位，PCH 是否加 1
          JMP     ($   |   0XFF)                               ; 跳躍至下一個 ROM PAGE
          ORG     ($   |   0XFF)
          ENDIF

          ADD     PCL, A                                       ; PCL=PCL+ACC
          ENDM
```

此巨集的功能是先計算 JMP TABLE 階數，若是會有跨 ROM 範圍情形出現，將 JMP TABLE 移至下一個 ROM PAGE，但是會有一個缺點，將會損失此巨集至 JMP TABLE 之間的 ROM 空間，當 JMP TABLE 範圍過大，損失的空間越多，例如：@JMP_A 位於 ROM 位址 00F0H，JMP TABLE 階數爲 20 階，所以執行 JMP TABLE 時，勢必遭遇跨越 ROM PAGE 情況，經由@JMP_A 巨集，運算 JMP TABLE 階數會導致跨越 ROM PAGE，將 JMP TABLE 移至 ROM 位址 0100H 開始執行，所以將損失 00F1H 至 00FFH 之間的 ROM，使用上需注意。

利用上述巨集程序修正之前的應用範例：

```
          B0MOV     A,BUF0      ; ACC=BUF0(0~6)
          @JMP_A    7           ; JMP 有 7 階
          JMP       A0POINT     ; 若 ACC = 0，跳躍至 A0POINT
          JMP       A1POINT     ; 若 ACC = 1，跳躍至 A1POINT
          JMP       A2POINT     ; 若 ACC = 2，跳躍至 A2POINT
          JMP       A3POINT     ; 若 ACC = 3，跳躍至 A3POINT
          JMP       A4POINT     ; 若 ACC = 4，跳躍至 A4POINT
          JMP       A5POINT     ; 若 ACC = 5，跳躍至 A5POINT
          JMP       A6POINT     ; 若 ACC = 6，跳躍至 A6POINT
          .           .
          .           .
```

在 JMP TABLE 的應用，建議事項如下：

- 將 JMP TABLE 移至程式前端 ROM 位址，修正 JMP TABLE 時較容易。
- 程式編寫完畢，經由 LISTING FILE 檢查 JMP TABLE 是否跨越 ROM PAGE，若有，修正之。

6. 堆疊緩衝區：

SN8P1603 的堆疊緩衝區可達 4 層，每一層有 10-bit 長度，此緩衝區設計爲執行中斷服務或呼叫副程式時，儲存 PC 值。STKP 暫存器是一個指標，設計爲母體電路從堆疊緩衝區推入(PUSH)或拉出(POP) PC 資料時，指示出動作階層。STKP 在資料推進堆疊緩衝區後，將會減少 1 層，從堆疊緩衝區取出資料前，會增加一層。一旦發生中斷，主中斷將把 STKP 的致能位元(GIE)由致能轉爲禁能，在 RETI 指令執行後，GIE 將再轉爲致能。

STKP 初值= 0XXX X111

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
STKP	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0

STKn 初值= XXXX XXXX XXXX XXXX, STKn = STKnH + STKnL (n = 7H ~ 4H)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
STKnH	-	-	-	-	-	-	SNPC9	SNPC8

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
STKnL	SNPC7	SNPC6	SNPC5	SNPC4	SNPC3	SNPC2	SNPC1	SNPC0

7. ACC 累積器與工作暫存器的保護：

SN8P1603 在執行中斷時，不會將 ACC 及工作暫存器內容推入堆疊緩衝區中儲存，因此一旦發生中斷，這些資料必須儲存在使用者所建立的資料記憶體中，如下所示：

在原始程式中宣告變數：

```
ACCBUF      EQU     00H         ；宣告 ACCBUF 在 BANK0 中的 00H 位址
PFLAGBUG    EQU     07H         ；宣告 PFLAGBUF 在 BANK0 中的 07H 位址
```

例如：推入(PUSH) ACC 及工作暫存器

```
PUSHBUF：
            B0XCH   ACCBUF,A    ；儲存 ACC 至 ACCBUF 中
            B0MOV   A,PFLAG     ；
            B0MOV   PFLAGBUF,A  ；儲存 PFLAG 至 PFLAGBUF 中
```

例如：拉出(POP) ACC 及工作暫存器

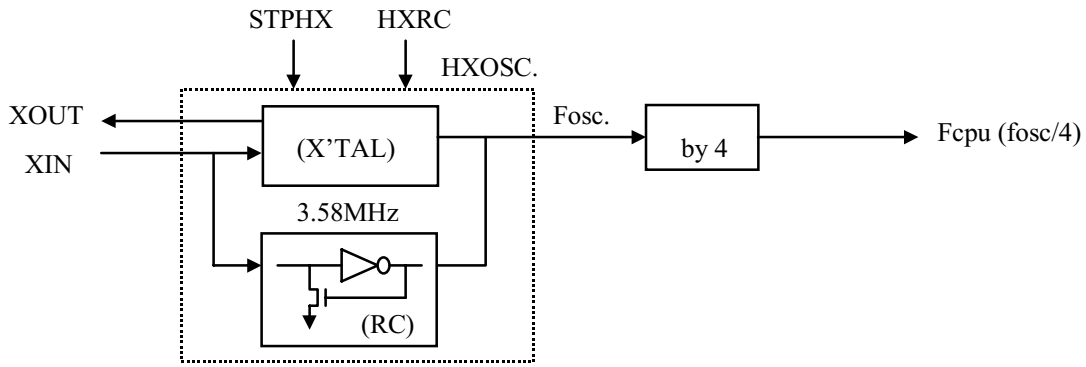
```
POPBUF：
            B0MOV   A,PFLAGBUF  ；
            B0MOV   PFLAG,A     ；還原 PFLAG
            B0XCH   A,ACCBUF    ；還原 ACC
```

注意：PUSH 及 POP 有先進後出的順序，確保數值正確。

第三章. 振盪電路

1. 振盪器：

SN8P1603 提供內部及外部兩種振盪器，有 RC 振盪電路、晶體振盪器、陶製振盪器及內部時脈等方式產生系統時脈源，使用者可選擇其中一種適當的振盪器架構作爲晶片的振盪器型式。當使用睡眠(power down)模式或內部低速時脈模式時，使晶片進入低耗電狀態。設定 CLKMD = 1，SN8P1603 選擇系統由一般模式切換爲內部低速時脈模式；在睡眠狀態中，系統可由埠 0 或埠 1 的觸發信號(高電位至低電位)，將系統喚醒，進入一般模式，在系統進入一般模式後，STPHX 及 CPUM0 位元自動被重置爲 ‘0’ ，內部低時脈及外部時脈自動開始振盪。



* HXRC is code option, 0 = crystal or resonator, 1 = RC

1.1. 振盪器暫存器：

OSCM 初值 = 0xxx 000x

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OSCM	WTCKS	WDRST	-	-	CPUM0	CLKMD	STPHX	-

WTCKS：看門狗計時器時脈源選擇。0 = fcpu，1 = 內部 RC 低時脈。
CLKMD：高/低速模式選擇位元 。0 = 一般(雙)模式，1 = 內部 RC 模式。
STPHX：高速振盪器停止控制位元 。0 = 自行運轉，1 = 停止。
CPUM0：CPU 操作模式控制位元。0 = 操作中(operating)，1 = 睡眠模式(power down)
及同時關閉高時脈及低時脈。

振盪器的振盪啓動穩定時間由組譯程式中的 OSC 碼選擇中的 SWARM 選擇之，01(低速 32KHz) = 13^{th}，0 = 18^{th}。

注意：在改變 CPU 操作模式後，再執行一個 NOP 指令。

例如：切換 CPU 操作由一般模式進入睡眠模式。

```
N2SLEEP:                    ;喚醒的振盪啓動穩定時間必須由組譯程式的 mask
                            ;程式碼選項選擇。
        B0BSET  FCUPM0      ;切換系統由一般模式進入睡眠模式
        NOP                 ;睡眠時，系統停留在此
```

2. 振盪器選項：

- 組譯程式中的碼選項(code option)如下所示：

SN8P1603 可選擇 4 種不同的振盪模式，使用者可設置 2 個位元(由 FOSC 的<1:0>)選擇 4 種中的其中一種。

•RC 振盪模式： 00
•低速晶體振盪 32KHz ~ 200KHz 01
•高速晶體振盪 12MHz 10
•4MHz 晶體振盪 11

設置 Xout 腳位成爲 P1.4 輸入/輸出腳位
影響啓動穩定時間及看門狗計時器

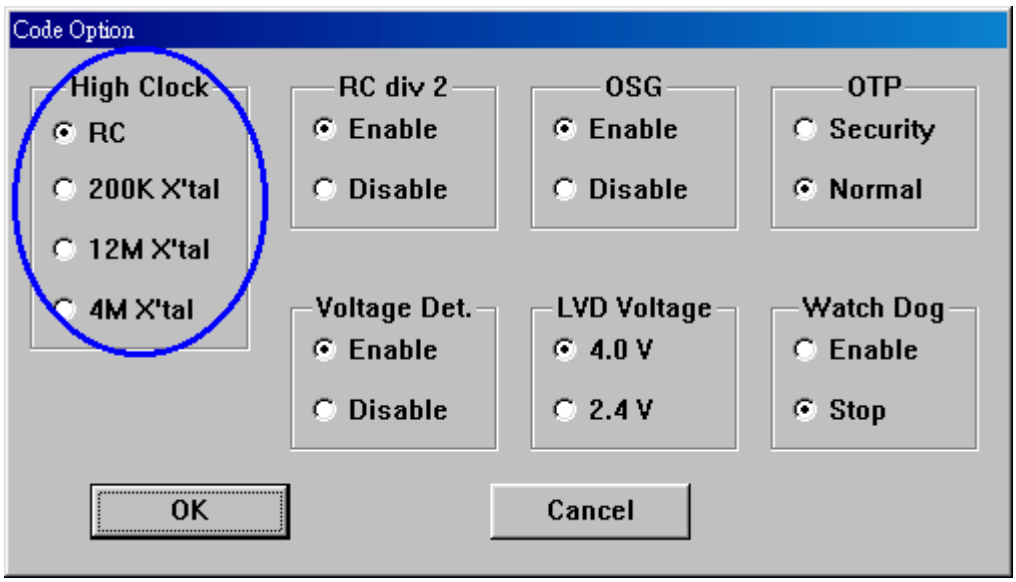
SONIX 組譯程式設定碼選項(Code Option)：

Mask option table

Function	Option value = 0	Option value = 1	Remark
HXRC 00	RC type	-	RC Oscillator and divide by 2
HXRC 01	X'TAL type	-	32Khz
HXRC 10	X'TAL type	-	12Mhz
HXRC 11	X'TAL type	-	4Mhz
SWARM	fhxosc ÷ 2^{18}	fhxosc÷ 2^{13}	The Warm-up timer
	fcpu ÷ 2^{14} ÷ 16	fcpu ÷ 2^{8} ÷ 16	The Watch-dog timer
Watch dog timer	Enable	Disable	
OTP security	Enable	Disable	

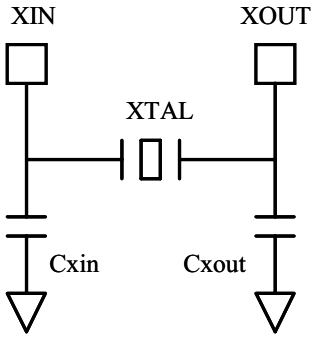
3. 振盪器應用電路接法：

SN8P1603 提供 3 種振盪電路，振盪型式分別爲 CRYSTAL(晶體振盪)、RC(電阻、電容振盪)及外部輸入振盪，使用者在 SONIX 組譯程式產生 '.BIN' 檔時，須決定、選擇適合硬體電路設計上所需振盪型式，若選擇的振盪型式爲外部輸入振盪，在組譯程式中選擇 'RC' 方式。



3.1. 晶體振盪器：

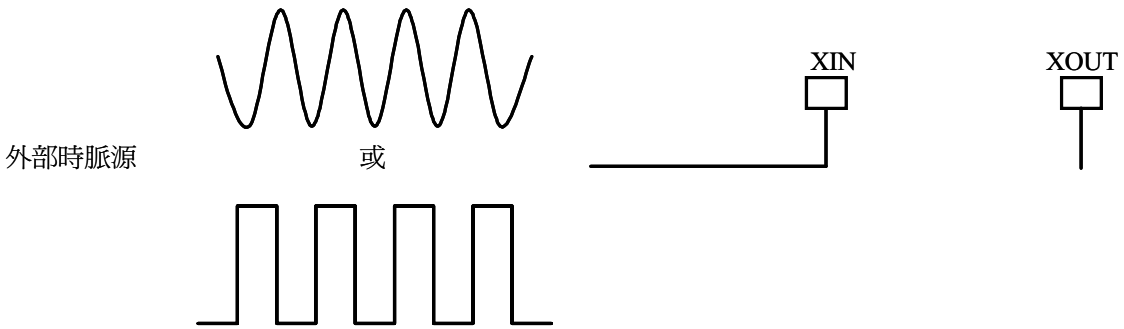
晶體振盪器分爲石英振盪器及陶瓷振盪器，接法相同，SN8P1603 選擇的晶體振盪器頻率最高爲 20MHZ，接法如下所示：



- XTAL 爲石英振盪器或陶瓷振盪器，其兩端接至 SN8P1603 的 XIN 及 XOUT 腳位。
- CXIN、CXOUT 是爲了振盪器的穩定度所增加的電容，電容值越大，穩定性越高，但是卻會影響脈波振盪初期的啓動穩定時間，建議 CXIN、CXOUT 數值爲 20P，就可以符合一般振盪器的應用，使用者可參閱振盪器生產廠商所提供的規格，加以修正相關外部電路。

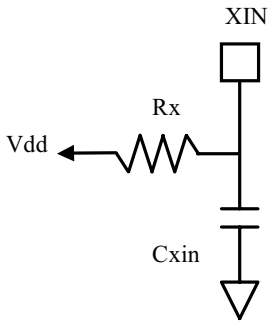
3.2. 外部時脈輸入：

SN8P1603 可接受外部時脈，作爲系統振盪源，時脈源可以爲直流方波及交流弦波信號，直接由 XIN 輸入，XOUT 閒置不接，在組譯程式時，振盪型式要選擇 RC 型式。



3.3. RC 振盪器：

RC 振盪電路是一種廉價的振盪電路結構，僅需電容、電阻，但是精確度不如晶體振盪結構，且振盪頻率會受電源電壓、電容值、電阻值及溫度影響，不如晶體振盪器精準，需要調整校正，振盪電路如下所示：



Cxin	RX(ohm)	振盪頻率(3Vdc)	振盪頻率(5Vdc)
20PF	1K	5.5MHz	7.1MHz
	3.3K	2.6MHz	3MHz
	5.1K	1.9MHz	2.1MHz
	10K	1MHz	1.2MHz
	100K	132KHz	131KHz

註：以上數值僅供參考。

測試 RC 數值選擇是否得到預期頻率，仍需經過測試，若直接使用示波器量測 XIN 的頻率，容易因爲探棒至示波器之間的阻抗效應，導致所測得的頻率產生偏差，所以建議方式爲利用指令週期，反推出實際系統振盪頻率，測試範例如下：

TOSC：

```
B0BSET   P1.1      ; 設 P1.1 爲 'H' ，此行爲 1 個指令週期
B0BCLR   P1.1      ; 設 P1.1 爲 'L' ，此行爲 1 個指令週期
JMP      TOSC      ; 跳至 TOSC 位址，此行爲 2 個指令週期
```

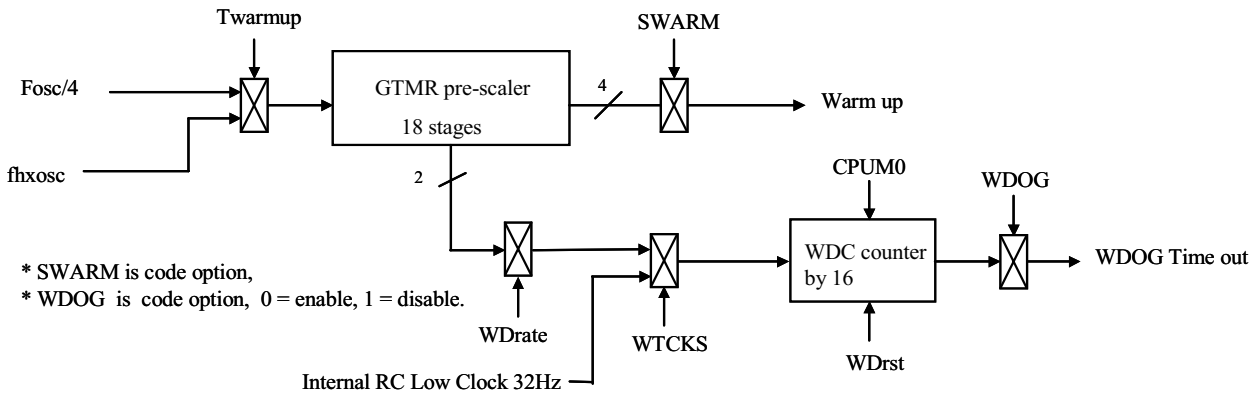
上述程序產生一個脈衝週期爲 1 個指令週期的方波輸出，利用示波器由 P1.1 量測週期寬度，若方波週期寬度爲 $1*10^{-6}$ sec，計算方式如下：

Fcpu = $1/10^{-6}$ = 1MHz ；系統運算速度爲 1MHz

因爲 Fcpu = Fosc ÷ 4

所以 Fosc = Fcpu * 4 = **4MHz** ；求得 RC 產生的時脈源爲 4 MHz

4. 振盪啓動穩定時間(WARM UP TIME) :



晶片進入一般模式前，晶片內部及振盪器都需要一段時間，讓振盪器起振、穩定，及晶片內部完成各項準備工作，如此系統才能順利運作、穩定。SN8P1603 提供 2 種振盪啓動穩定時間,應用在不同型態振盪器系統,fhxosc ÷ 2^{18} 設計在高速振盪器的應用， fhxosc ÷ 2^{13} 設計在低速振盪器的應用。在實際的應用，使用者可設置 SWARM 控制位元選擇適當的振盪啓動穩定時間，SWARM 控制位元同時設定 Wdrate；在送電之初，系統會自動選定設定模式中的最長喚醒時間作爲初始值(73.2 ms)。

範例 1：設定 SWARM = 0，則 Wdrate = 0，應用於高速振盪器(3.58MHz)。

SWARM = 0 → 振盪啓動穩定時間(warm up time) = 1/(fhxosc ÷ $2^{18)}$ = 73.2 ms
Wdrate = 0 → 看門狗計時器計數時間 = 1/(fcpu ÷ 2^{14} ÷ 16) = 293 ms

範例 2：設定 SWARM = 1，則 Wdrate = 1，應用於低速振盪器(32768Hz)。

SWARM = 1 → 振盪啓動穩定時間(warm up time) = 1/(fhxosc ÷ 2^{13}) = 250 ms
Wdrate = 1 → 看門狗計時器計數時間 = 1/(fcpu ÷ 2^{8} ÷16) = 0.5 ms

範例 3：對應於 HX_OSC 的振盪啓動穩定時間：

HX_osc	SWARM	喚醒時間
3.58 MHz	0	1/(fhxosc ÷ 2^{18}) = 73.2 ms
32768 Hz		1/(fhxosc ÷ 2^{13}) = 250 ms

註：SWARM 選項在 SONIX 組譯程式中，400K 代表 Fosc。

5. 看門狗計時器 (WDC) :

看門狗計時器(WDTMR)是一個 4-bit 二進制計數器(WDC)，爲了監控程式執行而設計。若程式因雜訊干擾而進入未知狀態下，導致 WDC 發生溢位，WDC 的溢位信號將會重置系統，程序重新操作。在一般操作流程中，使用者必須在溢位發生前插入一個指令，清除 WDC，使其重新計算，避免程式產生非預期的重置動作。要產生不同輸出時間，使用者可藉由設定 SWARM 控制 WDrate 位元(請參考上一節的範例 1 及範例 2)。

OSCM 初值 = 0xxx 000x

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OSCM	WTCKS	WDRST	-	-	CPUM0	CLKMD	STPHX	-

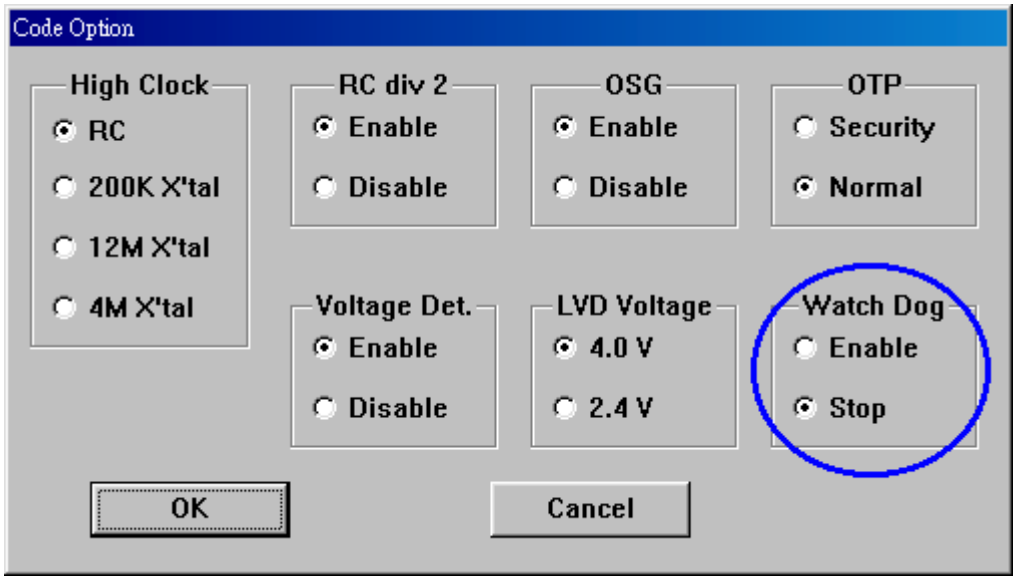
WDRST：看門狗計時器歸零控制位元。0 = 不歸零， 1 = 計時器歸零。
WTCKS：看門狗計時器時脈源選擇位元。0 = fcpu， 1 = 內部 RC 低時脈(32Hz)。
範例：

```
Main:                              ; 主程序開頭
        B0BSET  FWDRST             ; 歸零看門狗計時器
        Call    SUB1               ; 子程序 1
        Call    SUB2               ; 子程序 2
        .       .
        Jmp     Main.              ; 回主程序
```

利用程序週期來歸零看門狗計時器，防止因計時器溢位所導致系統重置。

Wdrate	看門狗計時器溢位時間
0	$1/(Fcpu \div 2^{14} \div 16) = 293$ ms (3.58MHz)
0	$1/(Fcpu \div 2^{14} \div 16) = 64$ ms (16384Hz)
1	$1/(Fcpu \div 2^{8} \div 16) = 4.5$ ms (3.58MHz)
1	$1/(Fcpu \div 2^{8} \div 16) = 1$ s (16384Hz)
看門狗計時器選擇爲內部 RC 低時脈	$1/(32 \div 16) = 0.5$ s (32Hz)

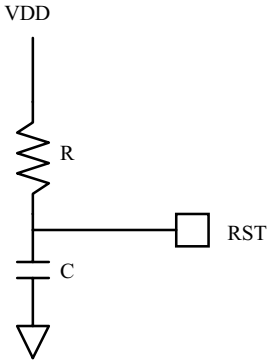
注意：看門狗計時器可由 SONIX 組譯程式中，選擇致能或禁能。



6. 外部重置保護電路：

SN8P1603 有外部重置偵測腳位(RST)，若 VDD 電位上升太慢或電源不穩時，RST 偵測到‘L’，代表外部重置線路輸出因 VDD 錯誤而輸出‘L’，系統停止工作，當 RST 偵測到‘H’時，系統執行重置工作，準備進入一般模式正常工作。對於 VDD 電壓準位在何種條件下爲正常電位，須靠外部重置電路處理，偵測 VDD 電位是否正常，配合不同應用環境，提供以下 3 種常用重置電路：

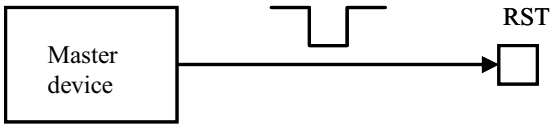
6.1. RC 重置電路：



電阻 R 的數值所造成壓降建議不要超過 0.3V，而 RST 腳位的漏電流爲 1uA，求得 R 爲 100K ohm，所以 R 的選用數值需小於 100K ohm，二極體的功用是加快電容放電時間。在產品開發測試階段，使用者可以在 RST 及 GND 之間接一個重置鍵，供系統手動重置之用，方便測試。

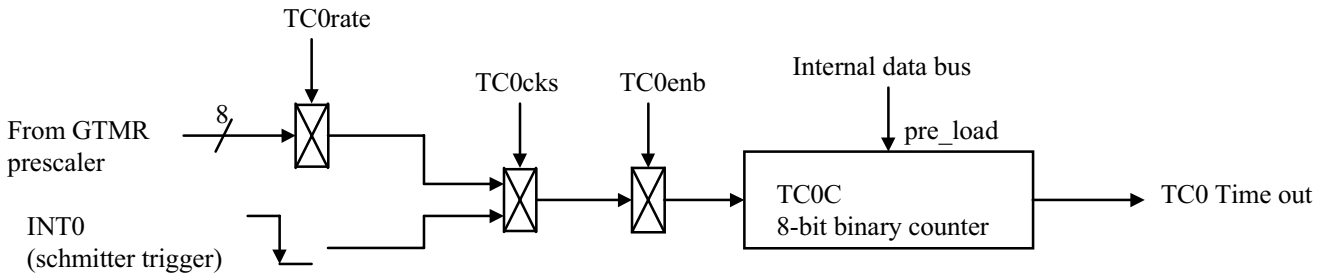
6.2. 外部重置信號：

SN8P1603 已經內建電壓偵測電路，也可以工作於被動端，由外部主器件傳送‘L’訊號，重置此晶片。



第四章. TC0 計時/事件計數器(TIMER/EVENT COUNTER)

TC0 是一個二進制的 8-bit 計時器及事件計數器，利用 TC0M 暫存器由 GTMR 或外部 INT0/P0.0 端點(falling edge trigger)選擇 TC0C 時脈源，計算一個精確的時間。若 TC0 計時器發生溢位(從 FFH 至 00H)，它會繼續計數，並且發布一個暫停信號，觸發 T0 中斷，請求中斷服務。



1. TC0M 模式暫存器：

TC0M 初值= 0xxx xxxx

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0RATE2	TC0RATE1	TC0RATE0	TC0CKS	-	-	-

TC0ENB：TC0 計數器致能位元。0 = 禁能，1 = 致能。
TC0RATE：TC0 內部時脈源選擇位元。
000 = Fcpu/256，001 = Fcpu/128， ... ，110 = Fcpu/4，111 = Fcpu/2。

TC0CKS：TC0 時脈資源選位元。0 = 內部時脈源，1 = 外部時脈源(INT0/P0.0)。

2. TC0C 計時暫存器：

TC0C 初值 = xxxx xxxx

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0C	X	x	X	x	x	x	x	x

TC0 中斷最大間隔時間如下所示：

TC0RATE	TC0CLOCK	High speed mode (fcpu = 3.58MHz / 4)	
		Max overflow interval	One step = max/256
000	fcpu/256	73.2 ms	286us
001	fcpu/128	36.6 ms	143us
010	fcpu/64	18.3 ms	71.5us
011	fcpu/32	9.15 ms	35.8us
100	fcpu/16	4.57ms	17.9us
101	fcpu/8	2.28ms	8.94us
110	fcpu/4	1.14ms	4.47us
111	fcpu/2	0.57ms	2.23us

註 1：TC0C 暫存器初值計算

TC0C 初值 = 256 - (TC0 中斷間隔時間 *輸入時脈)

例如：3.58MHz 高速模式下，在 TC0 設置 15 ms 中斷時間。
TC0C 數值(97H) = 256 - (15ms * Fcpu / 128)

TC0C 計算說明：

$$TC0C\ (\ 97H\ ) = 256 - (15 / 10^{-3} * 3.58 * 10^{6} / 4 / 128\ )$$

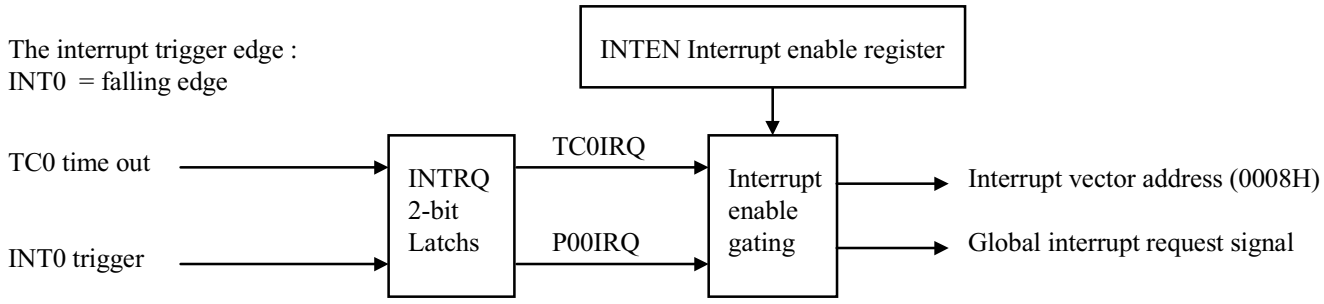
3. 設定程序說明：

TC0C 數值修正前必須先將 TC0 計時器禁能：

```
B0BCLR    FTC0IEN         ; 禁能 TC0 的中斷服務
B0BCLR    FTC0ENB         ; 禁能 TC0  計時器
MOV       A,#10H          ;
B0MOV     TC0M,A          ; 選擇 TC0 時脈 = Fcpu / 128
MOV       A,#97H
B0MOV     TC0C,A          ; 設定 TC0 中斷時間= 15 ms
B0BSET    FTC0IEN         ; 致能 TC0 的中斷服務
B0BCLR    FTC0IRQ         ; 清除 TC0 中斷要求
B0BSET    FTC0ENB         ; 致能 TC0 計時器
```

第五章. 中斷

SN8P1603 提供 2 個中斷源，包含 1 個內部中斷(TC0)及 1 個外部中斷(INT0)，當晶片在睡眠模式中，這些中斷具有喚醒功能，使系統進入高速模式；TC0 外部時脈輸入埠與 P0.0 共用，除此之外，P0.0 接腳具有喚醒功能。當中斷執行 1 次時，STPK 暫存器的 GIE 位元被清除爲 0，以停止其他中斷需求，在此情況下，當離開中斷服務程序，GIE 位元被設定爲 1，等待接受下一次中斷需求。所有的中斷需求訊號均儲存在 INTRQ 暫存器中，使用者可用程式檢查 INTRQ 暫存器的內容，自行設計安排中斷服務程序執行的優先權。



1. INTEN 中斷致能暫存器：

INTEN 初值 = xx0x xxx0

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	-	-	TC0IEN	-	-	-	-	P00IEN

P00IEN：外部的 P0.0 中斷要求位元。0 = 禁能，1 = 致能。
TC0IEN：計時器/事件計數器 0 中斷控制位元。0 = 禁能，1 = 致能。

2. INTRQ 中斷需求暫存器：

INTRQ 初值 = xx0x xxx0

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ	-	-	TC0IRQ	-	-	-	-	P00IRQ

P00IRQ：外部的 P0.0 中斷需求位元。 0 = 沒有需求，1 = 需求。
TC0IRQ：TC0 計時器/事件計數器中斷需求位元。0 = 沒有需求，1 = 需求。

3. 中斷服務程序範例說明：

3.1. 進入中斷需求/服務程序：

```
            ORG         0008H           ; 中斷程序位址
INT_RS：                                ; 中斷需求程序
            B0MOV       ACCBUF,A        ; 儲存 ACC 資料

            BOMOV       A,PFLAG         ; 儲存 PFLAG 資料
            B0MOV       PFLAGBUF,A
S

            B0BTS0      FP00IRQ         ; 檢查是否有 P00 中斷需求
            JMP         P00INTR         ; 跳至 P00 中斷服務程序
                                                          中斷需求優先
                                                          權排列
            B0BTS0      FTC0IRQ         ;檢查是否有 TC0 中斷需求
            JMP         TC0INTR         ; 跳至 TC0 中斷服務程序
            .           .                                 .
            .           .                                 .
QINT_RS：
            B0MOV       A,PFLAGBUF      ; 載入 PFLAG 資料
            B0MOV       PFLAG,A
            B0MOV       A,ACCBUF        ; 載入 ACC 資料

            RETI                        ; 離開中斷服務程序

P00INTR：                               ; P00 中斷服務程序
            B0BCLR      FP00IRQ         ; 清除 FP00IRQ 旗標
            .           .               .
            .           .               .
            JMP         QINT_RS         ; 跳出中斷服務程序

TCINTR：                                ; TC 中斷服務程序
            B0BCLR      FTCIRQ          ; 清除 FTCIRQ 旗標
            .           .               .
            .           .               .
            JMP         QINT_RS         ; 跳出中斷服務程序
```

3.2. *計時器(Timer)應用範例：*

在不同系統需求中，可能遇到需要不同時間長度的計時器(TIMER)，此時需利用中斷來達到精確時間計算之目的，提供下列說明，利用 TC0 作爲單位時間計算源，配合中斷服務程序分別計數不同的計時器。

3.2.1. 基底時間累加方式：

Fcpu = 3.58MHz/4，TC0 clock = Fcpu/64，time base = 5ms

```
TC0INTR：                                       ；TC0 中斷服務程序
                CALL    INCTM                   ；不同計時器的計數程序
                B0MOV   A,TMSTAT                ；檢查 TMSTAT 緩衝區
                JZ      QTC0INTR                ；爲 0，無中斷發生
                B0BTS0  TMSTAT.BIT10M           ；檢查 10ms 中斷旗標
                JMP     TC0INTR10M              ；跳至 10ms 中斷服務程序
                B0BTS0  TMSTAT.BIT100M          ；檢查 100ms 中斷旗標
                JMP     TC0INTR100M             ；跳至 100ms 中斷服務程序
                B0BTS0  TMSTAT.BIT1000M         ；檢查 1000ms 中斷旗標
                JMP     TC0INTR1000M            ；跳至 1000ms 中斷服務程序
                .       .
                .       .
QTC0INTR：                                      .
                B0BCLR  FTC0IRQ                 ；清除 FTC0IRQ 旗標
                MOV     A,#0BAH
                B0MOV   TC0C,A                  ；重置 TC0C
                JMP     QINT_RS                 ；跳出中斷服務程序
TC0INTR10M：
                B0BSET  INTGND.BIT10M           ；設定中斷介面旗標
                B0BCLR  TMSTAT10M               ；清除 TMSTAT10M
                JMP     QTC0INTR                ；跳出中斷服務程序
TC0INTR100M：
                B0BSET  INTGND.BIT100M          ；設定中斷介面旗標
                B0BCLR  TMSTAT100M              ；清除 TMSTAT100M
                JMP     QTC0INTR                ；跳出中斷服務程序
TC0INTR1000M：
                B0BSET  INTGND.BIT1000M         ；設定中斷介面旗標
                B0BCLR  TMSTAT1000M             ；清除 TMSTAT1000M
                JMP     QTC0INTR                ；跳出中斷服務程序
INCTM：
                B0BSET  INTGND.BIT5M            ；設 5ms 旗標
```

計時器中斷旗標判斷

計時器中斷介面旗標設定

```
                DECMS   INT10M              ; 計數緩衝器減 1
                JMP     INCTM10

                MOV     A,#2
                B0MOV   INT10M,A            ; 載入計數緩衝器初值
                B0BSET  TMSTAT.BIT10M       ; 設 10ms 旗標
INCTM10：
                DECMS   INT100M             ; 計數緩衝器減 1
                JMP     INCTM20

                MOV     A,#20
                B0MOV   INT100M,A           ; 載入計數緩衝器初值
                B0BSET  TMSTAT.BIT100M      ; 設 100ms 旗標
INCTM20：
                DECMS   INT1000M            ; 計數緩衝器減 1
                JMP     QINCTM

                MOV     A,#200
                B0MOV   INT1000M,A          ; 載入計數緩衝器初值
                B0BSET  TMSTAT.BIT1000M     ; 設 1000ms 旗標
QINCTM：
                RET
```

計時器計數程序

```
MNINTGND：
                B0MOV   A,INTGND            ; 檢查 INTGND 緩衝區
                JZ      QMNINTGND           ; 爲 0，無計時器旗標發生

                B0BTS0  INTGND.BIT5M        ; 檢查 5ms 中斷旗標
                JMP     MNINTGND5M          ; 跳至 5ms 中斷服務程序
MNINTGND1：
                B0BTS0  INTGND.BIT10M       ; 檢查 10ms 中斷旗標
                JMP     MNINTGND10M         ; 跳至 10ms 中斷服務程序
MNINTGND2：
                B0BTS0  INTGND.BIT100M      ; 檢查 100ms 中斷旗標
                JMP     MNINTGND100M        ; 跳至 100ms 中斷服務程序
MNINTGND3：
                B0BTS0  INTGND.BIT1000M     ; 檢查 1000ms 中斷旗標
                JMP     MNINTGND1000M       ; 跳至 1000ms 中斷服務程序
QMNINTGND：
                RET
```

各時間旗標判斷此程序置於主程序中

```
MNINTGND5M：
                B0BCLR  INTGND.BIT5M        ; 清除 5ms 旗標
                .       .                   ; 5ms 應用程序
                .       .
                JMP     MNINTGND1           ; 跳至下一組判斷

MNINTGND10M：
                B0BCLR  INTGND.BIT10M       ; 清除 10ms 旗標
```

```
                .               .                   ; 10ms 應用程序
                .               .
                JMP             MNINTGND2           ; 跳至下一組判斷
                                                                        各時間
MNINTGND100M：                                                          應用程
                B0BCLR          INTGND.BIT100M      ; 清除 100ms 旗標   序執行
                .               .                   ; 100ms 應用程序    區
                .               .
                JMP             MNINTGND3           ; 跳至下一組判斷

MNINTGND1000M：
                B0BCLR          INTGND.BIT1000M     ; 清除 1000ms 旗標
                .               .                   ; 1000ms 應用程序
                .               .
                JMP             QMNINTGND           ; 離開 MNINTGNDT 程序
```

註：
1.TMSTAT 是不同時間計時器計數終止旗標，INCTM 與 TC0INTR 之間介面旗標。
TMSTAT.0 = TMSTAT.BIT5M(5ms)，TMSTAT.1 = TMSTAT.BIT10M(10ms)，TMSTAT.2 = TMSTAT.BIT100M(100ms)，TMSTAT.3 = TMSTAT.BIT1000M(1000ms)。

2. INTGND 是不同時間計時器中斷旗標，TC0INTR 與主程序之間介面旗標。
INTGND.0 = INTGND.BIT5M(5ms)，
INTGND.1 = INTGND.BIT10M(10ms)，
INTGND.2 = INTGND.BIT100M(100ms)，
INTGND.3 = INTGND.BIT1000M(1000ms)。

3.此方式處理時間由於使用同一個基底時間，會有計數後時間同時到達的可能，所以對於各個時間的計數倍數選擇需要錯開，防止時間點偵測失誤發生。

3.2.2. 異相位(different phase)方式：

時間區段處理程序，有另一種方式處理，稱之爲異相位(different phase)方式，此方法將各個時間點錯開，產生不同的基底時間，可針對不同計時需求，選擇適當的基底時間做計數。使用暫存器 TMBUF 儲存不同時間點發生旗標：

TMBUF = XXXXXXXX

假設中斷發生單位時間爲 5ms，則：

TMBUF.0：$5ms*2^0$ = 5ms 旗標　　TMBUF.1：$5ms*2^1$ = 10ms 旗標
TMBUF.2：$5ms*2^2$ = 20ms 旗標　　TMBUF.3：$5ms*2^3$ = 40ms 旗標
TMBUF.4：$5ms*2^4$ = 80ms 旗標　　TMBUF.5：$5ms*2^5$ = 160ms 旗標
TMBUF.6：$5ms*2^6$ = 320ms 旗標　　TMBUF.7：$5ms*2^7$ = 640ms 旗標

下表說明 TMBUF 數值與時間點中斷服務程序執行時機：

TMBUF	中斷服務程序執行							
	5ms	10ms	20ms	40ms	80ms	160ms	320ms	640ms
XXXXXXX1	V	-	-	-	-	-	-	-
XXXXXX10	-	V	-	-	-	-	-	-
XXXXX100	-	-	V	-	-	-	-	-
XXXX1000	-	-	-	V	-	-	-	-
XXX10000	-	-	-	-	V	-	-	-
XX100000	-	-	-	-	-	V	-	-
X1000000	-	-	-	-	-	-	V	-
10000000	-	-	-	-	-	-	-	V

中斷程序：

```
T0CINTR：                                   ；TC0 中斷服務程序
        INCMS     TMBUF
        JMP       TC0INTR10                 ；TMBUF 不爲 ‘0’
                                            ；TMBUF 爲 ‘0’，代表由
                                            ；FFH 加 1 爲 ‘0’，發生
                                            ；溢位                          TMBUF 加 1 程序
        MOV       A,#00000001B              ；修正 TMBUF 數值爲
        B0MOV     TMBUF,A                   ；00000001B

TC0INTR10：
        B0BTS0    TMBUF.0
        JMP       INTR5MS                   ；TMBUF = XXXXXXX1
                                            ；跳至 5ms 中斷服務程序
        B0BTS0    TMBUF.1
        JMP       INTR10MS                  ；TMBUF = XXXXXX10
                                            ；跳至 10ms 中斷服務程序
        B0BTS0    TMBUF.2
        JMP       INTR20MS                  ；TMBUF = XXXXX100
                                            ；跳至 20ms 中斷服務程序
        B0BTS0    TMBUF.3
        JMP       INTR40MS                  ；TMBUF = XXXX1000
                                            ；跳至 40ms 中斷服務程序          中斷服務程序執行判斷
        B0BTS0    TMBUF.4
        JMP       INTR80MS                  ；TMBUF = XXX10000
                                            ；跳至 80ms 中斷服務程序
        B0BTS0    TMBUF.5
        JMP       INTR160MS                 ；TMBUF = XX100000
                                            ；跳至 160ms 中斷服務程序
        B0BTS0    TMBUF.6
        JMP       INTR320MS                 ；TMBUF = X1000000
                                            ；跳至 320ms 中斷服務程序
        B0BTS0    TMBUF.7
```

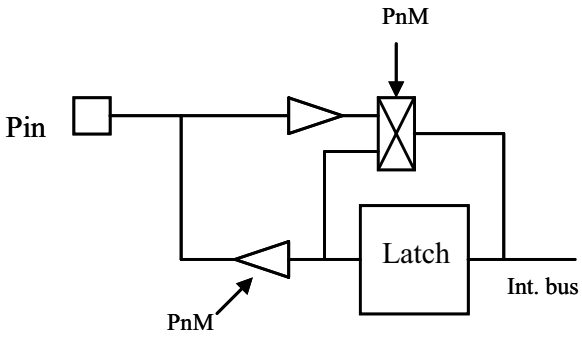
```
                JMP         INTR640MS       ; TMBUF = 10000000
                                            ;跳至 640ms 中斷服務程序

TC0INTR90 :
                B0BCLR      FTC0IRQ         ; 清除 FTC0IRQ 旗標
                MOV         A,#0BAH
                B0MOV       TC0C,A          ; 重置 TC0C
                JMP         QINT_RS         ; 跳出中斷服務程序

INTR5MS :                                   ; 5ms 中斷服務程序
                .           .
                .           .
                JMP         TC0INTR90

INTR10MS :                                  ; 10ms 中斷服務程序
                .           .
                .           .
                JMP         TC0INTR90

INTR20MS :                                  ; 20ms 中斷服務程序
                .           .
                .           .
                JMP         TC0INTR90

INTR40MS :                                  ; 40ms 中斷服務程序
                .           .
                .           .
                JMP         TC0INTR90

INTR80MS :                                  ; 80ms 中斷服務程序
                .           .
                .           .
                JMP         TC0INTR90

INTR160MS :                                 ; 160ms 中斷服務程序
                .           .
                .           .
                JMP         TC0INTR90

INTR320MS :                                 ; 320ms 中斷服務程序
                .           .
                .           .
                JMP         TC0INTR90

INTR640MS :                                 ; 640ms 中斷服務程序
                .           .
                .           .
                JMP         TC0INTR90
```

第六章. 輸入/輸出埠

SN8P1603 提供 3 組埠給使用者應用，包括 1 個輸入埠(P0)及 2 個輸入/輸出埠(P1、P2)，I/O 埠方向是由 PnM 暫存器所選擇的，當系統重置後，I/O 埠工作狀態是輸入埠，若使用者欲從 I/O 埠讀進一個數值，設置 I/O 埠爲輸入模式，再執行讀取指令(B0BTS0 M.b，B0BTS1 M.b，B0MOV A,M)。

Port0 structure

Port1, 2 structure



註：所有栓鎖輸出電路都是 push-pull 架構。

1. 埠 1 喚醒 (P1W)暫存器：

在睡眠模式或省電模式中，埠 1 的一個腳位有邏輯 ‘L’ 信號時，可喚醒晶片進入正常模式中運作，在這種情況下，P1.n 必須藉由 P1M 的控制，設定爲輸入模式，而其喚醒功能是由 P1W 暫存器設定，進入睡眠模式之前，有參予喚醒功能的位元需設定其數值爲 ‘1’ 。

P1W 初值 = xxx0 0000

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	-	-	-	P14W	P13W	P12W	P11W	P10W

P1nW：p1.n 喚醒控制位元。0 = 沒有喚醒功能，1 = 有喚醒功能。

2. 埠模式(PnM) 暫存器：

PnM 初值= 0000 0000

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PnM	Pn7M	Pn6M	Pn5M	Pn4M	Pn3M	Pn2M	Pn1M	Pn0M

PnM：n 表示 1、2。
Pn7M ~ Pn0M：Port n.7 ~ Port n.0 輸入/輸出模式控制位元。0 = 輸入模式，1 = 輸出模式。

設定埠的操作模式：

```
MOV     A,#00001111B
B0MOV   PnM,A           ; n=1、2，設定 Pn 的 0~3 位元爲輸出模式，
                        ; 其餘位元爲輸入模式。
```

註：P0 是固定輸入埠，不需再設定其操作模式。

3. 埠 (Pn)資料暫存器：

Pn 初值= xxxx xxxx

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pn	Pn7	Pn6	Pn5	Pn4	Pn3	Pn2	Pn1	Pn0

Pn：n 表示 0、1、2。
Pn7 ~ Pn0：Port n.7 ~ Port n.0 輸入/輸出資料位元。0 = 邏輯‘低’，1 = 邏輯‘高’。

讀取埠的輸入數值：

```
B0MOV     A,Pn          ; n=0、1、2，儲存埠的數值(Pn)儲存到 BUF0
B0MOV     BUF0,A
```

輸入埠單一位元判斷：

```
B0BTS1    Pn.0          ; n=0、1、2，判斷 Pn 的 0 位元數值是否爲 1
JMP       VAL0          ; 數值爲 0，跳至 VAL0 執行相關程序
JMP       VAL1          ; 數值爲 1，跳至 VAL1 執行相關程序

B0BTS0    Pn.0          ; n=0、1、2，判斷 Pn 的 0 位元數值是否爲 0
JMP       VAL1          ; 數值爲 1，跳至 VAL1 執行相關程序
JMP       VAL0          ; 數值爲 0，跳至 VAL0 執行相關程序
```

4. 埠模式改變注意事項：

- 埠模式改變時要先設定埠數值，在轉換埠模式，以確保埠端無突波發生及數值正確。
- 設定埠爲輸入模式，以 Pn(n=1,2)的 8 個位元說明之：

```
MOV       A,#00000000b
B0MOV     Pn,A          ; 設定 Pn 數值爲 00000000b

B0MOV     PnM,A         ; 設定 Pn 8 個位元均爲輸入模式
```

說明：由於輸入至埠的數值是未知的，若原來埠值爲‘1’，而外部電路爲‘0’，會造成模式切換後，因爲埠內部放電太慢，數值浮動而導致讀取錯誤，所以先將埠值設爲‘0’，再轉爲輸入埠模式。

- 設定埠爲輸入模式，有外接昇壓電阻，以 Pn(n=1,2) 的 8 個位元均有外接昇壓電阻說明之：

```
MOV       A,#11111111b
B0MOV     Pn,A          ; 設定 Pn 數值爲 11111111b

MOV       A,#00000000b
B0MOV     PnM,A         ; 設定 Pn 8 個位元均爲輸入模式
```

說明：由於輸入至埠的數值是未知的，若原來埠值爲 ‘0’ ，因有外接昇壓電阻，切換至輸入模式時，會因昇壓電阻而提昇電壓爲 ‘1’ ，此時會有一段數值浮動時期或突波發生，爲避免讀值錯誤，所以先將埠值設爲 ‘1’ ，再轉爲輸入埠。

- 設定埠爲輸出模式，以 Pn(n=1,2) 的輸出數值爲 01010101b 說明之：

```
MOV     A,#01010101b
B0MOV   Pn,A            ; 設定 Pn 數值爲 01010101b

MOV     A,#11111111b
B0MOV   PnM,A           ; 設定 Pn 8 個位元均爲輸出模式
```

說明：由於埠轉換模式前的數值未知，可能在埠模式轉換期間，造成輸出數值跳動，進而導致應用電路誤動作，所以先將欲輸出的埠數值設定，再轉換爲輸出模式，即可避免上述問題。

第七章. 電氣資料

1. 絕對最大範圍：

(所有電壓均以 Vss 爲參考電壓)

電源電壓(Vdd) ..... - 0.3V~6.0V
輸入埠輸入電壓(Vin) ..... VSS - 0.2V ~ VDD + 0.2V
操作溫度(Topr) .....0°C ~ + 70°C
保存溫度(Tstor) ..... -30°C ~ + 125°C
消耗功率(Pc) ..... 500 mW

2. 電氣特性：

(所有電壓參均以 Vss 爲參考電壓，Vdd = 5.0V，Fosc = 3.579545 MHz，環境溫度爲 25°C 其餘另外備註)

參數	標誌	敘述		最小值	標準值	最大值	單位
操作電壓	Vdd	一般模式，Vpp = Vdd		2.4	5.0	5.5	V
		燒錄模式，Vpp = 12.5V		4.5	5.0	5.5	
重置腳位輸入電壓	ViH	高電壓準位		0.7Vdd	-	-	V
	ViL	低電壓準位		-	-	0.3Vdd	
重置腳位漏電流	ILekg	Vin = Vdd		-	-	1	uA
I/O 埠輸入電壓	ViH	Port 0		-	0.7Vdd	-	V
	ViL			-	0.3Vdd	-	
	ViH	Port1 及 Port2		-	0.6Vdd	-	
	ViL			-	0.4Vdd	-	
I/O 埠輸入漏電流	ILekg	Vin = Vdd，Vin = Vss		-	-	2	uA
Port1 輸出電流源 Sink 電流	IoH	Vop = Vdd - 0.5V		-	15	-	mA
	IoL	Vop = Vss + 0.5V		-	15	-	
Port2 輸出電流源 Sink 電流	IoH	Vop = Vdd - 0.5V		-	15	-	mA
	IoL	Vop = Vss + 0.5V		-	15	-	
INTn 觸發脈衝寬度	Tint0	INT0 中斷請求脈衝寬度		2/Fcpu	-	-	cycle
振盪器頻率	fhxosc	晶體振盪器或陶瓷振盪器		0.03	20	-	MHz
		Vdd=3V，RC 振盪及外部模式		0.03	8	-	
		Vdd=5V，RC 振盪及外部模式		0.03	15	-	
供應電流	Idd1	操作模式	Vdd=5V 4MHz	-	5	-	mA
			Vdd=3V 4MHz	-	1.5	-	mA
			Vdd=3V 32768Hz	-	45	-	uA
	Idd2	內部 RC 模式 16KHz	Vdd=5V	-	368	-	uA
			Vdd=3V	-	265	-	uA
	Idd3	睡眠模式	Vdd=5V	-	350	-	uA
			Vdd=3V	-	250	-	uA
內部 POR	Vpor	內部 POR 偵測準位		-	2.6	-	V
低電壓偵測電流	Ivdet	低電壓偵測器操作電流		-	100	-	uA

睡眠模式所需電流主要源自於低電壓偵測器。

注意:
1.SN8P1602 低電壓偵測(LVD)功能由組譯程式選擇啓動或關閉。
2.SN8P1603 低電壓偵測(LVD)功能永遠啓動，睡眠模式所需電流約爲 100uA。

第八章. 指令集

類別	語法		說明	C	DC	Z	指令週期
MOVE	MOV	A,M	A←M	-	-	√	1
	MOV	M,A	M←A	-	-	-	1
	B0MOV	A,M	A←M(bnak 0)	-	-	√	1
	B0MOV	M,A	M(bank 0)←A	-	-	-	1
	MOV	A,I	A←I	-	-	-	1
	B0MOV	M,I	M←I，(M=工作暫存器，RBANK 與 PFLAG)	-	-	-	1
	XCH	A,M	A←→M	-	-	-	1
	B0XCH	A,M	A←→M(bank 0)	-	-	-	1
	MOVC		R，A←ROM[Y,Z]	-	-	-	2
ARITHMETIC	ADC	A,M	A←A+M+C，若發生溢位，C=1，否則 C=0	√	√	√	1
	ADC	M,A	M←A+M+C，若發生溢位，C=1，否則 C=0	√	√	√	1
	ADD	A,M	A←A+M，若發生溢位，C=1，否則 C=0	√	√	√	1
	ADD	M,A	M←M+A，若發生溢位，C=1，否則 C=0	√	√	√	1
	B0ADD	M,A	M(bank 0)←M(bank 0)+A， 若發生溢位，C=1，否則 C=0	√	√	√	1
	ADD	A,I	A←A+I，若發生溢位，C=1，否則 C=0	√	√	√	1
	SBC	A,M	A←A-M-/C，若發生借位，C=0，否則 C=1	√	√	√	1
	SBC	M,A	M←A-M-/C，若發生借位， C=0，否則 C=1	√	√	√	1
	SUB	A,M	A←A-M，若發生借位，C=0，否則 C=1	√	√	√	1
	SUB	M,A	M←A-M，若發生借位，C=0，否則 C=1	√	√	√	1
	SUB	A,I	A←A-I，若發生借位，C=0，否則 C=1	√	√	√	1
	DAA		調整 ACC 資料格式從 16 進制(HEX)爲 10 進制(DEC)	√	-	-	1
LOGIC	AND	A,M	A←A and M	-	-	√	1
	AND	M,A	M←A and M	-	-	√	1
	AND	A,I	A←A and I	-	-	√	1
	OR	A,M	A←A or M	-	-	√	1
	OR	M,A	M←A or M	-	-	√	1
	OR	A,I	A←A or I	-	-	√	1
	XOR	A,M	A←A xor M	-	-	√	1
	XOR	M,A	M ← A xor M	-	-	√	1
	XOR	A,I	A ← A xor I	-	-	√	1
PROCESS	SWAP	M	A(b3~b0，b7~b4) ←M(b7~b4，b3~b0)	-	-	-	1
	SWAPM	M	M(b3~b0，b7~b4) ← M(b7~b4，b3~b0)	-	-	-	1
	RRC	M	A ← RRC M	√	-	-	1
	RRCM	M	M ← RRC M	√	-	-	1
	RLC	M	A ← RLC M	√	-	-	1
	RLCM	M	M ← RLC M	√	-	-	1
	CLR	M	M ← 0	-	-	-	1
	BCLR	M.B	M.b ← 0	-	-	-	1
	BSET	M.B	M.b ← 1	-	-	-	1
	B0BCLR	M.B	M(bank 0).b ← 0	-	-	-	1
	B0BSET	M.B	M(bank 0).b ← 1	-	-	-	1
BRANCH	CMPRS	A,I	ZF，C ← A – I，若 A = I，則跳至下一個指令	√	-	√	1+S
	CMPRS	A,M	ZF，C ← A – M，若 A = M，則跳至下一個指令	√	-	√	1+S
	INCS	M	A ← M + 1，若 A = 0，則跳至下一個指令	-	-	-	1+S
	INCMS	M	M ← M + 1，若 M = 0，則跳至下一個指令	-	-	-	1+S
	DECS	M	A ← M – 1，若 A = 0，則跳至下一個指令	-	-	-	1+S
	DECMS	M	M ← M – 1，若 M = 0，則跳至下一個指令	-	-	-	1+S
	BTS0	M.B	若 M.b = 0，則跳至下一個指令	-	-	-	1+S
	BTS1	M.B	若 M.b = 1，則跳至下一個指令	-	-	-	1+S
	B0BTS0	M.B	若 M(bank 0).b = 0，則跳至下一個指令	-	-	-	1+S
	B0BTS1	M.B	若 M(bank 0).b = 1，則跳至下一個指令	-	-	-	1+S
	JMP	D	PC15/14 ← RomPages1/0， PC13~PC0 ← d	-	-	-	2
	CALL	D	Stack ← PC15~PC0，PC15/14 ← RomPages1/0，PC13~PC0 ← D	-	-	-	2
MIS	RET		PC←堆疊	-	-	-	2
	RETI		PC←堆疊，致能主中斷	-	-	-	2
	NOP		空白指令，無動作	-	-	-	1

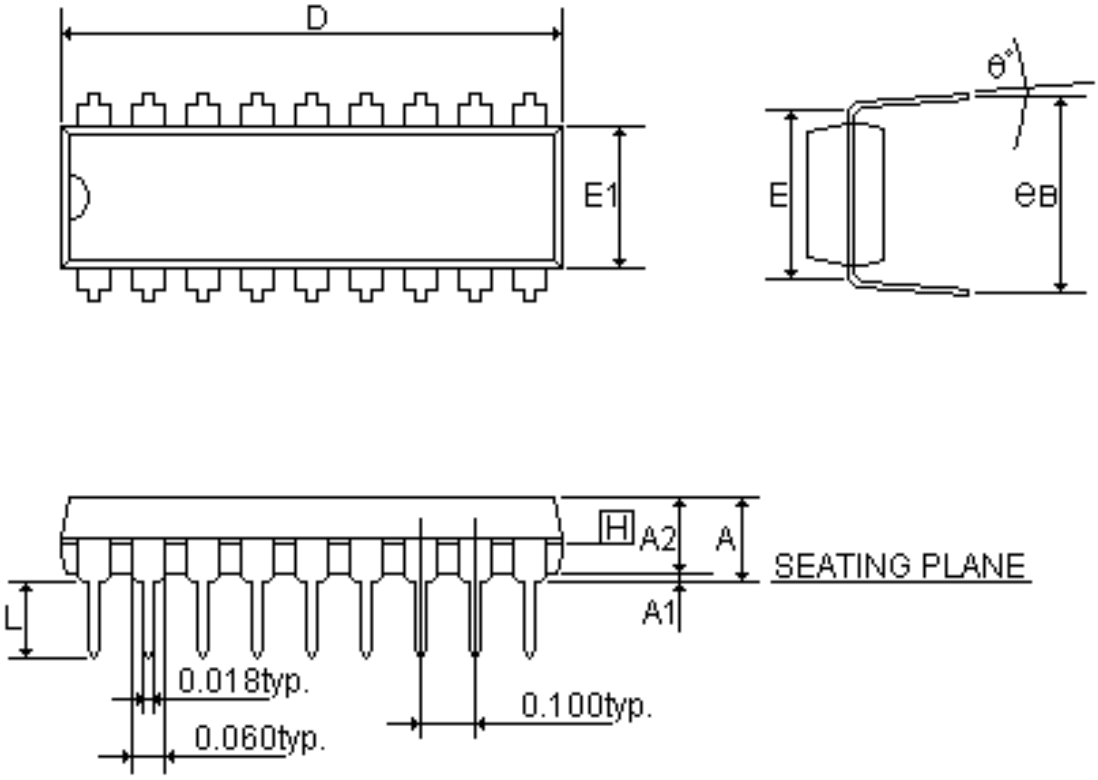
註：a).工作暫存器 = R, Y, Z。

b).記憶體存取於 RAM[Y,Z]位置，若 M = @YZ (位於 RAM bank 0 的 E7H 位址)。

c).所有指令都只佔一個指令週期，除了程序支線及 PC 更新情況下佔 2 個指令週期。

第九章. 包裝資訊：

P-DIP18 pin :



Symbols	MIN.	NOR.	MAX.
A	-	-	0.210
A1	0.015	-	-
A2	0.125	0.130	0.135
D	0.880	0.900	0.920
E		0.300BSC.	
E1	0.245	0.250	0.255
L	0.115	0.130	0.150
e_B	0.335	0.355	0.375
θ°	0	7	15

UNIT : INCH