SN8P04xx 系列

第一章. 概論 4

1. 綜述 : 4
2. 特性 : 4
3. 系統方塊圖 : 5
4. 包裝型式 : 5
5. 腳位的說明 : 8

第二章. 位址(ADDRESS) 9

1. 程式記憶體 (ROM) : 9
2. 資料記憶體 (RAM) : 9
   2.1. RAM BANK 位置 : 9
   2.2. 系統暫存器配置表 (BANK 0) : 10
   2.3. 系統暫存器表格 : 11
3. 累積器 (ACC) : 12
   3.1. 溢位旗標 : 13
   3.2. 小數溢位旗標 : 13
   3.3. 零旗標 : 13
   3.4. 範例 : 13
4. 工作暫存器 : 17
   4.1. H,L 暫存器 : 17
   4.2. Y,Z 暫存器 : 18
   4.3. 查詢表格說明 : 18
   4.4. 定址種類 : 19
5. 程式計數器 : 20
   5.1. 單一位址跳躍 : 20
   5.2. 多數位址跳躍 : 20
6. 堆疊緩衝區 : 22
7. ACC 累積器與工作暫存器的保護 : 23

第三章. 振盪電路 24

1. 振盪器 : 24
   1.1. 振盪器暫存器 : 24
2. 振盪器應用電路接法 : 25
   2.1. 晶體振盪器 : 26
   2.2. 外部時脈輸入 : 26
   2.3. RC 振盪器 : 27

3. 振盪啟動穩定時間(WARM UP TIME) : 28
4. 看門狗計時器 (WDC) : 29
5. 外部重置保護電路 : 30
5.1. RC 重置電路 : 30
5.2. 穩壓二極體重置電路 : 30
5.3. 電阻分壓方式重置電路 : 31
第四章. 計時器(TIMER)與計時/事件計數器(TIMER/EVENT COUNTER) 32
1. 基本計時器 (T0) : 32
1.1. T0M 模式暫存器 : 32
1.2. T0C 計數暫存器 : 32
1.3. 設定程序說明 : 33
2. 計時器/事件計數器(TC0) : 33
2.1. TC0M 模式暫存器 : 34
2.2. TC0C 計時暫存器 : 34
2.3. 設定程序說明 : 34
3. 計時器/事件計數器 (TC1) : 35
3.1. TC1M 模式計錄器 : 35
3.2. TC1C 計時暫存器 : 35
3.3. TC1R 自動重新載入 暫存器 : 36
3.4. 設定程序說明 : 36
3.5. TC1 自動重新載入 功能範例說明 : 36
第五章. 串列輸入/輸出 發射接收器 (SIO) 41
1. SIOM 模式暫存器 : 41
2. SIOB/SIOR 資料緩衝器 : 42
3. SIO 操作範例 : 42
3.1. 主動式 SIO : 42
3.2. 被動式 SIO : 45
第六章. 中斷 49
1. INTEN 中斷致能暫存器 : 49
2. INTRQ 中斷需求暫存器 : 50
3. 中斷服務程序範例說明 : 50
3.1. 進入中斷需求服務程序 : 50
3.2. 計時器(Timer) 應用範例 : 51
第七章. 輸入/輸出埠 56
1. 埠 1 喚醒 (P1W)暫存器 : 56
2. 上拉電阻器 (PnUR)暫存器 : 56
3. 埠模式(PnM) 暫存器 : 57

4. 埠 (Pn)資料暫存器 : 57
5. 埠模式改變注意事項 : 58
第八章. 8 頻道類比數位轉換器 59
1. ADM 暫存器 : 59
2. ADB&ADR 暫存器 : 59
3. ADC 範例說明 : 60
4. AIN 輸入電壓及 ADB 輸出資料對照表 : 60
第九章. 7-BIT 數位類比暫存器 61
1. DAM 暫存器 : 61
2. DAB 資料及 DAO 輸出電壓對照表 : 61
3. 範例說明 : 61
3.1. 設定及輸出類比信號 : 61
3.2. 輸出 sin 弦波 : 62
3.3. 直流馬達控制線路 : 63
第十章. 電氣資料 64
1. 絕對最大範圍 : 64
2. 電氣特性 : 64
第十一章. 指令集 65

第一章. 概論

1. 緒言:

SN8P04XX 系列是一個 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 4K*16 bits OTP ROM，128*8 bits 資料記憶體，一個 8-bit 基本計時器(T0)及兩個 8-bit 計時器/事件/計數器(TC0/TC1)，看門狗計時器(watchdog timer)，7 個中斷源(T0，TC0，TC1，SIO，INT0~INT2)，一組帶有 8-bit 解析度的 8 通道類數轉換器(ADC)，一組 7-bit 數類轉換器(DAC)，34 個輸入/輸出(I/O)腳位，8 層堆疊緩衝區(STACK)，除此之外使用者可自行選擇微處理器的振盪型式，有 3 種振盪型式可供選擇來產生系統時脈，包含高效能的石英振盪器，陶瓷震盪器及廉價的 RC 振盪模式。

2. 特性:

- **記憶體結構：**
  OTP ROM 容量：4096 * 16 bits
  RAM 容量：128 * 8 bits

- **I/O 端點型態 (總共 34 個腳位)：**
  1 組輸入埠：4 個腳位具有喚醒功能。
  1 組輸入/輸出埠：6 個腳位具有喚醒功能及一般用途功能。
  2 組輸入/輸出埠：16 個腳位作爲一般用途。
  1 組輸入/輸出埠：8 個腳位與 ADC 輸入共用。

- **1 個 8-bit 基本計時器**
- **1 個計時/事件計數器**
- **1 個計時/事件計數器帶有自動載入及驅動 Buzzer 功能**
- **1 個看門狗計時器**

- **59 個功能強大指令：**
  所有的指令皆爲 1 個或 2 個執行週期的單字元格式。
  執行時間：1 個週期爲 4 個振盪時脈。
  JMP 指令(無 ROM 範圍限制)。
  CALL 指令(無 ROM 範圍限制)。
  查表功能，**MOVC** 指令(無 ROM 範圍限制)。
  MUL 指令爲乘法算術。

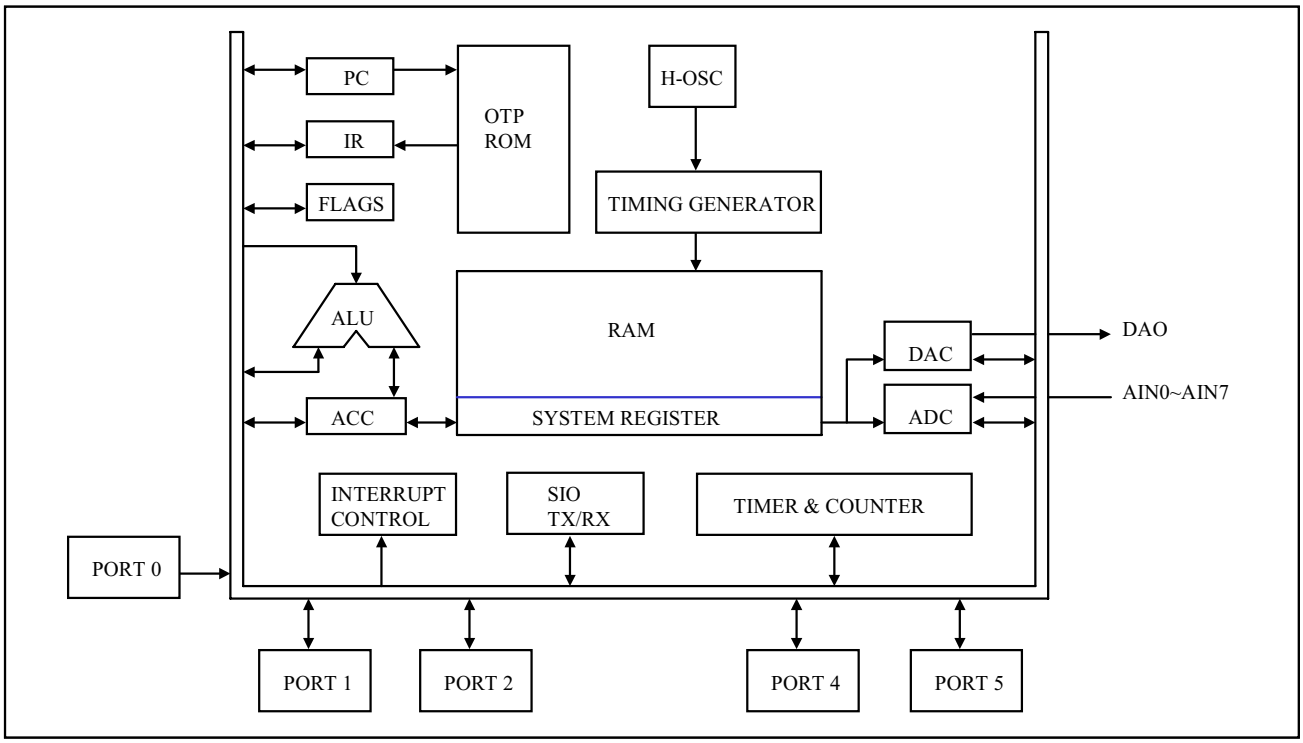
- **7 個中斷源：**
  4 個內部中斷：T0，TC0，TC1，SIO
  3 個外部中斷：INT0~INT2

- **8 層堆疊緩衝區**
- **1 個 8-bit 解析度的 8 通道類數轉換器**
- **1 個 7-bit 數類轉換器**

- **單一振盪器：**
  石英振盪器：速度最高爲 10MHz
  RC 型式：速度最高爲 4MHz

- **包裝型式：**
  PDIP：40 pin
  SKDIP：28 pin
  SOP：28，24 pin
  QFP：44 pin

3. 系統方塊圖：



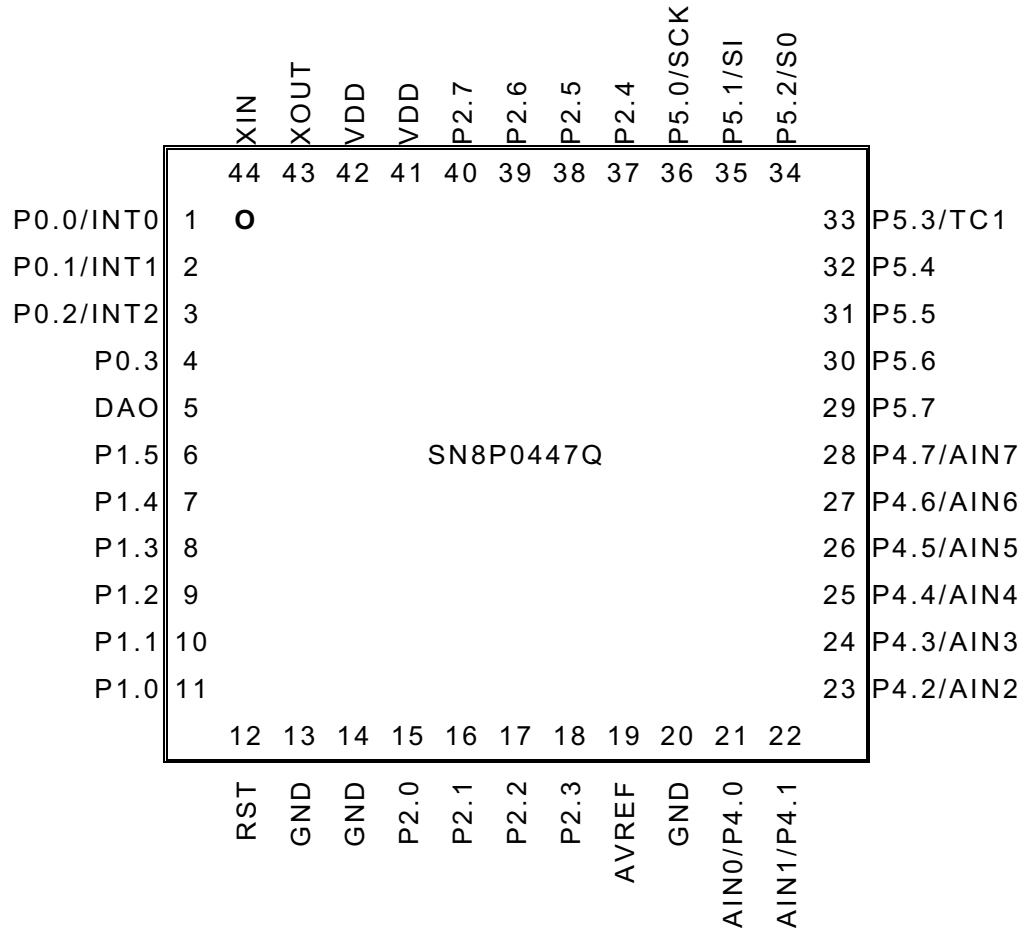
4. 包裝型式：

編號格式：SN8P04XXY

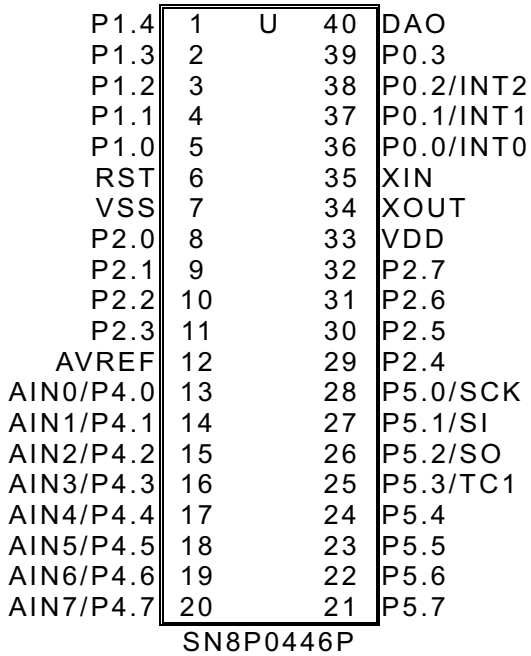
Y = Q > QFP，P > PDIP，K > SKDIP，S > SOP

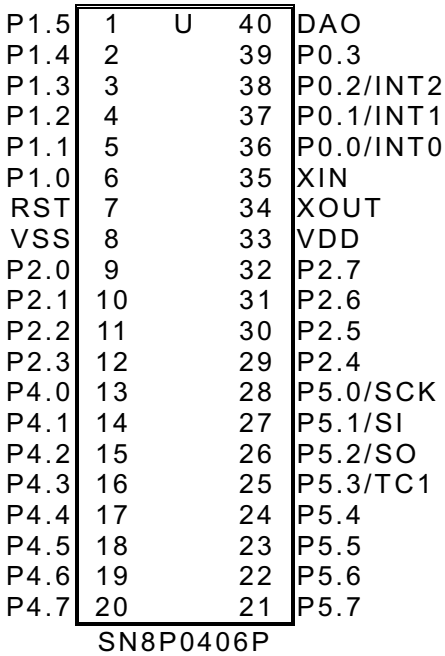
名稱	P1	P2	P4	P5	AD	DAO	SIO
SN8P0447Q	P1.0~P1.5	P2.0~P2.7	P4.0~P4.7	P5.0~P5.7	AIN0~7	V	V
SN8P0446P	P1.0~P1.4	P2.0~P2.7	P4.0~P4.7	P5.0~P5.7	AIN0~7	V	V
SN8P0406P	P1.0~P1.5	P2.0~P2.7	P4.0~P4.7	P5.0~P5.7	-	V	V
SN8P0444K	P1.0~P1.1	-	P4.0~P4.7	P5.0~P5.7	AIN0~7	V	V
SN8P0404K	P1.0~P1.3	-	P4.0~P4.7	P5.0~P5.7	-	-	V
SN8P0434S	P1.0~P1.2	-	P4.0~P4.7	P5.0~P5.7	-	V	V
SN8P0443S	P1.0~P1.1	-	P4.0~P4.5	P5.0~P5.5	AIN0~5	V	V
SN8P0403S	P1.0~P1.3	-	P4.0~P4.5	P5.0~P5.5	-	-	V

44PIN：

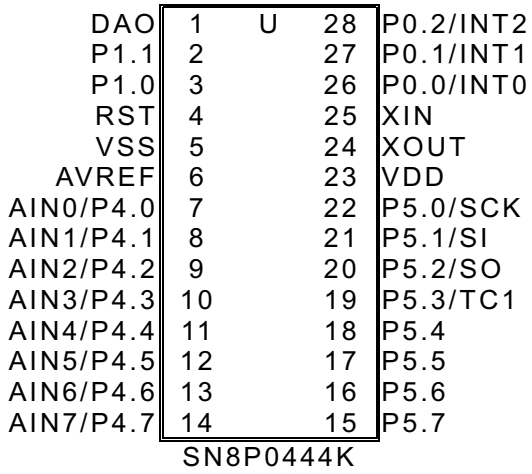

XIN
XOUT
VDD
VDD
P2.7
P2.6
P2.5
P2.4
P5.0/SCK
P5.1/SI
P5.2/SO
44 43 42 41 40 39 38 37 36 35 34
P0.0/INT0 1
P0.1/INT1 2
P0.2/INT2 3
P0.3 4
DAO 5
P1.5 6
P1.4 7
P1.3 8
P1.2 9
P1.1 10
P1.0 11
33 P5.3/TC1
32 P5.4
31 P5.5
30 P5.6
29 P5.7
28 P4.7/AIN7
27 P4.6/AIN6
26 P4.5/AIN5
25 P4.4/AIN4
24 P4.3/AIN3
23 P4.2/AIN2
SN8P0447Q
12 13 14 15 16 17 18 19 20 21 22
RST
GND
GND
P2.0
P2.1
P2.2
P2.3
AVREF
GND
AIN0/P4.0
AIN1/P4.1

40PIN：

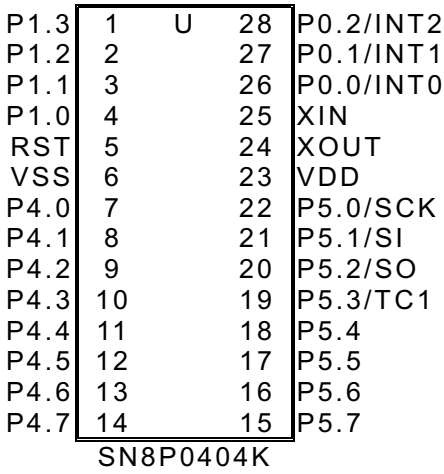

P1.4 1 U 40 DAO
P1.3 2 39 P0.3
P1.2 3 38 P0.2/INT2
P1.1 4 37 P0.1/INT1
P1.0 5 36 P0.0/INT0
RST 6 35 XIN
VSS 7 34 XOUT
P2.0 8 33 VDD
P2.1 9 32 P2.7
P2.2 10 31 P2.6
P2.3 11 30 P2.5
AVREF 12 29 P2.4
AIN0/P4.0 13 28 P5.0/SCK
AIN1/P4.1 14 27 P5.1/SI
AIN2/P4.2 15 26 P5.2/SO
AIN3/P4.3 16 25 P5.3/TC1
AIN4/P4.4 17 24 P5.4
AIN5/P4.5 18 23 P5.5
AIN6/P4.6 19 22 P5.6
AIN7/P4.7 20 21 P5.7
SN8P0446P


P1.5 1 U 40 DAO
P1.4 2 39 P0.3
P1.3 3 38 P0.2/INT2
P1.2 4 37 P0.1/INT1
P1.1 5 36 P0.0/INT0
P1.0 6 35 XIN
RST 7 34 XOUT
VSS 8 33 VDD
P2.0 9 32 P2.7
P2.1 10 31 P2.6
P2.2 11 30 P2.5
P2.3 12 29 P2.4
P4.0 13 28 P5.0/SCK
P4.1 14 27 P5.1/SI
P4.2 15 26 P5.2/SO
P4.3 16 25 P5.3/TC1
P4.4 17 24 P5.4
P4.5 18 23 P5.5
P4.6 19 22 P5.6
P4.7 20 21 P5.7
SN8P0406P

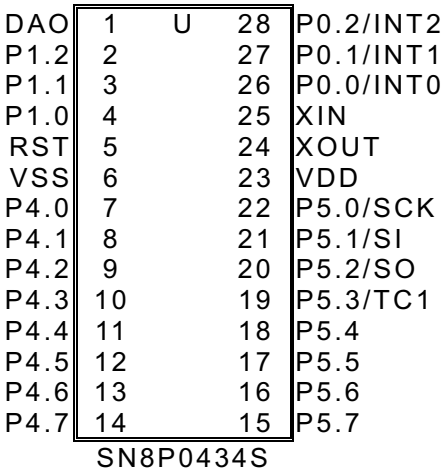
28PIN：



SN8P0444K

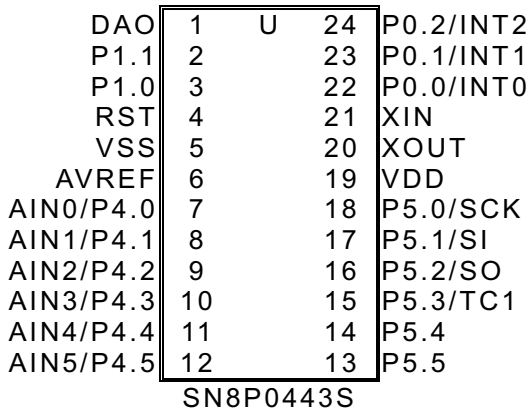


SN8P0404K

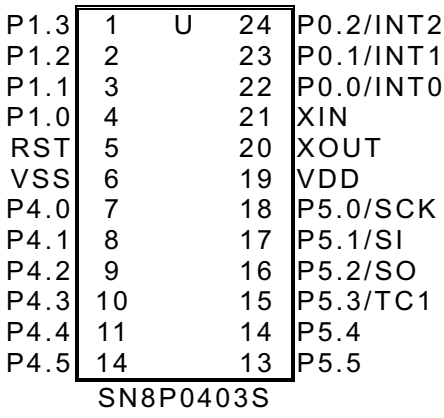


SN8P0434S

24PIN：



SN8P0443S



SN8P0403S

5. 腳位的說明：

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’ 。
XOUT，XIN	I/O	高速振盪器輸入腳位。
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)。
P0.1/INT1	I	P0.1 與 INT1 觸發腳位，有系統喚醒功能(史密特觸發結構)。
P0.2/INT2	I	P0.2 與 INT2 觸發腳位，有系統喚醒功能(史密特觸發結構)。
P0.3	I	P0.3 輸入腳位，有系統喚醒功能(史密特觸發結構)。
P1.0~P1.5	I/O	P1.2~P1.5 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。
P4.0~P4.7	I/O	P4.0~P4.7 雙向腳位。
P5.0/SCK	I/O	P5.0 雙向腳位，SIO 時脈(clock)輸入。
P5.1/SI	I/O	P5.1 雙向腳位，SIO 資料(data)輸入。
P5.2/SO	I/O	P5.2 雙向腳位，SIO 資料(data)輸出。
P5.3/TC1	I/O	P5.3 雙向腳位，TC1 ÷2 訊號輸出。
P5.4~P5.7	I/O	P5.4~P5.7 雙向腳位。
AVREF	I	ADC 參考電壓輸入腳位。
AIN0~AIN7	I	ADC 轉換器類比訊號輸入腳位。
DAO	O	DAC 信號輸出腳位。

第二章. 位址(ADDRESS)

1. 程式記憶體 (ROM) :

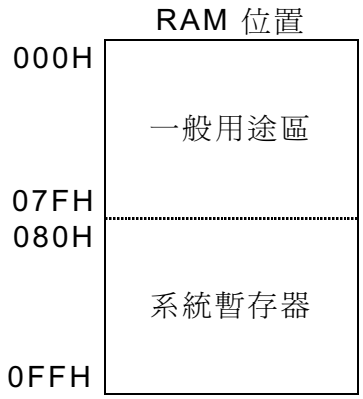
SN8P04XX 系列提供可定址的程式記憶體高達 4K*16 bits，及透過 12-bit 程式計數器(PC)取回指令，也可利用 ROM code 暫存器(R、X、Y、Z)去查閱 ROM 資料。所有的程式記憶體被分爲 2 個編碼區，從 0000H 至 000FH 用來執行中斷引導(特別區)。第二區，從 0010H 至 0FFFH 用來儲存指令程式碼及查尋表格資料，0FFFH 已被保留，無法由程式控制。

ROM 位址	
0000H	系統重置引導
0001H	
0002H	
0003H	
0004H	保留區
0005H	‘’
0006H	‘’
0007H	‘’
0008H	中斷引導
0009H	.
000AH	.
.	.
0010H	一般用途區
.	.
.	.
.	.
07FFH	.
0800H	一般用途區
.	.
.	.
.	.
.	.
0FFEH	.
0FFFH	保留區

2. 資料記憶體 (RAM) :

SN8P04XX 系列內建高達 128*8 bits(00H~7FH)資料記憶體，作爲儲存一般用途資料，除此之外，記憶體位址 80H~FFH 分配爲特殊暫存器應用。

2.1. RAM BANK 位置 :



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																
2																
3																
4																
5																
6																
7																
8	L	H	R	Z	Y	X	PFLAG	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B	DAM	ADM	ADB	ADR	SIOM	SIOR	SIOB	-	-	-	-	-	-	-	-	-
C	P1W	P1M	P2M	-	P4M	P5M	-	-	INTRQ	INTEN	OSCM	-	-	-	PCL	PCH
D	P0	P1	P2	-	P4	P5	-	-	T0M	T0C	TC0M	TC0C	TC1M	TC1C	TC1R	STKP
E	P0UR	P1UR	P2UR	-	P4UR	P5UR	@HL	@YZ	-	-	-	-	-	-	-	-
F	STK7	STK7	STK6	STK6	STK5	STK5	STK4	STK4	STK3	STK3	STK2	STK2	STK1	STK1	STK0	STK0

L，H= 工作暫存器及@HL 定址暫存器。

X= 工作暫存器。

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 。

PFLAG= 特殊旗標暫存器。

ADM= ADC 模式暫存器。

ADB= ADC 資料緩衝區。

DAM= DAC 模式暫存器。

ADR= ADC 解析度選擇暫存器 。

P1W= 埠 1 喚醒暫存器。

PnUR= 埠 n 昇壓(PULL-UP) 暫存器。

PnM= 埠 n 輸入/ 輸出模式暫存器。

Pn= 埠 n 資料緩衝區。

INTRQ= 中斷需求暫存器。

OSCM= 振盪器模式暫存器。

T0M= 計時器 0 模式暫存器。

T0C= 計時器 0 計數暫存器。

TC1M= 計時器/計數器 1 模式暫存器。

TC1C= 計時器/計數器 1 計數暫存器。

INTEN= 中斷致能暫存器。

PCH，PCL= 程式計數器。

TC0M= 計時器/計數器 0 模式暫存器。

TC0C= 計時器/計數器 0 計數暫存器。

STKP= 堆疊指標緩衝區。

TC1R= 計時器/計數器 1 自動重新載入資料緩衝區。

SIOM= SIO 模式控制暫存器 。

SIOR= SIO 時脈重新載入緩衝區。

SIOB= SIO 資料緩衝區。

@HL= RAM HL 間接定址索引指標。

@YZ= RAM YZ 間接定址索引指標。

STK0~STK7= 堆疊 0 ~ 堆疊 7 緩衝區。

2.3. 系統暫存器表格 :

位址	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	R/W	備註
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
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	XBIT7	XBIT6	XBIT5	XBIT4	XBIT3	XBIT2	XBIT1	XBIT0	R/W	X
086H	-	-	-	-	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	-	-	-	-	-	-
0B0H	DAENB	DAB6	DAB5	DAB4	DAB3	DAB2	DAB1	DAB0	R/W	DAM 資料暫存器
0B1H	ADENB	ADS	EOC	GCHS	-	CHS2	CHS1	CHS0	R/W	ADM 模式暫存器
0B2H	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	R	ADB 資料緩衝
0B3H	-	ADCKS	ADLEN	0	ADB3	ADB2	ADB1	ADB0	R/W	ADR 暫存器
0B4H	SENB	START	SRATE1	SRATE0	-	SCKMD	SEDGE	TXRX	W	SIOM 模式暫存器
0B5H	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0	W	SIOR 重新載入緩衝區
0B6H	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0	R/W	SIOB 資料緩衝
0B7H	-	-	-	-	-	-	-	-	-	-
0C0H	-	-	P15W	P14W	P13W	P12W	P11W	P10W	R/W	P1W 喚醒暫存器
0C1H	-	-	P15M	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	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M	R/W	P4M I/O 指示
0C5H	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5M I/O 指示
0C6H	-	-	-	-	-	-	-	-	-	-
0C7H	-	-	-	-	-	-	-	-	-	-
0C8H	-	TC1IRQ	TC0IRQ	T0IRQ	SIOIRQ	P02IRQ	P01IRQ	P00IRQ	R/W	INTRQ
0C9H	-	TC1IEN	TC0IEN	T0IEN	SIOIEN	P02IEN	P01IEN	P00IEN	R/W	INTEN
0CAH	-	WDRST	WDRATE	CPUM1	CPUM0	-	-	HXWUP	R/W	OSCM
0CBH	-	-	-	-	-	-	-	-	-	-
0CCH	-	-	-	-	-	-	-	-	-	-
0CDH	-	-	-	-	-	-	-	-	-	-
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH	-	-	-	-	PC11	PC10	PC9	PC8	R/W	PCH
0D0H	-	-	-	-	P03	P02	P01	P00	R/W	P0 資料緩衝
0D1H	-	-	P15	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	P47	P46	P45	P44	P43	P42	P41	P40	R/W	P4 資料緩衝
0D5H	P57	P56	P55	P54	P53	P52	P51	P50	R/W	P5 資料緩衝
0D6H	-	-	-	-	-	-	-	-	-	-
0D7H	-	-	-	-	-	-	-	-	-	-
0D8H	T0ENB	T0RATE2	T0RATE1	T0RATE0	-	-	-	-	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C

0DAH	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	-	-	-	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS	ALOAD	TC1OUT	-	R/W	TC1M
0DDH	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0	R/W	TC1C
0DEH	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0	R/W	TC1R
0DFH	GIE	-	-	-	STKPB3	STKPB2	STKPB1	STKPB0	R/W	STKP 堆疊指示
0E0H	-	-	-	-	P03R	P02R	P01R	P00R	W	P0UR
0E1H	-	-	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E2H	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R	W	P2UR
0E3H	-	-	-	-	-	-	-	-	-	-
0E4H	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R	W	P4UR
0E5H	P57R	P56R	P55R	P54R	P53R	P52R	P51R	P50R	W	P5UR
0E6H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL 索引指示
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ 索引指示
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R	STK7L
0F1H	-	-	-	-	S7PC11	S7PC10	S7PC9	S7PC8	R	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R	STK6L
0F3H	-	-	-	-	S6PC11	S6PC10	S6PC9	S6PC8	R	STK6H
“	“	“	“	“	“	“	“	“	“	“
“	“	“	“	“	“	“	“	“	“	“
“	“	“	“	“	“	“	“	“	“	“
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R	STK1L
0FDH	-	-	-	-	S1PC11	S1PC10	S1PC9	S1PC8	R	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R	STK0L
0FFH	-	-	-	-	S0PC11	S0PC10	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 WDRST 宣告爲 FWDRST，CPUM1 宣告爲 FCPUM1，CPUM0 宣告爲 FCPUM0。

3. 累積器 (ACC) :

ACC(程式中稱爲 A)是一個 8bit 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)、溢位或借位(C 或 DC)產生，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：

```
JC      MACRO   ADRS
        B0BTS0  FC
        JMP     <ADRS>
        ENDM
```

C=0，跳躍到 ADRS 位址：

```
JNC     MACRO   ADRS
        B0BTS1  FC
        JMP     <ADRS>
        ENDM
```

DC=1，跳躍到 ADRS 位址：

```
JDC     MACRO   ADRS
        B0BTS0  FDC
        JMP     <ADRS>
        ENDM
```

DC=0，跳躍到 ADRS 位址：

```
JNDC    MACRO   ADRS
        B0BTS1  FDC
        JMP     <ADRS>
        ENDM
```

Z=1，跳躍到 ADRS 位址：

```
JZ      MACRO   ADRS
        B0BTS0  FZ
        JMP     <ADRS>
        ENDM
```

Z=0，跳躍到 ADRS 位址：

```
JNZ     MACRO   ADRS
        B0BTS1  FZ
        JMP     <ADRS>
        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：

位於資料記憶體中 RAM bank0 80H 至 86H 儲存特殊定義的暫存器(H、L、R、X、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、X、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 H、L、Y 及 Z 暫存器作間接存取。

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

4.1. H,L 暫存器：

H 及 L 暫存器是 8-bit 緩衝區，有兩個主要功能，一是當作工作暫存器使用，另一個功能是當作資料指標，應用於存取 RAM 資料。@HL 是位於 RAM bank0 E6H 中的資料指標 0 索引(data point_0 index)緩衝區，當透過 ACC 去讀/寫資料時，使用 H 及 L 暫存器做 RAM 位址的定址工作，H 暫存器中較低的 4 個位元指示出 RAM BANK 編號，L 暫存器指示出 RAM 位址編號，H 暫存器中較高的 4 個位元在 RAM 間接存取模式中被截斷。

例如：若欲從 BANK_0 20H 位址讀取資料，可利用間接定址模式存取資料，如下所示：

```
B0MOV   H,#00H      ; 在 H 暫存器中設定 RAM bank0
B0MOV   L,#20H      ; 在 L 暫存器中設定位置 20H
B0MOV   A,@HL       ; 讀取 0020H 資料至 ACC 中
INCMS   L           ; 在 L 暫存器中設定位置 21，L 直接加 1，可直接讀取下一
                    ; 個 RAM 中的資料，依此類推就可連續讀取 RAM 資料
NOP
B0MOV   A,@HL       ; 讀取 0021H 資料至 ACC 中
```

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

```
INC_DP0X  MACRO
          INCMS     L       ; L=L+1
          JMP       @F      ; L 無溢位產生
          INCMS     H       ; H =H+1
          NOP
@@:
          ENDM
```

當增加 L 數值時，同時判斷 L 是否有大於 FFH，若大於 FFH，則代表超出 H 範圍，H 需修正爲 H+1，才不會讀取錯誤 RAM 位址，利用 INC_DP0X 巨集修正上述程式：

```
B0MOV   H,#00H      ; 在 H 暫存器中設定 RAM bank0
B0MOV   L,#20H      ; 在 L 暫存器中設定位置 20H
B0MOV   A,@HL       ; 讀取 0020H 資料至 ACC 中
INC_DP0X            ; RAM 位址加 1，準備讀取下一筆資料
                    ; 依此類推就可連續讀取 RAM 資料
NOP
B0MOV   A,@HL       ; 讀取 0021H 資料至 ACC 中
```

4.2. 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 暫存器類似 H 及 L 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，可結合 X 暫存器，在查詢 ROM 資料時作 ROM 位址的定址工作，此功能可應用於表格查詢。

4.3. 查詢表格說明：

在查詢 ROM 資料功能中，X 暫存器被指向最高的 8 個位元，Y 暫存器指向中間的 8 個位元，而 Z 暫存器指向最低的 8 個位元，在執行 MOVC 指令後，ROM 中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   X,#TABLE1$H     ;設定查詢表格的高位址
        B0MOV   Y, #TABLE1$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
        .       .               ;查詢表格結束
        .       .               ;
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 X、Y 範圍(BOUNDARY)，需對 X、Y、Z 加以處理，利用以下巨集便可解決此問題：

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

        INCMS   Y       ;Y=Y+1
        JMP     @F      ;Y 無溢位產生

        INCMS   X       ;X=X+1
        NOP
@@:
        ENDM
```

利用 INC_XYZ 巨集修正上述程式：

```
        B0MOV   X,#TABLE1$H     ;設定查詢表格的高位址
        B0MOV   Y, #TABLE1$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_XYZ                 ;查詢下一個 ROM 的資料
        NOP                     ;
        MOVC                    ;查詢資料，R = 51H，ACC = 05H
        .       .               ;查詢表格結束
        .       .               ;
TABLE1: DW      0035H           ;定義一個字元 (16 bits)資料
        DW      5105H           ;
        DW      2012H           ;
        .       .
```

4.4. 定址種類：

SN8P04XX 系列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(H/L 或 Y/Z)設定一個位址，利用 MOV 指令在 ACC 和@HL/@YZ 暫存器中讀/寫資料(B0MOV A,@HL，B0MOV @HL,A)。

4.4.1. 立即定址模式：

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

4.4.2. 直接定址塻式：

```
        MOV     A,12H           ;取得一個在 BANK 0 12H 位置的內容，儲存在 ACC
```

4.4.3. 間接定址模式含@HL 暫存器：

```
        CLR     H               ;清除 H 暫存器
        B0MOV   L,#12H          ;設置一個立即資料 12H 至 L 暫存器
        B0MOV   A, @HL          ;利用資料指標@HL 從 RAM 的 012H 位置讀一個資
                                ;料，儲存至 ACC 中
```

4.4.4. 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 位置讀一個資
                                ;料，儲存至 ACC 中
```

5. 程式計數器：

程式計數器(PC)是一個 12-bit 的二進制計數器，被分割爲高位元組的 4 個位元於 PCH 及低位元組的 8 個位元於 PCL，如下表格所示。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，當執行 CALL 和 JMP 指令時，PC 同時會被特殊位址所取代，CALL 及 JMP 指令執行時，目的位址被放置在 BIT0 ~BIT11。

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
PCH	-	-	-	-	PC11	PC10	PC9	PC8

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
PCL	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

PC 初值 = XXXX 0000 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	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：

SN8P04XX 系列的堆疊緩衝區可達 8 層，每一層有 12-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 1111

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
STKP	GIE	-	-	-	STKPB3	STKPB2	STKPB1	STKPB0

STKn 初值= XXXX XXXX XXXX XXXX, STKn = STKnH + STKnL (n = 7H ~ 0H)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
STKnH	-	-	-	-	SnPC11	SnPC10	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：

SN8P04XX 系列在執行中斷時，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
```

例如：推入(PUSH) ACC 及工作暫存器

```
PUSHBUF：
                PUSH                    ；推入工作暫存器
                B0MOV   ACCBUF,A        ；儲存 ACC 至 ACCBUF 中
```

例如：拉出(POP) ACC 及工作暫存器

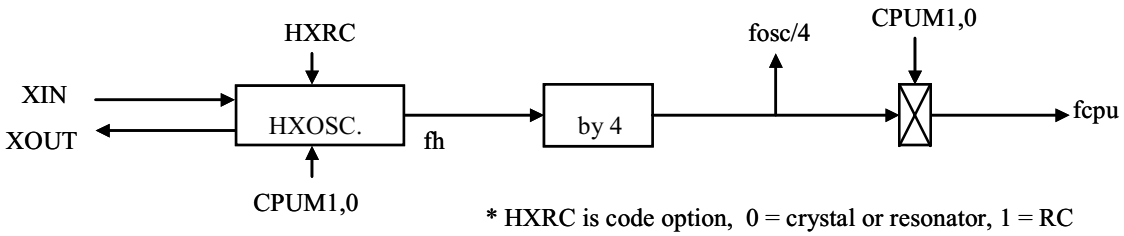
```
POPBUF：
                B0MOV   A,ACCBUF        ；還原 ACC
                POP                     ；拉出 工作暫存器
```

注意：PUSH 及 POP 有先進後出的順序，確保數值正確，所以先做 PUSH 再儲存 ACC，先還原 ACC 再做 POP 動作。

第三章. 振盪電路

1. 振盪器 :

SN8P04XX 系列採用單一振盪器，在不同應用時，使用者可從晶體振盪器、陶製振盪器或 RC 電路選擇適當的振盪架構，產生系統時脈源。當系統閒置時，可使用睡眠(power down)模式，讓晶片進入低耗電的閒置狀態，設定 CPUMx = 01，系統由一般模式切換爲睡眠模式，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，CPUMx 自動被設爲 00。



1.1. 振盪器暫存器 :

OSCM 初值 = XXX0 0X00

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OSCM	-	WDRST	WDRATE	CPUM1	CPUM0	-	-	HXWUP

HXWUP：高速振盪器的振盪啓動穩定時間控制位元 。0 = 18th，1 = 15th。
CPUMx：CPU 操作模式控制位元。00 = 操作中(operating)，01 = 睡眠模式(sleep / power down)，10 = 省電模式(green)，11 =保留的(reserved)。

注意：在改變 CPU 操作模式後，再執行一個 NOP 指令。

例如：切換 CPU 操作由一般模式至睡眠模式。

```
N2SLEEP:  B0BCLR  FHXWUP      ;設定系統下一次被喚醒的振盪啓動穩定時間
          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

          NOP                 ;當系統喚醒觸發信號發生(P0、P1)，程序有開始執
                              ;行， CPUM1、CPUM0 爲 00。
          .       .
          .       .
          .       .
```

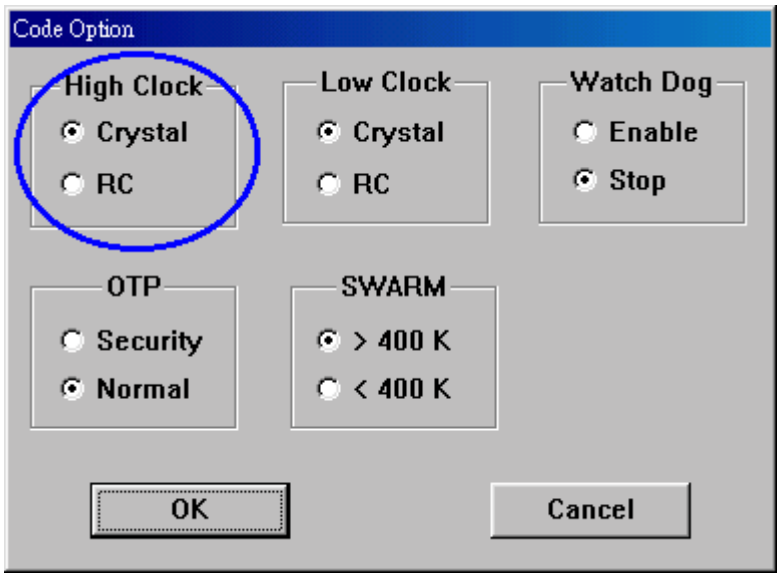
操作模式敘述：

模式	一般(normal)	省電(green)	睡眠(sleep，power down)	備註
HX 振盪器	運轉中	運轉中	停止	
CPU 指令	執行中	停止	停止	
T0 計時器	動作	動作	不動作	* 動作可由程序控制
TC0/TC1 計時器	動作	不動作	不動作	
WDOG 計時器	動作	不動作	不動作	程序控制
內部中斷	全部動作	T0 動作	全部不動作	
外部中斷	全部動作	全部動作	全部不動作	
喚醒源	-	P0、P1、T0、重置	P0、P1、重置	P0、P1 接受 ‘L’

注意：在省電模式，T0 觸發器信號可切換 CPU 返回一般模式操作，此時 T0IEN=1、T0ENB=1、T0IRQ=1。

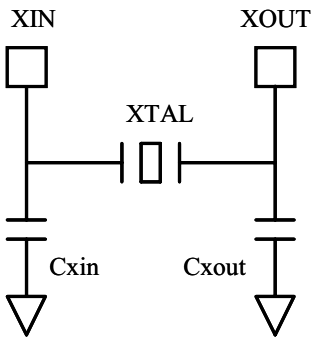
2. 振盪器應用電路接法：

SN8P04XX 系列提供 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。



2.1. 晶體振盪器：

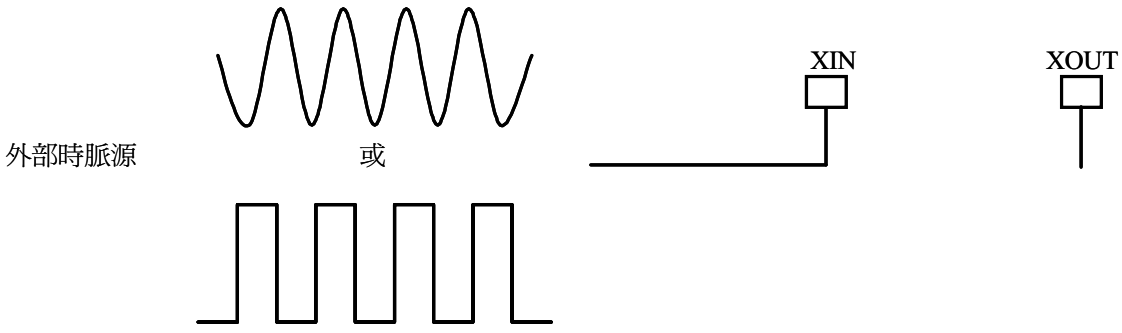
晶體振盪器分爲石英振盪器及陶瓷振盪器，接法相同，SN8P04XX 系列選擇的晶體振盪器頻率最高爲 10MHz，接法如下所示：



- XTAL 爲石英振盪器或陶瓷振盪器，其兩端接至 SN8P04XX 系列的 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。

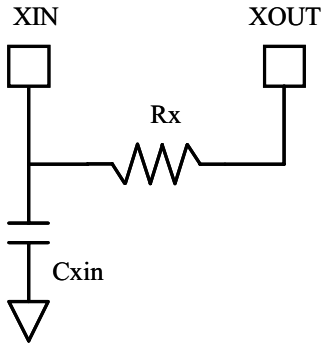
2.2. 外部時脈輸入：

SN8P04XX 系列可接受外部時脈，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。



2.3. 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(5Vdc)
20PF	3.3K	7.27MHz
	5K	5.52MHz
	7.5K	4.05MHz
	10K	3.32MHz
	100K	404.04KHz
100PF	3.3K	1.99MHz
	7.5K	1.45MHz
	10K	805.37KHz
	100K	89.39KHz
300PF	3.3K	806.72KHz
	7.5K	570.07KHz
	10K	317.46KHz
	100K	33.52KHz

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   P5.7        ;設 P5.7 爲 'H' ，此行爲 1 個指令週期
NOP                  ;此行爲 1 個指令週期
NOP                  ;此行爲 1 個指令週期
B0BCLR   P5.7        ;設 P5.7 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
```

上述程序產生一個週期爲 6 個指令週期的方波輸出，利用示波器由 P5.7 量測週期寬度，若方波週期寬度爲 $6*10^{-6}$ sec，計算方式如下：

$6*10^{-6} \div 6 = 10^{-6}$ sec ；單一指令週期寬度

$Fcpu = 1/10^{-6} = 1MHz$ ；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

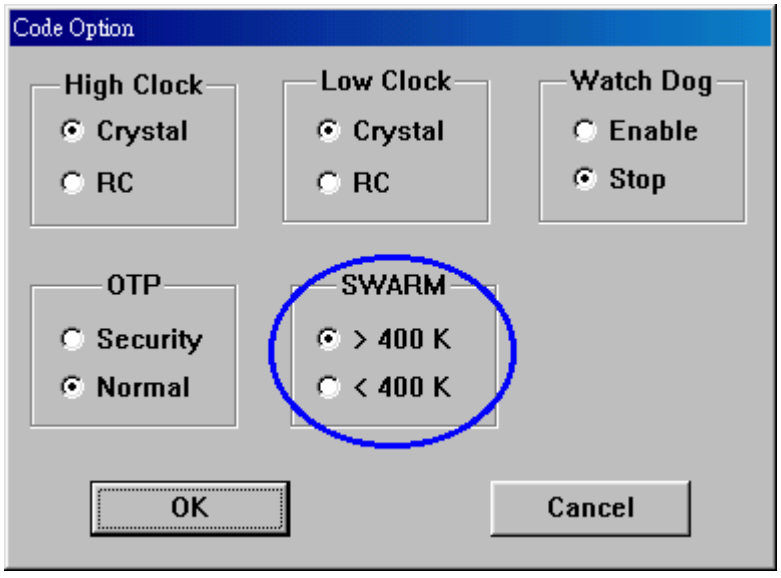
3. 振盪啓動穩定時間(WARM UP TIME) :

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。SN8P04XX 系列提供 6 種振盪啓動穩定時間，應用在不同型態振盪器系統，$fhxosc \div 2^{18}$ 及 $fhxosc \div 2^{15}$ 設計在高速振盪器的應用， $fhxosc \div 2^{14}$ 及 $fhxosc \div 2^{11}$ 設計在低速振盪器的應用。在實際的應用，使用者可設置 SWARM 控制位元及 OSCM 暫存器的 HXWUP 位元選擇適當的振盪啓動穩定時間，以結合停止高速振盪器前的振盪器起動時間；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 或 40.9 ms)。

對應於 HX_osc 的振盪啓動穩定時間：

HX_osc	SWARM	HXWUP	喚醒時間
3.58 MHz	0	0	$fhxosc \div 2^{18}$ = 73.2 ms
		1	$fhxosc \div 2^{15}$ = 9 ms
400 KHz	1	0	$fhxosc \div 2^{14}$ = 40.9 ms
		1	$fhxosc \div 2^{11}$ = 5.1 ms
40 KHz		0	$fhxosc \div 2^{14}$ = 40.9 ms
		1	$fhxosc \div 2^{11}$ = 51 ms

註：SWARM 選項在 SONIX 組譯程式中，400K 代表 Fosc。



4. 看門狗計時器 (WDC) :

看門狗計時器是一個 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，使用者可藉由修正 OSCM 的 WDRATE 位元控制 WDC；在省電模式及睡眠模式下，此計時器爲禁能狀態。

OSCM 初值 = XXX0 0X00

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OSCM	-	WDRST	WDRATE	CPUM1	CPUM0	-	STPHX	HXWUP

WDRATE：WDC 時脈源選擇位元。0 =14th， 1 =8th。
WDRST：看門狗計時器歸零控制位元。0 = 不歸零， 1 = 計時器歸零。

範例：

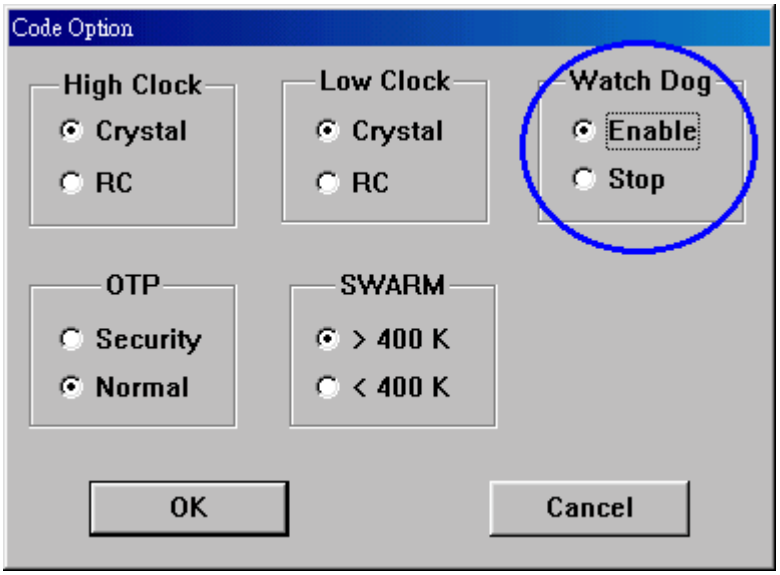
```
MAIN:                           ; 主程序開頭
        B0BSET  FWDRST          ; 歸零看門狗計時器
        CALL    SUB10           ; 子程序 1
        CALL    SUB20           ; 子程序 2
        CALL    SUB30           ; 子程序 3
        .       .
        .       .
        JMP     MAIN.           ; 回主程序
```

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。

WDTMR 計算時間 (高速模式， Fcpu = 3.58MHz / 4)

WDRATE	看門狗計時器溢位時間
0	Fcpu ÷ 2^{14} ÷ 16 = 293 ms
1	Fcpu ÷ 2^{8} ÷ 16 = 4.5 ms

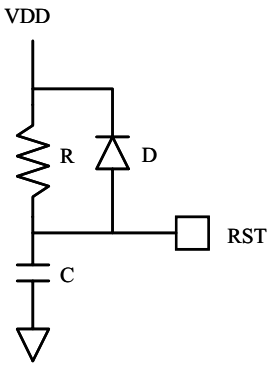
注意：看門狗計時器可由 SONIX 組譯程式中，選擇致能或禁能。



5. 外部重置保護電路：

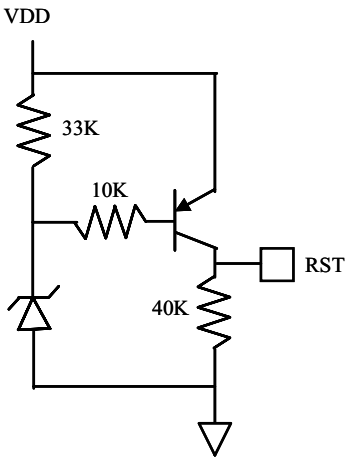
SN8P04XX 系列有外部重置偵測腳位(RST)，若 VDD 電位上升太慢或電源不穩時，RST 偵測到 ‘L’ ，代表外部重置線路輸出因 VDD 錯誤而輸出 ‘L’ ，系統停止工作，當 RST 偵測到 ‘H’ 時，系統執行重置工作，準備進入一般模式正常工作。對於 VDD 電壓準位在何種條件下爲正常電位，須靠外部重置電路處理，偵測 VDD 電位是否正常，配合不同應用環境，提供以下 3 種常用重置電路：

5.1. RC 重置電路：



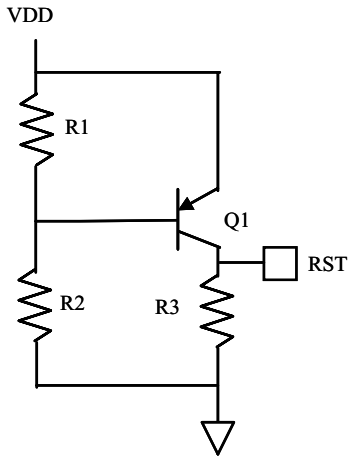
電阻 R 的數值所造成壓降建議不要超過 0.2V，而 RST 腳位的漏電流爲 1uA，求得 R 爲 100K ohm，所以 R 的選用數值需小於 100K ohm，二極體的功用是加快電容放電時間。在產品開發測試階段，使用者可以在 RST 及 GND 之間接一個重置鍵，供系統手動重置之用，方便測試。

5.2. 穩壓二極體重置電路：



VDD 電壓大於(VZ+0.7)V，Q1 導通，VDD 送至 RST 端，系統開始重置工作，當 VDD 電壓小於(VZ+0.7)V，代表電源不穩定，系統停止工作，等待 VDD 正常，再次重置系統，此重置電路動作精確，但成本較高。

5.3. 電阻分壓方式重置電路：



此電路是穩壓二極體重置電路的衍生，將穩壓二極體以電阻代替，精確度較前者差，但價格較爲低廉。當 VDD 電位上升至足以導通 Q1 時，RST 偵測到 ‘H’ ，系統執行重置工作，若 VDD 電位不足以使 Q1 導通時，RST 偵測到 ‘L’ ，系統停止工作。

算式 1：

VDD* (R1÷(R1+R2)) ≧ 0.7V　　(Q1 導通，輸出至 RST 爲 ‘H’)

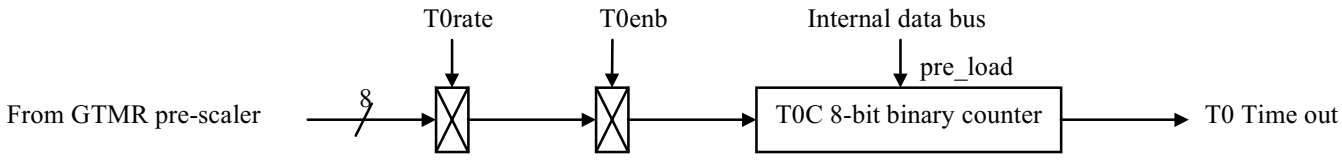
算式 2：

VDD* (R1÷(R1+R2)) < 0.7V　　(Q1 不導通，輸出至 RST 爲 ‘L’)

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

1. 基本計時器 (T0) :

基本計時器 T0 是一個二進制的 8-bit 計數器，可從 T0M 暫存器選擇 T0C 的輸入時脈，來計算一個精確的時間。若 T0 計時器發生溢位(從 FFH 至 00H)，基本計時器將會繼續計數，並且發布一個暫停信號，觸發 T0 中斷，請求中斷服務。



1.1. T0M 模式暫存器 :

T0M 初值 = 0XXX XXXX

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
T0M	T0ENB	T0RATE2	T0RATE1	T0RATE0	-	-	-	-

T0ENB：T0 計時器控制位元。 0 = 禁能，1 = 致能。
T0RATE：T0 的時脈資源選擇位元，產生 GTMR 預除信號。
000 = Fcpu/256， 001 = Fcpu/128，...，110 = Fcpu/4，111 = Fcpu/2。

1.2. T0C 計數暫存器：

T0C 初值 = XXXX XXXX

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
T0C	X	X	X	X	X	X	X	X

T0 中斷最大間隔時間如下所示：

T0RATE	T0CLOCK	高速率模式 (FCPU = 3.58MHz / 4)
000	Fcpu/256	73.2 ms
001	Fcpu/128	36.6 ms
010	Fcpu/64	18.3 ms
011	Fcpu/32	9.15 ms
100	Fcpu/16	4.57 ms
101	Fcpu/8	2.28 ms
110	Fcpu/4	1.14 ms
111	Fcpu/2	0.57 ms

註：T0C 暫存器的初值計算公式如下：

T0C 初值 = 256 - (T0 中斷間隔時間 *輸入時脈)

例如：3.58MHz 高速模式下，在 T0 中斷設置 10ms 的間隔時間。
T0C 數值(74H) = 256 - (10ms * Fcpu / 64)

T0C 計算說明：

$$T0C\ (\ 74H\ )\ = 256 - (10 / 10^{-3} * 3.58 * 10^{6} / 4 / 64\ )$$

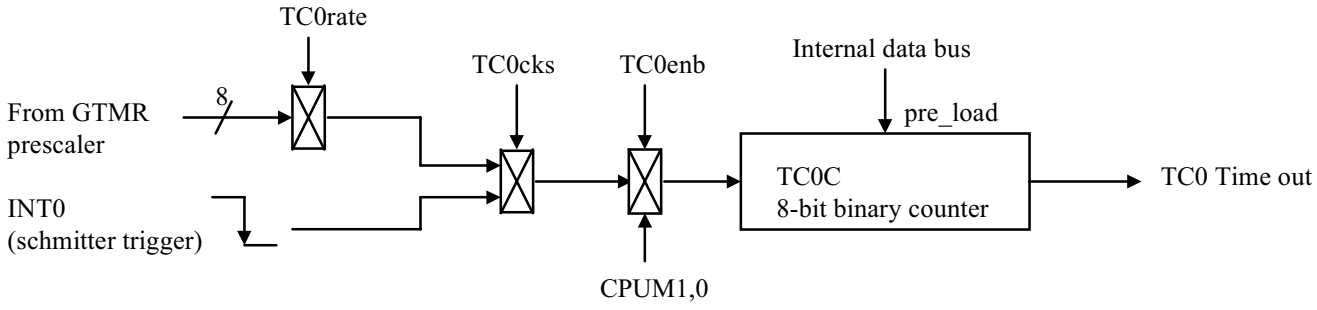
1.3. 設定程序說明：

T0C 數值修正前必須先將 T0 計時器禁能：

```
B0BCLR    FT0IEN          ; 禁能 T0 中斷服務
B0BCLR    FT0ENB          ; 禁能 T0 計時器
MOV       A,#20H
B0MOV     T0M,A           ; 設置 T0 時脈 = Fcpu / 64
MOV       A,#74H
B0MOV     T0C,A           ; 設定 T0C 初值= 74H (設置 T0 間隔 = 10 ms)
B0BSET    FT0IEN          ; 致能 T0 中斷服務
B0BCLR    FT0IRQ          ; 清除 T0 中斷要求
B0BSET    FT0ENB          ; 致能 T0 計時器
```

2. 計時器/事件計數器(TC0)：

TC0 是一個二進制的 8-bit 計時器及事件計數器，TC0 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。在省電模式，這個計時器/事件計數器將爲禁能。



2.1. 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)。

2.2. 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

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：

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

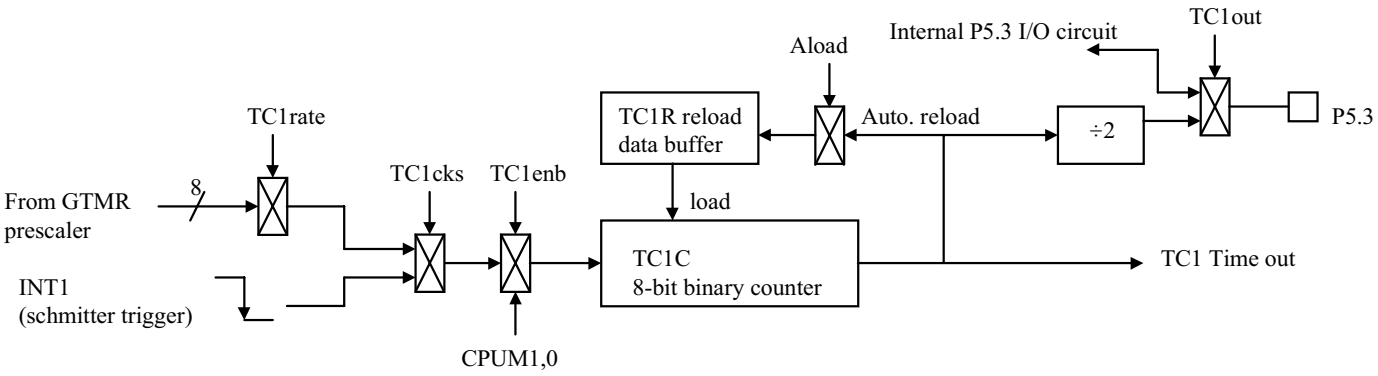
2.3. 設定程序說明 :

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
```

3. 計時器/事件計數器 (TC1) :

TC1 是一個二進制的 8-bit 計時器及事件計數器，具有自動重新載入功能。TC1 利用 TC1M 記錄器從 GTMR 或從外部的 INT1/P0.1 端點 (falling edge trigger)選擇 TC1C 的時脈源，計數一個精確時間。若 TC1 計時器發生一個溢位 (從 FFH 至 00H) ， TC1 會繼續計數，並且發布一個暫停信號，觸發 TC1 中斷，請求中斷服務。在省電模式，這個計時器/計數器將爲禁能。在一些應用中 (例如：InfraRed、IR、Buzzer)，需要利用溢位信號控制輸出埠，在這些例子中，控制程序會佔據許多 CPU 運算資源，因此，內建一組控制電路以減少 CPU 負載，TC1 動作流程如下所示：



3.1. TC1M 模式計錄器 :

TC1M 初值= 0XXX XX0X

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
TC1M	TC1ENB	TC1RATE2	TC1RATE1	TC1RATE0	TC1CKS	ALOAD	TC1OUT	-

TC1ENB：TC1 計數器致能位元。 0 = 禁能，1 = 致能。
TC1RATE：TC1 內部時脈選擇位元。
000 = Fcpu/256，001 = Fcpu/128， ... ，110 = Fcpu/4，111 = Fcpu/2。

TC1CKS：TC1 時脈源選擇位元。 0 = 內部時脈資源， 1 = 外部時脈資源(INT1/P0.1)。
ALOAD：自動重新載入控制位元 。 0 = 沒有自動重新載入，1 = 自動重新載入。
TC1OUT：TC1 暫停觸發信號輸出控制位元。 0 = 禁能 TC1 信號輸出及致能 P5.3 的 I/O 功能，
1 = 致能 TC1 的輸出信號及禁能 P5.3 的 I/O 功能。

3.2. TC1C 計時暫存器 :

TC1C 初值= XXXX XXXX

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
TC1C	X	X	X	X	X	X	X	X

3.3. TC1R 自動重新載入暫存器：

TC1R 初值 = XXXX XXXX

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
TC1R	X	X	X	X	X	X	X	X

註 1：TC1C 暫存器的初值計算

TC1C 初值= 256 - (TC1 中斷間隔時間* 輸入時脈)

3.4. 設定程序說明：

TC1C 數值修正前必須先將 TC1 計時器禁能：

```
B0BCLR    FTC1IEN          ; 禁能 TC1 的中斷服務
B0BCLR    FTC1ENB          ; 禁能 TC1  計時器
MOV       A,#10H           ;
B0MOV     TC1M,A           ; 選擇 TC1 時脈 = Fcpu / 128
MOV       A,#97H
B0MOV     TC1C,A           ; 設定 TC1 中斷時間= 15 MS
B0BSET    FTC1IEN          ; 致能 TC1 的中斷服務
B0BCLR    FTC1IRQ          ; 清除 TC1 中斷要求
B0BSET    FTC1ENB          ; 致能 TC1 計時器
```

3.5. TC1 自動重新載入功能範例說明：

TC1 的自動重新載入功能，就是當 TC1C 發生溢位時，系統自動將 TC1R 的數值載入 TC1C 中，使 TC1 以周期性運作，此功能可應用於週期性訊號輸出，例如：產生 1KHz 訊號驅動蜂音器(Beep tone)，使用 4MHz 振盪器。

```
SET_TC1：
            B0BCLR    FTC1IEN          ; 禁能 TC1 的中斷服務
            B0BCLR    FTC1ENB          ; 禁能 TC1 計時器

            MOV       A,#54H
            B0MOV     TC1M,A           ; 選擇 TC1 時脈 = Fcpu / 8
                                       ; 設定 TC1CKS = 0
                                       ; 設定 ALOAD = 1
                                       ; 設定 TC1OUT = 0
            MOV       A,#00H
            B0MOV     TC1C,A           ; 設定 TC1C 初值 = 0

            B0BCLR    FTC1IEN          ; 禁能 TC1 的中斷服務
            B0BCLR    FTC1IRQ          ; 清除 TC1 中斷要求
            B0BSET    FTC1ENB          ; 致能 TC1 計時器
```

```
ENTC1OUT：
        MOV     A,#194          ;
        B0MOV   TC1R,A          ；設定 TC1R 初值 = 194
        B0BSET  FTC1OUT         ；致能 TC1OUT
        .       .
        .       .
```

註：利用 TC1 由 P5.3 輸出時脈信號，不需要致能 TC1 中斷服務(TC1IEN)，只需要致能 TC1 計數器及 TC1OUT 控制，當 TC1OUT 輸出時，P5.3 將關閉一般 I/O 功能。

以下表格是利用 4MHz 振盪器，產生 TC1OUT ÷2 信號，選用的 TC1CLOCK 爲 FCPU/8：

TC1CLOCK=Fcpu/8					
TC1R	TC1OUT (KHz)	TC1OUT (Hz)	TC1R	TC1OUT (KHz)	TC1OUT (Hz)
1	0.245	245.098	11	0.255	255.102
2	0.246	246.063	12	0.256	256.148
3	0.247	247.036	13	0.257	257.202
4	0.248	248.016	14	0.258	258.264
5	0.249	249.004	15	0.259	259.336
6	0.250	250.000	16	0.260	260.417
7	0.251	251.004	17	0.262	261.506
8	0.252	252.016	18	0.263	262.605
9	0.253	253.036	19	0.264	263.713
10	0.254	254.065	20	0.265	264.831
21	0.266	265.957	31	0.278	277.778
22	0.267	267.094	32	0.279	279.018
23	0.268	268.240	33	0.280	280.269
24	0.269	269.397	34	0.282	281.532
25	0.271	270.563	35	0.283	282.805
26	0.272	271.739	36	0.284	284.091
27	0.273	272.926	37	0.285	285.388
28	0.274	274.123	38	0.287	286.697
29	0.275	275.330	39	0.288	288.018
30	0.277	276.549	40	0.289	289.352
41	0.291	290.698	51	0.305	304.878
42	0.292	292.056	52	0.306	306.373
43	0.293	293.427	53	0.308	307.882
44	0.295	294.811	54	0.309	309.406
45	0.296	296.209	55	0.311	310.945
46	0.298	297.619	56	0.313	312.500

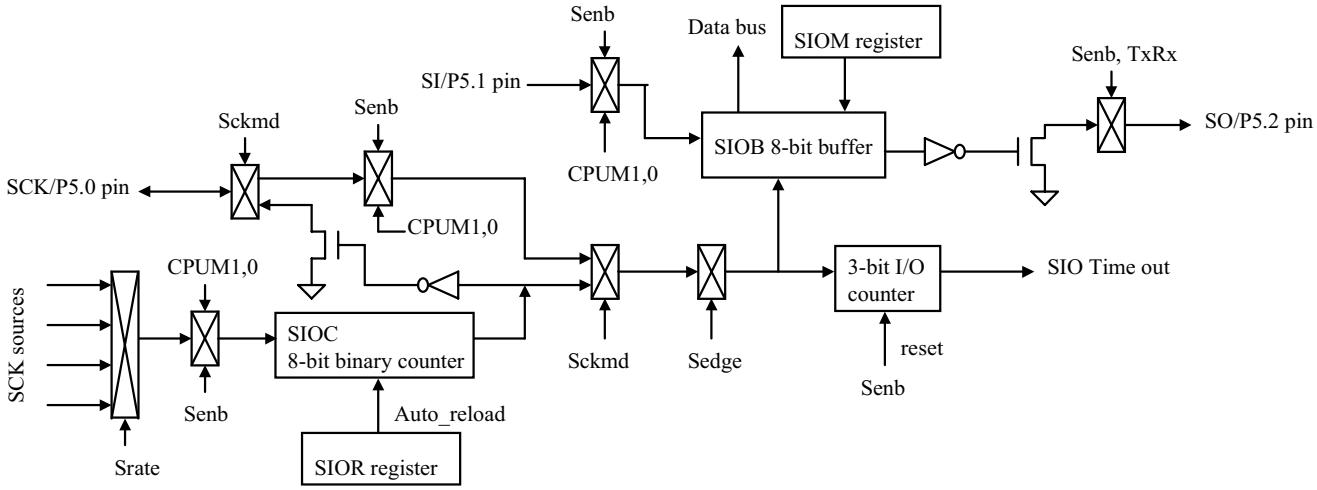
47	0.299	299.043	57	0.314	314.070
48	0.300	300.481	58	0.316	315.657
49	0.302	301.932	59	0.317	317.259
50	0.303	303.398	60	0.319	318.878
61	0.321	320.513	71	0.338	337.838
62	0.322	322.165	72	0.340	339.674
63	0.324	323.834	73	0.342	341.530
64	0.326	325.521	74	0.343	343.407
65	0.327	327.225	75	0.345	345.304
66	0.329	328.947	76	0.347	347.222
67	0.331	330.688	77	0.349	349.162
68	0.332	332.447	78	0.351	351.124
69	0.334	334.225	79	0.353	353.107
70	0.336	336.022	80	0.355	355.114
81	0.357	357.143	91	0.379	378.788
82	0.359	359.195	92	0.381	381.098
83	0.361	361.272	93	0.383	383.436
84	0.363	363.372	94	0.386	385.802
85	0.365	365.497	95	0.388	388.199
86	0.368	367.647	96	0.391	390.625
87	0.370	369.822	97	0.393	393.082
88	0.372	372.024	98	0.396	395.570
89	0.374	374.251	99	0.398	398.089
90	0.377	376.506	100	0.401	400.641
101	0.403	403.226	111	0.431	431.034
102	0.406	405.844	112	0.434	434.028
103	0.408	408.497	113	0.437	437.063
104	0.411	411.184	114	0.440	440.141
105	0.414	413.907	115	0.443	443.262
106	0.417	416.667	116	0.446	446.429
107	0.419	419.463	117	0.450	449.640
108	0.422	422.297	118	0.453	452.899
109	0.425	425.170	119	0.456	456.204
110	0.428	428.082	120	0.460	459.559
121	0.463	462.963	131	0.500	500.000
122	0.466	466.418	132	0.504	504.032
123	0.470	469.925	133	0.508	508.130
124	0.473	473.485	134	0.512	512.295

125	0.477	477.099	135	0.517	516.529
126	0.481	480.769	136	0.521	520.833
127	0.484	484.496	137	0.525	525.210
128	0.488	488.281	138	0.530	529.661
129	0.492	492.126	139	0.534	534.188
130	0.496	496.032	140	0.539	538.793
141	0.543	543.478	151	0.595	595.238
142	0.548	548.246	152	0.601	600.962
143	0.553	553.097	153	0.607	606.796
144	0.558	558.036	154	0.613	612.745
145	0.563	563.063	155	0.619	618.812
146	0.568	568.182	156	0.625	625.000
147	0.573	573.394	157	0.631	631.313
148	0.579	578.704	158	0.638	637.755
149	0.584	584.112	159	0.644	644.330
150	0.590	589.623	160	0.651	651.042
161	0.658	657.895	171	0.735	735.294
162	0.665	664.894	172	0.744	744.048
163	0.672	672.043	173	0.753	753.012
164	0.679	679.348	174	0.762	762.195
165	0.687	686.813	175	0.772	771.605
166	0.694	694.444	176	0.781	781.250
167	0.702	702.247	177	0.791	791.139
168	0.710	710.227	178	0.801	801.282
169	0.718	718.391	179	0.812	811.688
170	0.727	726.744	180	0.822	822.368
181	0.833	833.333	191	0.962	961.538
182	0.845	844.595	192	0.977	976.563
183	0.856	856.164	193	0.992	992.063
184	0.868	868.056	194	1.008	1008.065
185	0.880	880.282	195	1.025	1024.590
186	0.893	892.857	196	1.042	1041.667
187	0.906	905.797	197	1.059	1059.322
188	0.919	919.118	198	1.078	1077.586
189	0.933	932.836	199	1.096	1096.491
190	0.947	946.970	200	1.116	1116.071
201	1.136	1136.364	211	1.389	1388.889
202	1.157	1157.407	212	1.420	1420.455

203	1.179	1179.245	213	1.453	1453.488
204	1.202	1201.923	214	1.488	1488.095
205	1.225	1225.490	215	1.524	1524.390
206	1.250	1250.000	216	1.563	1562.500
207	1.276	1275.510	217	1.603	1602.564
208	1.302	1302.083	218	1.645	1644.737
209	1.330	1329.787	219	1.689	1689.189
210	1.359	1358.696	220	1.736	1736.111
221	1.786	1785.714	231	2.500	2500.000
222	1.838	1838.235	232	2.604	2604.167
223	1.894	1893.939	233	2.717	2717.391
224	1.953	1953.125	234	2.841	2840.909
225	2.016	2016.129	235	2.976	2976.190
226	2.083	2083.333	236	3.125	3125.000
227	2.155	2155.172	237	3.289	3289.474
228	2.232	2232.143	238	3.472	3472.222
229	2.315	2314.815	239	3.676	3676.471
230	2.404	2403.846	240	3.906	3906.250
241	4.167	4166.667	251	12.500	12500.000
242	4.464	4464.286	252	15.625	15625.000
243	4.808	4807.692	253	20.833	20833.333
244	5.208	5208.333	254	31.250	31250.000
245	5.682	5681.818	255	62.500	62500.000
246	6.250	6250.000			
247	6.944	6944.444			
248	7.813	7812.500			
249	8.929	8928.571			
250	10.417	10416.667			

第五章. 串列輸入/ 輸出發射接收器 (SIO)

SN8P04XX 系列提供一組具有時脈速率選擇的 8-bit SIO 介面電路， SIOM 暫存器可控制 SIO 的操作功能，例如：傳送/接收(TX/RX)，時脈速率(clock rate)，傳輸觸發邊界(transfer edge)及啓動電路(strating circuit)等等。SIO 電路藉由 SIOM 暫存器的 SENB 及 START 位元，控制自動發射(TX)或接收(RX) 8-bit 的資料，SIOB 是一個 8-bit 緩衝器，爲儲存傳輸資料而設計。SIOC 及 SIOR 是設計爲產生具有自動重新載入功能的 SIO 時脈源，3-bit I/O 計數器可監視 SIO 的操作，在發射/接收 8-bit 資料後，通告一個中斷需求。在傳輸 8-bit 資料後，電路將自動禁能，並藉由編制 SIOM 暫存器重新傳輸資料。



1. SIOM 模式暫存器 :

SIOM 初值 = 0XXX -XXX

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
SIOM	SENB	START	SRATE1	SRATE0	-	SCKMD	SEDGE	TXRX

SENB：SIO 功能控制位元。 0 = 禁能 (P5.0~P5.2 是一般用途的輸出/入埠)，
1 = 致能 (P5.0~P5.2 爲 SIO 接腳)。
START：SIO 進行資料傳輸控制位元。 0 = 傳輸結束，1 = 進行資料傳輸。
SRATE：SIO 時脈選擇位元。00 = Fcpu，01 = Fcpu/32，10 = Fcpu/16，11 = Fcpu/8。
SCKMD：SIO 的時脈模式選擇位元。0 = 內部模式，1 =外部模式。
SEDGE：SIO 轉換時脈邊界選位元。0 = 下緣觸發(falling edge)，
1 = 上緣觸發(rising edge)。
TXRX：SIO 資料傳輸方向選擇位元。0 =單向接收(RX)，1 = 雙向發射/接收(TX/RX)。

2. SIOB/SIOR 資料緩衝器 :

SIOB 初值 = XXXX XXXX

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
SIOB/SIOR	X	X	X	X	X	X	X	X

SIOB 是 8-bit 的資料緩衝器，爲資料發射接收暫存區。

SIOR 初值 = XXXX XXXX

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
SIOB/SIOR	X	X	X	X	X	X	X	X

SIOR 是 8-bit 的資料緩衝器， SIOR 的設計爲 SIOC 在計數結束後，重新載入計數的數值。
注意：設置 SIOR 的數值取得所想要的傳輸時脈公式如下：

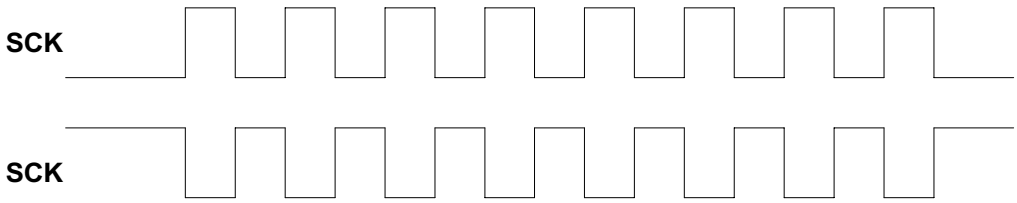
SIOR 數值= 256 - (想要的 SIO 時脈 * 選擇輸入時脈)

3. SIO 操作範例 :

SN8P04XX 系列所提供 SIO 與一般的 SPI 完全相容，主動式及被動式應用如下：

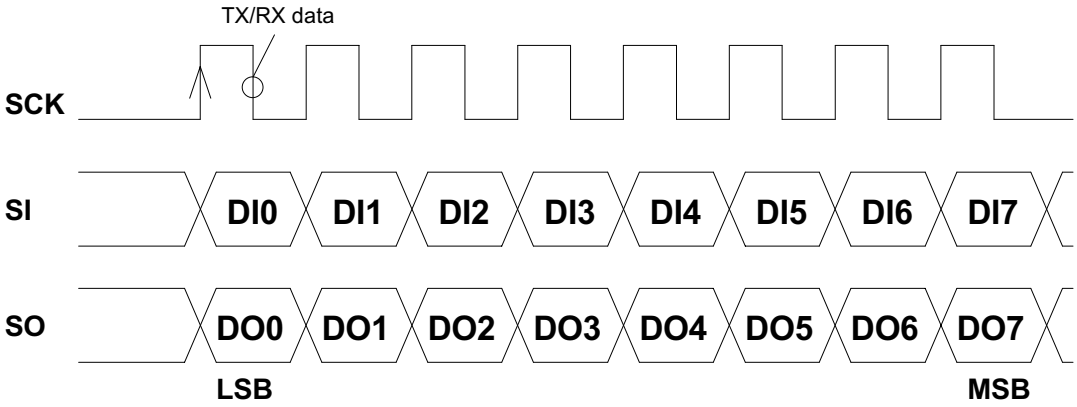
3.1. 主動式 SIO :

SN8P04XX 系列在主動式發射狀態之下，所輸出的 SCK 有以下 2 種相位：



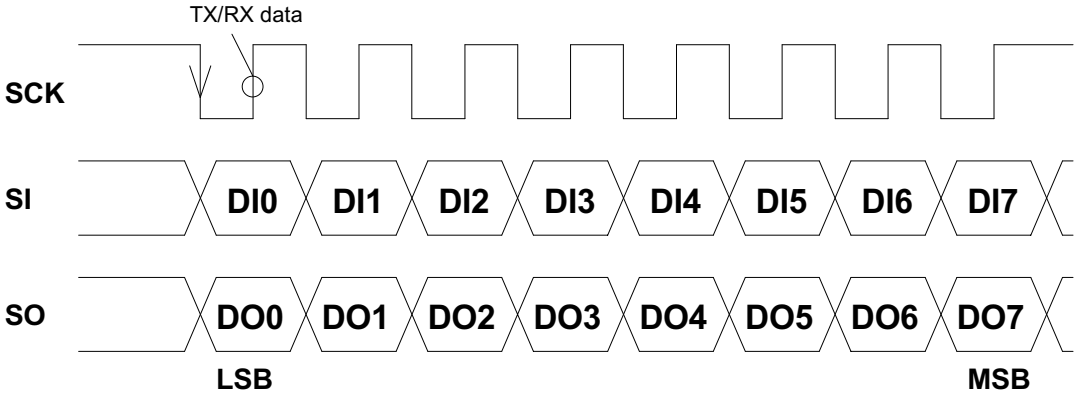
3.1.1. 主動式發射/接收上緣觸發 :

```
; Master TX/RX rising edge
                MOV     A,TXDATA        ; 載入發射資料至 SIOB 暫存器
                B0MOV   SIOB,A
                MOV     A,#0FFH         ; 設定 SIO 時脈具自動重新載入功能
                B0MOV   SIOR,A
                MOV     A,#11000011B    ; 致能 SIO 並且啓動 SIO，上緣觸發
                B0MOV   SIOM,A
WAIT_TXRXSIO :
                B0BTS0  FSTART          ; 等待 SIO 動作完成
                JMP     WAIT_TXRXSIO
                B0MOV   A,SIOB          ; 儲存 SIOB 資料至 RXDATA 位址
                MOV     RXDATA,A
```



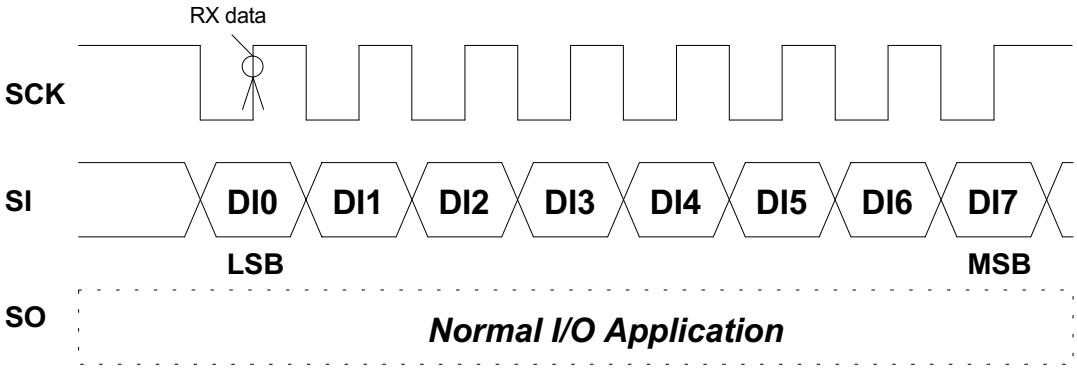
3.1.2. 主動式發射/接收下緣觸發：

```
；Master TX/RX falling edge
                MOV     A,TXDATA        ；載入發射資料至 SIOB 暫存器
                B0MOV   SIOB,A
                MOV     A,#0FFH         ；設定 SIO 時脈具自動重新載入功能
                B0MOV   SIOR,A
                MOV     A,#11000001B    ；致能 SIO 並且啓動 SIO，下緣觸發
                B0MOV   SIOM,A
WAIT_TXRXSIO：
                B0BTS0  FSTART          ；等待 SIO 動作完成
                JMP     WAIT_TXRXSIO
                B0MOV   A,SIOB          ；儲存 SIOB 資料至 RXDATA 位址
                MOV     RXDATA,A
```



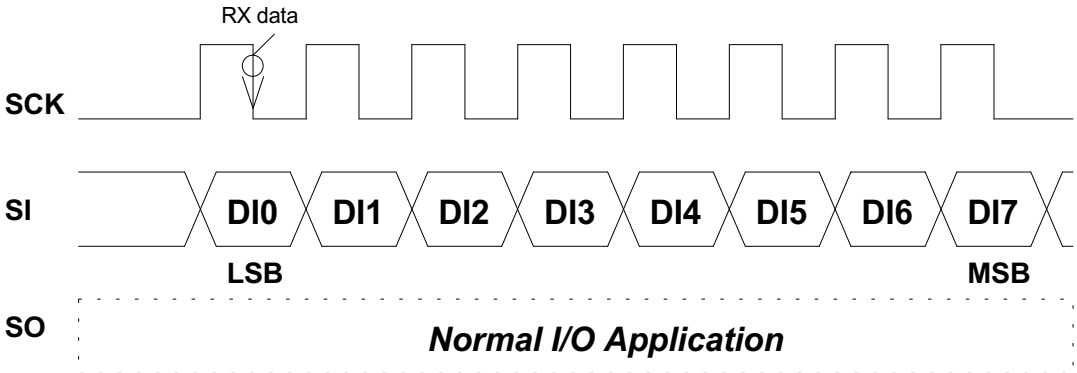
3.1.3. 主動式接收上緣觸發：

```
；Master RX rising edge
                MOV     A,#0FFH         ；設定 SIO 時脈具自動重新載入功能
                B0MOV   SIOR,A
                MOV     A,#11000010B    ；致能 SIO 並且啓動 SIO，上緣觸發
                B0MOV   SIOM,A
WAIT_RXSIO：
                B0BTS0  FSTART          ；等待 SIO 動作完成
                JMP     WAIT_RXSIO
                B0MOV   A,SIOB          ；儲存 SIOB 資料至 RXDATA 位址
                MOV     RXDATA,A
```



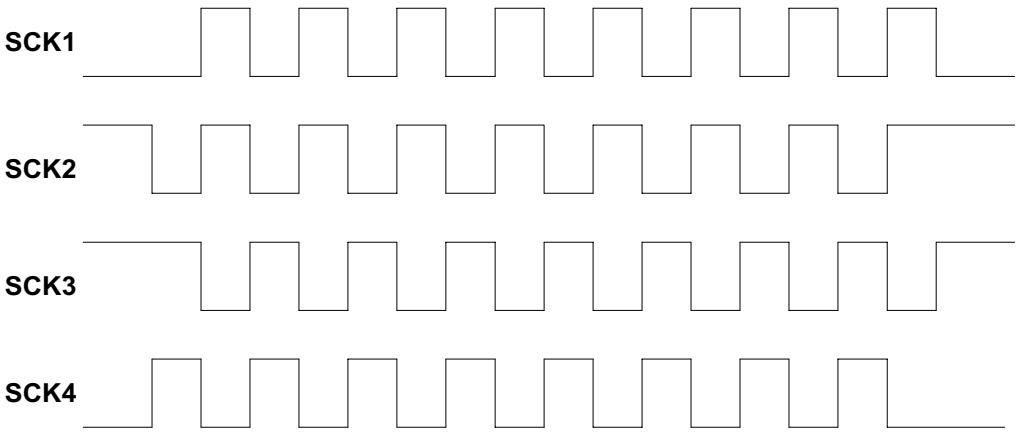
3.1.4. 主動式接收下緣觸發：

```
Master RX falling edge
                MOV     A,#0FFH         ；設定 SIO 時脈具自動重新載入功能
                B0MOV   SIOR,A
                MOV     A,#11000000B    ；致能 SIO 並且啓動 SIO，下緣觸發
                B0MOV   SIOM,A
WAIT_RXSIO：
                B0BTS0  FSTART          ；等待 SIO 動作完成
                JMP     WAIT_RXSIO
                B0MOV   A,SIOB          ；儲存 SIOB 資料至 RXDATA 位址
                MOV     RXDATA,A
```



3.2. 被動式 SIO :

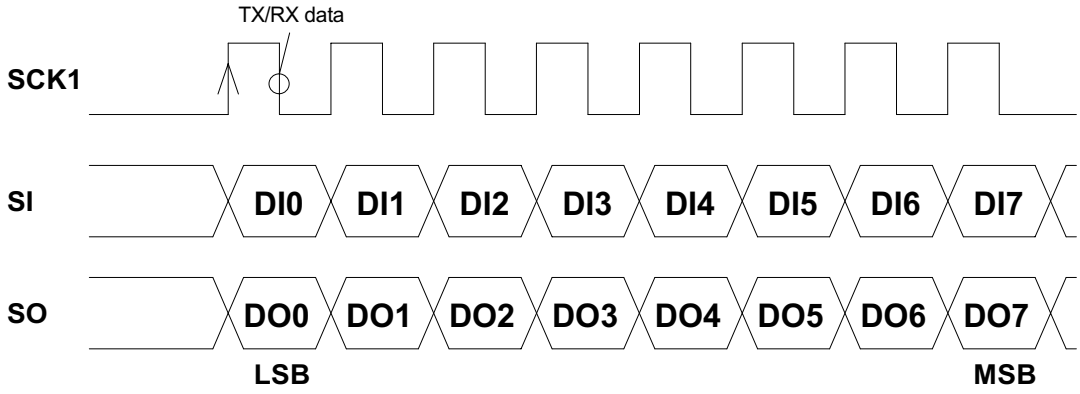
SN8P04XX 系列在被動式接收狀態之下，能接收的 SCK 有以下 4 種相位，這 4 種相位完全與 SPI 相容：

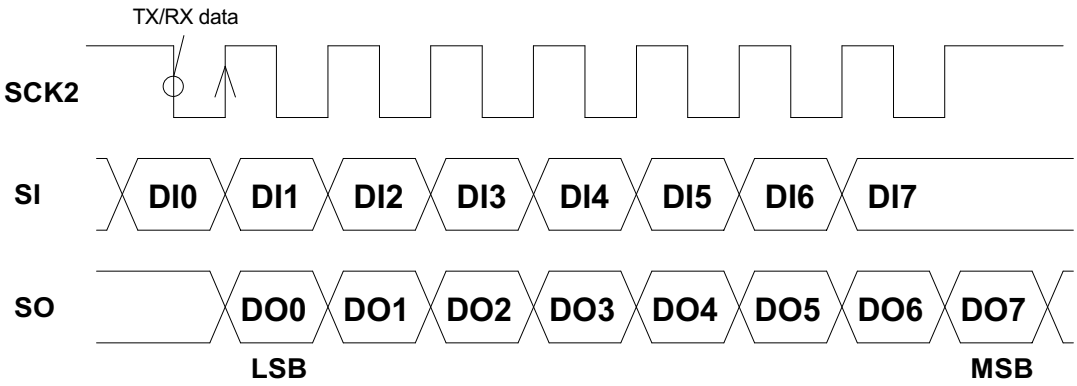


3.2.1. 被動式發射/接收上緣觸發：

Slave TX/RX rising edge

```
                MOV     A,TXDATA        ；載入發射資料至 SIOB 暫存器
                B0MOV   SIOB,A

                MOV     A,#11000111B    ；致能 SIO 並且啓動 SIO，上緣觸發
                B0MOV   SIOM,A
WAIT_TXRXSIO：
                B0BTS0  FSTART          ；等待 SIO 動作完成
                JMP     WAIT_TXRXSIO
                B0MOV   A,SIOB          ；儲存 SIOB 資料至 RXDATA 位址
                MOV     RXDATA,A
```

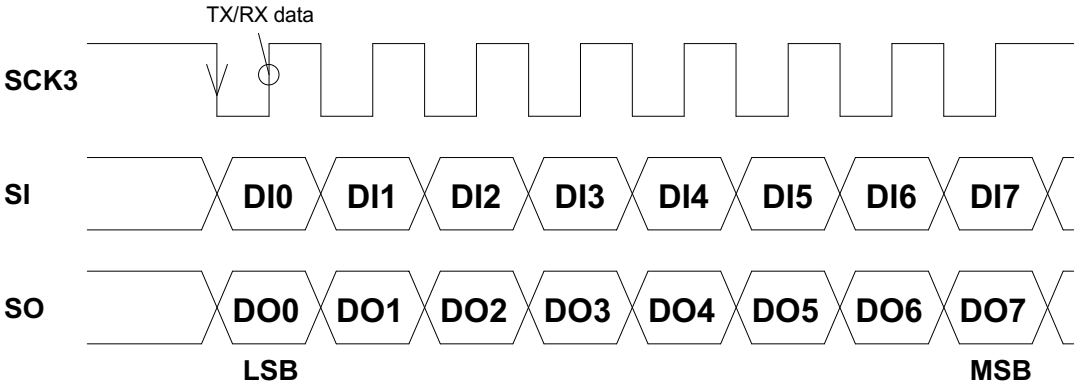


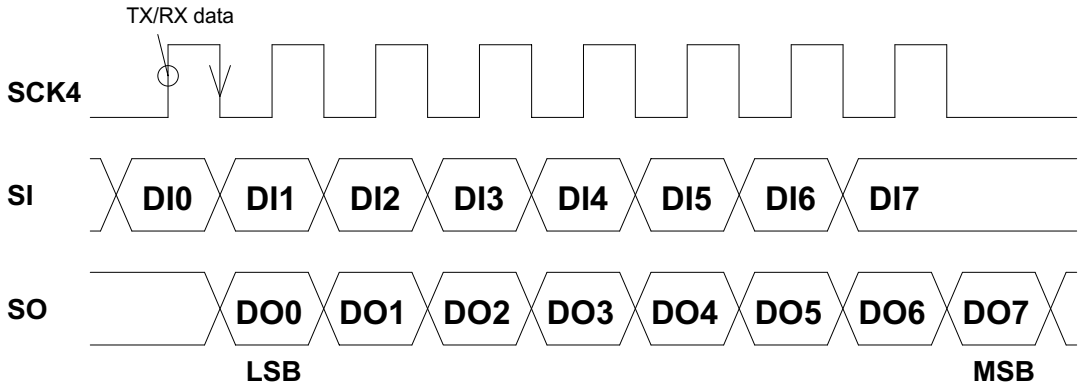


3.2.2. 被動式發射/接收下緣觸發：

Slave TX/RX falling edge

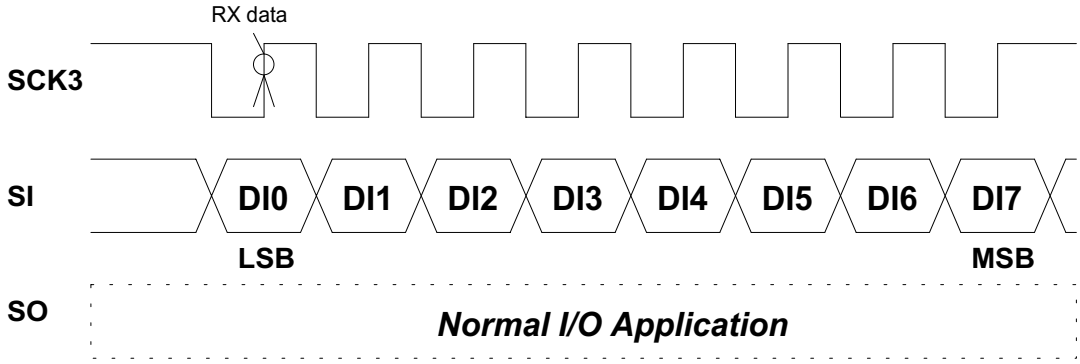
```
                MOV     A,TXDATA        ；載入發射資料至 SIOB 暫存器
                B0MOV   SIOB,A

                MOV     A,#11000101B    ；致能 SIO 並且啓動 SIO，下緣觸發
                B0MOV   SIOM,A
WAIT_TXRXSIO:
                B0BTS0  FSTART          ；等待 SIO 動作完成
                JMP     WAIT_TXRXSIO
                B0MOV   A,SIOB          ；儲存 SIOB 資料至 RXDATA 位址
                MOV     RXDATA,A
```

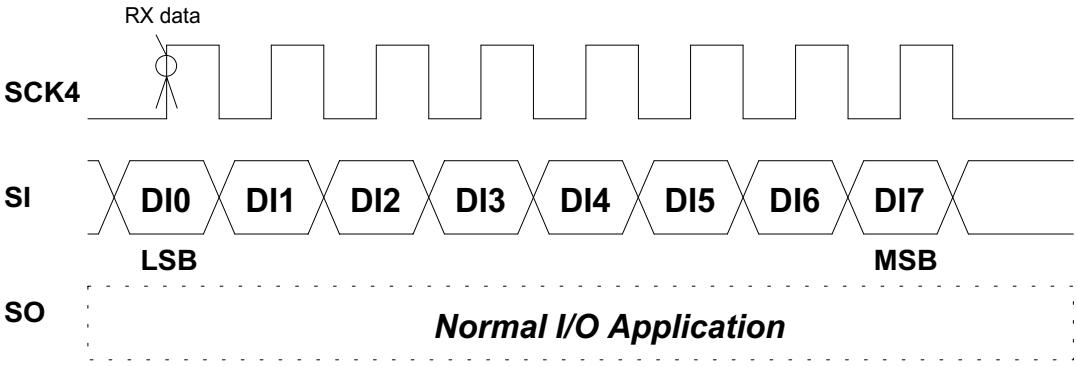




3.2.3. 被動式接收上緣觸發：
Slave RX raising edge

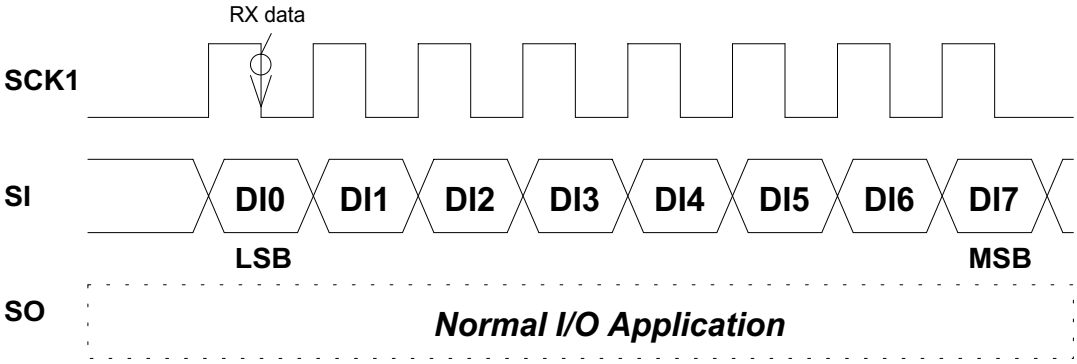
```
                MOV     A,#11000110B    ；致能 SIO 並且啓動 SIO，上緣觸發
                B0MOV   SIOM,A
WAIT_RXSIO:
                B0BTS0  FSTART          ；等待 SIO 動作完成
                JMP     WAIT_RXSIO
                B0MOV   A,SIOB          ；儲存 SIOB 資料至 RXDATA 位址
                MOV     RXDATA,A
```

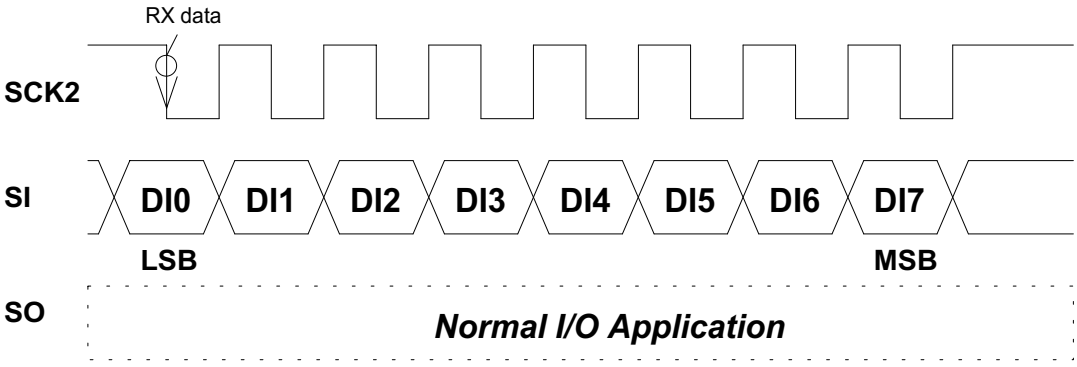




3.2.4. 被動式接收下緣觸發：
Slave RX falling edge

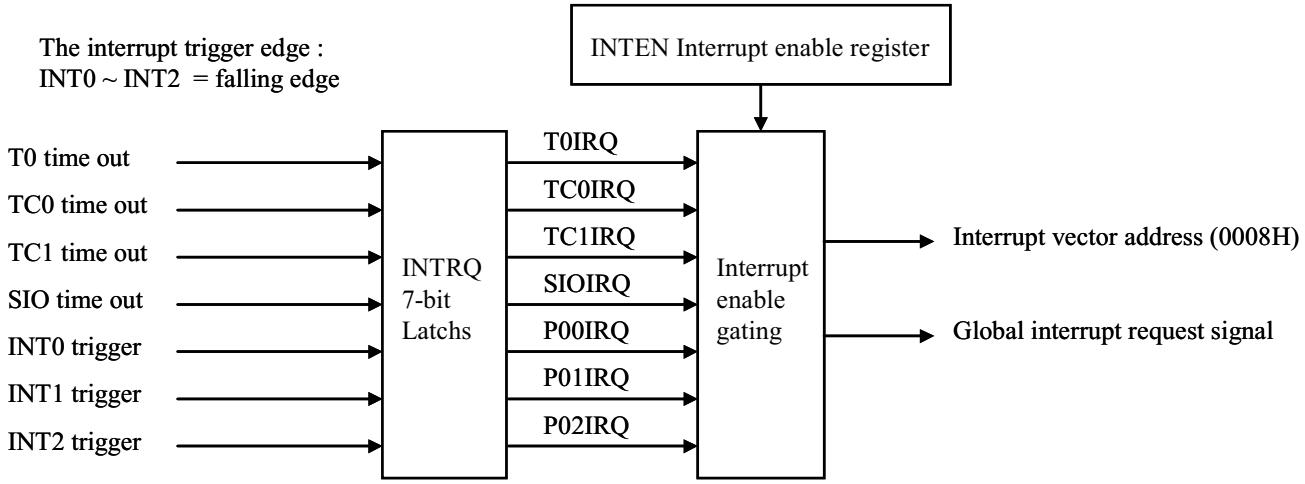
```
                MOV     A,#11000100B    ；致能 SIO 並且啓動 SIO，下緣觸發
                B0MOV   SIOM,A
WAIT_RXSIO：
                B0BTS0  FSTART          ；等待 SIO 動作完成
                JMP     WAIT_RXSIO
                B0MOV   A,SIOB          ；儲存 SIOB 資料至 RXDATA 位址
                MOV     RXDATA,A
```





第六章. 中斷

SN8P04XX 系列提供 7 個中斷源，包含 4 個內部中斷(T0、TC0、TC1、SIO) 及 3 個外部中斷(INT0、INT1、INT2)，當晶片在睡眠模式中，這些中斷具有喚醒系統進入高速模式的功能；TC0/TC1 外部時脈輸入埠與 P0.0、P0.1 共用，除此之外，所有 P0 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 = X000 0000

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
INTEN	-	TC1IEN	TC0IEN	T0IEN	SIOIEN	P02IEN	P01IEN	P00IEN

P00IEN：外部的 P0.0 中斷要求位元。0 = 禁能，1 = 致能。
P01IEN：外部的 P0.1 中斷要求位元。0 = 禁能，1 = 致能。
P02IEN：外部的 P0.2 中斷要求位元。0 = 禁能，1 = 致能。
SIOIEN：SIO 中斷要求位元。 0 = 禁能，1 = 致能。
T0IEN：T0 計時器中斷要求位元。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。
TC1IEN：計時器/事件計數器 1 中斷控制位元。 0 = 禁能，1 = 致能。

2. INTRQ 中斷需求暫存器：

INTRQ 初值 = X000 0000

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
INTRQ	-	TC1IRQ	TC0IRQ	T0IRQ	SIOIRQ	P02IRQ	P01IRQ	P00IRQ

P00IRQ：外部的 P0.0 中斷需求位元。 0 = 沒有需求，1 =需求。
P01IRQ：外部的 P0.1 中斷需求位元。 0 =沒有需求，1 =需求。

P02IRQ：外部的 P0.2 中斷需求位元。 0 =沒有需求，1 =需求。
SIOIRQ：SIO 中斷需求位元。 0 =沒有需求，1 =需求。
T0IRQ：T0 計時器中斷需求位元。 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。
TC1IRQ：TC1 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
                PUSH                ; PUSH 指令
                B0MOV   ACCBUF,A    ; 儲存 ACC 資料

                B0BTS0  FP00IRQ     ; 檢查是否有 P00 中斷需求
                JMP     P00INTR     ; 跳至 P00 中斷服務程序

                B0BTS0  FT0IRQ      ; 檢查是否有 T0 中斷需求
                JMP     T0INTR      ; 跳至 T0 中斷服務程序
                .       .
                .       .
QINT_RS:
                B0MOV   A,ACCBUF    ; 載入 ACC 資料
                POP                 ; POP 指令
                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

T0INTR:                             ; T0 中斷服務程序
                B0BCLR  FT0IRQ      ; 清除 FT0IRQ 旗標
                .       .       .
                .       .       .
                JMP     QINT_RS     ; 跳出中斷服務程序
```

中斷需求優先權排列

在判斷中斷旗標的程序中，使用者可排列各中斷需求執行的優先權，要注意實際中斷發生時機，例如，若外部中斷與內部中斷同時存在，需注意是否外部中斷發生過於頻繁，而導致內部中斷無法發生，當然可以執行一組中斷需求程序後，判斷下一組中斷，但是此方法會佔據太多中斷資源，不建議使用。

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 T0 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，T0 clock = Fcpu/64，time base = 5ms

```
T0INTR：                                        ；T0 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      QT0INTR                 ；爲 0，無中斷發生

                B0BTS0  TMSTAT.BIT10M           ；檢查 10ms 中斷旗標
                JMP     T0INTR10M               ；跳至 10ms 中斷服務程序

                B0BTS0  TMSTAT.BIT100M          ；檢查 100ms 中斷旗標
                JMP     T0INTR100M              ；跳至 100ms 中斷服務程序

                B0BTS0  TMSTAT.BIT1000M         ；檢查 1000ms 中斷旗標
                JMP     T0INTR1000M             ；跳至 1000ms 中斷服務程序

                .       .
                .       .

QT0INTR：                                       .
                B0BCLR  FT0IRQ                  ；清除 FT0IRQ 旗標
                MOV     A,#0BAH
                B0MOV   T0C,A                   ；重置 T0C
                JMP     QINT_RS                 ；跳出中斷服務程序


T0INTR10M：
                B0BSET  INTGND.BIT10M           ；設定中斷介面旗標
                B0BCLR  TMSTAT10M               ；清除 TMSTAT10M
                JMP     QT0INTR                 ；跳出中斷服務程序

T0INTR100M：
                B0BSET  INTGND.BIT100M          ；設定中斷介面旗標
                B0BCLR  TMSTAT100M              ；清除 TMSTAT100M
                JMP     QT0INTR                 ；跳出中斷服務程序

T0INTR1000M：
                B0BSET  INTGND.BIT1000M         ；設定中斷介面旗標
                B0BCLR  TMSTAT1000M             ；清除 TMSTAT1000M
                JMP     QT0INTR                 ；跳出中斷服務程序
```

計時器中斷旗標判斷

計時器中斷介面旗標設定

```
INCTM：
        B0BSET    INTGND.BIT5M        ；設 5ms 旗標

        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
                                                        各時間應
MNINTGND100M：                                          用程序執
                B0BCLR    INTGND.BIT100M    ；清除 100ms 旗標      行區
                .         .                 ；100ms 應用程序
                .         .
                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 T0INTR 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，T0INTR 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

```
T0INTR：                                    ；T0 中斷服務程序
        INCMS   TMBUF
        JMP     T0INTR10                    ；TMBUF 不爲 ‘0’
                                            ；TMBUF 爲 ‘0’ ，代表由
                                            ；FFH 加 1 爲 ‘0’ ，發生
                                            ；溢位
        MOV     A,#00000001B                ；修正 TMBUF 數值爲
        B0MOV   TMBUF,A                     ；00000001B

T0INTR10：
        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
        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
```

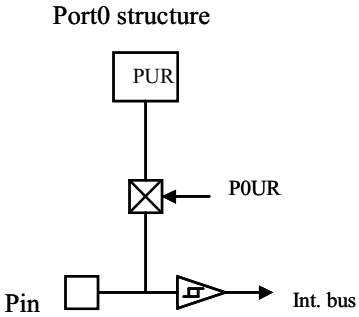
TMBUF 加 1 程序

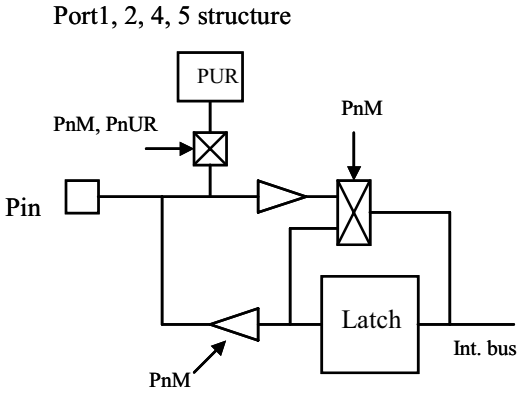
中斷服務程序執行判斷

```
T0INTR90 :
            B0BCLR      FT0IRQ          ; 清除 FT0IRQ 旗標
            MOV         A,#0BAH
            B0MOV       T0C,A           ; 重置 T0C
            JMP         QINT_RS         ; 跳出中斷服務程序

INTR5ms :                               ; 5ms 中斷服務程序
            .           .
            .           .
            JMP         T0INTR90

INTR10ms :                              ; 10ms 中斷服務程序
            .           .
            .           .
            JMP         T0INTR90

INTR20ms :                              ; 20ms 中斷服務程序
            .           .
            .           .
            JMP         T0INTR90

INTR40ms :                              ; 40ms 中斷服務程序
            .           .
            .           .
            JMP         T0INTR90

INTR80ms :                              ; 80ms 中斷服務程序
            .           .
            .           .
            JMP         T0INTR90

INTR160ms :                             ; 160ms 中斷服務程序
            .           .
            .           .
            JMP         T0INTR90

INTR320ms :                             ; 320ms 中斷服務程序
            .           .
            .           .
            JMP         T0INTR90

INTR640ms :                             ; 640ms 中斷服務程序
            .           .
            .           .
            JMP         T0INTR90
```

第七章. 輸入/輸出埠

SN8P04XX 系列提供 5 組埠給使用者應用，有 1 個輸入埠(P0)及 4 個輸入/輸出埠(P1、P2、P4、P5)所構成，每一個輸入埠腳位的昇壓電阻器可藉由 PnUR 暫存器設定選擇，I/O 埠方向是由 PnM 暫存器所選擇的，當系統重置後，I/O 埠工作狀態是無昇壓電阻器的輸入埠。





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，進入省電或睡眠模式之前，有參予喚醒功能的位元需設定其數值爲 ‘1’ 。

P1W 初值 = 0000 0000

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
P1W	P17W	P16W	P15W	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. 昇壓電阻器 (PnUR)暫存器 :

PnUR 初值= 0000 0000

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
PnUR	Pn7R	Pn6R	Pn5R	Pn4R	Pn3R	Pn2R	Pn1R	Pn0R

Pn：n 表示 0、1、2、4、5。

Pn7R ~ Pn0R：PORT n.7 ~PORT n.0 爲昇壓電阻器控制位元。0 = 沒有昇壓電阻器，1 = 昇壓電阻器。

設定輸入埠有昇壓電阻：

```
MOV     A,#00001111B
B0MOV   PnUR,A          ；n=0、1、2、4、5，設定 Pn 的 0~3 位元爲昇壓電阻型式，其餘位元爲無昇壓電阻。
```

建議當輸入信號爲浮動式信號時，必須設定昇壓電阻，以避免訊號浮動，而使耗電流增加；當訊號爲浮動或 ‘1’ 時，所讀取到的值爲 ‘1’ ，當訊號爲 ‘0’ 時，所讀取到的數值爲 ‘0’ 。

3. 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、4、5。
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、4、5，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。

4. 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、4、5。
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、4、5，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、4、5，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、4、5，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
```

5. 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，無昇壓電阻，以 Pn(n=1,2,4,5) 的 8 個位元均無昇壓電阻說明之：

```
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，有昇壓電阻，以 Pn(n=1,2,4,5) 的 8 個位元均有昇壓電阻說明之：

```
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’ ，因有昇壓電阻，切換至輸入模式時，埠內部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=1,2,4,5) 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。

第八章. 8 頻道類比數位轉換器

SN8P04XX 系列提供 8 個腳位作爲類比轉換器輸入源(AIN0~AIN7 與 PORT4 共用)，以 256 階的解析度，轉換類比訊號爲 8-bit 數位訊號。ADC 操作順序，首先選擇輸入源(AIN0~AIN7)，設定 GCHS 及 ADS 位元爲 ‘1’ 開始轉換類比訊號，經過 8 次比較步驟後，ADC 電路將 EOC 位元設置爲 ‘1’ ，並且將輸出最後結果儲存在 ADB 暫存器。ADC 電路透過 ADR 暫存器中的 ADLEN 位元，選擇在 8-bit 解析度下，做類比數位轉換工作。

1. ADM 暫存器 :

ADM 初值 = 0XX0 XXXX

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
ADM	ADENB	ADS	EOC	GCHS	-	CHS2	CHS1	CHS0

CHS2、1、0：8 通道選擇位元。
000 = AIN0，001 = AIN1，010 = AIN2，011 = AIN3， ... ，111 = AIN7。
GCHS：全通道選擇位元。0 = 禁能 AIN 通道，1 = 致能 AIN 通道。
EOC：ADC 狀態位元。 0 = 處理中，1 = 結束轉換工作及重置 ADENB 位元。
ADS：ADC 啓動位元。 0 = 結束轉換， 1 = 開始轉換。
ADENB：ADC 控制位元。 0 =禁能， 1 =致能 。

2. ADB&ADR 暫存器 :

ADB 初值= XXXX XXXX

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
ADB	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4

ADR 初值 = XXX0 XXXX

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
ADR	-	ADCKS	ADLEN	0	ADB3	ADB2	ADB1	ADB0

ADBn：ADC 資料緩衝區。ADB11~ADB4 位元儲存 8-bit ADC 轉換結果。
ADLEN：ADC 解析度選擇位元。 0 = 8-bit，需設定爲 0。
ADCKS：ADC 時脈源選擇位元。 0 = fosc/2，1 = fosc。

3. ADC 範例說明：

設置 AIN0 ~AIN1 爲 ADC 輸入，執行 8-bit ADC 轉換工作。

```
ADC0：      B0BCLR   FTADC          ；致能 ADC 電路和禁能測驗模式。
            B0BCLR   FADLEN         ；設置 8-bit 的 ADC 操作。
            MOV      A,#90H
            B0MOV    ADM,A          ；致能 ADC 和設置 AIN0 輸入。
            B0BSET   FADS           ；開始轉換。

WADC0：     B0BTS1   FEOC           ；如果轉換結果 =1，則跳過。
            JMP      WADC0          ；如果轉換結果不等於 1，則跳至 WADC 0。
            B0MOV    A,ADB          ；取得 AIN0 輸入資料。

ADC1：      MOV      A,#91H
            B0MOV    ADM,A          ；致能 ADC 和設置的 AIN1 輸入。
            B0BSET   FADS           ；開始轉換。
            .        .
QEXADC：    B0BCLR   FGCHS          ；關閉 AINn 輸出通道。
```

4. AIN 輸入電壓及 ADB 輸出資料對照表：

AINn	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4
1/256* AVREF	0	0	0	0	0	0	0	0
2/256* AVREF	0	0	0	0	0	0	0	0
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
254/256* AVREF	1	1	1	1	1	1	1	1
255/256* AVREF	1	1	1	1	1	1	1	1

第九章. 7-Bit 數位類比暫存器

SN8P04XX 系列提供一組數位類比轉換器，使用 7-bit 的架構，合成 128 階類比信號，此信號爲電流源輸出。在 DAENB 位元設爲 ‘1’ 後，DAC 電路轉爲致能，DAM 暫存器從位元 0 到位元 6 輸出數位信號到階梯電阻器，在 DAO 腳位產生類比信號。

1. DAM 暫存器 :

DAM 初值 = 0XXX XXXX

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DAM	DAENB	DAB6	DAB5	DAB4	DAB3	DAB2	DAB1	DAB0

DABn：數位輸入資料暫存區，n = 0~6。
DAENB：數類轉換器控制位元，0 = 禁能， 1 = 致能。

2. DAB 資料及 DAO 輸出電壓對照表 :

DAB6	DAB5	DAB4	DAB3	DAB2	DAB1	DAB0	MVO
0	0	0	0	0	0	0	VSS
0	0	0	0	0	0	1	IDAC
0	0	0	0	0	1	0	2 * IDAC
0	0	0	0	0	1	1	3 * IDAC
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
1	1	1	1	1	1	0	126 * IDAC
1	1	1	1	1	1	1	127 * IDAC

3. 範例說明 :

3.1. 設定及輸出類比信號：

```
MOV     A,#01111111B        ;設定 DABn 爲 1111111B
B0BSET  FDAENB              ;設定 DAENB = 1，則 DAO 輸出信號爲
                            127*IDAC
.       .
.       .
.       .
B0BCLR  FDAENB              ;設定 DAENB = 0，關閉 DAO 輸出
```

3.2. 輸出 sin 弦波：

DAO 可以模擬輸出 SIN 弦波，再經過簡單的 RC 電路處理，成爲完整的交流 SIN 弦波，此應用可用於 TONE 輸出及一些需要交流信號的情況下。

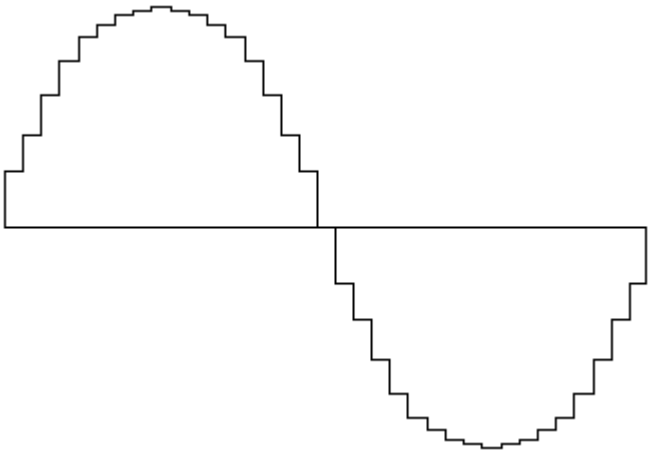
產生一個 32 階的模擬弦波，DAO 輸出以 DABn 數值(0~127)表示，計算如下：

DABn 數值 = sinωt ，(ω = 2πf，f = 1，t = x/32)

X	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DABn	76	88	99	108	116	122	126	127	126	122	116	108	99	88	76	63

X	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
DABn	51	39	28	19	11	5	1	0	1	5	11	19	28	39	51	63

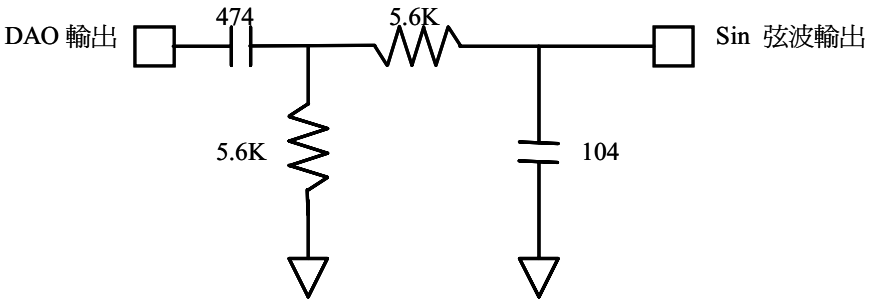
由於 DAO 是電流輸出，所以 DAO 腳位需並聯 1K ohm 電阻落地，將電流輸出轉換爲電壓輸出，得到的正弦波形如下所示：



DAO 的輸出僅決定弦波的電位，弦波的週期須以中斷計時處理，以頻率 60Hz 的弦波信號爲例：

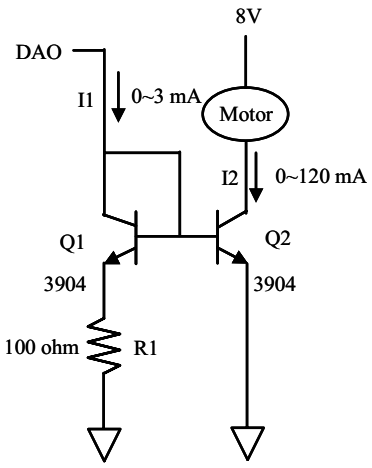
頻率 60Hz = 週期 16.67ms

因爲一個完整弦波被切割爲 32 階，所以每一階時間間距爲 16.67ms ÷32 = 0.52ms，所以設定一組 0.52ms 的中斷程序，當中斷需求發生，改變 1 階弦波電位，便可產生一個完整弦波，以下提供簡單的 RC 電路，將類比輸出轉換爲完整交流弦波：



3.3. 直流馬達控制線路：

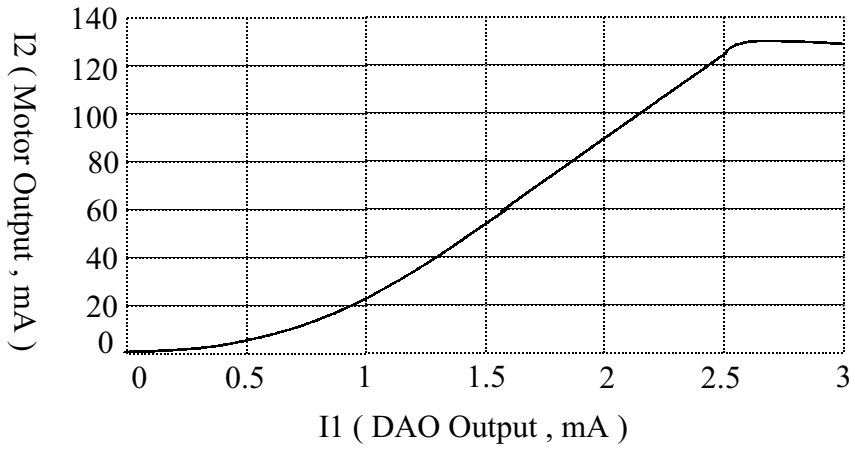
DAO 可作爲 PWM 信號驅動直流馬達，利用 DAO 電流輸出特性，外接 2 個 NPN 電晶體構成一組差動放大電路，如下所示：



假設馬達的直流負載有效阻抗爲 60ohm，利用此線路可得一簡單的電流放大：

$$I2 \approx \alpha \times I1$$

其中放大倍率 α 和 R1 以及 BJT 的特性有關，其電流放大倍率可由調整 R1 改變之。假設直流馬達阻抗爲 60ohm，DAO 輸出電流與流過直流馬達電流關係如下所示：



第十章. 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) .....-20°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 除非另備註)

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	2.8	5.0	5.5	V
		燒錄模式，VPP = 12.5V	4.5	5.0	5.5	
操作電流	IddH	Vdd = 5.0V，I/O 腳位無負載，一般模式	-	0.6	2.0	mA
	Istby	Vdd = 5.0V，I/O 腳位無負載，睡眠模式	-	-	2	uA
重置腳位輸入電壓	ViH		0.7Vdd	-	-	V
	ViL		-	-	0.3Vdd	
重置腳位漏電流	ILekg	Vin = Vdd	-	-	2	uA
I/O 埠輸入電壓	ViH		0.8Vdd	-	-	V
	ViL		-	-	0.2Vdd	
I/O 埠昇壓電阻	Rup	Vin = Vss	-	180	-	KΩ
I/O 埠輸入漏電流	ILekg	禁能昇壓電阻器， Vin = Vdd	-	-	2	uA
PORT1 輸出電流源	IoH	Vop = Vdd - 0.5V	2	4	-	mA
SINK 電流	IoL	Vop = Vss + 0.5V	4	8	-	
PORT2 輸出電流源	IoH	Vop = Vdd - 0.5V	2	4	-	mA
SINK 電流	IoL	Vop = Vss + 0.5V	4	8	-	
PORT4 輸出電流源	IoH	Vop = Vdd - 0.5V	2	4	-	mA
SINK 電流	IoL	Vop = Vss + 0.5V	4	8	-	
PORT5 輸出電流源	IoH	Vop = Vdd - 0.5V	2	4	-	mA
SINK 電流	IoL	Vop = Vss + 0.5V	4	8	-	
INTn 觸發脈衝寬度	Tint0	INT0 ~ INT2 中斷請求脈衝寬度	2/Fcpu	-	-	cycle
AVREF 輸入電壓	Varef	Vdd = 5.0V	2	-	Vdd	V
AIN0~AIN7 輸入電壓	Vani		Vss+0.2	-	Avref	V
振盪器頻率	fhxosc	晶體振盪器或陶瓷振盪器	-	3.58	10	MHz
		RC 振盪	-	-	4	

第十一章. 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[X,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
	MUL	A,M	R，A←A*M，運算結果的低位元組資料儲存在 ACC，高位元組資料儲存在 R 暫存器，零旗標隨 ACC 變動	-	-	√	2
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

B R A N C H	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
M I S C	RET		PC ←堆疊	-	-	-	2
	RETI		PC ←堆疊，致能主中斷	-	-	-	2
	PUSH		推入(PUSH)工作暫存器(080H~087H) 至緩衝區	-	-	-	1
	POP		拉出(POP)工作暫存器(080H~087H) 從緩衝區	√	√	√	1
	NOP		空白指令，無動作	-	-	-	1

註：A). 工作暫存器 = H, L, R, X, Y, Z, PFLAG 及 RBANK。
B). 記憶體存取於 RAM[H,L]位置，若 M = @HL (位於 RAM BANK 0 的 E6H 位址)。
C). 記憶體存取於 RAM[Y,Z]位置，若 M = @YZ (位於 RAM BANK 0 的 E7H 位址)。
D). 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。