第八章

計時器

8.1 計時器（TIMER）

計時器（TIMER）一般與計數器（counter）在微處理器中採同一電路，只是前者計數源是一週期性時脈，後者是用來計算非週期性事件次數。在一般微處理器大都會提供計時計數器，計時器經常被用來量測兩事件的間隔時間（interval），或者以上數的方式被用來當成碼錶（stopwatch）使用，有些應用下數模式當成計時器用，不同應用場合或許有不同的名詞，不過都是利用計時器所提供的功能。當今微處理器提供的計時器有些加上匹配暫存器（match register）之後用產生 PWM 輸出波形，第六章中提及 PWM 輸出產生的方式有一種就是利用數位產生的方式，而計時器就是最好的工具之一。

8.2 WT59F064 TIMER

8.2.1 WT59F064 TIMER 特性與使用說明

WT59F064 共提供 6 個計時器，每個計時器內含一個 16 位元計數器同時有一個 16 位元除頻計數器，計時的來源可以是 PCLK 或 EXTCLK，EXTCLK 的選用設定同 RTC 一致，靠設定系統暫存器（0x200020）LSEON 位元選用外部 32.768KHz 的時脈。TIMER 特性概述如下：

- 每個計時器內含一個 16 位元計數器同時有一個 16 位元除頻計數器
- 可設為計時或計數模式
- 每個計時器含有兩個 16 位元的捕捉通道（capture channel）
- 當下列兩種情況產生時可以產生中斷或要求進行 DMA 傳輸
  - 輸入捕捉出現
  - 輸出匹配出現
- 四個 16 位元匹配暫存器
  - 設為當匹配出現時計時器繼續計時
  - 設為當匹配出現時停止計時
  - 設為當匹配出現時計時器重置

- 當匹配發生時產生的兩個輸出信號可以下列方式呈現輸出
  - 當匹配出現時將輸出設為低電位
  - 當匹配出現時將輸出設為高電位
  - 當匹配出現時切換輸出電位
- 結合匹配暫存器可以產生 PWM 輸出波形

每個計時器模組只提供一個 16 位元計數器，可以計時亦可當成計數器使用，純粹視計數來源是否是週期性的計數源，當成計數器使用模式時，計數源可以是 mati0、mati1、capi0 或 capi1 其中一項，每個計時器中有兩個 16 位元的擷取通道，有兩個匹配暫存器，可以用來產生 PWM 輸出

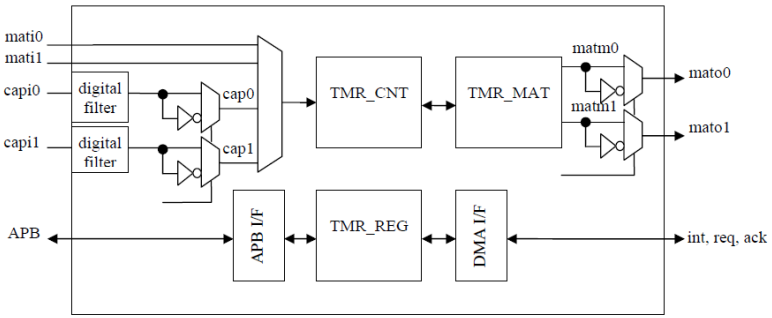

圖 8.2.1-1 WT59F064 TIMER 方塊圖

（擷取自 WT59F064 data sheet v091.pdf）

表 8.2.1-1 列出 WT59F064 中六個計時器的位址空間，表 8.2.1-2 列出計時器與通用輸出入暫存器共用接腳的情形，表 8.2.1-3 列出本書實驗單元使用的一些暫存器，TCTL（+0x00）暫存器用來控制計時器及計數器的啟動，工作模式及計數來源等，TCNT（+0x04）就是 16 位元的計數器，PSCL（+0x08）預除除頻值，先將計數源脈波頻率除以（PSCL+1）後才是 TCNT 的計數頻率，PCNT（+0x0C）除頻計數器，MATxn（+0x30、+0x34、+0x38、+0x3C）匹配暫存器 MAT0a、MAT0b、MAT1a、MAT1b 等。TCNT_IE（+0x44）中斷致能暫存器設定是否發出中斷請求信號，TCNT_IF（+0x40）中斷旗標暫存器標示何種狀況發生，寫入 1 可以清除該位元。

表 8.2.1-1 TIMER0 ~ 5 位址空間

TIMER0	0x0020_1C00~0x0020_1FFF
TIMER1	0x0020_2000~0x0020_23FF
TIMER2	0x0020_2400~0x0020_27FF
TIMER3	0x0020_2800~0x0020_2BFF
TIMER4	0x0020_2C00~0x0020_2FFF
TIMER5	0x0020_3000~0x0020_33FF

表 8.2.1-2 GPIO 與 TIMER 共享腳位對照表

GPIO 腳位	TIME 腳位 R	GPIO 腳位	TIME 腳位 R
GPIO_B12	TMR0_capi0	GPIO_C6	TMR3_capi0
GPIO_B13	TMR0_capi1	GPIO_C7	TMR3_capi1
GPIO_B14	TMR0_mato0	GPIO_C8	TMR3_mato0
GPIO_B15	TMR0_mato1	GPIO_C9	TMR3_mato1
GPIO_A0	TMR1_capi0	GPIO_D8	TMR4_capi0
GPIO_A1	TMR1_capi1	GPIO_D9	TMR4_capi1
GPIO_A2	TMR1_mato0	GPIO_D10	TMR4_mato0
GPIO_A3	TMR1_mato1	GPIO_D11	TMR4_mato1
GPIO_C13	TMR2_capi0	GPIO_C0	TMR5_capi0
GPIO_B5	TMR2_capi1	GPIO_C1	TMR5_capi1
GPIO_B6	TMR2_mato0	GPIO_C2	TMR5_mato0
GPIO_B7	TMR2_mato1	GPIO_C3	TMR5_mato1

表 8.2.1-3 計時器暫存器功能設定

索引位址	位元位置	名稱	功能設定	R/W	預設初值
0x00	15	tctl_en(timer enable)	啟動計時器 1：啟動計時器	R/W	0
	7	tctl_de(debug enable)	1：除錯模式下，計時器繼續一般計數工作，0：除錯模式下，計時器停止計時並保持原來的值。	R/W	1
	6	tctl_iosw (I/O swap)	1：將 capix/matox I/O 定義及方向互換 0：沒有影響	R/W	1
	5, 4	tctl_sel(counter input selec)	當成計數器時，選擇計數源 00：cap0 01：cap1 10：mati0 11：mati1	R/W	cr3.DE

	3, 2	tctl_mode(counter/timer mode)	工作模式 00：timer（計時模式） 01：rising edge counter（正緣計數器） 10：falling edge counter（負緣計數器） 11：both edge counter（雙緣計數器）	R/W	0
	1	counter reset(counter reset)	計數器重置，清除為 0	R/W	0
	0	tctl_st(timer start/stop)	計數器/計時器開始計數	R/W	0
0x04	15：0	tcnt(timer/counter)	每（pscl+1）計數源計數一次	R/W	0
0x08	15：0	pscl (prescaler setting)	計數來源的預除器的除頻值	R/W	0
0x0C	15：0	pcnt	根據計數來源計數的計數器	R/W	0
0x10	11:10	cctl_capdf(digital filter setting for capi0 & capi1)	數位濾波設定值 00：沒作用 01：2CLKs 10：4CLKs 11：8CLKs	R/W	0
	9	cctl_c1sw(invert capi1 for cap1)	將 capi1 的輸入反相	R/W	0
	8	cctl_c0sw(invert capi0 for cap0)	將 capi0 的輸入反相	R/W	0
	3	cctl_cap1f(capture enable on cap1 falling edge)	當 cap1 的信號出現負緣時，啟動擷取致能	R/W	0
	2	cctl_cap1r(capture enable on cap1 rising edge)	當 cap1 的信號出現正緣時，啟動擷取致能	R/W	0
	1	cctl_cap0f(capture enable on cap0 falling edge)	當 cap0 的信號出現負緣時，啟動擷取致能	R/W	0
	0	cctl_cap0r(capture enable on cap0 rising edge)	當 cap0 的信號出現正緣時，啟動擷取致能	R/W	0
0x1C	7	moctl_m1bs(mat1b stop counter)	mat1b=counter 時停止計數	R/W	0
	6	moctl_m1br(mat1b reset counter)	mat1b=counter 時重置計數器	R/W	0
	5	moctl_m1as(mat1a stop counter)	mat1a=counter 時停止計數	R/W	0
	4	moctl_m1ar(mat1a reset counter)	mat1a=counter 時重置計數器	R/W	0
	3	moctl_m0bs(mat0b stop counter)	mat0b=counter 時停止計數	R/W	0
	2	moctl_m0br(mat0b reset counter)	mat0b=counter 時重置計數器	R/W	0
	1	moctl_m0as(mat0a stop	mt0a=counter 時停止計數	R/W	0

		counter)			
	0	moctl_m0ar(mat0a reset counter)	mat0a=counter 時重置計數器	R/W	0
0x30	15:0	mat0a (cap0r) match mat0a register	匹配輸出工作模式下當成 mat0a 匹配暫存器，擷取輸入模式下當成 cap0r 暫存器	R/W	0
0x34	15:0	mat0b (cap0f) match mat0a register	匹配輸出工作模式下當成 mat0b 匹配暫存器，擷取輸入模式下當成 cap0f 暫存器	R/W	0
0x38	15:0	mat1a (cap1r) match mat0a register	匹配輸出工作模式下當成 mat1a 匹配暫存器，擷取輸入模式下當成 cap1r 暫存器	R/W	0
0x3C	15:0	mat1b (cap1f) match mat0a register	匹配輸出工作模式下當成 mat1b 匹配暫存器，擷取輸入模式下當成 cap1f 暫存器	R/W	0
0x40	15	of_tcnt(overflow flag for tcnt)	計數器溢位旗標，tcnt 計數值由 0xffffh 進位到 0x0000 時，將此位元設為 1	R/W	cr3.DRDE
	11	of_cap1f(overflow flag for cap1f)	cap1f 溢位旗標，產生二次或更多中斷	R/W	0
	10	of_cap1r(overflow flag for cap1r)	cap1r 溢位旗標	R/W	0
	9	of_cap0f(overflow flag for cap0f)	cap0f 溢位旗標	R/W	0
	8	of_cap0r(overflow flag for cap0r)	cap0r 溢位旗標	R/W	0
	7	if_mat1b(interrupt flag for mat1b)	mat1b 中斷旗標	R/W	0
	6	if_mat1a(interrupt flag for mat1a)	mat1a 中斷旗標	R/W	0
	5	if_mat0b(interrupt flag for mat0b)	mat0b 中斷旗標	R/W	0
	4	if_mat0a(interrupt flag for mat0a)	mat0a 中斷旗標	R/W	0
	3	if_cap1f(interrupt flag for cap1f)	cap1f 中斷旗標	R/W	0
	2	if_cap1r(interrupt flag for cap1r)	cap1r 中斷旗標	R/W	0
	1	if_cap0f(interrupt flag for cap0f)	cap0f 中斷旗標	R/W	0
	0	if_cap0r(interrupt flag for cap0r)	cap0r 中斷旗標	R/W	0
0x44	7	ie_mat1b (interrupt enable for mat1b)	中斷致能	R/W	0

	6	ie_mat1a(interrupt enable for mat0a)	中斷致能	R/W	0
	5	ie_mat0b(interrupt enable for mat0b)	中斷致能	R/W	0
	4	ie_mat0a(interrupt enable for mat0a)	中斷致能	R/W	0
	3	ie_cap1f(interrupt enable for cap1f)	中斷致能	R/W	0
	2	ie_cap1r(interrupt enable for cap1r)	中斷致能	R/W	0
	1	ie_cap0f(interrupt enable for cap0f)	中斷致能	R/W	0
	0	ie_cap0r(interrupt enable for cap0r)	中斷致能	R/W	0

8.2.2 WT59F064 TIMER 實作

本實驗單元旨在探討計時器簡易的計時功能，利用 Timer0 完成此實驗。主程式 main( ) 首先設定系統時脈來源為外部 24MHz 的石英振盪晶體 [System_Control_00 = (System_Control_00 | 0x000E); System_Control_04 = (System_Control_04 | 0x0001)]，接著呼叫 InitialTMR0() 初始化計時器，其中停止計數，清除計時器值 [TMR0_TCTL |= (1<< TCTL_RST_OffSet);]，設定工作模式為計時器，設定除頻器的除頻值為 23999[TMR0_PSCL = 23999;]，因為計數時脈是 24MHz 的時脈，因此真正送到 TCNT 的時脈將成為 24MHz/24000=1KHz 的方波，最後停止計時。回到主程式後先設定匹配暫存器 MAT0a 為 1000[*((unsigned int *)(TIMER0+MAT0A_OFF)) = 1000;]，意即計數器每計數 1000 後產生中斷設定中斷旗標位元 [*((unsigned int *)(TIMER0+TCIE_OffSet)) |= 1 << MAT0AIE_OffSet]，最後啟動計時 [TMR0_TCTL |= 1 << TCTL_START_OffSet]，接著設定一變數 counter 用以記錄計數器與匹配暫存器匹配的次數，原則上是一秒增加一次，在 while 迴圈中檢查匹配暫存器 mat0a 是否產生中斷 [*((unsigned int *) (TIMER0+TMSTA_OffSet)) 看是否為 1 決定]。若出現中斷旗標位元表示已經經歷一秒將變數 counter 加 1，並重置計數器 [TMR0_TCTL |= (1<< TCTL_RST_OffSet)]，並將旗標位元清除 [*((unsigned int *)(TIMER0+TMS

TA_OffSet)) |= (1<<MAT0AIF_OffSet)]，然後開始將 counter 的值顯示於 LCM 上，顯示的格式是「 counter=XXXX 」，其中 XXXX 是 counter 轉換成 10 進制的值。程式中每當計時一秒時便會以程式清除計數器，如果要設定成自動清除的話，可以利用索引為位址 0x1C 的暫存器中 moctl_m0ar 位元來進行。以下是執行步驟：

Step 1. 將滑鼠移入專案瀏覽（ Project Viewer ）子視窗中點擊滑鼠右鍵後，選 import 做專案匯入 ADP-WT59F064-TIMER 專案。

Step 2. 打開 General 選擇「 Existing Projects into Workspace 」後，按 Next > 。

Step 3. 因為匯入的是專案壓縮檔，因此點擊「 Select archive file 」後再打開點擊「 Browse 」選項選擇要匯入的專案。

Step 4. 點選「 ADP-WT59F064-TIMER.zip 」這個專案壓縮檔後，按開啟舊檔(O)。

Step 5. 開啟專案壓縮檔後可見到專案檔的完整名稱（ ADP-WT59F064-TIMER ），在專案檔名稱左方勾選者表示要匯入此專案，然後點擊下方「 Finish 」選項。

Step 6. 透過「 Build Project 」編譯專案檔裡的程式碼。先以滑鼠左鍵點選「 ADP-WT59F064-TIMER 」使其出現反白然後按滑鼠右鍵後再點選「 Build Project 」進行編譯連結的工作。

Step 7. 把 Step 6. 中建置完成的「 ADP-WT59F064-TIMER.bin 」檔透過燒錄程式 ISP_WT59F064.exe 燒錄到晶片中。對專案 ADP-WT59F064-TIMER 按右鍵出現下拉式選單，再點擊「 Flash Burner 」以開啟燒錄視窗。

Step 8. 燒錄程式視窗中可以見到燒錄程式檔名，要燒錄到晶片中的程式碼路徑及檔案名稱，燒錄前請再次確認是否正確。透過點擊「 Auto 」一次完成整個燒錄過程，當出現「 Verify Successful 」，表示燒錄成功。

Step 9. 程式執行時可以到到 LCM 第一列出現 counter=XXXX 等文字，等號後的數字每秒增加 1 持續增加直到出現 9999 後再經一秒回復到

0000。圖 8.2.2-1 ADP-WT59F064-TIMER 專案執行 LCM 顯示結果是 ADP-WT59F064-TIMER 專案執行時 LCM 顯示的結果。

圖 8.2.2-1 ADP-WT59F064-TIMER 專案執行 LCM 顯示結果

8.3 ADP-XC5-for-N801-S Real-Time Clock（RTC）

8.3.1 ADP-XC5-for-N801-S TIMER 特性

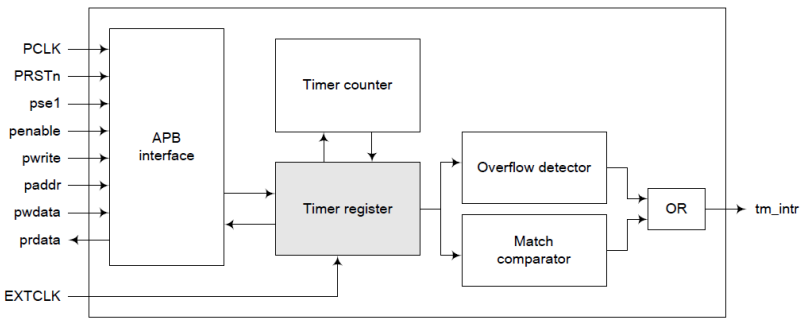

圖 8.3.1-1 FTTMR010 方塊圖

ADP-XC5-for-N801-S 提供的 TIMER 採智原科技的計時器模組，每個計時器模組內部含有 3 組各自獨立作業的計時器。由圖 8.3.1-1 FTTMR010 方塊圖可以知道每個計時器是由控制暫存器（Timer register）、計時計數器（Timer counter）、溢位偵測器、匹配比較器等方塊組合而成，計數的來源可以選擇 PCLK 或 EXTCLK，計數器可以上數亦可選擇下數模式，每次重置時會由 TmLoad 載入計時器初值，計時器載入初值後必須等待被致能啟動才可開始計數，不論是計時器溢位或是其值等於任一匹配暫存器皆可產生中斷請求信號，不過中斷信號是否確實送出將依控制暫存器的設定而定。

8.3.2 ADP-XC5-for-N801-S TIMER 特性與使用設定

表 8.3.2-1 列出了 FTTMR010 所有的暫存器，其中匹配暫存器（TmxMatchn）與計時器初值（TmxLoad）需視計數週期事先設定，至於控制暫存器（TmCR）決定計數來源、上數或下數，以及匹配暫存器發生與計時器匹配時是否產生中斷或是計時器溢位時是否產生中斷，以上皆可由表 8.3.2-2 得到詳細說明。請注意當計時器上數時溢位是指由 0xFFFFFFFF 繼續計數產生溢位，但是當下數時溢位是指下數到 0 時繼續下數產生的。至於中斷遮罩的設定與中斷發生時如何判斷中斷來源，可分別由中斷遮罩暫存器（IntrMask）以及中斷狀態暫存器（IntrState）設定與識別，詳見表 8.3.2-3 與表 8.3.2-4。

表 8.3.2-1 ADP-XC5-for-N801-S TIMER 暫存器列表（基底位址：0x00F04000）

索引位址	R/W	位元寬度	名稱	設定說明	預設初值
0x00	R/W	32	Tm1Counter	Timer1 計數器	0x--
0x04	R/W	32	Tm1Load	Timer1 自動載入計時器的設定初值	0x--
0x08	R/W	32	Tm1Match1	Timer1 匹配暫存器 1	0x--
0x0C	R/W	32	Tm1Match2	Timer1 匹配暫存器 2	0x--
0x10	R/W	32	Tm2Counter	Timer2 計數器	0x--
0x14	R/W	32	Tm2Load	Timer2 自動載入計時器的設定初值	0x--
0x18	R/W	32	Tm2Match1	Timer2 匹配暫存器 1	0x--
0x1C	R/W	32	Tm2Match2	Timer2 匹配暫存器 2	0x--
0x20	R/W	32	Tm3Counter	Timer3 計數器	0x--
0x24	R/W	32	Tm3Load	Timer3 自動載入計時器的設定初值	0x--
0x28	R/W	32	Tm3Match1	Timer3 匹配暫存器 1	0x--
0x2C	R/W	32	Tm3Match2	Timer3 匹配暫存器 2	0x--
0x30	R/W	12	TmCR	Timer1, Timer2, Timger3 控制暫存器	0x0
0x34	R/W	9	IntrState	FTTMR010 中斷狀態暫存器	0x0
0x38	R/W	9	IntrMask	FTTMR010 中斷遮罩暫存器	0x0

表 8.3.2-2 計時器控制暫存器（Timer control register）位元功能設定

位元	名稱	功能設定
0	Tm1Enable	Timer1 enable bit， 1：致能啟動計時器
1	Tm1Clock	Timer1 clock source 0：計時器計數源選擇 PCLK 1：計時器計數源選擇 EXT1CLK
2	Tm1OFEnable	Timer1 overflow interrupt enable bit 1：計時器溢位時產生有效的中斷
3	Tm2Enable	Timer2 enable bit 1：致能啟動計時器
4	Tm2Clock	Timer2 clock source 0：計時器計數源選擇 PCLK 1：計時器計數源選擇 EXT1CLK
5	Tm2OFEnable	Timer2 overflow interrupt enable bit 1：計時器溢位時產生有效的中斷
6	Tm3Enable	Timer3 enable bit 1：致能啟動計時器
7	Tm3Clock	Timer3 clock source 0：計時器計數源選擇 PCLK 1：計時器計數源選擇 EXT1CLK
8	Tm3OFEnable	Timer3 overflow interrupt enable bit 1：計時器溢位時產生有效的中斷
9	Tm1UpDown	Timer1 Up or Down count 0：計時器 1 由計數初值下數到 0，當計數器 1 再往下數時產生溢位中斷並由 Tm1Load 重新載入計數初值。 1：計時器 1 由計數初值上數到 0XFFFFFFFF，當計數器 1 再往上數時產生溢位中斷並由 Tm1Load 重新載入計數初值。
10	Tm2UpDown	Timer2 Up or Down count 0：計時器 2 由計數初值下數到 0，當計數器 2 再往下數時產生溢位中斷並由 Tm2Load 重新載入計數初值。 1：計時器 2 由計數初值上數到 0XFFFFFFFF，當計數器 2 再往上數時產生溢位中斷並由 Tm2Load 重新載入計數初值。
11	Tm3UpDown	Timer3 Up or Down count 0：計時器 3 由計數初值下數到 0，當計數器 3 再往下數時產生溢位中斷並由 Tm3Load 重新載入計數初值。 1：計時器 3 由計數初值上數到 0XFFFFFFFF，當計數器 3 再往上數時產生溢位中斷並由 Tm3Load 重新載入計數初值。

表 8.3.2-3 FTTMR010 計時器遮罩暫存器

Bit	Name	Description
0	MTm1Match1	Mask Tm1Match1 interrupt. 1：遮罩因此暫存器值等於計時器時而產生的中斷
1	MTm1Match2	1：遮罩因此暫存器內含值等於計時器時產生的中斷
2	MTm1Overflow	Tm1Overflow interrupt. 1：遮罩因 Tmr1 溢位產生的中斷
3	MTm2Match1	Tm2Match1 interrupt. 1：遮罩因此暫存器值等於計時器時產生的中斷
4	MTm2Match2	Tm2Match2 interrupt. 1：遮罩因此暫存器值等於計時器時產生的中斷
5	MTm2Overflow	Tm2Overflow interrupt. 1：遮罩因 Tmr2 溢位產生的中斷
6	MTm3Match1	Tm3Match1 interrupt. 1：遮罩因此暫存器值等於計時器時產生的中斷
7	MTm3Match2	Tm3Match2 interrupt. 1：遮罩因此暫存器值等於計時器時產生的中斷
8	MTm3Overflow	Tm3Overflow interrupt. 1：遮罩因 Tmr1 溢位產生的中斷

表 8.3.2-4 FTTMR010 中斷狀態暫存器

位元	名稱	功能設定
0	Tm1Match1	Tm1Match1 interrupt 1：表示 Tmr1 計數器值等於 Tm1 匹配暫存器 1
1	Tm1Match2	Tm1Match2 interrupt 1：表示 Tmr1 計數器值等於 Tm1 匹配暫存器 2
2	Tm1Overflow	Tm1Overflow interrupt 1：表示 Tmr1 計時器溢位發生
3	Tm2Match1	Tm2Match1 interrupt 1：表示 Tmr2 計數器值等於 Tm2 匹配暫存器 1
4	Tm2Match2	Tm2Match2 interrupt 1：表示 Tmr2 計數器值等於 Tm2 匹配暫存器 2
5	Tm2Overflow	Tm2Overflow interrupt 1：表示 Tmr2 計時器溢位發生
6	Tm3Match1	Tm3Match1 interrupt 1：表示 Tmr3 計數器值等於 Tm3 匹配暫存器 1
7	Tm3Match2	Tm3Match2 interrupt 1：表示 Tmr3 計數器值等於 Tm3 匹配暫存器 2
8	Tm3Overflow	Tm3Overflow interrupt 1：表示 Tmr3 計時器溢位發生

8.3.3 ADP-XC5-for-N801-S TIMER 實作

- **N8_Timer_XC5 實作**

本實作單元旨在熟悉 N801 SoC 中 Timer 的特性，利用 Timer 計時的特性，設定計時器向下計時，每秒發生一次溢位，溢位時計時器會設定狀態暫存器，並且將一計數器變數加 1，將此計數器的值轉化成 BCD 碼後，將十位與個位數顯示於 XC5 的七段顯示器上。以上針對本單元的工作原理加以說明，緊接著將以專案架構與程式碼的內容進一步說明，本專案名稱 N8_Timer_XC5，其中只有 main.c 與 interrupt.c 兩支程式，interrupt.c 中提供 Tmr_TickInit() 作為 Timer 的初始化設定與啟動計時，由於選用系統時脈 15MHz 為計時來源而且本實驗目的是每秒將計數器加 1 達到計時的目的，因此將計時器設定為向下計數且將計數初值設為 15000000，換句話說計數器每計數 15000000 個脈波就會產生溢位（underflow），也就是一秒溢位一次，由於溢位產生時 Timer 會主動從 Tm1Load 暫存器主動載入計數初值，因此就會循迴不斷地計時。請注意 Tmr_TickInit( ) 是從範例專案 demo-int 中改編而來。至於主程式 main( ) 除了呼叫 Tmr_TickInit( ) 進行計時器初始化設定外，必須不斷地讀取狀態暫存器 IntrState 檢查 Tm1Overflow 位元是否為 1，如果為 1 表示產生溢位也就是已經經過 1 秒，必須將計數器 cnt 加 1。

```
#define MB_PCLK 15000000
void Tmr_TickInit()
{
        unsigned int period;
        period = (1*(MB_PCLK)/1) >> 0;//計數初值設為15000000
        outw(TM1_LOAD, period); //計數載入初值15000000;
        outw(TM1_CNTR, period); //計數初值設為15000000;
        outw(TM_MSK, ~0x4); //除了Timer1的溢位中斷，遮罩所有中斷請求信號;
        //下數，計時器1啟動，TM1溢位中斷致能，TM1選擇PCLK作為計時來源
        outw(TM_CR, (TM_CR_TM1ENA | TM_CR_TM1OEN));
}
```

計數器加 1 後必須透過 GPIO 送到七段顯示器上，由於 XC5 平台七段顯器只有兩個因此只能顯示十位數與個位數，透過運算「(cnt%100)/10」可得到十位數的 BCD 碼，運算「(cnt%100)%10」可得個位數 BCD 碼。請注意使用者必須自行清除狀態暫存器否則會不斷地誤判溢位中斷產生，造成計數時間不正確。

```
int main()
{
        unsigned int cnt=0;
        Tmr_TickInit();//初始化計時器，啟動計時器
        while(1)
        {
                if (inw(TM_STA) & TM_STA_TM1OV) //檢查是否產生溢位
                {
                        cnt++;
                        //取個位數與十位數送到七段顯示器顯示計數器的值
                        outw(LED_BASE, (cnt%100)%10|((cnt%100)/10)<<4);
                        outw(TM_STA, 0x1FF); //清除狀態暫存器以免誤判          }
        }
        return 0;
}
```

因為此一專案建置容易，因此將示範從無到有的建置方式完成，以下將詳列每一步驟：

Step 1. 以圖 8.3.3-1、圖 8.3.3-2、圖 8.3.3-3 等三種方式皆可開啟相同的專案建置視窗，依照圖中編號順序操作即可開啟以一般建置專案的方式。

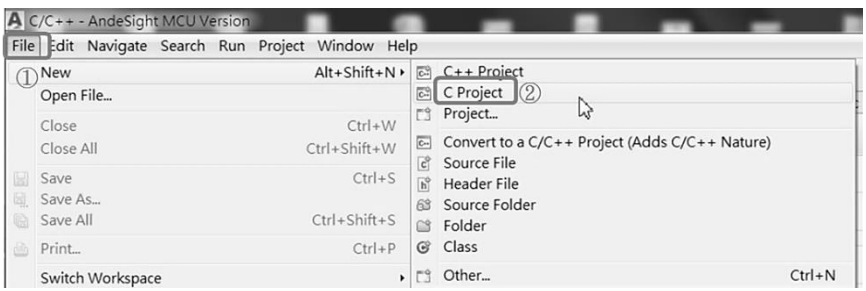

圖 8.3.3-1 工具列上「File」選單建置專案

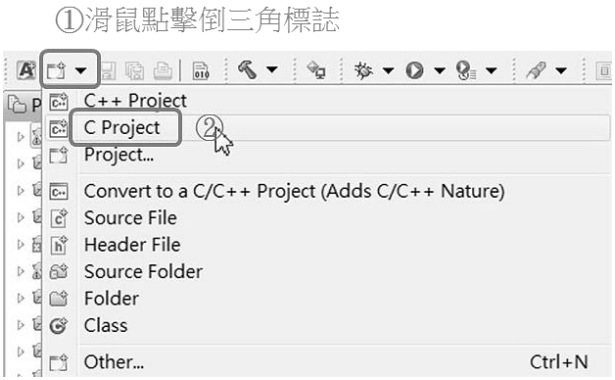

圖 8.3.3-2 工具列上圖控方式建置專案

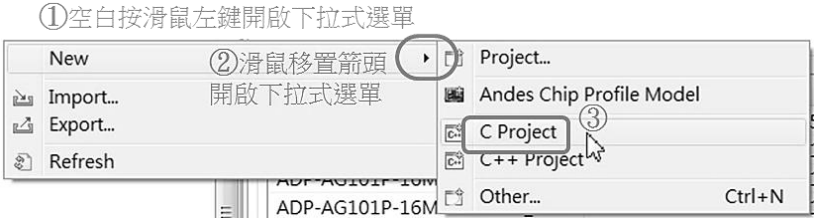

圖 8.3.3-3 專案瀏覽視窗空白處按壓滑鼠右鍵建置專案

Step 2. 在 Step 1. 中開啟輸入專案名稱設定視窗，依圖 8.3.3-4 所示依序點選，並於 ③ 處依需求選擇工具鏈，此一 SoC 可以使用 nds32le-elf-mculib-v3m 或 nds32le-elf-newlib-v3m，最終記得輸入專案名稱完成專案建置設定（圖 8.3.3-4 ④）。

Step 3. 專案名稱設定完畢後在專案瀏覽視窗中可以見到新建專案（圖 8.3.3-5 ①），點選此專案在空白處按滑鼠右鍵開啟下拉式選單，並將滑鼠移到「New」處可再開啟另一層下拉式選單。若需要建立程式碼目錄以專門存放程式碼，則依圖 8.3.3-5 右上角的步驟進行並開啟圖 8.3.3-6 輸入目錄名稱。以上完成後直接依圖 8.3.3-5 右下角直接進入圖 8.3.3-7 建立程式檔的程序。

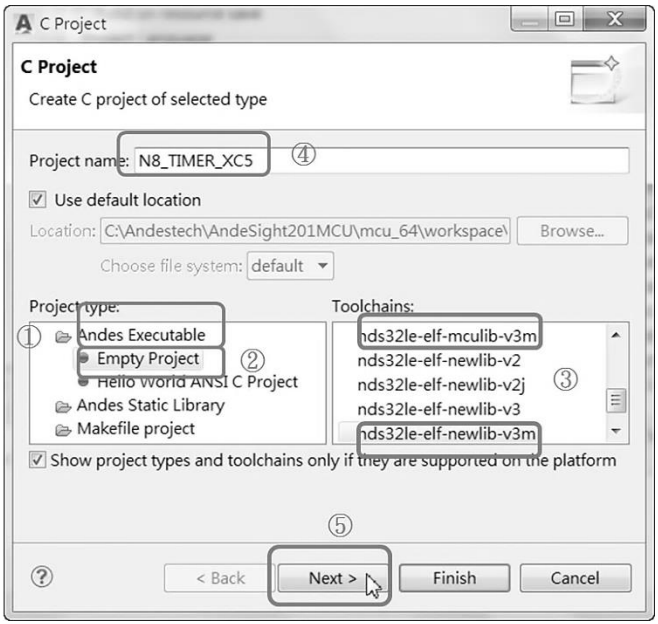

圖 8.3.3-4 輸入專案名稱、選擇工具鏈

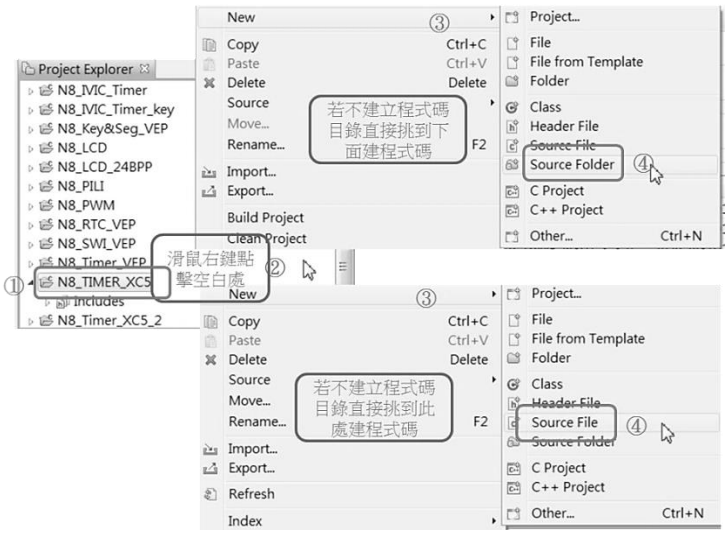

圖 8.3.3-5 建立程式檔目錄及程式檔或標頭檔 (*.h)

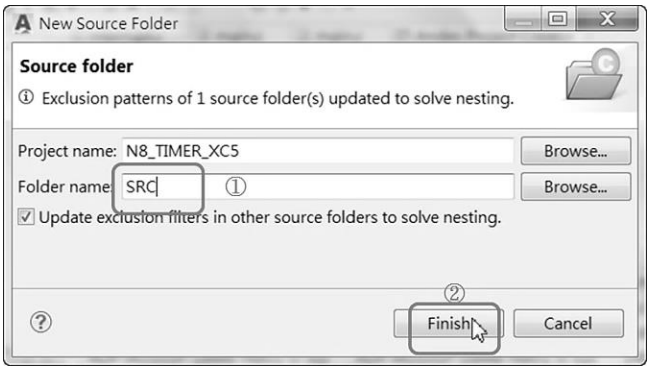

圖 8.3.3-6 建立程式檔目錄

Step 4. 圖 8.3.3-5 完成後不論是否建置檔案目錄，圖 8.3.3-7 是必經的程序，在 ① 輸入檔案名稱，點擊圖 8.3.3-7 ② 的倒三角標誌開啟下拉式選單選擇預設範本程式後，按下「Finish」完成程式檔建置。Step 3. 及 4. 不僅可建立程式檔也可用來編輯 Header File 或是其他檔案。在此步驟請完成「main.c」及「interrupt.c」和其他標頭檔的建置（ag101_16mb.h），使用者可直接匯入或是複製貼上亦可，無需自行編輯，重點是要知道建置專案的流程。最後要複製「nds32-16mb.ld」到專案目錄下，該檔案是本專案編譯連結時的連結敘述檔。

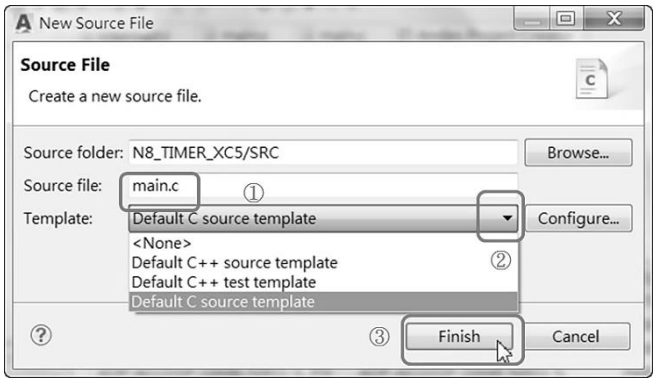

圖 8.3.3-7 輸入程式檔檔名以進一步編輯程式

圖 8.3.3-8 AndeSight 提供的 C 語言程式範本

Step 5. 專案所需檔案編輯完畢後，必須設定專案執行的工作平台，開啟設定視窗如圖 8.3.3-9，開啟設定視窗（圖 8.3.3-10）後，點擊左方「Target Configuration」後出現右方「Target Configuration」視窗，以滑鼠左鍵點擊「Browse」選項（②）選擇目標工作平台「ADP-AG101P-16MB-N801-S」，接著點選主機工作平台連線方式「AICE」，點擊「Apply」引用以上的設定在此專案，並按「OK」（圖 8.3.3-10 ⑤）完成設定或略過此步驟直接點擊左方「Setting」選項（圖 8.3.3-11 ①）。

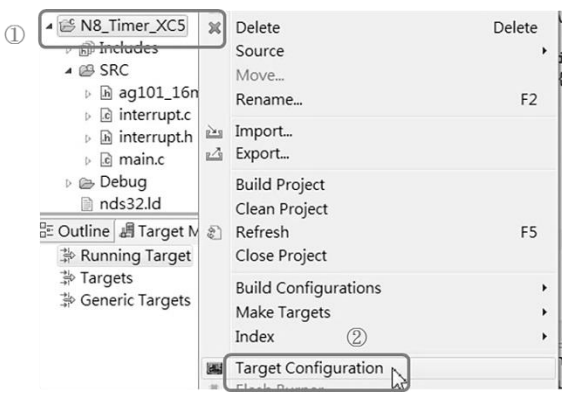

圖 8.3.3-9 開啟設定專案工作平台視窗

Step 6. 由 Step 5. 直接跳轉此項（圖 8.3.3-11）或依圖 8.3.3-10 開啟圖 8.3.3-11，依標號順序進行到連結檔的設定，記得在圖 8.3.3-11 ④輸入「../nds32-16mb.ld」。

Step 7. 在圖 8.3.3-12 左方依序完成編譯連結後，可以看見該圖右方「專案瀏覽視窗」N8_TIMER_XC5 多了目錄「Debug」。

Step 8. 執行步驟請參閱 N8_UART_XC5 專案 Step 5.。圖 8.3.3-13 是執行結果。

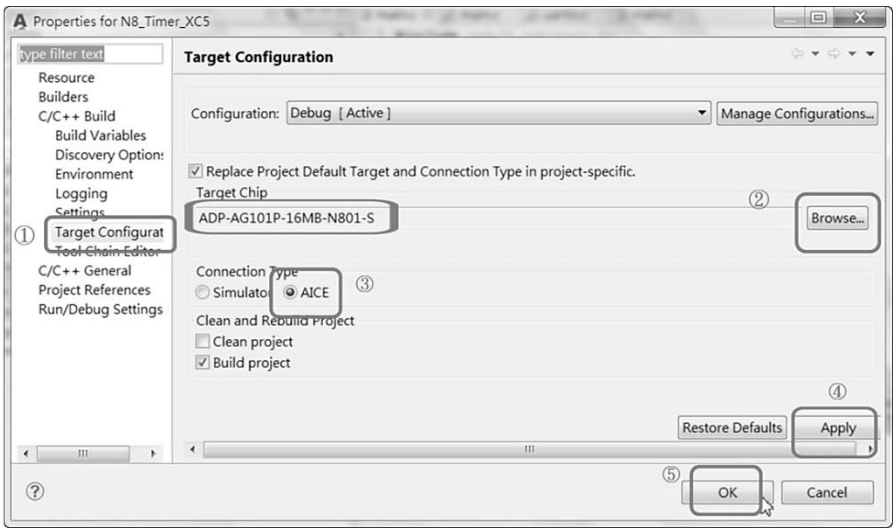

圖 8.3.3-10 工作平台設定程序

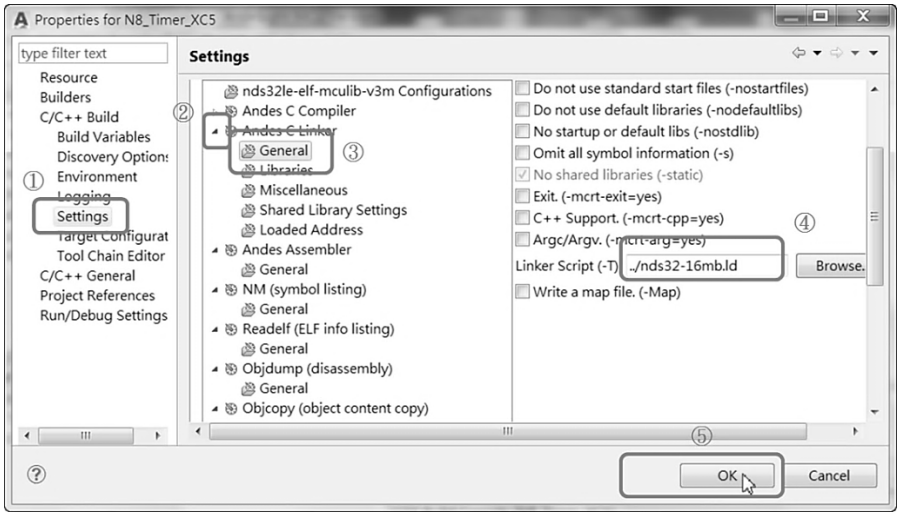

圖 8.3.3-11 設定編譯選項

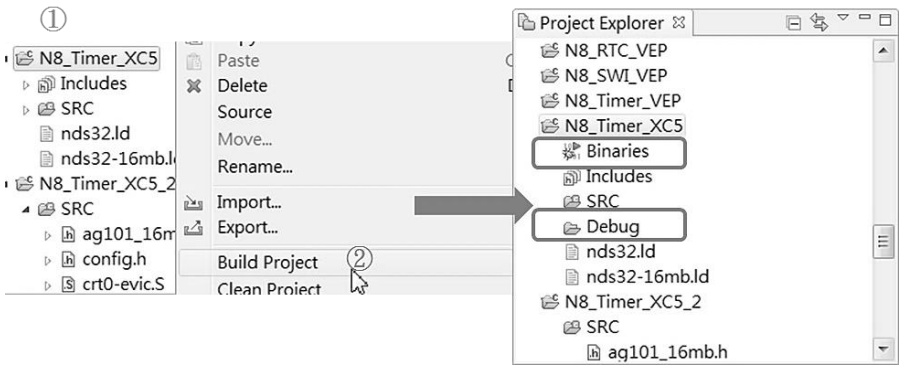

圖 8.3.3-12 編譯連結程式

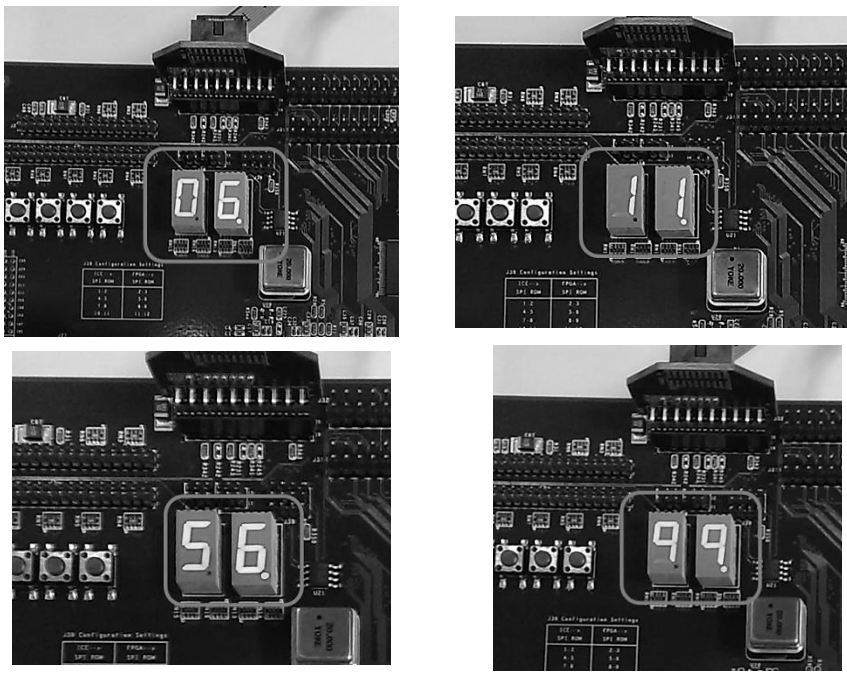

圖 8.3.3-13 N8_TIMER_XC5 執行畫面