Time Keeper A

計時模組

版本: V2.0

產品介紹: 利基 Time Keeper A 模組可以提供多樣的計時與提醒功能，內建有星期對應，輸入日期就可以知道當天是星期幾。另外提供第二時間可供設定，以及五個額外的倒數計時器，滿足使用者同時處理多個，需要同時判斷的倒數計時需求。也設有微調值，如果需要較精準的時間，可以透過微調設定，縮小每天的時間誤差，使其在 0.08 秒內。

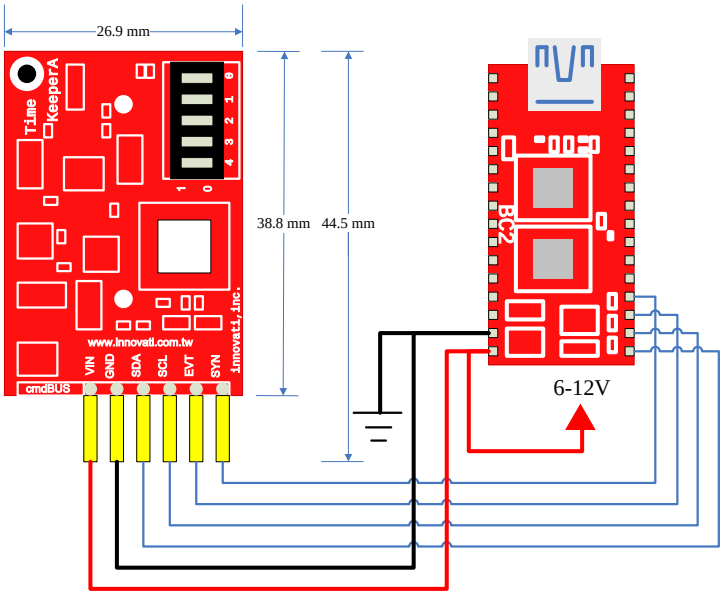
應用方向:

- 可加上 LCD 模組即時顯示時間，就是簡單的電子時鐘。
- 與動態的其他種類模組結合，可以於所需要的時間啟動模組動作。
- 可以輕鬆做出日曆的應用，並以各種提醒方式，做出多樣化備忘記事。
- 可以加上開關做出各種家電定時開關或是預設啟動。

產品特色:

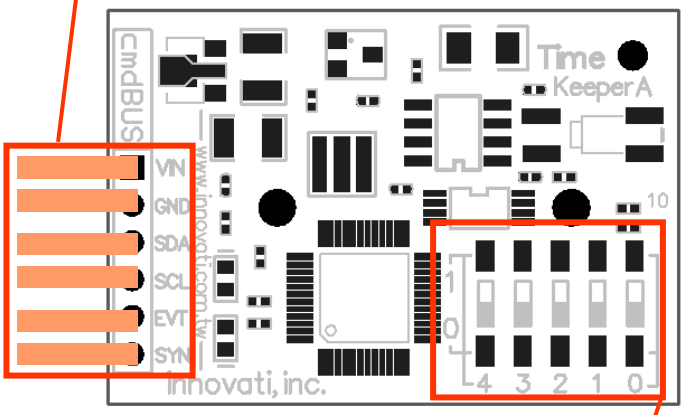
- 西元 2000~2099 年的年，月，日，星期，時，分，秒自動計時。
- 提供二十四小時與十二小時制時間顯示。
- 次要時間設定，可提供時，分，秒的輸入設定。
- 多樣化的提醒方式，可設定每秒，每分，每小時，每天，每星期，每月等不同模式，且可多項並存。
- 5 個倒數計時器，可以同時設定最大以日為單位，最小為秒的倒數值。
- 可微調精準度，讓時間誤差減少。最佳可達 3.052ppm。
- 可透過 I2C 方式，下達指令。

連接方式: 直接將 ID 開關撥至欲設定的編號，再將 cmdBUS 連接至 Basic Commander 上對應的腳位，就可透過 Basic Commander 執行操作。



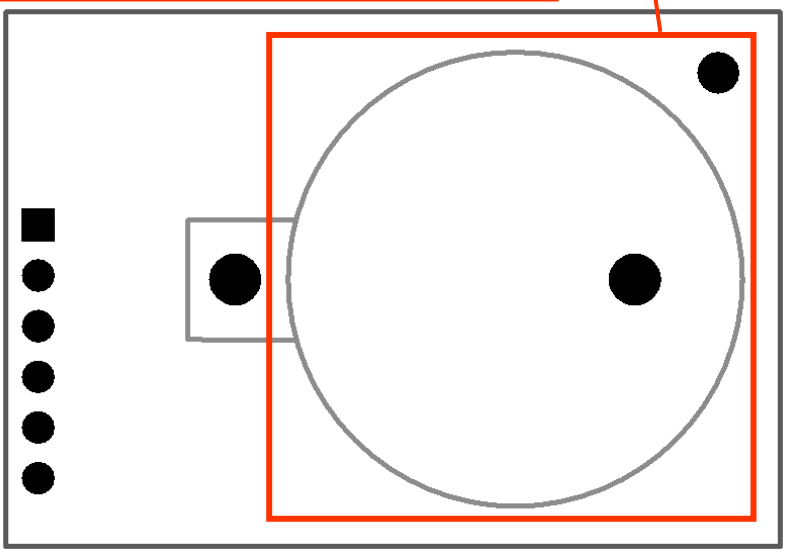
產品規格:

cmdBUS接腳，將此處腳位與Basic Commander對應腳位相接，即可透過Basic Commander操控Timer模組(連接時請注意腳位對應，將Vin對接Basic Commander上的Vin腳位，若是腳位錯誤可能造成模組損毀)



模組編號設定開關，由右至左以二進制設定Timer模組的模組編號，編號可以讓Basic Commander操控時，判斷想要控制的模組。

電池安裝位置，請將電池安裝於此處，再開始Timer模組的操作。



使用電流量: 約 9 mA

操作注意事項:

請使用 CR2032 電池，更換不同電池請確定電池的操作溫度限制。

模組操作溫度　　0 ℃~ 70 ℃ (此為模組溫度，不含電池)

模組儲存溫度　　-50 ℃~125℃

模組下達指令的方式可分為兩種：cmdBUS、I2C 控制方式

cmdBUS 指令表:

下面的指令表是專供控制 Time Keeper A 的各種指令，必要輸入的指令名稱與參數，以粗底或粗斜體表示，粗體的文字在輸入時請不要更改，粗斜體的文字請自行定義適當格式的參數填入。輸入時請注意 innoBASIC Workshop 大寫與小寫會視為相同字。在執行 Time Keeper A 指令前，請先於程式開頭定義對應參數與編號，例:

Peripheral *ModuleName* As TimeKeeperA @ *ModuleID*

I2C 通訊協議(Protocol):

為了使更廣泛的使用者能控制模組，提供了部份指令的通訊協議讓使用者應用。透過通訊規格，使用者可使用 I2C 通訊協議為模組下達命令。

通訊協議常見的封包如下：

MID：模組 ID 編號，空間大小為 Byte 的變數。對應於硬體的指播開關。

CID：命令 ID 編號，空間大小為 Byte 的變數。依不同命令而改變。

CheckSum1：驗證位元_1，空間大小為 Byte 的變數。

定義方式：255 – (MID * 2) – CID

CheckSum2：驗證位元_2，空間大小為 Byte 的變數。

定義方式：255 – (CheckSum1~CheckSum2 之間的變數總和)

CheckSum3：驗證位元_3，空間大小為 Byte 的變數。

定義方式：255 – MID-(MID~CheckSum3 之間的變數總和)

Dummy：虛設位元，可為任意變數。空間大小為 Byte 的變數。

於通訊規格每筆資料空間大小階為 Byte，若資料空間大小超過一個 Byte 時，需將資料拆開，並由 Low Byte 開始傳送。

Ex:傳送資料 Temp 為一筆空間大小為 Word 的資料,則需將 Temp 拆開,分為 Temp_L、Temp_H，並且先傳送 Temp_L。

Ex1 模組編號為 2，命令編號為 153，傳送參數 Byte 為 100，通訊協議為

MID+CID+CheckSum1+Byte+CheckSum2+Dummy 則：

MID = 2

CID = 153

CheckSum1 = 255 – (2*2) – 153 = 98

Byte = 100

CheckSum2 = 255 – 100

Dummy = 0~255 之間的任意數

Ex2 模組編號為 2，命令編號為 153，傳送參數 Temp 為 511，通訊協議為

MID+CID+CheckSum1+Temp_L+Temp_H+CheckSum2+Dummy 則：

MID = 2

CID = 153

CheckSum1 = 255 – (2*2) – 153 = 98

Temp_L = 255，Temp_H = 1

CheckSum2 = 255 – Temp_L–Temp_H = 255

Dummy = 0~255 之間的任意數

指令格式	指令功能
時間設定相關指令	
Set12Hour(*Hour, AMPM*)	以 ***Hour*** 值設定為模組時間的小時 (***Hour*** 請輸入 1~12 之間的整數)，並根據 ***AMPM*** 值決定為上午或下午 (***AMPM*** 為 0 表示 AM，1 表示 PM)
SetDay(*Day*)	將 ***Day*** 設定為模組時間的日期 (***Day*** 請輸入 1~31 之間的整數，若所設定的月份沒有該日期，將會自動設為 1)
SetHour(*Hour*)	將 ***Hour*** 設定為模組時間的小時 (***Hour*** 的輸入為二十四小時制，請輸入 0~23 之間的整數)
SetMinute(*Minute*)	將 ***Minute*** 值設定為模組時間的分鐘 (***Minute*** 請輸入 0~59 之間的整數)
SetMonth(*Month*)	將 ***Month*** 值設定為模組時間的月份 (***Month*** 請輸入 1~12 的整數)
CmdBUS： **SetTimeAndDate(*Year, Month, Day, Hour, Minute, Second*)** **I2C：** **MID+88+CheckSum1** **+Year_L+Year_H** **+Month+Day+Hour** **+Minute+Second** **+CheckSum2+Dummy**	依據使用者輸入的 ***Year*** (年)，***Month*** (月)，***Day*** (日)，***Hour*** (時)，***Minute*** (分)，Seco ***Second*** nd(秒)，設定模組現在的時間 (***Year*** 請輸入 0~99 之間的整數，代表西元 2000~2099 年，其他日期輸入請參考單獨設定的範圍值)
SetSecond(*Second*)	將 ***Second*** 值設定為模組時間的秒 (***Second*** 請輸入 0~59 之間的整數)
SetSubTime(*SubID, Hour, Minute, Second*)	以 ***SubID*** 值選擇所要設定的次要時間，並將 ***Hour***，***Minute***，***Second*** 分別設為該時間的時分秒值 (共有五組次要時間，***SubID*** 請輸入 0~4 之間的整數，時間的設定值請參考單獨設定的範圍值)
SetYear(*Year*)	將 ***Year*** 值設定為模組時間的年份 (***Year*** 請輸入 0~99 之間的整數，代表西元 2000~2099 年)
時間取得相關指令	
Get12Hour(*Hour, AMPM*)	以十二小時制取得系統現在的小時，存放於 ***Hour*** 中，並以 ***AMPM*** 值表示上午或下午(***AMPM*** 為 0 表示上午，為 1 表示下午)，***Hour*** 會回傳 0~11 之間的整數
GetDay(*Day*)	取得系統現在的日存放於 ***Day*** 中，***Day*** 會回傳 1~31 之間的整數
GetHour(*Hour*)	取得系統現在的小時存放於 ***Hour*** 中，***Hour*** 會回傳 0~23 之間的整數

GetMinute(*Minute*)	取得系統現在的分存放於 ***Minute*** 中，***Minute*** 會回傳 0~59 之間的整數
GetMonth(*Month*)	取得系統現在的月存放於 ***Month*** 中，***Month*** 會回傳 1~12 之間的整數值
CmdBUS： **GetTimeAndDate(*Year, Month, Day, WeekDay, Hour, Minute, Second*)** **I2C：** **Out：** **MID+96+CheckSum1 +Dummy** **In：** **MID+Year_L+Year_H +Month+Day+Hour +Minute+Second +CheckSum3**	取得系統現在時間，年存放於 ***Year***，月存放於 ***Month***，日存放於 ***Day***，星期存放於 ***WeekDay***，小時存放於 ***Hour***，分存放於 ***Minute***，秒存放於 ***Second*** (***WeekDay*** 規則為，若是星期日以 0 代表，星期一到六則依序是 1~6，其他回傳值範圍請參考單獨取回的範圍值)
GetSecond(*Second*)	取得系統現在的秒存放於 ***Second***，***Second*** 會回傳 0~59 之間的整數
GetSubTime(*SubID, Hour, Minute, Second*)	取得 ***SubID*** 指定的次要時間，並將小時存放於 ***Hour***，分存放於 ***Minute***，秒存放於 ***Second*** (***SubID*** 的範圍請輸入介於 0~4 的整數，其他回傳值範圍請參考單獨取回的範圍值)
GetWeekDay(*WeekDay*)	取得系統現在的星期值存放於 ***WeekDay*** 中，***WeekDay*** 會回傳 0~6 之間的整數
GetYear(*Year*)	取得系統現在的年存放於 ***Year*** 中，***Year*** 會回傳 0~99 之間的整數
提醒事件相關指令	
DailyAlarmOn(*DailyAlarmID*)	啟動 ***DailyAlarmID*** 所指定的每日提醒，***DailyAlarmID*** 請輸入 0~7 之間的整數
DailyAlarmOff(*DailyAlarmID*)	關閉 ***DailyAlarmID*** 所指定的每日提醒，***DailyAlarmID*** 請輸入 0~7 之間的整數
DisableDailyEvent()	關閉 **DailyEvent** 事件
DisableHourlyEvent()	關閉 **HourlyEvent** 事件
DisableMinutelyEvent()	關閉 **MinutelyEvent** 事件
DisableSecondlyEvent()	關閉 **SecondlyEvent** 事件
EnableDailyEvent()	開啟 **DailyEvent** 事件
EnableHourlyEvent()	開啟 **HourlyEvent** 事件
EnableMinutelyEvent()	開啟 **MinutelyEvent** 事件
EnableSecondlyEvent()	開啟 **SecondlyEvent** 事件

GetDailyAlarm(*AlarmID*, *Hour*, *Minute*)	取得 ***AlarmID*** 指定的每日提醒設定值，請輸入 0~7 之間的整數，取得小時為 ***Hour***，會回傳 0~23 之間的整數，取得分為 ***Minute***，會回傳 0~59 之間的整數
GetHourlyAlarm(*AlarmID*, *Minute*)	取得 ***AlarmID*** 指定的每小時提醒設定值，請輸入 0~7 之間的整數，取得分為 ***Minute***，會回傳 0~59 之間的整數
GetMonthlyAlarm(*AlarmID*, *Day*, *Hour*, *Minute*)	取得 ***AlarmID*** 指定的每月提醒設定值，請輸入 0~7 之間的整數，取得日期為 ***Day***，會回傳 1~31 之間的整數，取得小時為 ***Hour***，會回傳 0~23 之間的整數，取得分為 ***Minute***，會回傳 0~59 之間的整數
GetWeeklyAlarm(*AlarmID*, *WeekDay*, *Hour*, *Minute*)	取得 ***AlarmID*** 指定的每星期提醒設定值，請輸入 0~7 之間的整數， ***WeekDay*** 會回傳 0~6 之間的整數，***Hour*** 會回傳 0~23 之間的整數，***Minute*** 會回傳 0~59 之間的整數
HourlyAlarmOff(*AlarmID*)	關閉 ***AlarmID*** 指定的每小時提醒 (***AlarmID*** 請輸入 0~7 之間的整數)
HourlyAlarmOn(*AlarmID*)	啟動 ***AlarmID*** 指定的每小時提醒 (***AlarmID*** 請輸入 0~7 之間的整數)
MonthlyAlarmOff(*AlarmID*)	關閉 ***AlarmID*** 指定的每月提醒 (***AlarmID*** 請輸入 0~7 之間的整數)
MonthlyAlarmOn(*AlarmID*)	啟動 ***AlarmID*** 指定的每月提醒 (***AlarmID*** 請輸入 0~7 之間的整數)
SetDailyAlarm(*AlarmID*, *Hour*, *Minute*)	設定 ***AlarmID*** 所指定的每日提醒時間，***Hour*** 請輸入 0~23 之間的整數，***Minute*** 請輸入 0~59 之間的整數 (***AlarmID*** 請輸入 0~7 之間的整數)
SetHourlyAlarm(*AlarmID*, *Minute*)	設定 ***AlarmID*** 所指定的每小時提醒時間，***Minute*** 請輸入 0~59 之間的整數 (***AlarmID*** 請輸入 0~7 之間的整數)
SetMonthlyAlarm(*AlarmID*, *Day*, *Hour*, *Minute*)	設定 ***AlarmID*** 所指定的每月提醒時間，***Day*** 請輸入 1~31 之間的整數，***Hour*** 請輸入 0~23 之間的整數，***Minute*** 請輸入 0~59 之間的整數 (***AlarmID*** 請輸入 0~7 之間的整數)
SetWeeklyAlarm(*AlarmID*, *WeekDay*, *Hour*, *Minute*)	設定 ***AlarmID*** 所指定的每星期提醒時間，***WeekDay*** 請輸入 0~6，***Hour*** 請輸入 0~23，***Minute*** 請輸入 0~59 (***AlarmID*** 請輸入 0~7 之間的整數)
WeeklyAlarmOff(*AlarmID*)	關閉 ***AlarmID*** 指定的每星期提醒 (***AlarmID*** 請輸入 0~7 之間的整數)
WeeklyAlarmOn(*AlarmID*)	啟動 ***AlarmID*** 指定的每星期提醒 (***AlarmID*** 請輸入 0~7 之間的整數)

倒數計時器相關指令	
CmdBUS： **CountDownTimerOn(*TimerID*)** **I2C：** **MID+133+CheckSum1** **+TimerID+CheckSum2+Dummy**	啟動編號為 ***TimerID*** 的倒數計時器。
CmdBUS： **CountDownTimerOff(*TimerID*)** **I2C：** **MID+134+CheckSum1** **+TimerID+CheckSum2+Dummy**	關閉編號為 ***TimerID*** 的倒數計時器。
CmdBUS： **GetCountDownTimer(*TimerID, Day, Hour, Minute, Second*)** **I2C：** **Out：** **MID+132+CheckSum1** **+Dummy** **In：** **MID+TimerID+Day+Hour** **+Miunte+Second** **+CheckSum3**	取得 ***TimerID*** 所指定的倒數計時器計時資訊，請輸入 0~7 之間的整數，***Day*** 會回傳剩餘天數，為 0~255 之間的整數，***Hour*** 會回傳剩餘小時，為 0~23 之間的整數，***Minute*** 會回傳剩餘分鐘，為 0~59 之間的整數，***Second*** 會回傳剩餘秒數，為 0~59 之間的整數
CmdBUS： **SetCountDownTimer(*TimerID, Day, Hour, Minute, Second*)** **I2C：** **MID+131+CheckSum1** **+TimerID+Day+Hour** **+Miunte+Second** **+CheckSum2+Dummy**	設定 ***TimerID*** 所指定的倒數計時器計時資訊，***Day*** 請輸入 0~255 之間的整數，***Hour*** 請輸入 0~23 之間的整數，***Minute*** 請輸入 0~59 之間的整數，***Second*** 請輸入 0~59 之間的整數 (***TimerID*** 請輸入 0~7 之間的整數)
微調與重置相關指令	
ResetTimeAndDate()	重置模組，使其回到預設值
GetAdjustment(*AdjValue*)	取得微調值 ***AdjValue***，為 0~255 之間的整數
SetAdjustment(*AdjValue*)	以 ***AdjValue*** 設定微調值，請輸入 0~255 之間的整數 (請參閱附錄 3 進行設定)

模組提供應用事件:

事件名稱 (Event)	啟動條件
倒數計時事件	
CountDownTimer0~7Event	執行 **SetCountDownTimer** 後，當倒數計時的時間為零時
特定時間提醒事件	
MonthlyAlarm0~7Event	執行 **MonthlyAlarmOn** 後，每月當模組時間到達所設定的日期與時分
WeeklyAlarm0~7Event	執行 **WeeklyAlarmOn** 後，每月當模組時間到達所設定的星期與時分
DailyAlarm0~7Event	執行 **DailyAlarmOn** 後，每天當模組時間到達所設定的時分
HourlyAlarm0~7Event	執行 **HourlyAlarmOn** 後，每小時模組時間到達所設定的分時
固定周期提醒事件	
DailyEvent	執行 **EnableDailyEvent** 後，每當模組時間的日期改變時
HourlyEvent	執行 **EnableHourlyEvent** 後，每當模組時間的小時改變時
MinutelyEvent	執行 **EnableMinutelyEvent** 後，每當模組時間的分鐘改變時
SecondlyEvent	執行 **EnableSecondlyEvent** 後，每當模組時間的秒改變時

範例程式:

```
Peripheral MyTime As TimeKeeperA @ 0                    '   設定模組編號為 0

Dim CurYear As Byte                                     '   儲存現在 年
Dim CurMonth As Byte                                    '   儲存現在 月
Dim CurDay As Byte                                      '   儲存現在 日
Dim CurWeek As Byte                                     '   儲存現在 星期
Dim CurHour As Byte                                     '   儲存現在 時
Dim CurMinute As Byte                                   '   儲存現在 分
Dim CurSecond As Byte                                   '   儲存現在 秒
Dim SecondCnt As Byte                                   '   計算顯示時間

Sub Main()                                              '   主程式
    MyTime.SetTimeAndDate(7, 9, 17, 15, 47, 0)          '   設定現在時間
    SecondCnt=0
    MyTime.EnableSecondlyEvent()                        '   啟動每秒提示事件

'   下面的迴圈會等待顯示至少五次時間後才會跳出
    Do
    Loop Until SecondCnt>5

    MyTime.DisableSecondlyEvent()                       '   取消每秒提示事件

    MyTime.SetMinute(48)                                '   設定分鐘為 48
    MyTime.SetSecond(55)                                '   設定秒為 0
    MyTime.SetHourlyAlarm(0, 49)                        '   設定編號 0 的每小時提醒於 49 分提醒
    SecondCnt=0
    MyTime.HourlyAlarmOn(0)                             '   啟動編號 0 的每小時提醒

'   下面的迴圈會等待每小時提醒發生後才會跳出
    Do
    Loop Until SecondCnt>0

    MyTime.HourlyAlarmOff(0)                            '   關閉編號 0 的每小時提醒
    MyTime.SetCountDownTimer(0, 0, 0, 0, 3)             '   設定編號 0 的倒數計時器，倒數三秒
    SecondCnt=0
    MyTime.CountDownTimerOn(0)                          '   啟動編號 0 的倒數計時器

'   下面的迴圈會等待倒數計時完成才會跳出
    Do
    Loop Until SecondCnt>0
```

```
    MyTime.CountDownTimerOff(0)                    '    關閉編號 0 的倒數計時器
End Sub

Event MyTime.SecondlyEvent()
    MyTime.GetTimeAndDate(CurYear, CurMonth, CurDay, CurWeek, CurHour, CurMinute, CurSecond)    '    取得現在時間
    Debug CurYear, "/", CurMonth, "/", CurDay, " ", CurHour, ":", CurMinute, ":", CurSecond, CR
    SecondCnt+=1
End Event

Event MyTime.HourlyAlarm0Event()
    Debug "現在是四十九分", CR
    SecondCnt+=1
End Event

Event MyTime.CountDownTimer0Event()
    Debug "倒數三秒", CR
    SecondCnt+=1
End Event
```

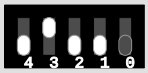
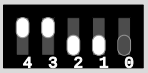
附錄

1. 已知問題:

   V1.0:

   - 使用 SetHour 設定小時候，執行 Pause 暫停一段時間，再以 GetHour 取得的小時值，有時會出現錯誤值。
   - 在閏年(2000，2004，…，2096)，會出現 31 號之後跳到 32 號
   - 設定 Hour 提醒事件，須在提醒前兩秒以上設定，才會執行最近一次的提醒
   - Week 提醒事件設定，會設定到 Day

2. 模組編號開關對應編號表:

	0		8		16		24
	1		9		17		25
	2		10		18		26
	3		11		19		27
	4		12		20		28
	5		13		21		29
	6		14		22		30
	7		15		23		31

3. 時間調整公式:

   想要使用 SetAdjustment 調整更精準的時間差異，請先使用 SetTimeAndDate 將模組設定在可取得的精確時間(如報時台)，同時記錄下所設定的時間，再將模組擺置一段時間後，觀察模組所讀取的時間與現在時間的差異值，先算出以下參數值➔

   總測試時間 = 設定時間 – 現在量測時間 (以秒為單位)

   現在差異時間 = 模組現在量測時間 – 實際時間 (以秒為單位)

   百萬差異值(即 ppm) = (現在差異時間/總測試時間) * 1000000 (註一)

   如果現在差異時間為正值，則所需填入的值為

   128 – (百萬差異值/3.052)”取整數值”

   如果現在差異時間為負值，則所需填入的值為

(百萬差異值/3.052)”取正整數值” + 1

註一　　可調整的範圍需要百萬差異值在-195.3 到 192.2 之間，如果算出的時間差異超出此範圍，則無法使用微調值調整。