DMATEK

ARM周邊配件系列

Android / ZigBee 無線感測器平台

用户手册

作業系統：WinCE 6.0 / Android 2.1 / Windows XP/7.0

適用平台：DMA-210U / DMA-6410XP / DMA-6410
HMI700-210S / HMI700-6410S

產品編號：ZB2530P-AN

目錄

目錄

目錄

目錄

第〇章　導 讀

0-1　如何開始

在本手冊中，筆者將於第一章中簡述 ZigBee 的功能與特色，於第二章說明本公司的 Android / ZigBee 無線感測器平台；第三章簡要介紹各個感測器模組以及硬體電路；第四章介紹了 Android 下平台通訊的實現過程；第五章是針對 CC2530 單晶片介紹一下基本開發流程；附錄一是 ZigBee Dubugger 模擬器介紹；附錄二是 HMI700-6410S 開發平台的介紹；附錄三則爲 ZB2530-01 模組的相關電路說明；附錄四至附錄二一則是各感測器模組的 Data 資料，依次爲：

附錄四　人體紅外感測器模組（B110）資料
附錄五　GSensor 三軸位移加速感測器模組（B120）資料
附錄六　AD590 雙端 IC 溫度感測器模組（B130）資料
附錄七　TGS822 酒精感測器模組（B140）資料
附錄八　ST-1KLA/B 光電晶體感測器模組（B150）資料
附錄九　光敏電阻感測器模組（B160）資料
附錄十　EEPROM 感測器模組（B170）資料

0-2　光碟內容說明

本通訊實驗平台產品光碟中包含了許多實用的東西，讀者可以從產品光碟的README檔案瞭解光碟的內容，主要包含以下的內容：

目錄名稱	主要檔描述
IAR\CC2530_lib\ basicrf	CC2530 單片機的無線函數庫
IAR\CC2530_lib\ board	ZIGBEE 模組上的資源函數庫
IAR\CC2530_lib\ common	CC2530 單片機的通用函數庫
IAR\CC2530_lib\ module	本公司 8 個感測器模組函數庫
IAR\CC2530_lib\ utils	CC2530 單片機的工具函數庫
IAR\software	IAR 安裝軟體以及模擬器的驅動軟體
IAR\source_exp\B110	人體感測器原始程式碼
IAR\source_exp\B130	類比溫度感測器原始程式碼
IAR\source_exp\B140	酒精感測器原始程式碼

IAR\source_exp\B170	EEPROM 原始程式碼
IAR\source_exp\B120	重力感測器原始程式碼
IAR\source_exp\B150&B160	光敏電阻感測器和光電感測器原始程式碼
IAR\source_exp\B180	直流電機原始程式碼
IAR\source_exp\ A120	溫濕度感測器原始程式碼
IAR\source_exp\ A110	陀螺儀感測器原始程式碼
IAR\source_exp\ A130	可燃氣體感測器原始程式碼
IAR\source_exp\ A140	數位溫度感測器原始程式碼
IAR\source_exp\ E130	氣體品質感測器原始程式碼
IAR\source_exp\ E110	GPS 資料獲取原始程式碼
IAR\source_exp\rf_send	Led 無線發送端根源程式
IAR\source_exp\rf_recv	Led 無線接收端根源程式
IAR\hex\ M1_IMC.hex	人體感測器燒寫檔案
IAR\hex\ M2_AD590.hex	模擬溫度感測器燒寫檔
IAR\hex\ M2_TGS.hex	酒精感測器燒寫檔案
IAR\hex\ M3_EEPROM.hex	EEPROM 燒寫檔案
IAR\hex\ M3_BMA.hex	重力感測器燒寫檔案
IAR\hex\ M4_GM.hex	光敏電阻感測器燒寫檔案
IAR\hex\ M4_GD.hex	光電感測器燒寫檔案
IAR\hex\ M5.hex	直流馬達感測器燒寫檔案
IAR\hex\ SHT10.hex	溫濕度感測器燒寫檔案
IAR\hex\ itg.hex	陀螺儀感測器燒寫檔案
IAR\hex\ KR.hex	可燃氣體感測器燒寫檔案
IAR\hex\ M2_TC72.hex	數位溫度感測器燒寫檔
IAR\hex\ TGS2602.hex	氣體品質感測器燒寫檔
IAR\hex\ gps.hex	GPS 資料獲取燒寫檔
IAR\hex\ rf_send_recv.hex	感測器無線接受端（主）燒寫檔
IAR\hex\ rf_send_led.hex	Led 測試無線發送端燒寫檔

IAR\hex\ rf_recv_led.hex	Led 測試無線接收端燒寫檔案
IAR\hex\ rf_send.hex	無線測試發送燒寫檔
IAR\hex\ rf_recv.hex	無線測試接收燒寫檔案
IAR\hex\ led_test.hex	測試四個 Led 燈的燒寫檔
IAR\hex\ sw.hex	按鍵測試程式
上層介面\APK 原始程式碼\sensor	Android 下感測器介面 APK 原始程式碼+so
上層介面\APK 原始程式碼（更新）\sensor	Android 下感測器介面 APK 原始程式碼+so（更新）
上層介面\APK 原始程式碼\led	Android 下 Led 介面 APK 原始程式碼
上層介面\APK\Sensor.apk	感測器介面的安裝檔
上層介面\APK\Light_zigbee.apk	Led 介面的安裝檔
上層介面\APK（更新）\Sensor.apk	感測器介面更新後的安裝檔

以下是有關HMI700-6410S平台光碟內容 (只限於ZigBee進階版)

上層目錄	說明
SCH	平台的電路圖，以 PDF 格式的檔案提供。
Datasheet	平台所採用電子元件的 Datasheet 檔。
Android	包含以下子目錄： 1. **Cross compiler**。 2. **U-boot**。 3. **U-Boot_MMC**：支援由 SD 卡啓動的 U-Boot 版本。 4. **Kernel**。 5. **Android**：Android 的程式碼，包含編譯 Android 的腳本檔等。 6. **Sample Image**：在這個目錄內應放置 u-boot.bin、zImage、ramdisk-uboot.img、system.img 及 userdata.img 等映射檔。 7. **NDK**：包含 Android NDK、cygwin 及一個 LED 控制的實驗範例。 8. **USB Driver**

Tools	包含以下項目： 1. 微軟公司的 **ActiveSync 4.5** 及 USB 驅動程式。 2. **ConvertZ**：這是繁簡中文轉換的工具。 3. 串列終端工具 **DNW** 及 USB 驅動程式。 4. **TFTPD32**：TFTP 伺服器的 Windows 版。 5. **IROM_Fusing_Tool**：製作啓動用 SD 卡的工具。

0-3　Android / ZigBee 無線感測器平台實驗簡要說明

1. ZB2530P-AN 平台感測器實驗（基於Android）

實驗前準備：

硬體：

- HMI700-6410S平台一個
- LCD 7吋螢幕一塊
- ZB2530-01模組兩個
- 八個不同的感測器模組
- 模擬器一個
- 串列埠線一條（一頭公頭，一頭母頭）

軟體：

十三個感測器的燒寫檔（一次只能燒寫一個）：

- M1_IMC.hex
- M2_AD590.hex
- M3_BMA.hex
- M2_TGS.hex
- M3_EEPROM.hex

- M5.hex
- M4_GM.hex
- M4_GD.hex
- SHT10.hex
- itg.hex
- KR.hex
- M2_TC72.hex
- TGS2602.hex
- 無線接收端的燒寫文件： rf_send_recv.hex
- Android下感測器介面APK：Sensor.apk（或者更新後的APK）

注意：M5.hex是直流馬達模組燒寫檔，我們程式中只是控制它的轉動，沒有實現無線控制轉速，程式將在以後完善。

實驗步驟：

（1） 利用模擬器將M1_IMC.hex燒寫進其中一個ZB2530-01模組，並把這個ZB2530-01模組當作發送端（從）。

（2） 利用模擬器將rf_send_recv.hex燒寫進另外一個ZB2530-01模組，並把這個ZB2530-01模組當作接收端（主）。

（3） 透過串列埠線將HMI700-6410S平台上的串列埠2與接收端的ZB2530-01模組連接起來

（4） 將Sensor.apk拷貝進SD卡中，然後將SD卡裝入HMI700-6410S平台上的SD卡插槽，最後選擇Sensor.apk進行安裝，安裝好後就可以進行感測器實驗了。

說明：以上是實驗的簡要介紹，詳見第四章。

2. ZB2530P-AN平台 GPS 資料獲取實驗（基於PC）

實驗前準備：

硬體：

- GPS模組一個

- ZB2530-01模組兩個
- 模擬器一個
- 串列埠線一條

軟體：

- GPS資料無線發送燒寫檔：GPS_S.hex
- GPS資料無線接收燒寫檔：GPS_R.hex

實驗步驟：

（1） 利用模擬器將GPS_S.hex燒寫進其中一個ZB2530-01模組，並把這個ZB2530-01模組當作發送端（從）。

（2） 利用模擬器將GPS_R.hex燒寫進另外一個ZB2530-01模組，並把這個ZB2530-01模組當作接收端（主）。

（3） 將GPS模組的串列埠與發送端ZB2530-01模組的串列埠相接，並給GPS模組上電。

（4） 透過串列線將PC上的串列埠與接收端的ZB2530-01模組連接起來。

（5） 打開串列埠終端，即可看見GPS資料。

3. ZB2530P-AN平台 LED DEMO 實驗（基於Android）

實驗前準備：

硬體：

- HMI700-6410S平台一個
- LCD 7吋螢幕一塊
- ZB2530-01模組兩個
- 模擬器一個
- 串列線一條（一頭公頭，一頭母頭）

軟體：

- Led無線發送燒寫文件：rf_send_led.hex

- Led無線接收燒寫文件：rf_recv_led.hex
- Android下LED介面APK：Light_zigbee.apk

實驗步驟：

（6） 利用模擬器將rf_send_led.hex燒寫進其中一個ZB2530-01模組，並把這個ZB2530-01模組當作發送端（主）。

（7） 利用模擬器將rf_recv_led.hex燒寫進另外一個ZB2530-01模組，並把這個ZB2530-01模組當作接收端（從）。

（8） 透過串列埠線將HMI700-6410S平台上的串列埠2與發送端的ZB2530-01模組連接起來

（9） 將Light_zigbee.apk拷貝進SD卡中，然後將SD卡裝入HMI700-6410S平台上的SD卡插槽，最後選擇Light_zigbee.apk進行安裝，安裝好後就可以進行Led測試實驗了。

說明：以上是實驗的簡要介紹，詳見第四章。

4. ZB2530P-AN平台無線資料測試實驗（基於Android）

實驗前準備：

硬體：

- HMI700-6410S平台一個
- LCD 7吋螢幕一塊
- ZB2530-01模組兩個
- 模擬器一個
- 串列線一條（一頭公頭，一頭母頭）

軟體：

- 無線測試發送燒寫檔：rf_send.hex
- 無線測試接收燒寫文件：rf_recv.hex
- Android下串列埠除錯工具 （PC下串列埠除錯工具）

實驗步驟：

（1） 利用模擬器將rf_send.hex燒寫進其中一個ZB2530-01模組，並把這個ZB2530-01模組當作發送端（從）。

（2） 利用模擬器將rf_recv.hex燒寫進另外一個ZB2530-01模組，並把這個ZB2530-01模組當作接收端（主）。

（3） 透過串列埠線將HMI700-6410S平台上的串列埠2（或者是PC的串列埠）與接收端的ZB2530-01模組連接起來

（4） 打開Android下的串列埠除錯工具（或者PC上的串列埠除錯工具），看列印結果。

說明：以上是實驗的簡要介紹，詳見第二章。

5. ZB2530P-AN平台上四個LED燈的測試實驗

實驗前準備：

硬體：

- ZB2530-01模組一個
- 模擬器一個

軟體：

- 測試四個LED燈的燒寫檔：led_test.hex

實驗步驟：

利用模擬器將led_test.hex燒寫進ZB2530-01模組即可

實驗現象： 模組上的四個LED燈將同時閃爍

6. ZB2530P-AN平台上的按鍵測試實驗

實驗前準備：

硬體：

- ZB2530-01模組兩個
- 模擬器一個
- 串列埠線一條（一頭公頭，一頭母頭）

軟體：

- 測試按鍵的燒寫檔（無線發送端）：sw.hex
- 無線接收端的燒寫文件： rf_send_recv.hex

實驗步驟：

（1） 利用模擬器將sw.hex燒寫進ZB2530-01模組(發送端)。

（2） 利用模擬器將rf_send_recv.hex燒寫進ZB2530-01模組(接受端)。

（3） 利用串列線將接收端的串列埠與 PC 端串列埠相連後，打開串列埠終端。

實驗現象： 按一下按鍵，ZigBee模組上的紅燈和綠燈同時閃爍一次，同時串列埠終端上顯示01；按兩下按鍵，ZigBee模組上的紅燈和綠燈同時閃爍兩次，同時串列埠終端上顯示02；以此類推，當按到10次的時候，ZigBee模組上的紅燈和綠燈同時閃爍一次，同時串列埠終端上顯示01，10次一個迴圈。

第一章　ZigBee 概述

■ 什麼是 ZigBee

ZigBee是IEEE 802.15.4 協議的代名詞。根據這個協定規定的技術是一種短距離、低功耗的無線通信技術。ZigBee 的命名，源自於蜜蜂的八字舞，由於蜜蜂(bee)是靠飛翔和"嗡嗡" (zig) 地抖動翅膀的"舞蹈"來與同伴傳遞花粉所在方位資訊，也就是說蜜蜂依靠這樣的方式構成了群體中的通信網路。其特點是短距離、低複雜度、低功耗、低資料速率、低成本。主要應用的方向在於家庭裝置自動化、環境安全與控制、個人醫療照護、自動控制和遠端控制領域、嵌入式各種設備。

簡而言之，ZigBee 就是一種便宜，低功耗的近距離無線組網通訊技術。

■ ZigBee 的起源

ZigBee 在中國被譯爲"紫蜂"，它與藍芽相類似.是一種新興的短距離無線技術，用於感測控制應用 (sensor and control)。此想法在 IEEE 802.15 工作組中提出，於是成立了 TG4 工作組，並制定規範 IEEE 802.15.4。

2002年，ZigBee Alliance 成立。

2004年，ZigBee V1.0 誕生，它是 ZigBee 第一個規範，但由於推出倉促，存在一些錯誤。

2006年，推出 ZigBee 2006，比較完善。

2007 年底，ZigBee PRO 推出。

ZigBee 的底層技術基於IEEE 802.15.4.

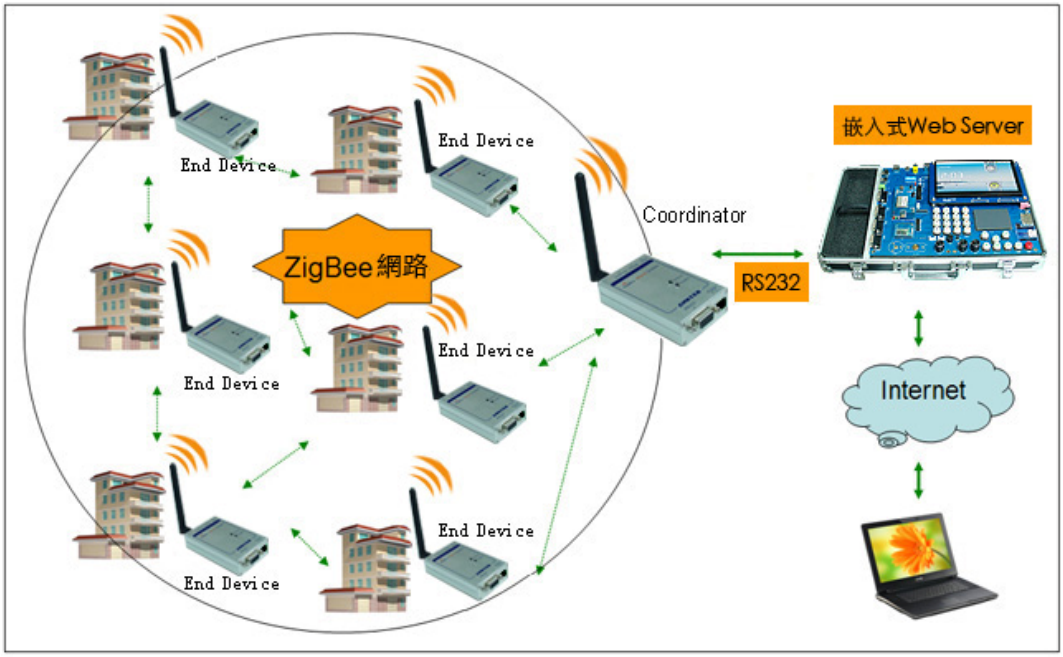
物理層和 MAC 層直接引用了IEEE 802.15.4

在藍芽技術的使用過程中，人們發現藍芽技術儘管有許多優點，但仍存在許多缺陷。對工業，家庭自動化控制和工業遙測遙控領域而言，藍芽技術顯得太複雜、功耗大、距離近、組網規模太小等，而工業自動化，對無線資料通信的需求越來越強烈，而且，對於工業現場，這種無線資料傳輸必須是高可靠性的，並能抵抗工業現場的各種電磁干擾。因此，經過人們長期努力，ZigBee 協議在 2003 年正式問世。 另外，ZigBee 使用了在它之前所研究過的面向家庭網路的通信協定 Home RF Lite。

長期以來，低價、低傳輸率、短距離、低功率的無線通訊市場一直存在著。自從Bluetooth 出現以後，曾讓工業控制、家用自動控制、玩具製造商等業者雀躍不已，但是 Bluetooth 的售價一直居高不下，嚴重影響了這些廠商的使用意願。如今，這些業者都參加了IEEE802.15.4 小組，負責制定 ZigBee 的物理層和媒體介入控制層。IEEE802.15.4 規範是一種經濟、高效、低資料速率（<250kbps）、工作在 2.4 GHz 和 868/928MHz 的無線技術，用於個人區域網和對等網路。它是 ZigBee 應用屋和網路層協定的基礎。ZigBee是一種新興的近距離、低複雜度、低功耗、低資料速率、低成本的無線網路技術，它是一種介於無線標記技術和藍芽之間的技術提案。主要用於近距離無線連接。它依據 802.15.4 標準，在數千個微小的感測器之

間相互協調實現通信。這些感測器只需要很少的能量，以接力的方式透過無線電波將資料從一個感測器傳到另一個感測器，所以它們的通信效率非常高。

■ ZigBee 無線資料傳輸網路描述



圖：ZigBee 無線抄表系統

簡單的說，ZigBee 是一種高可靠的無線數傳網路，類似於 CDMA 和 GSM 網路。ZigBee 數傳模組類似於移動網路基站。通訊距離從標準的 75m 到幾百米、幾公里，並且支持無限擴展。

ZigBee 是一個由可多到 65000 個無線數傳模組組成的一個無線數傳網路平台，在整個網路範圍內，每一個 ZigBee 網路數傳模組之間可以相互通信，每個網路節點間的距離可以從標準的 75m 無限擴展。

與移動通信的 CDMA 或 GSM 不同的是，ZigBee 網路主要是爲工業現場自動化控制資料傳輸而建立。因此，它必須具有簡單，使用方便，可靠高，價格低的特點。而移動通信網主要是爲語音通信而建立，每個基地台價值一般都在百萬元以上，而每個 ZigBee 設備卻不到 4000 元。每個 ZigBee 網路節點不僅本身可以作爲監控

物件，例如其所連接的感測器直接進行資料獲取和監控，還可以自動中轉別的網路節點傳過來的資料資料。除此之外，每一個 ZigBee 網路節點(FFD)還可在自己信號覆蓋的範圍內，和多個不承擔網路資訊中轉任務的孤立的子節點(RFD)無線連接。

■ ZigBee 採用的自組織網通信方式

➢ ZigBee 技術所採用的自組織網是怎麼回事？

舉一個簡單的例子就可以說明這個問題，當一隊傘兵空降後，每人持有一個 ZigBee 網路模組終端，降落到地面後，只要他們彼此間在網路模組的通信範圍內，透過彼此自動尋找，很快就可以形成一個互聯互通的 ZigBee 網路。而且，由於人員的移動，彼此間的聯絡還會發生變化。因而，模組還可以透過重新尋找通信物件，確定彼此間的聯絡，對原有網路進行更新。這就是自組織網。

➢ ZigBee 技術爲什麼要使用自組織網來通信？

網狀網通信實際上就是多通道通信，在實際工業現場，由於各種原因，往往並不能保證每一個無線通道都能夠始終暢通，就像城市的街道一樣，可能因爲車禍，道路維修等，使得某條道路的交通出現暫時中斷，此時由於我們有多個通道，車輛（相當於我們的控制資料）仍然可以透過其他道路到達目的地。而這一點對工業現場控制而言則非常重要。

➢ 爲什麼自組織網要採用動態路由的方式？

所謂動態路由是指網路中資料傳輸的路徑並不是預先設定的，而是傳輸資料前，透過對網路當時可利用的所有路徑進行搜索，分析它們的位置關係以及遠近，然後選擇其中的一條路徑進行資料傳輸。在我們的網路管理軟體中，路徑的選擇使用的是"梯度法"，即先選擇路徑最近的一條通道進行傳輸，如傳不通，再使用另外一條稍遠一點的通路進行傳輸，以此類推，直到資料送達目的地爲止。在實際工業現場，預先確定的傳輸路徑隨時都可能發生變化，或者因各種原因路徑被中斷了，或者過於繁忙不能進行及時傳送。動態路由結合網狀拓撲結構，就可以很好解決這個問題，從而保證資料的可靠傳輸。

■ ZigBee 自身的技術優勢

① 低功耗。在低耗電待機模式下，2節 5號乾電池可支持 1 個節點工作 6～24 個月，甚至更長。這是 ZigBee 的突出優勢。相比較，藍芽能工作數周、WiFi 可工作數小時。

現在，TI 公司和德國的 MicropeIt 公司共同推出新能源的 ZigBee 節點。該節點採用 MicropeIt 公司的熱電發電機給 TI 公司的 ZigBee 提供電源。

② 低成本。透過大幅簡化協議(不到藍芽的1/10)，降低了對通信控制器的要求，按預測分析，以8051 的 8 位微控制器測算，全功能的主節點需要 32KB 程式碼，子功能節點少至 4KB 程式碼，而且 ZigBee 免協議專利費。每塊晶片的價格大約爲 2美元。

③ 低速率。ZigBee 工作在20～250 kbps的較低速率，分別提供250 kbps (2.4GHz)、40kbps (915 MHz)和 20kbps(868 MHz) 的原始資料吞吐率，滿足低速率傳輸資料的應用需求。

④ 近距離。傳輸範圍一般介於10～100m之間，在增加RF發射功率後，亦可增加到1～3 km。這指的是相鄰節點間的距離。如果透過路由和節點間通信的接力，傳輸距離將可以更遠。

⑤ 短時延。ZigBee的回應速度較快，一般從睡眠轉入工作狀態只需15ms，節點連接進入網路只需30 ms，進一步節省了電能。相比較，藍芽需要3～10s、WiFi 需要3 s。

⑥ 高容量。ZigBee可採用星狀、片狀和網狀網路結構，由一個主節點管理若干子節點，最多一個主節點可管理 254個子節點；同時主節點還可由上一層網路節點管理，最多可組成65000個節點的大網。

⑦ 高安全。ZigBee 提供了三級安全模式,包括無安全設定、使用接入控制清單(ACL)防止非法獲取資料以及採用高級加密標準(AES 128) 的對稱密碼，以靈活確定其安全屬性。

⑧ 免執照頻段。採用直接序列擴頻在工業科學醫療(ISM)頻段，2.4 GHz (全球)、915MHz(美國)和 868 MHz (歐洲)。

■ ZigBee 的頻率

1) 868MHZ 傳輸速率爲 20KB/S 適用於歐洲

2) 915MHZ 傳輸速率爲 40KB/S 適用於美國

3) 2.4GHZ 傳輸速率爲 250KB/S 全球通用

由於此三個頻帶物理層並不相同，其各自通道帶寬也不同，分別爲 0.6MHZ，2MHz 和 5MHz；分別有 1 個 10 個和 16 個通道。

不同頻帶的擴頻和調製方式有區別，雖然都使用了直接擴頻(DSSS)的方式，但從比特到碼片的變換方式有較大的差別。

調製方式都用了調相技術，但 868MHZ 和 915MHZ 頻段採用的是 BPSK，而 2.4GHZ 頻段採用的是 OQPSK。

在發射功率爲 0dBm 的情況下，BLUETOOTH 通常能用 10M 的作用範圍。

而基於IEEE 802.15.4 的ZigBee 在室內通常能達到30-50米作用距離，在室外如果障礙物少，甚至可以達到 100 米作用距離。

所以 ZigBee 可歸爲低速率的短距離無線通信技術。

■ ZigBee 性能分析

1、**資料速率比較低** 在2.4GHZ 的頻段只有250Kb/S，而且只是鏈路上的速率，除掉通道競爭應答和重傳等消耗，真正能被應用所利用的速率可能不足 100Kb/S，並且餘下的速率可能要被鄰近多個節點和同一個節點的多個應用所瓜分。因此不適合做視頻之類事情。

適合的應用領域 — 傳感和控制

2、**可靠性** 在可靠性方面，ZigBee 有很多方面進行保證。物理層採用了擴頻技術，能夠在一定程度上抵抗干擾。

MAC應用層(APS 部分)有應答重傳功能。

MAC層的CSMA機制使節點發送前先監聽信道，可以起到避開干擾的作用。

當 ZigBee 網路受到外界干擾，無法正常工作時，整個網路可以動態的切換到另一個工作通道上。

3、**時延** 由於 ZigBee 採用隨機接入 MAC 層，且不支援時分複用的通道接入方式，因此不能很好的支持一些即時的業務。

4、**能耗特性**　能耗特性是 ZigBee 的一個技術優勢。

通常ZigBee節點所承載的應用資料速率都比較低，在不需要通信時，節點可以進入很低功耗的休眠狀態，此時能耗可能只有正常工作狀態下的千分之一。由於一般情況下，休眠時間占總執行時間的大部分，有時正常工作的時間還不到百分之一，因此達到很高的節能效果。

5、**組網和路由性** — 網路層特性

ZigBee 大規模的組網能力— 每個網路 65000 個節點

Bluetooth — 每個網路 8 個節點

因爲ZigBee 底層採用了直擴技術，如果採用非信標模式，網路可以擴展得很大，因爲不需同步而且節點加入網路和重新加入網路的過程很快，一般可以做到 1 秒以內，甚至更快。

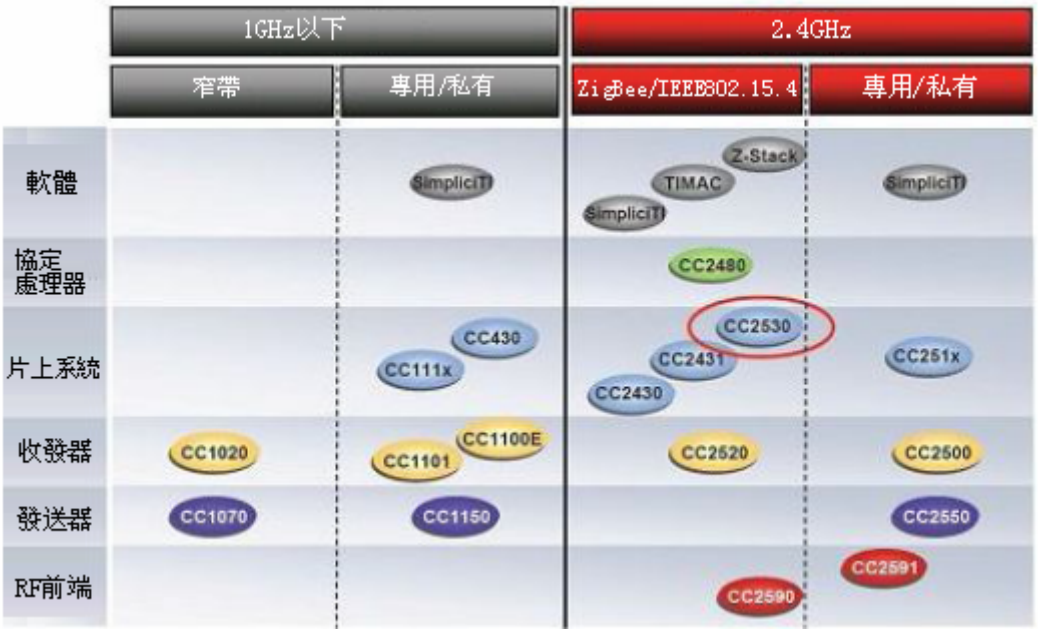
Bluetooth 通常需要 3 秒。

在路由方面，ZigBee支持可靠性很高的網狀網的路由，所以可以佈置範圍很廣的網路，並支援多播和廣播特性，能夠給豐富的應用帶來有力的支持。

■ ZigBee 的應用前景

ZigBee並不是用來與藍芽或者其他已經存在的標準競爭，它的目標定位于現存的系統還不能滿足其需求的特定的市場，它有著廣闊的應用前景。ZigBee 聯盟預言在未來的四到五年，每個家庭將擁有 50 個 ZigBee 器件，最後將達到每個家庭150個。據估計，到2007年，ZigBee 市場價值將達到數億美元。其應用領域主要包括：

- ◆ 家庭和樓宇網路：空調系統的溫度控制、照明的自動控制、窗簾的自動控制、煤氣計量控制、家用電器的遠程控制等；
- ◆ 工業控制：各種監控器、感測器的自動化控制；
- ◆ 商業：智慧型標籤等；
- ◆ 公共場所：煙霧探測器等；
- ◆ 農業控制：收集各種土壤資訊和氣候資訊；
- ◆ 醫療：老人與行動不便者的緊急呼叫器和醫療感測器等。



圖：TI RF 產品 Road Map

■ ZigBee 聯盟

ZigBee 聯盟是一個高速成長的非盈利業界組織，成員包括國際著名半導體生產商、技術提供者、技術集成商以及最終使用者。聯盟制定了基於 IEEE802.15.4，具有高可靠、高性價比、低功耗的網路應用規格。

ZigBee 聯盟的主要目標是以透過加入無線網路功能，爲消費者提供更富有彈性、更容易使用的電子產品。ZigBee 技術能融入各類電子產品，應用範圍橫跨全球的民用、商用、公共事業以及工業等市場。使得聯盟會員可以利用ZigBee 這個標準化無線網路平台，設計出簡單、可靠、便宜又節省電力的各種產品來。

ZigBee 聯盟所鎖定的焦點爲制定網路、安全和應用軟體層；提供不同產品的協調性及互通性測試規格；在世界各地推廣ZigBee 品牌並爭取市場的關注；管理技術的發展。

ZigBee 聯盟對 ZigBee 標準的制定：IEEE802.15.4 的物理層、MAC層及資料連結層，標準已在 2003 年 5 月發佈。ZigBee 網路層、加密層及應用描述層的制定也取得了較大的進展。V1.0 版本已經發佈。其他應用領域及其相關的設備描述也會陸續發佈。由於 ZigBee 不僅只是 802.15.4 的代名詞，而且 IEEE 僅處理低

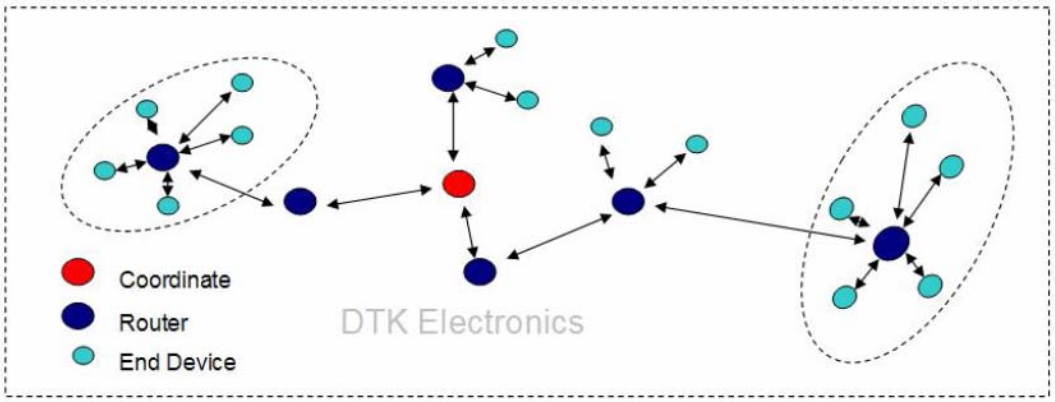
級MAC 層和物理層協定，因此 ZigBee 聯盟對其網路層協定和 API 進行了標準化。完全協定用於一次可直接連接到一個設備的基本節點的 4K 位元組或者作爲 Hub 或路由器的協調器的 32K 位元組。每個協調器可連接多達 255 個節點，而幾個協調器則可形成一個網路，對路由傳輸的數目則沒有限制。

ZigBee 聯盟還開發了安全層，以保證這種便攜設備不會意外洩漏其標識，而且這種利用網路的遠距離傳輸不會被其他節點獲得。

■ ZigBee 的網路拓撲

在 ZigBee 網路中，定義了三種網路角色，分別是 Coordinator（網路協調器節點），Router（網路路由器節點），End Device（網路終端節點）。

- Coordinator（網路協調器節點）：負責網路的建立（WPAN Formation）及網路位址（Short Address）的分配。
- Router（網路路由器節點）：負責找尋、建立及修復資料包路由路徑（Routing Path），並負責轉送資料包，同時也可配置網路位址（Short Address）給子節點（Child）。
- End Device（網路終端節點）：只能選擇加入已經形成的網路，可傳送資料。



第二章
Android / ZigBee
無線感測器平台介紹

2-1　基本介紹

Android / ZigBee 無線感測器平台（以下簡稱ZB2530P-AN 模組）硬體參考資訊下列的工具，能夠在這個板上使用。且這些工具的用戶手冊可以從TI 網站下載：

1、SmartRF® Studio

(http://focus.ti.com/docs/toolsw/folders/print/smartrftm-studio.html)

2、SmartRF Flash Programmer

(http://focus.ti.com/docs/toolsw/folders/print/flash-programmer.html)

3、Texas Instruments Packet Sniffer

(http://focus.ti.com/docs/toolsw/folders/print/packet-sniffer.html?DCMP=MSP430&HQS=Other+OT+smartrfsniffer)

4、IAR-EW8051-7.51A and C-Spy debugger(www.IAR.se)

ZB2530P-AN 模組包含若干功能和應用程式，它允許快速測試RF 介面和CC2530 的外部設備。它能作爲以下例子使用：

1、ZigBee Debugger模擬器。

2、模組功能：ZB2530P-AN模組，包含1個Debugger介面用來線上模擬及燒錄。

2-2 Android／ZigBee無線感測器平台產品內容

ZB2530P-AN平台產品包含下列兩種套件（可選）：

2-2.1 基本型配件

1、3 片ZB2530-01 模組

2、8 個感測器模組

3、3 條USB電源連接線（搭配 ZB2530-01 模組）

4、3 條USB資料傳輸線（搭配 ZB2530-01 模組）

5、用戶手冊 （本手冊）

6、資料光碟（內含Android 測試及圖控軟體）

7、資料傳輸線（配合相應的平台，選擇其一）：

a) DB9公 To DB9 母資料傳輸線：用於DMA-6410XP平台

b) DB9公 To 3Pin 資料傳輸線：用於DMA-6410L平台

c) DB9公 To Mini 5Pin 資料傳輸線：用於HMI700-6410S人機介面平台

8、1 個 ZigBee Debugger 模擬器

9、5V 2A 電源 (選購品)

2-2.2 進階型配件

1、9 片ZB2530-01 模組

2、8 個感測器模組

3、4 個進階型感測器模組

4、9 條USB電源連接線（搭配 ZB2530-01 模組）

5、9 條USB資料傳輸線（搭配 ZB2530-01 模組）

6、用戶手冊（本手冊）

7、資料光碟（內含Android 測式及圖控軟體）

8、資料傳輸線（配合相應的平台，選擇其一）：

a) DB9公 To DB9 母資料傳輸線：用於 DMA-6410XP 平台

b) DB9公 To 3Pin 資料傳輸線：用於 DMA-6410L 平台

c) DB9公 To Mini 5Pin 資料傳輸線：用於HMI700-6410S 人機介面平台

9、1 個 ZigBee Debugger 模擬器

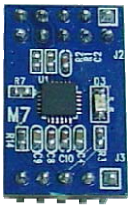
10、HMI700-6410S開發平台

11、5V 2A電源 (選購品)

2-2.3　基本型感測器模組

1、光敏電阻感測器 	2、 光電晶體感測器 
3、 酒精感測器 	4、人體感測器 
5、類比溫度感測器 	6、EEPROM 模組 
7、直流馬達模組 	8、GSensor 三軸位移加速度感測器 

2-2.4 進階型感測器模組

1、陀螺儀感測器 	2、可燃性氣體感測器 
3、SHT10溫濕度感測器 	4、溫濕度感測器 

2-2.5 進階型擴充感測器模組

1、13.56M NFC RFID 	2、空氣品質感測器 
3、GPS衛星接收模組 	4、RFID射頻IC卡讀寫器 

2-3 功能特色

ZB2530P-AN模組是採用 TI 最新一代CC2530 ZigBee 標準晶片，適用於2.4GHz、IEEE 802.15.4、ZigBee 和 RF4CE 應用。CC2530 晶片包括了極好性能的一流 RF 收發器，工業標準增強性8051 MCU，系統中可編程的快閃記憶體，8KB RAM 以及許多其他功能強大的特性，可廣泛應用在2.4-GHz IEEE 802.15.4 系統、RF4CE 控制系統、ZigBee 系統，其應用領域可爲：家庭/醫院/建築物自動化、工業控制測量和監視、低功耗無線感測器網路等各方面應用。

2-4 規格特性

- 主晶片：CC2530F256，256K Flash，TI 公司最新一代 ZigBee SOC 晶片
- 輸入電壓：DC 5-12V
- 溫度範圍：-30℃ --85℃
- 串列速率：38400bps（預設），可設置 9600bps，19200bps， 38400bps，115200bps
- 無線頻率：2.4GHz
- 無線協議：ZigBee2007 /PRO
- 傳輸距離：可視距離 10 米
- 發射電流：34mA（最大）
- 接收電流：25mA（最大）
- 低功耗模式：可設定進入低功耗模式（與應用模式相關），2 節 AA 電池可使用半年
- 接收靈敏度：-96DBm
- 搭配平台： Android2.1 / Linux2.6.X / Windows XP / Windows 7 / CE6.0 (選配)
- 尺寸大小：長 74mm × 寬 42mm

2-5 應用領域

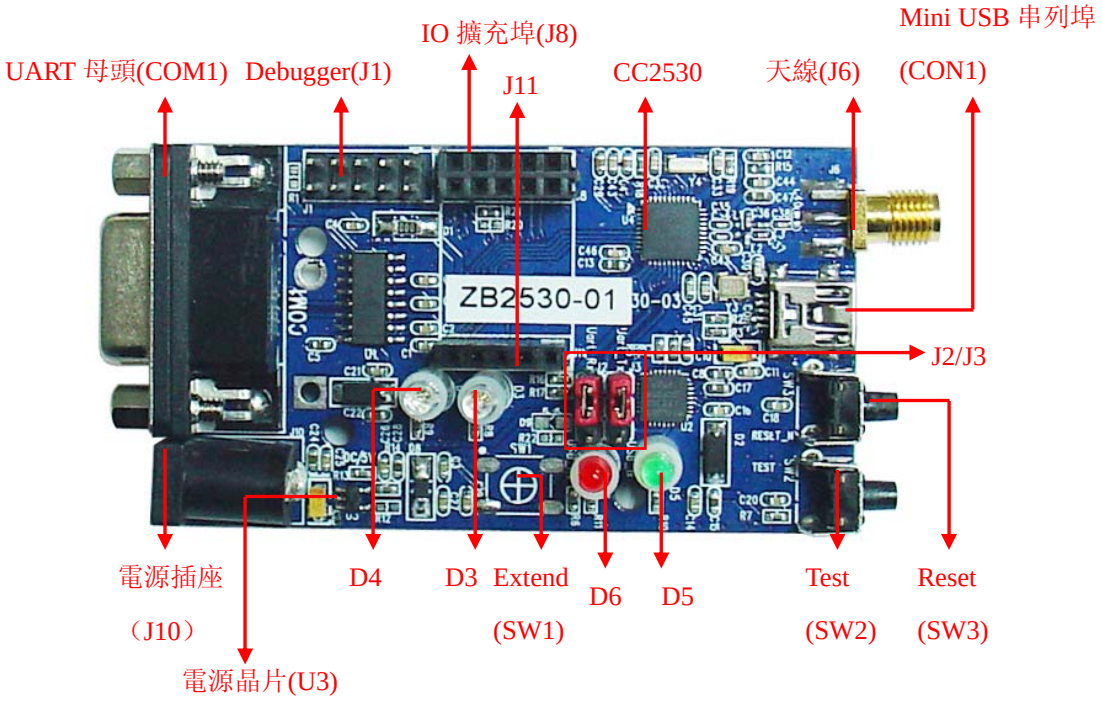
- 2.4GHz 資料傳輸應用
- RF4CE 無線遙控器
- ZigBee 無線感測網路
- 工業測量與監控
- 消費電子領域應用

2-6 準備工作

ZB2530-01 模組在連接電腦之前要先安裝SmartRF® Studio，因爲它需要在電腦上安裝硬體驅動。SmartRF® Studio可以從TI的網站下載：

http://focus.ti.com/docs/toolsw/folders/print/smartrftm-studio.html

2-7　硬體描述



ZB2530-01 模組正面圖

ZB2530-01 模組背面圖

1) ZB2530-01 供電電源

ZB2530-01 模組，其供電方法有三種：

1. 外電插入：這個插座在板的上面，如上圖所示的 J10，輸入電壓 5V。經過板上的 U3 轉3.3V 後供電主板使用，這時的串列埠轉 USB 電路 CC2530 將同時工作。
2. 第二，Mini USB 供電：當在CON1插入 Mini USB 線時經過 CC2530 將串列埠轉換成 USB 並提供一個 3.3V/100Ma 電源給系統供電，這時 RS232 電路將同時工作。
3. 第三，電池供電：當裝上電池座 J9 時，兩節7 號 AA 電池串聯成 3V 給系統供電，但這時的串列埠轉 USB 電路 CC2530 將不工作。

注意：三種電源可以同時存在，互不干擾。

2) 用戶介面

ZB2530-01 模組，包含三個按鍵作爲用戶的輸入設備，並且有4 個 LED作爲用戶輸出設備及一個天線收發信號介面。

兩按鍵分別爲： SW1--爲外部擴展中斷輸入功能
SW2---爲TEST功能
SW3---爲RESET功能

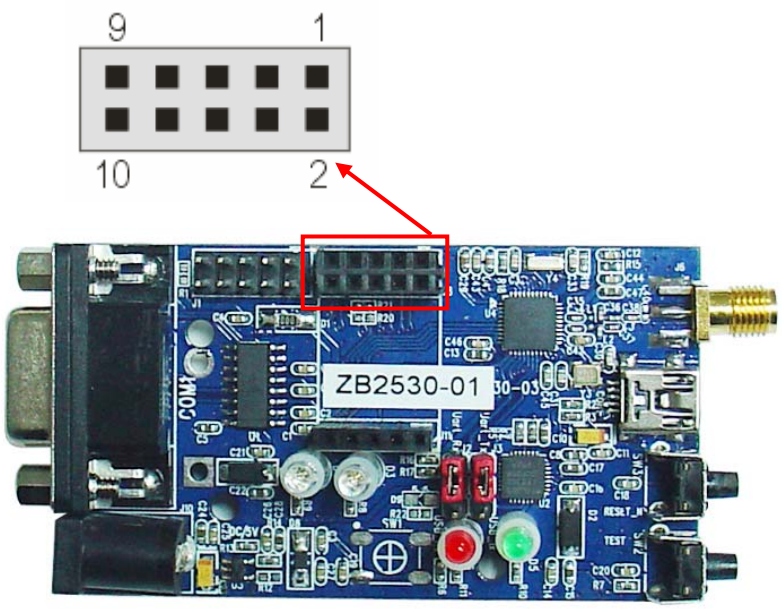
兩LED燈分別爲：D3---爲 ACTIVE 電源指示
D4---爲 TX/RX 動作指示
D5，D6---爲外部擴展輸出指示功能

天線介面爲： J6---爲天線信號介面

3) I/O 連接器

I/O 連接器 J8 拉出 CC2530 信號。透過 I/O 連接器可以方便介接 CC2530I/O 腳。讓外部電路可以連接到 I/O 連接器。

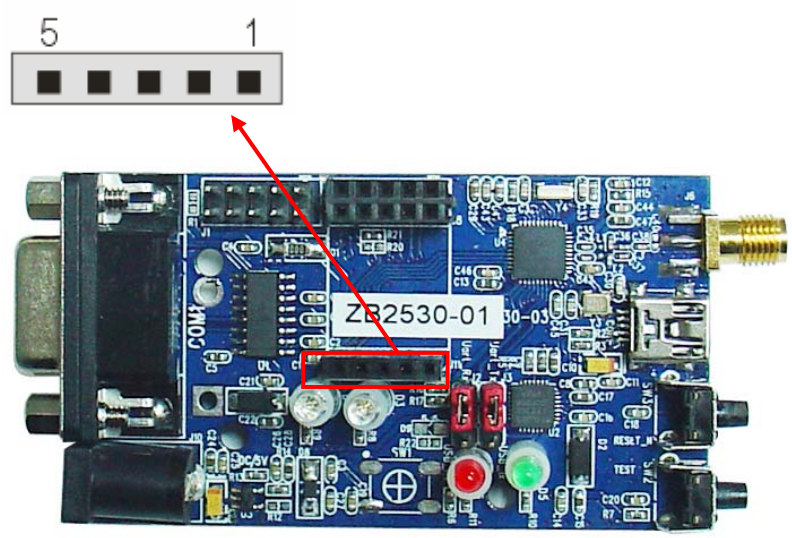
J8 連接器的接腳圖：



接腳	功能
1	P1.7
2	P2.0
3	SDI
4.	SCL
5	SDO
6	SSS
7	I2C_SCL
8	INT
9	I2C_SDA
10	ADC0

I/O 連接器 J8 接腳

J11 連接器的接腳圖：



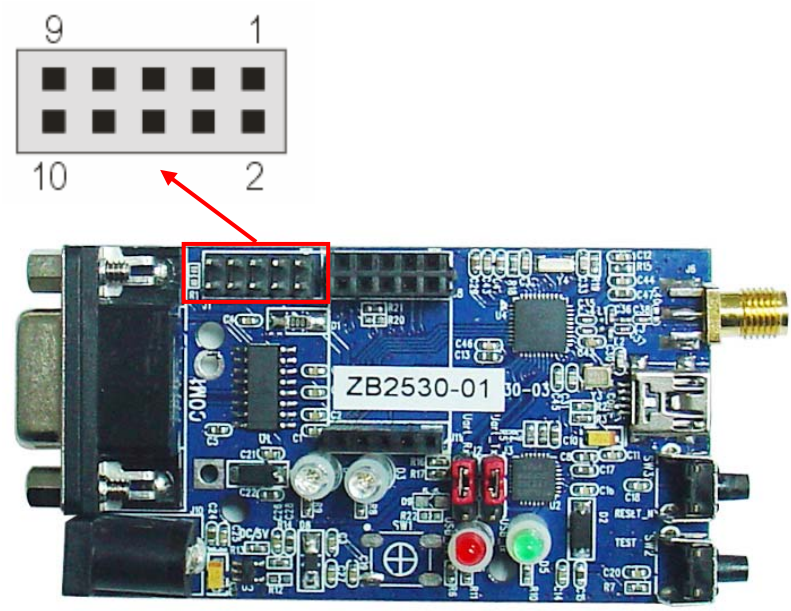
接腳	功能
1	VDD_5V
2	GND
3	VDD_3.3V
4	GND
5	GND

I/O 連接器 J11 接腳

4) ZigBee Debugger 連接器

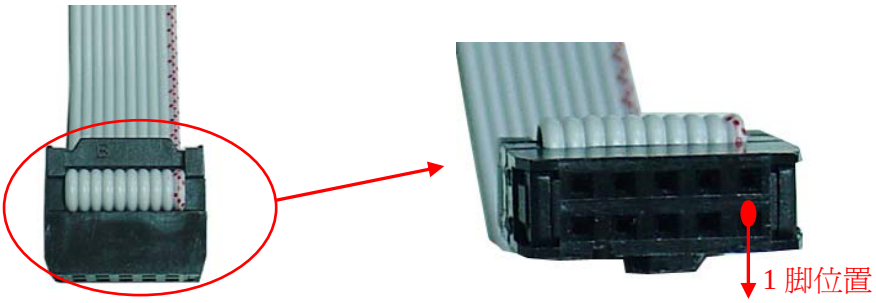
爲了方便用戶下載及除錯 ZB2530-01 ZigBee 模組，可透過模組上的 J1 連接器來下載及除錯，連接時請注意插線的腳位順序如圖。

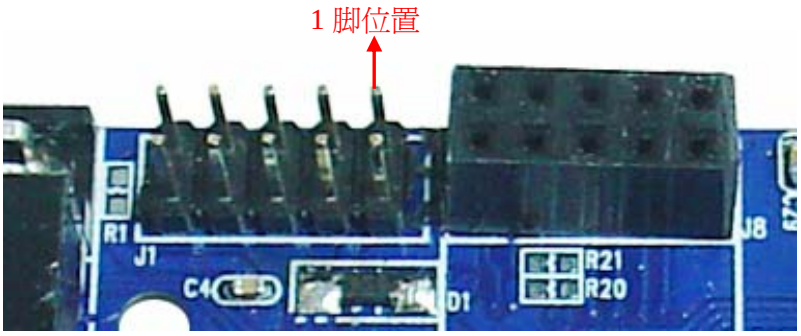
連接器J1的接腳圖：

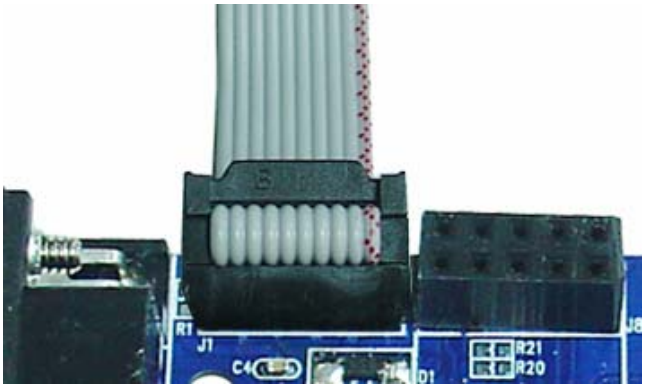


接腳定義：

接腳	功能
1	ND
2	VCC
3	DC
4.	DD
5	NC
6	NC
7	RESET_N
8	NC
9	VCC（NC）
10	NC


1 脚位置

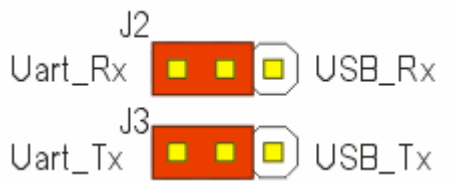

1 脚位置
R1
J1
C4
D1
R21
R20
J8


R1
J1
C4
D1
R21
R20

接好後如下圖所示：

5) UART介面

第一，ZB2530-01 模組的UART 介面可以透過 DP9 母頭COM1訪問，其信號電平爲 3.3V。板內帶 RS232 驅動電路將TTL 電平轉成 RS232 電平，將跳線帽 J2、J3 插上（如下圖所示），然後連接至PC 的。



ZB2530-01 DTYPE 串列跳線

DB9定義爲：

2 = TX

3 = RX

5 = GND

與其他設備連接時應該是 TX 接RX，RX 接TX，GND 接GND電腦的COM1 爲標準的針式 DB9，可透過標準線連接（或者直接連接）。

串列埠的配置應該是：

數據位元 = 8

停止位 = 1

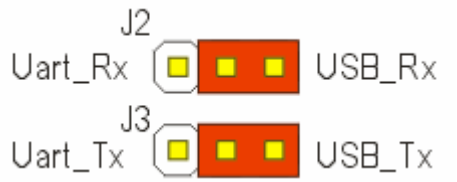
效驗位 = None

流控制 = None

具體定義可參考原理圖

第二，ZB2530-01 模組的UART 介面可以透過 Mini USB 頭 CON1 訪問，其信號電平爲 3.3V。板內透過 ZB2530 串列埠轉USB 驅動電路，將跳線帽J2、J3 插上（如下圖所示），然後連接至PC 的USB埠。

注意：兩種連接方式不可能同時存在，只能任選一種連接！



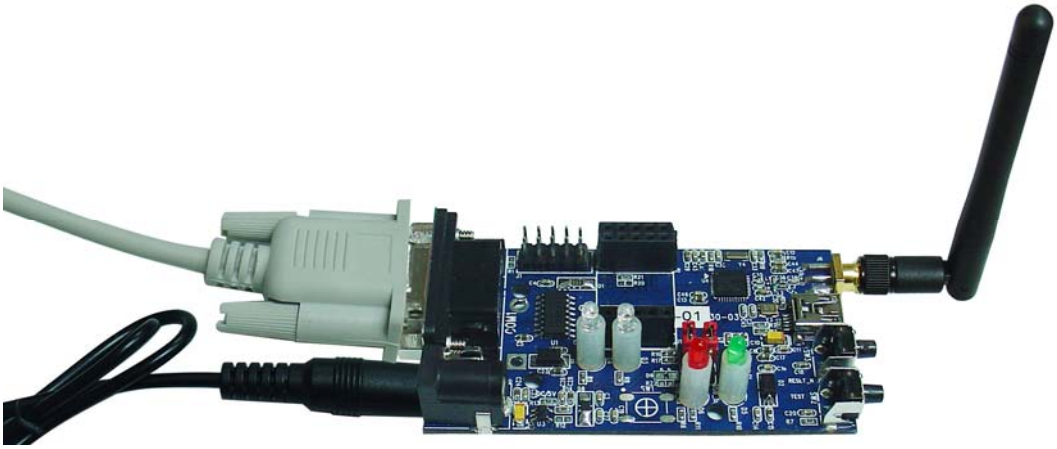
ZB2530-01模組USB 串列跳線

2-8　各平台連接方式

1.　ZB2530-01 模組接收端（主）與 PC 端的連接

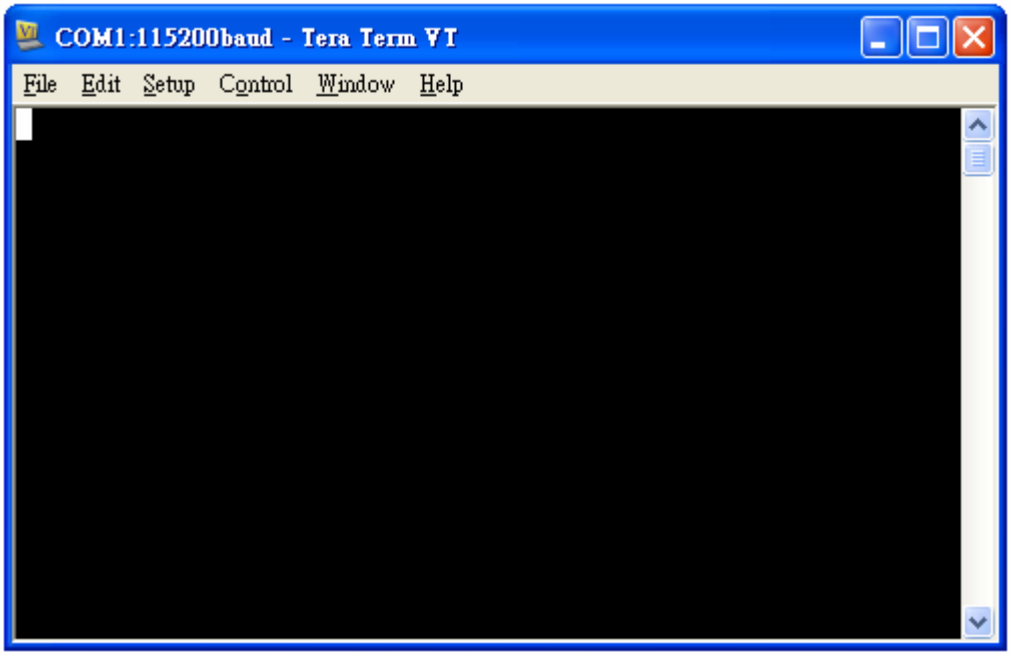
先將DB9 公 TO DB9 母線連接於 PC及 ZB2530-01 模組接收端裝置

並將ZB2530-01 模組接收端接上電源

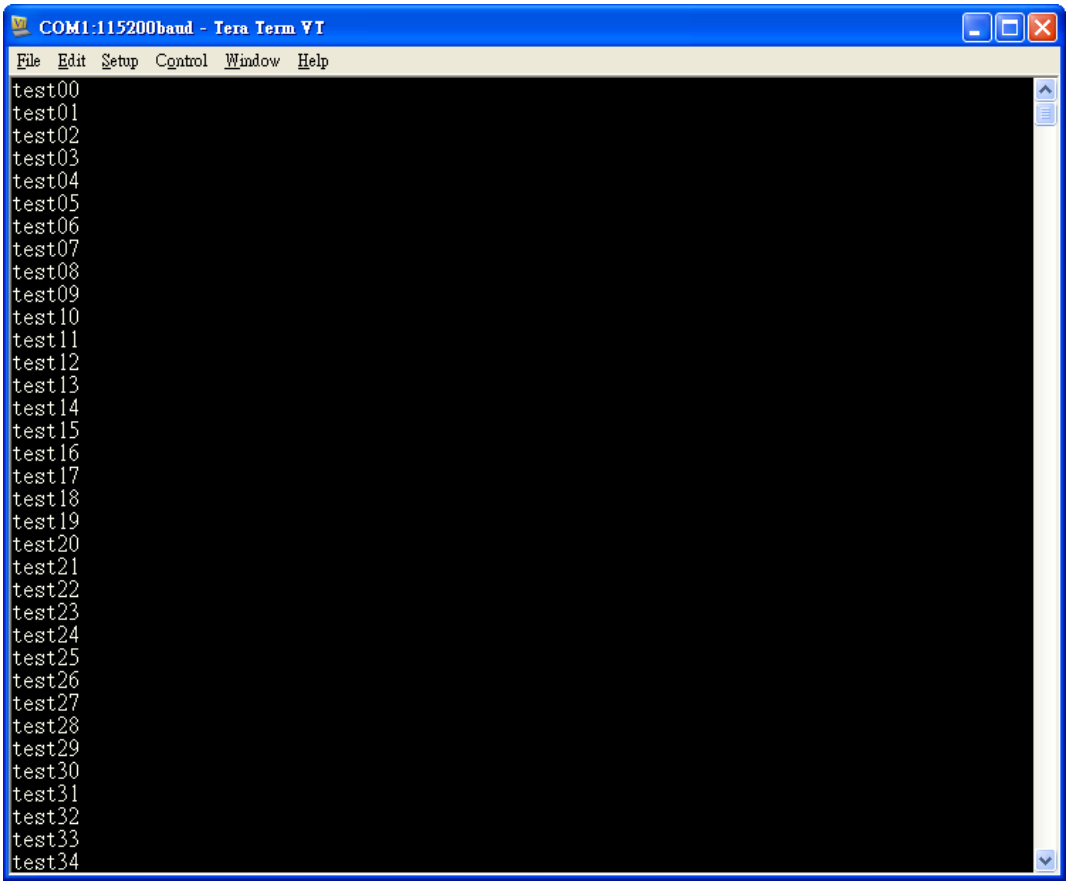


在PC端開啓終端機軟體並開啓PC端相對應COM Port

鮑率(Baud Rate) 設定爲 115200



此時在將 ZB2530-01 發射端（從）接上電源



2.　ZB2530-01 模組接收端（主）與 DMA-6410L 開發平台連接：

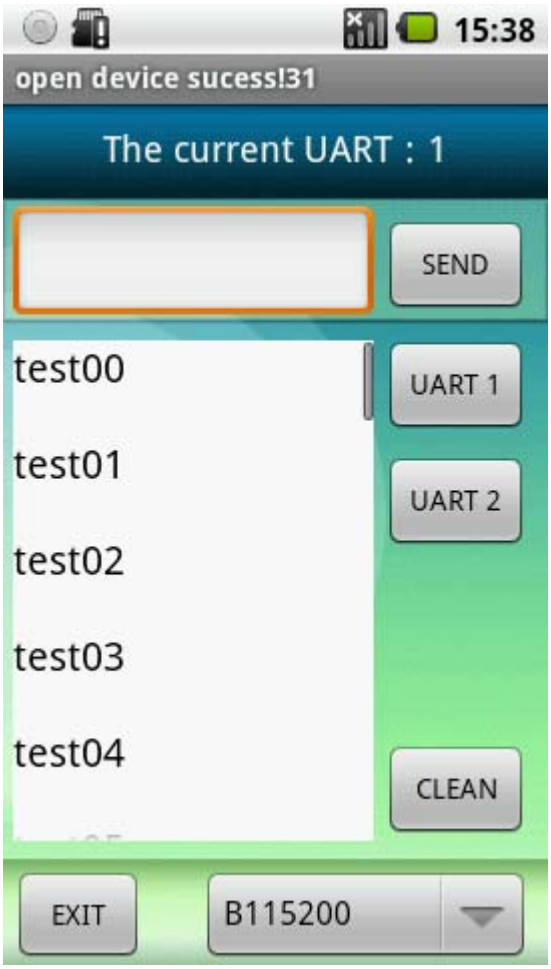
ZB2530-01 模組接收端（主）與 DMA-6410L 開發平台連接圖

接著在 6410L 上執行 DMATEK_TEST 程式

接著選取程式內的 UART 程式

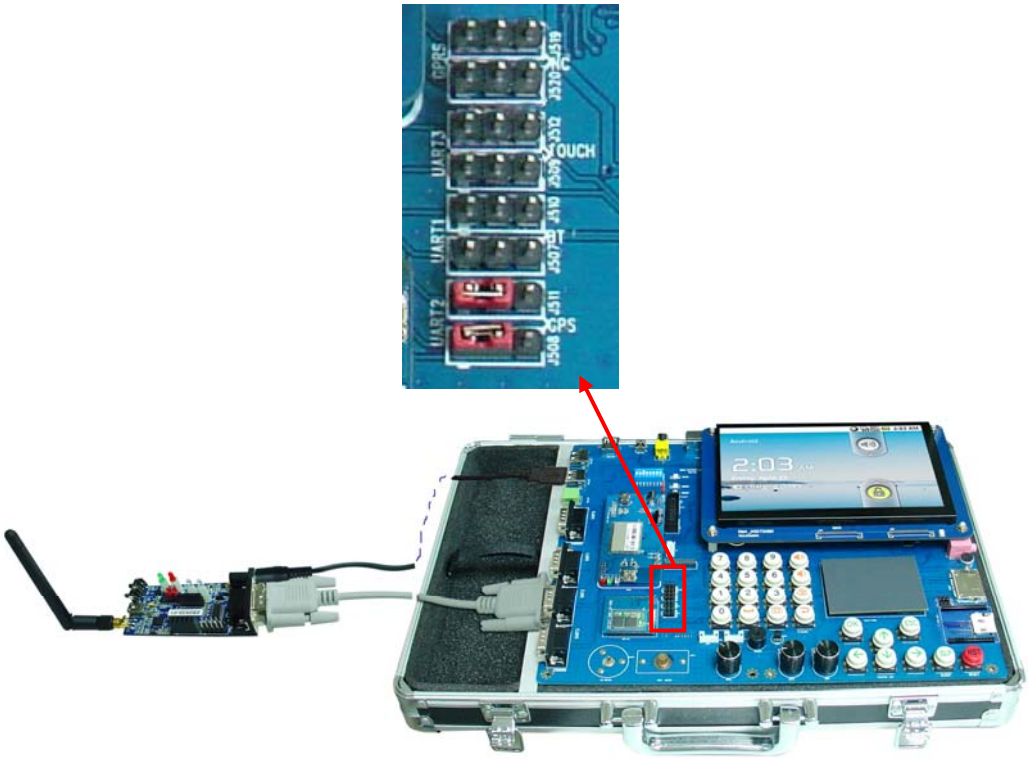
如果 ZB2530-01 模組接收端(主)接於 6410L 開發平台 UART 1 則點選 UART 1，UART 2 則點選 UART 2，鮑率(Baud Rate) 設定爲 115200

設定完成後將 ZB2530-01 模組發射端（從）接上電源或電池，6410L 將會自動收到 ZB2530-01 模組發射端（從） 發出的訊息

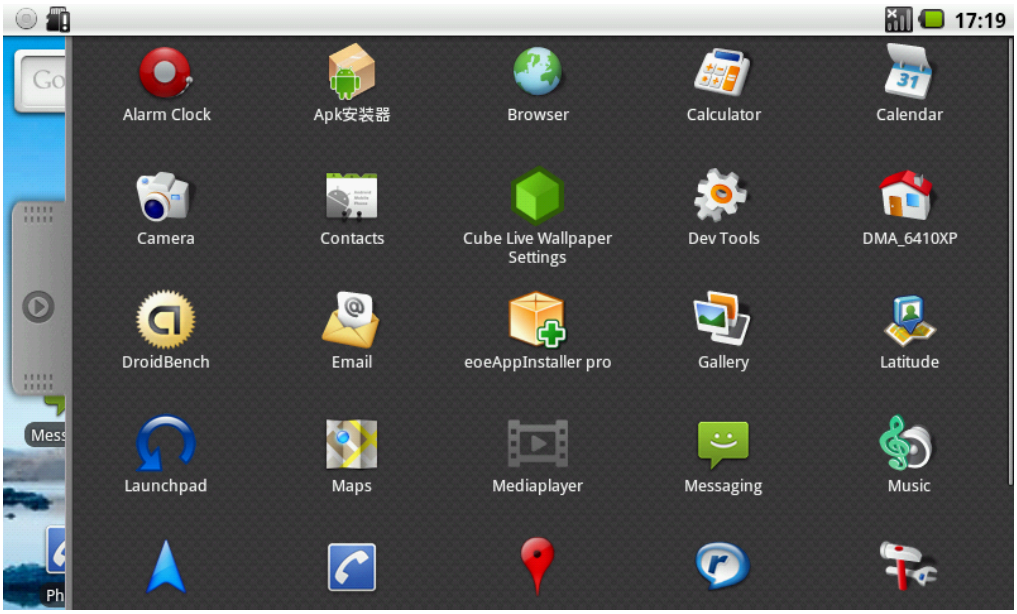


3. ZB2530-01 模組接收端（主）與 DMA-6410XP 教學平台連接

先將 ZB2530-01 模組接收端（主）與 DMA-6410XP 教學平台連接至 UART2，首先請注意 DMA-6410XP 串列埠 2 跳線地設定如圖

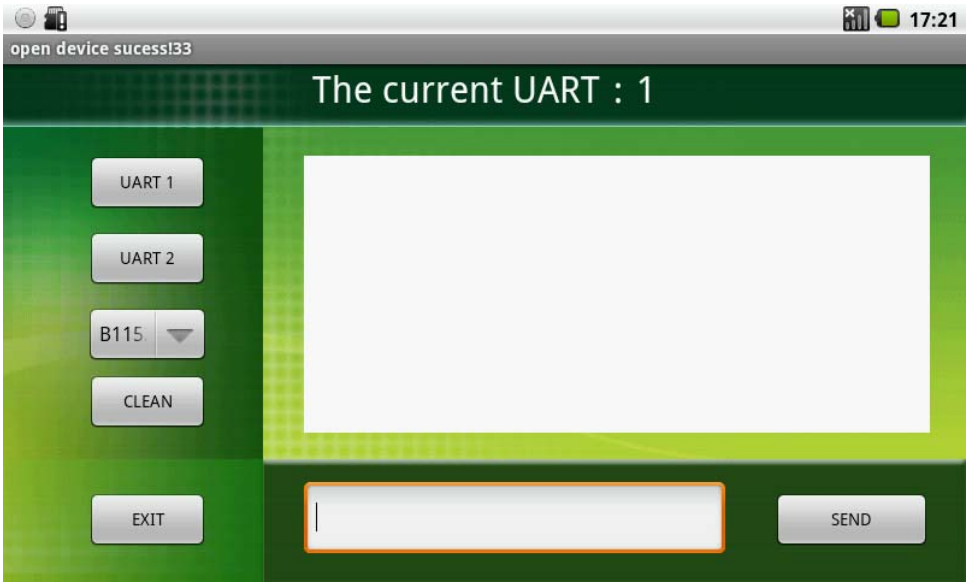


執行 6410XP 程式 DMA_6410XP 測試程式

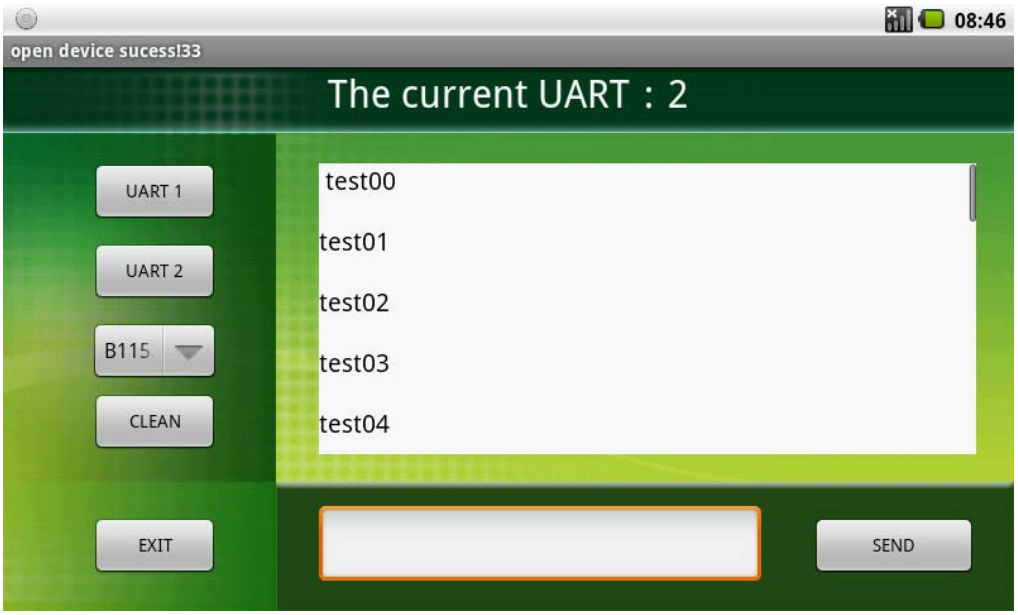


選取裡面 UART 測試程式

選取 UART 2 請不要選取 UART1，因爲會跟 6410XP 教學平台上的 GPRS 模組相衝突，鮑率(Baud Rate) 設定爲 115200。



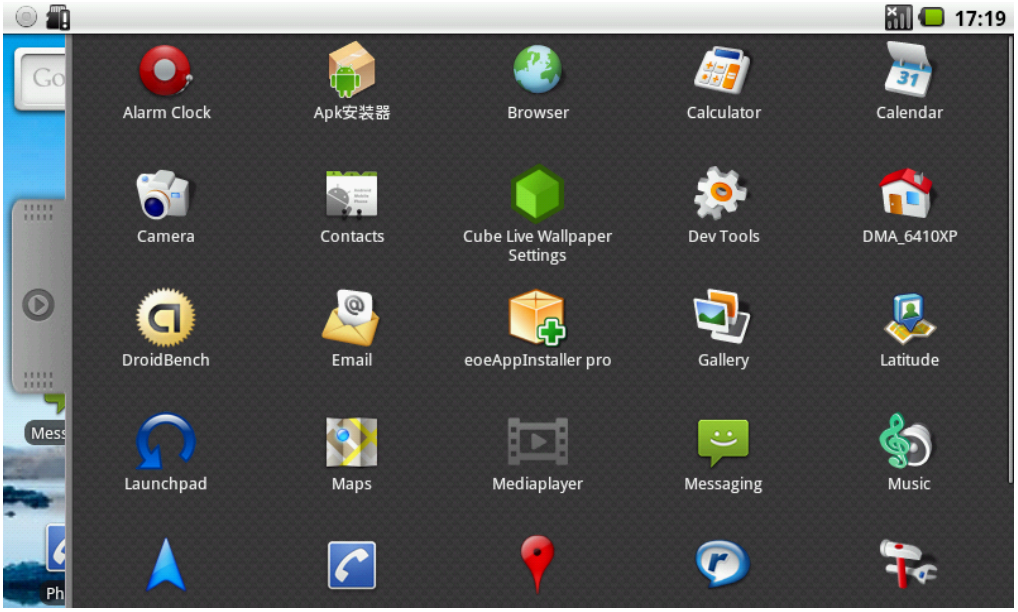
接著將 ZB2530-01 模組發射端(從)接上電源，就可從畫面上看到從 ZB2530-01 模組發射端(從)傳送給 ZB2530-01 模組接收端(主)傳至 6410XP 教學平台 UART 2 的訊息。



4.　ZB2530-01 模組接收端（主）與 HMI700-6410S平台連接

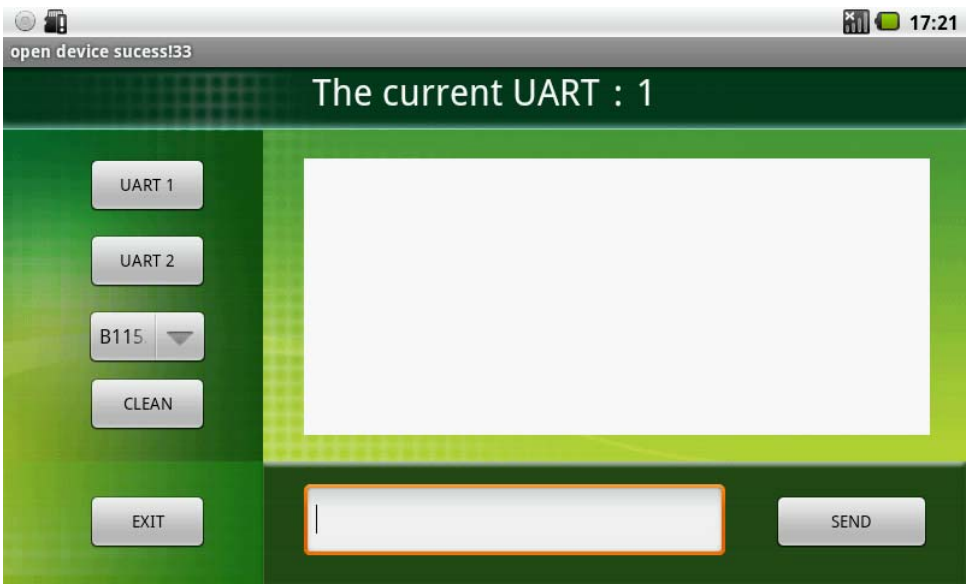
先將 ZB2530-01 模組接收端（主）與 HMI700-6410S 平台連接至 UART2，如下圖所示：

執行 HMI700-6410S 測試程式

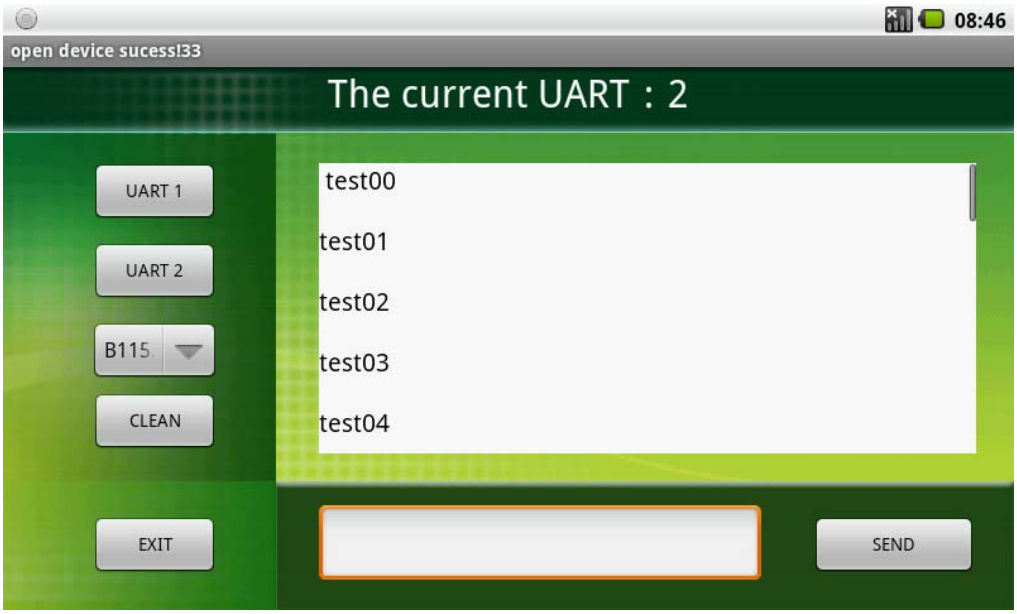


選取裡面 UART 測試程式

選取 UART 2 請不要選取 UART1，鮑率(Baud Rate) 設定爲 115200



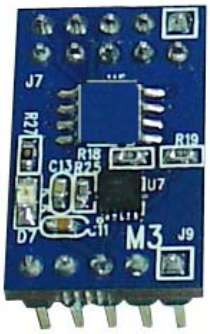
接著將 ZB2530-01 模組發射端(從)接上電源,就可從畫面上看到從 ZB2530-01 模組發射端（從）傳送給 ZB2530-01 模組接收端（主）傳至 HMI700-6410S 平台 UART 2 的訊息。

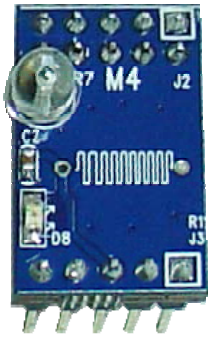
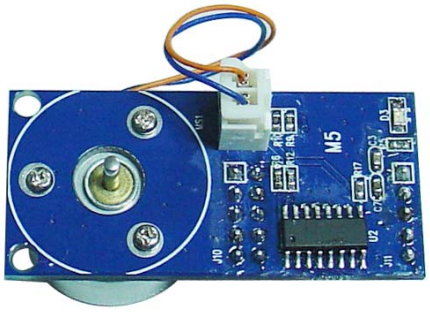


第三章 平台感測器模組介紹

本平台的感測器模組分爲基本型、進階型和進階擴充型三種。

I. 基本型有以下幾種：

1、 人體紅外線感測器模組（B110） 	2、 Gsensor 三軸位移加速感測器模組（B120） 

3、 AD590 雙端 IC 溫度感測器模組（B130） 	4、 TGS822 酒精感測器模組（B140） 
5、 ST-1KLA/B 光電晶體感測器模組（B150） 	6、 光敏電阻感測器模組（B160） 
7、 EEPROM 感測器模組（B170） 	8、 直流馬達感測器模組（B180） 

II. 進階型有以下幾種：

9、ITG-3200 陀螺儀感測器模組（A110） 	10、SHT10 溫濕度感測器模組（A120） 
11、TGS813 可燃性氣體感測器模組（A130） 	12、TC72 數位溫度感測器模組（A140） 

III. 進階擴充型有以下幾種：

13、 M-91S 串列式 GPS 衛星接收模組（E110） 	14、 NFC RFID 射頻 IC 卡讀寫器模組（E120） 
15、 TGS2602 空氣品質感測器模組（E130） 	16、 RFID 射頻 IC 卡讀寫器模組（E140） 
17、 SHT11 溫濕度感測器模組（E150） 	18、 SHT15 溫濕度感測器模組（E160） 

一、基本型感測器模組

3-1 人體紅外感測器模組（B110）

產品特點：

人體感測器，如圖（6）所示。人體感測器一種可探測運動人體的紅外熱釋感應器，由透鏡、感光元件、感光電路組成。一旦人體是移動，感光元件可產生極化壓差，感光電路發出有人的識別信號，達到探測運動人體的目的。此紅外熱釋感應器可應用於人體感應控制方面，並實現紅外防盜和紅外控制一體化，擴大了人體紅外熱釋感應器的應用範圍。

圖（6）

技術參數：

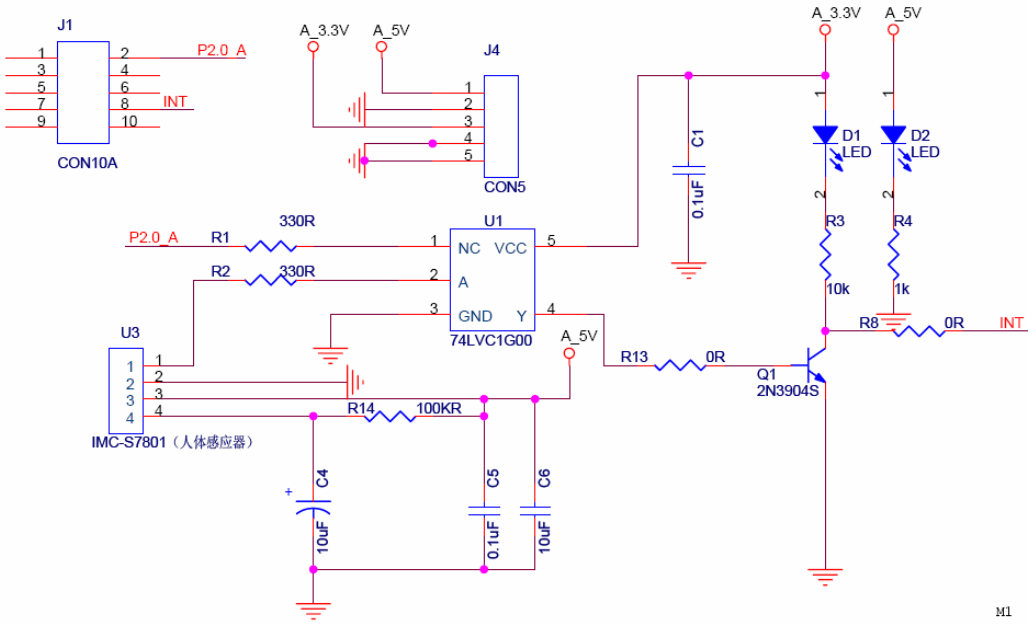
1．工作電壓：DC5V 至 20V
2．靜態功耗：65 微安
3．電平輸出：高 3.3V，低 0V
4．延遲時間：可調（0.3 秒~10 分鐘）
5．封鎖時間：0.2 秒
6．觸發方式：L 不可重複，H 可重複，預設值爲 H
7．感應範圍：小於 120 度錐角，7 米以內
8．工作溫度：-15℃~70℃
9．PCB 外型尺寸：32＊24mm，螺絲孔距 28mm，螺絲孔徑 2mm
10．感應透鏡尺寸：直徑 23mm

應用範圍：

走廊、樓道、化妝室、地下室、倉庫、車庫等場所的自動照明，排氣扇地自動抽風以及其他電器，特別適用於企業、飯店、商場、庫房敏感區域或安全區域的報警系統

電路介紹：

本電路透過 IMC-S7801 人體感應器當檢測到有人移動時，IMC-S7801 的第一腳輸出底電平再經過非門電路 U1 輸出高電平使 Q1 導通，這時 D1 亮，同時送給CPU的 INT 腳。電路如下所示

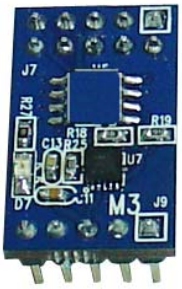


3-2 Gsensor 三軸位移加速感測器模組（B120）

產品特點：

GSensor 三軸位移加速感測器，如圖（5）所示。 GSensor 感測器的尺寸小巧、整合先進的移動設備電源管理系統、工作電流僅爲 200μA，非常適合應用於可攜式移動消費類產品。新一代加速度感測器透過參數和設置調整可滿足用戶在特定應用領域的需求。

BMA023 主要特點有三軸 ±2g/±4g/±8g 加速度感測器，可選 g 測量範圍和帶寬正常工作電流低至 200μA，支援 SPI 和 I2C 介面喚醒中斷，可編程自動喚醒模式工作電流低至 1μA，喚醒時間 1 毫秒，支援自測功能等。



圖（5）

技術參數：

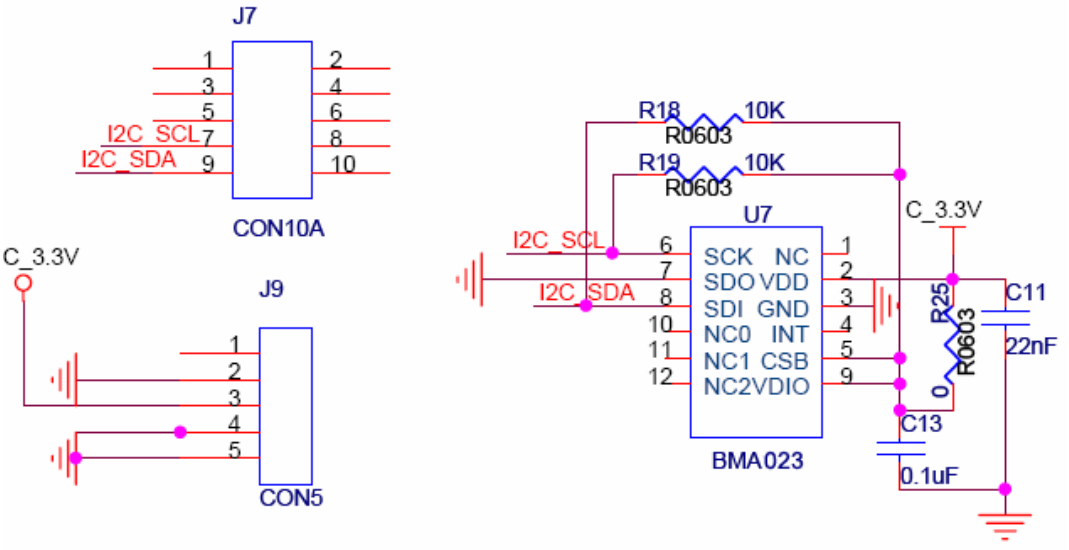
1．規格尺寸：3X3X0.9mm
2．分辨率：10 Bit
3．量程：±2g / ±4g / ±8g
4．功耗：工作 200μA，待機 1Ma
5．介面：SPI / I2C
6．封裝：LGA
7．品牌：德國 BOSCH Sensor tec

應用範圍：

偵測物體移動速率的變化量、遊戲、震動、消費性產品遊戲及電源管理等應用

電路介紹：

本電路透過 CPU 的 IIC 信號對 BMA023 進行操作。電路如下所示



3-3 AD590 雙端 IC 溫度感測器 (B130)

產品特點：

AD590 是一款雙端積體電路溫度感測器，其輸出電流與絕對溫度成比例。在4 V至30 V電源電壓範圍內，該器件可充當一個高阻抗、恒流調節器，調節係數爲1 μA/K。片內薄膜電阻經過鐳射調整，可用於校準器件，使該器件在298.2K (25°C)時輸出298.2 μA電流。

AD590 特別適合遠端檢測應用。它提供高阻抗電流輸出，對長線路上的壓降不敏感。任何絕緣良好的雙絞線都適用，與接收電路的距離可達到數百英尺。這種輸出特性還便 於AD590實現多工：輸出電流可以通過一個CMOS多工器切換，或者電源電壓可以透過一個邏輯門輸出切換。

技術參數：

1．具有線性電流輸出型態 (1μA／。K)

2．寬廣之量測範圍 (-55℃~150℃)

3．兩端元件 (電壓輸入，電流輸出)

4．寬廣的電源供應範圍 (+4V~+30V)

5．與本體隔離的感測器

6．封裝：CAN

7．品牌：美國 Analog Device

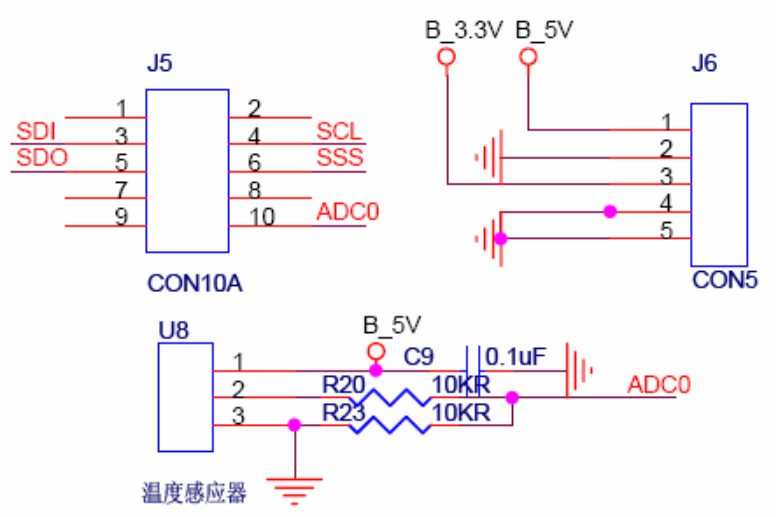
應用範圍：

樓宇控制、暖通空調及各種溫度量測的應用

電路介紹：

本電路透過 AD590 感測器對溫度的檢測，產生內部電流的變化，從而改變輸出電壓的大小，再由 ADC0 送給 CPU 處理。電路圖如下所示

圖（3）



3-4 TGS822 酒精感測器模組（B140）

產品特點：

酒精感測器，如圖（4）所示。酒精感測器 TGS 822 是一種 $Sn0_2$ 半導體氣體感測器，它用來偵測生物性的揮發性氣體，它的電阻值隨著大氣所含的氣體的增加而減少，這種電阻的減少現象，便是 TGS 822 用來偵測氣體的電氣輸出信號。其特性如下：

☆ 對酒精具有高靈敏度

☆ 具有長時間之穩定及信賴度

☆ 輸出信號大，價格低

技術參數：

1．測量範圍：50~5,000ppm

2．靈敏度（電阻比）：0.3-0.5

3．電路電壓：24V(DC/AC)

4．加熱器電壓：5V±0.2V(AC/DC)

5．封裝：塑膠、SUS 雙重金屬網

6．日本 FIGARO

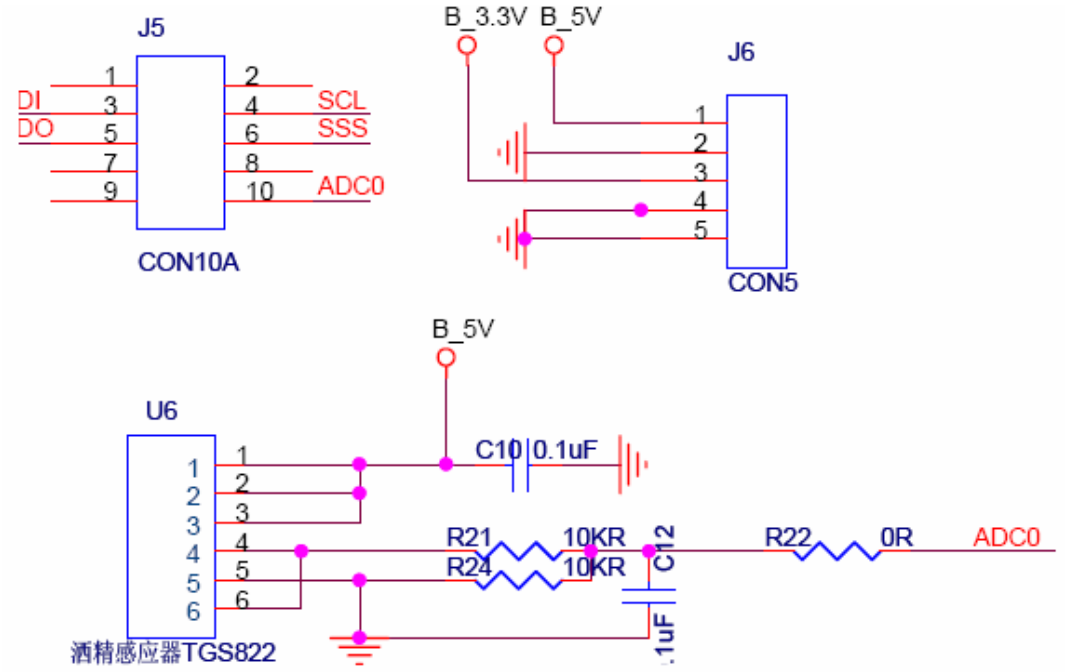
應用範圍：

酒精探測器、工業控制

圖（4）

電路介紹：

本電路透過 TGS822 感測器對氣體的檢測，產生內部電流的變化，從而改變輸出電壓的大小，再由 ADC0 送給 CPU 處理。電路圖如下所示



3-5 ST-1KLA/B 光電晶體感測器模組（B150）

產品特點：

光電晶體 (PhototranSistor) 是將半導體對光的反應特性與電晶體的放大作用整合成一體的元件，其放大方式與一般電晶體相似。

光電晶體（phototransistor）的結構與一般電晶體相同，只是在基極和集極接面的外殼上開有一處窗口，使光能照射到C-B接面，因爲光電晶體是由照射來當輸入信號，所以有些光電晶體沒有基極接腳。光電晶體因有放大作用，故比光二極體更爲靈敏，另有一種達靈頓連接之光電晶體，其放大倍數更，因此其靈敏度也比單一光電晶體高。

技術參數：

1．功率特性：中功率

2．頻率特性：高頻

3．極性：NPN型

4．入射光波長：以500—1000nm 較佳

5．封裝：金屬封裝

6．品牌: 日本KODENSHI

應用範圍：

1．汽車倒車雷達的反射波偵測

2．汽車測速照相

3．一般的讀卡機

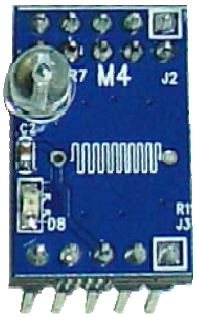
4．光電控制

5．計算機邏輯電路

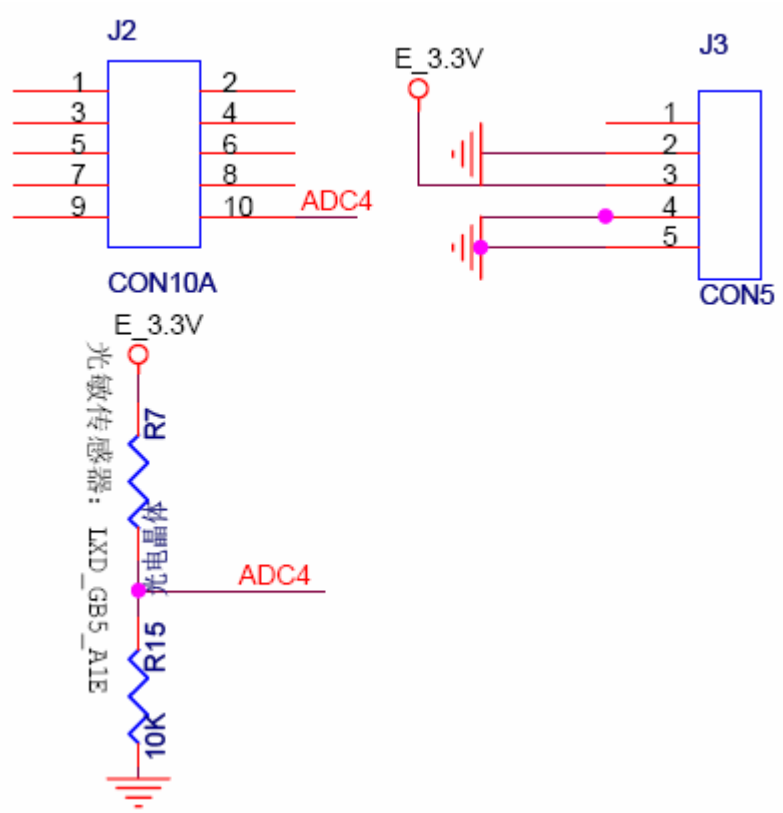
6．光轉速計

電路介紹

本電路透過光電晶體對光線強弱檢測，產生內部電阻值的變化，從而改變輸出電壓的大小，再由 ADC4 送給 CPU 處理。電路圖如下所示



圖（2）



3-6 光敏電阻感測器模組（B160）

產品特點：

光敏電阻感測器，如圖（1）所示。光敏電阻感測器是利用半導體的光電效應製成的一種電阻值隨入射光的強弱而改變的電阻器；入射光強，電阻減小；入射光弱，電阻增大。

常用的材料有硫化鎘 (cds) 、晒化鎘 (cdse) 與硫化鉛 (紅外線偵測用)，當光照射於CDS或CDSE上時，光子撞擊材料使價電子由原子軌道中釋放出來轉變爲傳導電子，使導電性增加而降低材料電阻。

光敏電阻兩端的電阻大小隨入射光強度成反比關係 。 光敏電阻本身不會產生能量，因此必須外加電源來提供電流流動 。

光敏電阻器一般用於光的測量、光的控制和光電轉換（將光的變化轉換爲電的變化）。常用的光敏電阻器爲硫化鎘光敏電阻器，它是由半導體材料製成的。光敏電阻器的阻值隨入射光線（可見光）的強弱變化而變化，在黑暗條件下，它的阻值（暗阻）可達1~10M歐，在強光條件下，它阻值（亮阻）僅有幾百至數千歐姆。光敏電阻器對光的敏感性（即光譜特性）與人眼對可見光（0.4~0.76）μm的回應很接近，只要人眼可感受的光，都會引起它的阻值變化。

圖（1）

技術參數：

1．型號：5528

2．最大電壓（V-dc）：150

3．最大功耗（mW）：100

4．環境溫度（°C）：-30--- +70

5．光譜峰值（nm）：540

6．回應時間（ms）：上升：20， 下降：30

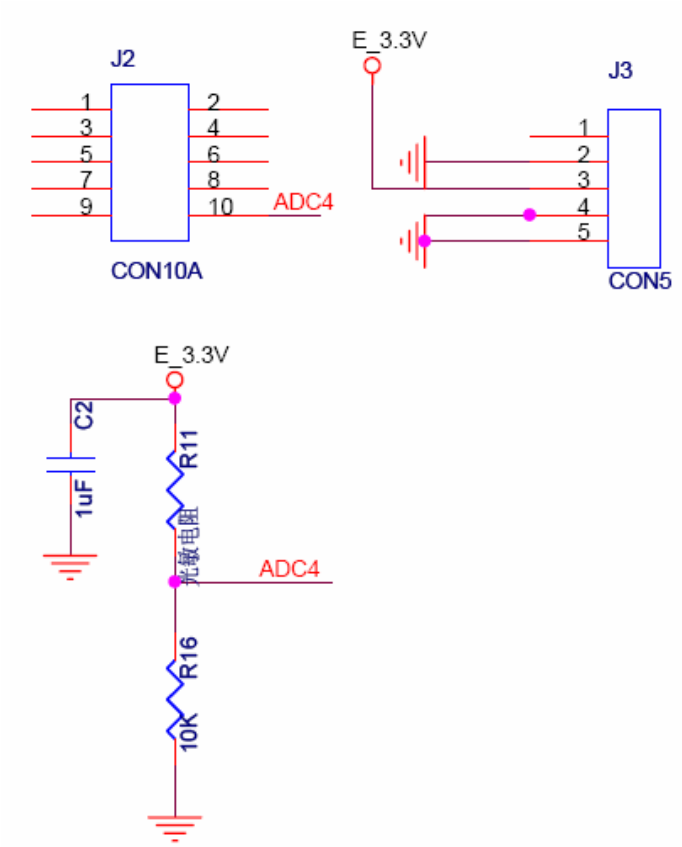
應用範圍：

1．路燈之自動點滅

2．照相機之測光計

電路介紹：

本電路透過光敏電阻對光線強弱檢測，產生內部電阻值的變化，從而改變輸出電壓的大小，再由 ADC4 送給 CPU 處理。電路圖如下所示


J2
1
3
5
7
9
2
4
6
8
10
ADC4
CON10A
E_3.3V
J3
1
2
3
4
5
CON5
E_3.3V
C2
1uF
R11
光敏电阻
ADC4
R16
10K

3-7 EEPROM 感測器模組（B170）

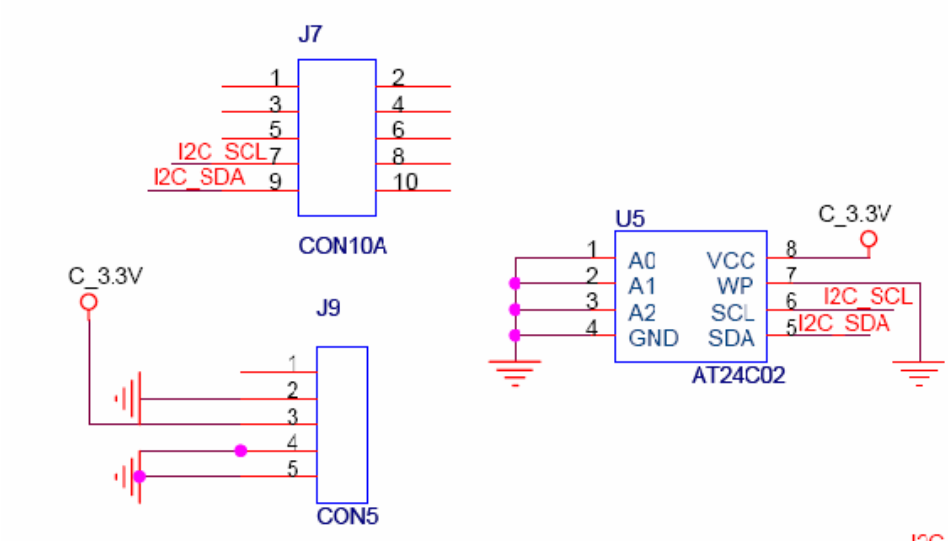
產品特點：

EEPROM 感測器模組，如圖（7）所示。本模組的EEPROM 感測器使用的是AT24C02 晶片，它是一個 2K 位元串列 CMOS EEPROM，內部含有 256 個 8 位元組，CATALYST 公司的先進 CMOS 技術實質上減少了器件的功耗。AT24C02 有一個 16 位元組頁寫緩衝器。該器件透過 IC 匯流排介面進行操作，有一個專門的防寫功能。

圖（7）

電路介紹：

本電路透過 CPU 的 IIC 信號對 AT24C02 進行操作。電路如下所示



3-8 直流馬達感測器模組（B180）

產品特點：

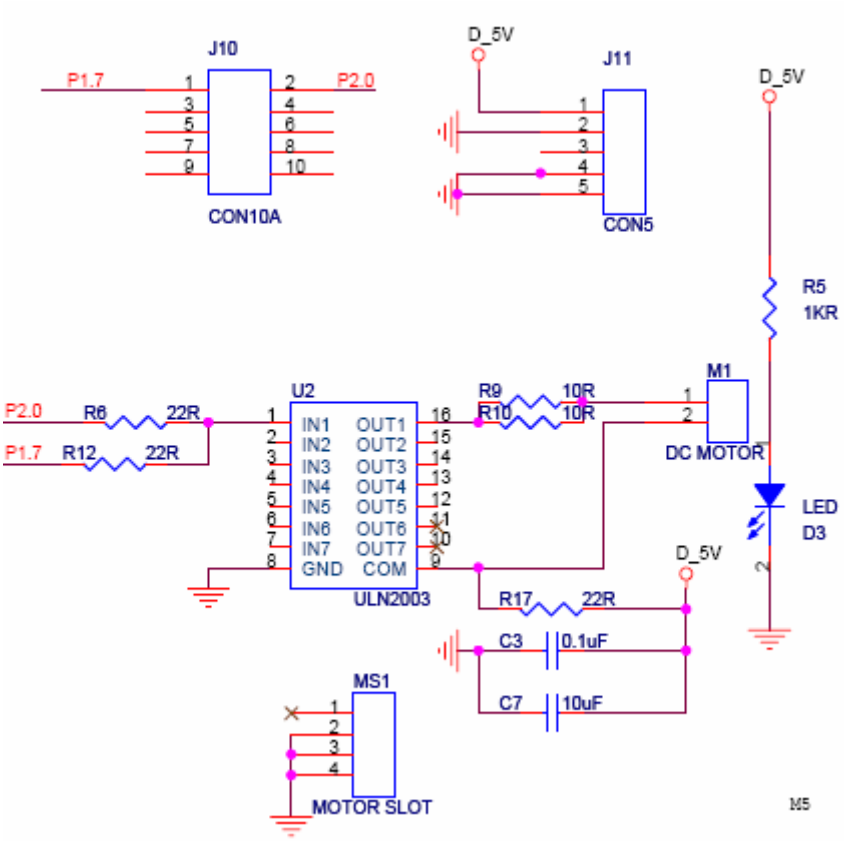
直流馬達就是將直流電能轉換成機械能的馬達。直流馬達的勵磁方式是指對勵磁繞組如何供電、產生勵磁磁通勢而建立主磁場的問題。

按類型主要分爲：直流有刷馬達和直流無刷馬達

我們使用的是直流有刷馬達，如下圖所示。它的工作原理是：定子勵磁繞組通入直流勵磁電流，產生勵磁磁場；當電樞從外界引入直流電，經碳刷給換向器，再通過換向器將此直流電轉化爲交流電引入電樞繞組，產生電樞電流，此電流產生磁場，與勵磁磁場合成爲氣隙磁場。電樞繞組切割氣隙合成磁場，按左手定則可判斷出電樞產生轉矩。

電路介紹：

本電路配直流馬達（M5）一台，供使用者實驗用。電路如下所示


J10
P1.7
P2.0
CON10A
D_5V
J11
CON5
D_5V
R5
1KR
M1
DC MOTOR
U2
P2.0
R6
22R
P1.7
R12
22R
IN1
IN2
IN3
IN4
IN5
IN6
IN7
GND
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
OUT7
COM
ULN2003
R9
10R
R10
10R
LED
D3
D_5V
R17
22R
C3
0.1uF
C7
10uF
MS1
MOTOR SLOT
M5


M5

圖（8）

二、進階型感測器模組

3-9 ITG-3200 陀螺儀感測器模組（A110）

產品特點：

ITG-3200 是首個在單一矽晶片上，整合了回轉儀和三個旋轉軸（傾斜、轉動、偏轉），功耗減少了 50%，封裝尺寸減少 67%。器件採用 QFN 封裝，尺寸爲 4mmx4mmx0.9mm，其三軸融合消除了軸交叉干擾和零偏移率。數位輸出無需外部、高解析度 ADC。其他性能包括低通濾波器，嵌入式溫度感測器，I2C 串列介面，滿刻度範圍高達 2000°/s，噪音係數爲 0.03º/s/Hz，交叉軸隔離爲±2%。

ITG3200 陀螺儀感測器，是一種空間相角感測器，主要檢測空間某些相位的傾角變化、位置變化，主要用於空間物理領域，特別在航空、航海方面有較多的用途，如：飛機上的陀螺儀，當飛機在做 360˚翻轉的時候，陀螺儀將會保持原始的基準狀態不變，從而讓駕駛員找到本飛機在空間狀態的相位變化，也就是：當時飛機處在什麼相位。

技術參數：

1．一晶片並以數位元方式輸出的三軸 MEMS 陀螺儀 IC
2．運用三個 16-bit 的類比 /數位轉換器(ADCs)來數位化陀螺儀輸出端
3．內建溫度感測器與精准差僅 2%的內建震盪設計
4．快速模式的 I²C(Fast Mode I²C)(400kHz)介面
5．包含 10,000g 的耐震容忍度來符合掌上型消費性產品的需求
6．封裝：QFN Package，24Pin，4mm x 4mm x 0.9mm
7．品牌：美國 InvenSense

應用範圍：

1．運動感測遊戲遙控

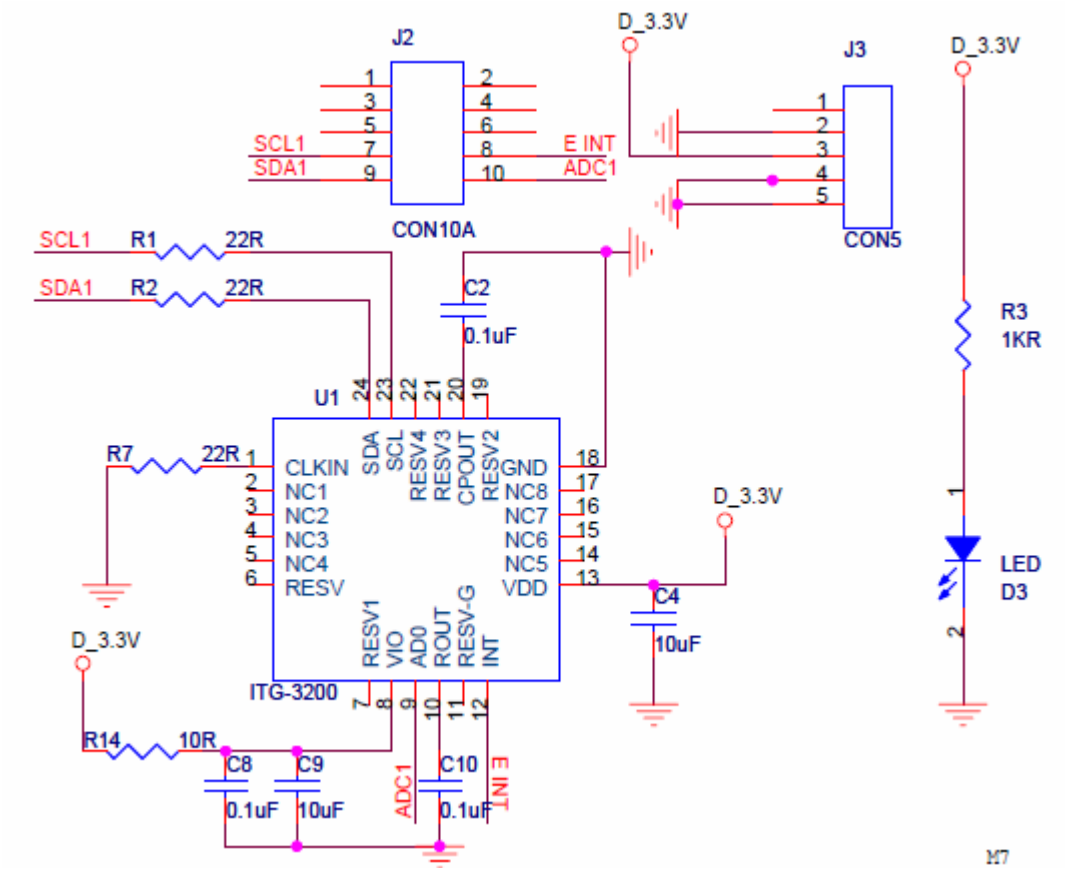
2．運動感測掌上型遊戲

3．運動感測 3D 滑鼠及 3D 遙控

4．“無觸控”人機介面

5．健康與運動的監控

電路介紹：



3-10 SHT10 溫濕度感測器模組（A120）

產品特點：

瑞士 Sensirion 公司推出的 SHT10 單片數位溫濕度集成感測器，如圖（10）所示，是採用 CMOS 過程微加工專利技術（CMOSens technology），確保產品具有極高的可靠性和出色的長期穩定性。該感測器由 1 個電容式聚合體測濕元件和 1 個能隙式測溫元件組成，並與 1 個 14 位 A/D 轉換器以及 1 個 2-wire 數位介面在單晶片中無縫結合，使得該產品具有功耗低、反應快、抗干擾能力強等優點。SHT10 的主要特點如下：

- ◆ 相對濕度和溫度的測量兼有露點輸出
- ◆ 全部校準，數位輸出
- ◆ 介面簡單（2-wire），回應速度快
- ◆ 超低功耗，自動休眠
- ◆ 出色的長期穩定性
- ◆ 超小體積（表面貼裝）
- ◆ 測濕精度±4.5%RH，測溫精度±0.5℃（25℃）

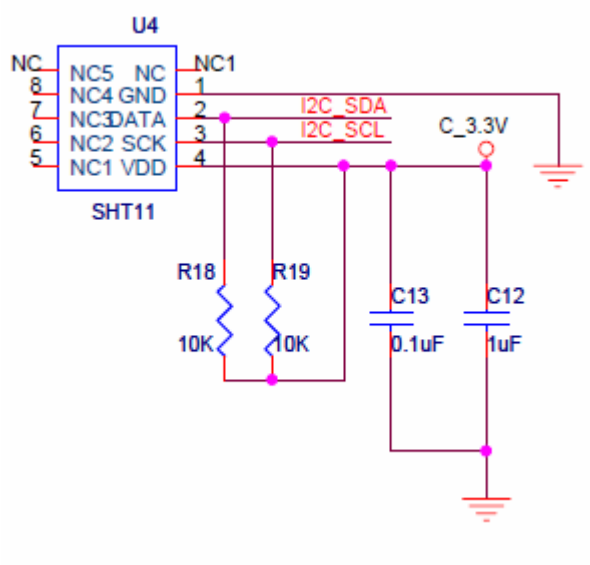
技術參數：

1．SHT10 系列爲貼片型溫濕度感測器晶片
2．全量程標定，兩線數位輸出
3．濕度測量範圍：0～100%RH
4．溫度測量範圍：-40～+123.8℃
5．濕度測量精度： ±4.5%RH
6．溫度測量精度： ±0.5℃
7．封裝：SMD（LCC）
8．品牌：瑞士 SENSIRION

應用範圍：

資料獲取器、變送器、自動化程序控制、汽車行業、樓宇控制&暖通空調、電力、計量測試、醫藥業

電路介紹：



3-11 TGS813 可燃性氣體感測器模組（A130）

產品特點：

TGS813 可燃氣體感測器的原理和 TGS811 酒精感測器的原理是一樣的，在這裏就不做具體介紹了。TGS813 模組如下圖所示：

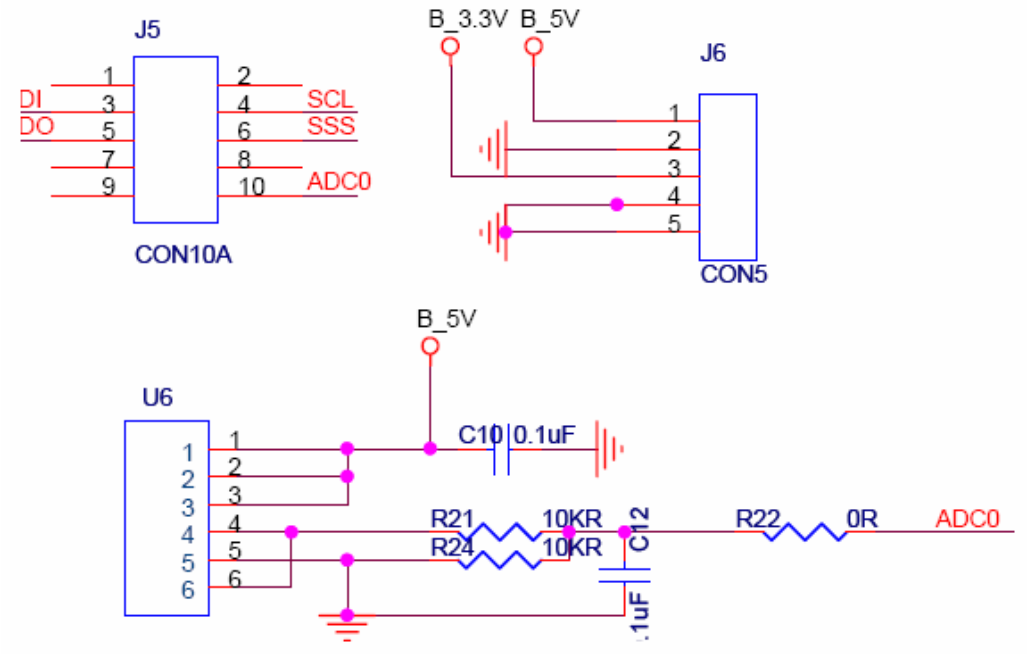
技術參數：

1．電路電壓：〈24V（AC/DC）
2．測量範圍：500-10,000ppm
3．靈敏度（電阻比）：0.55-0.65
4．加熱器電壓：5V±0.2V（AC/DC）
5．封裝：塑膠、SUS 雙重金屬網
6．日本 FIGARO

應用範圍：

1．家庭用洩漏氣體報警器
2．工業用可燃氣體報警器
3．便攜式氣體檢知器

電路介紹：



3-12 TC72 數位溫度感測器模組（A140）

產品特點：

數位溫度感測器，如圖（9）所示。TC72 是一個數位溫度感測器，能夠檢測 -55 攝氏度至+125 攝氏度的溫度範圍。有一個串列介面，可以與主機控制器或其他週邊設備進行通信。該TC72 介面與SPI 協定、IIC 協定相容，不需要任何額外的外部元件。它可連續轉換，也可以在溫度轉換模式或單次轉換模式下使用。如設置連續轉換模式，溫度大約每150 ms 更新存儲在暫存器中的溫度資料。與此相反，單次模式執行一個單一的溫度測量。TC72 不但具有很高的精度，而且非常節能。

圖（9）

技術參數：

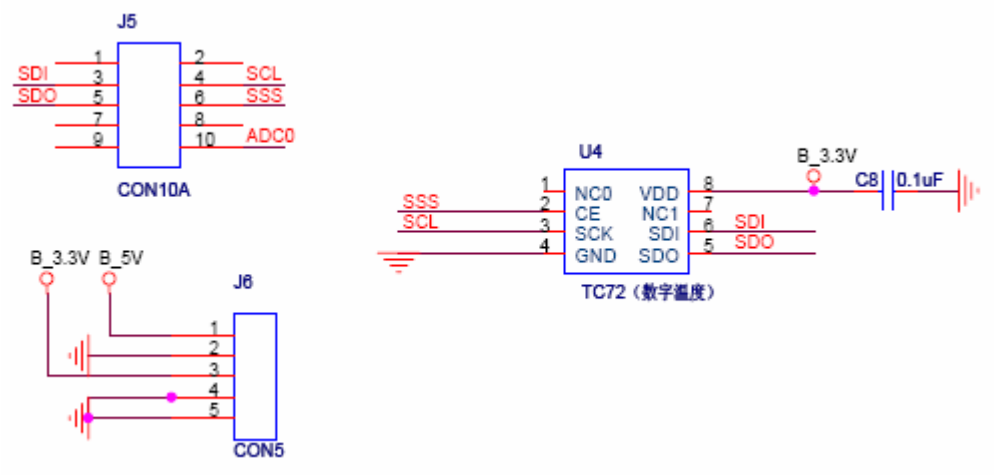
1．SPI 相容介面
2．10-Bit 解析度(0.25˚C/Bit)
3．±2˚C（最大）精度從-40˚C 到+85˚C
4．±3˚C（最大）精度從-55˚C 到+125˚C
5．2.65V 到 5.5V 工作範圍
6．低功耗：- 250 µA (typ.)連續溫度轉換模式
　　- 1 µA（最大）關斷模式
7．封裝：8-Pin MSOP
8．品牌：美國 Microship

應用範圍：

1．個人電腦和伺服器
2．硬碟驅動器和其他 PC 外部設備
3．辦公設備、資料通信設備、移動電話
4．通用溫度監測

電路介紹：

本電路配有高精度的 TC72 數位溫度感測器，供使用者實驗用。電路如下所示

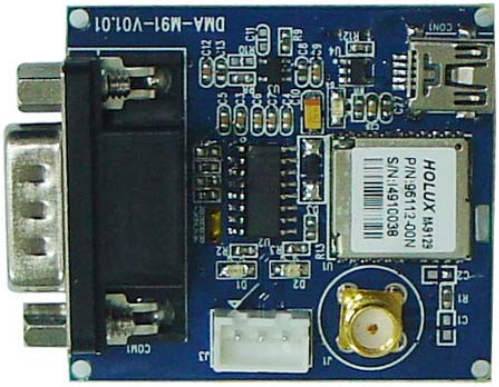


三、進階型擴充感測器模組

3-13　M-91S 串列式 GPS 衛星接收模組（E110）

產品特點：

M-91S 是一種根據低耗電 Mediatek GPS 解決方案設計的超小型 13 x 15 x 2.2 mm GPS 引擎模組。它對於導航應用提供高達 -159dBm 的絕佳靈敏度與快速的第一次定位時間。**M-91S** 拉出 DTYPE 及 3P RS232 信號，非常方便於在外接 GPS 設備的平台使用。



技術參數：

1．精巧外型：13 * 15 * 2.2 mm

2．高靈敏性 159dBm

3．可搜尋多達 32 個衛星頻道

4．快速位置修正、低耗電量

5．支援 NMEA0183 V 3.01 數據通訊協定

6．可外接電池、獨立供電

7．品牌：台灣 HOLUX

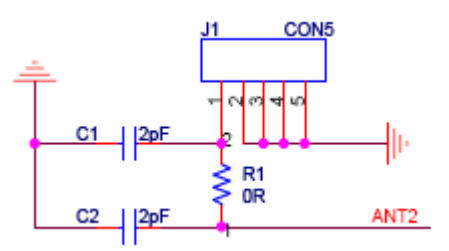
應用範圍：

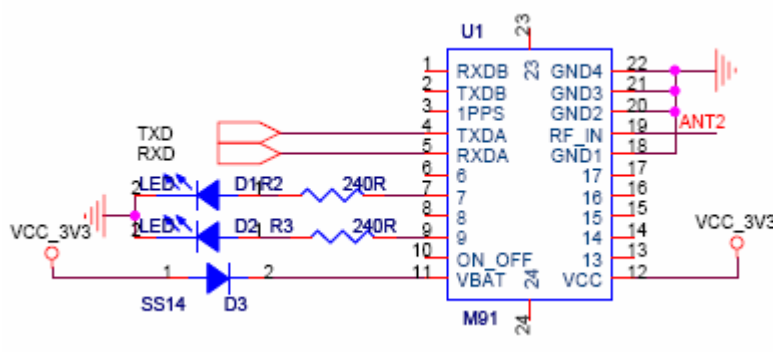
1．定位服務的即時導航

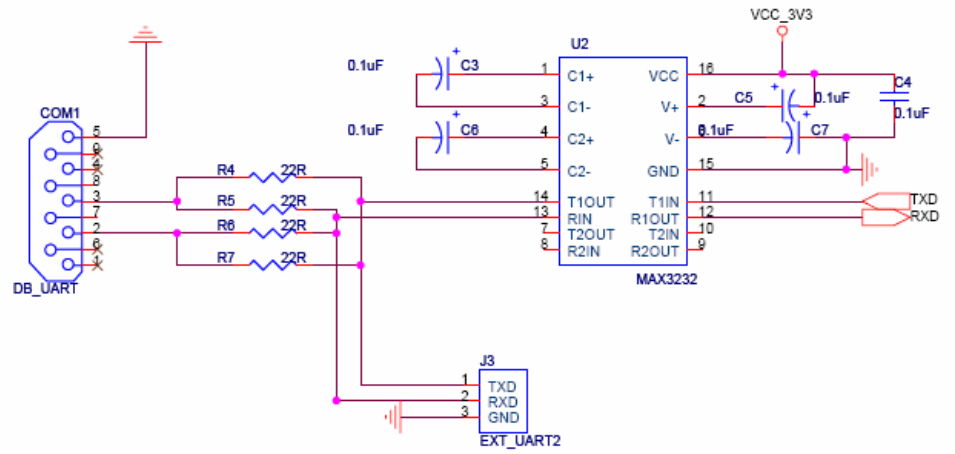
2．適用於汽車導航、船隻導航、艦隊管理、AVL 和定位服務、自動導航

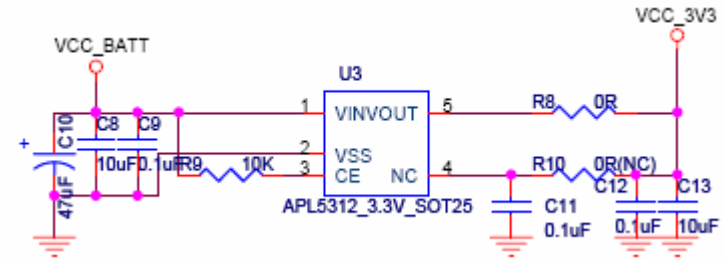
3．個人導航或旅遊裝置、追蹤裝置／系統和地圖裝置應用

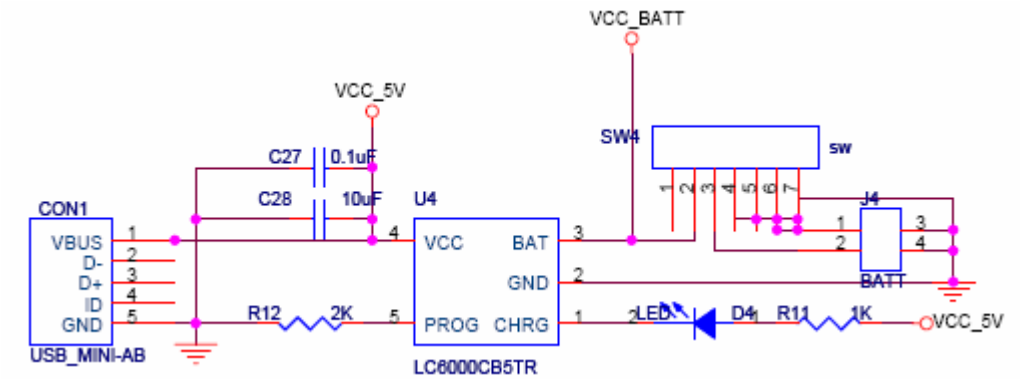
電路介紹：






VCC_3V3
U2
0.1uF
C3
C1+
VCC
C1-
V+
C5
0.1uF
C4
0.1uF
C6
C2+
V-
C7
C2-
GND
COM1
R4 22R
R5 22R
R6 22R
R7 22R
T1OUT
T1IN
TXD
RXD
RIN
R1OUT
T2OUT
T2IN
R2IN
R2OUT
MAX3232
DB_UART
J3
TXD
RXD
GND
EXT_UART2


VCC_BATT
VCC_3V3
U3
VINVOUT
R8 0R
C10
C8
C9
VSS
CE
NC
R10 0R(NC)
C12
C13
10uF
0.1uF
R9 10K
APL5312_3.3V_SOT25
C11
0.1uF
0.1uF
10uF
47uF


VCC_BATT
VCC_5V
SW4
sw
C27 0.1uF
C28 10uF
U4
J4
CON1
VBUS
BAT
D-
GND
D+
ID
GND
R12 2K
PROG
CHRG
LED
D4
R11 1K
VCC_5V
BATT
USB_MINI-AB
LC6000CB5TR

3-14 NFC RFID 射頻 IC 卡讀寫器模組（E120）

產品特點：

NFC-ACR122S 是一款串列埠型 NFC 非接觸式智慧射頻 IC 卡讀寫器。NFC-ACR122S 採用 13.56 MHz 非接觸式（RFID）技術並符合 ISO/IEC 18092 NFC 標準，能夠支援 Mifare®、ISO14443 A 類和 B 類、FeliCa 和 NFC 技術，包括所有四類 NFC 標籤和三種 NFC 模式（讀寫器、peer-to-peer 和類比卡片模式）。

技術參數：

1．RS232 串列埠、USB 介面供電
2．讀寫速度高達 424 kbps
3．內置天線，讀取速度高達 50 mm
4．支援 ISO 14443 第四部分 A 類和 B 類、Mifare、FeliCa 和所有四類 NFC（ISO/IEC 18092）標籤
5．支援新的 Mifare Ultralight C、Mifare Plus SL1 (4 Byte UID) 和 SL3
6．支援所有三種 NFC 模式：讀寫器、peer-to-peer 和類比卡片模式

應用範圍：

1．自動收費系統
2．自動販賣機
3．公交終端、考勤管理

4．非接觸式公用電話

5．網路訪問控制

6．門禁控制

3-15　TGS2602 空氣品質感測器模組（E130）

產品特點：

TGS 空氣品質感測器的敏感材料是金屬氧化物，最具代表性的是 SnO2。金屬氧化物晶體如 SnO2 在空氣中被加熱到一定高的溫度時，氧被吸附在帶一個負電荷的晶體表面。然後，晶體表面的供與電子被轉移到被吸附的氧上，結果在一個空間電荷層留下正電荷。這樣，表面勢能形成一個勢壘，從而阻礙電子流動。

在感測器的內部，電流流過 SnO2 微晶的結合部位（晶粒邊界）。在晶粒邊界，吸附的氧形成一個勢壘阻止載流子自由移動，感測器的電阻即緣於這種勢壘。還原性氣體出現時，帶有負電荷的氧的表面濃度降低，導致晶粒邊界的勢壘降低。降低了的勢壘使感測器的阻值減小了。

綜上所述，簡單的說，敏感元件由一個以金屬鋁做襯底的金屬氧化物敏感晶片和一個完整的加熱器組成。在檢測氣體時，感測器的傳導率依賴於空氣中氣體濃度的變化。

TGS2600 對空氣中的低濃度香煙污染物，像 H2、CO 等有較高得敏感度，感測器能檢測到在幾個 ppm 級 H2 含量。

空氣品質感測器模組如下圖所示：

技術參數：

1．空氣品質感測器可測量範圍：1-30ppm

2．靈敏度：0.15～0.5（10ppmH2 阻值/空氣中阻值）

3．空氣品質感測器輸出信號：可變電阻值

4．環境溫度：-10～50℃

5．金屬網

6．品牌：日本 FIGARO

應用範圍：

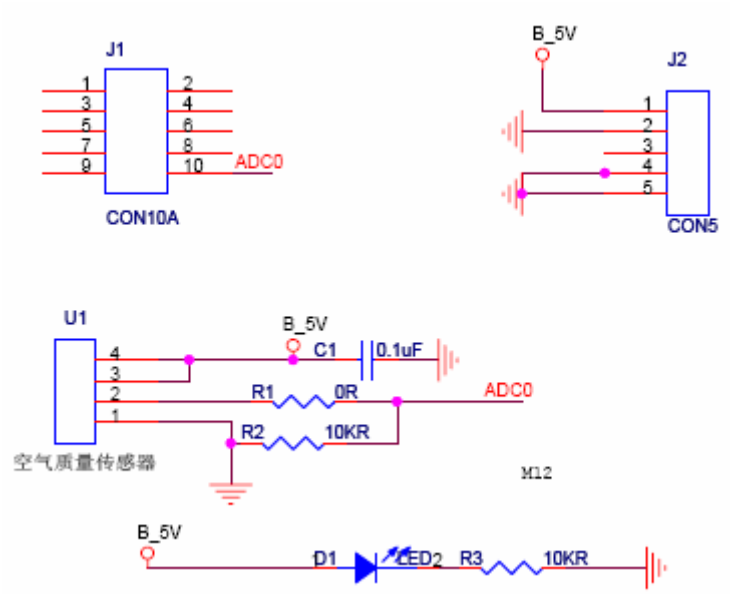
1．空氣清新機

2．換氣扇控制檢測器

3．脫臭器控制檢測器

4．室內空氣監視器

電路介紹：



3-16 RFID 射頻 IC 卡讀寫器模組（E140）

產品特點：

可讀寫符合 ISO14443 標準的射頻卡，PHLIPS 公司的 TYPEA MF1ICS50、MF1ICL10、MF1ICS70、MIFAREPRO、MIFAREDES、MF Ultralight、DESFire，SIEMENS 公司的 TYPEA、上海華虹的 SHC1102、復旦籌碼卡以及相容上述卡型的卡片；SAM 卡，符合 T=0 和 T=1 協議的 CPU 卡

技術參數：

1．工作頻率：13.56MHz
2．通訊介面：RS232
3．讀寫卡距離：0～70mm
4．串列通訊鮑率：1200BPS～15200BPS
5．外型尺寸：143mm×110mm×28mm
6．提供手攜式設備的 WinCE5.0/6.0 、Linux 2.6.X 及 Android 2.1 控制 DRIVER 及 AP 應用程式

應用範圍：

1．考勤管理
2．非接觸式公用電話
3．網路訪問控制
4．門禁控制

3-17 SHT11 溫濕度感測器模組（E150）

產品特點：

瑞士 Sensirion 公司推出的 SHT11 單片數位溫濕度集成感測器，如圖（10）所示，是採用 CMOS 過程微加工專利技術（CMOSens technology），確保產品具有極高的可靠性和出色的長期穩定性。該感測器由 1 個電容式聚合體測濕元件和 1 個能隙式測溫元件組成，並與 1 個 14 位 A/D 轉換器以及 1 個 2-wire 數位介面在單晶片中無縫結合，使得該產品具有功耗低、反應快、抗干擾能力強等優點。SHT11 的主要特點如下：

- ◆ 相對濕度和溫度的測量兼有露點輸出
- ◆ 全部校準，數位輸出
- ◆ 介面簡單（2-wire），回應速度快
- ◆ 超低功耗，自動休眠
- ◆ 出色的長期穩定性
- ◆ 超小體積（表面貼裝）
- ◆ 測濕精度±3.0%RH，測溫精度±0.4℃（25℃）

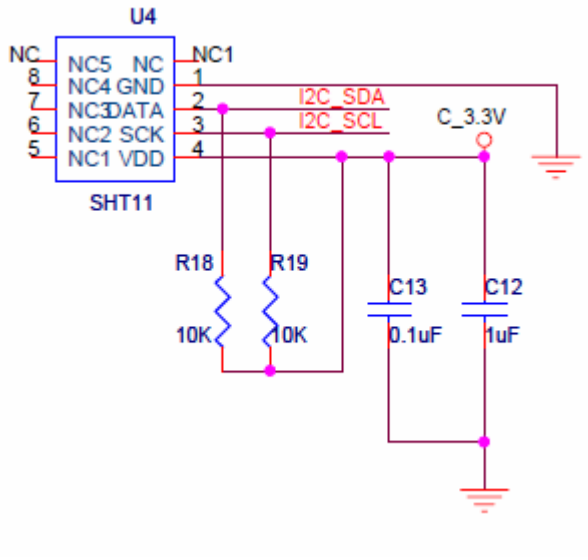
技術參數：

1．SHT11 系列爲貼片型溫濕度感測器晶片
2．全量程標定，兩線數位輸出
3．濕度測量範圍：0～100%RH
4．溫度測量範圍：-40～+123.8℃
5．濕度測量精度： ±3.0%RH
6．溫度測量精度： ±0.4℃
7．封裝：SMD（LCC）
8．品牌：瑞士 SENSIRION

應用範圍：

資料獲取器、變送器、自動化程序控制、汽車行業、樓宇控制&暖通空調、電力、計量測試、醫藥業

電路介紹：



3-18 SHT15 溫濕度感測器模組（E160）

產品特點：

瑞士 Sensirion 公司推出的 SHT15 單片數位溫濕度集成感測器，如圖（10）所示，是採用 CMOS 過程微加工專利技術（CMOSens technology），確保產品具有極高的可靠性和出色的長期穩定性。該感測器由 1 個電容式聚合體測濕元件和 1 個能隙式測溫元件組成，並與 1 個 14 位 A/D 轉換器以及 1 個 2-wire 數位介面在單晶片中無縫結合，使得該產品具有功耗低、反應快、抗干擾能力強等優點。SHT15 的主要特點如下：

- ◆ 相對濕度和溫度的測量兼有露點輸出
- ◆ 全部校準，數位輸出
- ◆ 介面簡單（2-wire），回應速度快
- ◆ 超低功耗，自動休眠
- ◆ 出色的長期穩定性
- ◆ 超小體積（表面貼裝）
- ◆ 測濕精度±2.0%RH，測溫精度±0.3℃（25℃）

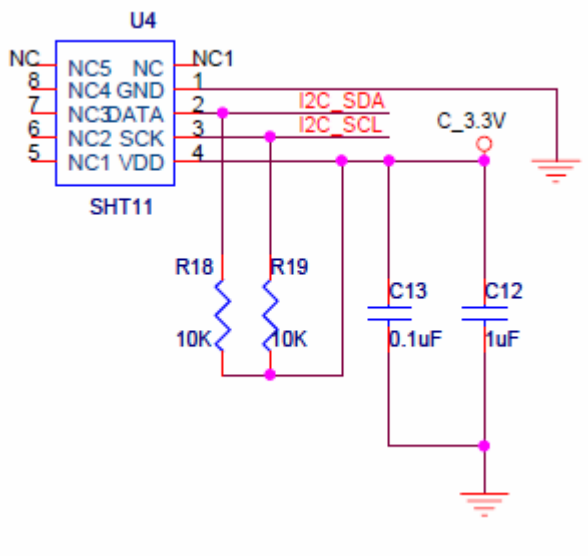
技術參數：

1．SHT15 系列爲貼片型溫濕度感測器晶片
2．全量程標定，兩線數位輸出
3．濕度測量範圍：0～100%RH
4．溫度測量範圍：-40～+123.8℃
5．濕度測量精度： ±2.0%RH
6．溫度測量精度： ±0.3℃
7．封裝：SMD（LCC）
8．品牌：瑞士 SENSIRION

應用範圍：

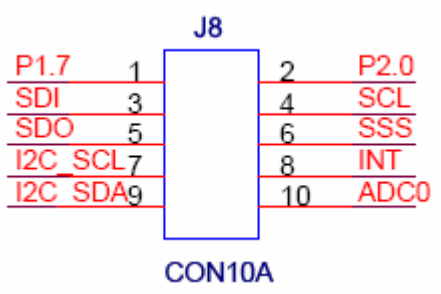
資料獲取器、變送器、自動化程序控制、汽車行業、樓宇控制&暖通空調、電力、計量測試、醫藥業

電路介紹：



3-19 擴充介面電路

本電路提供 10 路由 CPU 外擴 GPIO 介面。電路如下所示

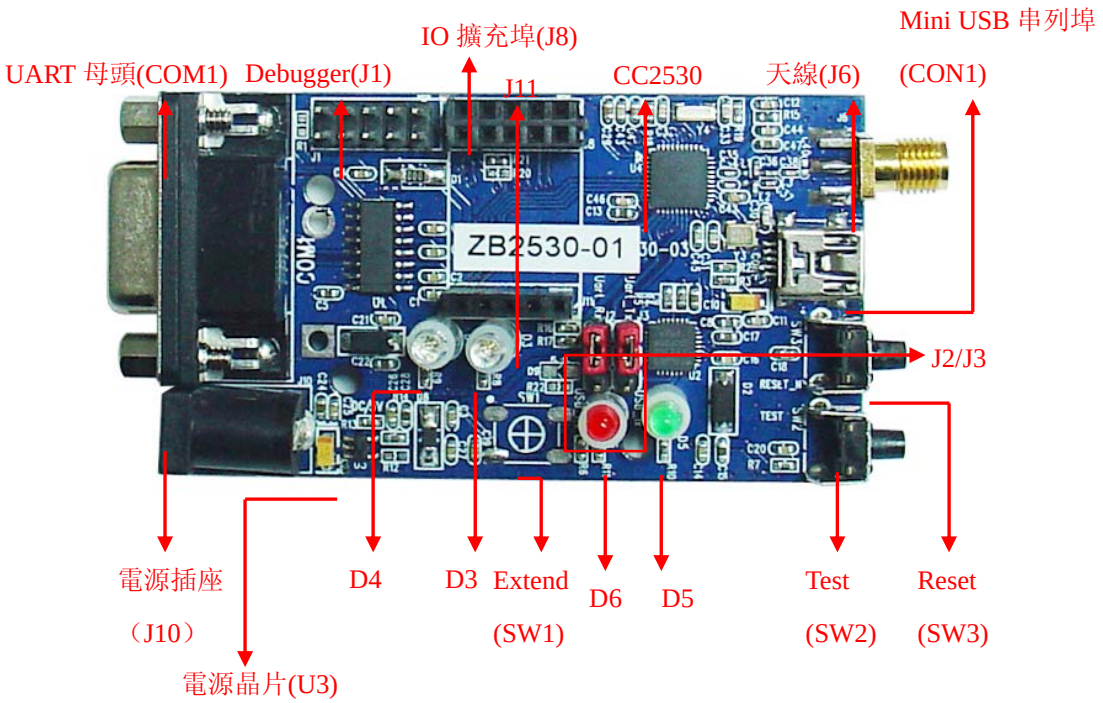


連接器的接腳圖：

接腳	功能
1	P1.7--通用 GPIO
2	P2.0--通用 GPIO
3	SDI–SPI 資料登錄
4.	SCL–SPI 時鐘
5	SDO–SPI 資料輸出
6	SSS–SPI 控制腳
7	I2C_SCL–I2C 時鐘
8	INT---中斷
9	I2C_SDA–I2C 資料埠
10	ADC0----模擬信號輸入口

3-20 感測器模組與 ZB2530-01 連接

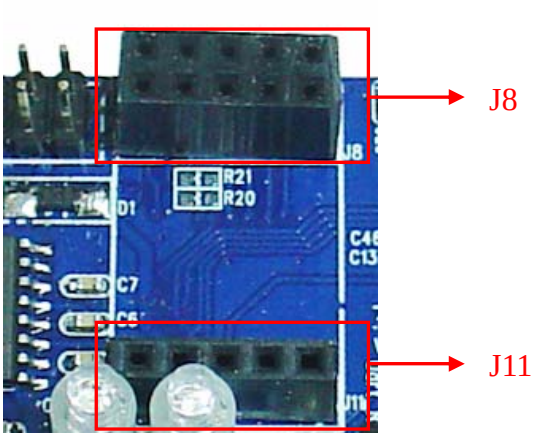
本操作步驟以ZB2530-01模組作爲硬體載體，介面如圖（10）所示

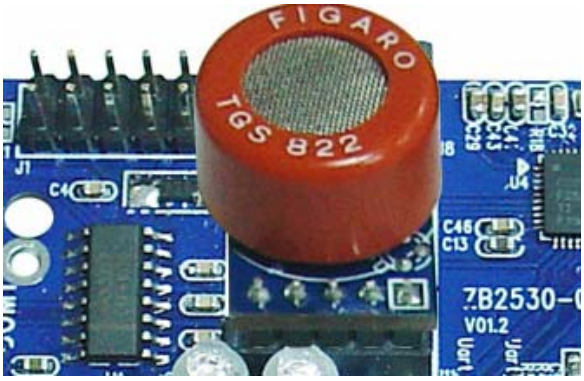


圖（10）

連接步驟：

a. 首先將感測器模組模組插入ZB2530-01 模組的 IO 擴充埠（J8）和J（11）處，如圖（11）所示。





已插上模組（酒精感測器模組）

圖（11）

其他模組的連接類似，如下圖所示：

人體紅外感測器的連結

溫度感測器的連結

GSensor 三軸位移加速感測器的連結

光電晶體感測器的連結

EEPROM 模組的連結

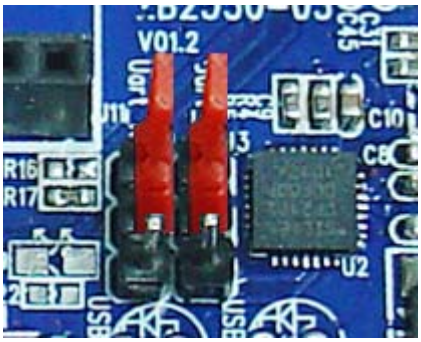
直流馬達模組的連結

酒精感測器的連結

光敏電阻感測器的連結

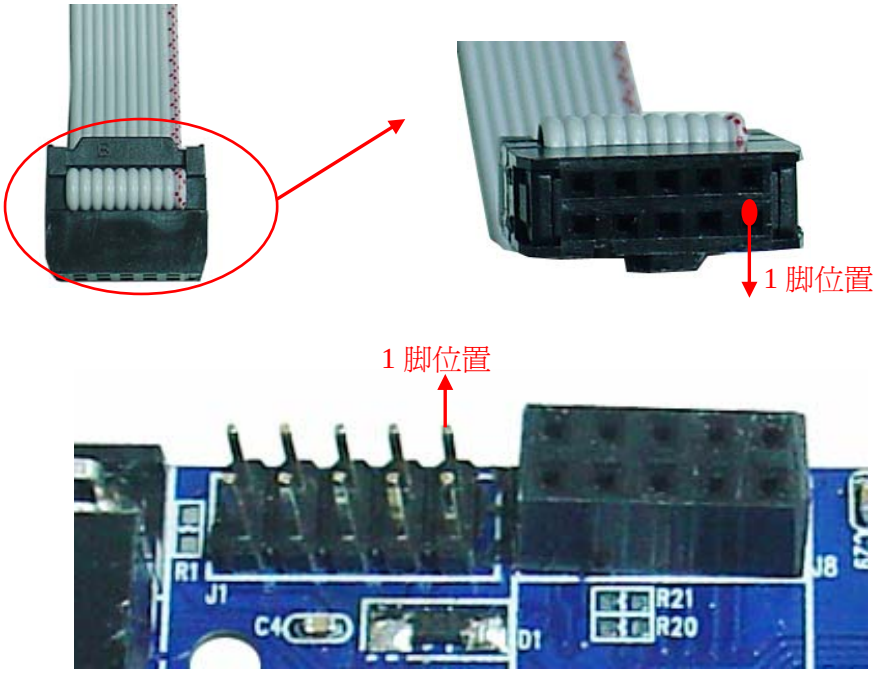
圖（11）

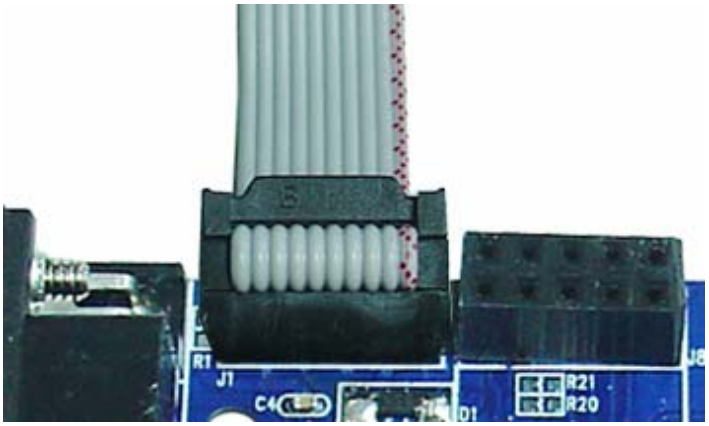
b. 將ZB2530-01 模組的 J2/J3 作如下跳線設置：



圖（12）

c. 將 Debugger 燒寫器的排線連接到 ZB2530-01 模組的 Debugger(J1)插針處，注意插線的腳位順序如圖（13）。





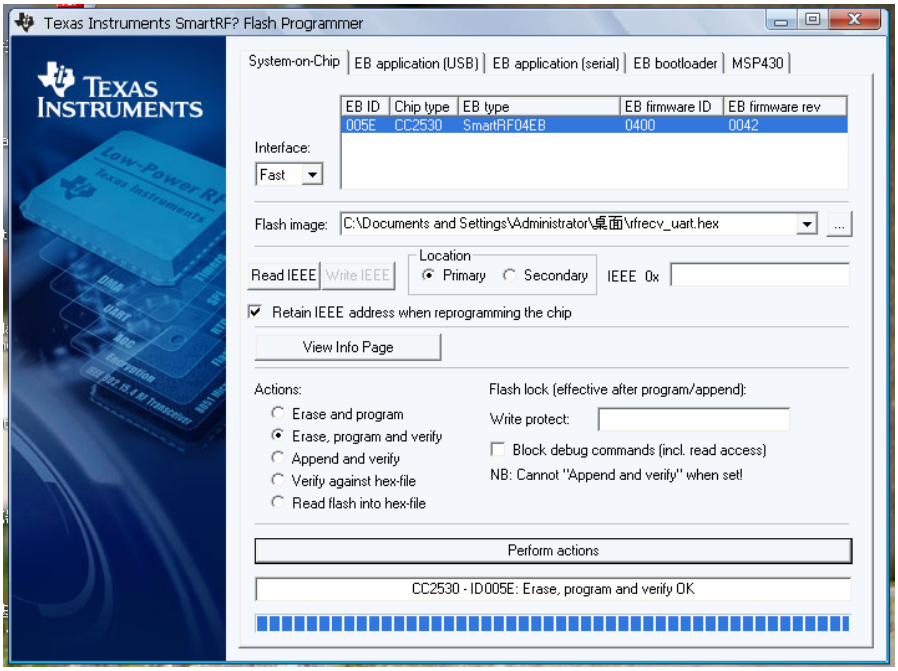
圖（13）

d. 接好後如下圖（14）所示：

圖（14）

e. 將 5V/2A 電源連接到 ZB2530-01 模組的電源插座（J10）處。

f. 透過 Debugger 模擬器將寫好的感測器模組的測試檔（hex 檔）燒錄進 CC2530 晶片內，如圖（15）所示，燒錄完成後將 Debugger 模擬器從 ZB2530-01 模組的 Debugger(J1) 處取下。



圖（15）Flash rpgramming 介面

接收端可以重複發送端的步驟，只需取消步驟a，並將步驟 f 的燒寫檔換成rf_send_recv.hex，最後將ZB2530-01 模組的 COM1 埠透過 DB9公轉Mini 5Pin USB線連接到平台（如 HMI700-6410S）上，如圖（16）。

圖(16) 與 HMI700-6410S 連接

第四章 Android / ZigBee 無線感測器平台圖控軟體介紹

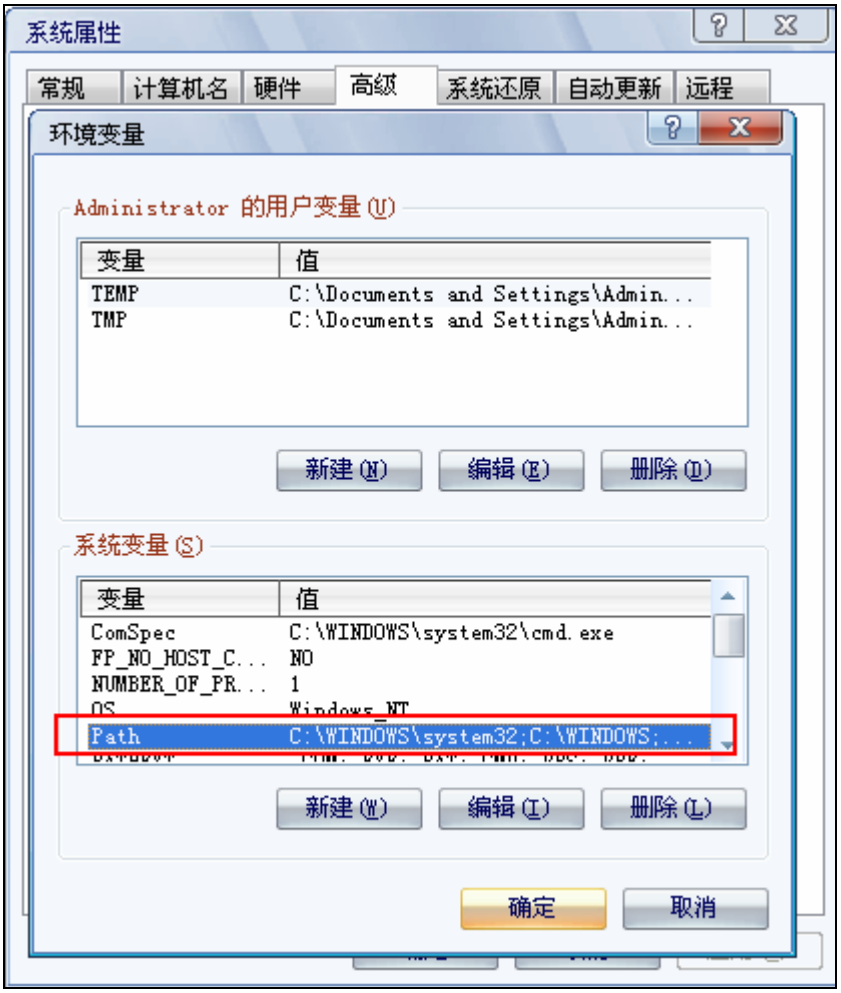
4-1 Windows 環境搭建

4-1.1 安裝 JDK

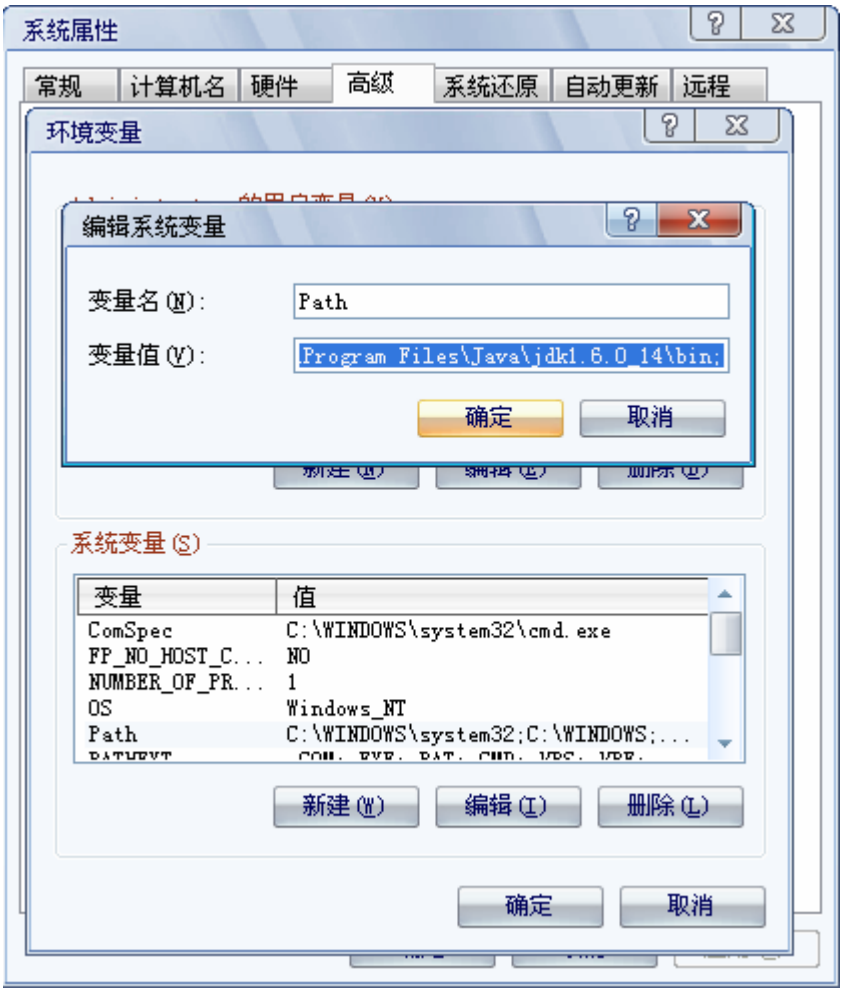
安裝 Eclipse 的開發環境需要 JRE 的支援，如果沒有 JRE，則啓動Eclipse時會報告錯誤。

在 Windows上安裝JRE/JDK 非常簡單，首先在 Sun 官方網站上下載 JDK，網址爲：http://java.sun.com/javase/downloads，下載完後，點兩下JDK安裝檔，打開安裝精靈，然後按照預設的設置進行安裝。預設的情況下JDK安裝路徑爲：C:\Program Files\Java。

安裝完成後還需配置 JDK 的環境變數，將 JDK 的 bin 檔的路徑C:\Program Files\Java\jdk1.6.0_14\bin 添加到 Path 中，右鍵打開“我的電腦”，依次選擇“屬性”->“高級”->“環境變數”選項，選擇“系統變數”中的“PATH“選項，如下圖所示：

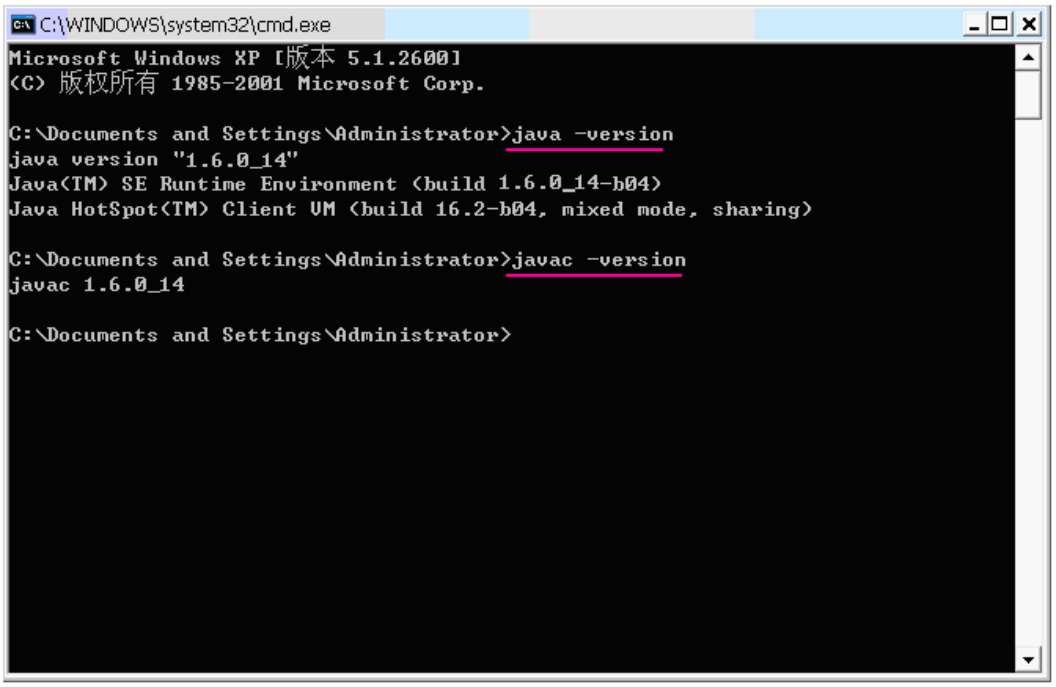


點選編輯按鈕，將 JDK 所在路徑添加進入即可，句尾以分號結尾，如下圖所示：



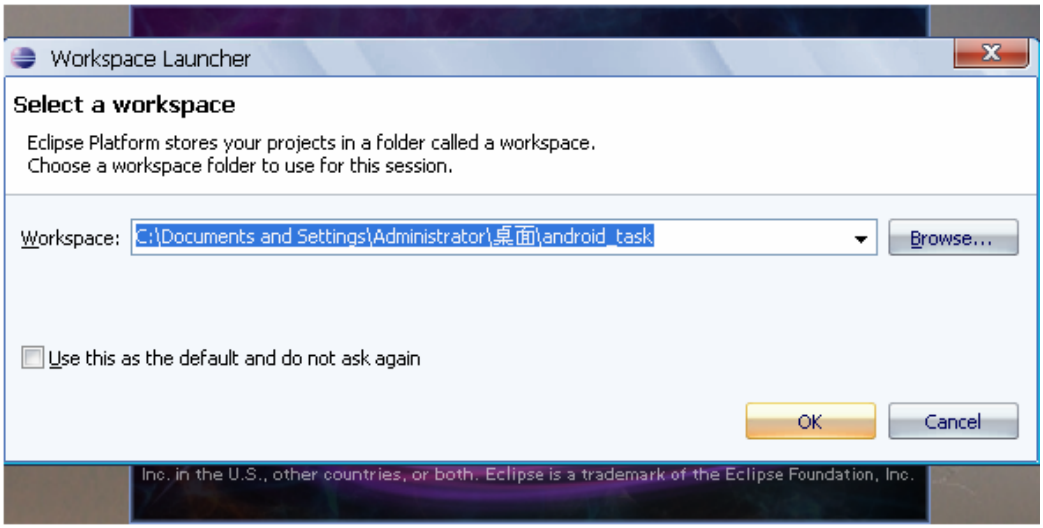
配置完成後，依次點選確定。

配置完成後，打開“開始”，選擇“執行”，在出現的對話方塊中輸入 cmd 命令，打開 cmd 視窗，在視窗輸入命令：java -version，如果出現下圖所示資訊說明 JDK 安裝成功。


C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.
C:\Documents and Settings\Administrator>java -version
java version "1.6.0_14"
Java(TM) SE Runtime Environment (build 1.6.0_14-b04)
Java HotSpot(TM) Client VM (build 16.2-b04, mixed mode, sharing)
C:\Documents and Settings\Administrator>javac -version
javac 1.6.0_14
C:\Documents and Settings\Administrator>

4-1.2 安裝 Eclipse

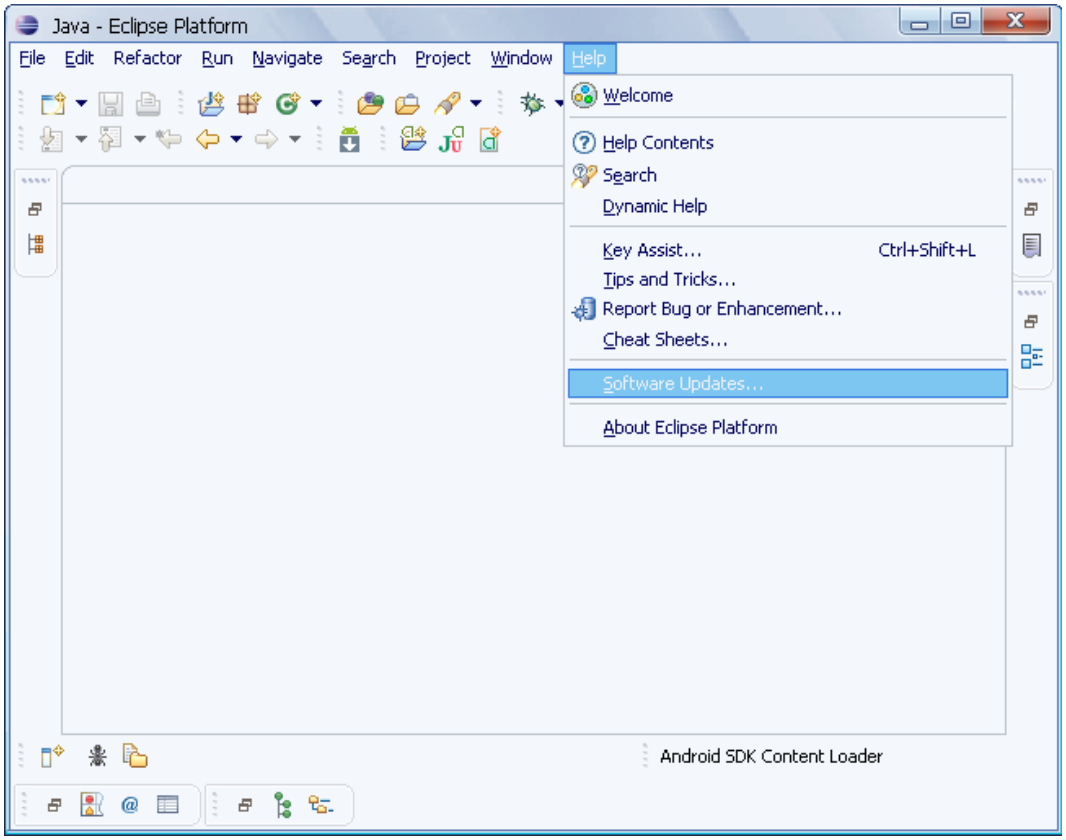
安裝好 JDK 後，就可以安裝 Eclipse 了，打開 Eclipse 下載頁面，網址爲：http://www.eclipse.org/downloads/，下載完成後，解壓下載的壓縮包檔就可以使用。進入解壓目錄，可以看到一個名爲 eclipse.exe 的可執行檔，點兩下此檔直接執行 Eclipse，如果用戶是第一次啓動 Eclipse，將會看到選擇工作空間的提示，如下圖所示：



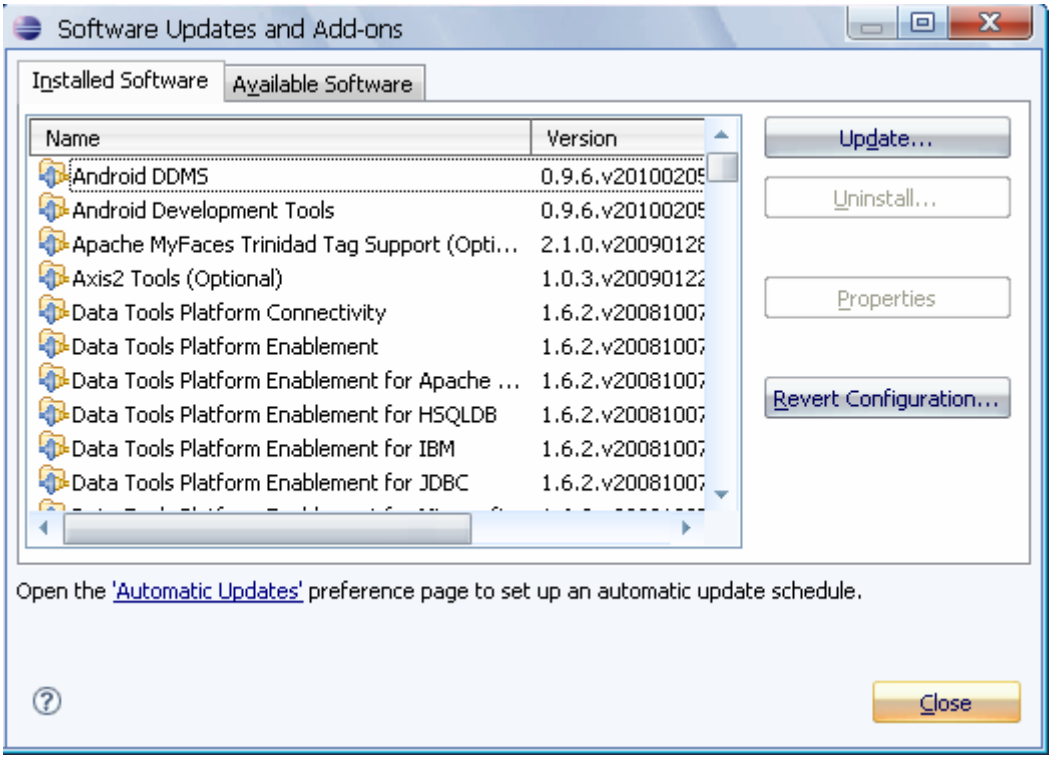
選擇工作空間路徑，然後點選“OK“按鈕即可。

4-1.3 安裝 ADT

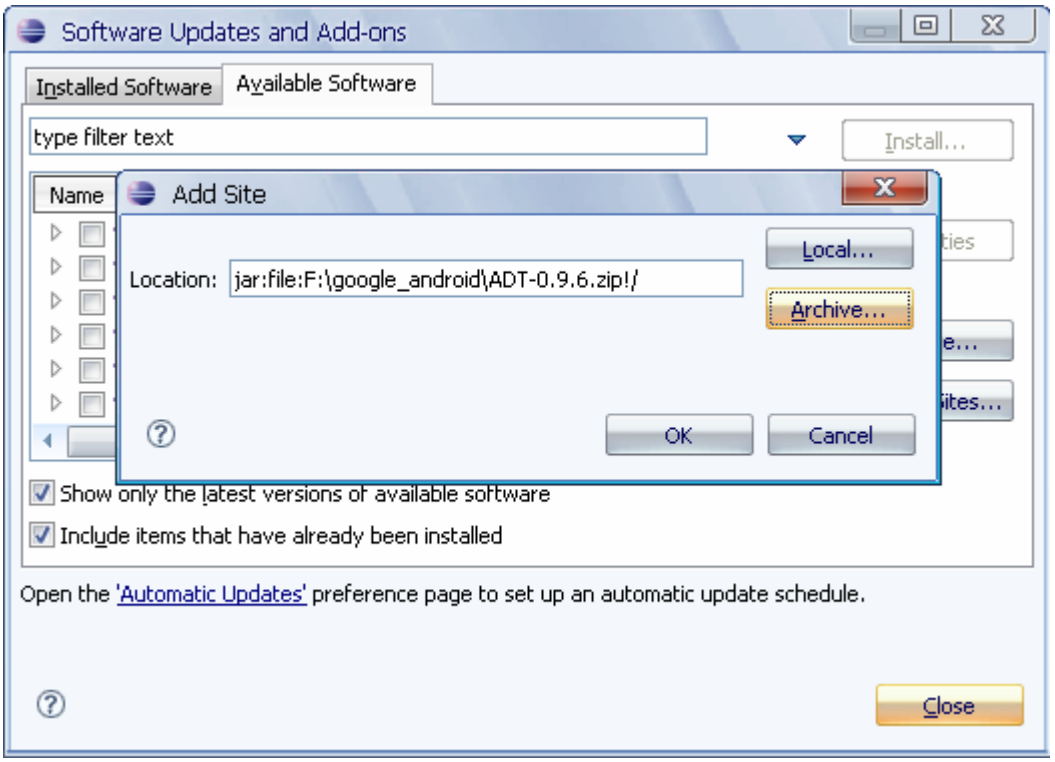
打開 ADT 下載頁面，下載 ADT-0.9.6，網址爲：http://androidappdocs.appspot.com/sdk/eclipse-adt.html。下載完成後，啓動 Eclipse，點選選擇「Help > Software Updates」。



進入如下圖所示對話方塊：

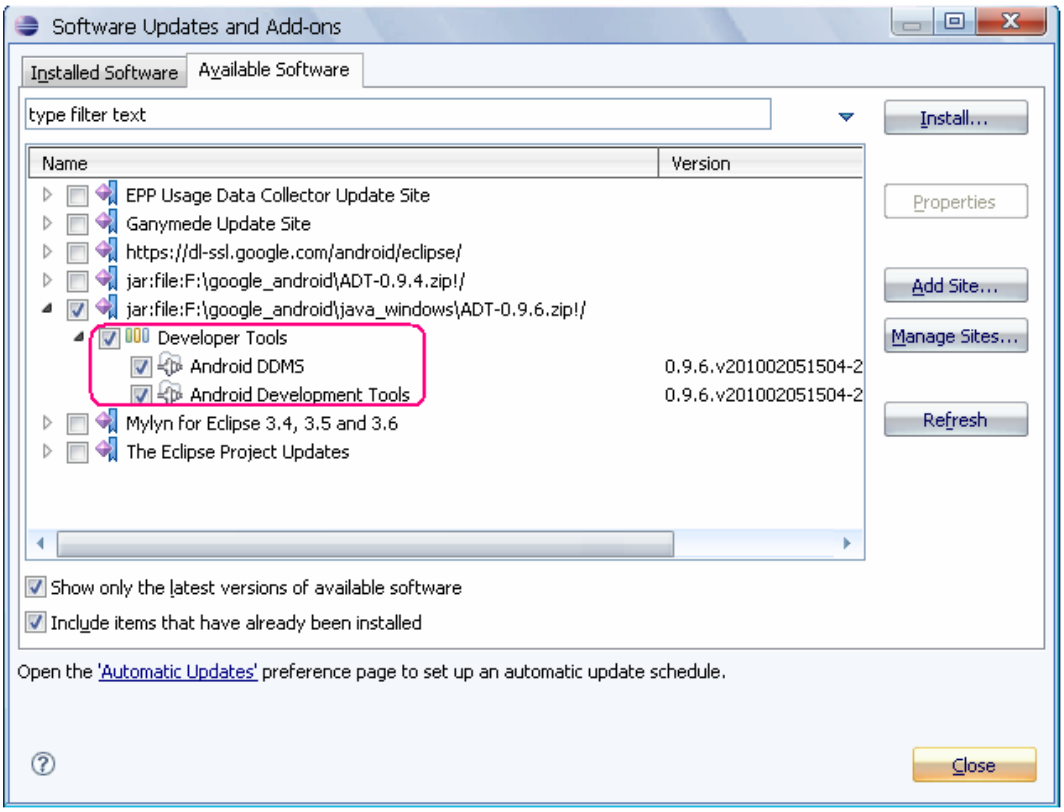


選擇 Available Software 頁面。然後點選 Add Site 按鈕，然後在 Add Site 對話方塊中輸入 ADT 存放的路徑，如下圖所示：



點選"OK"確定。

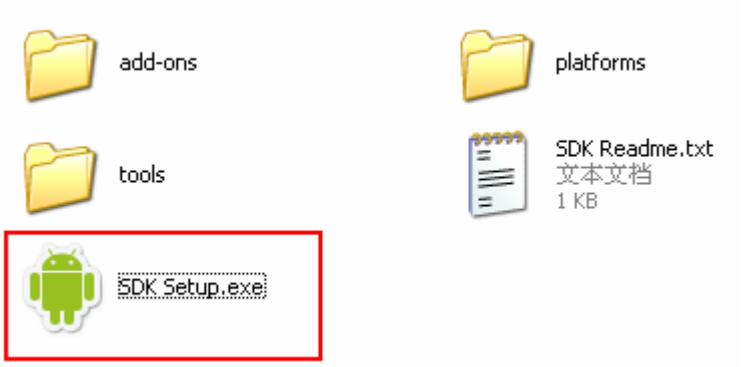
然後勾選住 Developer Tools 選項，點選 Install 安裝。如下圖所示：



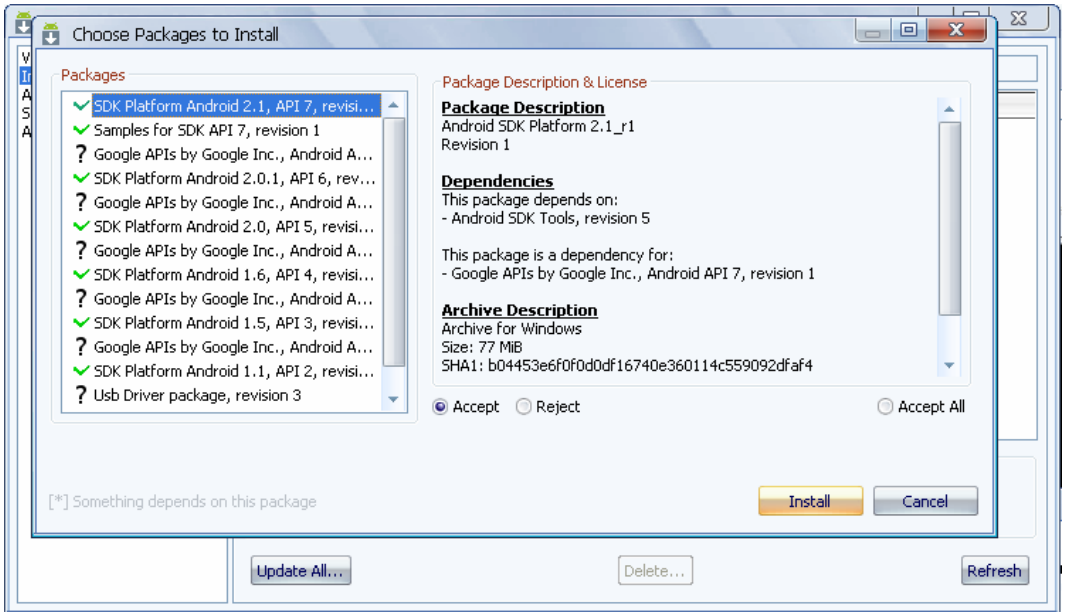
按照預設的設置進行安裝，安裝完成後重新啓動 Eclipse 即可。

4-1.4 安裝 SDK

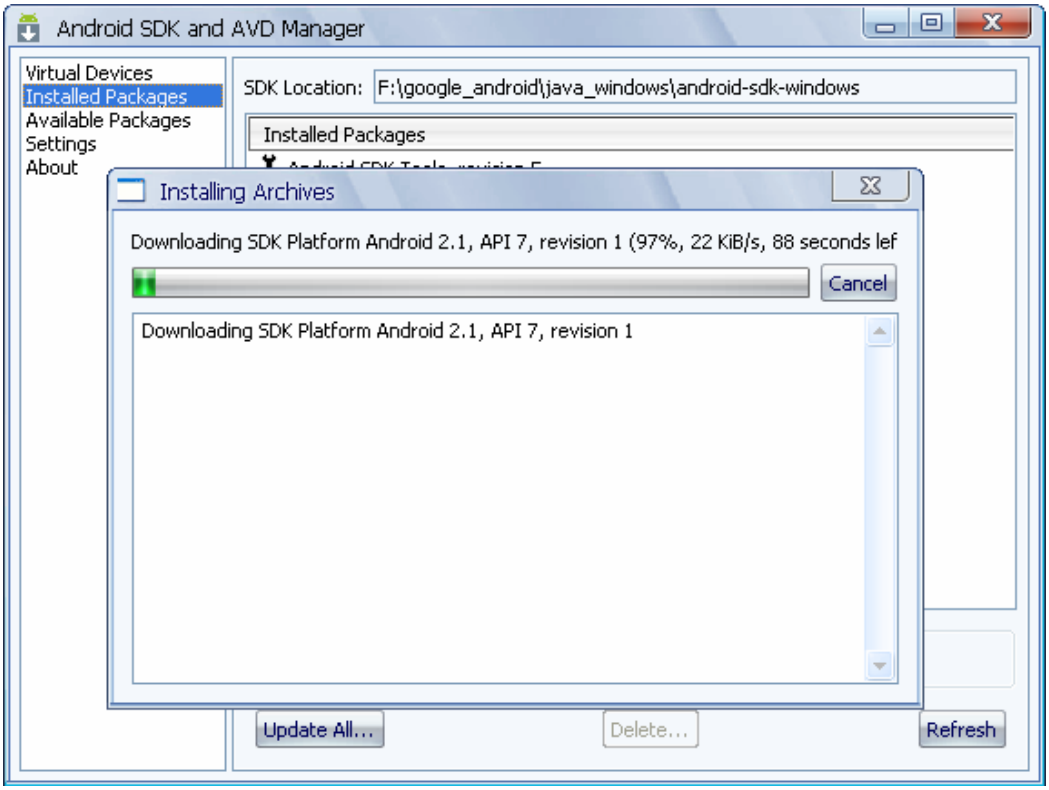
首先得下載 SDK2.1，網址爲 http://androidappdocs.appspot.com/sdk/，下載 Windows 平台下的 SDK 包 android-sdk_r05-windows.zip，解壓後的路徑爲：F:\google_android\java_windows\android-sdk-windows\tools。由於 Android SDK2.1 不再捆綁 platform 和 add-on，因此這兩部分需要手工下載。點選 SDK 裏附帶的"SDK Setup.exe"，如下圖所示：



進入如下圖介面：



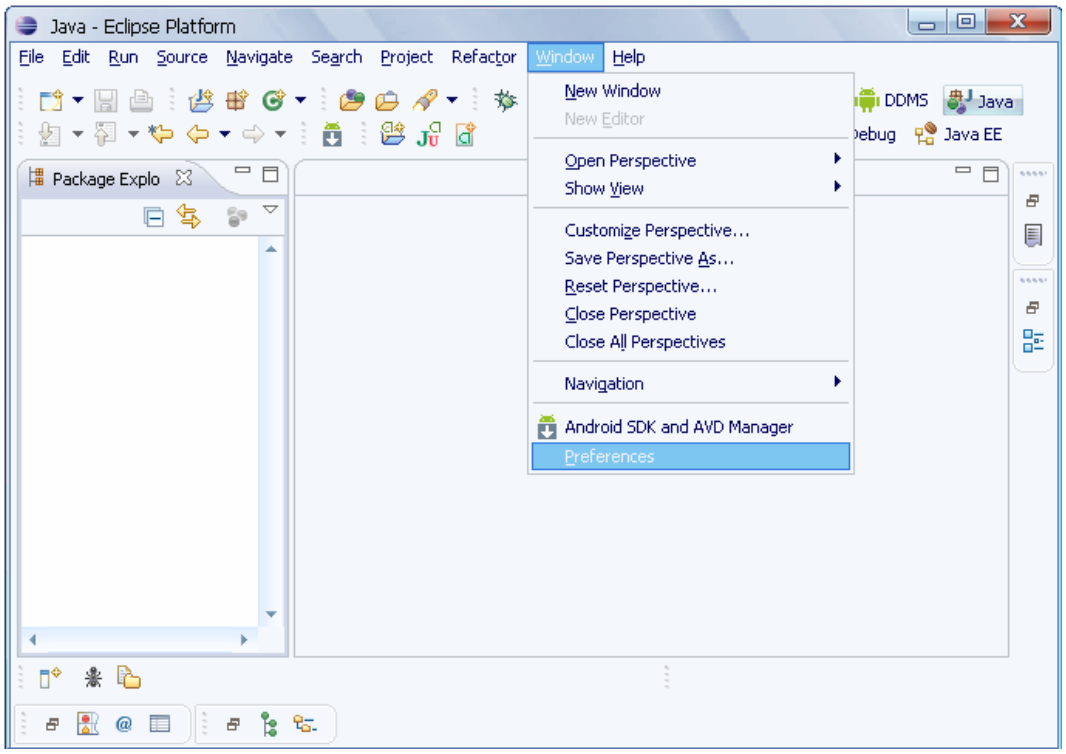
選擇 Accept All，然後點選 Install 即可開始安裝並下載。



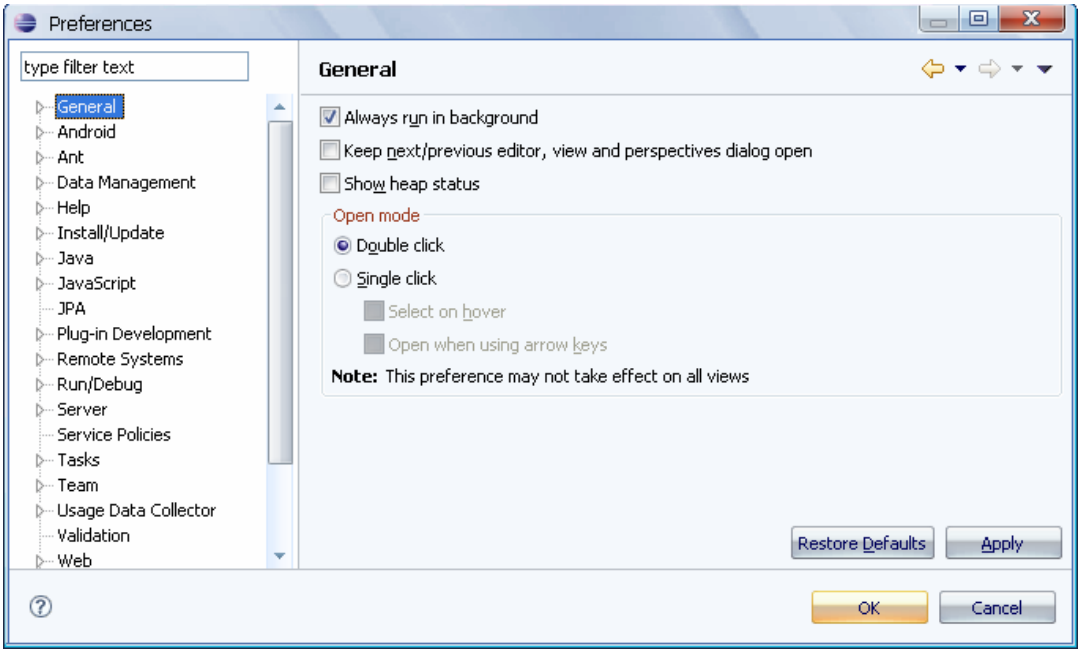
安裝完成後，退出此頁面，然後對 SDK 進行配置。

需要將 Android SDK 安裝目錄中的 tools 檔案夾路徑(F:\google_android\java_windows\android-sdk-windows\tools)添加到環境變數 PATH 中，具體配置過程在此不再講述，請讀者參看 JDK 環境配置過程。

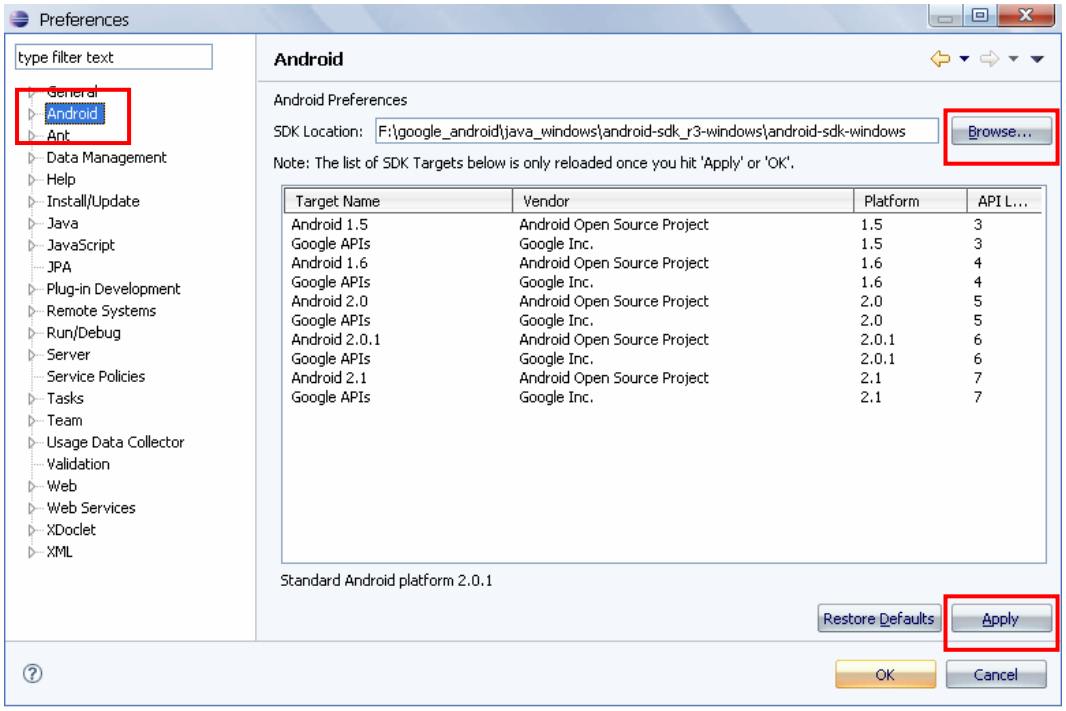
接著重新啓動 Eclipse，並在 Eclipse 的 Preferences 中添加 Android SDK 的路徑。如下圖所示，由 Windows 選單開啓 Preferences 設定。



然後會顯示如下圖所示的對話方塊



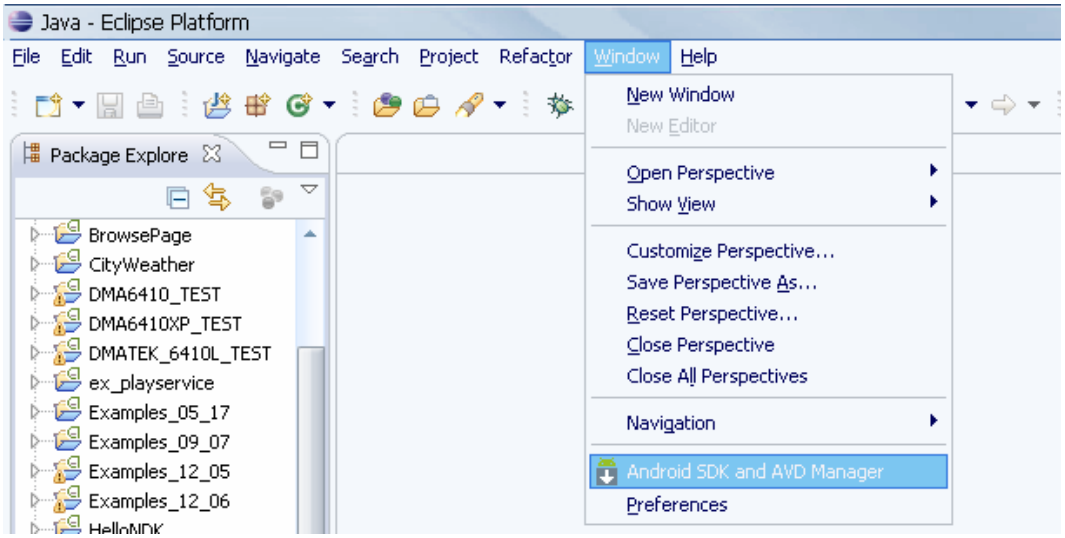
點選 Android，按下 Browse 按鈕並選擇 Android SDK 的路徑。如下圖所示：



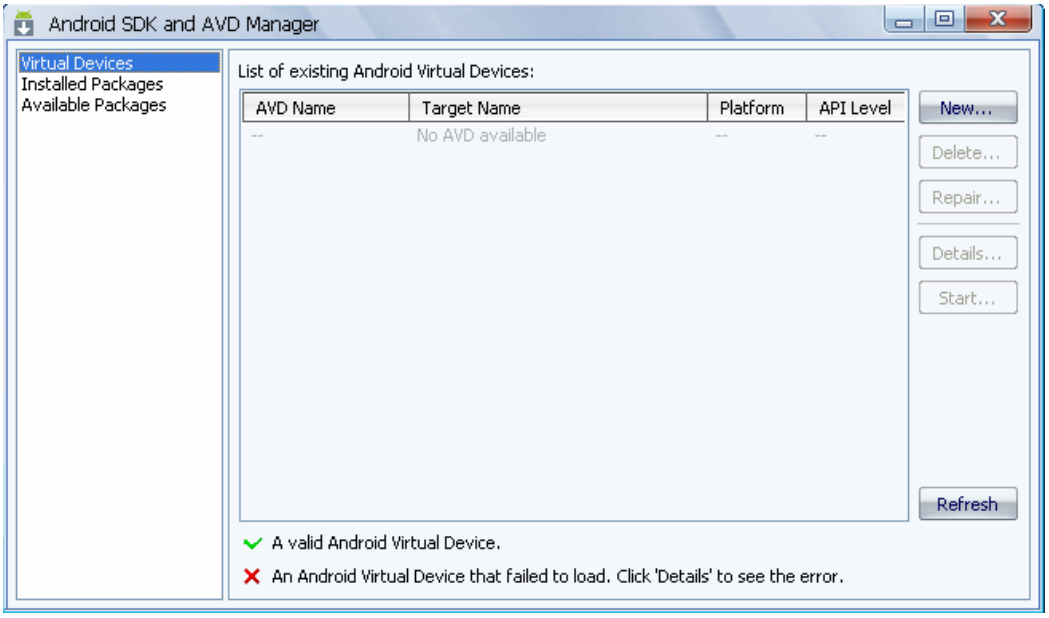
點選“Apply“按鈕進行載入，載入完成後點選“OK“退出。

4-1.5 創建 Android 虛擬設備(AVD)

如下圖所示，依次點選“Window”->“Android SDK and AVD Manager”。

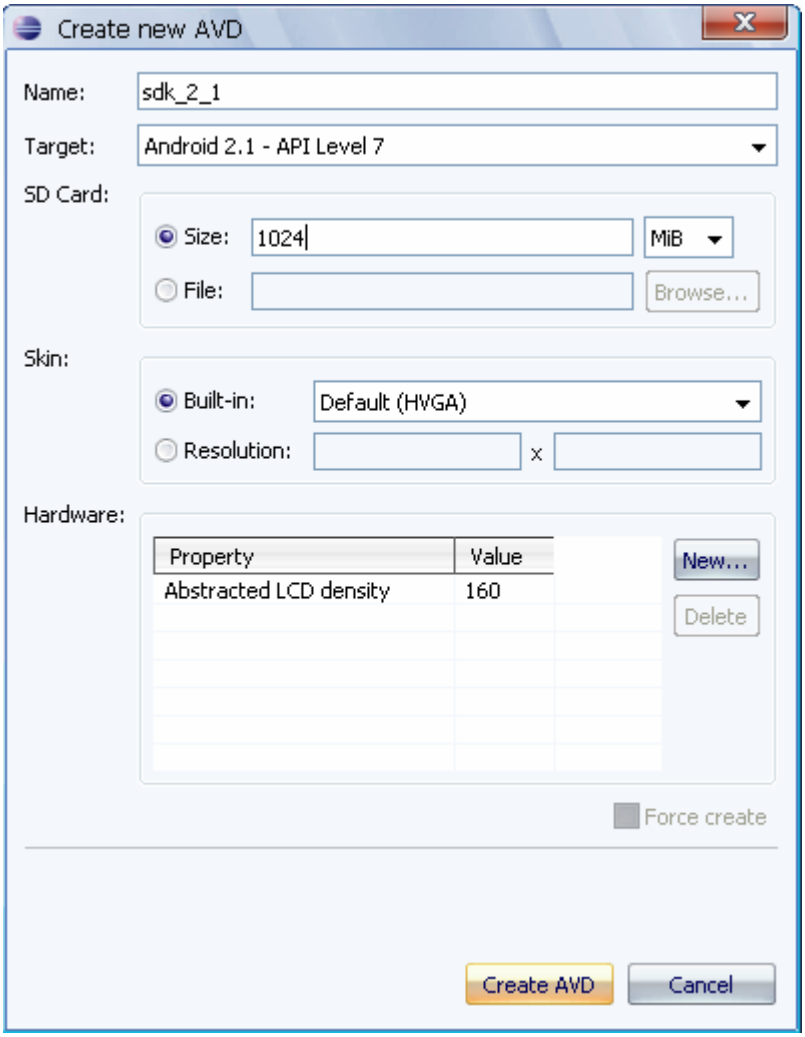


將進入下圖所示介面：



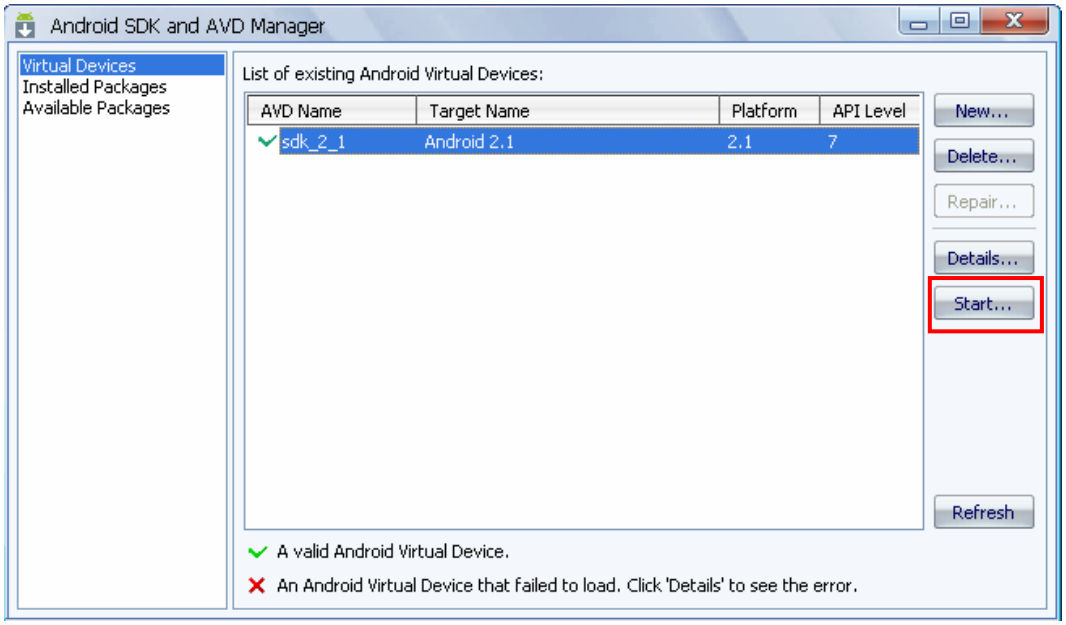
此時無任何 AVD 設備，AVD 虛擬設備是類比了一套虛擬設備來執行 Android 平台，這個平台有自己的核心和資料分區。現在需要創建一個 AVD 設備。點選“New”按鈕。

出現下圖所示對話方塊：

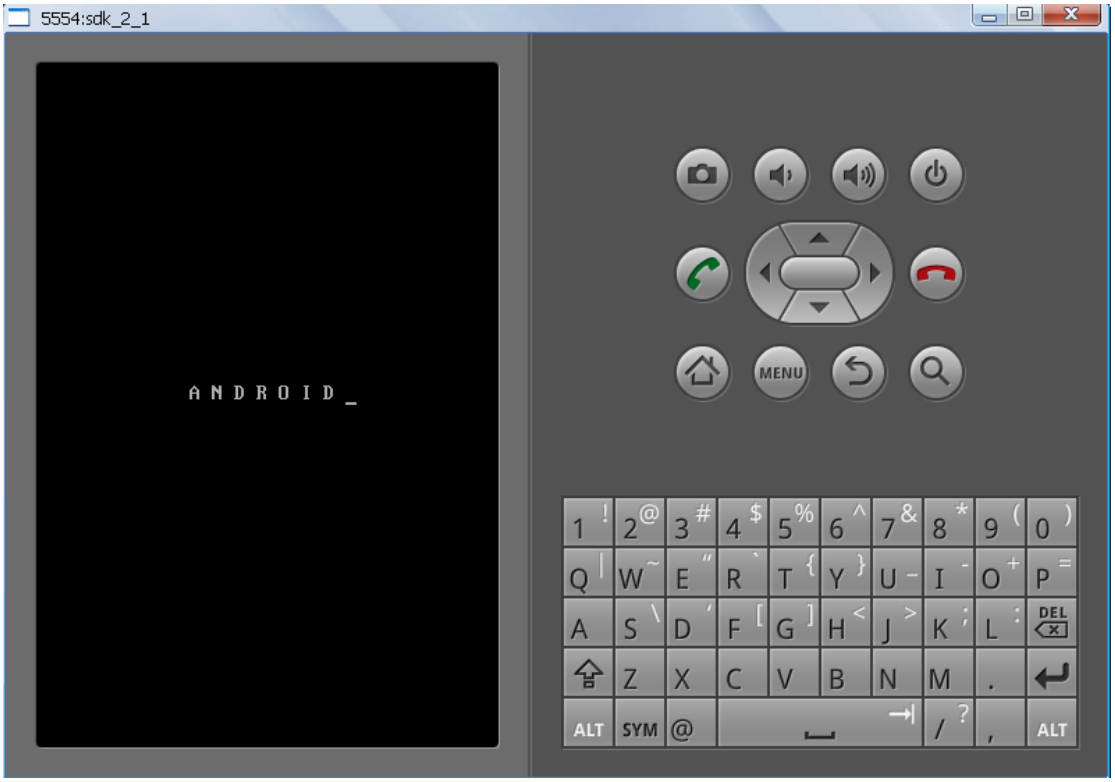


Name 選項爲創建的模擬器的名稱，Target 爲創建模擬器的版本，Size 爲 SD 卡的容量。

在此對話方塊中填入 Name，Target 等選項，創建完畢後點選“Create AVD”按鈕即可。



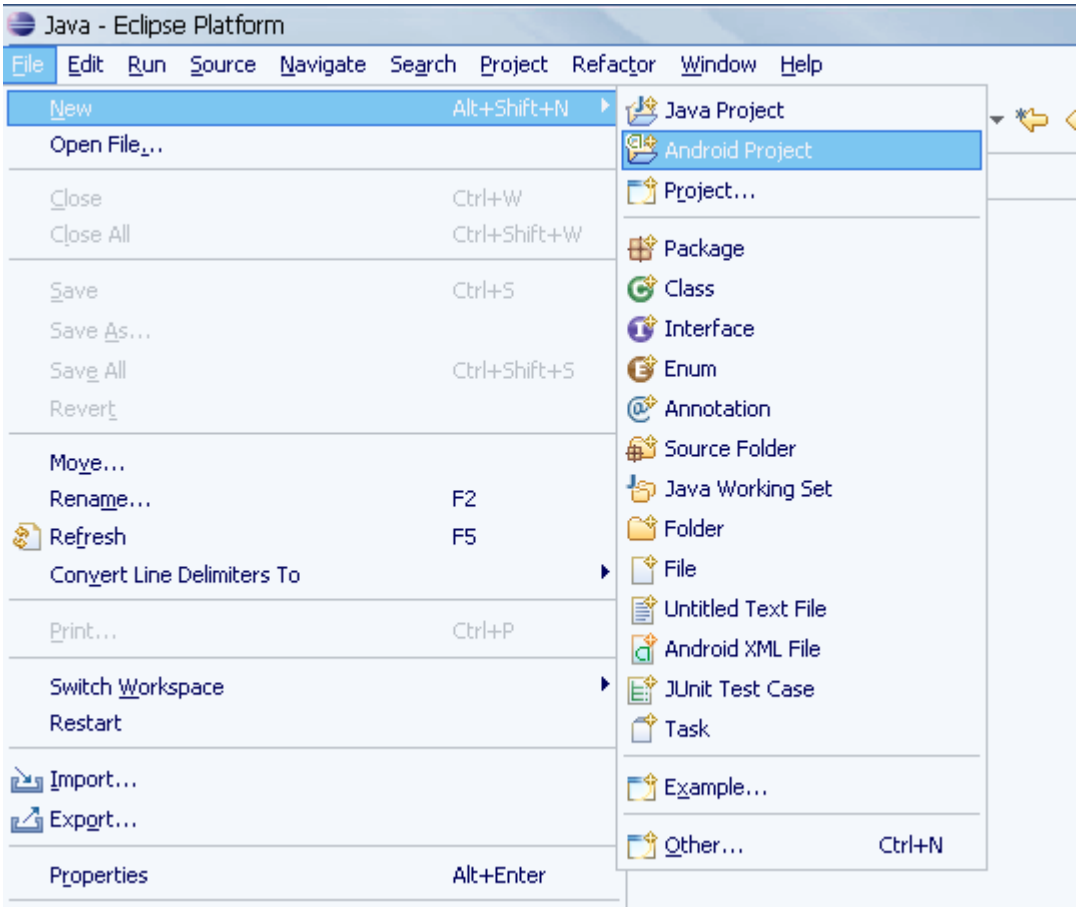
此時虛擬設備創建成功，選中創建的虛擬設備，然後點選“Start“按鈕，啓動模擬器。



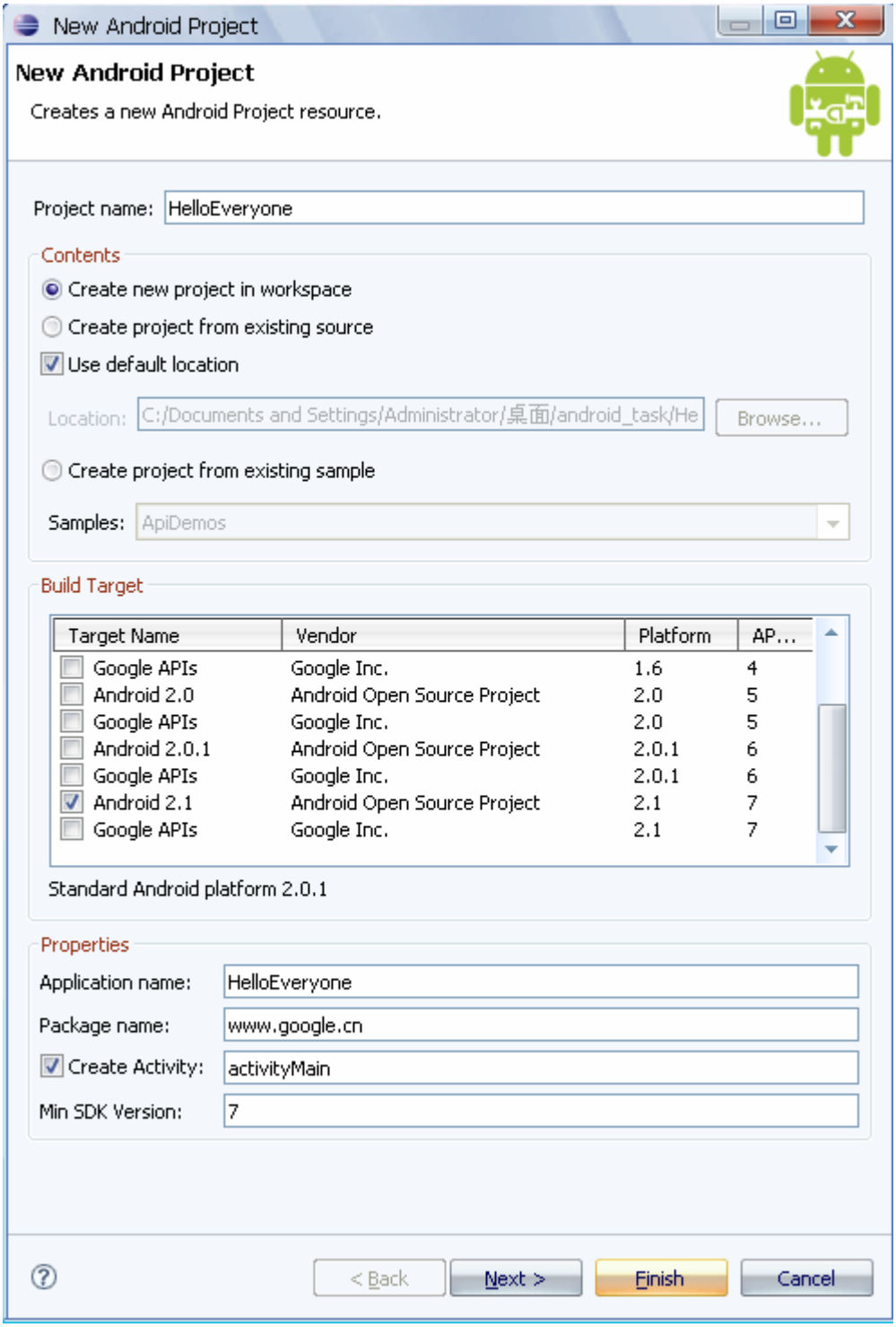
進入上圖所示介面表示模擬器創建成功。

4-1.6 創建第一個 Android 項目 HelloEveryone

打開 Eclipse，依次選擇“File”->“New”->“Android Project”，如下圖所示：

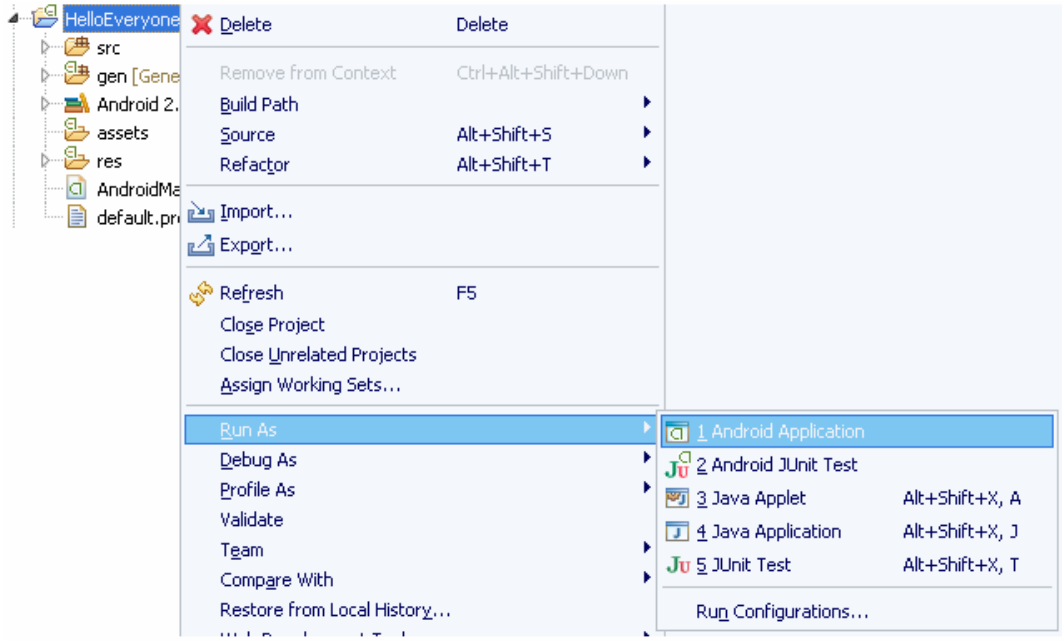


點選 Android Project 後進入下圖所示介面，需要填寫相關工程資訊。下圖是已經填好的工程資訊：

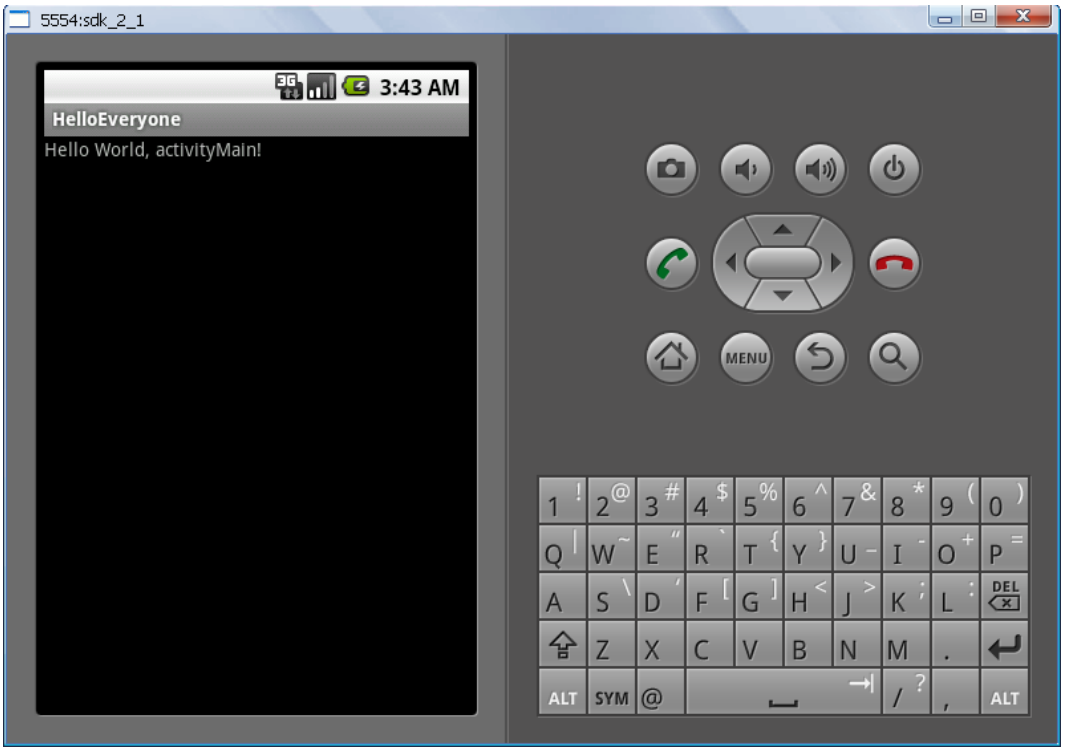


填寫完畢後點選 Finish 按鈕即可。

右鍵打開“HelloEveryone“，依次選擇“Run As”->“Android Application”，啓動模擬器並執行此程式。



模擬器顯示如下表明 Android 開發環境配置成功！


5554:sdk_2_1
3:43 AM
HelloEveryone
Hello World, activityMain!
MENU
ALT
SYM
ALT

4-2 APK 安裝器

APK 安裝器在 Android 系統中起到一個橋樑的作用，透過它可以把外部應用程式植入到 Android 系統內部，同時也可以卸載部分應用程式，大大方便了對應用程式的除錯，所以 APK 安裝器在 Android 系統中具有舉足輕重的作用。下面就介紹一下如何使用 APK 安裝器來安裝應用程式。

點選桌面圖示 

進入如下圖所示介面：

點選“安裝 Installer”將進入如下圖所示的 APK 安裝介面：

選中所需要安裝的 apk 檔進行安裝即可。點選“管理 Manager”鍵進入 apk 卸載介面，選中要卸載的軟體，點選就可以開始卸載了。點選“退出 Exit”按鈕即可退出此應用程式。

4-3 介面介紹

安裝好我們編寫好的APK後，我們現在可以進行各個模組的測試。

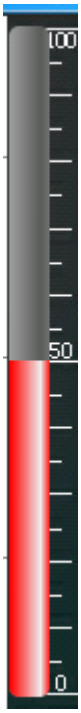
點選如下圖紅框處的APK 圖示：

進入如下圖的主介面：

感測器功能模組：這裏有我們編寫的 10 個感測器模組，當測試每一個模組時，可以選擇進行測試。如下圖所示：

曲線繪製：這是我們繪製曲線區域，在這裏我們繪製出，當您選擇某個模組時對應這個模組電壓值的即時曲線。例如，溫度值、電壓值。如下所示：

光柱圖繪製：這是我們自己編寫的光柱圖，當您選擇某個模組對應的即時值，例如，溫度值、電壓值等，這個值與曲線值對應的。如下圖所示



資料顯示：這是資料顯示區域，當您選擇某個模組時，模組名稱會變成您選擇模組的名稱，下面會出現相關的資料標籤和相應的值，在後面我們會一一看到。如下圖所示：

系統功能按鈕：當您選擇了某個模組時，點選開始，就可以進行資料的採集了，當您要離開這個介面是點選退出按鈕，就可以離開當前介面。如下圖所示：

到此，我們就把這個介面介紹完成了，下面將介紹各個模組如何進行測試，和資料怎樣的顯示。

4-4 感測器測試

接下來介紹下各個感測器的測試及顯示，首先我們進入到下面的介面：

1）光電模組

點選光電模組按鈕，介面出現如下所示：

然後點選開始按鈕，介面如下所示：

曲線繪圖區域處有曲線了，這是我們給的一個初始值，同時光柱圖也有了變化，也是我們給的初始值，在資料顯示區域除了模組名稱，還有三個值，意義如下：

AD 值：這是光電晶體模組返還回來的一個 AD 值。

電壓值：這是光電晶體模組 AD 值對應的電壓值。

光度值：這是對應於某個光強度的光度值，這個我們目前沒有演算法，進行計算。當您插上了光電晶體模組時，您可以使用不同的光源進行測試，觀測曲線、光柱、資料的變化。

2) 光敏模組

點選光敏模組按鈕，介面變成如下所示：

然後點選開始按鈕，介面變成了如下所示：

這時，曲線繪圖區域有曲線了，這是我們給的一個初始值，光柱圖也有了變化，也是我們給的初始值，在資料顯示區域除了模組名稱，還有三個值，意義如下：

AD 值：這是光敏模組返還回來的一個 AD 值。

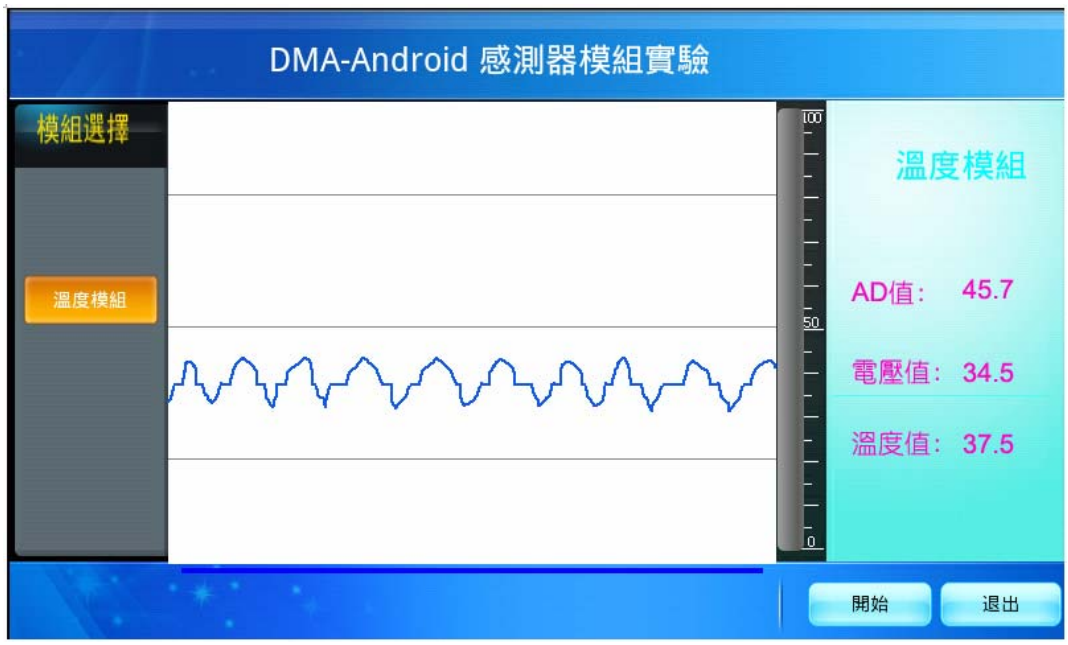
電壓值：這是光敏模組 AD 值對應的電壓值。

光度值：這是對應於某個光強度的光度值，這個我們目前沒有演算法，進行計算。當您插上了光敏模組時，您可以使用不同的光源進行測試，觀察曲線、光柱、資料的變化。

3） 溫度模組

點選溫度模組按鈕，介面變成如下所示：

然後點選開始按鈕，介面變成了如下所示：



此時也有曲線了，這是我們給的一個初始值，光柱圖也有了變化，也是我們給的初始值，在資料顯示區除了模組名稱，還有三個值，意義如下：

AD 值：這是溫度模組（AD590）返還回來的一個 AD 值。

電壓值：這是溫度模組 AD 值對應的電壓值。

溫度值：這是對應於某個溫度下的溫度值，這個值我們已經計算出來了。當您插上了溫度模組時，您可以使用不同的溫度進行測試，觀測曲線、光柱、資料的變化。

4） Gsensor 三軸位移加速感測器模組

點選 GSensor 模組按鈕，介面變成如下所示：

然後點選開始按鈕，出現如下所示介面：

在資料顯示區除了模組名稱，還有三個值，意義如下：

X：這是個 GSensor 感測模組返還回來的 X 軸的資料，有正和負。

Y：這是個 GSensor 感測模組返還回來的 Y 軸的資料，有正和負。

Z：這是個 GSensor 感測模組返還回來的 Z 軸的資料，有正和負。

當您插上了 GSensor 感測器模組時，您可以上下、左右、前後的搖動 BMA023 三軸位移加速感測器模組，觀測這三個值的資料和符號變化。

5） EEPROM 感測器模組

點選 EEPROM 模組按鈕，出現如下所示介面：

然後點選開始按鈕，介面變成了如下圖所示：

在資料顯示區除了模組名稱，還有一個值，意義如下：

輸出：這個值是我們往 EEPROM 內寫入的值，寫入的方法是對 ZB2530-01 模組使用下載器燒錄進去的 hex 檔案，檔案中是對 EEPROM 進行寫入 5 和 7 值，並做了求和，輸出到串列埠上。我們寫入的是 12，當您接好模組進行測試時，輸出爲 12，證明 EEPROM 的讀寫正確。

6） 人體紅外模組

點選人體紅外模組按鈕，出現如下所示介面：

然後點選開始按鈕，介面變成了如下圖所示：

在資料顯示區處除了模組名稱，還有一個值，意義如下：

開關值：當沒有人時，我們可以看到 OFF，表示沒有人；當您使用手測量人體感應模組時，可以看到 OFF 變成了 ON，人體遠離變成了人體接近。

7） 酒精模組

點選酒精模組按鈕，出現如下圖所示介面：

然後點選開始按鈕，介面變成了如下所示：

此時，曲線繪圖區域有曲線了，這是我們給的一個初始值，同時光柱圖也有了變化，也是我們給的初始值，在資料顯示區除了模組名稱，還有三個值，意義如下：

AD 值：這是酒精模組返還回來的一個 AD 值。

電壓值：這是光敏模組 AD 值對應的電壓值。

酒精值：這是對應於某個酒精濃度下的酒精值，這個我們目前沒有演算法，進行計算。當您插上了酒精模組時，您可以使用不同的酒精濃度進行測試，觀測曲線、光柱、資料的變化。(提示，也可以使用打火機的汽體進行測試)

8) 溫湿度模組

點選溫濕度模組按鈕，出現如下圖所示介面：

然後點選開始按鈕，介面變成了如下所示：

由於這裡有兩個模組，所以就沒有在顯示其他曲線及光柱了。溫度值是溫濕度模組中溫度模組採集的數據，濕度值是溫濕度模組中濕度模組採集的數據。

9） 濕度模組

點選濕度模組按鈕，介面變成如下所示：

然後點選開始按鈕，介面變成了如下所示：

此時也有曲線了，這是我們給的一個初始值，光柱圖也有了變化，也是我們給的初始值，在資料顯示區除了模組名稱，還有三個值，意義如下：

濕度值：這是對應於某個溫度下的溫度值，這個值我們已經計算出來了。當您插上了濕度模組時，您可以使用不同的濕度進行測試，觀測曲線、光柱、資料的變化。

10）陀螺儀感測器模組

點選陀螺儀模組按鈕，介面變成如下所示：

然後點選開始按鈕，出現如下所示介面：

在資料顯示區除了模組名稱，還有三個值，意義如下：

X：這是個陀螺儀感測模組返還回來的 X 軸的資料，沒有正和負。

Y：這是個陀螺儀感測模組返還回來的 Y 軸的資料，沒有正和負。

Z：這是個陀螺儀感測模組返還回來的 Z 軸的資料，沒有正和負。

當您插上了陀螺儀感測器模組時，您可以上下、左右、前後的搖動陀螺儀三軸位移加速感測器模組，觀測這三個值的資料和符號變化。

11）空氣品質感測器模組

點選空氣模組按鈕，介面變成如下所示：

然後點選開始按鈕，出現如下所示介面：

在資料顯示區除了模組名稱，還有一個值，意義如下：

空氣污染等級：空氣污染等級有三個值，分別如下

空氣清晰：表示當前的空氣清晰，以基準值（第一次開始檢測到的值）偏離的絕對值小於等於 2 的值。

一般污染：表示當前的空氣一般污染，以基準值（第一次開始檢測到的值）偏離的絕對值大於 2 的值小於等於 4 的值。

嚴重污染：表示當前的空氣嚴重污染，以基準值（第一次開始檢測到的值）偏離的絕對值大於 4 的值。

注意：當您測試完一個模組，沒有退出時，開始按鈕與剛打開的不同時，測量模組可以不要點選開始按鈕了。

4-5 ZB2530-01 LED DEMO 實驗

硬體連線

將我們的 HMI700-6410S 平台上的 Mini 5Pin 串列埠 CON2 與 ZB2530-01 模組連接，如下圖所示，並接上電源，如下圖所示。

實驗

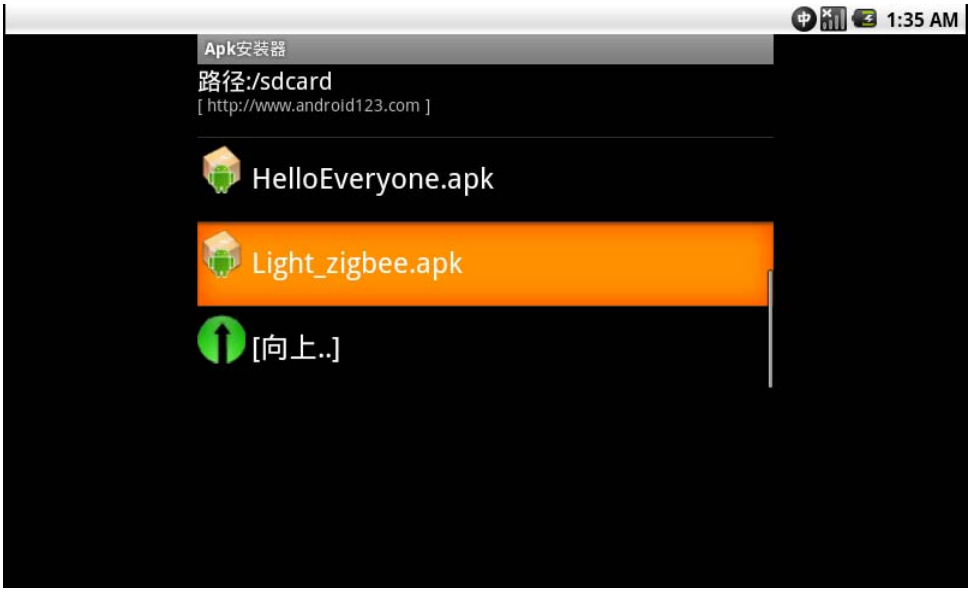
一、 使用 APK 安裝器安裝我們編寫好的程式

APK 安裝器在 Android 系統中起到一個橋樑的作用，透過它可以把外部應用程式植入到 Android 系統內部，同時也可以卸載部分應用程式，大大方便了對應用程式的除錯，所以 APK 安裝器在 Android 系統中具有舉足輕重的作用。下面就介紹一下如何使用 APK 安裝器來安裝應用程式。

點選桌面圖示 

進入如下圖所示介面：

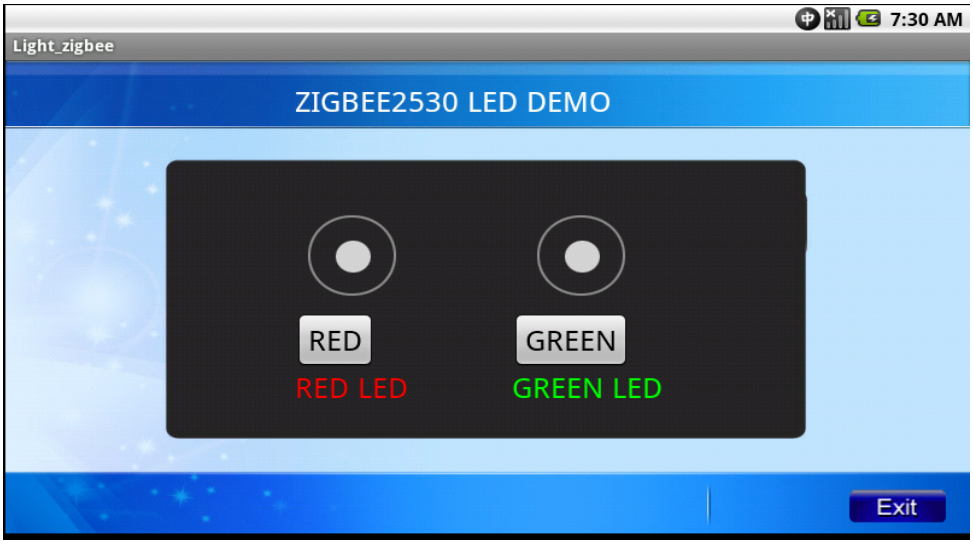
點選“安裝 Installer”將進入如下圖所示的 APK 安裝介面：



選中所需要安裝的apk檔進行安裝即可。點選“管理 Manager”按鍵進入apk卸載介面，選中要卸載的軟體，點選就可以開始卸載了。點選“退出 Exit”按鈕即可退出此應用程式。

二、 進行實驗

打開安裝好的APK，如下圖：



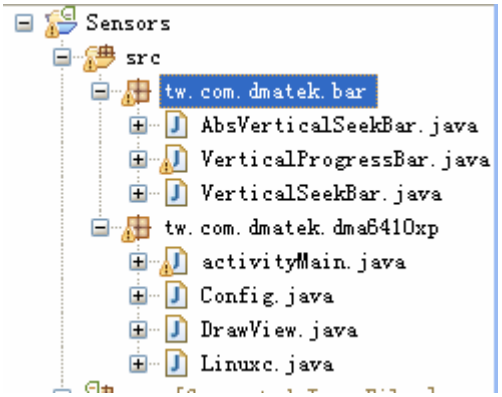
1. 我們先來測試左邊的紅燈，您點擊左邊的紅燈按鈕，這時您可以看到左邊的紅燈在不停的閃爍，同時您觀測 ZigBee 另一端的紅燈，也在不停的閃爍。再點擊時，介面的紅燈不在閃爍，ZigBee 另一端的也停止閃爍。

2. 現在測試右邊的綠燈，點擊綠燈按鈕，綠燈亮，同時另一端 ZigBee 上的綠燈也亮，再點擊綠燈，綠燈滅，同時另一端的綠燈也滅。

如此，這個實驗展示完成了。

4-6 程式碼編譯說明

感測器的 Java 程式碼結構如下圖所示：



其中有兩個包，一個是對 progressBar 和 seekBar 進行的重寫的包，由於 android 下的 progressBar 和 seekBar 是水準的，並且設置都很不方便，爲此，在這裏對這兩個類進行了重構。這樣我們就可以設計出我們自己想要的 progressBar 和 seekBar 了；另一個是感測器模組的主介面與曲線類以及調用 SO 本地方法的 Linuxc.Java 的包。

下面我會對每一個檔進行分析：

先從第一個包開始，裏面有三個 Java 檔，AbsVerticalSeekBar.java、VerticalProgressBar.java、VerticalSeekBar.java 這三個類之間有一定的繼承關係。VerticalProgressBar 是父類，AbsVerticalSeekBar 繼承之，VerticalSeekBar 又繼承 AbsVerticalSeekBar。我們先從 VerticalProgressBar 這個類來看看。

打開 VerticalProgressBar.Java，程式碼如下：

```
package tw.com.dmatek.bar;
import tw.com.dmatek.dma6410xp.R;
import android.content.Context;
import android.content.res.TypedArray;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Rect;
```

```
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.ClipDrawable;
import android.graphics.drawable.Drawable;
import android.graphics.drawable.LayerDrawable;
import android.graphics.drawable.ShapeDrawable;
import android.graphics.drawable.StateListDrawable;
import android.graphics.drawable.shapes.RoundRectShape;
import android.graphics.drawable.shapes.Shape;
import android.util.AttributeSet;
import android.view.Gravity;
import android.view.View;
import android.view.ViewDebug;
import android.view.ViewParent;
import android.widget.RemoteViews.RemoteView;
import android.os.Parcel;
import android.os.Parcelable;
/********
```

以上是所使用到的包，東西很多酒不一一說明，主要就說說

`import android.view.View;`

View是很多元件的父類，這個類主要用於顯示我們可以看到的東西，空間，下面我把android下定義如下This class represents the basic building block for user interface components。這是用戶介面元件的基礎，換句話就是使用到元件的話都一般繼續這個類，當然也有少許直接寫的**********/

```
@RemoteView
public class VerticalProgressBar extends View {
    private static final int MAX_LEVEL = 10000;

    int mMinWidth;
```

```
  int mMaxWidth;
  int mMinHeight;
  int mMaxHeight;

  private int mProgress;
  private int mSecondaryProgress;
  private int mMax;

  private Drawable mProgressDrawable;
  private Drawable mCurrentDrawable;
  Bitmap mSampleTile;
  private boolean mNoInvalidate;
  private RefreshProgressRunnable mRefreshProgressRunnable;
  private long mUiThreadId;

  private boolean mInDrawing;
  protected int mScrollX;
   protected int mScrollY;
   protected int mPaddingLeft;
   protected int mPaddingRight;
   protected int mPaddingTop;
   protected int mPaddingBottom;
   protected ViewParent mParent;

/************以上是定義的一些變數*********/

  public VerticalProgressBar(Context context)
   {
   this(context, null);
```

```
    }

    public VerticalProgressBar(Context context, AttributeSet
attrs)
    {
        this(context, attrs, android.R.attr.progressBarStyle);
    }

    public VerticalProgressBar(Context context, AttributeSet
attrs, int defStyle)
     {
        super(context, attrs, defStyle);
        mUiThreadId = Thread.currentThread().getId();
        initProgressBar();
/*******以上是這個類的重構部分,根據不同的參數就行重構******/
        TypedArray a =
            context.obtainStyledAttributes(attrs,
R.styleable.ProgressBar, defStyle, 0);
        mNoInvalidate = true;

        Drawable drawable =
a.getDrawable(R.styleable.ProgressBar_android_progressDrawab
le);
        if (drawable != null)
{
            drawable = tileify(drawable, false);
            setProgressDrawable(drawable);
        }
```

```
        mMinWidth =
a.getDimensionPixelSize(R.styleable.ProgressBar_android_minW
idth, mMinWidth);
        mMaxWidth =
a.getDimensionPixelSize(R.styleable.ProgressBar_android_maxW
idth, mMaxWidth);
        mMinHeight =
a.getDimensionPixelSize(R.styleable.ProgressBar_android_minH
eight, mMinHeight);
        mMaxHeight =
a.getDimensionPixelSize(R.styleable.ProgressBar_android_maxH
eight, mMaxHeight);

setMax(a.getInt(R.styleable.ProgressBar_android_max, mMax));
setProgress(a.getInt(R.styleable.ProgressBar_android_progres
s, mProgress));
        setSecondaryProgress(
a.getInt(R.styleable.ProgressBar_android_secondaryProgress,
mSecondaryProgress));
        mNoInvalidate = false;
        a.recycle();
    }

    private Drawable tileify(Drawable drawable, boolean clip) {

        if (drawable instanceof LayerDrawable) {
         LayerDrawable background = (LayerDrawable) drawable;
            final int N = background.getNumberOfLayers();
            Drawable[] outDrawables = new Drawable[N];
```

```
            for (int i = 0; i < N; i++) {
                int id = background.getId(i);
                outDrawables[i] =
tileify(background.getDrawable(i),
                (id == android.R.id.progress || id ==
android.R.id.secondaryProgress));
            }

            LayerDrawable newBg = new
LayerDrawable(outDrawables);

            for (int i = 0; i < N; i++) {
                newBg.setId(i, background.getId(i));
            }
            return newBg;
        } else if (drawable instanceof StateListDrawable) {
            StateListDrawable in = (StateListDrawable) drawable;
            StateListDrawable out = new StateListDrawable();
            return out;

        } else if (drawable instanceof BitmapDrawable) {
            final Bitmap tileBitmap = ((BitmapDrawable)
drawable).getBitmap();
            if (mSampleTile == null) {
                mSampleTile = tileBitmap;
            }

            final ShapeDrawable shapeDrawable = new
```

```
ShapeDrawable(getDrawableShape());
            return (clip) ? new ClipDrawable(shapeDrawable,
Gravity.LEFT,
            ClipDrawable.HORIZONTAL) : shapeDrawable;
        }
        return drawable;
    }

    Shape getDrawableShape() {
        final float[] roundedCorners = new float[] { 5, 5, 5, 5,
5, 5, 5, 5 };
        return new RoundRectShape(roundedCorners, null, null);
    }

    private void initProgressBar() {
        mMax = 100;
        mProgress = 0;
        mSecondaryProgress = 0;
        mMinWidth = 24;
        mMaxWidth = 48;
        mMinHeight = 24;
        mMaxHeight = 48;
    }

    public Drawable getProgressDrawable() {
        return mProgressDrawable;
    }

    public void setProgressDrawable(Drawable d) {
```

```
        if (d != null) {
            d.setCallback(this);
            int drawableHeight = d.getMinimumHeight();
            if (mMaxHeight < drawableHeight) {
                mMaxHeight = drawableHeight;
                requestLayout();
            }
        }
        mProgressDrawable = d;
        mCurrentDrawable = d;
        postInvalidate();
    }

    Drawable getCurrentDrawable() {
        return mCurrentDrawable;
    }

    @Override
    protected boolean verifyDrawable(Drawable who) {
        return who == mProgressDrawable ||
super.verifyDrawable(who);
    }

    @Override
    public void postInvalidate() {
        if (!mNoInvalidate) {
            super.postInvalidate();
        }
    }
```

```
    private class RefreshProgressRunnable implements Runnable {

        private int mId;
        private int mProgress;
        private boolean mFromUser;

        RefreshProgressRunnable(int id, int progress, boolean
fromUser) {
            mId = id;
            mProgress = progress;
            mFromUser = fromUser;
        }

        public void run() {
            doRefreshProgress(mId, mProgress, mFromUser);
            mRefreshProgressRunnable = this;
        }

        public void setup(int id, int progress, boolean fromUser)
{
            mId = id;
            mProgress = progress;
            mFromUser = fromUser;
        }
    }

    private synchronized void doRefreshProgress(int id, int
progress, boolean fromUser) {
```

```
  float scale = mMax > 0 ? (float) progress / (float) mMax : 0;
        final Drawable d = mCurrentDrawable;
        if (d != null) {
            Drawable progressDrawable = null;

            if (d instanceof LayerDrawable) {
                progressDrawable = ((LayerDrawable)
d).findDrawableByLayerId(id);
            }
            final int level = (int) (scale * MAX_LEVEL);
            (progressDrawable != null ? progressDrawable :
d).setLevel(level);
        } else {
            invalidate();
        }

        if (id == android.R.id.progress) {
            onProgressRefresh(scale, fromUser);
        }
    }

    void onProgressRefresh(float scale, boolean fromUser) {
    }

    private synchronized void refreshProgress(int id, int
progress, boolean fromUser) {
        if (mUiThreadId == Thread.currentThread().getId()) {
            doRefreshProgress(id, progress, fromUser);
        } else {
```

```
            RefreshProgressRunnable r;
            if (mRefreshProgressRunnable != null) {
                r = mRefreshProgressRunnable;
                mRefreshProgressRunnable = null;
                r.setup(id, progress, fromUser);
            } else {
                r = new RefreshProgressRunnable(id, progress,
fromUser);
            }
            post(r);
        }
    }

    public synchronized void setProgress(int progress) {
        setProgress(progress, false);
    }

    synchronized void setProgress(int progress, boolean
fromUser) {
        if (progress < 0) {
            progress = 0;
        }

        if (progress > mMax) {
            progress = mMax;
        }

        if (progress != mProgress) {
            mProgress = progress;
```

```
            refreshProgress(android.R.id.progress, mProgress,
fromUser);
        }
    }

    public synchronized void setSecondaryProgress(int
secondaryProgress) {
        if (secondaryProgress < 0) {
            secondaryProgress = 0;
        }

        if (secondaryProgress > mMax) {
            secondaryProgress = mMax;
        }

        if (secondaryProgress != mSecondaryProgress) {
            mSecondaryProgress = secondaryProgress;
            refreshProgress(android.R.id.secondaryProgress,
mSecondaryProgress, false);
        }
    }

    @ViewDebug.ExportedProperty
    public synchronized int getProgress() {
        return mProgress;
    }

    @ViewDebug.ExportedProperty
    public synchronized int getSecondaryProgress() {
```

```
        return mSecondaryProgress;
    }

    @ViewDebug.ExportedProperty
    public synchronized int getMax() {
        return mMax;
    }

    public synchronized void setMax(int max) {
        if (max < 0) {
            max = 0;
        }
        if (max != mMax) {
            mMax = max;
            postInvalidate();

            if (mProgress > max) {
                mProgress = max;
                refreshProgress(android.R.id.progress,
mProgress, false);
            }
        }
    }

    public synchronized final void incrementProgressBy(int diff)
{
        setProgress(mProgress + diff);
    }
```

```
    public synchronized final void
incrementSecondaryProgressBy(int diff) {
        setSecondaryProgress(mSecondaryProgress + diff);
    }

    @Override
    public void setVisibility(int v) {
        if (getVisibility() != v) {
            super.setVisibility(v);
        }
    }

    @Override
    public void invalidateDrawable(Drawable dr) {
        if (!mInDrawing) {
            if (verifyDrawable(dr)) {
                final Rect dirty = dr.getBounds();
                final int scrollX = mScrollX + mPaddingLeft;
                final int scrollY = mScrollY + mPaddingTop;

        invalidate(dirty.left + scrollX, dirty.top + scrollY,
        dirty.right + scrollX, dirty.bottom + scrollY);
            } else {
                super.invalidateDrawable(dr);
            }
        }
    }

    @Override
```

```
    protected void onSizeChanged(int w, int h, int oldw, int oldh)
{
        int right = w - mPaddingRight - mPaddingLeft;
        int bottom = h - mPaddingBottom - mPaddingTop;

        if (mProgressDrawable != null) {
            mProgressDrawable.setBounds(0, 0, right, bottom);
        }
    }

    @Override
    protected synchronized void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        Drawable d = mCurrentDrawable;
        if (d != null) {
            canvas.save();
            canvas.translate(mPaddingLeft, mPaddingTop);
            d.draw(canvas);
            canvas.restore();
        }
    }

    @Override
    protected synchronized void onMeasure(int widthMeasureSpec,
int heightMeasureSpec) {
        Drawable d = mCurrentDrawable;
        int dw = 0;
        int dh = 0;
```

```
        if (d != null) {
            dw = Math.max(mMinWidth, Math.min(mMaxWidth,
d.getIntrinsicWidth()));
  dh = Math.max(mMinHeight, Math.min(mMaxHeight,
d.getIntrinsicHeight()));
        }
        dw += mPaddingLeft + mPaddingRight;
        dh += mPaddingTop + mPaddingBottom;

        setMeasuredDimension(resolveSize(dw,
widthMeasureSpec),
        resolveSize(dh, heightMeasureSpec));
    }

    @Override
    protected void drawableStateChanged() {
        super.drawableStateChanged();

        int[] state = getDrawableState();

        if (mProgressDrawable != null &&
mProgressDrawable.isStateful()) {
            mProgressDrawable.setState(state);
        }
    }

    static class SavedState extends BaseSavedState {
        int progress;
        int secondaryProgress;
```

```
        SavedState(Parcelable superState) {
            super(superState);
        }

        private SavedState(Parcel in) {
            super(in);
            progress = in.readInt();
            secondaryProgress = in.readInt();
        }

        @Override
        public void writeToParcel(Parcel out, int flags) {
            super.writeToParcel(out, flags);
            out.writeInt(progress);
            out.writeInt(secondaryProgress);
        }

        public static final Parcelable.Creator<SavedState>
CREATOR
                = new Parcelable.Creator<SavedState>() {
            public SavedState createFromParcel(Parcel in) {
                return new SavedState(in);
            }

            public SavedState[] newArray(int size) {
                return new SavedState[size];
            }
        };
```

```
    }

    @Override
    public Parcelable onSaveInstanceState() {
        Parcelable superState = super.onSaveInstanceState();
        SavedState ss = new SavedState(superState);

        ss.progress = mProgress;
        ss.secondaryProgress = mSecondaryProgress;
        return ss;
    }

    @Override
    public void onRestoreInstanceState(Parcelable state) {
        SavedState ss = (SavedState) state;
        super.onRestoreInstanceState(ss.getSuperState());

        setProgress(ss.progress);
        setSecondaryProgress(ss.secondaryProgress);
    }
}
/**********以上部分就是存儲初始化的第一進度和第二進度**********/
```

程式碼比較長不能一個個介紹到，有興趣的可以對 Android 下的 processBar 進行瞭解，再在只是對我們使用到的進行了重構，增加自己需要的功能。

現在我們來看看第二個 java 檔：AbsVerticalSeekBar

```
package tw.com.dmatek.bar;

import tw.com.dmatek.dma6410xp.R;
import android.content.Context;
```

```
import android.content.res.TypedArray;
import android.graphics.Canvas;
import android.graphics.Rect;
import android.graphics.drawable.Drawable;
import android.util.AttributeSet;
import android.view.KeyEvent;
import android.view.MotionEvent;
/**
這個類直接繼承了VerticalProgressBar，也就是說它已具體了
VerticalProgressBar的屬性 **/
public class AbsVerticalSeekBar extends VerticalProgressBar{

    private Drawable mThumb;
    private int mThumbOffset;
    float mTouchProgressOffset;
    boolean mIsUserSeekable = true;
    private int mKeyProgressIncrement = 1;
    private static final int NO_ALPHA = 0xFF;
    private float mDisabledAlpha;
    public AbsVerticalSeekBar(Context context) {
        super(context);
    }

    public AbsVerticalSeekBar(Context context, AttributeSet
attrs) {
        super(context, attrs);
    }

    public AbsVerticalSeekBar(Context context, AttributeSet
```

```
attrs, int defStyle) {  super(context, attrs, defStyle);

       TypedArray a = context.obtainStyledAttributes(attrs,
              R.styleable.SeekBar, defStyle, 0);
       Drawable thumb =
a.getDrawable(R.styleable.SeekBar_android_thumb);
       setThumb(thumb);
       int thumbOffset =
a.getDimensionPixelOffset(R.styleable.SeekBar_android_thumbO
ffset, getThumbOffset());
       setThumbOffset(thumbOffset);
       a.recycle();

       a = context.obtainStyledAttributes(attrs,
              R.styleable.Theme, 0, 0);
       mDisabledAlpha =
a.getFloat(R.styleable.Theme_android_disabledAlpha, 0.5f);
       a.recycle();
   }

   public void setThumb(Drawable thumb) {
       if (thumb != null) {
           thumb.setCallback(this);
           mThumbOffset = (int)thumb.getIntrinsicHeight() / 2;
       }
       mThumb = thumb;
       invalidate();
   }
```

```
    public int getThumbOffset() {
        return mThumbOffset;
    }

    public void setThumbOffset(int thumbOffset) {
        mThumbOffset = thumbOffset;
        invalidate();
    }

    public void setKeyProgressIncrement(int increment) {
        mKeyProgressIncrement = increment < 0 ? -increment :
increment;
    }

    public int getKeyProgressIncrement() {
        return mKeyProgressIncrement;
    }

    @Override
    public synchronized void setMax(int max) {
        super.setMax(max);

        if ((mKeyProgressIncrement == 0) || (getMax() /
mKeyProgressIncrement > 20)) {
            setKeyProgressIncrement(Math.max(1,
Math.round((float) getMax() / 20)));
        }
    }
```

```
    @Override
    protected boolean verifyDrawable(Drawable who) {
        return who == mThumb || super.verifyDrawable(who);
    }

    @Override
    protected void drawableStateChanged() {
        super.drawableStateChanged();

        Drawable progressDrawable = getProgressDrawable();
        if (progressDrawable != null) {
        progressDrawable.setAlpha(isEnabled()？NO_ALPHA：(int)
(NO_ALPHA * mDisabledAlpha));
        }

        if (mThumb != null && mThumb.isStateful()) {
            int[] state = getDrawableState();
            mThumb.setState(state);
        }
    }

    @Override
    void onProgressRefresh(float scale, boolean fromUser) {
        Drawable thumb = mThumb;
        if (thumb != null) {
            setThumbPos(getHeight(), thumb, scale,
Integer.MIN_VALUE);
            invalidate();
        }
```

```
    }

    @Override
    protected void onSizeChanged(int w, int h, int oldw, int oldh)
{
        Drawable d = getCurrentDrawable();
        Drawable thumb = mThumb;
        int thumbWidth = thumb == null ? 0 :
thumb.getIntrinsicWidth();
        int trackWidth = Math.min(mMaxWidth, w - mPaddingRight
- mPaddingLeft);

        int max = getMax();
        float scale = max > 0 ? (float) getProgress() / (float)
max : 0;
      if (thumbWidth > trackWidth) {
       int gapForCenteringTrack = (thumbWidth - trackWidth) / 2;
          if (thumb != null) {
              setThumbPos(h, thumb, scale,
gapForCenteringTrack * -1);
          }
          if (d != null) {
              d.setBounds(gapForCenteringTrack, 0,
                      w - mPaddingRight - mPaddingLeft -
gapForCenteringTrack,
                      h - mPaddingBottom - mPaddingTop);
          }
      } else {
          if (d != null) {
```

```
            d.setBounds(0, 0, w - mPaddingRight - mPaddingLeft,
h - mPaddingBottom - mPaddingTop);
        }
        int gap = (trackWidth - thumbWidth) / 2;
        if (thumb != null) {
            setThumbPos(h, thumb, scale, gap);
        }
    }
}

    private void setThumbPos(int h, Drawable thumb, float scale,
int gap) {
        int available = h - mPaddingTop - mPaddingBottom;
        int thumbWidth = thumb.getIntrinsicWidth();
        int thumbHeight = thumb.getIntrinsicHeight();
        available -= thumbHeight;
        available += mThumbOffset * 2;
        int thumbPos = (int) ((1-scale) * available);
        int leftBound, rightBound;
        if (gap == Integer.MIN_VALUE) {
         Rect oldBounds = thumb.getBounds();
            leftBound = oldBounds.left;
            rightBound = oldBounds.right;
        } else {
            leftBound = gap;
            rightBound = gap + thumbWidth;
        }
        thumb.setBounds(leftBound, thumbPos, rightBound,
thumbPos + thumbHeight);
```

```
    }

    @Override
    protected synchronized void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        if (mThumb != null) {
            canvas.save();

            canvas.translate(mPaddingLeft, mPaddingTop -
mThumbOffset);
            mThumb.draw(canvas);
            canvas.restore();
        }
    }

    @Override
    protected synchronized void onMeasure(int widthMeasureSpec,
int heightMeasureSpec) {
        Drawable d = getCurrentDrawable();

        int thumbWidth = mThumb == null ? 0 :
mThumb.getIntrinsicWidth();
        int dw = 0;
        int dh = 0;
        if (d != null) {
            dw = Math.max(mMinWidth, Math.min(mMaxWidth,
d.getIntrinsicWidth()));
            dw = Math.max(thumbWidth, dh);
            dh = Math.max(mMinHeight, Math.min(mMaxHeight,
```

```
d.getIntrinsicHeight()));
        }
        dw += mPaddingLeft + mPaddingRight;
        dh += mPaddingTop + mPaddingBottom;

        setMeasuredDimension(resolveSize(dw,
widthMeasureSpec),
        resolveSize(dh, heightMeasureSpec));
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if (!mIsUserSeekable || !isEnabled()) {
            return false;
        }
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                setPressed(true);
                onStartTrackingTouch();
                trackTouchEvent(event);
                break;

            case MotionEvent.ACTION_MOVE:
                trackTouchEvent(event);
                attemptClaimDrag();
                break;

            case MotionEvent.ACTION_UP:
                trackTouchEvent(event);
```

```
                onStopTrackingTouch();
                setPressed(false);
                invalidate();
                break;

            case MotionEvent.ACTION_CANCEL:
                onStopTrackingTouch();
                setPressed(false);
                invalidate();
                break;
        }
        return true;
    }

    private void trackTouchEvent(MotionEvent event) {
        final int height = getHeight();
        final int available = height - mPaddingTop -
mPaddingBottom;
        int y = height - (int)event.getY();
        float scale;
        float progress = 0;
        if (y < mPaddingBottom) {
            scale = 0.0f;
        } else if (y > height - mPaddingTop) {
            scale = 1.0f;
        } else {
            scale = (float)(y - mPaddingBottom) /
(float)available;
            progress = mTouchProgressOffset;
```

```
    }

    final int max = getMax();
    progress += scale * max;

    setProgress((int) progress, true);
}

private void attemptClaimDrag() {
    if (mParent != null) {
        mParent.requestDisallowInterceptTouchEvent(true);
    }
}

void onStartTrackingTouch() {
}
void onStopTrackingTouch() {
}
void onKeyChange() {
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    int progress = getProgress();

    switch (keyCode) {
        case KeyEvent.KEYCODE_DPAD_DOWN:
            if (progress <= 0) break;
            setProgress(progress - mKeyProgressIncrement,
```

```
true);
                onKeyChange();
                return true;

            case KeyEvent.KEYCODE_DPAD_UP:
                if (progress >= getMax()) break;
                setProgress(progress + mKeyProgressIncrement,
true);
                onKeyChange();
                return true;
        }

        return super.onKeyDown(keyCode, event);
    }

}
```

現在我們來看看第三個 java 檔：AbsVerticalSeekBar

```
package tw.com.dmatek.bar;
import android.content.Context;
import android.util.AttributeSet;

/**
 *
 *
 */
public class VerticalSeekBar extends AbsVerticalSeekBar {
public interface OnSeekBarChangeListener {
void onProgressChanged(VerticalSeekBar seekBar, int
```

```
progress,boolean fromUser);
void onStartTrackingTouch(VerticalSeekBar seekBar);
void onStopTrackingTouch(VerticalSeekBar seekBar);
    }
private OnSeekBarChangeListener mOnSeekBarChangeListener;
public VerticalSeekBar(Context context) {
this(context, null);
}
public VerticalSeekBar(Context context, AttributeSet attrs) {
this(context, attrs, android.R.attr.seekBarStyle);
}

public VerticalSeekBar(Context context, AttributeSet attrs, int
defStyle) {
super(context, attrs, defStyle);
}

@Override
void onProgressRefresh(float scale, boolean fromUser) {
super.onProgressRefresh(scale, fromUser);
if (mOnSeekBarChangeListener != null) {
mOnSeekBarChangeListener.onProgressChanged(this,
getProgress(),
fromUser);
}
}

public void
setOnSeekBarChangeListener(OnSeekBarChangeListener l) {
```

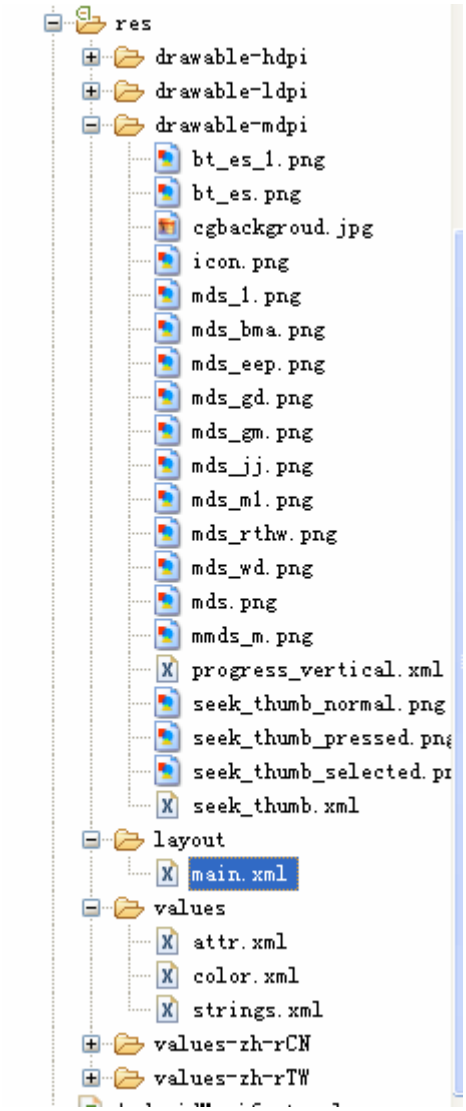
```
mOnSeekBarChangeListener = l;
}

@Override
void onStartTrackingTouch() {
if (mOnSeekBarChangeListener != null) {
mOnSeekBarChangeListener.onStartTrackingTouch(this);
}
}

@Override
void onStopTrackingTouch() {
if (mOnSeekBarChangeListener != null) {
mOnSeekBarChangeListener.onStopTrackingTouch(this);
}
}

}
/****這個類又繼承了AbsVerticalSeekBar，它的功能就更加完善了，可以進行定位****/
```

到此我們就把第一個包的 java 檔介紹完了，其主要功能就是定義出我們自己想要的進度條。

現在我們來介紹我們主要使用到的另一個包。

我們先來看看最主要的類及開始執行起來的介面的 java 程式：activityMain.java, 這個程式碼我會詳細的介紹給您如何設計的，由於它繼承了 Activity 這個機動程式類，所以我們有必要先介紹下它的 XMl 的構造。

先介紹 xml 會使用到的源結構如下所示：



由於我們使用的大解析度的圖片，所以把圖片都放在了 drawable-mdpi 下面，現在讓我來說說我們這個機動程式使用到的 xml 檔 main.xml，其程式碼如下

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
```

```
android:background="@drawable/cgbackgroud"
>

<tw.com.dmatek.dma6410xp.DrawView
 android:id="@+id/drawview"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 />

 <com.clb.barapp.VerticalProgressBar
        android:id="@+id/ProgressBar01"

 android:progressDrawable="@drawable/progress_vertical"
        android:layout_height="335dip"
        android:layout_width="20dip"
        android:layout_marginLeft="581px"
        android:layout_marginTop="75px"
         />

         <TextView
  android:id="@+id/textview0"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textColor="#FFffffff"
android:textSize="25dip"
android:text="@string/app_table"
android:layout_marginLeft="200dip"
android:layout_marginTop="20dip"/>
    <TextView
```

```
 android:id="@+id/textview1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textColor="#FF00ffff"
android:textSize="25dip"
android:text="@string/mk_name"
android:layout_marginLeft="670dip"
android:layout_marginTop="100dip"/>

    <TextView
 android:id="@+id/textview2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textColor="#FFff00ff"
android:textSize="25dip"
android:layout_marginLeft="640dip"
android:layout_marginTop="160dip"/>
<TextView
 android:id="@+id/textview3"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textColor="#FFff00ff"
android:textSize="25dip"
android:drawablePadding="20px"
android:layout_marginLeft="640dip"
android:layout_marginTop="240dip"/>
 <TextView
 android:id="@+id/textview4"
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:textColor="#FFff00ff"
android:textSize="25dip"
android:layout_marginTop="320dip"
android:layout_marginLeft="640dip"
/>

<TextView
 android:id="@+id/textview5"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textColor="#FF000000"
android:textSize="25dip"
android:layout_marginLeft="730dip"
android:layout_marginTop="160dip"/>
<TextView
 android:id="@+id/textview6"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textColor="#FF000000"
android:textSize="25dip"
android:drawablePadding="20px"
android:layout_marginLeft="730dip"
android:layout_marginTop="240dip"/>
 <TextView
 android:id="@+id/textview7"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textColor="#FF000000"
```

```
android:textSize="25dip"
android:layout_marginTop="320dip"
android:layout_marginLeft="730dip"
/>

<Button
android:id="@+id/bt_b_exit"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
android:layout_marginBottom="10dip"
 android:background="@drawable/bt_es"
android:layout_marginLeft="700dip"
android:text="@string/bt_exit"
android:textColor="@drawable/white"
/>

<Button
android:id="@+id/bt_b_start"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
android:layout_marginBottom="10dip"
android:background="@drawable/bt_es"
android:layout_marginLeft="600dip"
android:text="@string/bt_start"
android:textColor="@drawable/white"
/>
```

```
<Button
android:id="@+id/S_mk"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
android:layout_marginBottom="370dip"
android:layout_marginLeft="13dip"
android:background="@drawable/mmds_m"
android:text="@string/mk_select"
android:textColor="@drawable/blue"
android:textSize="22px"
/>

 <Button
android:id="@+id/S_gd"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
android:layout_marginBottom="320dip"
android:background="@drawable/mds"
android:layout_marginLeft="10dip"
android:text="@string/mk_gd"
android:textColor="@drawable/white"
/>
  <Button
android:id="@+id/S_gm"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
```

```
android:layout_marginBottom="280dip"
android:background="@drawable/mds"
android:layout_marginLeft="10dip"
android:text="@string/mk_gm"
android:textColor="@drawable/white"
/>
  <Button
android:id="@+id/S_wd"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
android:layout_marginBottom="240dip"
android:background="@drawable/mds"
android:layout_marginLeft="10dip"
android:text="@string/mk_wd"
android:textColor="@drawable/white"
/>

<Button
android:id="@+id/S_bma150"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
android:layout_marginBottom="200dip"
android:background="@drawable/mds"
android:layout_marginLeft="10dip"
android:text="@string/mk_bma"
android:textColor="@drawable/white"
/>
```

```
  <Button
android:id="@+id/S_ep"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
android:layout_marginBottom="160dip"
android:background="@drawable/mds"
android:layout_marginLeft="10dip"
android:text="@string/mk_eep"
android:textColor="@drawable/white"
/>
<Button
android:id="@+id/S_rt"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
android:layout_marginBottom="120dip"
 android:background="@drawable/mds"
android:layout_marginLeft="10dip"
android:text="@string/mk_rt"
android:textColor="@drawable/white"
/>

<Button
android:id="@+id/S_jj"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
```

```
    android:layout_marginBottom="80dip"
    android:layout_marginLeft="10dip"
    android:background="@drawable/mds"
    android:text="@string/mk_jj"
    android:textColor="@drawable/white"
    />
</RelativeLayout>
```

這是個佈局檔，是對我們的 activity 進行的設置，如果您設定完了，效果就是我們進入的頁面一樣了。

現在讓我們來看看 activityMain.java 文件吧。

其程式碼如下：

```
package tw.com.dmatek.dma6410xp;
import java.text.DecimalFormat;
import tw.com.dmatek.bar.VerticalProgressBar;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.TextView;
/******使用到的包，在此就不多介紹**********************/

public class activityMain extends Activity                //繼續了 Activity
```

```
{


        private Button
mBt_exit,mBt_start,mBt_mk,mBt_gd,mBt_gm,mBt_wd,mBt_BMA150,mBt_EEPROM,
mBt_rt,mBt_jj;
        private TextView mTV_mk,mTV_7,mTV_2,mTV_3,mTV_4,mTV_5,mTV_6;
        private static int
fg=1,fg_gd=1,fg_gm=1,fg_wd=1,fg_bma150=1,fg_eeprom=1,fg_rt=1,fg_jj=1;
        private static int fg_mk;
        private boolean pth_read1;
        private static float
v_gd,v_gm,v_wd,v_wd_x,v_bma_x,v_bma_x_sm,v_bma_y,v_bma_y_sm,v_bma_z,v_bm
a_z_sm,v_eeprom,v_rt,v_jj;
        private static float
dv_gd,dv_gm,dv_wd,dv_bma_x,dv_bma_x_sm,dv_bma_y,dv_bma_y_sm,dv_bma_z,dv_
bma_z_sm,dv_eeprom,dv_rt,dv_jj;
        private String
sv_gd,sv_gm,sv_wd,sv_bma_x,sv_bma_y,sv_bma_z,sv_eeprom,sv_rt,sv_jj;
        private int fd;
         VerticalProgressBar vProgressBar;
         DecimalFormat df;
/***************以上是進行的變數定義*************************/
         Handler myHandler = new Handler()
         {
              public void handleMessage(Message msg)
              {

                   if(msg.what==1)
                   {
```

```
        if(fg_mk==1)
        {
         mTV_2.setVisibility(TextView.VISIBLE);
         mTV_5.setVisibility(TextView.VISIBLE);
         mTV_3.setVisibility(TextView.VISIBLE);
         mTV_6.setVisibility(TextView.VISIBLE);

         mTV_2.setText(R.string.ad_v);
         mTV_5.setText(""+(int)v_gd);
         int ss=(int)((0-v_gd)*350/32000+390);
         if(Config.num1<=45)
         {
                 if(Config.num1<45)
                         Config.num1++;
                 for(int sss=Config.num1;sss>0;sss--)
                         Config.Y[sss]=Config.Y[sss-1];
                 Config.Y[0]=ss;
         }
         dv_gd=v_gd/32000*10/3;
         vProgressBar.setProgress((int)v_gd/320+5);
         mTV_3.setText(R.string.dy_v);
         sv_gd=df.format(dv_gd).toString();
         mTV_6.setText(" "+sv_gd);

         mTV_4.setVisibility(TextView.VISIBLE);
         mTV_7.setVisibility(TextView.VISIBLE);
         mTV_4.setText(R.string.gd_v);
mTV_7.setText("    ");
        }
```

```
        if(fg_mk==2)
{
        mTV_2.setVisibility(TextView.VISIBLE);
mTV_5.setVisibility(TextView.VISIBLE);
mTV_3.setVisibility(TextView.VISIBLE);
        mTV_6.setVisibility(TextView.VISIBLE);
        mTV_2.setText(R.string.ad_v);
mTV_5.setText(""+(int)v_gm);

        int ss=(int)((3000-v_gm)*350/30000+390);
        if(Config.num1<=45)
        {
                if(Config.num1<45)
                        Config.num1++;
                for(int sss=Config.num1;sss>0;sss--)
                        Config.Y[sss]=Config.Y[sss-1];
                Config.Y[0]=ss;
        }
        dv_gm=(v_gm-3000)/30000*10/3;
vProgressBar.setProgress((int)((v_gm-3000)/300+5));
mTV_3.setText(R.string.dy_v);
        sv_gm=df.format(dv_gm).toString();
mTV_6.setText(" "+sv_gm);
mTV_4.setVisibility(TextView.VISIBLE);
mTV_7.setVisibility(TextView.VISIBLE);
mTV_4.setText(R.string.gm_v);
mTV_7.setText("    ");

}
```

```
        if(fg_mk==3)
        {

         mTV_2.setVisibility(TextView.VISIBLE);
mTV_5.setVisibility(TextView.VISIBLE);
mTV_3.setVisibility(TextView.VISIBLE);
mTV_6.setVisibility(TextView.VISIBLE);
mTV_4.setVisibility(TextView.VISIBLE);
mTV_7.setVisibility(TextView.VISIBLE);
mTV_4.setText(R.string.wd_v);
mTV_7.setText(""+(int)v_wd);
        vProgressBar.setProgress((int)(v_wd));
        int ss=(int)((0-v_wd)+340);
        if(Config.num1<=45)
        {
                if(Config.num1<45)
                        Config.num1++;
                for(int sss=Config.num1;sss>0;sss--)
                        Config.Y[sss]=Config.Y[sss-1];
                Config.Y[0]=ss;
        }

        mTV_3.setText(R.string.dy_v);
        dv_wd=(float) (v_wd*0.001+2.69);
        sv_wd=df.format(dv_wd).toString();
        mTV_6.setText("    "+sv_wd);

        mTV_2.setText(R.string.ad_v);
v_wd=((v_wd*10+v_wd_x)+26810)*16384/13810;
```

```
mTV_5.setText(""+(int)v_wd);



        }
        if(fg_mk==4)
        {
mTV_2.setVisibility(TextView.VISIBLE);
mTV_5.setVisibility(TextView.VISIBLE);
mTV_2.setText("          X:");
if(v_bma_x_sm==43)
{
mTV_5.setText("+"+(int)(v_bma_x));
}
if(v_bma_x_sm==45)
{
mTV_5.setText("-"+(int)(v_bma_x));
}

mTV_3.setVisibility(TextView.VISIBLE);
mTV_6.setVisibility(TextView.VISIBLE);
mTV_3.setText("y:");

if(v_bma_y_sm==43)
{
mTV_6.setText("+"+(int)v_bma_y);
}
if(v_bma_y_sm==45)
{
mTV_6.setText("-"+(int)v_bma_y);
```

```
}

mTV_4.setVisibility(TextView.VISIBLE);
mTV_7.setVisibility(TextView.VISIBLE);
mTV_4.setText("          z:");
if(v_bma_z_sm==43)
{
mTV_7.setText("+"+(int)v_bma_z);
}
if(v_bma_z_sm==45)
{
        mTV_7.setText("-"+(int)v_bma_z);
        }

        }
        if(fg_mk==5)
        {
        mTV_3.setVisibility(TextView.VISIBLE);
        mTV_6.setVisibility(TextView.VISIBLE);
        mTV_3.setText(R.string.eep_v_out);
        mTV_6.setText(" "+(int)v_eeprom);
        }
        if(fg_mk==6)
        {
        mTV_2.setVisibility(TextView.VISIBLE);
        mTV_3.setVisibility(TextView.VISIBLE);

        if(v_rt==0)
        {
```

```
            mTV_2.setText("ON:");
            mTV_3.setText(R.string.rt_v_on);

        }
        else
        {
            mTV_2.setText("OFF:");
            mTV_3.setText(R.string.rt_v_off);
        }


    }
    if(fg_mk==7)
    {
     mTV_2.setVisibility(TextView.VISIBLE);
     mTV_5.setVisibility(TextView.VISIBLE);
     mTV_3.setVisibility(TextView.VISIBLE);
     mTV_6.setVisibility(TextView.VISIBLE);
     mTV_2.setText(R.string.ad_v);
     mTV_5.setText(" "+(int)v_jj);

     int ss=(int)((1416-v_jj)*350/24000+390);
     if(Config.num1<=45)
     {
              if(Config.num1<45)
                   Config.num1++;
              for(int sss=Config.num1;sss>0;sss--)
                   Config.Y[sss]=Config.Y[sss-1];
              Config.Y[0]=ss;
```

```
                        }

                        dv_jj=(v_jj-1416)/24000*10/3;
                        vProgressBar.setProgress((int)((v_jj-1416)/240+5));
                    mTV_3.setText(R.string.dy_v);
                    sv_jj=df.format(dv_jj).toString();
                    mTV_6.setText(" "+sv_jj);
                    mTV_4.setVisibility(TextView.VISIBLE);
                    mTV_7.setVisibility(TextView.VISIBLE);
                    mTV_4.setText(R.string.jj_v);
                    mTV_7.setText("    ");
                        }
                }
                super.handleMessage(msg);
            }
        };

/*****************以上是對消息進行處理**********************/
class myThread1 implements Runnable
{
    public void run()
    {
        while (!Thread.currentThread().isInterrupted())
        {
            Message message = new Message();
            message.what = 1;
            activityMain.this.myHandler.sendMessage(message);
            try {
                if(pth_read1)
```

```
        {
            Thread.sleep(100);
            Log.v("thread:","rrr");
            //sv_wd=Linuxc.receiveMsgSensors(3);
            v_gd=Linuxc.receiveSensors(1);
            v_gm=Linuxc.receiveSensors(2);
            v_wd=Linuxc.receiveSensors(3);
            v_wd_x=Linuxc.receiveSensors(33);

            v_bma_x=Linuxc.receiveSensors(4);
            v_bma_x_sm=Linuxc.receiveSensors(8);
            v_bma_y=Linuxc.receiveSensors(44);
            v_bma_y_sm=Linuxc.receiveSensors(88);
            v_bma_z=Linuxc.receiveSensors(444);
            v_bma_z_sm=Linuxc.receiveSensors(888);

            v_eeprom=Linuxc.receiveSensors(5);
            Log.v("thread:","run"+v_eeprom);

            v_rt=Linuxc.receiveSensors(6);
            v_jj=Linuxc.receiveSensors(7);

            Log.v("thread:","run"+v_eeprom);

        }
        else break;
    } catch (InterruptedException e)
    {
            // TODO Auto-generated catch block
```

```
                    e.printStackTrace();
                }
            }
        }
}
/*********************以上是對線程繼承的介面的實現*****************/

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState); //調用父類的構造方法
        requestWindowFeature(Window.FEATURE_NO_TITLE);

     getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN ,
                          WindowManager.LayoutParams.FLAG_FULLSCREEN);
        /**********************實現全螢幕顯示*********************/
        setContentView(R.layout.main);    //調用佈局文件
        vProgressBar = (VerticalProgressBar)findViewById(R.id.ProgressBar01);
        vProgressBar.setMax(100);//設置進度條的最大值
        vProgressBar.setProgress(0);//設置進度條的初始值
        df = new DecimalFormat("0.00"); //設置顯示格式
       // vProgressBar.setSecondaryProgress(90);

、
        mTV_mk=(TextView)findViewById(R.id.textview1);
        mTV_2=(TextView)findViewById(R.id.textview2);
        mTV_3=(TextView)findViewById(R.id.textview3);
        mTV_4=(TextView)findViewById(R.id.textview4);
```

```
        mTV_5=(TextView)findViewById(R.id.textview5);
        mTV_6=(TextView)findViewById(R.id.textview6);
        mTV_7=(TextView)findViewById(R.id.textview7);

        /*******************獲取 TextVIew 的資源 ID 號****************/

        mBt_exit=(Button)findViewById(R.id.bt_b_exit);
        mBt_exit.setOnClickListener(exit_button_listener);
        mBt_start=(Button)findViewById(R.id.bt_b_start);
        mBt_start.setOnClickListener(start_button_listener);

        mBt_mk=(Button)findViewById(R.id.S_mk);
        mBt_gd=(Button)findViewById(R.id.S_gd);
        mBt_gm=(Button)findViewById(R.id.S_gm);
        mBt_wd=(Button)findViewById(R.id.S_wd);
        mBt_BMA150=(Button)findViewById(R.id.S_bma150);
        mBt_EEPROM=(Button)findViewById(R.id.S_ep);
        mBt_rt=(Button)findViewById(R.id.S_rt);
        mBt_jj=(Button)findViewById(R.id.S_jj);
    /************設置 Button 的資源 ID 號******************/
        mBt_gd.setVisibility(Button.INVISIBLE);
        mBt_gm.setVisibility(Button.INVISIBLE);
        mBt_wd.setVisibility(Button.INVISIBLE);
        mBt_BMA150.setVisibility(Button.INVISIBLE);
        mBt_EEPROM.setVisibility(Button.INVISIBLE);
        mBt_rt.setVisibility(Button.INVISIBLE);
        mBt_jj.setVisibility(Button.INVISIBLE);

    /****************設置 button 的可見度********************/
```

```
            mTV_2.setVisibility(TextView.INVISIBLE);
            mTV_3.setVisibility(TextView.INVISIBLE);
            mTV_4.setVisibility(TextView.INVISIBLE);
            mTV_5.setVisibility(TextView.INVISIBLE);
            mTV_6.setVisibility(TextView.INVISIBLE);
            mTV_7.setVisibility(TextView.INVISIBLE);
/**************************設置 TextView 的可見度****************/
            mBt_mk.setOnClickListener(mk_button_listener);
            mBt_gd.setOnClickListener(gd_button_listener);
            mBt_gm.setOnClickListener(gm_button_listener);
            mBt_wd.setOnClickListener(wd_button_listener);
            mBt_BMA150.setOnClickListener(BMA150_button_listener);
            mBt_EEPROM.setOnClickListener(EEPROM_button_listener);
            mBt_rt.setOnClickListener(rt_button_listener);
            mBt_jj.setOnClickListener(jj_button_listener);
/*****************設置 button 的監聽事件*******************************/
            DrawView.flags=0;//定義曲線繪製

    }
    private Button.OnClickListener exit_button_listener=new Button.OnClickListener()
     {

            public void onClick(View v)
            {
                  // TODO Auto-generated method stub
                  pth_read1=false;
                  fd=Linuxc.closeSensors(0);
                  finish();
```

```
        }
};

private Button.OnClickListener start_button_listener=new Button.OnClickListener()
{

        public void onClick(View v)
        {
                // TODO Auto-generated method stub

                  //Linuxc.setSensors(7);
                 mBt_start.setBackgroundResource(R.drawable.bt_es_1);
                 fd=Linuxc.openSensors(2);
                if(fd>0)
                {
                        pth_read1=true;
                        Linuxc.setSensors(9);
                 Log.v("open is success", fd+"");
                }
                else Log.v("open is false",""+fd);
                DrawView.flags=1;
                pth_read1=true;
                 new Thread(new myThread1()).start();
        }
};
private Button.OnClickListener mk_button_listener=new Button.OnClickListener()
{
        public void onClick(View v)
```

```
{
        // TODO Auto-generated method stub

        if(fg==1)
        {
             mBt_gd.setVisibility(Button.VISIBLE);
            mBt_gm.setVisibility(Button.VISIBLE);
            mBt_wd.setVisibility(Button.VISIBLE);
            mBt_BMA150.setVisibility(Button.VISIBLE);
            mBt_EEPROM.setVisibility(Button.VISIBLE);
            mBt_rt.setVisibility(Button.VISIBLE);
            mBt_jj.setVisibility(Button.VISIBLE);
            mBt_mk.setBackgroundResource(R.drawable.mds_m1);
            mBt_mk.setText("");
            fg=0;

        }
        else
        {
                 mBt_gd.setVisibility(Button.INVISIBLE);
                mBt_gm.setVisibility(Button.INVISIBLE);
                mBt_wd.setVisibility(Button.INVISIBLE);
                mBt_BMA150.setVisibility(Button.INVISIBLE);
                mBt_EEPROM.setVisibility(Button.INVISIBLE);
                mBt_rt.setVisibility(Button.INVISIBLE);
                mBt_jj.setVisibility(Button.INVISIBLE);
                mTV_mk.setText(R.string.mk_name);
                mTV_2.setVisibility(TextView.INVISIBLE);
                 mTV_3.setVisibility(TextView.INVISIBLE);
```

```
                    mTV_4.setVisibility(TextView.INVISIBLE);
                    mTV_5.setVisibility(TextView.INVISIBLE);
                    mTV_6.setVisibility(TextView.INVISIBLE);
                    mTV_7.setVisibility(TextView.INVISIBLE);
                    fg=1;
                   // mBt_mk.setTextColor(R.drawable.blue);

            }
        }
};
private Button.OnClickListener gd_button_listener=new Button.OnClickListener()
{

        public void onClick(View v)
        {

            // TODO Auto-generated method stub

            if(fg_gd==1)
            {

                mBt_gm.setVisibility(Button.INVISIBLE);
                mBt_wd.setVisibility(Button.INVISIBLE);
                mBt_BMA150.setVisibility(Button.INVISIBLE);
                mBt_EEPROM.setVisibility(Button.INVISIBLE);
                mBt_rt.setVisibility(Button.INVISIBLE);
                mBt_jj.setVisibility(Button.INVISIBLE);
                mTV_mk.setText(R.string.mk_gd);
                fg_mk=1;
```

```
                fg_gd=0;
                DrawView.flags=1;
                mBt_gd.setBackgroundResource(R.drawable.mds_1);
            }
            else
            {
                    mBt_gd.setVisibility(Button.VISIBLE);
                    mBt_gm.setVisibility(Button.VISIBLE);
                    mBt_wd.setVisibility(Button.VISIBLE);
                    mBt_BMA150.setVisibility(Button.VISIBLE);
                    mBt_EEPROM.setVisibility(Button.VISIBLE);
                    mBt_rt.setVisibility(Button.VISIBLE);
                    mBt_jj.setVisibility(Button.VISIBLE);
                    mTV_mk.setText(R.string.mk_name);
                    mTV_2.setVisibility(TextView.INVISIBLE);
                    mTV_3.setVisibility(TextView.INVISIBLE);
                    mTV_4.setVisibility(TextView.INVISIBLE);
                    mTV_5.setVisibility(TextView.INVISIBLE);
                    mTV_6.setVisibility(TextView.INVISIBLE);
                    mTV_7.setVisibility(TextView.INVISIBLE);
                    fg_mk=0;
                    fg_gd=1;
                    DrawView.flags=0;
                    mBt_gd.setBackgroundResource(R.drawable.mds);

            }
        }
    };
    private Button.OnClickListener gm_button_listener=new Button.OnClickListener()
```

```
{

    public void onClick(View v)
    {
        if(fg_gm==1)
        {

            mBt_gd.setVisibility(Button.INVISIBLE);
            mBt_wd.setVisibility(Button.INVISIBLE);
            mBt_BMA150.setVisibility(Button.INVISIBLE);
            mBt_EEPROM.setVisibility(Button.INVISIBLE);
            mBt_rt.setVisibility(Button.INVISIBLE);
            mBt_jj.setVisibility(Button.INVISIBLE);
            mTV_mk.setText(R.string.mk_gm);
            fg_mk=2;
            fg_gm=0;
            DrawView.flags=1;
            mBt_gm.setBackgroundResource(R.drawable.mds_1);
        }
        else
        {
             mBt_gd.setVisibility(Button.VISIBLE);
            mBt_gm.setVisibility(Button.VISIBLE);
            mBt_wd.setVisibility(Button.VISIBLE);
            mBt_BMA150.setVisibility(Button.VISIBLE);
            mBt_EEPROM.setVisibility(Button.VISIBLE);
            mBt_rt.setVisibility(Button.VISIBLE);
            mBt_jj.setVisibility(Button.VISIBLE);
            mTV_mk.setText(R.string.mk_name);
```

```
                    mTV_2.setVisibility(TextView.INVISIBLE);
                    mTV_3.setVisibility(TextView.INVISIBLE);
                    mTV_4.setVisibility(TextView.INVISIBLE);
                    mTV_5.setVisibility(TextView.INVISIBLE);
                    mTV_6.setVisibility(TextView.INVISIBLE);
                    mTV_7.setVisibility(TextView.INVISIBLE);
                    fg_mk=0;
                   fg_gm=1;
                   DrawView.flags=0;
                   mBt_gm.setBackgroundResource(R.drawable.mds);

            }
        }
};
private Button.OnClickListener wd_button_listener=new Button.OnClickListener()
{

    public void onClick(View v)
    {
        // TODO Auto-generated method stub
        if(fg_wd==1)
        {

            mBt_gm.setVisibility(Button.INVISIBLE);
            mBt_gd.setVisibility(Button.INVISIBLE);
            mBt_BMA150.setVisibility(Button.INVISIBLE);
            mBt_EEPROM.setVisibility(Button.INVISIBLE);
            mBt_rt.setVisibility(Button.INVISIBLE);
            mBt_jj.setVisibility(Button.INVISIBLE);
```

```
                mTV_mk.setText(R.string.mk_wd);
                fg_mk=3;
                fg_wd=0;
                DrawView.flags=1;
                mBt_wd.setBackgroundResource(R.drawable.mds_1);
            }
            else
            {
                    mBt_gd.setVisibility(Button.VISIBLE);
                   mBt_gm.setVisibility(Button.VISIBLE);
                   mBt_wd.setVisibility(Button.VISIBLE);
                   mBt_BMA150.setVisibility(Button.VISIBLE);
                   mBt_EEPROM.setVisibility(Button.VISIBLE);
                   mBt_rt.setVisibility(Button.VISIBLE);
                   mBt_jj.setVisibility(Button.VISIBLE);
                   mTV_mk.setText(R.string.mk_name);
                   mTV_2.setVisibility(TextView.INVISIBLE);
                    mTV_3.setVisibility(TextView.INVISIBLE);
                    mTV_4.setVisibility(TextView.INVISIBLE);
                    mTV_5.setVisibility(TextView.INVISIBLE);
                    mTV_6.setVisibility(TextView.INVISIBLE);
                    mTV_7.setVisibility(TextView.INVISIBLE);
                   fg_mk=0;
                  fg_wd=1;
                  DrawView.flags=0;
                  mBt_wd.setBackgroundResource(R.drawable.mds);

            }
        }
```

```
    };
    private Button.OnClickListener BMA150_button_listener=new
Button.OnClickListener()
    {

        public void onClick(View v)
        {
            // TODO Auto-generated method stub
            if(fg_bma150==1)
            {

                mBt_gm.setVisibility(Button.INVISIBLE);
                mBt_wd.setVisibility(Button.INVISIBLE);
                mBt_gd.setVisibility(Button.INVISIBLE);
                mBt_EEPROM.setVisibility(Button.INVISIBLE);
                mBt_rt.setVisibility(Button.INVISIBLE);
                mBt_jj.setVisibility(Button.INVISIBLE);
                mTV_mk.setText(R.string.mk_bma);
                fg_mk=4;
                fg_bma150=0;
                DrawView.flags=1;
                mBt_BMA150.setBackgroundResource(R.drawable.mds_1);
            }
            else
            {
                 mBt_gd.setVisibility(Button.VISIBLE);
                mBt_gm.setVisibility(Button.VISIBLE);
                mBt_wd.setVisibility(Button.VISIBLE);
                mBt_BMA150.setVisibility(Button.VISIBLE);
```

```
                    mBt_EEPROM.setVisibility(Button.VISIBLE);
                    mBt_rt.setVisibility(Button.VISIBLE);
                    mBt_jj.setVisibility(Button.VISIBLE);
                    mTV_mk.setText(R.string.mk_name);
                    mTV_2.setVisibility(TextView.INVISIBLE);
                     mTV_3.setVisibility(TextView.INVISIBLE);
                     mTV_4.setVisibility(TextView.INVISIBLE);
                     mTV_5.setVisibility(TextView.INVISIBLE);
                     mTV_6.setVisibility(TextView.INVISIBLE);
                     mTV_7.setVisibility(TextView.INVISIBLE);
                     fg_mk=0;
                    fg_bma150=1;
                    DrawView.flags=0;
                    mBt_BMA150.setBackgroundResource(R.drawable.mds);
                }
            }
        };
        private Button.OnClickListener EEPROM_button_listener=new
Button.OnClickListener()
        {

            public void onClick(View v)
            {
                // TODO Auto-generated method stub
                if(fg_eeprom==1)
                {

                    mBt_gm.setVisibility(Button.INVISIBLE);
                    mBt_wd.setVisibility(Button.INVISIBLE);
```

```
        mBt_BMA150.setVisibility(Button.INVISIBLE);
        mBt_gd.setVisibility(Button.INVISIBLE);
        mBt_rt.setVisibility(Button.INVISIBLE);
        mBt_jj.setVisibility(Button.INVISIBLE);
        mTV_mk.setText(R.string.mk_eep);
        fg_mk=5;
        fg_eeprom=0;
        DrawView.flags=1;
        mBt_EEPROM.setBackgroundResource(R.drawable.mds_1);
    }
    else
    {
            mBt_gd.setVisibility(Button.VISIBLE);
            mBt_gm.setVisibility(Button.VISIBLE);
            mBt_wd.setVisibility(Button.VISIBLE);
            mBt_BMA150.setVisibility(Button.VISIBLE);
            mBt_EEPROM.setVisibility(Button.VISIBLE);
            mBt_rt.setVisibility(Button.VISIBLE);
            mBt_jj.setVisibility(Button.VISIBLE);
            mTV_mk.setText(R.string.mk_name);
            mTV_2.setVisibility(TextView.INVISIBLE);
            mTV_3.setVisibility(TextView.INVISIBLE);
            mTV_4.setVisibility(TextView.INVISIBLE);
            mTV_5.setVisibility(TextView.INVISIBLE);
            mTV_6.setVisibility(TextView.INVISIBLE);
            mTV_7.setVisibility(TextView.INVISIBLE);
            fg_mk=0;
            fg_eeprom=1;
            DrawView.flags=0;
```

```
                mBt_EEPROM.setBackgroundResource(R.drawable.mds);

            }
        }
};
private Button.OnClickListener rt_button_listener=new Button.OnClickListener()
{

    public void onClick(View v)
    {
        // TODO Auto-generated method stub
        if(fg_rt==1)
        {
            mBt_gm.setVisibility(Button.INVISIBLE);
            mBt_wd.setVisibility(Button.INVISIBLE);
            mBt_BMA150.setVisibility(Button.INVISIBLE);
            mBt_EEPROM.setVisibility(Button.INVISIBLE);
            mBt_gd.setVisibility(Button.INVISIBLE);
            mBt_jj.setVisibility(Button.INVISIBLE);
            mTV_mk.setText(R.string.mk_rt);

            fg_mk=6;
            fg_rt=0;
            DrawView.flags=1;
            mBt_rt.setBackgroundResource(R.drawable.mds_1);
        }
        else
        {
            mBt_gd.setVisibility(Button.VISIBLE);
```

```
                    mBt_gm.setVisibility(Button.VISIBLE);
                    mBt_wd.setVisibility(Button.VISIBLE);
                    mBt_BMA150.setVisibility(Button.VISIBLE);
                    mBt_EEPROM.setVisibility(Button.VISIBLE);
                    mBt_rt.setVisibility(Button.VISIBLE);
                    mBt_jj.setVisibility(Button.VISIBLE);
                    mTV_mk.setText(R.string.mk_name);
                    mTV_2.setVisibility(TextView.INVISIBLE);
                     mTV_3.setVisibility(TextView.INVISIBLE);
                     mTV_4.setVisibility(TextView.INVISIBLE);
                     mTV_5.setVisibility(TextView.INVISIBLE);
                     mTV_6.setVisibility(TextView.INVISIBLE);
                     mTV_7.setVisibility(TextView.INVISIBLE);
                     fg_mk=0;
                    fg_rt=1;
                    DrawView.flags=0;
                    mBt_rt.setBackgroundResource(R.drawable.mds);

            }
        }
};
private Button.OnClickListener jj_button_listener=new Button.OnClickListener()
{

        public void onClick(View v)
        {
            // TODO Auto-generated method stub
            if(fg_jj==1)
            {
```

```
        mBt_gm.setVisibility(Button.INVISIBLE);
        mBt_wd.setVisibility(Button.INVISIBLE);
        mBt_BMA150.setVisibility(Button.INVISIBLE);
        mBt_EEPROM.setVisibility(Button.INVISIBLE);
        mBt_rt.setVisibility(Button.INVISIBLE);
        mBt_gd.setVisibility(Button.INVISIBLE);
        mTV_mk.setText(R.string.mk_jj);
        fg_mk=7;
        fg_jj=0;
        DrawView.flags=1;
        mBt_jj.setBackgroundResource(R.drawable.mds_1);

    }
    else
    {
            mBt_gd.setVisibility(Button.VISIBLE);
            mBt_gm.setVisibility(Button.VISIBLE);
            mBt_wd.setVisibility(Button.VISIBLE);
            mBt_BMA150.setVisibility(Button.VISIBLE);
            mBt_EEPROM.setVisibility(Button.VISIBLE);
            mBt_rt.setVisibility(Button.VISIBLE);
            mBt_jj.setVisibility(Button.VISIBLE);
            mTV_mk.setText(R.string.mk_name);
            mTV_2.setVisibility(TextView.INVISIBLE);
            mTV_3.setVisibility(TextView.INVISIBLE);
            mTV_4.setVisibility(TextView.INVISIBLE);
            mTV_5.setVisibility(TextView.INVISIBLE);
            mTV_6.setVisibility(TextView.INVISIBLE);
```

```
                    mTV_7.setVisibility(TextView.INVISIBLE);
                    fg_mk=0;
                    fg_jj=1;
                    DrawView.flags=0;
                    mBt_jj.setBackgroundResource(R.drawable.mds);
                }
            }
        };
        /**********************實現 button 事件介面**********************/
}
```

到此主介面設計就介紹完成了。

下面介紹曲線繪製使用到的 Config.java 和 drawView.java 文件。

Config.java 程式碼如下：

```
package tw.com.dmatek.dma6410xp;
public class Config {
    public static int Y[]={
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0
        };
    public static int X[]={
        130,140,150,160,170,180,190,200,
        210,220,230,240,250,260,270,280,290,300,
        310,320,330,340,350,360,370,380,390,400,
        410,420,430,440,450,460,470,480,490,500,
        510,520,530,540,550,560,570,580
```

```
        };
    public static int num1=45;
}
/**************這個是曲線繪製的初始化,包括 X 軸和 Y 軸******************/
```

DrawView.java 程式碼如下：

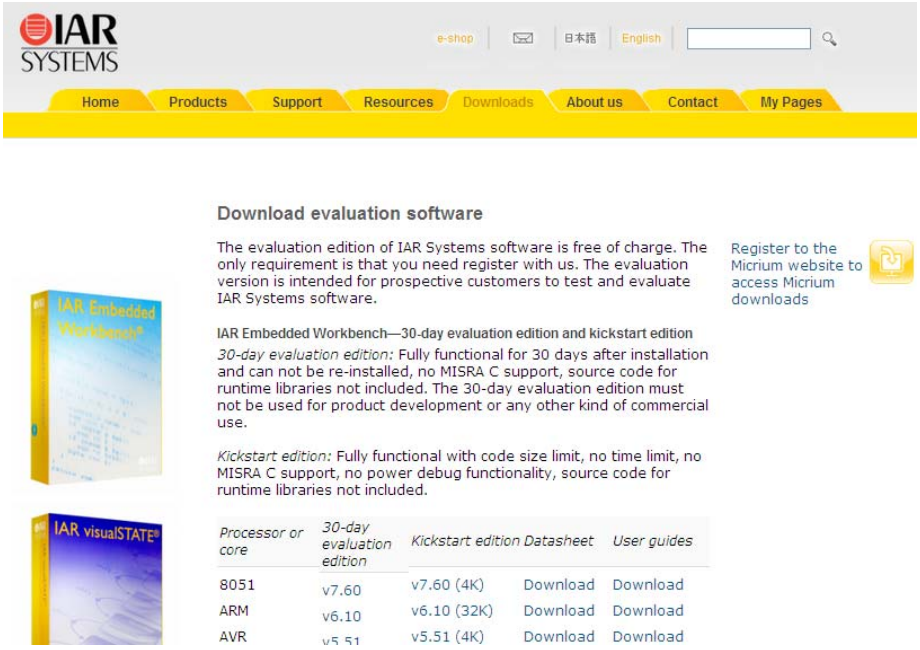
```
package tw.com.dmatek.dma6410xp;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.util.AttributeSet;
import android.view.View;

public class DrawView extends View{
    private Rect    rect=new Rect();
    public DrawView(Context context) {
        super(context);
        // TODO Auto-generated constructor stub
    }
    public DrawView(Context context, AttributeSet attrs) {
        super(context, attrs);
        // TODO Auto-generated constructor stub
    }

    public DrawView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        // TODO Auto-generated constructor stub
```

```
    }

    protected void onDraw(Canvas canvas)
    {
        final Paint pa=new Paint();
        pa.setColor(Color.WHITE);
        rect.set(120, 70, 580, 420);
        canvas.drawRect(rect, pa);

        pa.setColor(Color.GRAY);
        canvas.drawLine(120, 140, 580, 140, pa);
        canvas.drawLine(120, 240, 580, 240, pa);
        canvas.drawLine(120, 340, 580, 340, pa);
        pa.setColor(Color.BLUE);
        pa.setStrokeWidth(4);

        if(flags==1)
        {
        for(int i=0;i<Config.num1-1;i++)
        canvas.drawLine(Config.X[i],Config.Y[i], Config.X[i+1], Config.Y[i+1], pa);
        }
        invalidate();
    }
    public static int flags=0;
}
/***************以上是曲線繪製**********************/
```

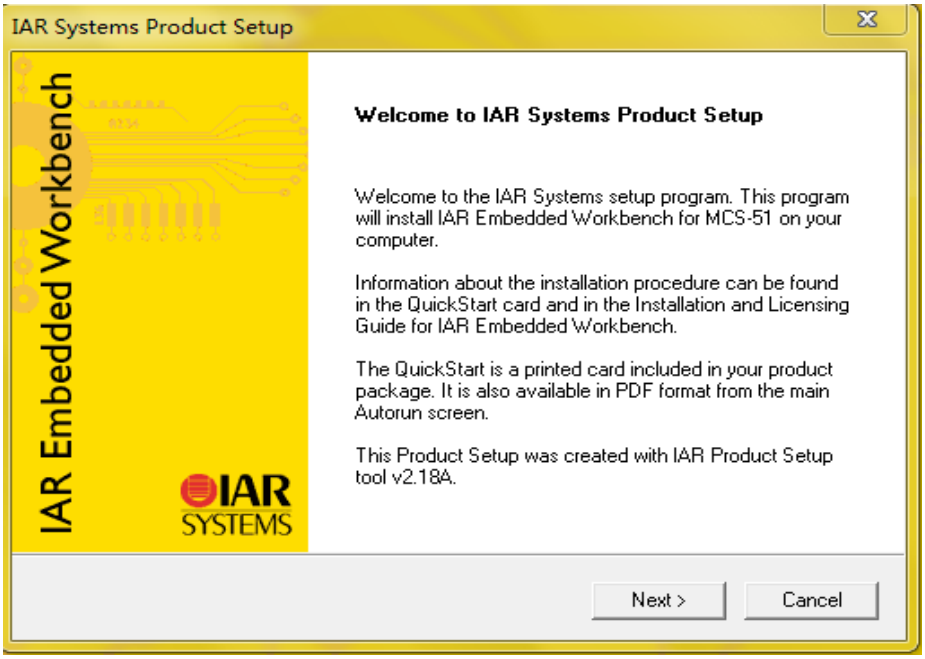
第五章 IAR 開發環境的架設和開發流程

5-1 IAR 軟體的安裝

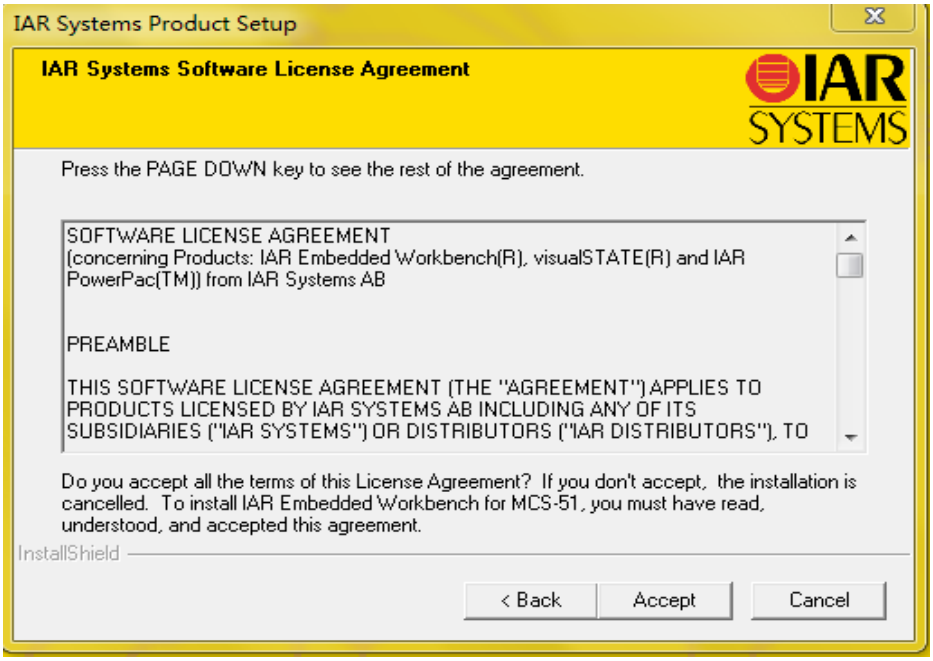
可至IAR官網，購買正版軟體或到至首頁下的Downloads下載30天試用版，記住要下載對應 C8051 的 IAR 軟體），如下圖所示：



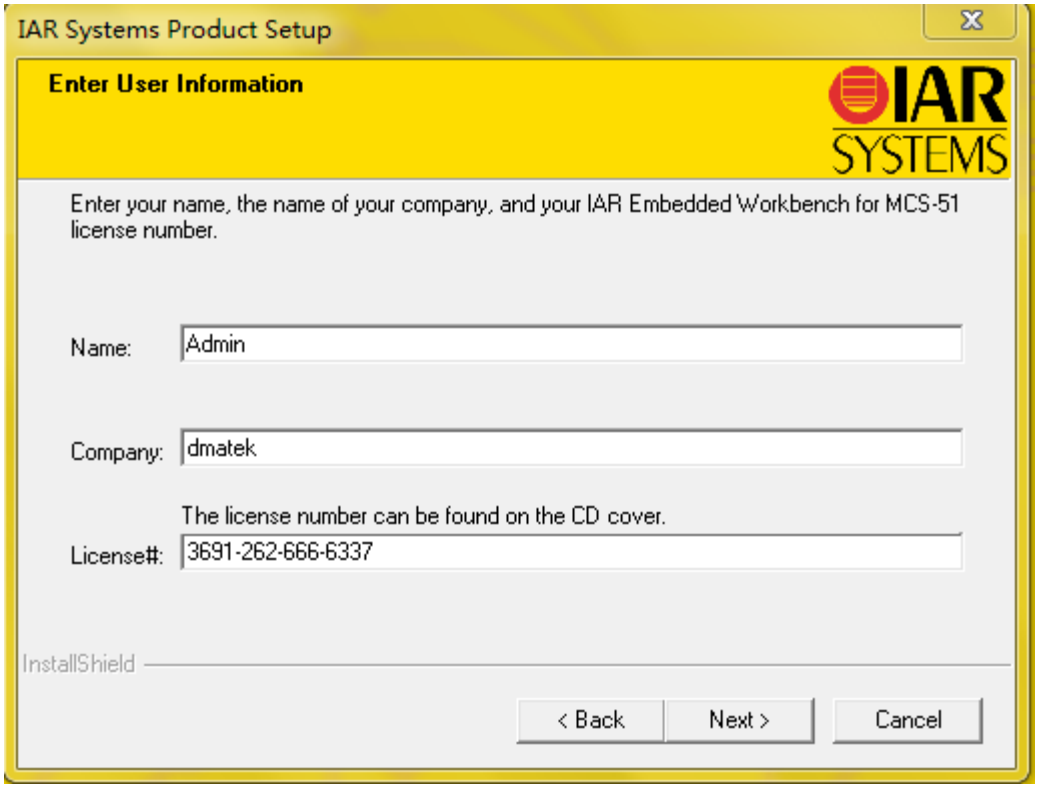
下面是以IAR 8051 7.51A 安裝爲例：



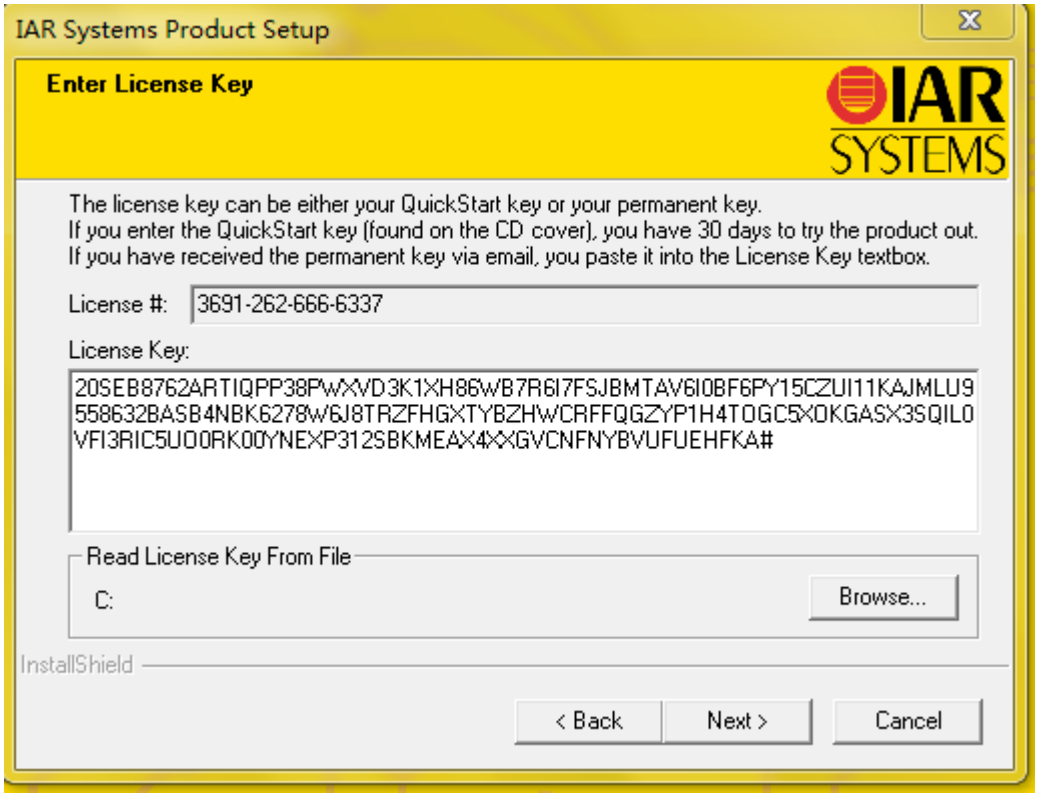
點選 Next，進入如下圖所示：



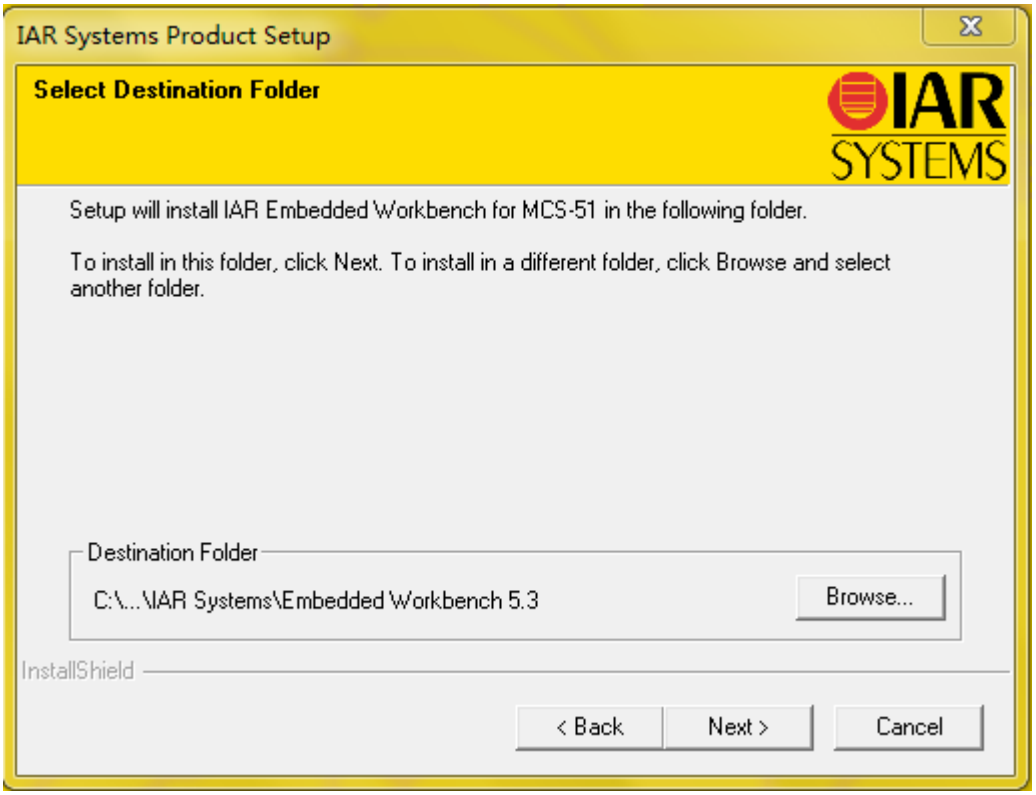
點選 Accept，進入如下圖所示：



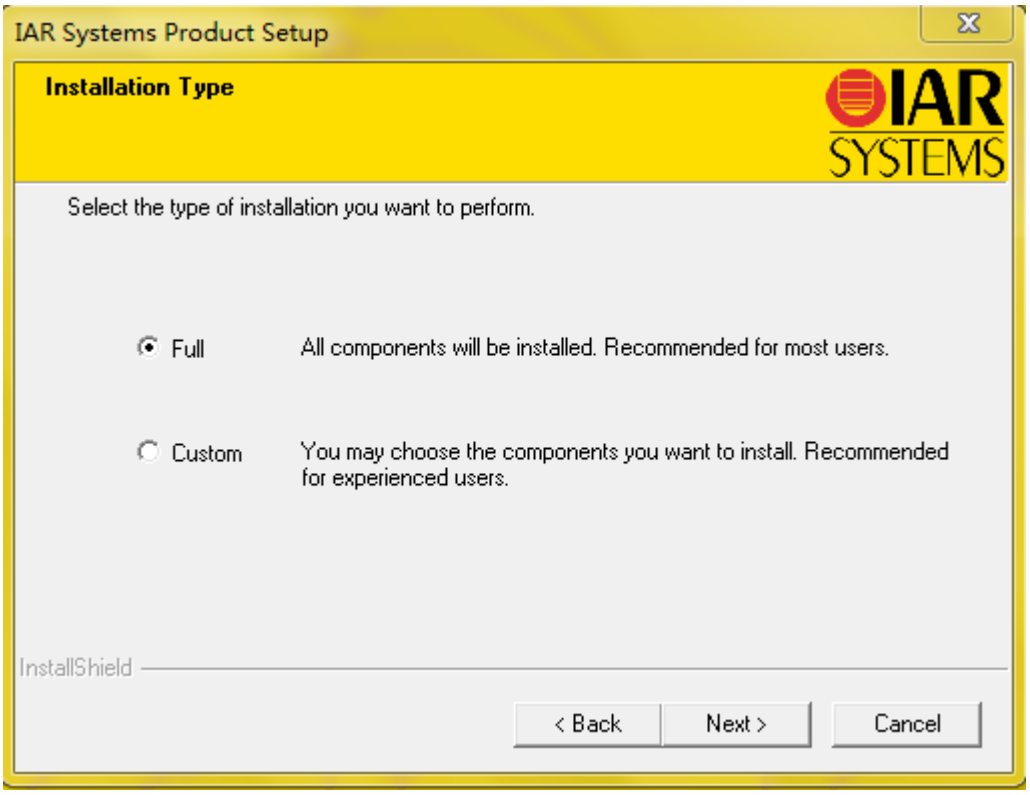
輸入相關資訊後點選 Next



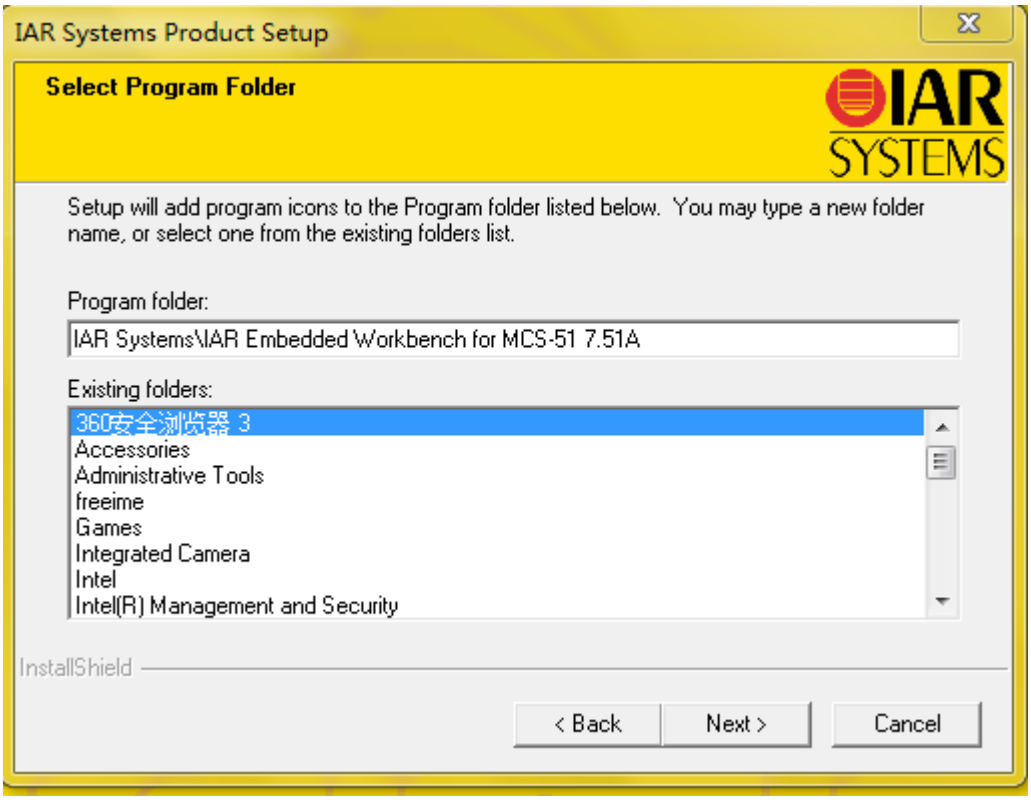
點選 Next，如下圖所示：



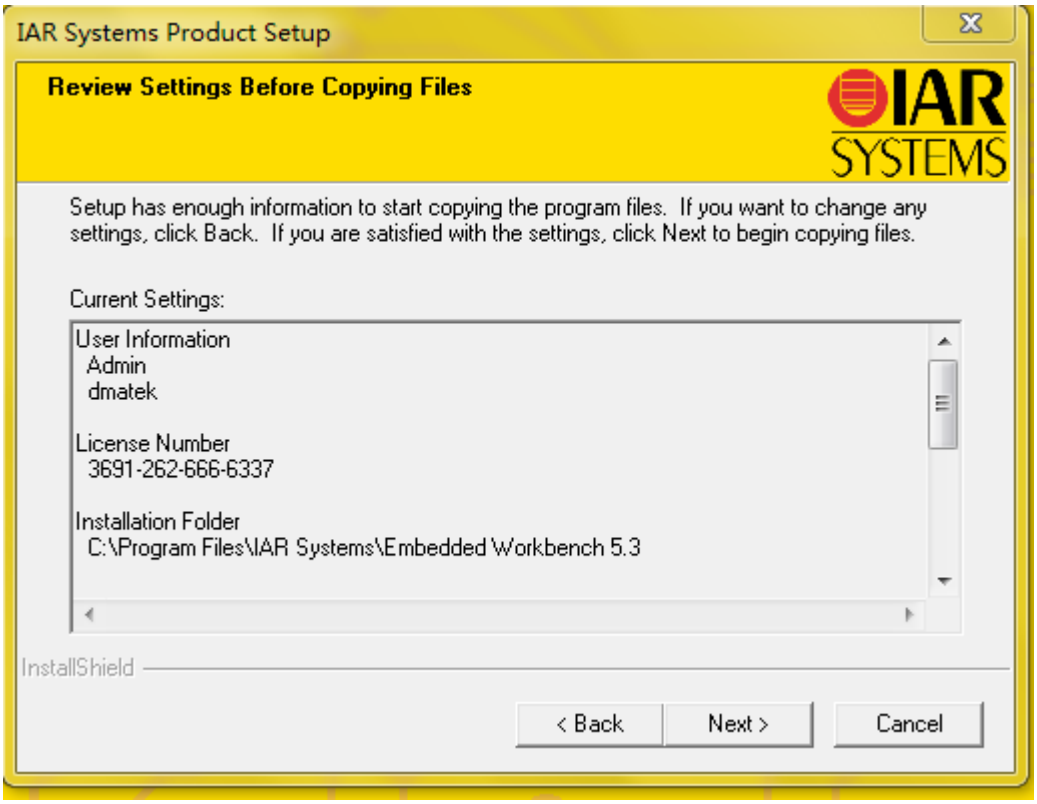
點選 Next，如下圖所示：



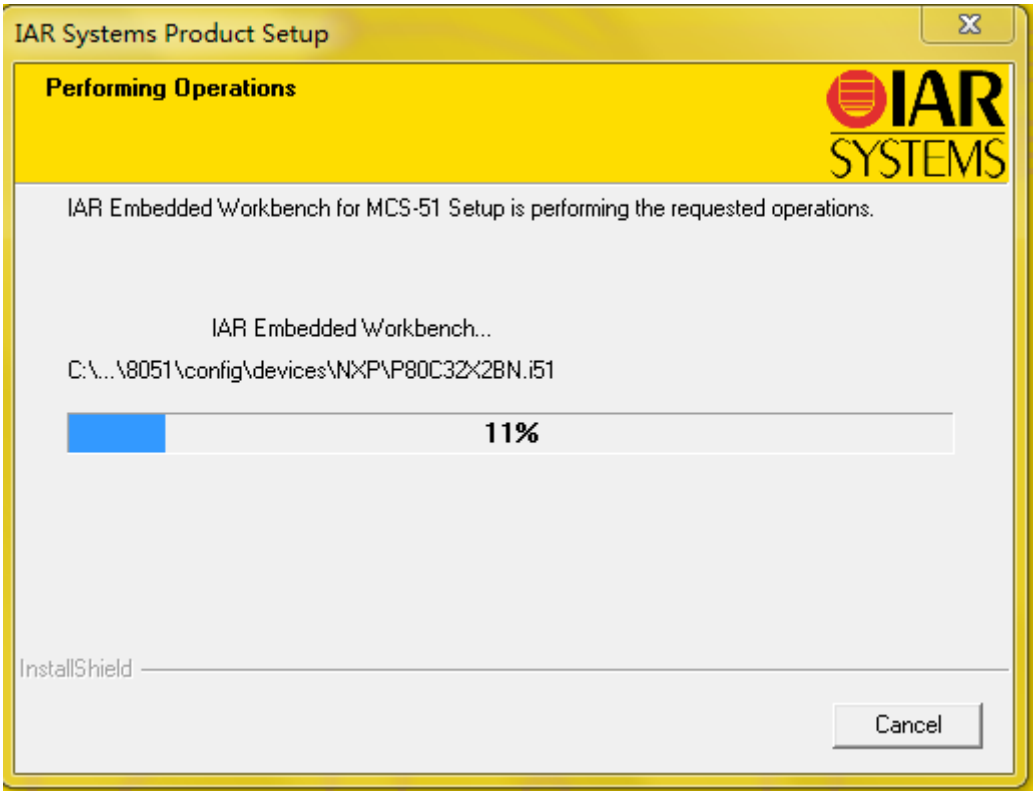
點選 Next，如下圖所示：



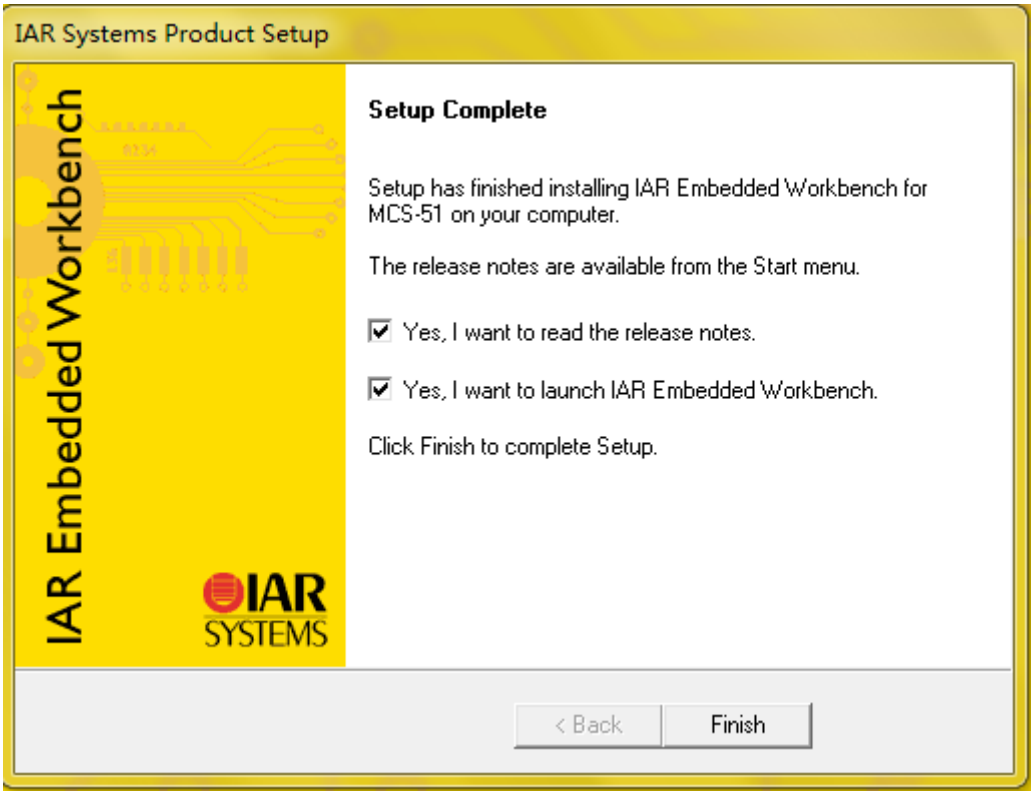
點選 Next，如下圖所示：



點選 Next，如下圖所示：


IAR Systems Product Setup
Performing Operations
IAR SYSTEMS
IAR Embedded Workbench for MCS-51 Setup is performing the requested operations.
IAR Embedded Workbench...
C:\...\8051\config\devices\NXP\P80C32X2BN.i51
11%
InstallShield
Cancel

安裝完成後，如下圖所示：

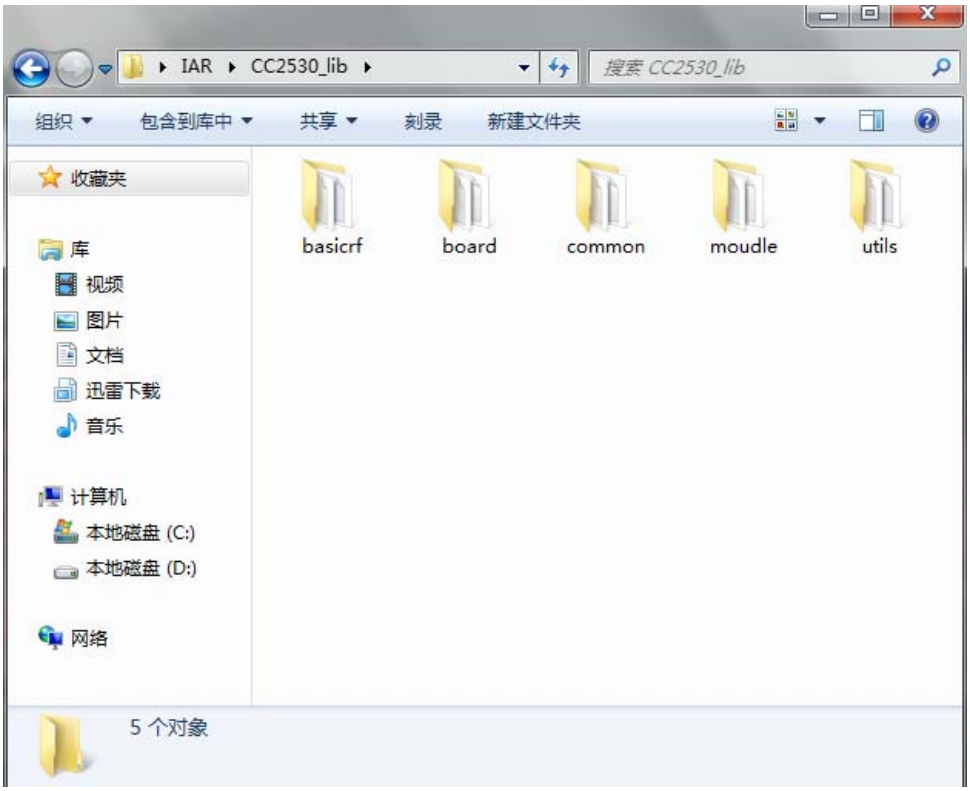


點選 Finish，就完成了 IAR 軟體的安裝。下面將以光敏電阻感測器爲實例介紹如何使用 IAR 軟體來編譯原始程式碼生成 HEX 檔，並透過下載器將 HEX 檔下載到 CC2530 晶片的 RAM 中，這樣就可以執行相應的動作了。

5-2 在 IAR 環境下開發光敏電阻感測器的應用程式

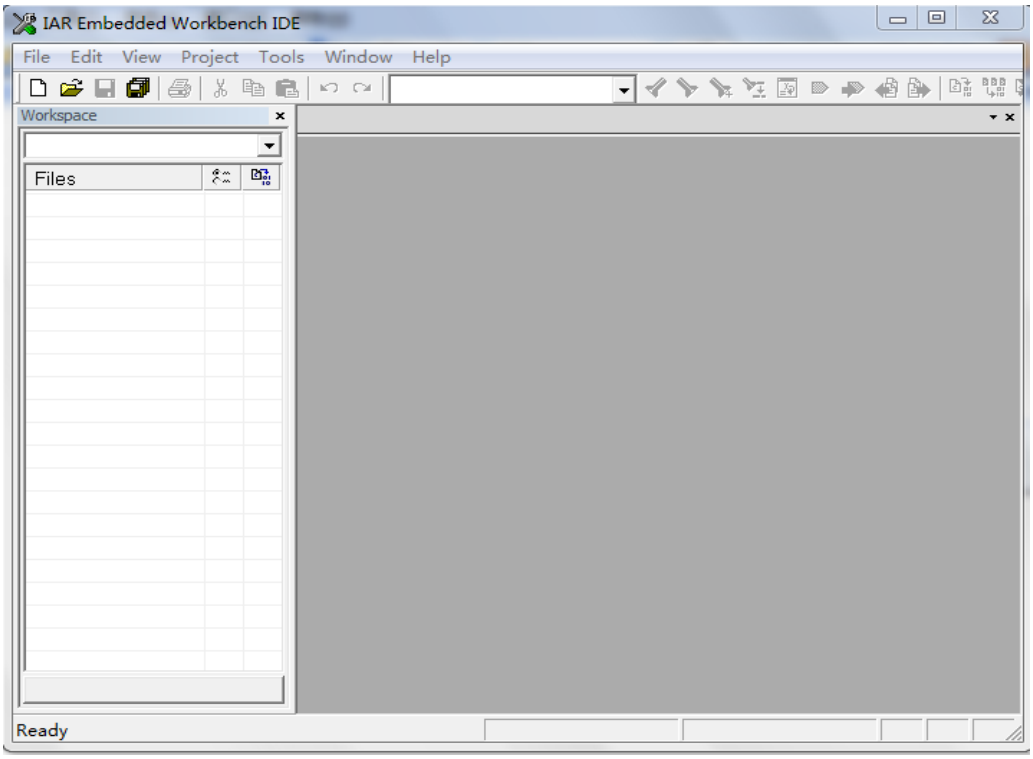
1.固件庫的使用

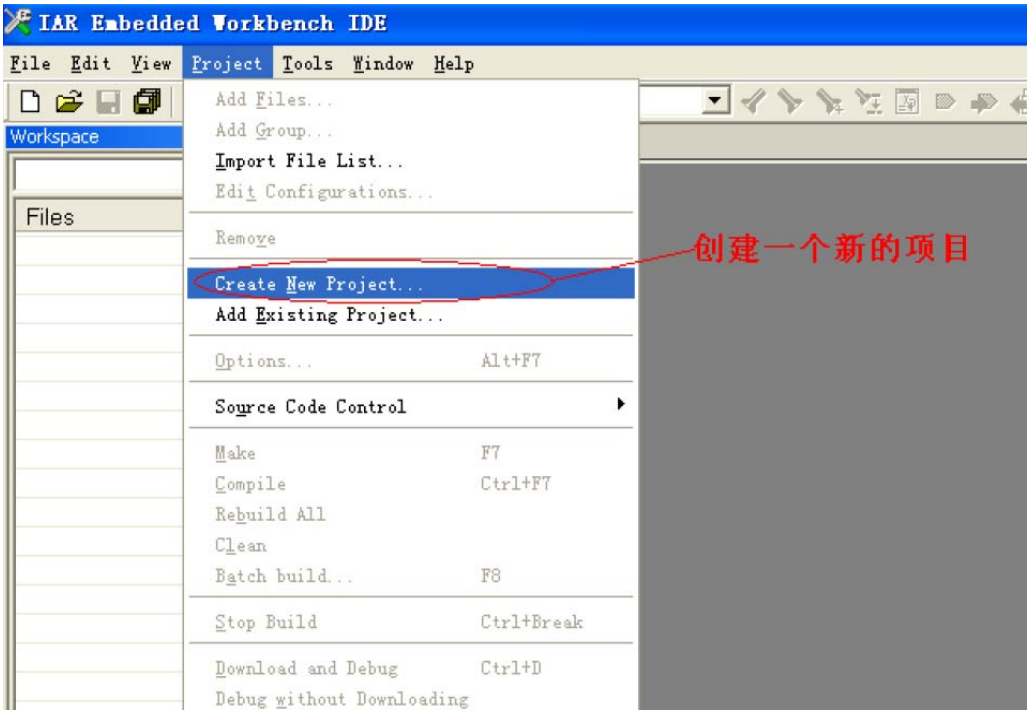
在利用 IAR 環境開發應用之前，先將關於 CC2530 單晶片的 ST 固件庫（也可以進入 ST 官網自己下載）複製到自己創建的工程目錄下，我們提供的固件庫在光碟根目錄下 IAR/ CC2530_lib 的檔案夾中，如下圖所示：

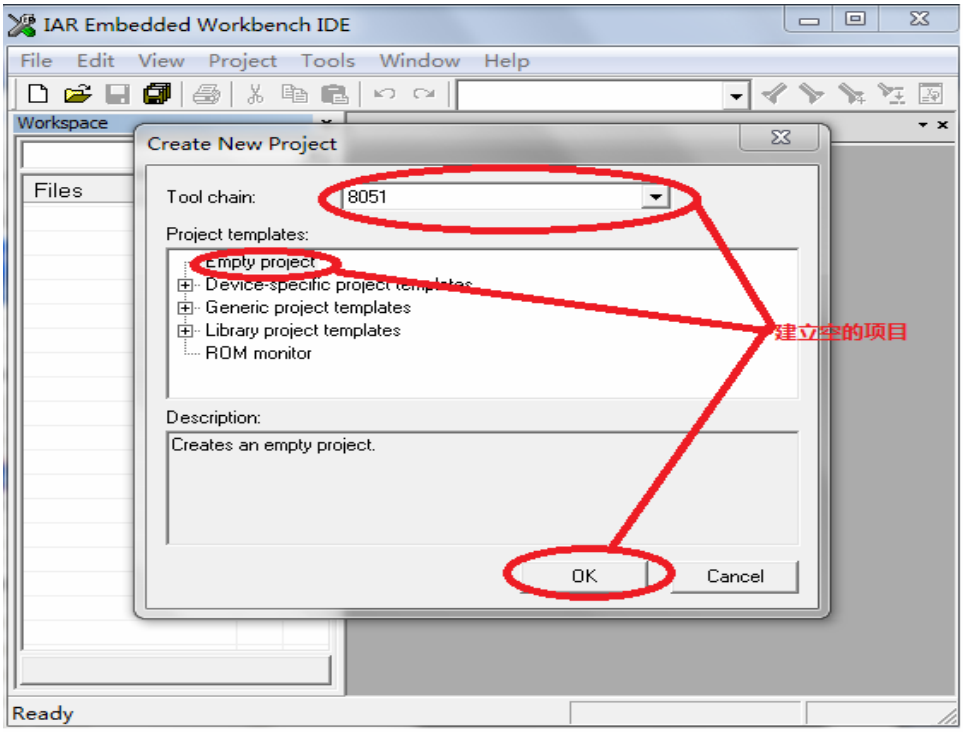


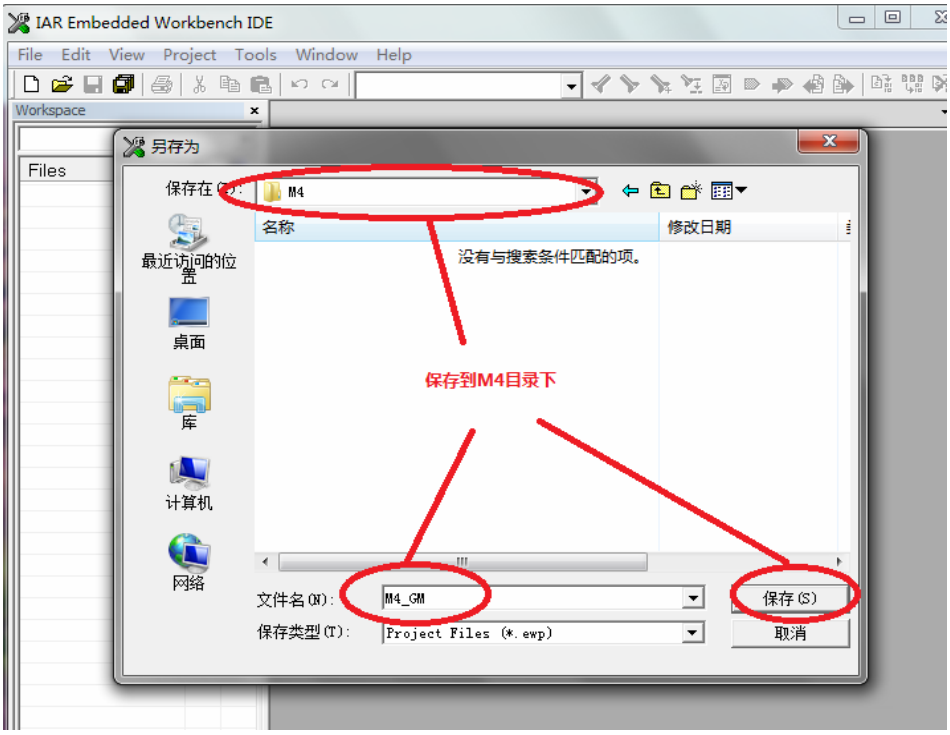
2.利用 IAR 環境建立專案工程（以光敏電阻感測器爲例）

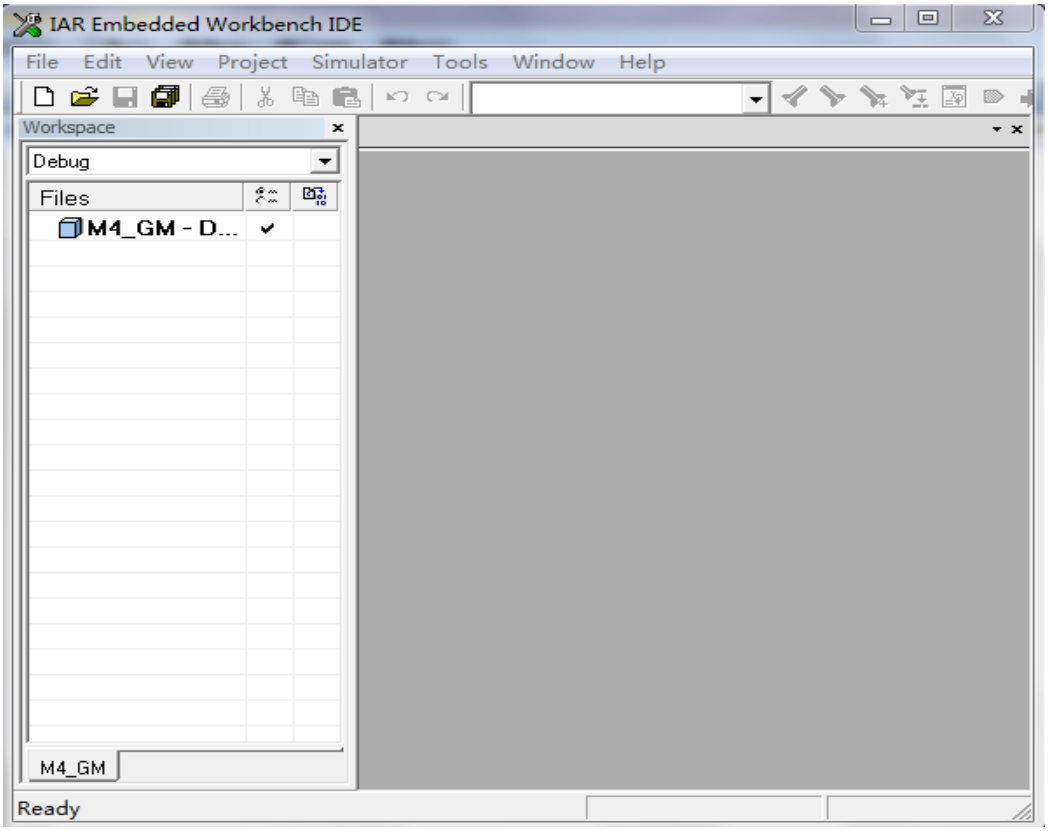
打開安裝好的 IAR IDE，如下圖所示：


IAR Embedded Workbench IDE
File Edit View Project Tools Window Help
Workspace
Files
Ready

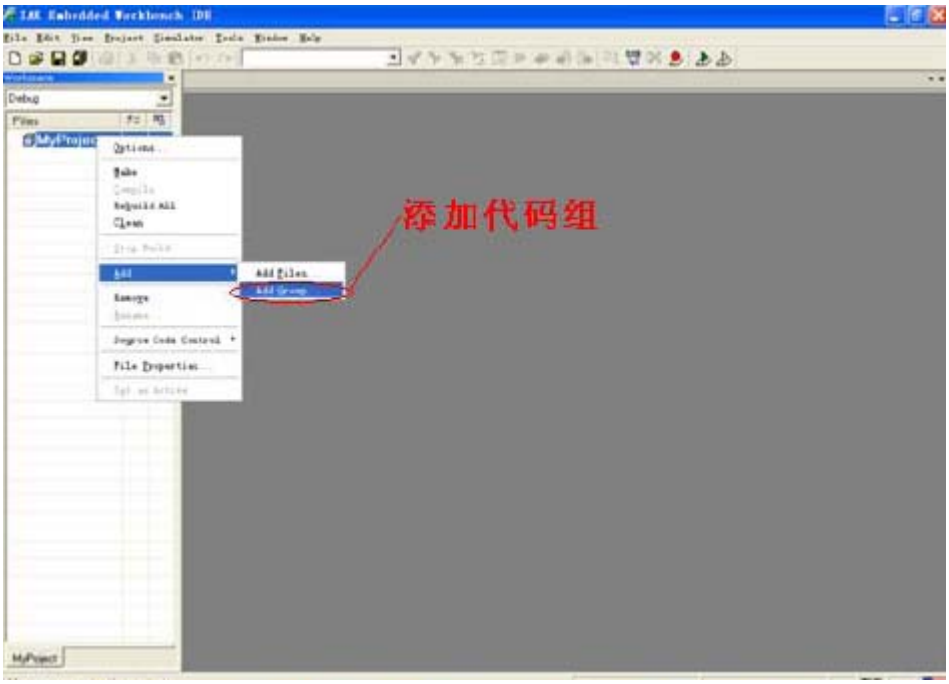

IAR Embedded Workbench IDE
File Edit View Project Tools Window Help
Workspace
Files
Add Files...
Add Group...
Import File List...
Edit Configurations...
Remove
Create New Project...
Add Existing Project...
Options... Alt+F7
Source Code Control
Make F7
Compile Ctrl+F7
Rebuild All
Clean
Batch build... F8
Stop Build Ctrl+Break
Download and Debug Ctrl+D
Debug without Downloading
创建一个新的项目


IAR Embedded Workbench IDE
File Edit View Project Tools Window Help
Workspace
Files
Create New Project
Tool chain:
8051
Project templates:
Empty project
Device-specific project templates
Generic project templates
Library project templates
ROM monitor
Description:
Creates an empty project.
OK
Cancel
建立空的項目
Ready

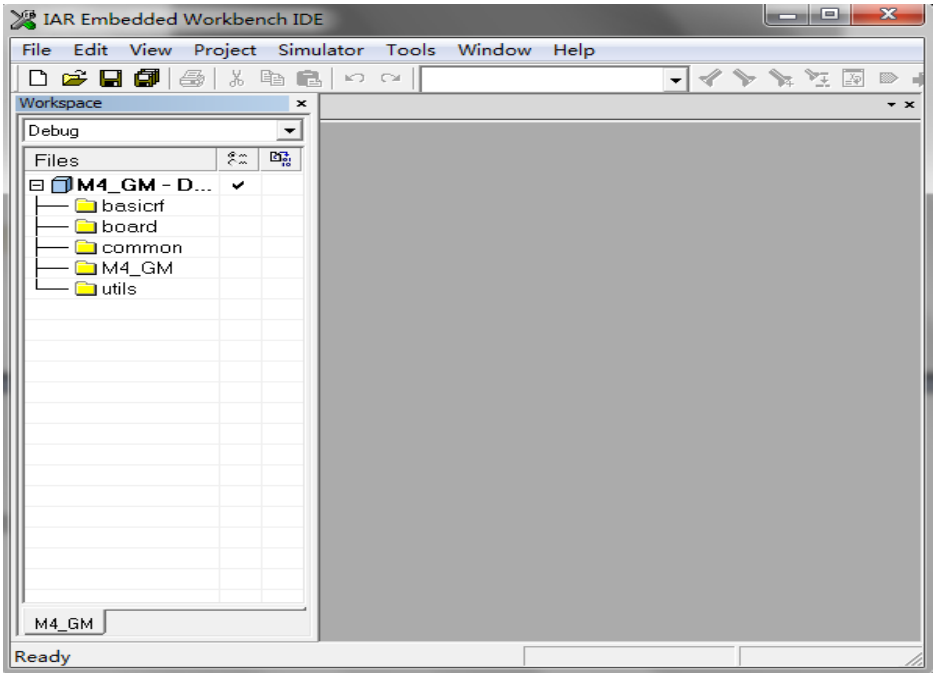

IAR Embedded Workbench IDE
File Edit View Project Tools Window Help
Workspace
Files
另存为
保存在(I):
M4
名称
修改日期
没有与搜索条件匹配的项。
最近访问的位置
桌面
库
计算机
网络
保存到M4目录下
文件名(N):
M4_GM
保存(S)
保存类型(T):
Project Files (*.ewp)
取消



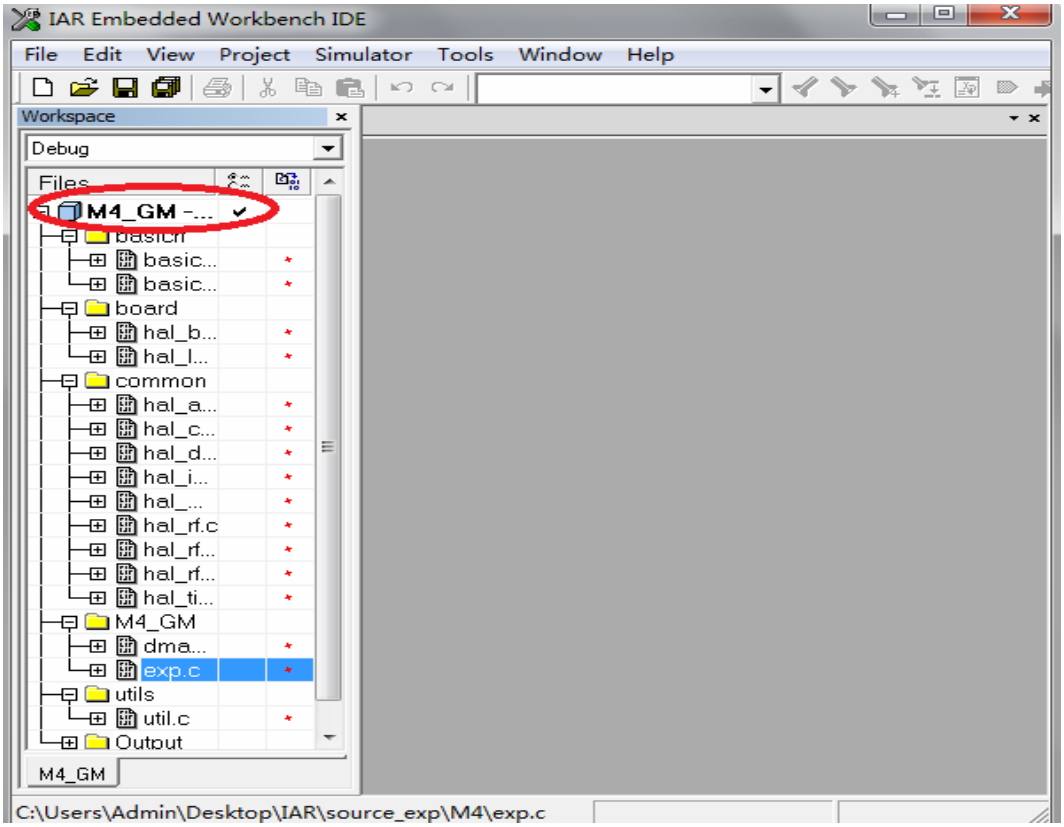
用戶可以向項目中添加*.c 文件；也可以添加代碼組；對代碼進行分組可以很好的進行原始程式碼的管理，也有助於生成較好的目標代碼，如下圖所示：



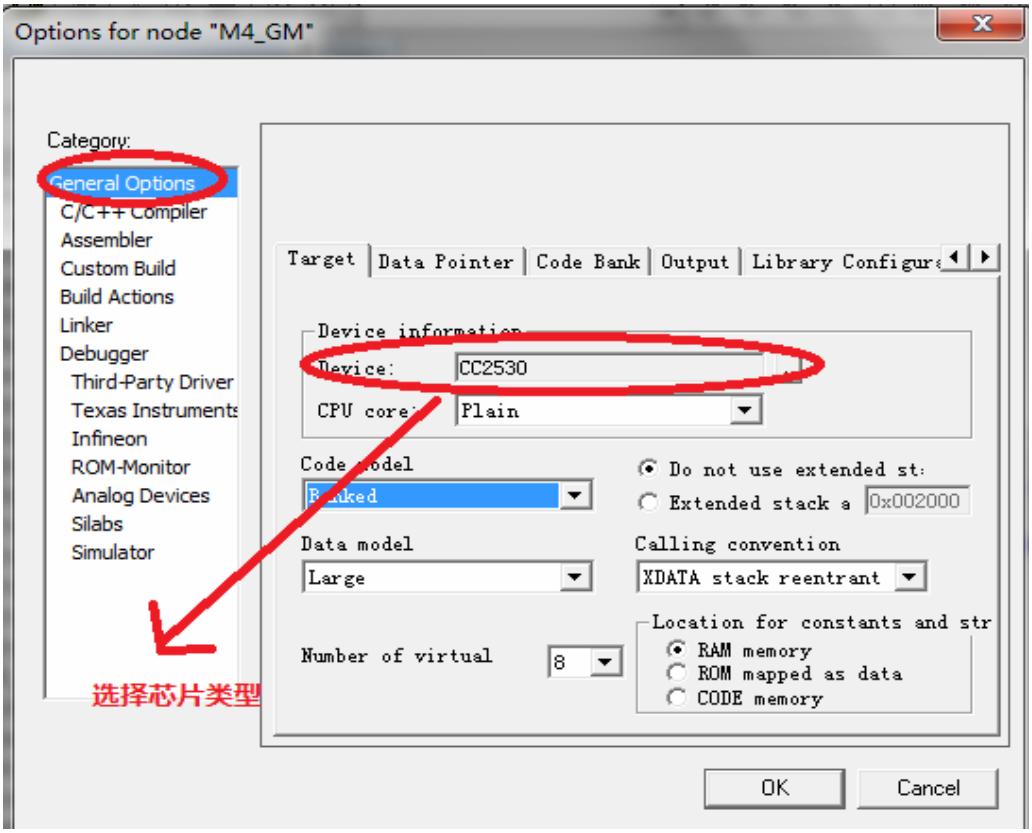
最後生成代碼組結構，如下圖所示：

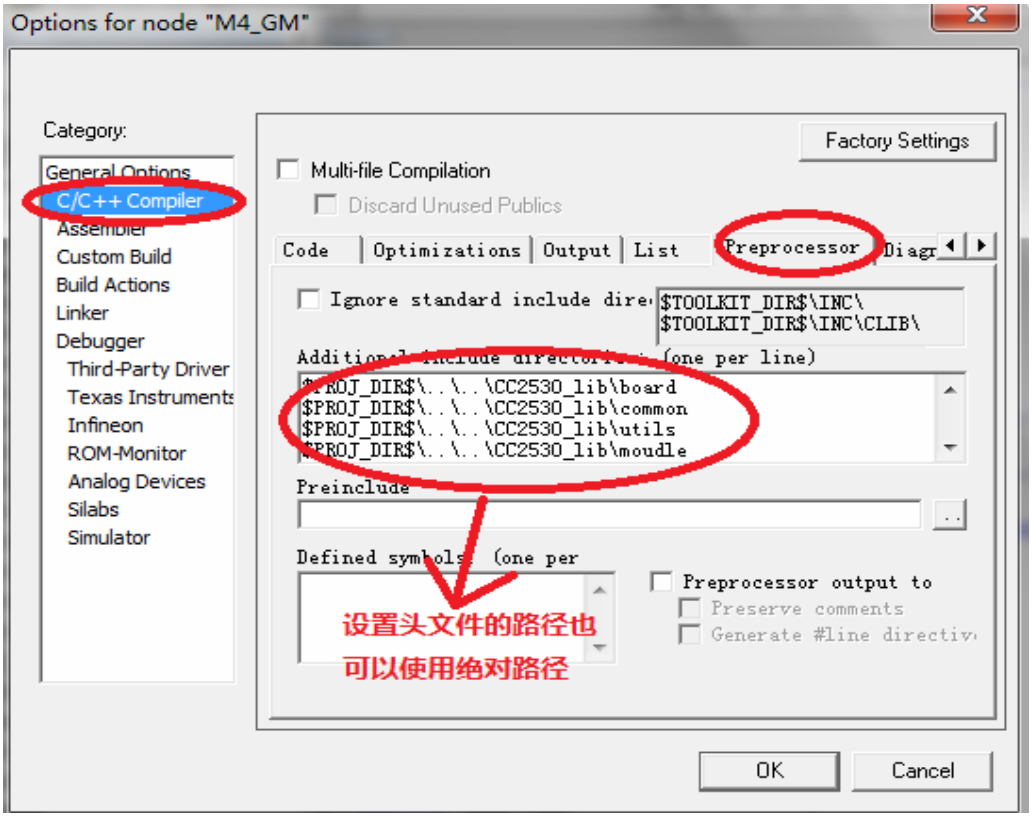


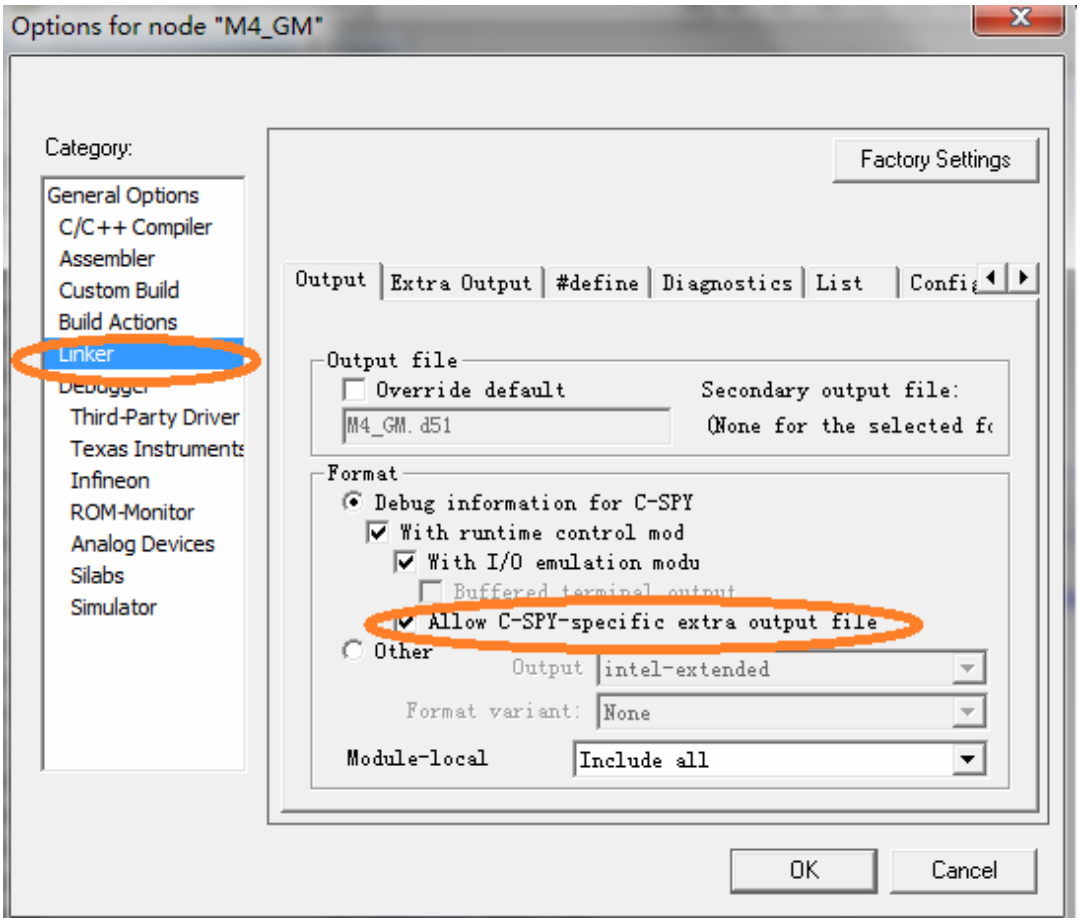
接著在對應代碼組中添加我們已經準備好的庫檔和自己編寫的代碼，M4_GM 添加的是 moudle 下的 dma_m4.c 和 exp.c 文件，如下圖所示：

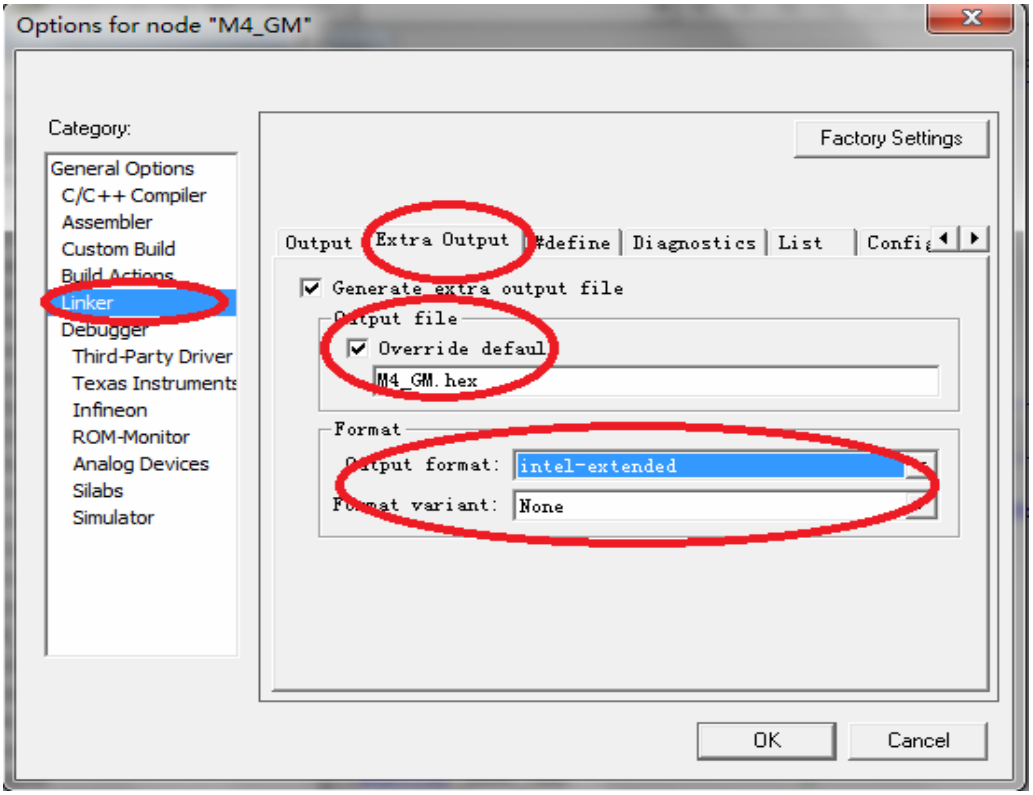


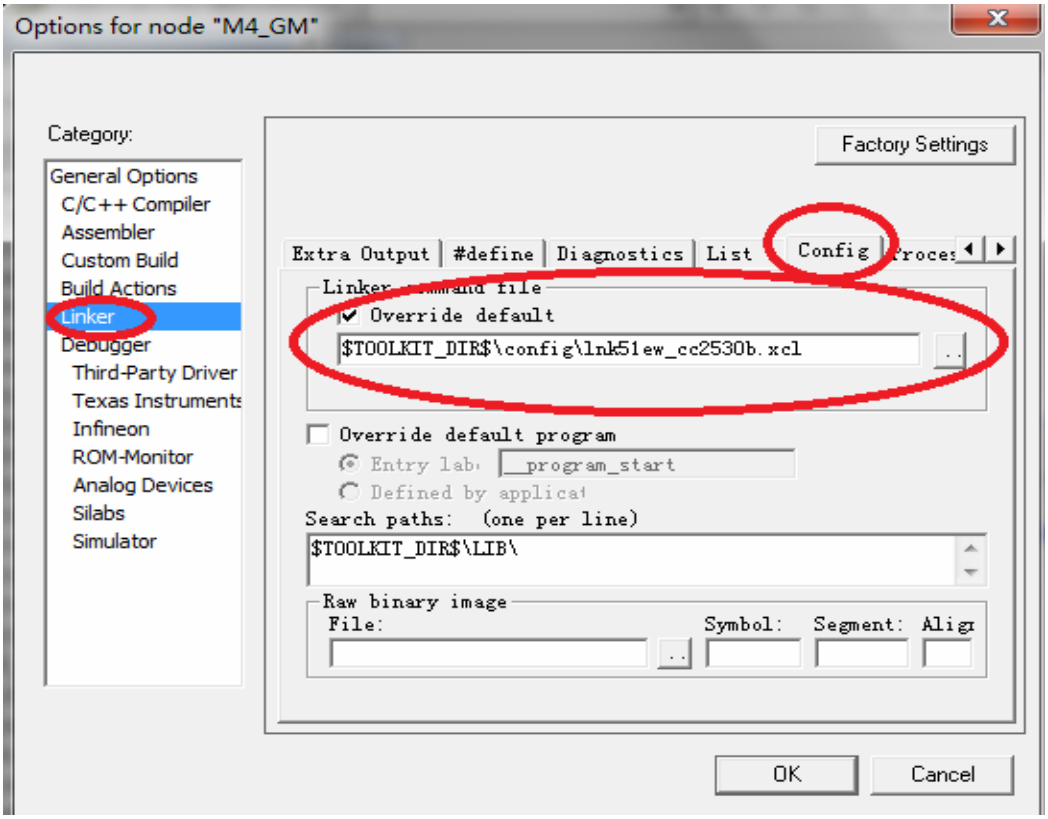
在上圖紅色框地方滑鼠右鍵，選擇 Option，如下圖所示：

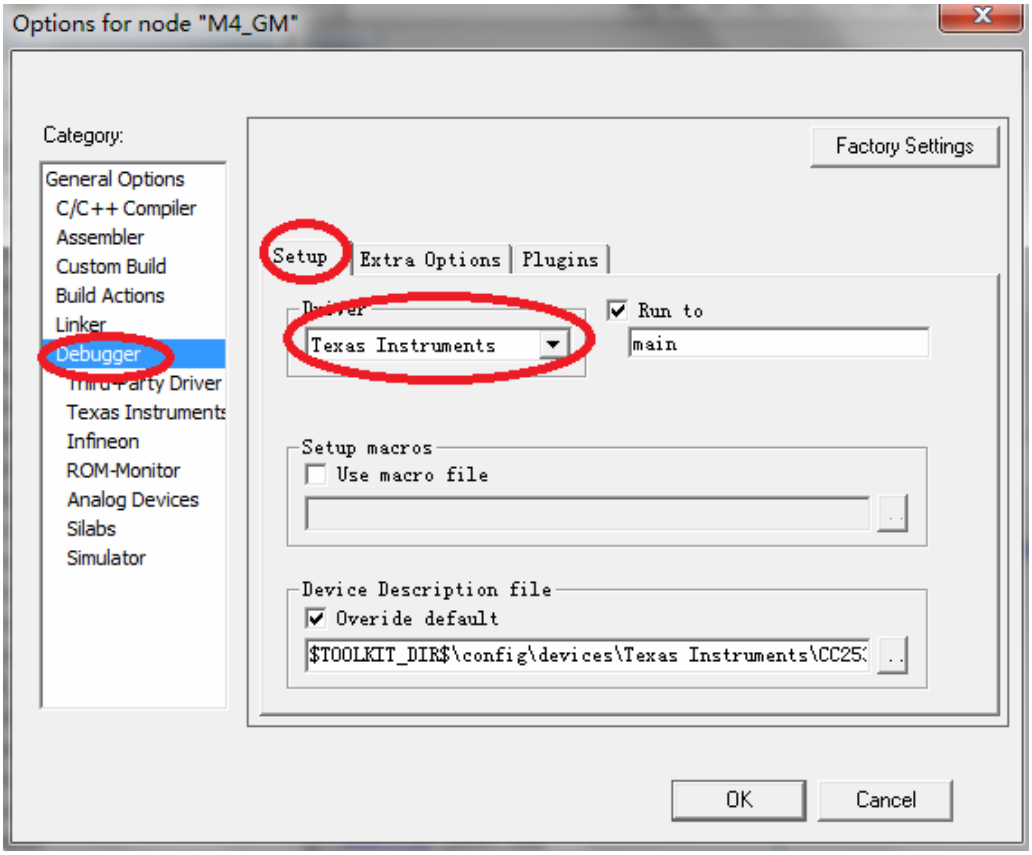

Options for node "M4_GM"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Debugger
Third-Party Driver
Texas Instruments
Infineon
ROM-Monitor
Analog Devices
Silabs
Simulator
Target
Data Pointer
Code Bank
Output
Library Configur
Device information
Device:
CC2530
CPU core:
Plain
Code model
Banked
Do not use extended st:
Extended stack a
0x002000
Data model
Large
Calling convention
XDATA stack reentrant
Number of virtual
8
Location for constants and str
RAM memory
ROM mapped as data
CODE memory
选择芯片类型
OK
Cancel


Options for node "M4_GM"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Debugger
Third-Party Driver
Texas Instruments
Infineon
ROM-Monitor
Analog Devices
Silabs
Simulator
Factory Settings
Multi-file Compilation
Discard Unused Publics
Code
Optimizations
Output
List
Preprocessor
Ignore standard include dire
$TOOLKIT_DIR$\INC\
$TOOLKIT_DIR$\INC\CLIB\
(one per line)
$PROJ_DIR$\..\..\CC2530_lib\board
$PROJ_DIR$\..\..\CC2530_lib\common
$PROJ_DIR$\..\..\CC2530_lib\utils
$PROJ_DIR$\..\..\CC2530_lib\moudle
Preinclude
Defined symbols (one per
Preprocessor output to
Preserve comments
Generate #line directiv
设置头文件的路径也
可以使用绝对路径
OK
Cancel

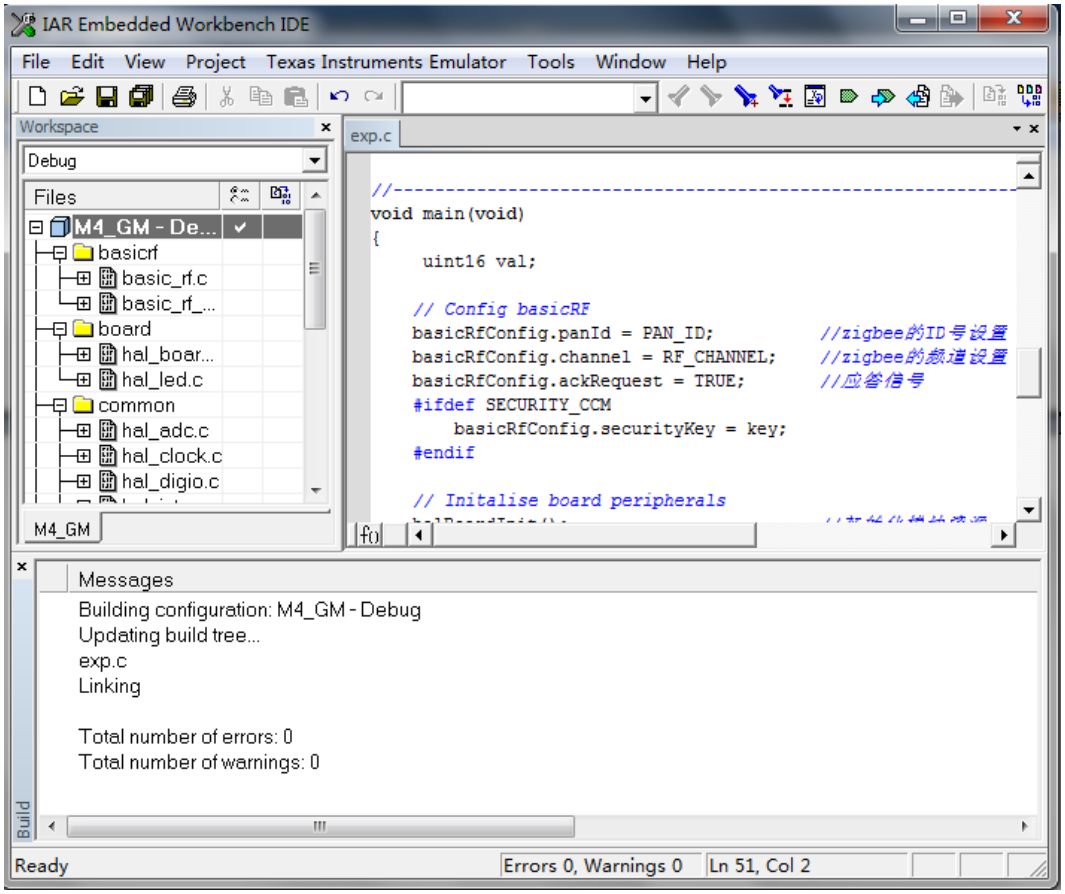

Options for node "M4_GM"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Third-Party Driver
Texas Instruments
Infineon
ROM-Monitor
Analog Devices
Silabs
Simulator
Factory Settings
Output
Extra Output
#define
Diagnostics
List
Output file
Override default
M4_GM.d51
Secondary output file:
(None for the selected f
Format
Debug information for C-SPY
With runtime control mod
With I/O emulation modu
Allow C-SPY-specific extra output file
Other
Output
intel-extended
Format variant:
None
Module-local
Include all
OK
Cancel


Options for node "M4_GM"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Debugger
Third-Party Driver
Texas Instruments
Infineon
ROM-Monitor
Analog Devices
Silabs
Simulator
Factory Settings
Output
Extra Output
#define
Diagnostics
List
Generate extra output file
Override defaul
M4_GM.hex
Format
intel-extended
None
OK
Cancel

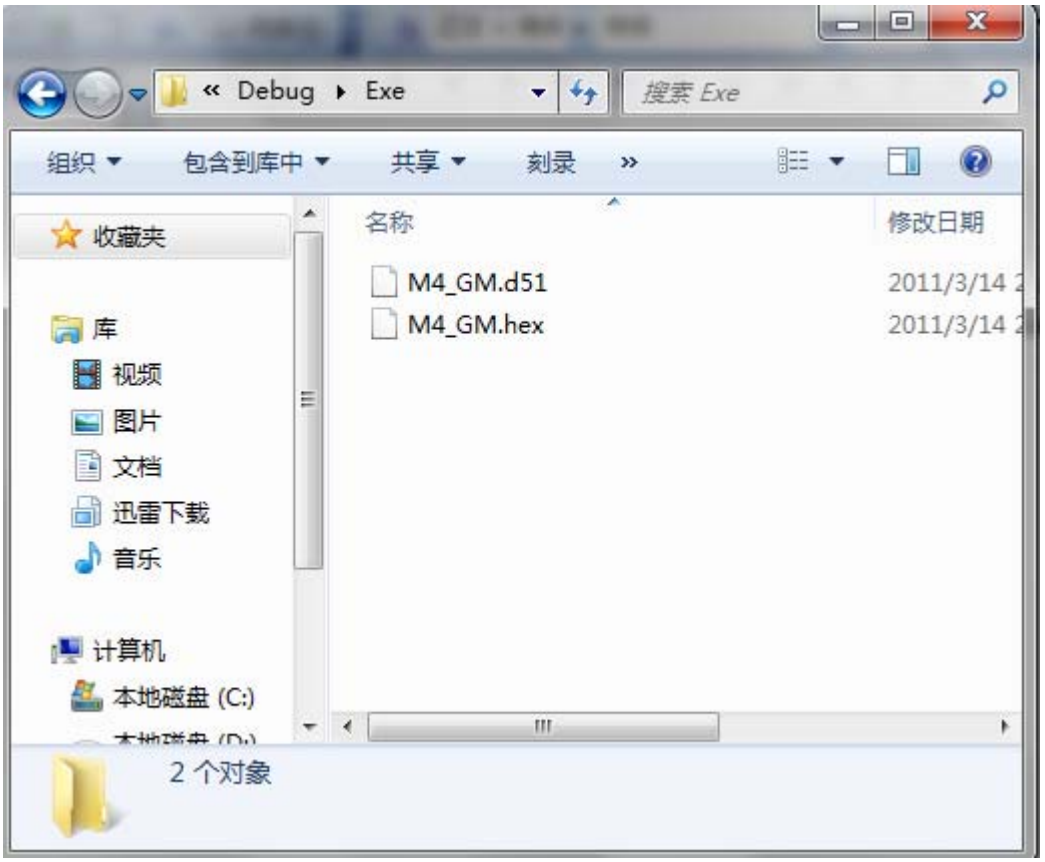

Options for node "M4_GM"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Debugger
Third-Party Driver
Texas Instruments
Infineon
ROM-Monitor
Analog Devices
Silabs
Simulator
Factory Settings
Extra Output
#define
Diagnostics
List
Config
Override default
$TOOLKIT_DIR$\config\lnk51ew_cc2530b.xcl
Override default program
__program_start
Search paths: (one per line)
$TOOLKIT_DIR$\LIB\
Raw binary image
File:
Symbol:
Segment:
OK
Cancel



配置完成後，點選 OK，然後編譯，成功如下圖所示：



最後生成的 M4_GM.hex，在 M4\Debug\Exe 下，如下圖所示：



最後將 hex 檔透過模擬器燒錄進晶片內。

3.原始程式碼分析（以光敏電阻感測器爲例）

透過上一章節瞭解光敏電阻感測器模組的原理和模組的原理圖後，我們知道模組採集的是類比信號，需要透過 AD 轉化成數位信號，這樣就可以進行資料的提取與傳輸，具體程式如下：

AD 轉換程式：

```
//-------------------------------------------------------------------
// Filename: dma_m4.c
// Description: dma_m4 library (A/D)
//-------------------------------------------------------------------
//-------------------------------------------------------------------
// INCLUDES
```

```
//------------------------------------------------------------------
#include "hal_defs.h"
#include "hal_cc8051.h"
#include "hal_mcu.h"
#include "hal_board.h"
#include "hal_digio.h"
#include "hal_adc.h"
#include "dma_m4.h"

//------------------------------------------------------------------
// @fn          dma_m4_Init
// @brief       Set up port dma_m4 ad
// @return    none
//------------------------------------------------------------------
void dma_m4_Init(void)
{
    MCU_IO_PERIPHERAL(HAL_BOARD_IO_ADC_PORT,
HAL_BOARD_IO_ADC_CH);

}
//------------------------------------------------------------------
// @fn          dma_m4_GetValue
// @brief     Get this dma_m4 module value
// @param     none
// @return    none
//------------------------------------------------------------------
uint16 dma_m4_GetValue(void)
{
    uint16 adcValue;
```

```
    adcValue=adcSampleSingle(ADC_REF_AVDD,ADC_12_BIT,
                                        HAL_BOARD_IO_ADC_CH);
    return adcValue;
}
```

資料的提取以及 ZigBee 無線發送資料程式：

```
//------------------------------------------------------------------
// INCLUDES
//------------------------------------------------------------------
#include "hal_defs.h"
#include "hal_cc8051.h"
#include "hal_int.h"
#include "hal_mcu.h"
#include "hal_board.h"
#include "hal_led.h"
#include "hal_rf.h"
#include "basic_rf.h"
#include "hal_uart.h"
#include "dma_m4.h"
//------------------------------------------------------------------
// CONSTANTS
//------------------------------------------------------------------
#define RF_CHANNEL                18        // 2.4 GHz RF channel

// BasicRF address definitions
#define PAN_ID                    0x1111
#define SEND_ADDR                 0x2222
#define RECV_ADDR                 0x3333
#define APP_PAYLOAD_LENGTH        32
```

```
// Application states
#define IDLE                    0
#define SEND_CMD                1

//-------------------------------------------------------------------
// LOCAL VARIABLES
//-------------------------------------------------------------------
static uint8 pTxData[APP_PAYLOAD_LENGTH]; //定義發送緩衝區的大小
static basicRfCfg_t basicRfConfig;          //zigbee 的配置資訊

//-------------------------------------------------------------------
void main(void)
{
    uint16 val;

  // Config basicRF
  basicRfConfig.panId = PAN_ID;             //zigbee 的 ID 號設置
  basicRfConfig.channel = RF_CHANNEL;       //zigbee 的頻道設置
  basicRfConfig.ackRequest = TRUE;          //應答信號
  #ifdef SECURITY_CCM
      basicRfConfig.securityKey = key;
  #endif

  // Initalise board peripherals
  halBoardInit();                           //初始化模組資源
  halLedSet(1);

  basicRfConfig.myAddr = SEND_ADDR;         //設置本機位址爲：發送地址
```

```
    if (basicRfInit(&basicRfConfig) == FAILED){}    //檢測 zigbee 的參數是否配置成功

    while (1)
    {
        dma_m4_Init();
        while(1)
        {
            halLedClear(2);
            halMcuWaitMs(100);

            val = dma_m4_GetValue();
            pTxData[0]='2';     //光敏
            pTxData[1]=val/10000;
            pTxData[2]=val%10000/100;
            pTxData[3]=val%100;
            basicRfSendPacket(RECV_ADDR, pTxData,4);

            halLedSet(2);
            halMcuWaitMs(100);
        }
    }
}
```

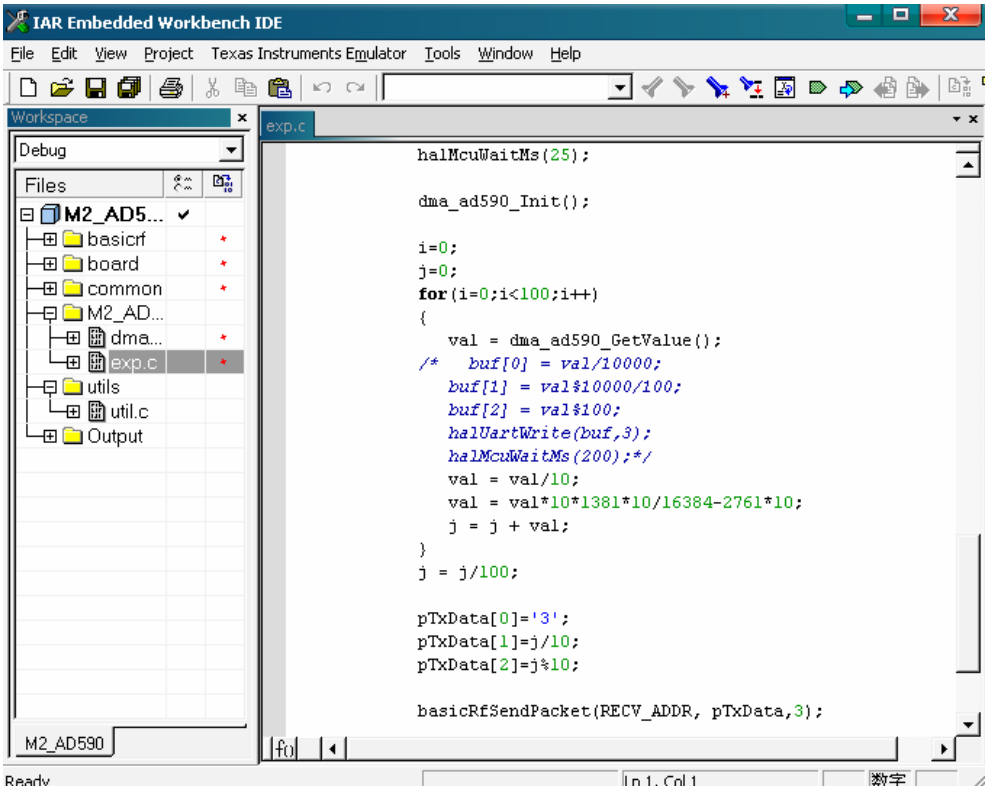
其餘的感測器模組開發環境的搭建與上相同在這裏就不一一詳解了，在附錄光碟內有詳細的介紹。

注意：由於類比溫度感測器（AD590）在硬體上我們沒有使用精度高的電阻，以及 AD590 感測器本身的精度非常高，因此透過 AD 值所算出來的溫度值會與正確

的溫度有比較大的差距。但是我們可以通過程式中演算法的調節到正常的溫度值。下面具體來介紹如何修正溫度值。

5-3 模擬溫度感測器的溫度修正方法

首先，我們利用 IAR 軟體打開，AD590 的工程目錄（IAR/source_exp/M2_AD590），如下圖所示：



其次，將以上的程式編譯完後，將 M2_AD590.hex 燒寫進 ZB2530-01 模組中去，然後記錄下在 android 介面上顯示的錯誤的溫度值，比如：248 度。

然後，根據代碼中的溫度演算法（如下所示），進行修改。

```
val = val*10*1381*10/16384-2761*10;
```

如果現在的正確溫度是 28 度，那麼將剛剛記錄下的錯誤溫度值（248 度）與正確溫度值相減（248-28 = 220）就得到相差的值（220），最後將演算法改成如下所示：

```
val = val*10*1381*10/16384-(2761+220)*10;
```

這樣修改後，編譯，再將燒錄檔燒寫進模組中，就可以正常檢測溫度值了。

附錄一 ZigBee Debugger 模擬器介紹

1-1 介紹

目前TI 的CC 系列射頻晶片已經成爲主流的低耗電射頻解決方案，尤其是在ZigBee 應用中，CC2430、CC2431、CC2530、CC2531 爲眾多國內實驗教學和公司產品開發所使用。這一系列產品整合了一個8051處理器，使得可以單晶片組成ZigBee 節點，方便了擴展及應用。

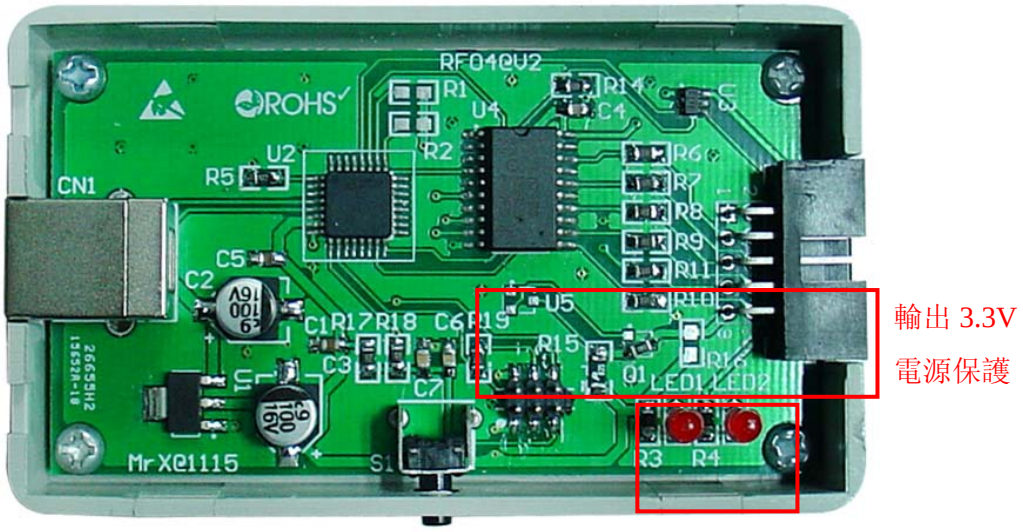
用於CC系列的模擬器主要有三種，分別是CC-DEBUGGER、SmartRF04EB、和SmartRF05EB 。其中CC-DEBUGGER 是國外最普遍應用的。而SmartRF04EB和SmartRF05EB 是和開發平台一起研製的，相當於評估板的一部分，而CC-DEBUGGER 和SmartRF05EB 硬體結構類似。國內TI 原版CC-DEBUGGER售價昂貴且購買管道較少，SmartRF04EB/SmartRF05EB 必須購買整套評估板，故價格更貴。

爲了解決這個問題，本公司推出了基於SmartRF04EB 評估板的ZigBee DEBUGGER 模擬器，該模擬器完全尊重原版的硬體設計，保留了所有功能。並且在電源部分進行了創新的設計，提供了原版沒有的目標板供電保護功能。

1-2 特點

- 尊重原版設計，支援所有功能，品質更有保障。
- 設計有供電保護電路，當模擬器給目標供電時能夠穩定輸出3.3V 電壓，且當電流超過0.3A 會自動關斷，以免造成對電腦USB埠、模擬器、和目標板的損害。
- 設計有電平相容電路，當目標板不從模擬器取電而使用另外的2V-3.6V 的任意電壓供電時，模擬器可以完全正常工作。
- 設計有邏輯介面保護電路，不會因爲誤插或者靜電等原因使模擬器或者目標板晶片燒掉。
- 尺寸小巧，設計精美，穩定性高。
- 支援眾多CC 系列晶片：CC1110/CC1111/CC2430/CC2431/CC2510/CC2511/CC2530/CC2531 等等。
- 下載速度高達150KB/S。
- 自動辨識晶片和自動控制下載速度。

- 可透過TI 相關軟體更新最新版本韌體，使模擬器能夠支援最新的晶片。目前版本能夠很好的支援CC25XX 系列晶片，解決了之前版本眾多問題。
- 官方標準的10PIN 介面。
- 具有一個模擬器執行指示燈和一個目標板連接指示燈。
- 可IAR for 8051 整合開發環境無縫連接。支援最新的IAR Embedded Workbench for MCS-51 7.51A
- 支援最新版本的SmartRF Flash Programmer 軟體。
- 支援最新版本的SmartRF Studio 軟體。
- 支援最新版本的IEEE Address Programmer 軟體。
- 支援最新版本的Packet Sniffer 軟體。
- 每個模擬器具有唯一的ID 號，一台電腦能夠同時使用多台模擬器。
- 產品含USB 連接線、10PIN 扁平排線和相關資料及軟體光碟。



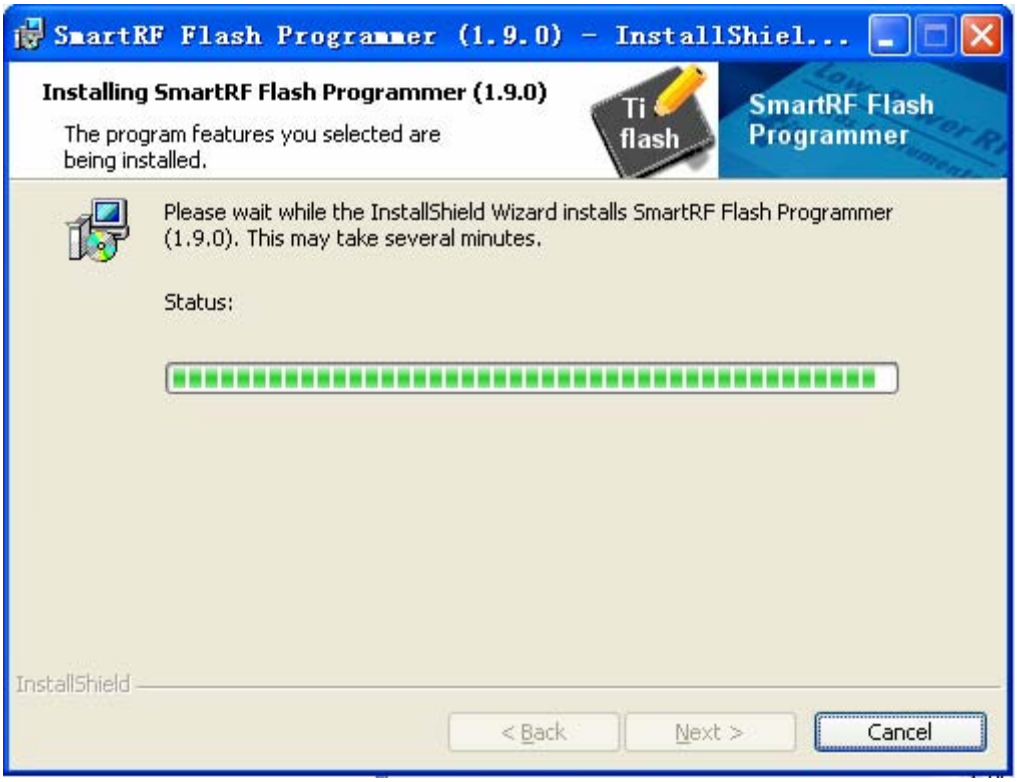
USB連接線

10 Pin扁平排線

1-3 使用說明

1、驅動安裝

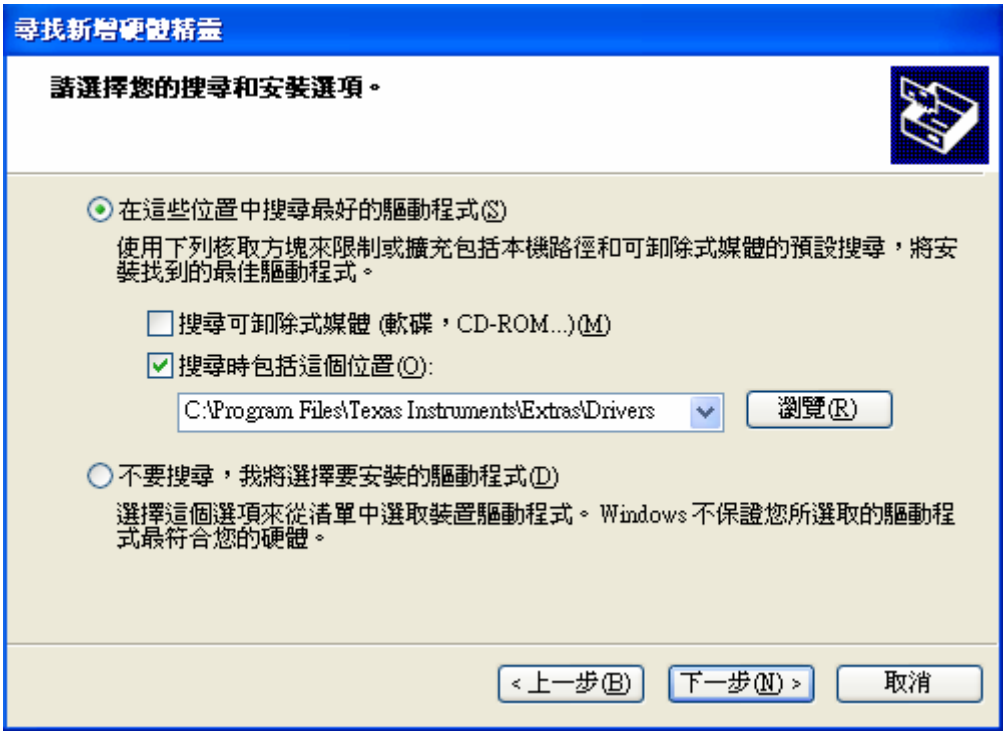
請先安裝光碟中的軟體SmartRF Flash Programmer 1.9.0。



建議全部按照預設選項安裝即可。

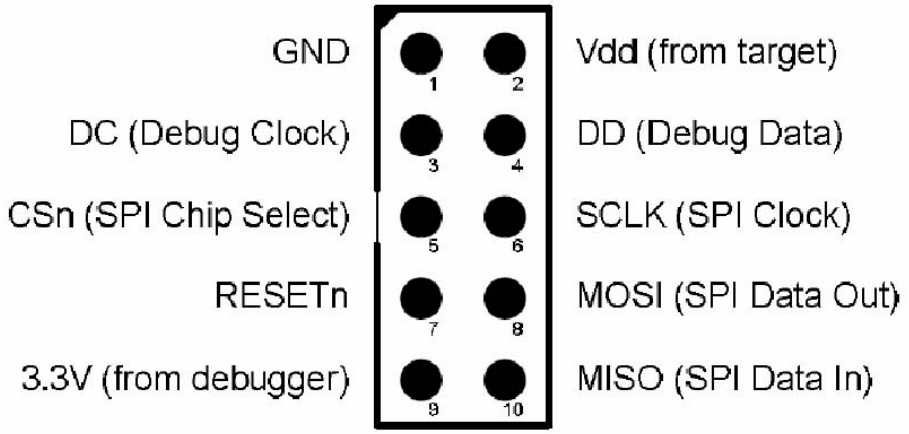
將模擬器透過USB 連接線接入電腦，會彈出尋找硬體精靈視窗：

選擇“從清單或特定位置安裝（進階）”。



選擇“在這些位置上搜尋最好的驅動程式”，勾選“在搜尋時包括這個位置”，點擊瀏覽按鈕，指定我們剛剛安裝的SmartRF Flash Programmer 的軟體目錄下的C:\Program Files\Texas Instruments\Extras\Drivers目錄。然後點擊下一步即可完成驅動安裝。

2、硬體連接



Pin	Function	Note
1	**GND**	
2	**VDD**	Used to set correct voltage for the voltage level converter
3	**Debug Clock (DC)**	
4	**Debug Data (DD)**	
5	CSn	
6	SCLK	
7	**Reset_N**	
8	MOSI	
9	3.3V VDD, alt. NC	Delivers VDD from SmartRF®04EB
10	MISO	

注意1：在使用模擬器之前，請確認目標板的模擬介面與此官方標準介面相同，才可以使用，否則可能造成模擬器或目標板的損壞！

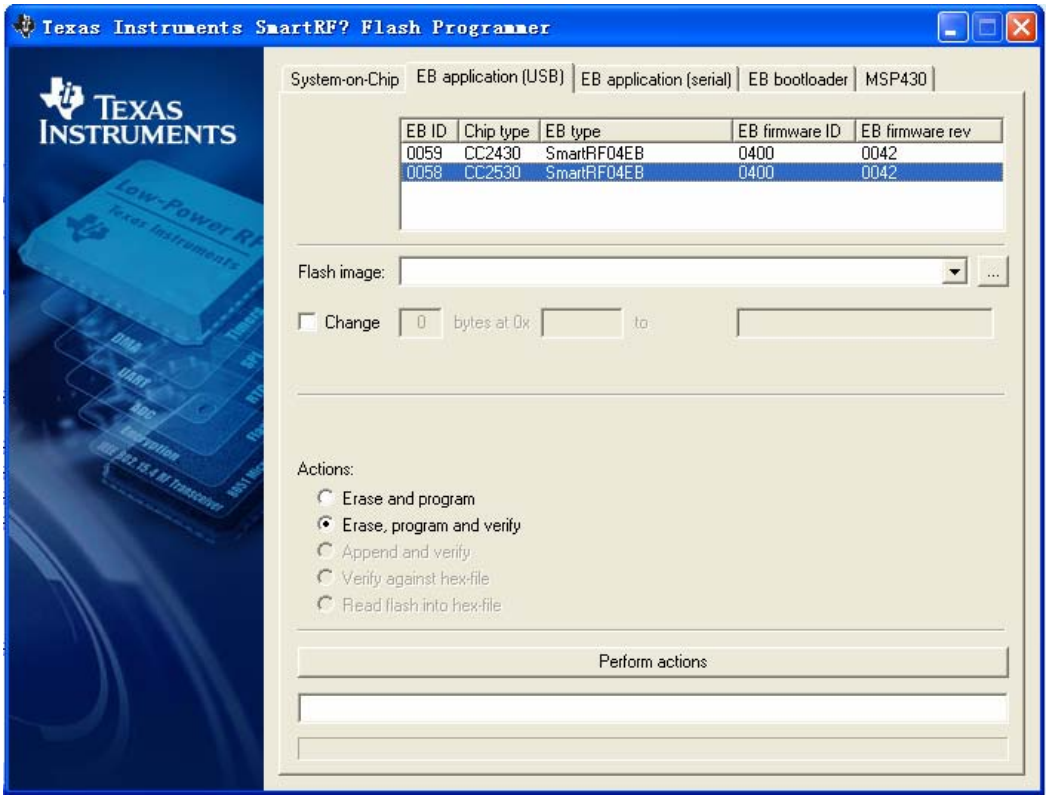
注意2：如果目標板使用單獨的2.0V-3.6V電源供電，則必須將目標板上9 腳和目標板電源斷開，否則可能會影響正常使用。

3、SmartRF Flash Programmer 使用

SmartRF 快閃記憶體編程器可用於對德州儀器 (TI) 射頻片上系統器件中的快閃記憶體進行編程，並對 SmartRF04EB 、SmartRF05EB 和 CC2430DB 上找到的 USB MCU 中的軔體進行升級。此外，快閃記憶體編程器還可透過 MSP-FET430UIF 和 eZ430 對 MSP430 器件的快閃記憶體進行編程。

在一般情況下，模擬器連接到電腦安裝驅動後，打開SmartRF Flash Programmer 軟體，會自動辨識出模擬器，當模擬器連接到目標板後，按一下模擬器的RESET 按鈕，就會自動辨識出模擬器連接的晶片。使用這個軟體可以直接給晶片下載程式或者指定晶片的物理位址。

更多資訊請參考（預設在C:\Program Files\Texas Instruments\Documentation 目錄下）SmartRF Flash Programmer User Manual.pdf 文檔。



4、IAR EW8051 使用

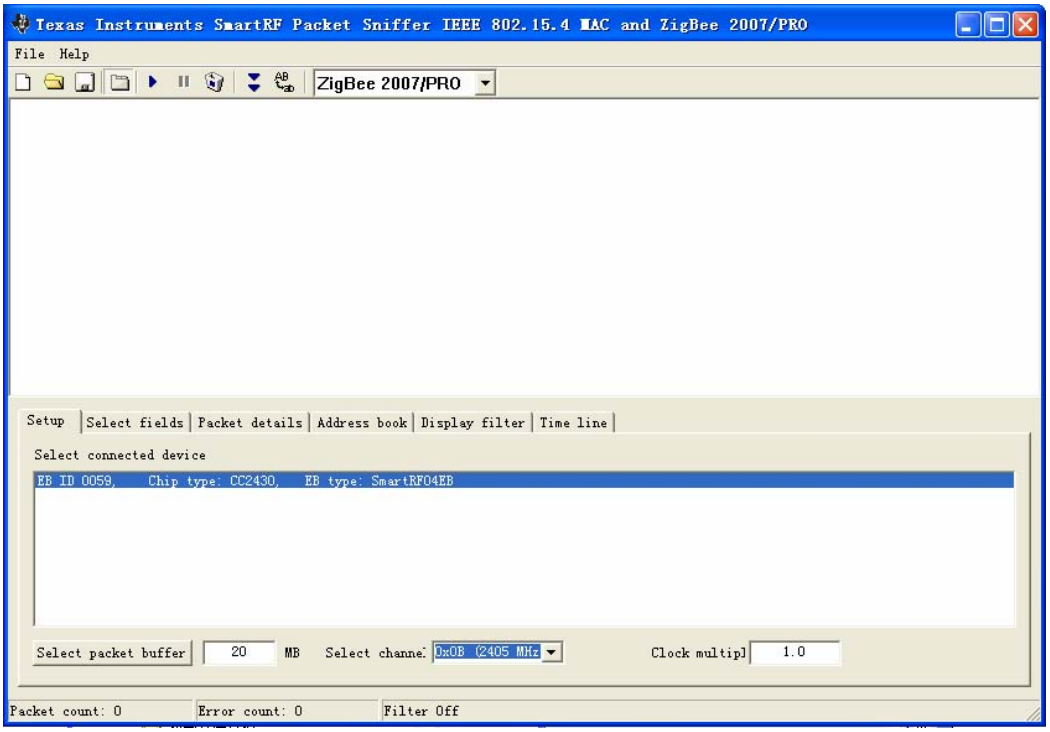
7.51A 爲最新版本，開啓不同版本範例程式碼可以選擇不同版本程式，如果開發自己的新程式，建議使用7.51A。

5、Packet Sniffer 使用

Packet Sniffer 是一套PC端軟體網路封包攔截分析軟體。該射頻器件透過 USB 與 PC 相連。支援各種射頻協定。封包攔截分析軟體可以對封包進行過濾和解碼，並以簡便的方式顯示它們，同時提供用於過濾和儲存爲二進位檔格式的選項。

安裝光碟中的Packet Sniffer (2.12.3) ，建議直接按照預設選項安裝即可。

在一般情況下，模擬器連接到電腦安裝驅動並當模擬器連接到目標板後，打開 Packet Sniffer 軟體，會自動辨識出模擬器。如果沒有辨識出則需按一下模擬器的 RESET 按鈕，就會自動辨識出。



使用這個軟體可以透過一個目標板對當前的ZigBee 網路進行協定分析。

更多資訊請參考（預設在C:\Program Files\Texas Instruments\Documentation 目錄下）SmartRF Packet Sniffer User Manual.pdf 文檔。

DMATEK

ARM晶片發展系列

單板型7吋HMI人機介面

● 操作應用手冊

作業系統：Android 2.1 / 2.2

適用平台：HMI700-6410S

目録

目錄

目錄

第〇章
導　讀

0-1　如何開始

在本手冊中，筆者將于第一章中簡述 HMI700-6410S 開發平台，並于第二章說明硬體電路，讀者可以由這兩章的內容來瞭解 HMI700-6410S 開發平台的硬體電路設計。第三章中將簡述如何使用 HMI700-6410S 開發平台。

第四章是有關 Android 的核心編譯介紹；第五章則爲讀者介紹如何燒錄和啓動 Android 2.1 系統；第六章則著重介紹了 Android 的系統程式及一些主要 AP 的測試；第七章則主要講解了 NDK 環境搭建及編譯；最後附錄一是有關 HMI700-6410S 外觀結構圖。

0-2 光碟內容說明

在 HMI700-6410S 開發平台的產品光碟中包含了許多實用的東西，讀者可以從產品光碟的 README 檔瞭解光碟的內容，主要包含以下的內容：

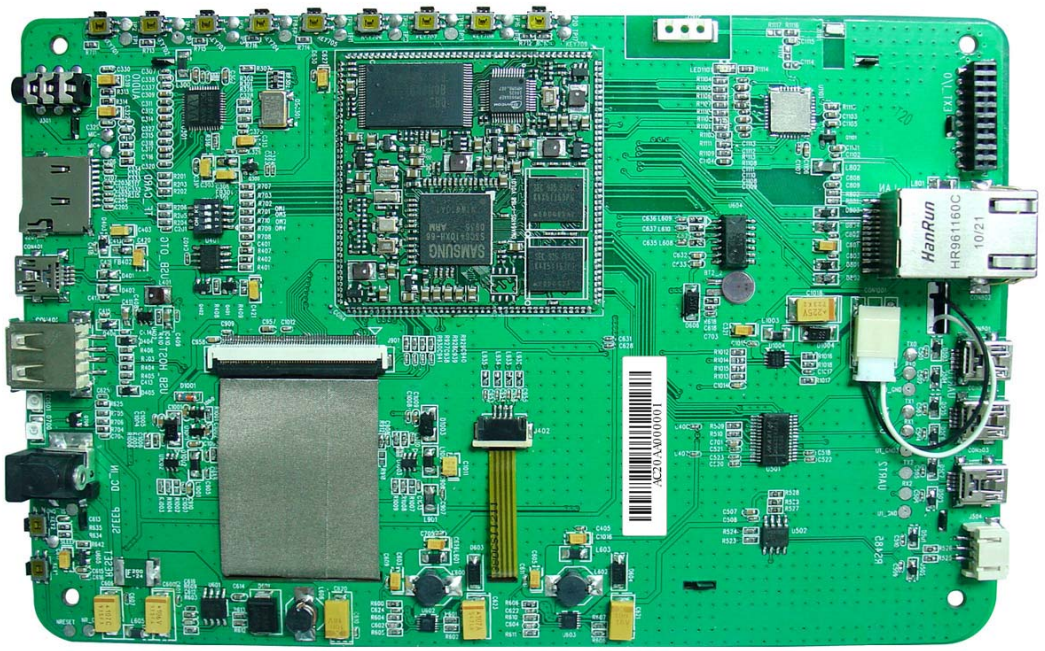
上層目錄	說明
User Manual	平台的操作手冊 PDF 檔。
SCH	平台的電路圖，以 PDF 格式的檔案提供。
Datasheet	平台所採用電子元件的 Datasheet 檔。
Android	包含以下子目錄： 1. **Cross compiler**。 2. **U-boot**。 3. **U-Boot_MMC**：支援由 SD 卡啓動的 U-Boot 版本。 4. **Kernel**。 5. **Android**：Android 的程式碼，包含編譯 Android 的腳本檔等。 6. **Sample Image**：在這個目錄內應放置 u-boot.bin、zImage、ramdisk-uboot.img、system.img 及 userdata.img 等映射檔。 7. **NDK**：包含 Android NDK、cygwin 及一個 LED 控制的實驗範例。 8. **USB Driver** 9. **Lab**：包含以下項目 (1) **Driver**：驅動程式的程式碼。 (2) **LIB_SO**：libXXX.so 的程式碼及二進位檔案。 (3) **AP_SDK2.1**：以 Android 2.1 SDK 開發的實驗程式碼等。

Tools	包含以下項目： 1. 微軟公司的 **ActiveSync 4.5** 及 USB 驅動程式。 2. **ConvertZ**：這是繁簡中文轉換的工具。 3. 串列終端工具 **DNW** 及 USB 驅動程式。 4. **TFTPD32**：TFTP 伺服器的 Windows 版。 5. **IROM_Fusing_Tool**：製作啓動用 SD 卡的工具。

第一章
HMI700-6410S 人機介面簡介

1-1　人機介面外觀

外觀正面圖



外觀背面圖

1-2 簡介與特色

HMI700-6410S 採用 Samsung S3C6410 處理器，以 32 位元高速地 ARM 做爲主 CPU，OS 作業系統搭配 WinCE 6.0 / Android 1.6 / Android 2.1 爲主作業系統，本系統結合強而有力的主核心板，內嵌到主控板並與液晶觸控介面結合成一體化，所以本單板型 HMI 觸控人機介面能減少傳統式的人機介面的外殼厚度的三分之二，所以在外型輕薄之下，很方便結合各種環境下地使用觸控人機介面，使產品地應用設計變得更爲簡單，功能更加強大，很方便應用於如智能家居、醫療儀器及各種人機介面的介接與使用

HMI700-6410S 單板型 HMI 人機介面，採用 800×480 的 7 吋高清晰的液晶 LCD，並配以高精度地觸控板，同時具備了 100M 網路、高速 SD 卡存儲設備、USB

Host、USB Device、RS232、RS485 及音效輸出介面，並提供寬電壓的電源輸入工作方式，很方便搭配各種環境下地電源使用。

1-3　人機介面硬體資源

■ 硬體規格

- 中央處理器

  CPU：Samsung S3C6410XH-66，主頻為 667MHz

- 外部記憶體

＊ SDRAM 記憶體：

平台上提供 64MB＊2 片 Mobile DDR SDRAM ，共 128MB（Option）

平台上提供 128MB＊2 片 Mobile DDR SDRAM ，共 256MB 的標準配置

＊ NAND Flash：

SLC NAND Flash：128MByte 記憶體（Option）

SLC NAND Flash：256MByte 記憶體（標準配置）

MLC NAND Flash：2G / 4G / 8G Byte 記憶體(Option)

- 網路介面

  一個 10/100M Ethernet，採用 DM9000AE，帶指示燈 RJ-45 介面

- USB 介面

  一個 USB HOST（USB 1.1 規範）介面，支援全速(12Mbps)或低速(1.5Mbps)傳輸

  一個 USB HS OTG（USB 2.0 規範）介面，最高支援 480Mbps 高速傳輸

- 串列埠

  三個三線制串列埠

- RS485

  一個 RS485

- 音效介面

採用 AC97 的介面晶片，立體聲音效輸出介面可接耳機

- 外擴 GPIO 介面

  板上內置一個 20PIN 接座，其中含一路 SPI 1 埠、4 通道 10bit ADC、12 個 GPIO 埠；可直接供給擴充外部設備用

- LCD 介面

  系統平台標準配備為800×480 / 7.0英吋TFT液晶螢幕，附加4線電阻式觸控功能

- SDIO WiFi 介面 **(Option)**

  標準的SDIO WiFi 設備，支援IEEE802.11I/b/g

- Micro SD（T-Flash）卡介面

  一個 T-Flash 存儲設備

- RTC 時鐘

  S3C6410 內部整合，外部提供可充 RTC 電池，無須更換

- 時鐘設計

  採用無源晶振模式設計

- Reset 電路

  採用手動重置和晶片重置相結合的方式，晶片採用 MAX811，重置穩定可靠

- 電源介面

  本電路採用工業級標準的寬電壓輸入，寬電壓輸入範圍：DC/5V -30V 2A 電源供電

- 平台其他功能

  * 一個電源指示 LED
  * 一個系統指示 LED
  * 增加系統按鍵：9 個
  * 增加 RTC 時鐘模組
  * 過電流保護：採用自我恢復保護元件

- 尺寸大小及重量

  PCB 大小：189mm×123mm

■ 工具和原始程式碼

- HMI700-6410S WinCE 6.0 版本的 IMAGE、SDK
- HMI700-6410S 人機介面使用手冊（pdf 格式）
- WinCE 實現軟體資料保存及註冊表保存功能，能夠在 WinCE 根目錄下創建 Residentflash 檔案夾，將 Nandflash 剩餘空間做為該檔案夾的存儲空間，實現資料的永久保存及註冊表永久保存功能。

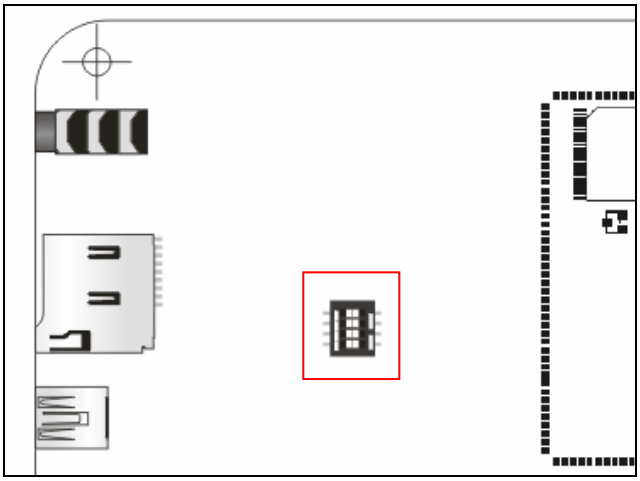
1-4　平台硬體資源分配

1-4.1　位址空間分配以及晶片選擇信號定義

HMI700-6410S 支援二種啓動模式：

- 一種是從 NAND FLASH 啓動；
- 一種是從 TF Card 啓動。

本平台的設計是透過底板上的指撥開關(如下圖紅框所示)來決定啓動模式的：



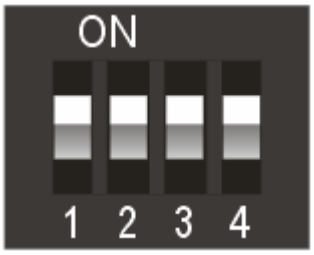
底板局部示意圖

- 若 NAND FLASH 爲 128MB，則將撥碼開關的第三腳撥到 ON，其他腳撥到 OFF 上，則從 NAND FLASH 啓動，如下圖：

示意圖

- 若 NAND FLASH 爲 256MB，則將撥碼開關的第三、第四腳撥到 ON，其他腳撥到 OFF 上，則從 NAND FLASH 啓動，如下圖：

- 若將撥碼開關的全部撥到 ON，則從 TF Card 啓動，如下圖：



示意圖

下圖爲 Memory 地址分配圖：

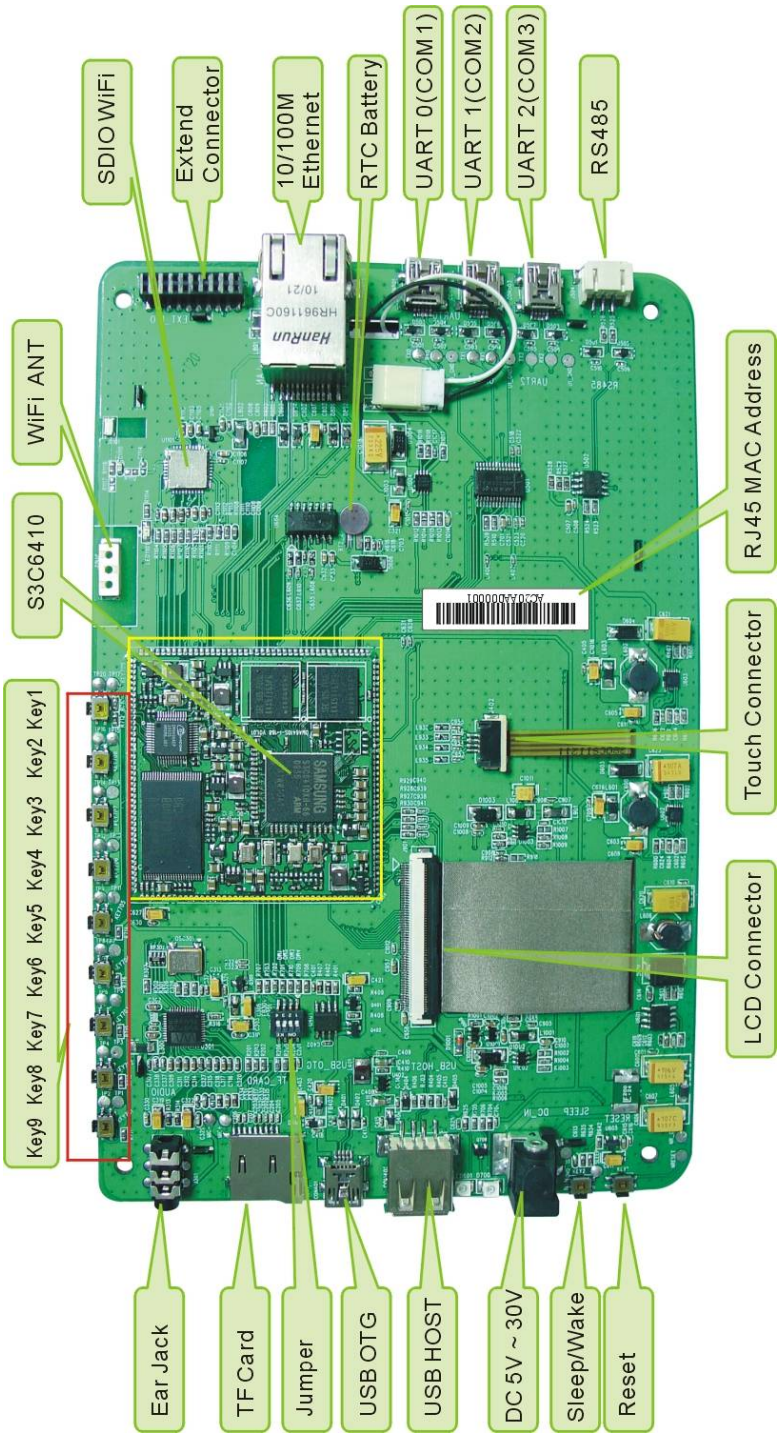
Start Address	End Address	Int. ROM	Stepping Stone (NAND Ctrl.)	SROM Ctrl.	One NAND Ctrl. 0	One NAND Ctrl. 1	DRAM Ctrl 1
0x00000000	0x07FFFFFF	O[1]	O[1]	O[1]	O[1]	-	-
0x08000000	0x0BFFFFFF	O	-	-	-	-	-
0x0C000000	0x0FFFFFFF	-	O	-	-	-	-
0x10000000	0x17FFFFFF	-	-	O	-	-	-
0x18000000	0x1FFFFFFF	-	-	O	-	-	-
0x20000000	0x27FFFFFF	-	-	O[2]	O[2]	-	-
0x28000000	0x2FFFFFFF	-	-	O[2]		O[2]	-
0x30000000	0x37FFFFFF	-	-	O	-	-	-
0x38000000	0x3FFFFFFF	-	-	O	-	-	-
0x40000000	0x47FFFFFF	-	-	-	-	-	-
0x48000000	0x4FFFFFFF	-	-	-	-	-	-
0x50000000	0x5FFFFFFF	-	-	-	-	-	O
0x60000000	0x6FFFFFFF	-	-	-	-	-	O

1-4.2　跳線說明

◆　跳線分配表：

元件名稱	說 明
撥碼開關	控制平台系統的啓動方式： ● 若 NAND FLASH 爲 128MB 時，則將撥碼開關的第三腳撥到 ON，其他腳撥到 OFF 上，則從 NAND FLASH 啓動。 ● 若 NAND FLASH 爲 256MB 時，則將撥碼開關的第三、第四腳撥到 ON，其他腳撥到 OFF 上，則從 NAND FLASH 啓動。 ● 若將撥碼開關的全部撥到 ON，則從 TF Card 啓動

1-4.3 介面說明


SDIO WiFi
Extend Connector
10/100M Ethernet
RTC Battery
UART 0(COM1)
UART 1(COM2)
UART 2(COM3)
RS485
WiFi ANT
S3C6410
RJ45 MAC Address
Touch Connector
LCD Connector
Key1 Key2 Key3 Key4 Key5 Key6 Key7 Key8 Key9
Ear Jack
TF Card
Jumper
USB OTG
USB HOST
DC 5V ~ 30V
Sleep/Wake
Reset

名稱	說明
DC 6V~12V/2A(J601)	外部 DC 6V~12V 電源輸入端
USB OTG(CON401)	USB OTG 介面
USB HOST 1.1(CON402)	USB HOST 介面
SW602	重置鍵
SW601	睡眠/喚醒鍵
EXT_INF(J703)	外部擴充介面
UART0(CON501)	串列埠 0
UART1(CON502)	串列埠 1
UART2(CON503)	串列埠 2
RS485(J503)	RS485 介面
TF Card(J201)	TF 卡插槽
SDIO WIFI（U1101）	SDIO WIFI 介面 (Option)
Ear PHONE Jack(J301)	耳機聽筒輸出端
10/100M Ethernet(J801)	10/100M 網路介面
LCD(J901)	60pin LCD 介面
S3C 6410(U1)	S3C6410 晶片
Battery(BT1)	3.3V 電池
Keyboard	9 個功能鍵
RTC	高精度的 RTC 時鐘模組

1-4.4 介面尺寸結構圖

詳見附錄一。

1-5 HMI700-6410S 人機介面配件

1-5.1 基本配件

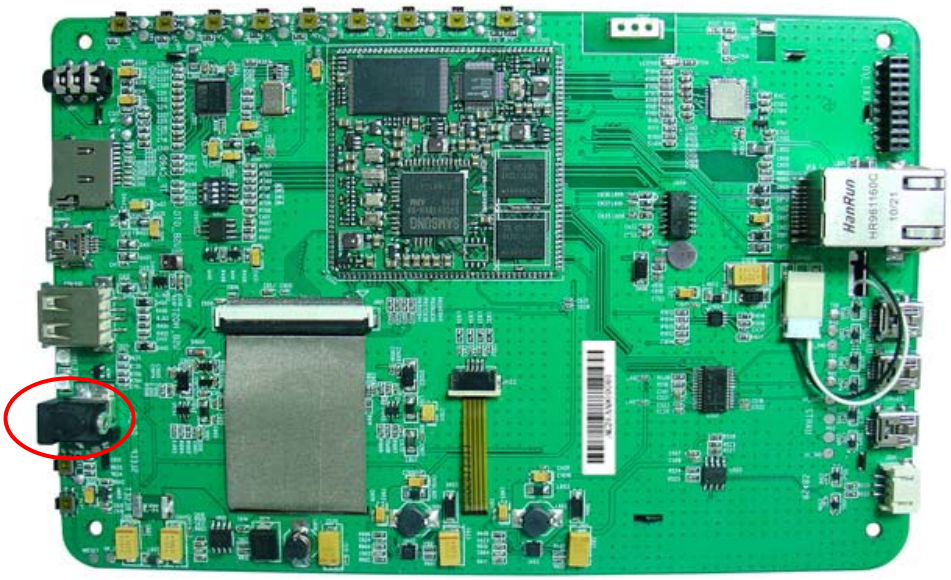
1). HMI700-6410S 人機介面 	2). HMI700-6410S 資料光碟 
3). USB 資料傳輸線 	4). DC5V/2A 電源 
5). 交叉網路線 	6). Mini USB 轉串列埠線 

第二章
硬體介紹

2-1　電源電路

2-1.1　電源輸入部分

電源是由外部變壓器提供 DC 8V-30V/2A 電壓，對整個平台供電，電源輸入埠(J601)實物圖如下：



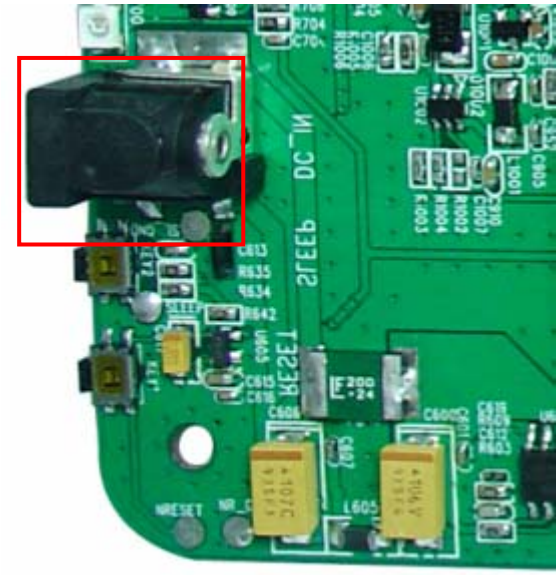


圖 2-1

電路如圖 2-2 所示：

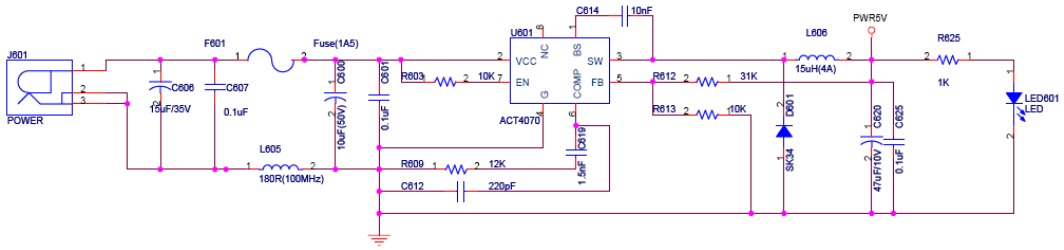


圖 2-2 外部電源輸入電路

2-1.2　底板部分電源

HMI700-6410S 底板上主要用到了 5V 和 3.3V 電源，3.3V 電源可由 DC/DC 轉換晶片來獲得，該底板上採用了 MIC2207 電源轉換晶片，具體電路如圖 2-3 所示：

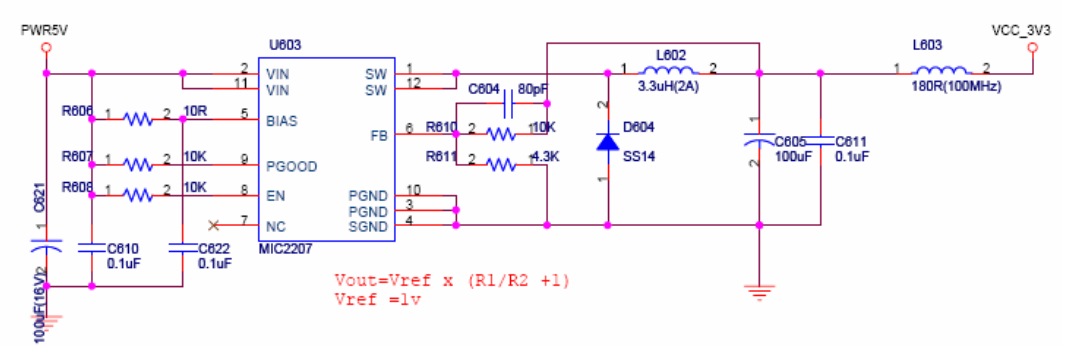


圖 2-3 底板上 3.3V 電路

該電路一共 2 組，分別提供給 VCC_ C3V3（核心板 3.3V 專用電源）以及 VCC_3V3（底板 3.3V 專用電源）。

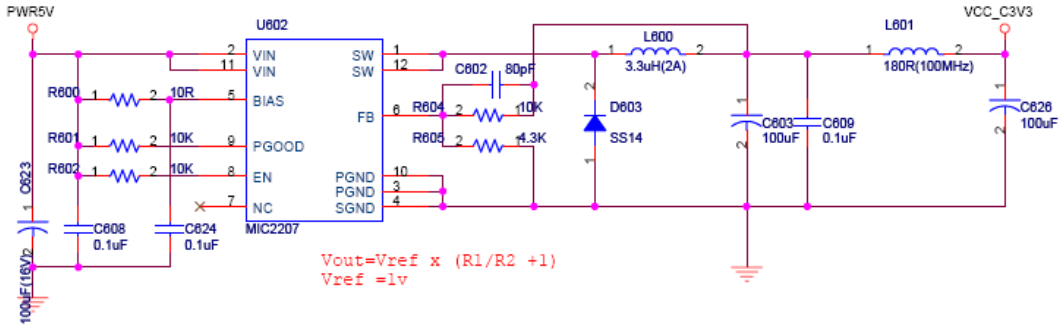


圖 2-4 供給核心板的 3.3V 電路

2-1.3 核心板部分電源

從底板上提供的 5V 和 3.3V 電源透過電源晶片轉換，可以爲核心板上的 RAM、ROM 及 CPU 等供電。CPU 上各電源分佈情況如圖 2-5 所示：

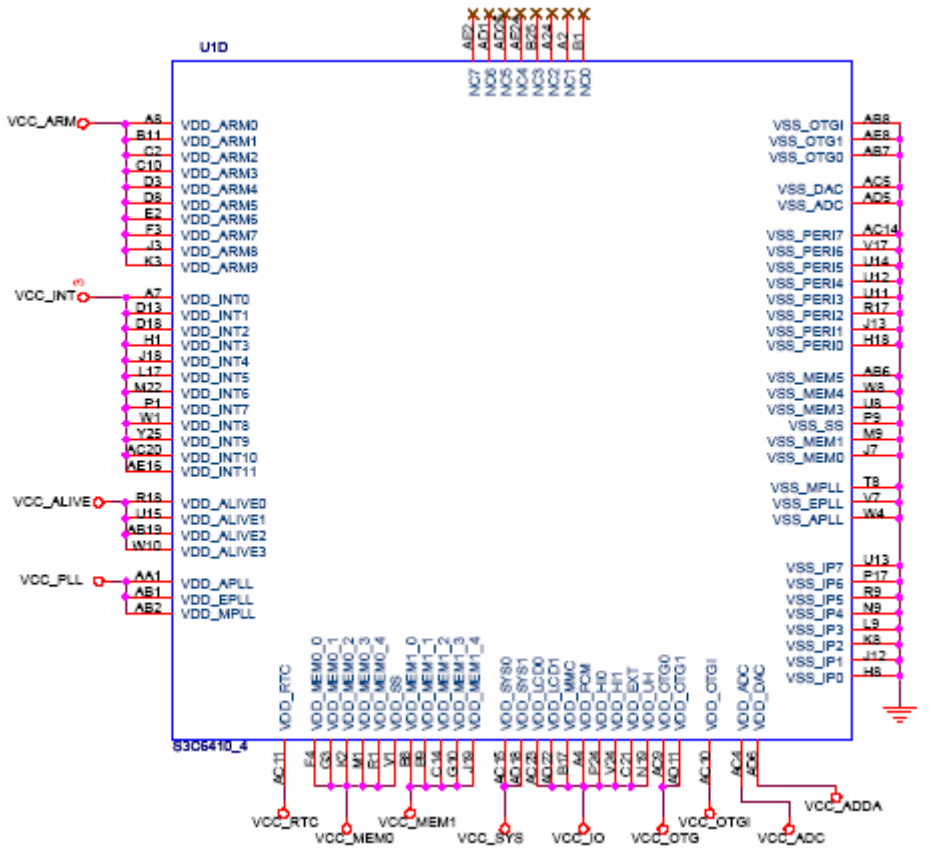


圖 2-5 CPU 上各電源分佈圖

VCC_ARM、VCC_INT、VCC_MEM1/VCC_DDR 的電源轉換電路如圖 2-6 所示：

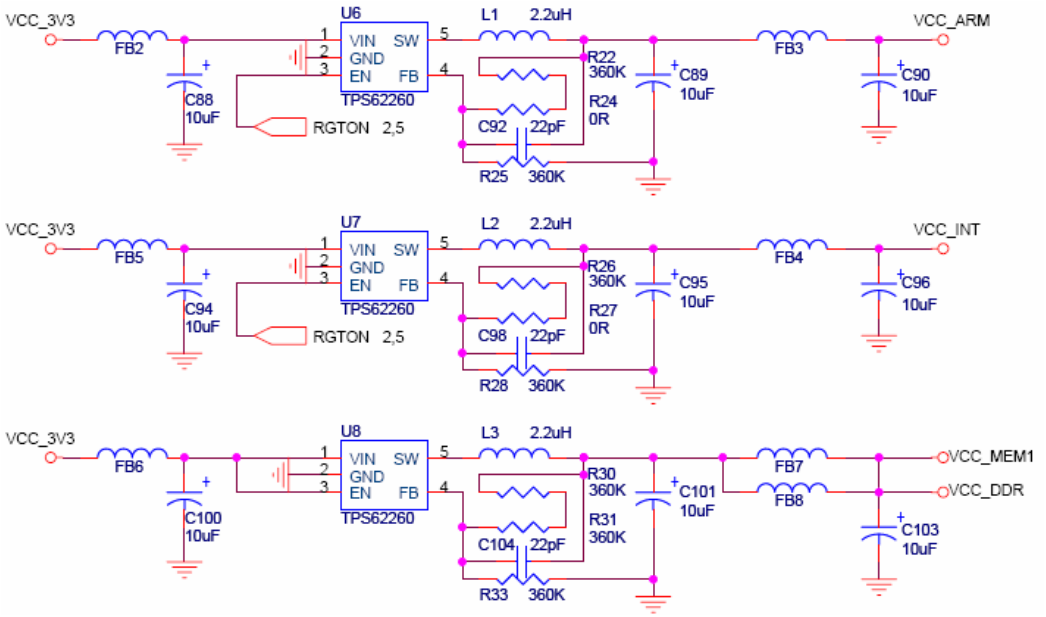
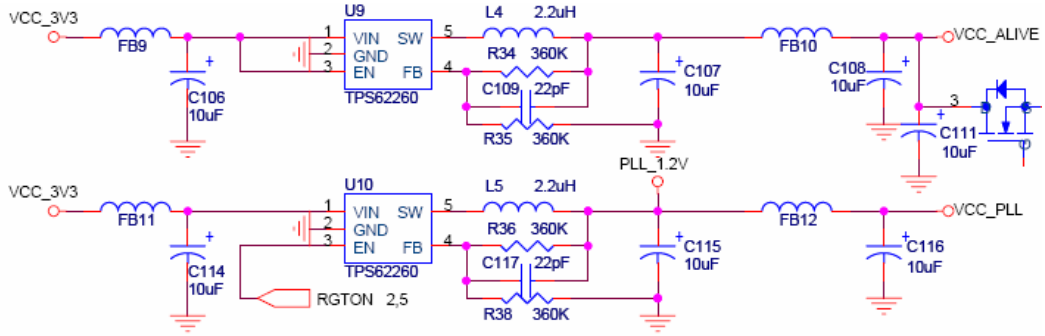


圖 2-6　VCC_ARM、VCC_INT、VCC_MEM0/VCC_MEM1/VCC_DDR 電路

VCC_ALIVE、VCC_APLL/VCC_MPLL/VCC_EPLL、VCC_OTGI 的電源轉換電路如圖 2-7 所示：



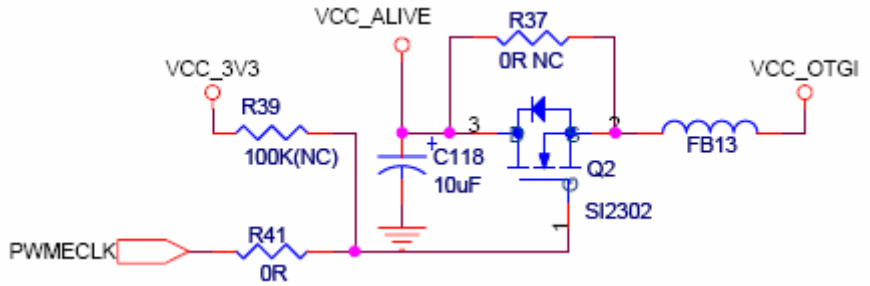


圖 2-7　VCC_ALIVE、VCC_APLL/VCC_MPLL/VCC_EPLL、VCC_OTGI 電路

VCC_OTG 電源轉換電路如圖 2-8 所示：

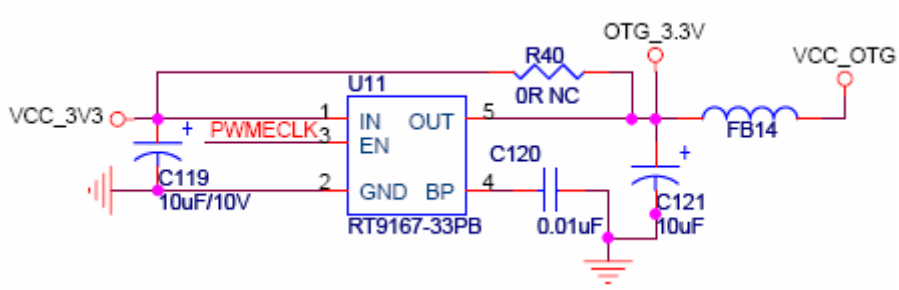
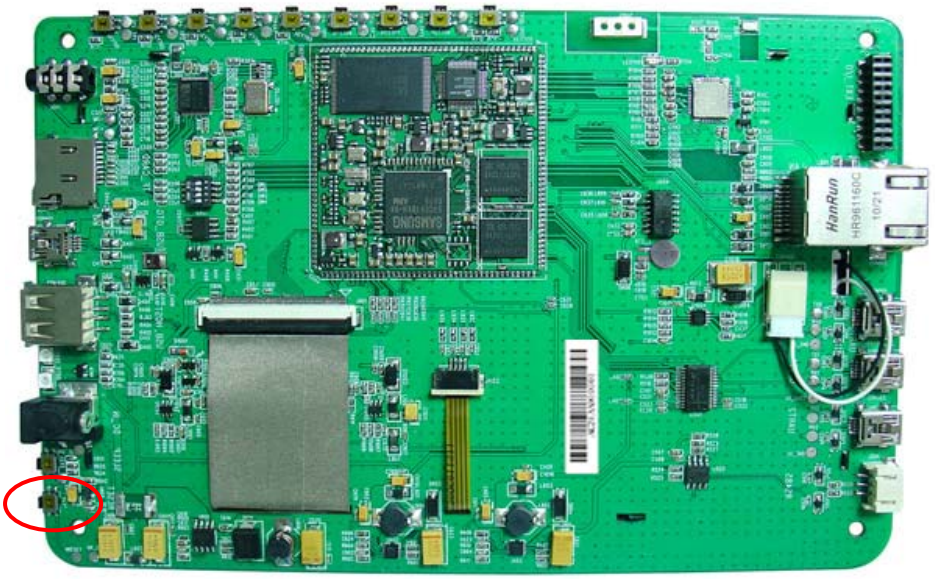


圖 2-8　VCC_RTC VCC_OTG

2-2　RESET 電路

底板上的 RESET 電路採用專用 MAX811 重置控制晶片。RESET 按鍵(KEY1)實物圖如下：



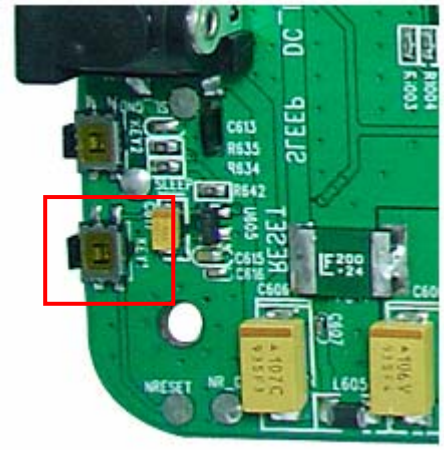


圖 2-9

具體電路如下圖：

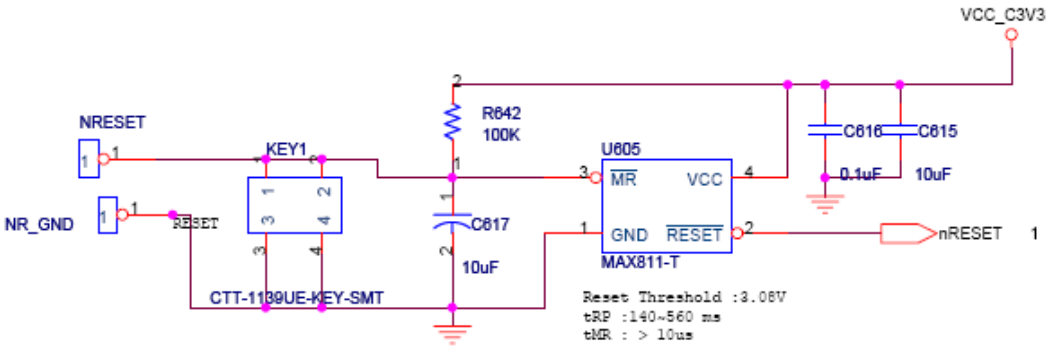
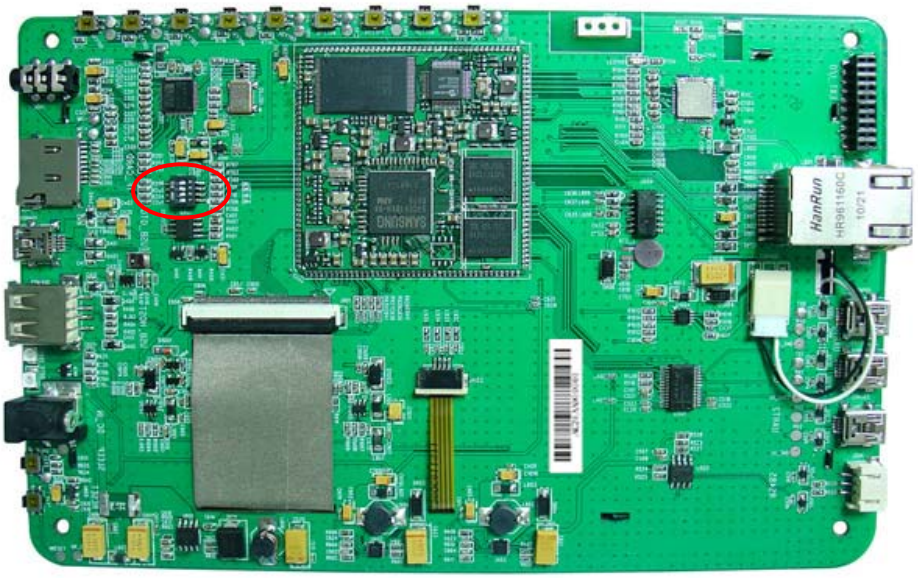


圖 2-10 底板上的重置電路

2-3 啟動方式選擇電路

本平台由底板上的 J701 來選擇系統的啓動方式。



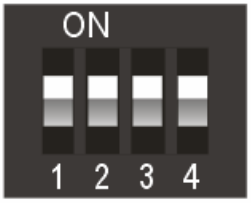
- 若 NAND FLASH 爲 128MB，則將撥碼開關的第三腳撥到 ON，其他腳撥到 OFF 上，則從 NAND FLASH 啓動，如下圖：

示意圖

- 若 NAND FLASH 爲 256MB，則將撥碼開關的第三、第四腳撥到 ON，其他腳撥到 OFF 上，則從 NAND FLASH 啓動，如下圖：

示意圖

- 若將撥碼開關的全部撥到 ON，則從 TF Card 啓動，如下圖：



示意圖

電路如圖 2-11 所示

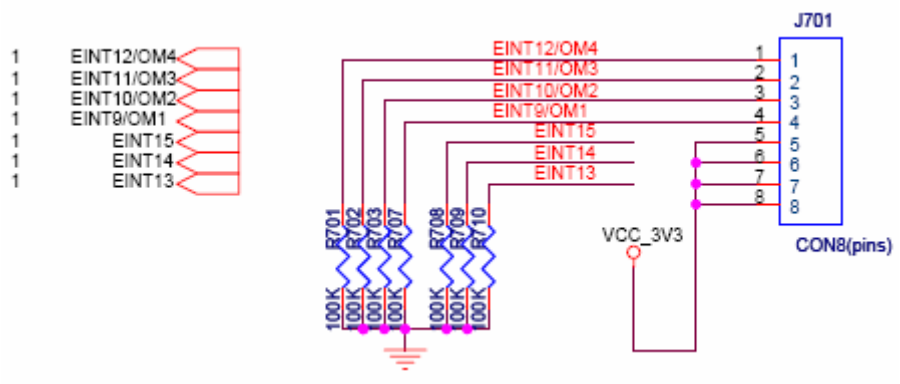


圖 2-11　啓動方式選擇電路

2-4　核心板上的 Flash 電路

本平台配置了 Nand Flash， Nand Flash 採用 K9F1G08U0M。K9F1G08U0M(U5)的實物圖如下：

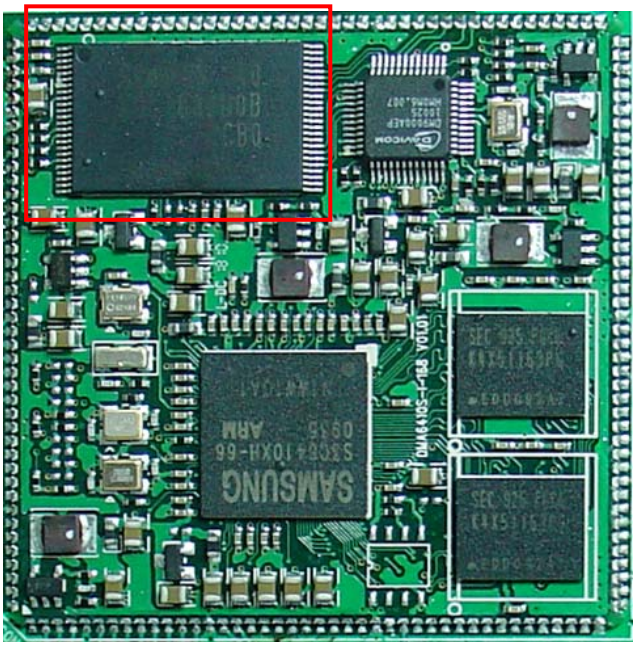


圖 2-12

具體電路如圖 2-13 所示：

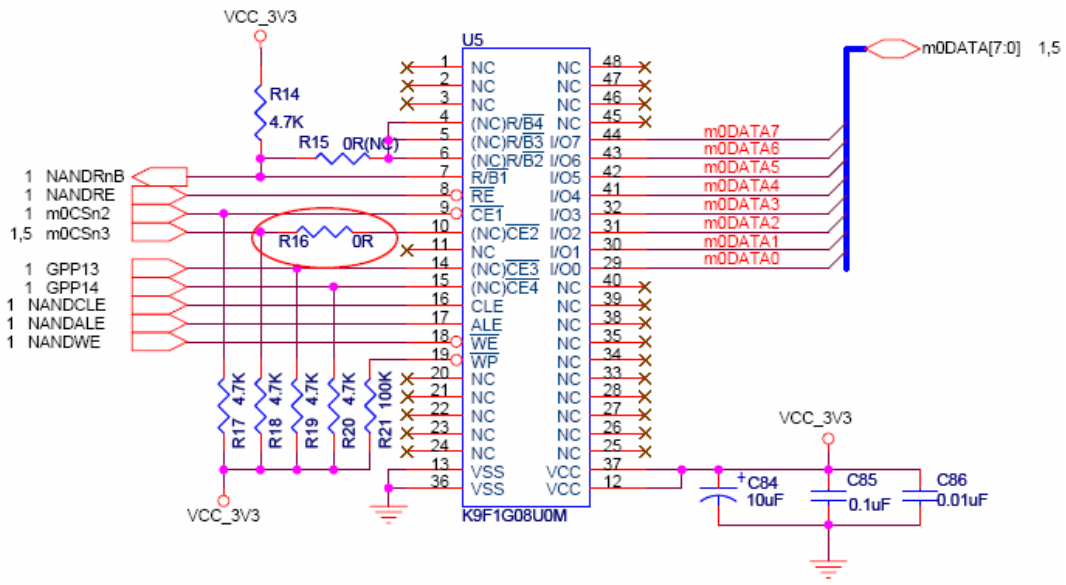


圖 2-13　NAND Flash 電路

2-5 核心板上的 SDRAM 電路

本平台採用兩片 K4X51163 組成 128MB RAM 存儲，主要用於系統執行。128MB DDR（U2，U3）實物圖如下：

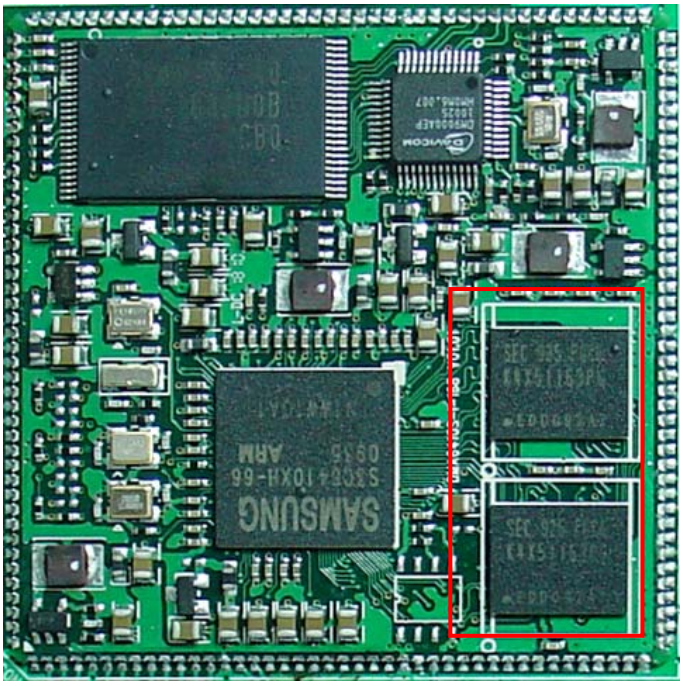
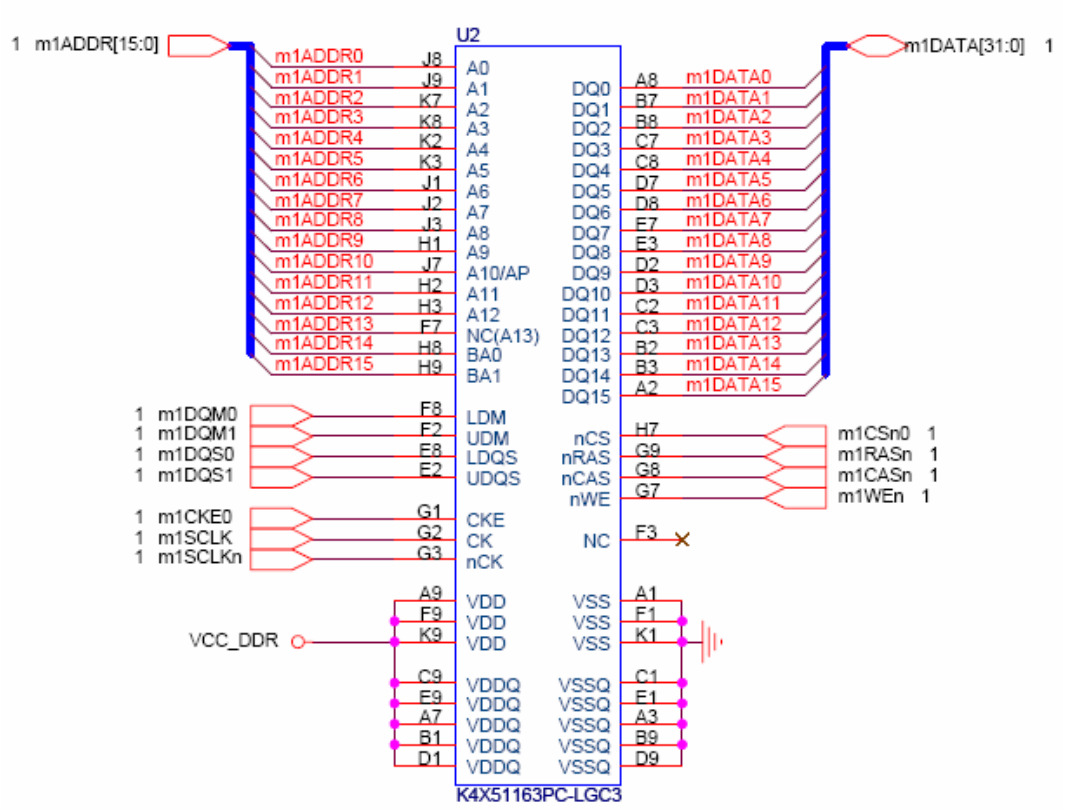


圖 2-14

具體電路如圖 2-15 所示：



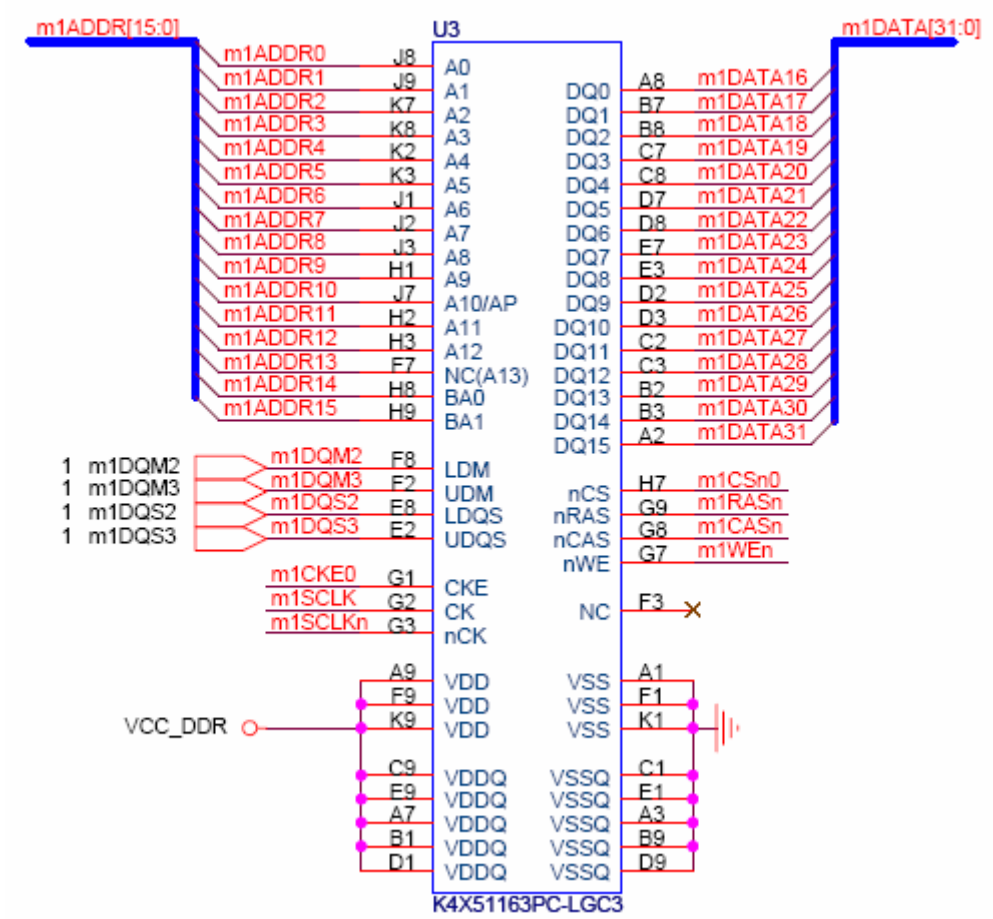


圖 2-15 128Mb RAM 電路

2-6　核心板上接腳所有信號說明

核心板上採用貼片方式貼在底板上。具體說明如下圖：

連接器一的信號說明如圖 2-16：

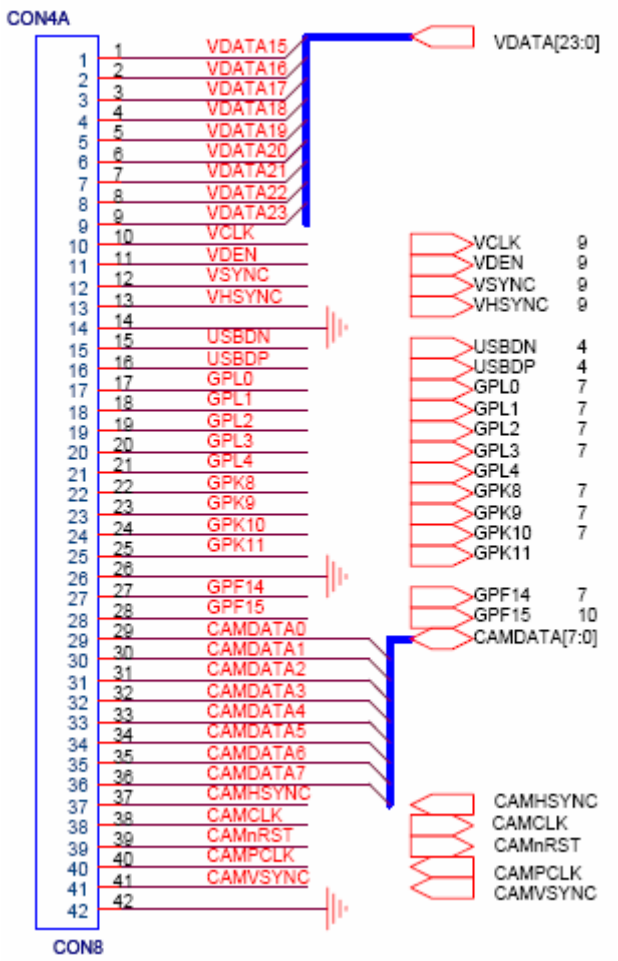


圖 2-16　連接器一的信號說明

連接器二的信號說明如圖 2-17：

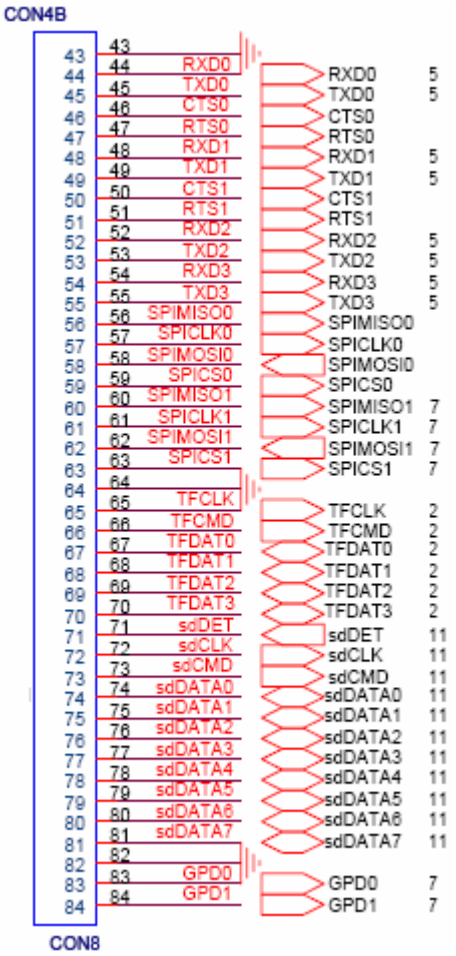


圖 2-17 連接器二的信號說明

連接器三的信號說明如圖 2-18：

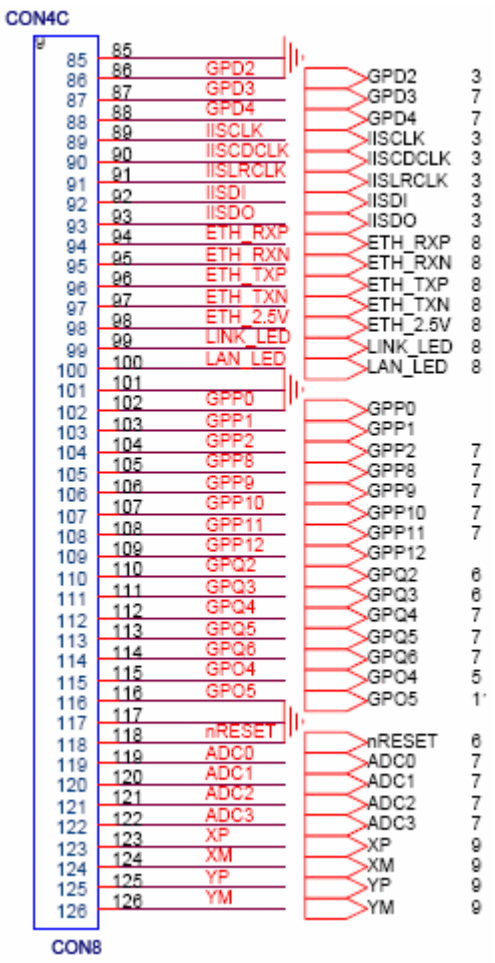


圖 2-18 連接器三的信號說明

連接器四的信號說明如圖 2-19：

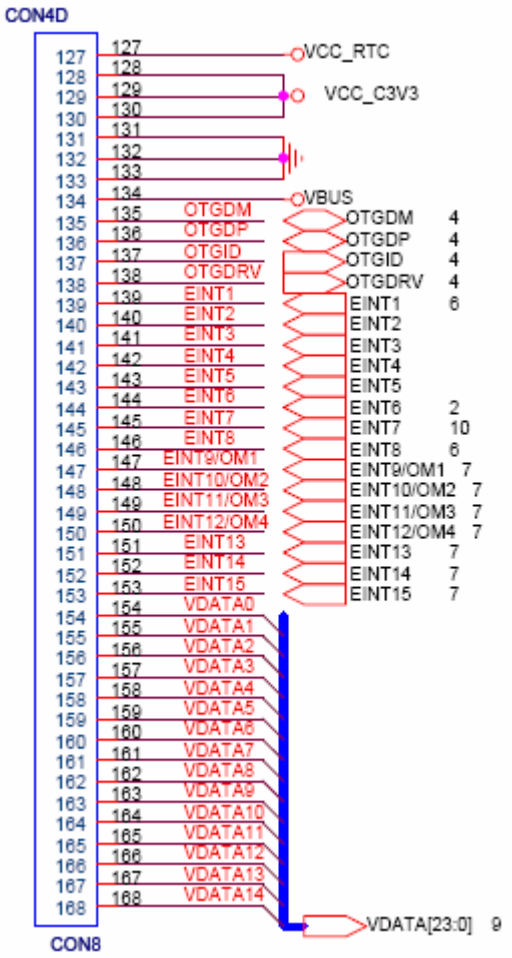


圖 2-19　連接器四的信號說明

2-7　AUDIO 電路

音效電路採用了 WM9713G 晶片，耳機（J301）輸出實物圖如下：

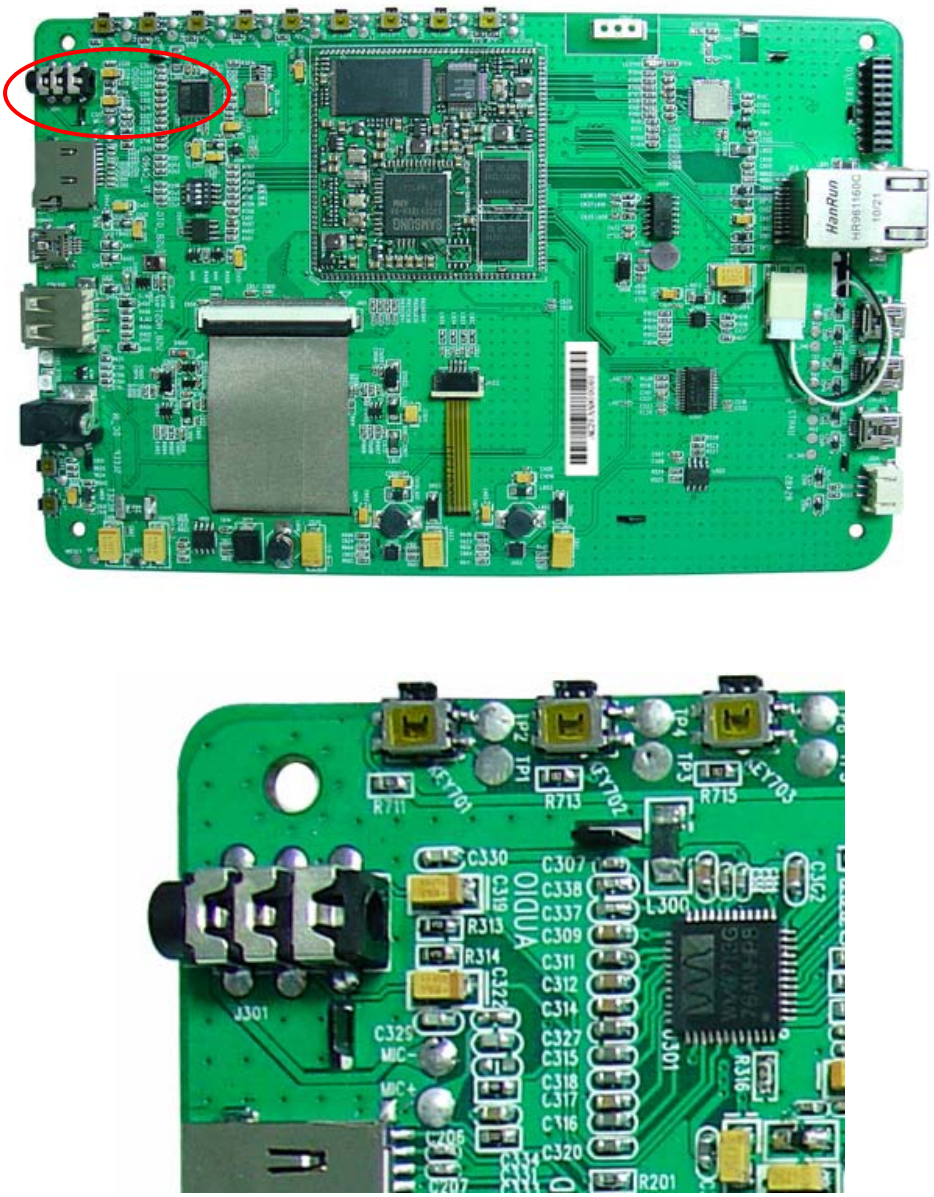
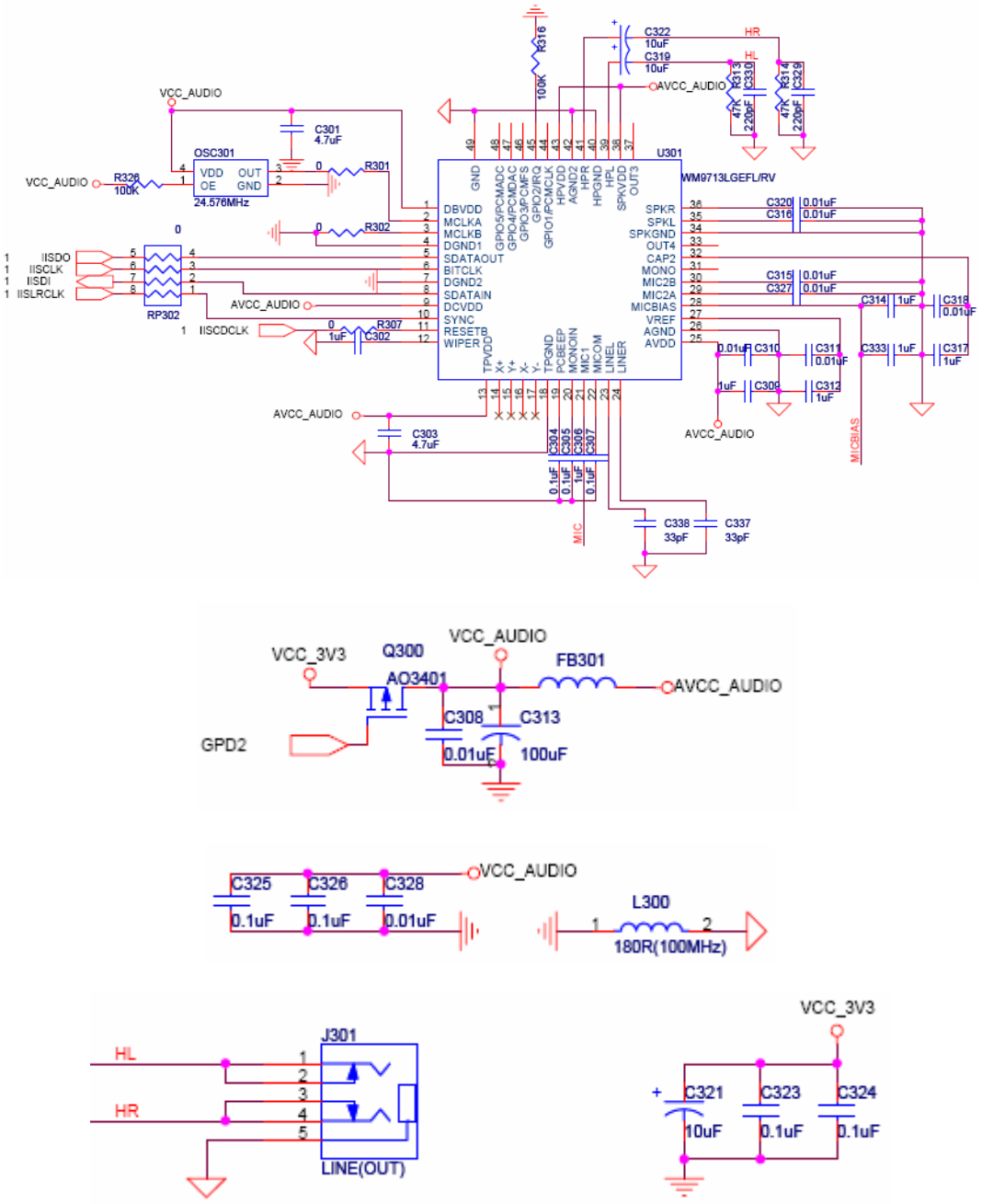


圖 2-20

數位音效編/解碼具體電路如圖 2-21：



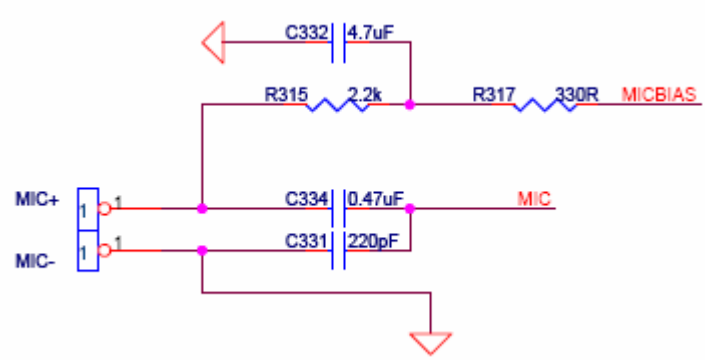
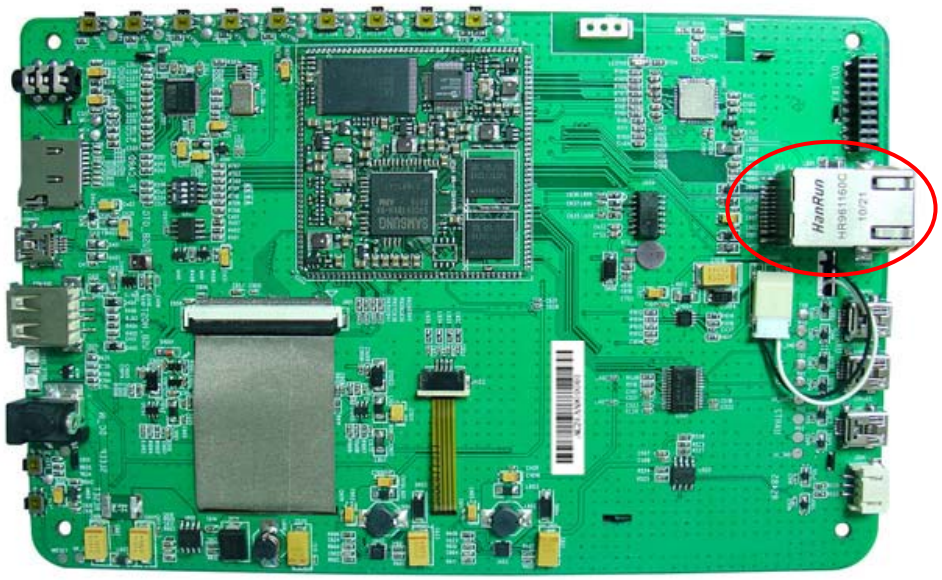


圖 2-21 數位音效編/解碼電路

2-8 網路

平台上配置了一個 10/100M 的網路介面，採用的晶片是 DM9000A。網路介面（CON802）實物圖如下：



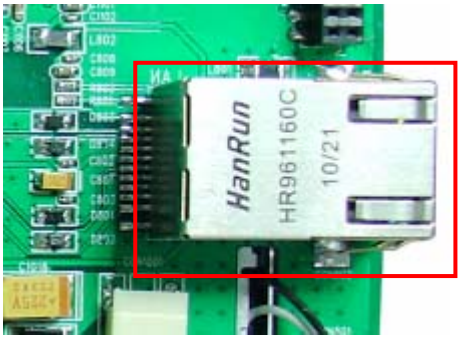
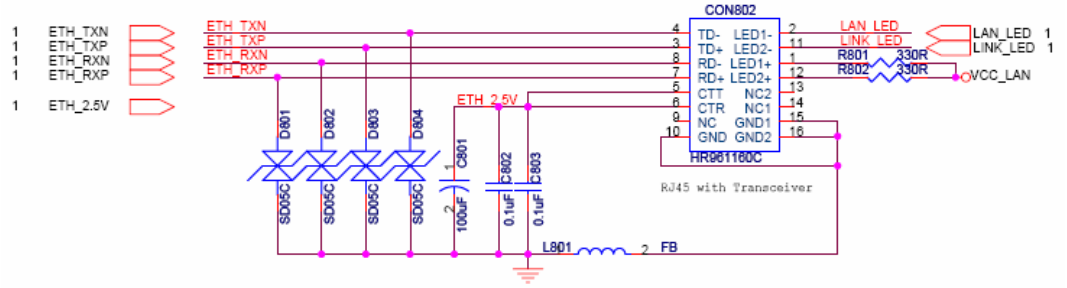


圖 2-22

具體電路如圖 2-23 所示：



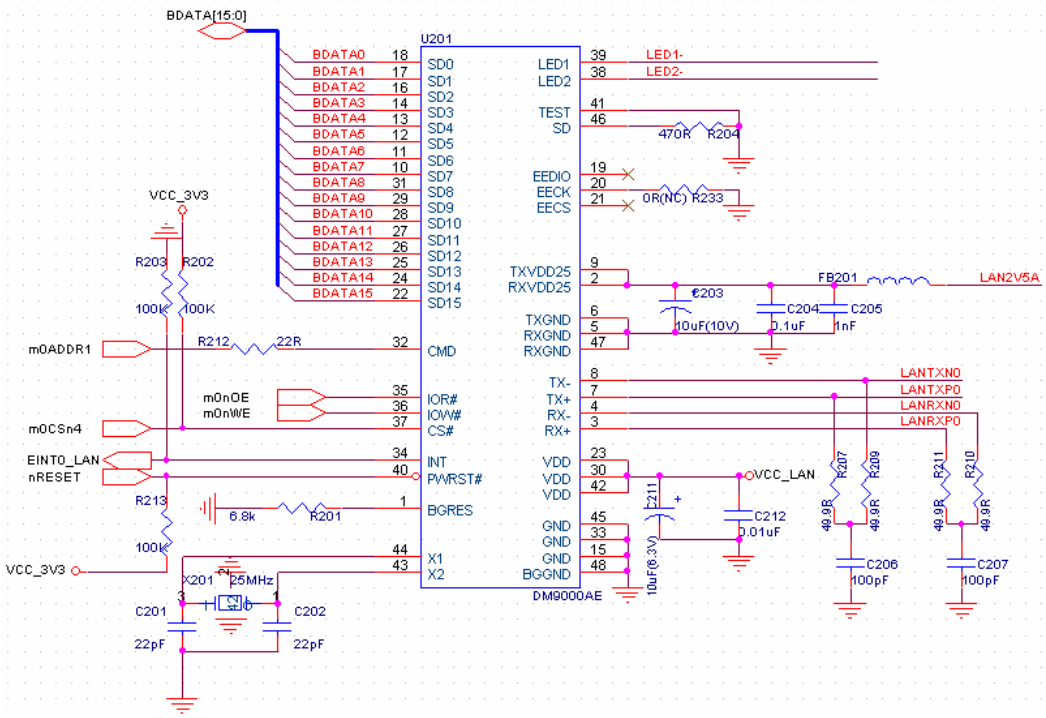


圖 2-23 網路介面電路

網路介面信號定義如表 2-1 所示：

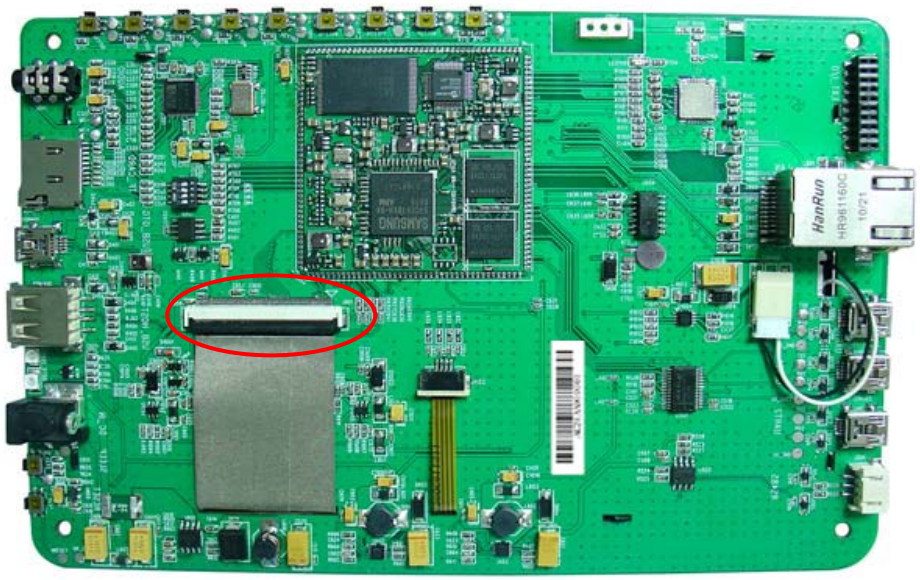
接腳	信號	方向	描述
1	LED+	P	LAN LED+
2	LED-	N	LAN LED-
3	TX+	I	Data output Positive
4	TX-	/	Data output Negative
5	CTT	P	VCC
6	CTR	P	VCC
7	RX-	/	Data input Negative
8	RX+	/	Data input Positive

9	NC	/	
10	GND	/	EMI Grounded
11	LED-		LINK LED-
12	LED+		LINK LED+
13	NC	/	
14	NC	/	
15	GND	/	EMI Grounded
16	GND	/	EMI Grounded

表 2-1 10/100M 網路介面信號定義

2-9 LCD 介面

3.3V 供電，支援尺寸 7”。LCD 介面（J901）實物圖如下：



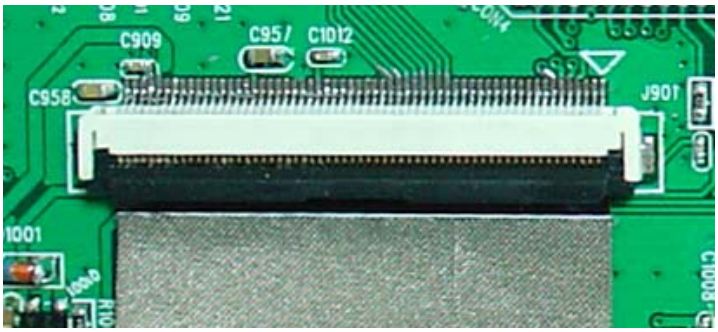


圖 2-24

具體電路如圖 2-25 所示：

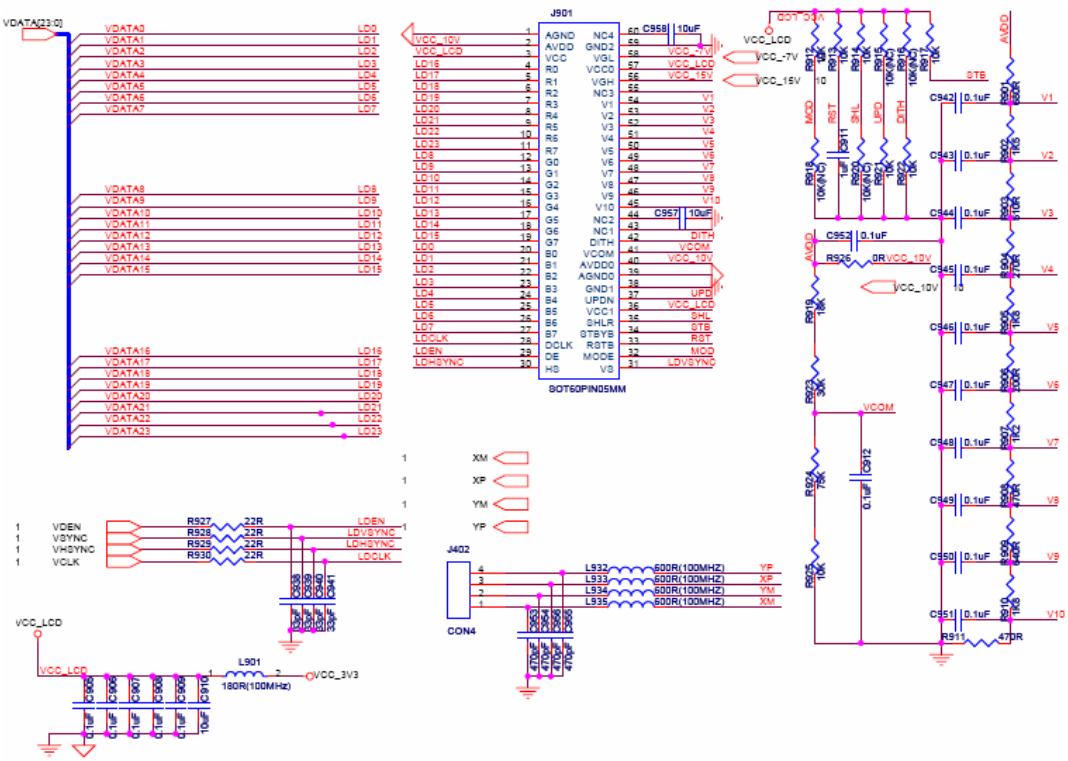


圖 2-25 LCD 介面電路

2-10 USB 介面

USB是英文Universal Serial Bus的縮寫，中文含義是「通用串列匯流排」。早在1995年，就已經有配置有USB介面的PC了，但由於缺乏軟體及硬體設備的支援，這些PC的USB介面都閒置未用。1998年後，隨著微軟在Windows 98中內配置了對USB介面的驅動支援，加上USB設備的日漸增多，USB介面才逐步走進了實用階段。

目前，USB設備已經被大量應用在各個領域，主要原因爲USB介面具有以下優點：

1、 支援熱插拔。熱插拔的功能可以讓使用者在使用外接設備時，不需要重複「關機將並列埠或串列埠電纜接上後再開機」這樣的動作，而是直接在PC開機時或運作中，就可以將USB電纜插上使用。

2、 攜帶方便。USB設備大多以「小、輕、薄」見長，對使用者來說，同樣20G的硬碟，USB硬碟比IDE硬碟要輕一半的重量，在想要隨身攜帶大量資料時，當然USB硬碟會是首要之選了。

3、 標準統一。大家常見的是IDE介面的硬碟，序列埠的滑鼠鍵盤，並列埠的印表機掃描器，可是有了USB之後，這些應用外接設備統統可以用同樣的標準與PC連接，這時就有了USB硬碟、USB滑鼠、USB印表機等等。

4、 可以連接多個設備。USB在PC上往往具有多個連接埠，可以同時連接幾個設備，如果接上一個有4個埠的USB HUB時，就可以再連上4個USB設備，以此類推，盡可能連下去，將你家的設備都同時連在一台PC上而不會有任何問題（註：最高可連接至127個設備）。

目前USB設備雖已被廣泛應用，但比較普遍的却是USB 1.1介面，它的傳輸速度僅爲12Mbps。當你用USB 1.1的掃描器掃描一張大小爲40Mb的圖片，需要4分鐘之久。這樣的速度，讓使用者覺得非常不方便，如果有好幾張圖片要掃的話，就得要有很好的耐心來等待了。

不過由COMPAQ、Hewlett Packard、Intel、Lucent、Microsoft、NEC和PHILIPS這7家廠商聯合制定了USB 2.0介面標準後，就大大改善了傳輸過慢的問題了。USB 2.0將設備之間的資料傳輸速度增加到了480Mbps，比USB 1.1標準快40倍左右，速度

的提高對於使用者的最大好處就是意味著使用者可以使用到更高效的外部設備，而且具有多種速度的周邊設備都可以被連接到USB 2.0的線路上，而且無需擔心資料傳輸時發生瓶頸效應。

如果你用USB 2.0的掃描器，就完全不同了，掃一張40Mb的圖片只需半分鐘左右的時間，一眨眼就過去了，效率大大提高。而且，USB 2.0可以使用原來USB定義中同樣規格的電纜，接頭的規格也完全相同，在高速的前提下一樣保持了USB 1.1的優秀特色，並且，USB 2.0的設備不會和USB 1.1設備在共同使用的時候發生任何衝突。

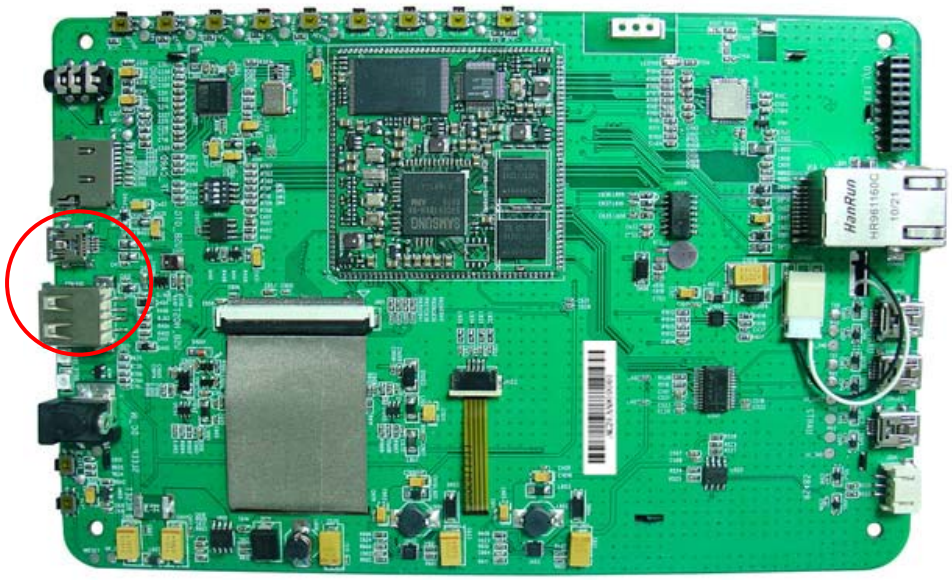
自 1996 年 USB1.0 規範以後，USB-IF(Universal Serial Bus Implementers Forums)又陸續公布了 USB1.0 、USB2.0 和 USB OTG 等幾個規範，其中 USB2.0 的傳輸帶寬達到 480Mbps，而 USB OTG 更使 USB 裝置擺脫了原來主從架構的限制，實現了端對端的傳輸模式，隨著 USB 規範的不斷完善，USB 的應用領域也得到了拓展。USB 誕生的初期是簡化電腦和其外部設備的連接，最早是用在鍵盤和滑鼠上，而現在，USB 的應用已經從 PC 外接設備跨越到了消費電子產品和通信電子產品領域，最顯著的應用是在數位相機等消費電子產品上。目前，USB 低功耗、便於連接和高速傳輸的特性已經使它成爲一個可方便應用的架構，它與其他技術結合便誕生出新的應用。如在移動存儲領域，USB 與快閃記憶體結合，創造性產生了目前廣爲流行的移動存儲設備 USB 隨身碟。

USB OTG 是 USB On-The-Go 的縮寫，是近年發展起來的技術，2001 年 12 月 18 日由 USB Implementers Forum 公佈，主要應用於各種不同的設備或移動設備間的聯接，進行資料交換。特別是 PDA、移動電話、消費類設備。改變如數位照相機、攝影機、印表機等設備間多種不同制式連接器，多達 7 種制式的存儲卡間資料交換的不便。USB 技術的發展，使得 PC 和周邊設備能够透過簡單方式、適度的製造成本將各種設備連接在一起，上述我們提到應用，都可以透過 USB 匯流排，作爲 PC 的周邊，在 PC 的控制下進行資料交換。但這種方便的交換方式，一旦離開了 PC，各設備間無法利用 USB 介面進行操作，因爲沒有一個設備能够充當 PC 一樣的 Host。On-The-Go，即 OTG 技術就是實現在沒有 Host 的情況下，實現設備間的資料傳送。例如數位相機直接連接到印表機上，透過 OTG 技術，連接兩台設備間的 USB 介面，

將拍出的相片立即列印出來；也可以將數位照相機中的資料，透過 OTG 發送到 USB 介面的移動硬碟上，野外作業就沒有必要攜帶價格昂貴的存儲卡，或者背一個簡易電腦。

“In-stat/MDR 於今年 2 月的統計資料也顯示，未來幾年 USB2.0 OTG 介面的周邊設備將從 2002 年的 11 萬台增長至 2007 年的 1.68 億台，增長 1527 倍。可以看出，隨著周邊設備的多樣化與高速傳輸的需求，USB 2.0 OTG 的後續發展態勢十分樂觀。” 從業界應用來看，目前高通公司(Qualcomm Inc.)已經宣佈將在其最新的 3G 手持電話基帶套片中採用該 USB OTG。索尼電子(Sony Electronics)也宣佈會選用飛利浦的 USB OTG 晶片爲其最新的可攜式設備提供 USB OTG 連接性。索尼 CLIE 是業內第一個具備 USB OTG 功能的可攜式產品，可以與其他 USB 設備實現點對點通訊。可以預見，USB OTG 會成爲未來電子產品的基本配置功能。

本平台配置了一個 USB HOST（USB 1.1）和一個 USB HS OTG（USB 2.0）介面，其中 USB HOST 支援全速 12M bps 及低速 1.5M bps，USB HS OTG 最高支援 480M bps 傳輸。USB HOST（CON402）實物圖如下：



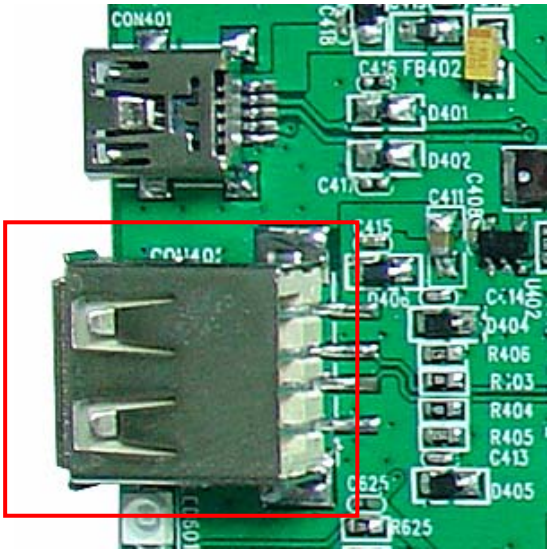


圖 2-26

USB HS OTG（CON401）實物圖如下：

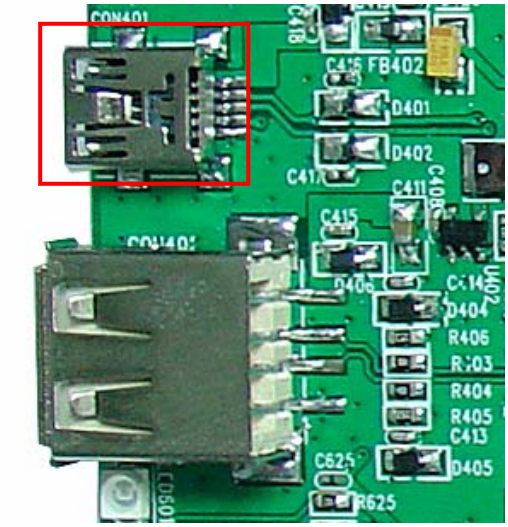
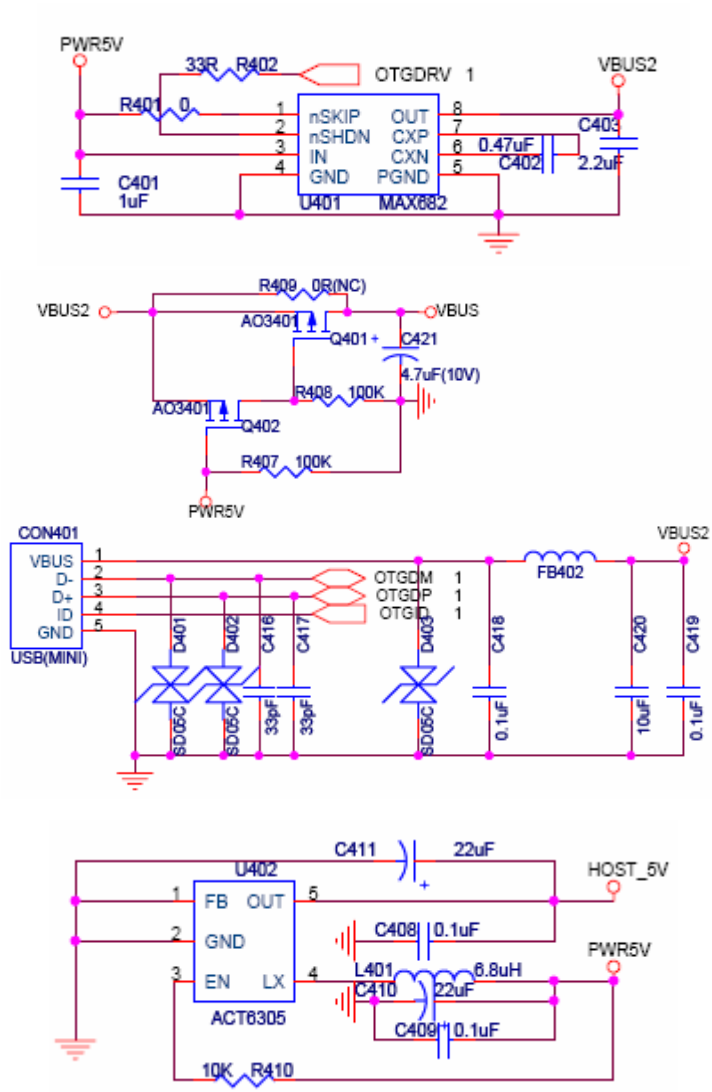


圖 2-27

具體電路如圖 2-28 所示：



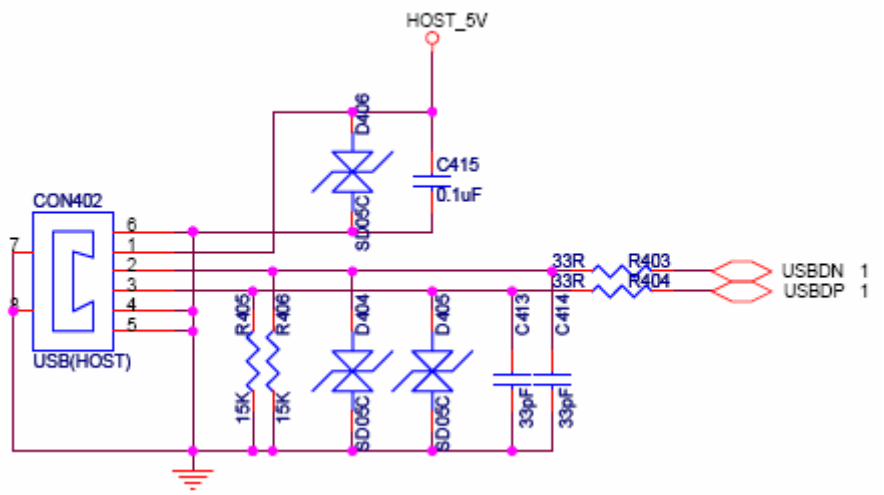


圖 2-28 USB 部分電路

USB DEVICE 信號定義如表 2-2 所示：

接腳	信號	方向	描述
1	VBUS	O	+5V Power Supply
2	USBH_N	IO	Host Data Negative
3	USBH_P	IO	Host Data Positive
4	GND	/	System Ground
5	PGND	/	Port Ground
6	PGND	/	Port Ground

表 2-2 USB DEVICE 信號定義

USB HOST 信號定義如表 2-3 所示：

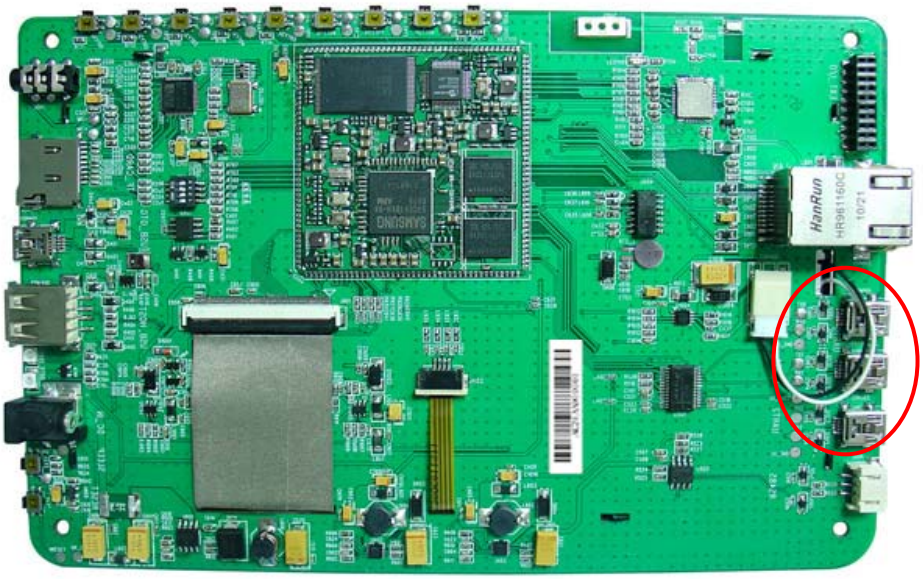
接腳	信號	方向	描述
1	NC	/	No Connection
2	USBC_N	IO	Device Data Negative
3	USBC_P	IO	Device Data Positive
4	GND	/	System Ground

5	PGND	/	Port Ground
6	PGND	/	Port Ground
7	PGND	/	Port Ground
8	PGND	/	Port Ground

表 2-3 USB HOST 信號定義

2-11 串列埠介面

串列埠一共三個 5 線制，5 線式串列埠（COM505）主要作除錯用，實物圖如下：



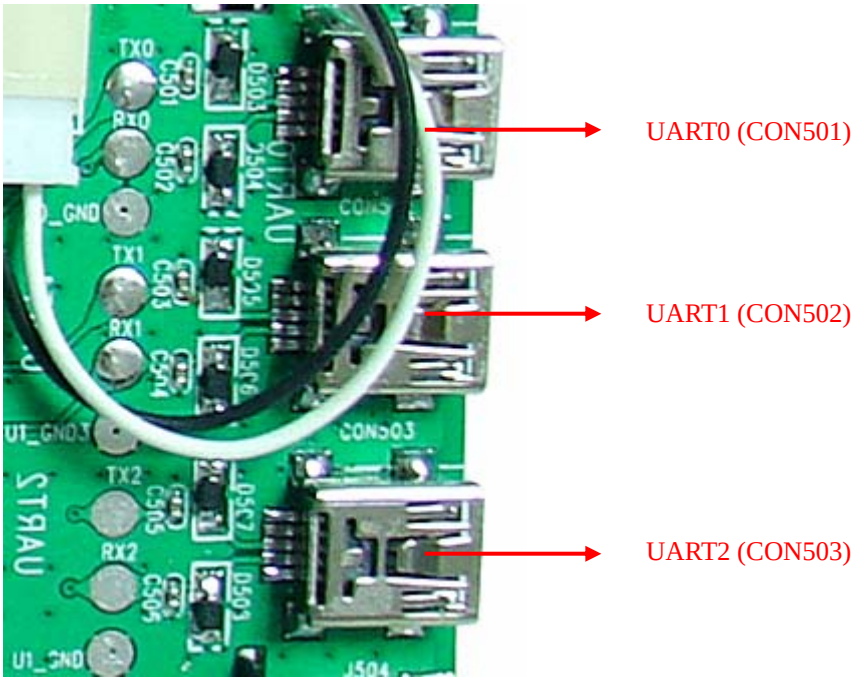
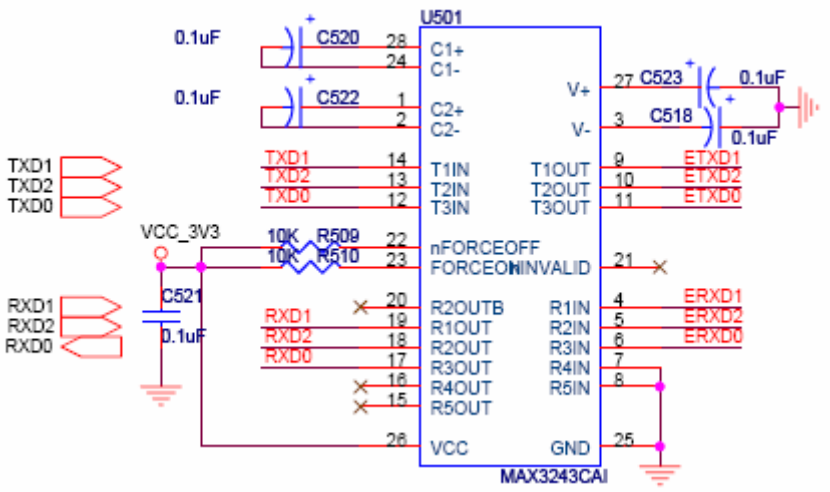


圖 2-29

具體電路如圖 2-30 所示：



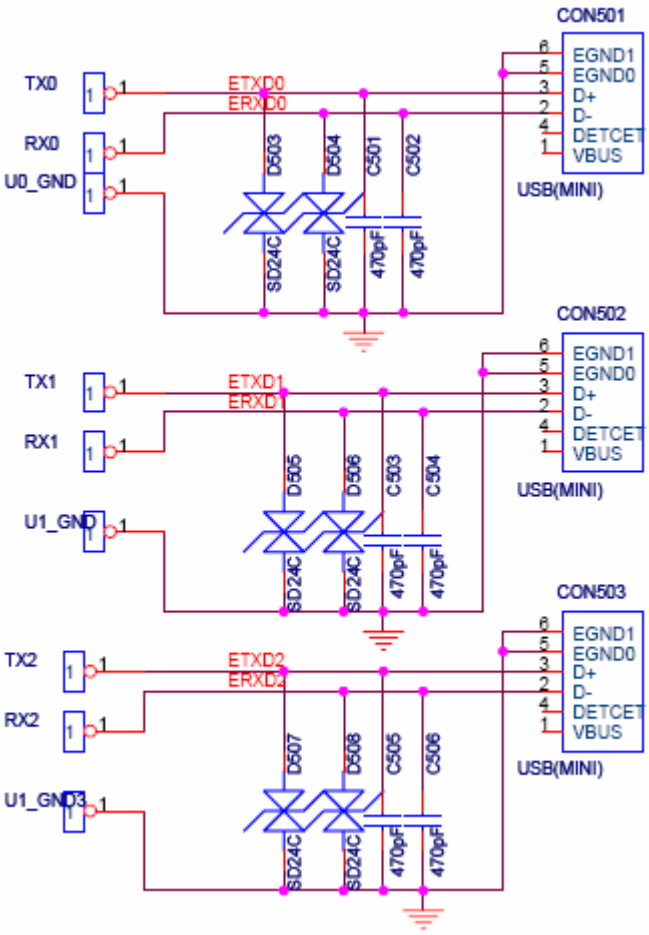


圖 2-30 串列埠電路

5 線制串列埠信號定義如表 2-5 所示：

接腳	信號	方向	描述	電壓
2	RXD	I	Receive Data	±6V
3	TXD	O	Transmit Data	±6V
5/6	GND	/	System Ground	±6V

表 2-5 5 線制串列埠信號定義

2-12　RS485 介面

本平台提供 RS485 介面 J504，其實物圖如下：

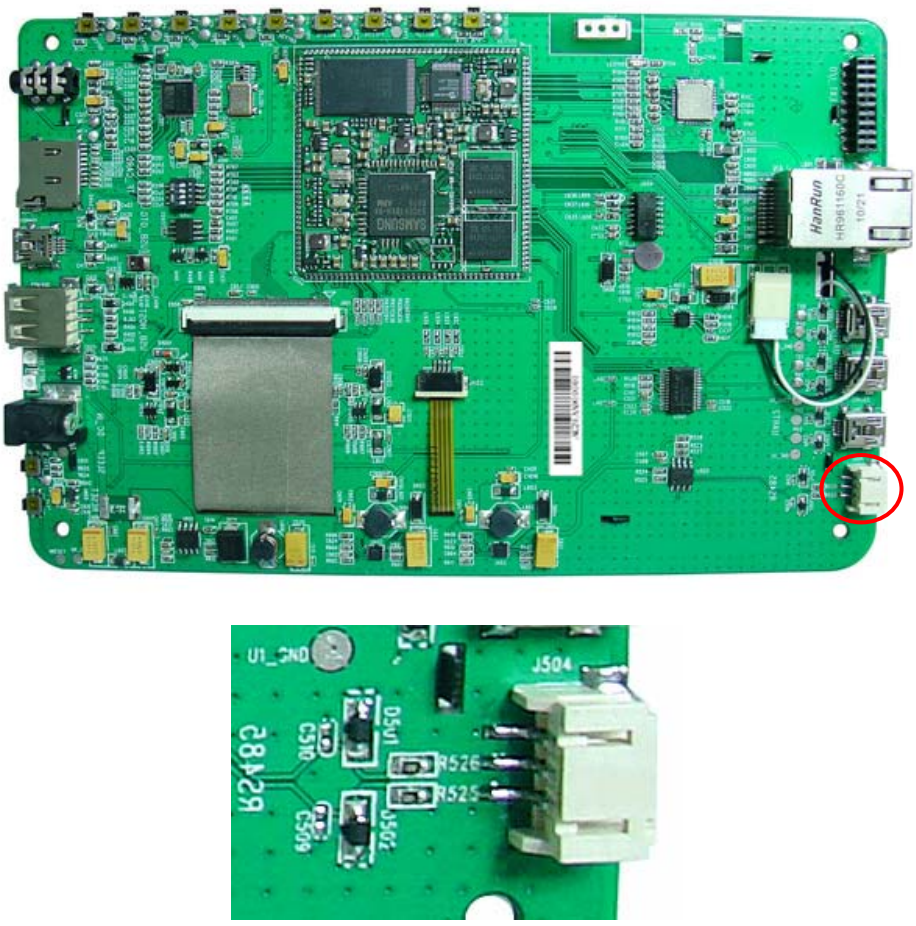


圖 2-31

具體電路如圖 2-32 所示：

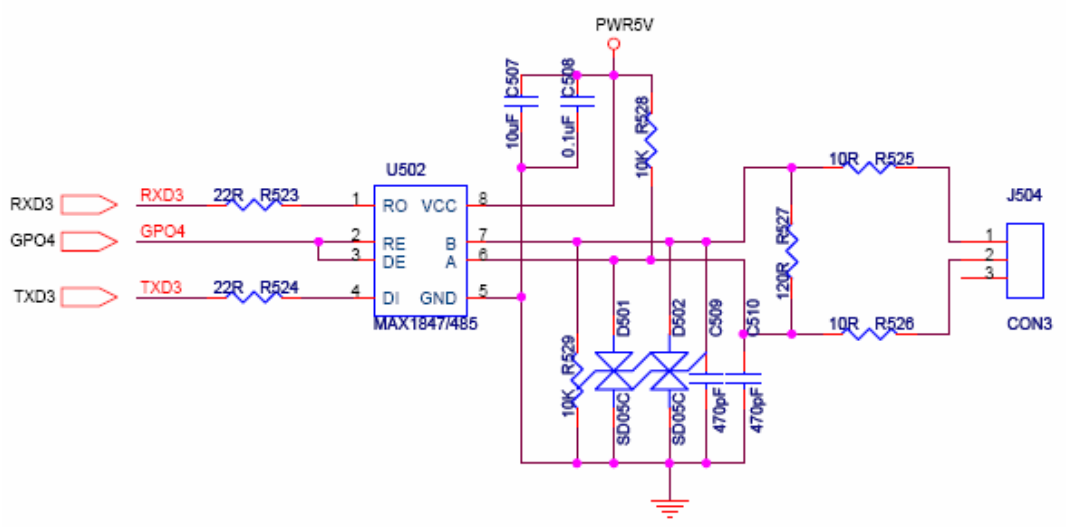


圖 2-32 RS485 介面電路

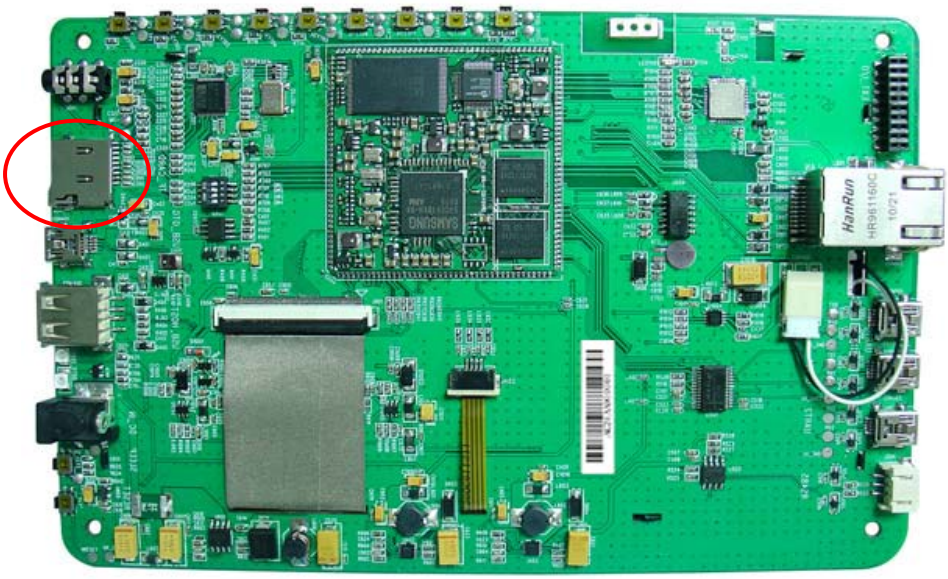
2-13 TF 卡介面

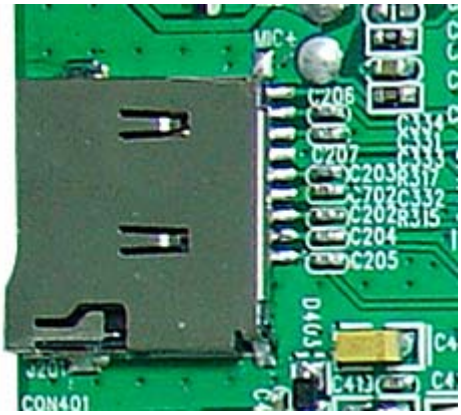
TF 卡又稱 T-Flash 卡，全名：Trans Flash，又名：Micro SD，由摩托羅拉與 SANDISK 共同研發，在 2004 年推出。是一種超小型卡（11*15*1MM），約爲 SD 卡的 1/4，可以算目前最小的儲存卡了。TF 卡可經 SD 卡轉換器後，當 SD 卡使用。利用電源供應器可以在使用 SD 作爲存儲介質的設備上使用。Trans Flash 主要是爲照相手持電話拍攝大幅圖像以及能夠下載較大的視訊片段而開發研製的。Trans Flash 卡可以用來儲存個人資料，例如數位照片、MP3、遊戲及用於手持電話的應用和個人資料等，還內設置版權保護管理系統，讓下載的音樂、影像及遊戲受保護；未來推出的新型 Trans Flash 還備有加密功能，保護個人資料、財政紀錄及健康醫療檔。體積小巧的 Trans Flash 讓製造商無須顧慮電話體積即可採用此設計，而另一項彈性運用是可以讓供應商在交貨前隨時按客戶不同需求做替換，這個優點是嵌入式快閃記憶體所沒有的。

Micro SD 卡是一種極細小的快閃記憶體卡，其格式源自 SanDisk 創造，原本這種記憶卡稱爲 T-Flash，及後改稱爲 Trans Flash；而重新命名爲 Micro SD 的原因是因爲被 SD 協會 (SDA) 採納。另一些被 SDA 採納的記憶卡包括 Mini SD 和 SD 卡。

其主要應用於行動電話，但因它的體積微小和儲存容量的不斷提升，現在已經使用於 GPS 設備、可攜式音樂播放器和一些快閃記憶體中。

它的體積爲 15mm x 11mm x1mm ，差不多相等於手指的大小，是現時最細小的記憶卡。它也能透過 SD 轉接卡來轉接於 SD 卡插槽中使用。 現時 Micro SD 卡提供 128MB、256MB、512MB、1GB、2GB、4GB、8GB、16GB 和 32GB 的容量。TF 卡插座（J201）實物圖如下：





具體電路如圖 2-33 所示：

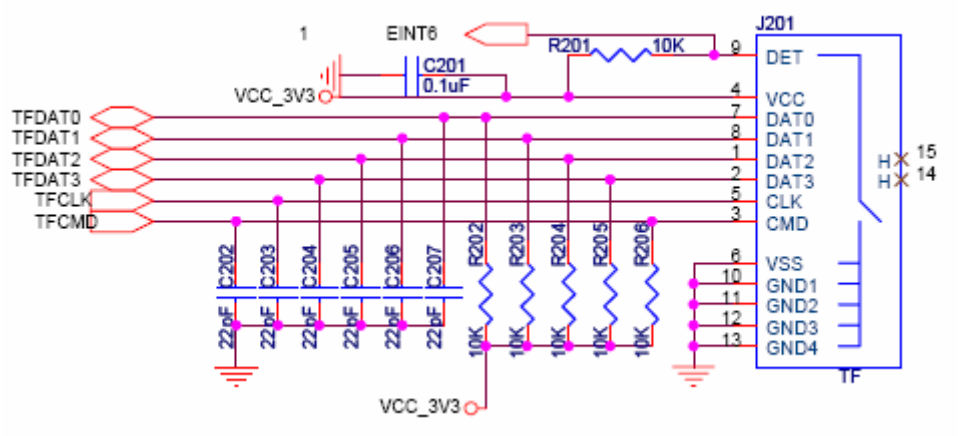
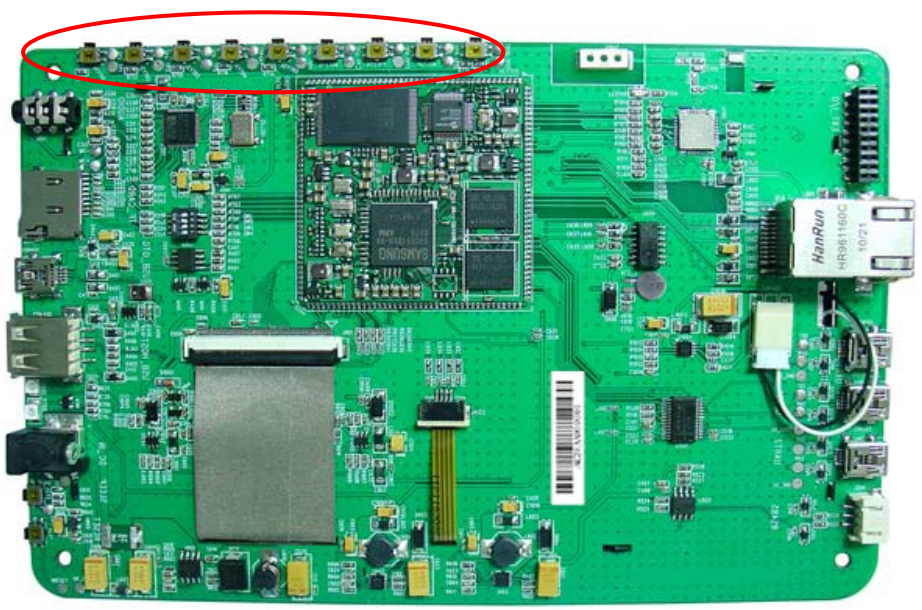


圖 2-33 TF 卡電路

2-14 矩陣鍵盤

本開發平台提供 9 個可配置型鍵盤，各按鍵功能用戶可自行設置，其實物圖如下：



具體電路如圖 2-34 所示：

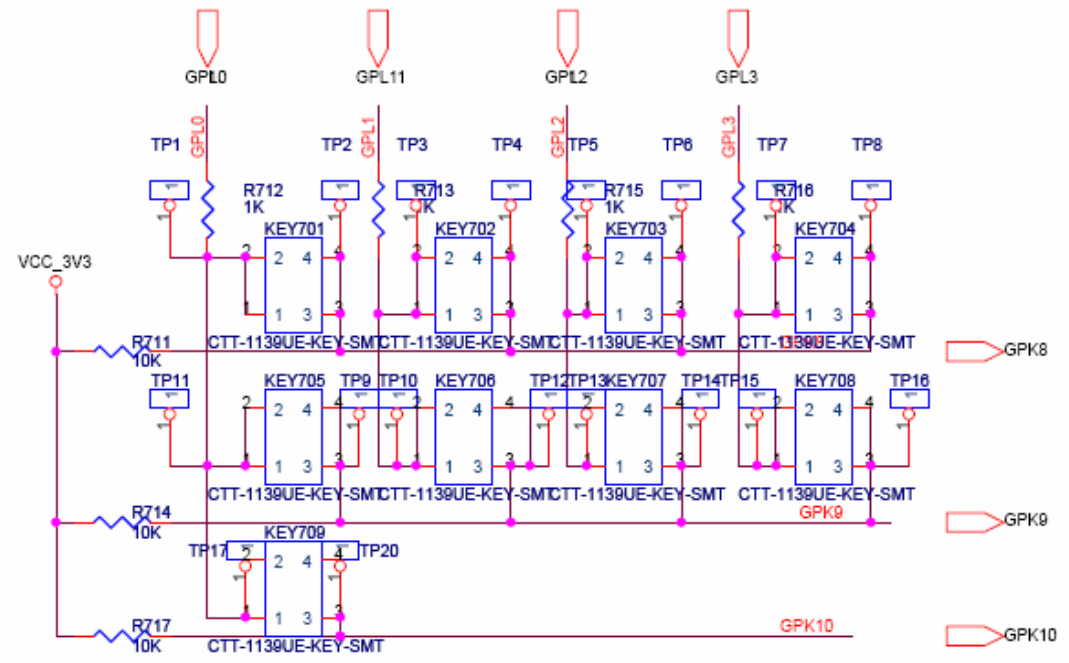


圖 2-34 矩陣鍵盤電路

2-15 多功能擴充介面

本開發平台提供擴充介面 J703，J703 包括 11 個 GPIO，1 個高速 SPI 埠，4 個 ADC 埠、一個 PWM 埠；其實物圖如下：

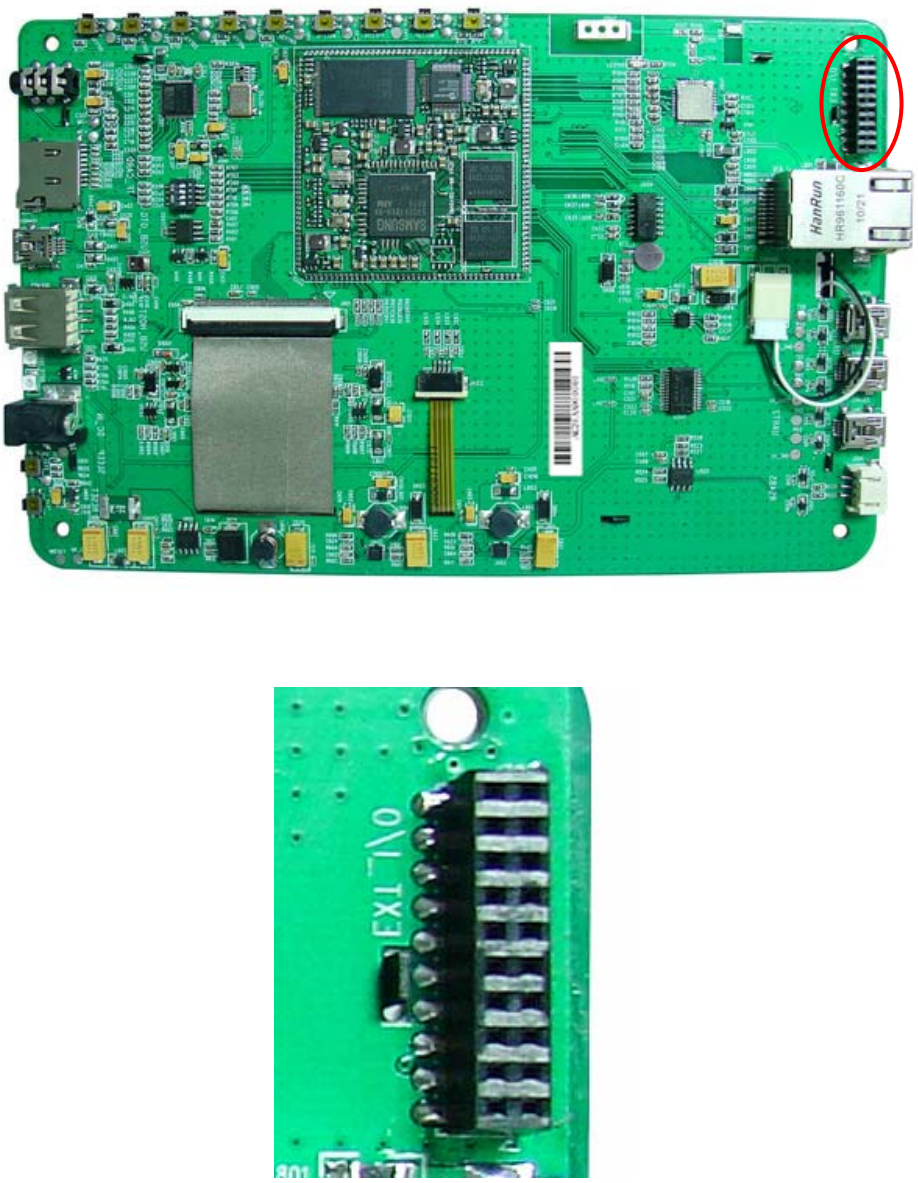


圖 2-35

具體電路如圖 2-36 所示：

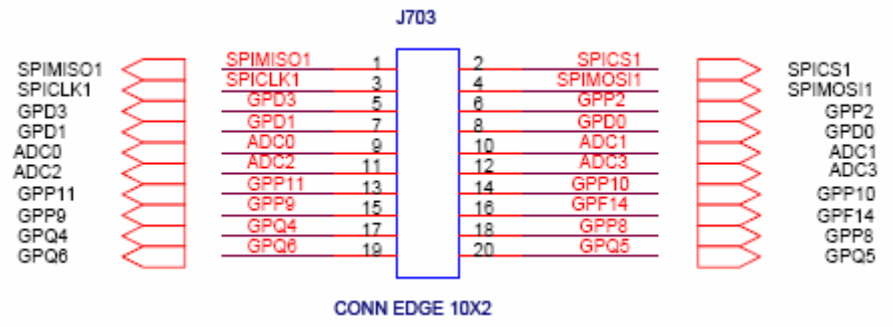
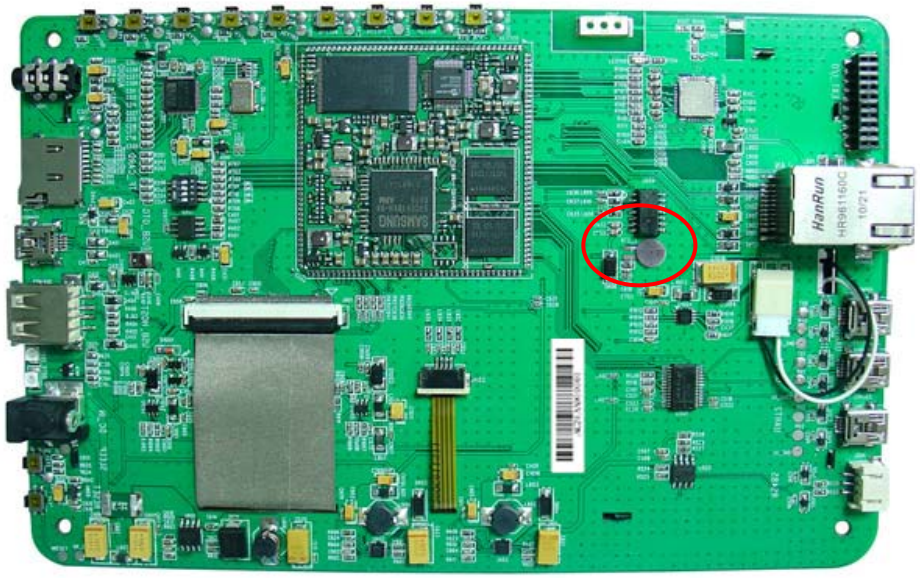
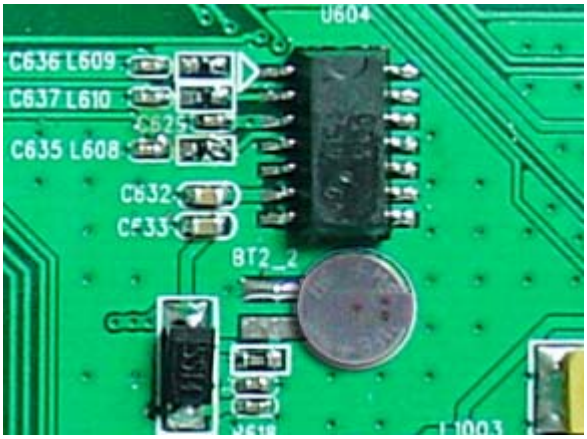


圖 2-36　多功能擴充介面電路

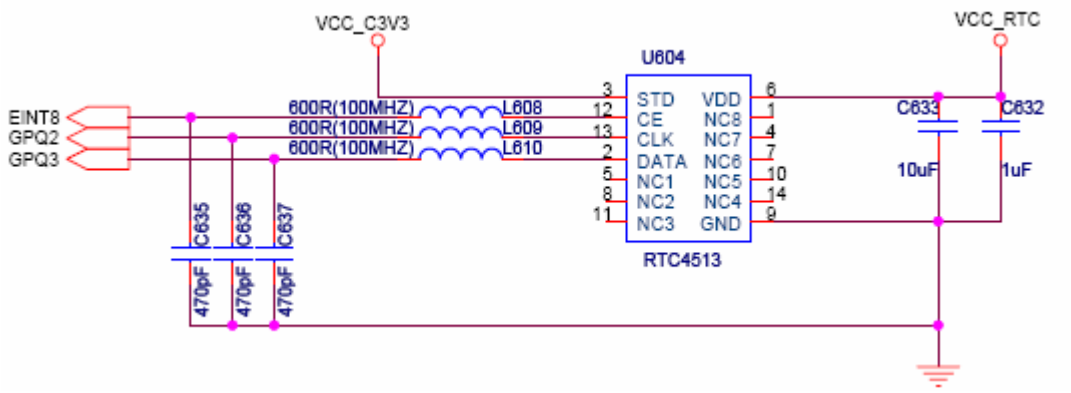
2-16　RTC 介面

本系統提供高精度的 RTC 時鐘，年誤差在 2 分鐘，電路如下；其實物圖如下：





電路圖如下：



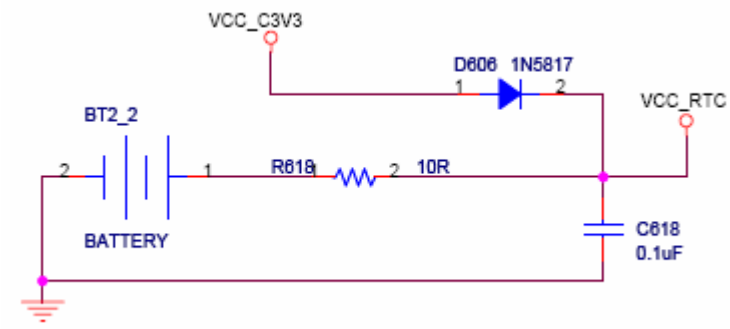


圖 2-37 RTC 介面電路

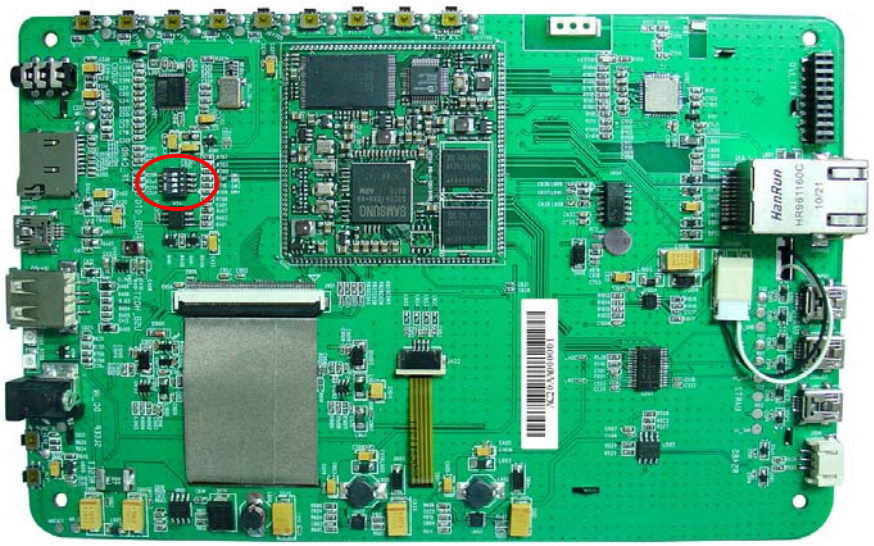
第三章
HMI700-6410S 人機介面使用

3-1　平台設置及連接

在出廠前，通常已將 WinCE 核心或 Linux 核心燒錄到開發平台上了，這一章節介紹的工作環境是在 Windows 下的相關操作。

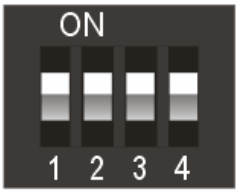
3-1.1　啟動模式選擇

本開發平台可以選擇 Nand Flash 或 TF 卡啓動，啓動模式的選擇透過核心板上的 4 位元撥碼開關（J701）來決定啓動模式的：



- J701 設置爲：0010，從 NAND FLASH（128M）啓動。

- J701 設置爲：1111，從 TF 卡啓動。



出廠時開發平台的啓動方式爲從 Nand Flash 啓動。

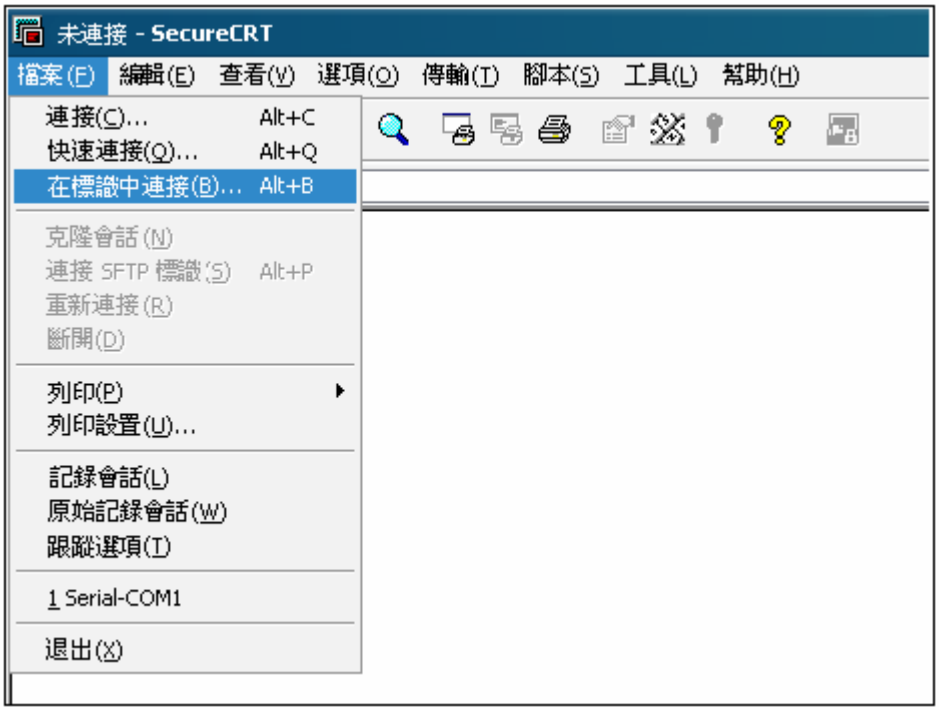
3-1.2 外部硬體連接

1) 使用交叉串列線將開發平台上的 MiniUSB 介面（CON501）與 PC 串列埠相連
2) 將 5V 電源接到開發平台 DC-5V（J601）介面上

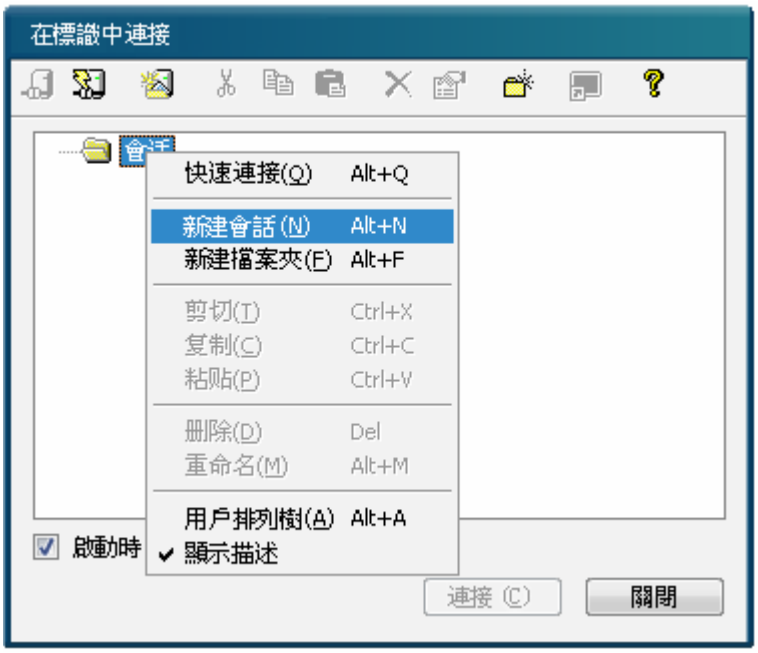
3-1.3 除錯終端配置

這裏除錯終端軟體我們選用 SecureCRT。開啓串列埠終端軟體 Secure CRT，在 Secure CRT 新建一個串列埠會話，步驟如下：

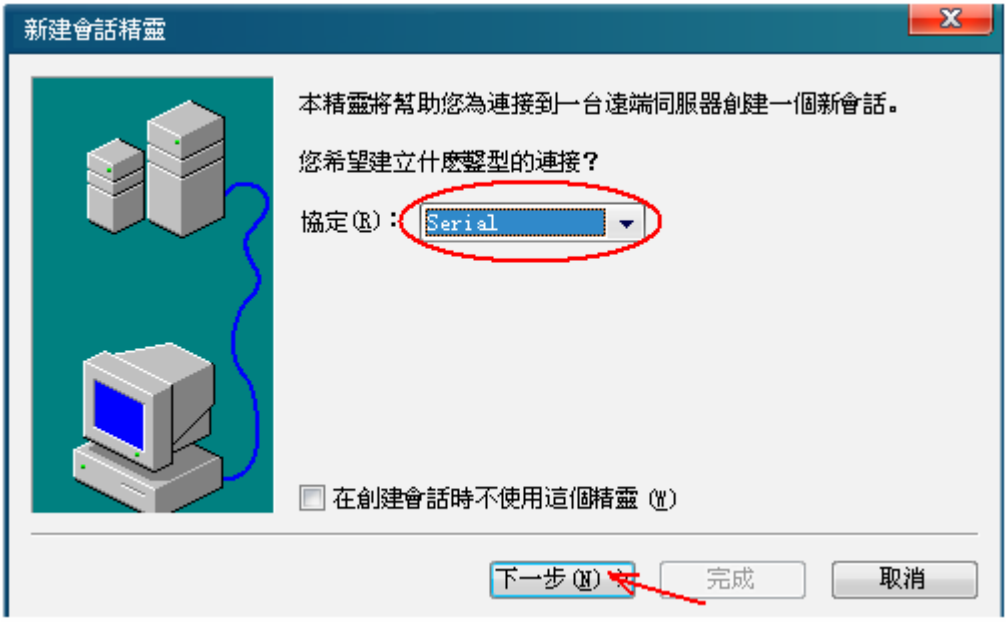
打開軟體 SecureCRT，點選“文件”->“在標籤中連接”



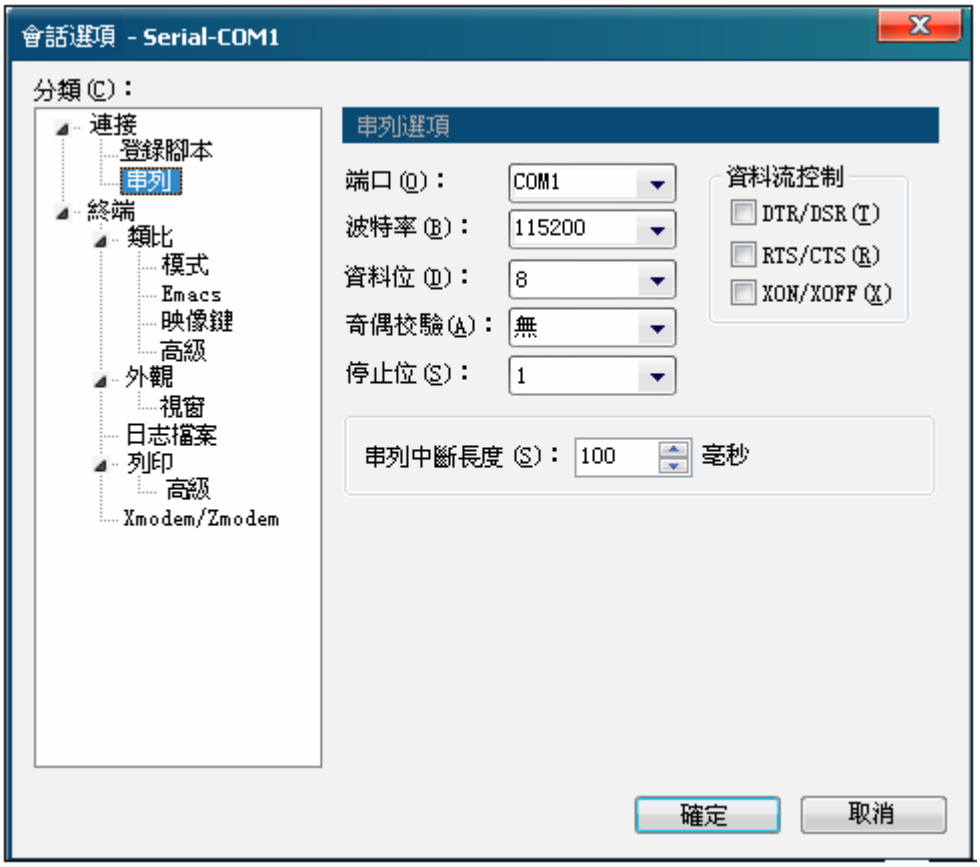
出現如下圖框：



在“會話”上點選右鍵，點選“新建會話”出現如下圖框：



在協定中選擇“Serial”點選“下一步”出現會話選項框，會話選項如圖：


會話選項 - Serial-COM1
分類(C)：
連接
登錄腳本
串列
終端
類比
模式
Emacs
映像鍵
高級
外觀
視窗
日志檔案
列印
高級
Xmodem/Zmodem
串列選項
端口(O)：
COM1
波特率(B)：
115200
資料位(D)：
8
奇偶校驗(A)：
無
停止位(S)：
1
資料流控制
DTR/DSR(T)
RTS/CTS(R)
XON/XOFF(X)
串列中斷長度(S)：
100
毫秒
確定
取消

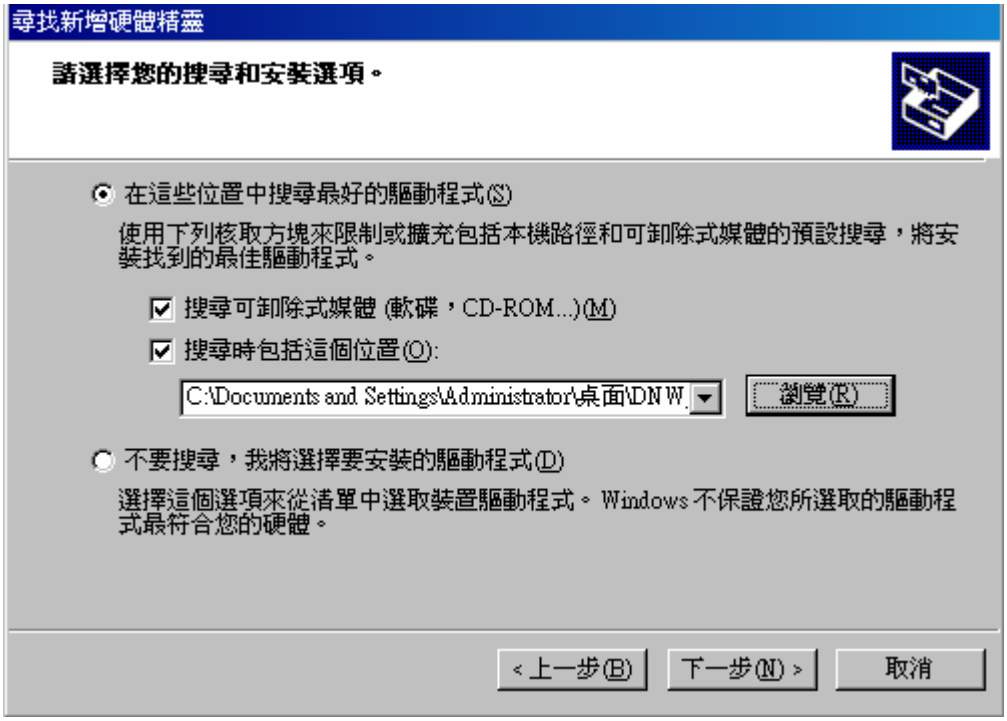
點選“確定”完成會話建立。

3-2 開發平台開機使用

（1） 用 USB 資料線把主機 USB 介面和開發平台 USB_OTG（CON401）連接起來，打開電源進入 u-boot 命令行，在串列埠終端輸入命令：dnw 即會出現硬體安裝精靈

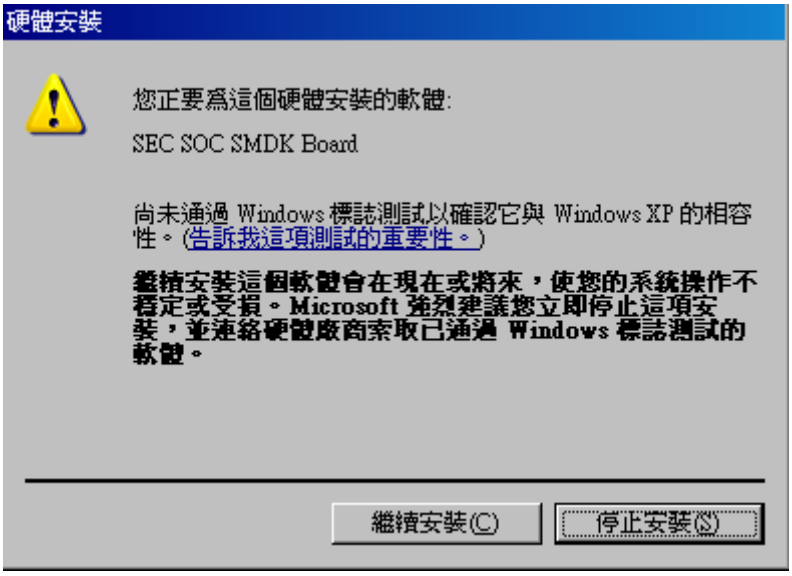
選擇“指定位置安裝”，點“下一步”


尋找新增硬體精靈
歡迎使用尋找新增硬體精靈
這個精靈協助您安裝軟體於:
SEC SOC SMDK Board
如果您的硬體附有安裝 CD 或磁片，現在將它插入。
您要精靈執行什麼工作?
自動安裝軟體(建議選項)(I)
從清單或特定位置安裝(進階)(S)
請按 [下一步] 繼續。
< 上一步(B)
下一步(N) >
取消


尋找新增硬體精靈
請選擇您的搜尋和安裝選項。
在這些位置中搜尋最好的驅動程式(S)
使用下列核取方塊來限制或擴充包括本機路徑和可卸除式媒體的預設搜尋，將安裝找到的最佳驅動程式。
搜尋可卸除式媒體 (軟碟，CD-ROM...)(M)
搜尋時包括這個位置(O):
C:\Documents and Settings\Administrator\桌面\DNW
瀏覽(R)
不要搜尋，我將選擇要安裝的驅動程式(D)
選擇這個選項來從清單中選取裝置驅動程式。Windows 不保證您所選取的驅動程式最符合您的硬體。
< 上一步(B)
下一步(N) >
取消

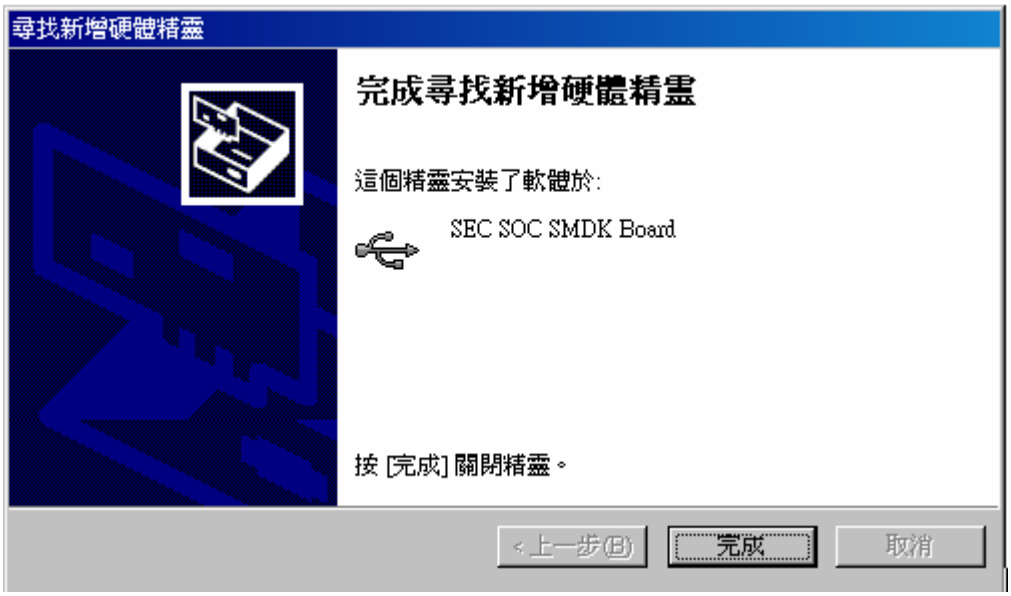
點選“瀏覽”選擇 USB 驅動所在檔案夾：(請選擇隨機附贈光碟片內 Tools 資料夾下面的 DNW_v2.0 資料夾內，DNW2.0 USB Driver 資料夾內的驅動程式安裝，在此筆者習慣將此資料夾內全部資料複製到桌面上)

點選“下一步”：

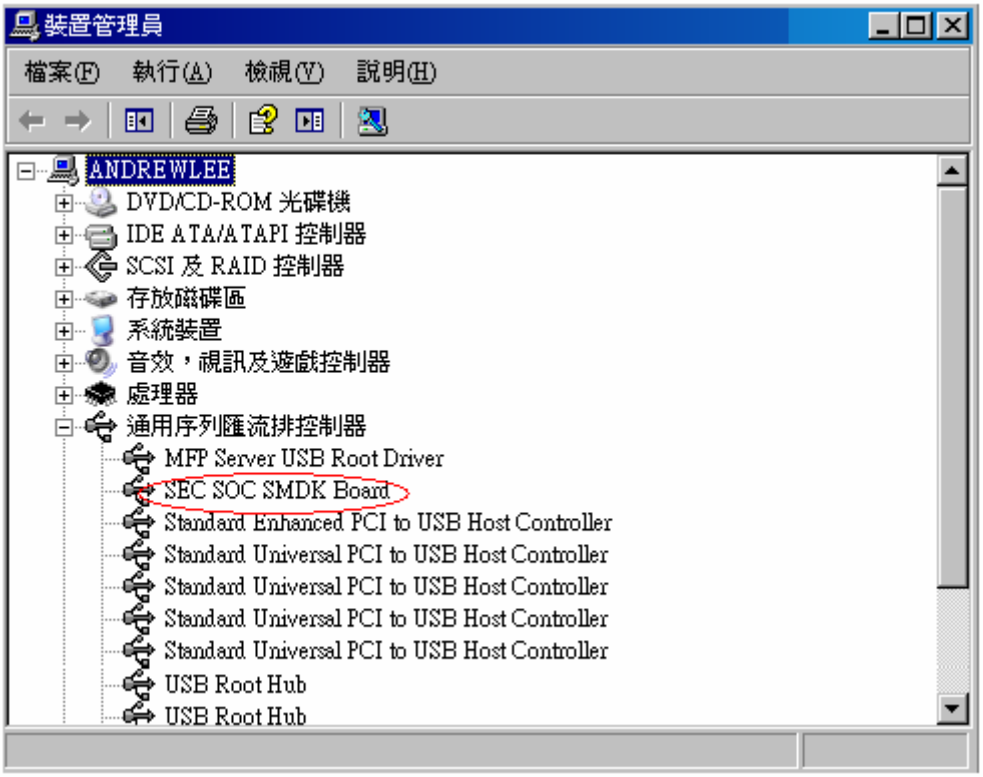


選擇“仍然繼續”：

接下來系統會自動完成驅動安裝，最後點選“完成”來結束安裝：



在系統的設備管理器中，我們可以看到安裝完成了的開發平台。



設備管理器中出現 SEC SOC SMDK Board 說明驅動安裝成功。

第四章
Android 核心編譯

HMI700-6410S 開發平台涉及到的系統檔案主要包括：

1、引導程式 u-boot

2、核心檔案 zImage

3、Android 系統檔案 ramdisk-uboot.img、system.img 和 userdata.img

下面我們就各個檔案的編譯過程進行簡單的講解。而在編譯以上檔案之前我們首先要安裝交叉編譯器，否則我們的編譯就無法成功。

4-1　安裝交叉編譯工具

編譯Linux核心前必須安裝交叉編譯工具，在開發平台的配套光碟中的Linux/cross_compile/4.3.1-eabi-armv6-20080707.tar.bz2是提供的相關交叉編譯器。針對HMI700-6410S開發平台，不論編譯核心還是編譯u-boot都可以使用4.3.1-eabi-armv6作爲交叉編譯器。安裝過程如下：

1、在 Linux 伺服器中創建一個 linux_system 工作目錄，然後把光碟中提供的交叉編譯器 4.3.1-eabi-armv6-20080707.tar.bz2 複製到該目錄下，如下圖 4-1 所示：

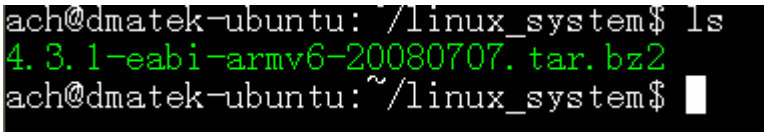


圖 4-1

2、建立存放交叉編譯器的目錄，執行 mkdir /usr/local/arm 這個命令在/usr/local 目錄下創建一個 arm 目錄，如下圖 4-2 所示：

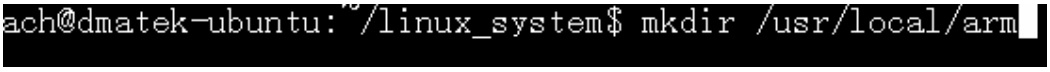


圖 4-2

3、使用命令把交叉編譯器 4.3.1-eabi-armv6-20080707.tar.bz2 解壓到當前目錄下，命令如下：

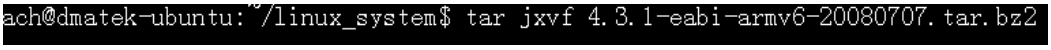


圖 4-3

成功解壓後將得到一個 4.3.1-eabi-armv6 目錄檔，交叉編譯工具所包含的全部檔都包含在該目錄內。下面我們將 4.3.1-eabi-armv6 拷貝到存放交叉編譯器的目錄 /usr/local/arm 內，命令如下：注意下面操作要使用超級用戶 root 許可權。

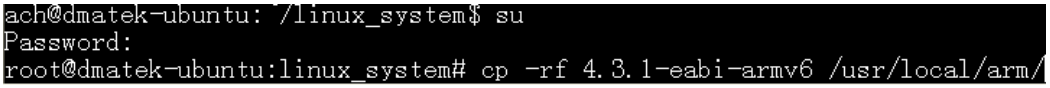


圖 4-4

拷貝完成之後退出超級用戶許可權。爲了以後編譯核心的方便，我們在編譯核心前，首先執行：

export PATH=/usr/local/arm/4.3.1-eabi-armv6/usr/bin:$PATH

這個命令，以後則在編譯核心及其他應用程式時均可用 arm-linux-來指定交叉編譯器。

4-2　解壓 Linux 核心

1、把光碟提供的 Linux 核心壓縮檔案 dma6410-linux-2.6.29_hmi_110719.tar.bz2 複製到 HMI7000_110718 目錄下，如下圖 4-5 所示：

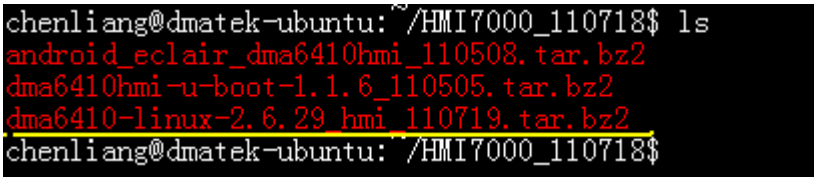


圖 4-5

在 Linux 終端命令行執行 tar –xjvf dma6410-linux-2.6.29_hmi_110719.tar.bz2 這個命令將 Linux 核心解壓縮，解壓完成後將得到一個名爲 dma6410-linux-2.6.29_hmi 的目錄，如下圖 4-6 所示：

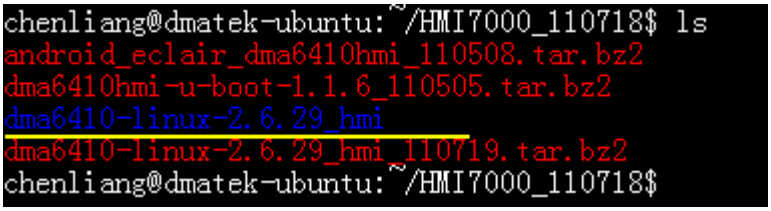


圖 4-6 解壓 Linux 核心包

4-3 編譯 Linux 核心

1. 執行命令：cd dma6410-linux-2.6.29_hmi，進入 dma6410-linux-2.6.29_hmi 核心目錄，執行命令：make clean，清除先前編譯生成的檔案。如下圖 4-7 所示：

```
chenliang@dmatek-ubuntu:~/HMI7000_110718$ cd dma6410-linux-2.6.29_hmi/
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410-linux-2.6.29_hmi$ make clean
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410-linux-2.6.29_hmi$
```

圖 4-7

2. 修改核心目錄下的 Makefile 檔案中交叉編譯器的路徑，執行 vi Makefile 這個命令使用 VI 編輯器開啓 Makefile，找到「CROSS_COMPILE =」這一行，如下圖 4-8 所示：

```
ARCH                    := arm
CROSS_COMPILE   := $(shell if [ -f .cross_compile ]; then \
                                        cat .cross_compile; \
                                        fi)
```

圖 4-8 查看 makefile 檔案

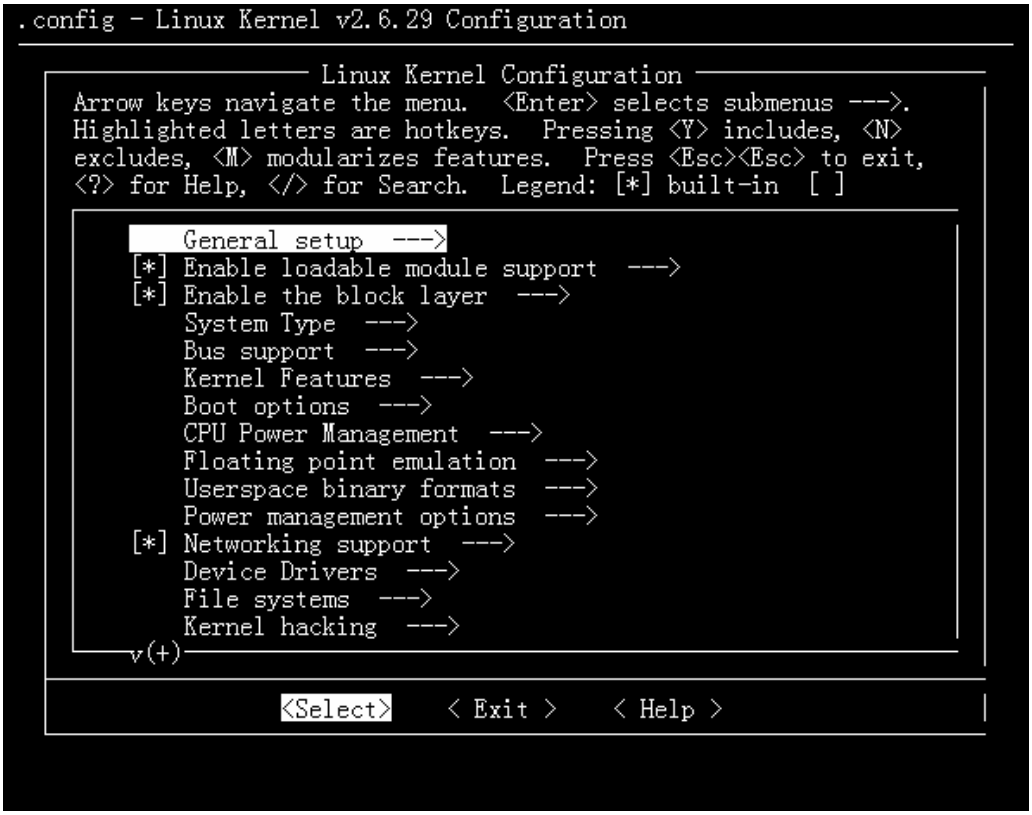
透過上圖可以知道，交叉編譯器的內容是在核心根目錄下的.cross_compile 檔案內，退出 Makefile 檔案，在當前目錄下執行 cat .cross_compile 這個命令來查看.cross_compile 內的內容，如下圖所示：

```
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410-linux-2.6.29_hmi$ cat .cross_compile
/usr/local/arm/4.3.1-eabi-armv6/usr/bin/arm-linux-
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410-linux-2.6.29_hmi$
```

圖 4-9 查看.cross_compile 檔案

若.cross_compile 檔案中的內容不是「/usr/local/arm/4.3.1-eabi-armv6/usr/bin/arm-linux-」，則需要修改成上面的交叉編譯器及其路徑，修改完成後儲存退出。

3. 輸入命令：make menuconfig，進入如下圖所示介面。



HMI700-6410S 標配的 LCD 大小爲 800X480，所以在編譯核心前需要進行相關配置，如下：依次進入下列目錄：

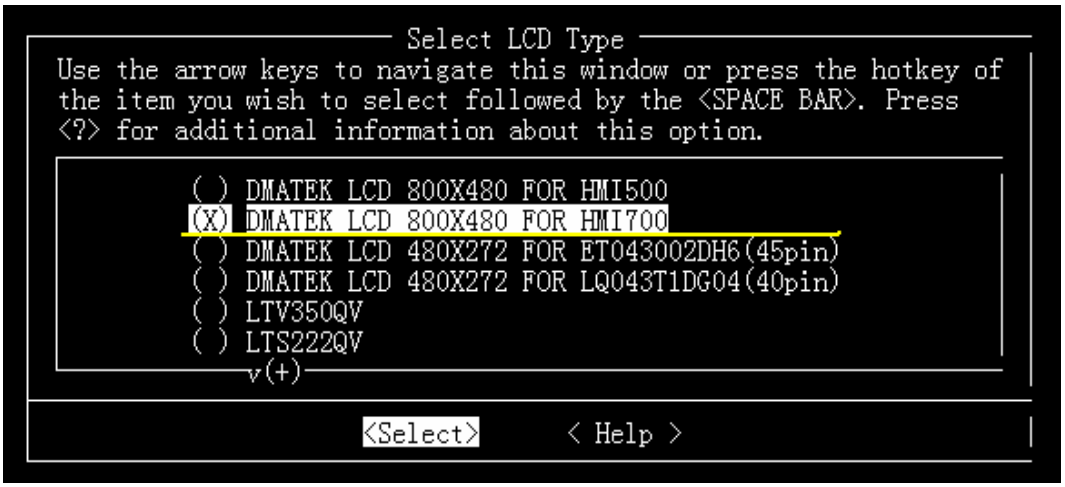
Device Drivers--->

Graphics support--->

Support for frame buffer devices--->

Select LCD Type (DMATEK LCD 800X480 FOR HMI700) --->

進入下圖所示介面：



把游標停留在 DMATEK LCD 800X480 FOR HMI700 選項，按 ENTER 健即可選中此選項。然後退出即可。

4. 執行 make zImage 這個命令編譯核心，如下圖 4-10 所示：

   注意：smdk6410_android_128M_defconfig 是我們針對 HMI700-6410S 開發平台 NAND FLASH128M 的 Android 核心配置檔，它被保存在核心 arch/arm/configs/ 目錄內，在配置核心出錯的情況下 User 可以調用它來還原初始化配置。

```
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410-linux-2.6.29_hmi$ make zImage
  HOSTLD  scripts/kconfig/conf
scripts/kconfig/conf -s arch/arm/Kconfig
drivers/staging/android/Kconfig:3:warning: type of 'ANDROID' redefined from 'tristate' to 'boolean'
  CHK     include/linux/version.h
  UPD     include/linux/version.h
  Generating include/asm-arm/mach-types.h
  CHK     include/linux/utsrelease.h
  UPD     include/linux/utsrelease.h
  SYMLINK include/asm -> include/asm-arm
  CC      kernel/bounds.s
  GEN     include/linux/bounds.h
  CC      arch/arm/kernel/asm-offsets.s
  GEN     include/asm/asm-offsets.h
  CALL    scripts/checksyscalls.sh
<stdin>:1097:2: warning: #warning syscall fadvise64 not implemented
<stdin>:1265:2: warning: #warning syscall migrate_pages not implemented
<stdin>:1321:2: warning: #warning syscall pselect6 not implemented
<stdin>:1325:2: warning: #warning syscall ppoll not implemented
<stdin>:1365:2: warning: #warning syscall epoll_pwait not implemented
  CC      scripts/mod/empty.o
  HOSTCC  scripts/mod/mk_elfconfig
  MKELF   scripts/mod/elfconfig.h
  HOSTCC  scripts/mod/file2alias.o
  HOSTCC  scripts/mod/modpost.o
```

圖 4-10 編譯核心

5. 編譯成功後將會在 arch/arm/boot/目錄下生成核心映像檔案 zImage，如下圖 4-11 所示：

```
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410-linux-2.6.29_hmi$ ls
arch/arm/boot/
bootp  compressed  Image  install.sh  Makefile  zImage
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410-linux-2.6.29_hmi$
```

圖 4-11

6. 前述步驟編譯所得到的核心映像檔 zImage 是可以被下載寫入開發平台的，但是若要在開發平台上完整啓動 Linux 則必須將 zImage 及根檔案系統寫入到平台的 Flash 裡。若僅將 zImage 寫入平台，則可以見到核心啓動的訊息，但並不能進入讀取系統設定及進入 Linux 終端。

   在光碟內提供有根檔案系統的二進位檔，讀者可以使用光碟內的檔案；也可以自行製作一個根檔案系統。

4-4 修改 Linux 核心 DM9000 網路的 MAC 位址

由於 DM9000 網路的 MAC 位址是被直接寫到 Linux 核心中，所以我們在更改網路的 MAC 位址時，必須得從核心著手。修改方法如下：

1、在核心目錄 arch/arm/palt-s3c64xx/下找到 devs.c 程式文件。

2、進入 devs.c 程式中，找到對 DM9000 網路的暫存器配置項，如下：

```
/* DM9000 LAN via ROM interface  add ach 2010/01/04*/
static struct resource s3c_dm9000_resources[] = {
    [0] = {
          .start  = S3C64XX_PA_DM9000 + 0x300,
          .end    = S3C64XX_PA_DM9000 + 0x300 + 0x4 - 1,
          .flags  = IORESOURCE_MEM,
    },
    [1] = {
          .start = IRQ_EINT(0),
          .end   = IRQ_EINT(0),
          .flags = IORESOURCE_IRQ,
     }
};

static struct dm9000_plat_data dm9000_setup = {
   .flags        = DM9000_PLATF_16BITONLY,
   .dev_addr  = {0x00,0xe0,0x4a,0xbc,0x15,0xe7}
};

struct platform_device s3c_device_dm9000 = {
     .name          = "dm9000",
     .id            =  0,
     .num_resources  = ARRAY_SIZE(s3c_dm9000_resources),
     .resource       = s3c_dm9000_resources,
     .dev ={
                .platform_data = &dm9000_setup,
      }
};

EXPORT_SYMBOL(s3c_device_dm9000);
```

3、從上圖可以看到一個 dm9000_setup 的結構體配置項，其中對應的.dev_addr 項就是對網路 MAC 位址的設定。這裏也就是需要修改 MAC 位址的地方。

4、如我們爲 USER 提供了一個新的 MAC 位址，如下：

5、把新的 MAC 位址號{0xac,0x20,0xaa,0x00,0x00,0x01}直接替換掉原始的 MAC 地址{0x00,0xe0,0x4a,0xbc,0x15,0xe7}即可。如下：

```
static struct dm9000_plat_data dm9000_setup = {
    .flags        = DM9000_PLATF_16BITONLY,
    //.dev_addr  = {0x00,0xe0,0x4a,0xbc,0x15,0xe7}
    .dev_addr  = {0xac,0x20,0xaa,0x00,0x00,0x01}
};
```

6、替換以後保存退出。從新編譯核心並把得到的 zImage 檔燒寫到開發平台上。

7、爲了驗證我們的網路 MAC 位址是否修改正確。在 Android 系統終端輸入 busybox ifconfig 命令，如下。

```
# busybox ifconfig
eth0      Link encap:Ethernet  HWaddr AC:20:AA:00:00:01
          inet addr:192.168.1.202  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::ae20:aaff:fe00:1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1070 (1.0 KiB)  TX bytes:468 (468.0 B)
          Interrupt:101 Base address:0x8300
```

8、得到如上圖所示的資訊就說明我們對網路的 MAC 位址修改成功。

4-5　編譯 U-BOOT

U-Boot 是一個通用的 Bootloader，它也是開發平台 Linux 系統能否順利啓動的重要關鍵。以下的內容將說明如何編譯 U-Boot。

1、把光碟提供的 u-boot 壓縮檔拷貝到 HMI7000_110718 目錄內，然後執行命令：tar jxvf dma6410hmi-u-boot-1.1.6_110505.tar.bz2，將 U-Boot 原始程式碼解壓縮，成功解壓縮後將得到 dma6410hmi-u-boot-1.1.6 這個目錄，如下圖 4-12 所示：

```
chenliang@dmatek-ubuntu:~/HMI7000_110718$ ls
android_eclair_dma6410hmi_110508.tar.bz2
dma6410hmi-u-boot-1.1.6
dma6410hmi-u-boot-1.1.6_110505.tar.bz2
dma6410-linux-2.6.29_hmi
dma6410-linux-2.6.29_hmi_110719.tar.bz2
chenliang@dmatek-ubuntu:~/HMI7000_110718$
```

圖 4-12

2、進入目錄 dma6410hmi-u-boot-1.1.6，並執行 make clean 這個命令清除以前編譯生成的檔案，如下圖 4-13 所示：

```
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410hmi-u-boot-1.1.6$ make clean
find /home/chenliang/HMI7000_110718/dma6410hmi-u-boot-1.1.6 -type f \
                \( -name 'core' -o -name '*.bak' -o -name '*~' \
                -o -name '*~' -o -name '.depend*' \
                -o -name '*.o'  -o -name '*.a'  \) -print \
                | xargs rm -f
rm -f u-boot*
rm -f examples/hello_world examples/timer \
              examples/eepro100_eeprom examples/sched \
              examples/mem_to_mem_idma2intr examples/82559_eeprom \
              examples/smc91111_eeprom examples/interrupt \
              examples/test_burst
rm -f tools/img2srec tools/mkimage tools/envcrc \
                tools/gen_eth_addr
rm -f tools/mpc86x_clk tools/ncb
rm -f tools/easylogo/easylogo tools/bmp_logo
rm -f tools/gdb/astest tools/gdb/gdbcont tools/gdb/gdbsend
rm -f tools/env/fw_printenv tools/env/fw_setenv
rm -f board/cray/L1/bootscript.c board/cray/L1/bootscript.image
rm -f board/netstar/eeprom board/netstar/crcek board/netstar/crcit
rm -f board/netstar/*.srec board/netstar/*.bin
rm -f board/trab/trab_fkt board/voiceblue/eeprom
rm -f board/integratorap/u-boot.lds board/integratorcp/u-boot.lds
rm -f include/bmp_logo.h
rm -f nand_spl/u-boot-spl nand_spl/u-boot-spl.map
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410hmi-u-boot-1.1.6$
```

圖 4-13 make clean

3、修改 Makefile，請依據採用的交叉編譯器指定正確的路徑，如下圖 4-14 所示：(提示：編譯 HMI700-6410S 的 u-boot 所用的交叉編譯器爲 4.3.1-eabi-armv6。)

```
CROSS_COMPILE = /usr/local/arm/4.3.1-eabi-armv6/usr/bin/arm-linux-
export  CROSS_COMPILE

# load other configuration
include $(TOPDIR)/config.mk
```

圖 4-14 修改 Makefile

4、執行 make smdk6410_config 這個命令選擇使用 smdk6410_config 這個設定選項。如下圖 4-15 所示。

```
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410hmi-u-boot-1.1.6$ make smdk6410_config
Configuring for smdk6410 board...
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410hmi-u-boot-1.1.6$
```

圖 4-15

5、執行命令：make，進行 u-boot 的編譯。編譯成功後，可以在當前目錄內找到 u-boot.bin 這個檔案，如下圖 4-16 所示：

```
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410hmi-u-boot-1.1.6$ ls
arm_config.mk                   examples                mips_config.mk
avr32_config.mk                 fs                      mkconfig
based_on_2006-12-06             i386_config.mk          nand_spl
blackfin_config.mk              include                 net
board                           lib_arm                 nios2_config.mk
build_lcd_43.sh                 lib_avr32               nios_config.mk
build_lcd_7050.sh               lib_blackfin            post
CHANGELOG                       lib_generic             ppc_config.mk
CHANGELOG-before-U-Boot-1.1.5   lib_i386                README
Changelog_Samsung               lib_m68k                rtc
common                          lib_microblaze          rules.mk
config.mk                       lib_mips                System.map
COPYING                         lib_nios                tools
cpu                             lib_nios2               u-boot
CREDITS                         lib_ppc                 u-boot.bin
CVS                             m68k_config.mk          u-boot.dis
disk                            MAINTAINERS             u-boot.map
doc                             MAKEALL                 u-boot.srec
drivers                         Makefile
dtt                             microblaze_config.mk
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410hmi-u-boot-1.1.6$
```

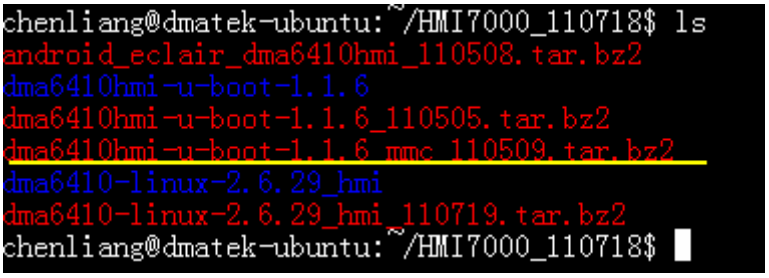
圖 4-16 編譯 u-boot

到此已經成功編譯了 u-boot 的二進位映像檔案 u-boot.bin，接著可以利用 TF 卡將 u-boot.bin 寫入 Nand Flash 裡，然後驗證 U-Boot 的功能是否正常。

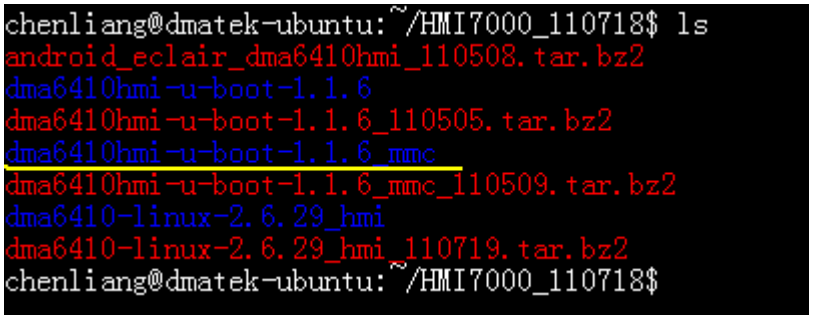
4-6 編譯 U-BOOT-MMC

HMI700-6410S 平台主要使用 TF 卡燒寫系統檔案。在 HMI700-6410S 平台從 TF 卡啓動之前需將 u-boot_mmc.bin 檔案燒寫到 SD 卡中。下面將介紹如何編譯 u-boot_mmc.bin

1、把光碟中提供的 u-boot_mmc 壓縮檔 dma6410hmi-u-boot-1.1.6_mmc_110509.tar.bz2 拷貝到 HMI7000_110718 目錄內，如下圖所示：



2、執行命令：tar jxvf dma6410hmi-u-boot-1.1.6_mmc_110509.tar.bz2。將 u-boot_mmc 壓縮檔解壓縮，成功解壓縮後將得到 dma6410hmi-u-boot-1.1.6_mmc 這個目錄，如下圖所示：



3、執行命令：cd dma6410hmi-u-boot-1.1.6_mmc，進入到 dma6410hmi-u-boot-1.1.6_mmc 目錄。執行 make clean 這個命令清楚以前編譯生成的檔案，如下圖所示：

```
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410hmi-u-boot-1.1.6_mmc$ make clean
find /home/chenliang/HMI7000_110718/dma6410hmi-u-boot-1.1.6_mmc -type f \
                \( -name 'core' -o -name '*.bak' -o -name '*~' \
                -o -name '*~' -o -name '.depend*' \
                -o -name '*.o'  -o -name '*.a'  \) -print \
                | xargs rm -f
rm -f u-boot* xa* temp
rm -f examples/hello_world examples/timer \
              examples/eepro100_eeprom examples/sched \
              examples/mem_to_mem_idma2intr examples/82559_eeprom \
              examples/smc91111_eeprom examples/interrupt \
              examples/test_burst
rm -f tools/img2srec tools/mkimage tools/envcrc \
                tools/gen_eth_addr
rm -f tools/mpc86x_clk tools/ncb
rm -f tools/easylogo/easylogo tools/bmp_logo
rm -f tools/gdb/astest tools/gdb/gdbcont tools/gdb/gdbsend
rm -f tools/env/fw_printenv tools/env/fw_setenv
rm -f board/cray/L1/bootscript.c board/cray/L1/bootscript.image
rm -f board/netstar/eeprom board/netstar/crcek board/netstar/crcit
rm -f board/netstar/*.srec board/netstar/*.bin
rm -f board/trab/trab_fkt board/voiceblue/eeprom
rm -f board/integratorap/u-boot.lds board/integratorcp/u-boot.lds
rm -f include/bmp_logo.h
rm -f nand_spl/u-boot-spl nand_spl/u-boot-spl.map
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410hmi-u-boot-1.1.6_mmc$
```

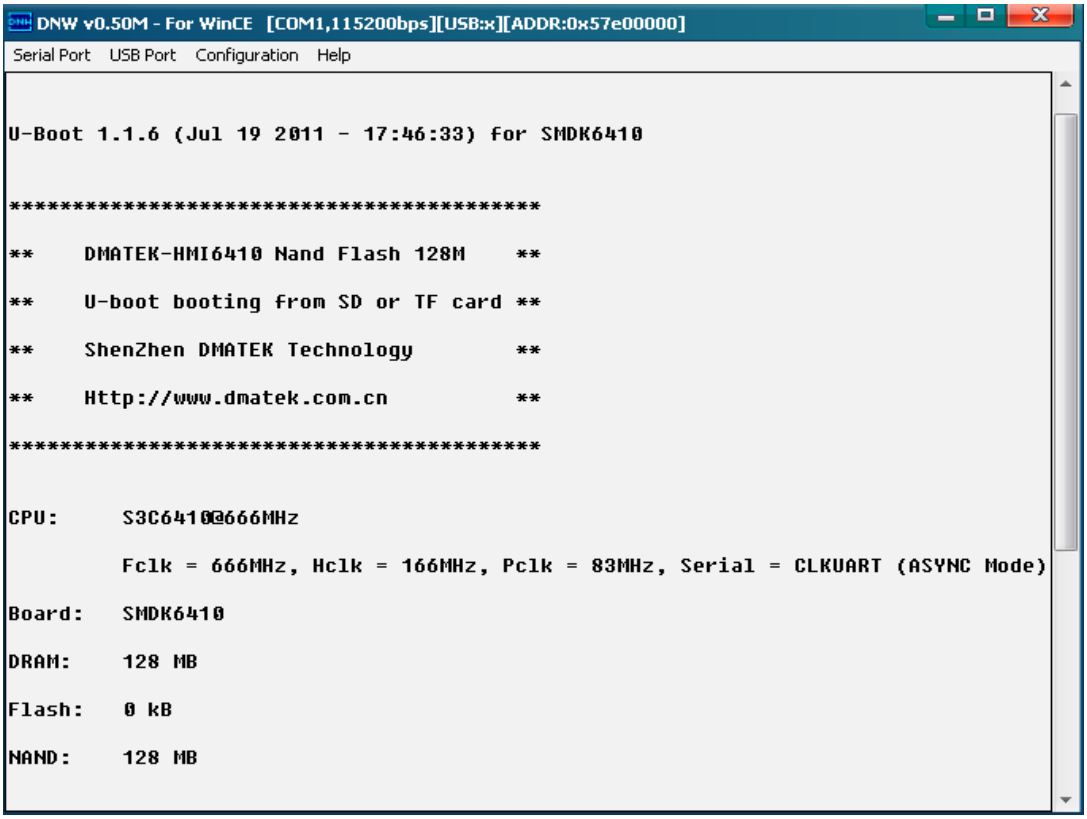
4、修改 Makefile，根據採用的交叉編譯器指定正確的路徑，如下圖所示：

```
CROSS_COMPILE = /usr/local/arm/4.3.1-eabi-armv6/usr/bin/arm-linux-
export   CROSS_COMPILE

# load other configuration
include $(TOPDIR)/config.mk
```

5、執行命令：source build_mmc.sh 編譯 u-boot 原始程式碼。編譯成功後在當前目錄下可以找到 u-boot_mmc.bin 檔案，如下圖所示：

```
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410hmi-u-boot-1.1.6_mmc$
ls
arm_config.mk                   fs                      mkconfig
avr32_config.mk                 i386_config.mk          nand_spl
based_on_2006-12-06             include                 net
blackfin_config.mk              lib_arm                 nios2_config.mk
board                           lib_avr32               nios_config.mk
build_mmc.sh                    lib_blackfin            post
CHANGELOG                       lib_generic             ppc_config.mk
CHANGELOG-before-U-Boot-1.1.5   lib_i386                README
Changelog_Samsung               lib_m68k                rtc
common                          lib_microblaze          rules.mk
config.mk                       lib_mips                System.map
COPYING                         lib_nios                tools
cpu                             lib_nios2               u-boot
CREDITS                         lib_ppc                 u-boot.bin
CVS                             m68k_config.mk          u-boot.dis
disk                            MAINTAINERS             u-boot.map
doc                             MAKEALL                 u-boot_mmc.bin
drivers                         Makefile                u-boot.srec
dtt                             microblaze_config.mk
examples                        mips_config.mk
chenliang@dmatek-ubuntu:~/HMI7000_110718/dma6410hmi-u-boot-1.1.6_mmc$
```

至此，已成功編譯了 u-boot_mmc 的二進位映像檔 u-boot_mmc.bin。可以將 u-boot_mmc.bin 燒寫到 TF 卡中，讓 HMI700-6410S 平台從 TF 啓動，進入如下圖所示介面。



進入上圖所示介面後，說明 u-boot_mmc.bin 編譯成功，可以進行其他燒寫操作了。

注意：build_mmc.sh 只是一個腳本檔。在 build_mmc.sh 檔案中有製作 u-boot_mmc.bin 檔案的詳細命令。

4-7 安裝 Android 系統開發套件

本小節說明開發 Android 系統時所需要安裝的套件，在 Android 的說明檔裡也相當完整的說明，建議讀者在閱讀完本節的說明後，也可以花些時間由 Android 原始碼網站（http://source.android.com/）閱讀 Android 的說明檔，如此一來就會更清楚了。

若讀者想深入瞭解每一個工具套件的用途，Ubuntu 的工具套件都包含了許多說明檔，讀者可以查詢 apt-get 的使用方法以瞭解如何取得或閱讀這些說明檔，另外在 Ubuntu 的 Ubuntu Packages Search 頁面（http://packages.ubuntu. com/）也可以查詢每一個工具套件的詳細資料。

在開始安裝套件之前，請先更新套件資訊，所執行的命令如下：

```
sudo apt-get update
```

接著以下依次說明所需要安裝的工具套件：

1. JAVA JDK

```
sudo apt-get install sun-java6-jdk
```

注意：我們選擇安裝的是 sun-java6-jdk 的 JDK，讀者也可以安裝 sun-java5-jdk 的 JDK 來編譯 android。但在編譯事需要修改對應的腳本檔，下面我們都有介紹。

2. FLEX

Flex 是一個用來生成程式碼掃描器的工具。

```
sudo apt-get install flex
```

3. BISON

Bison 是一個多方面用途的解析產生器。

```
sudo apt-get install bison
```

4. gperf

gperf 是一個可以產生完美 hash 函數的程式。

```
sudo apt-get install gperf
```

5. libsdl-dev

SDL 是一個函式庫，允許合適的程式存取視訊緩衝區、音訊輸出，或滑鼠和鍵盤等。

```
sudo apt-get install libsdl-dev
```

6. libesd0-dev

這些程式套件把數位音訊流整合在一起，以便設備能夠重播。

```
sudo apt-get install libesd0-dev
```

7. libwxgtk2.6-dev

該包爲編譯 wxWidgets 程式提供需要的檔。

```
sudo apt-get install libwxgtk2.6-dev
```

8. zlib1g-dev

Zlib 是一個實現壓縮檔的函式庫。

```
sudo apt-get install zlib1g-dev
```

9. build-essential

該套件包括爲建立 Debian 套件必不可少的包資訊清單。

```
sudo apt-get install build-essential
```

10. libncurses5-dev

該套件包括一些標頭檔、靜態程式庫和符號連結。

```
sudo apt-get install libncurses5-dev
```

4-8 編譯 Android 檔案系統

Android 系統能否正常啓動與我們的編譯過程直接關聯。下面就詳細講述一下 Android 檔案系統的編譯過程。

1、把光碟提供的 Android 壓縮檔 android_eclair_dma6410hmi_110508.tar.bz2 複製到 HMI7000_110718 目錄下，如下圖 4-17 所示：

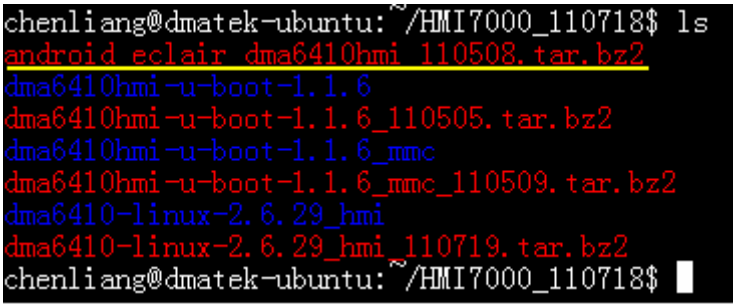


圖 4-17

2、輸入命令：tar -xjvf android_eclair_dma6410hmi_110508.tar.bz2 解壓縮 Android 原始程式碼檔，如下：

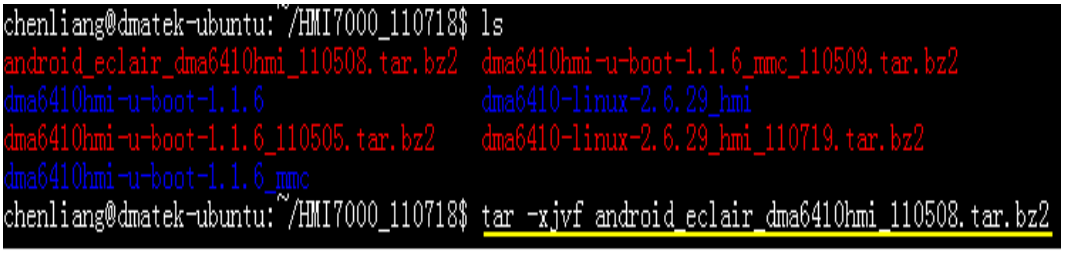


圖 4-18

3、成功解壓縮後，進入 Android 目錄檔 android_eclair，仔細查看一下 Android2.1 原始程式碼下的相關目錄檔，其中 android_eclair 目錄內包含了編譯 Andorid 的所有源程式碼，如下：

```
chenliang@dmatek-ubuntu:~/HMI7000_110718$ ls
android_eclair
android_eclair_dma6410hmi_110508.tar.bz2
dma6410hmi-u-boot-1.1.6
dma6410hmi-u-boot-1.1.6_110505.tar.bz2
dma6410hmi-u-boot-1.1.6_mmc
dma6410hmi-u-boot-1.1.6_mmc_110509.tar.bz2
dma6410-linux-2.6.29_hmi
dma6410-linux-2.6.29_hmi_110719.tar.bz2
chenliang@dmatek-ubuntu:~/HMI7000_110718$
```

圖 4-19

在 Android 原始程式碼檔內，Makefile 是頂層的腳本檔，進入 Makefile 檔內部，我們發現真正的編譯腳本檔是 build/core/main.mk，爲了方便讀者的操作，我們對 main.mk 檔進行了修改，使得 Android 原始程式碼即支援 sun-java6-jdk，同時又支援 sun-java5-jdk。這樣一來，讀者就不用再爲自己安裝的是那個版本的 JDK 而發愁了。可以直接編譯 Android 程式碼。程式碼修改部分請查看 main.mk 檔案。

4、爲了方便 USER 對 android source 的編譯，我們在 android_eclair 目錄下設置了一個 script 目錄，如下圖所示，script 目錄又中包含了三個檔，分別是 build_android.sh 和 mkimage，其中 mkimage 是一個檔轉換工具，build_android.sh 是編譯腳本檔案，如下圖所示：

```
chenliang@dmatek-ubuntu:~/HMI7000_110718$ cd android_eclair/
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair$ ls
bionic          cts          external    Makefile  script
bootable        dalvik       filesystem  ndk       sdk
build           development  frameworks  packages  system
build_android   device       hardware    prebuilt  vendor
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair$ ls script/
build_android.sh  mkimage
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair$
```

圖 4-20

5、在 android_eclair 目錄下，我們再創建一個設置編譯環境變數的腳本檔案 build_android，輸入命令：vi build_android 進入腳本內部，如下圖所示：

```
#!/bin/bash

ROOT_DIR=$(pwd)

export SEC_PRODUCT=generic
export SEC_OUTDIR=$ROOT_DIR/filesystem
export PATH=$PATH:$ROOT_DIR/script
```

圖 4-21

其中 ROOT_DIR 爲當前目錄的絕對路徑。“SEC_PRODUCT=generic”表示我們使用的是 Android 的基本架構， “SEC_OUTDIR=$ROOT_DIR /filesystem”該命令設置的是檔案輸出目錄，Android 檔案系統編譯最後會生成三個.img 檔案，該處設置輸出目錄爲 android_eclair/filesystem，因此最後生成的三個.img 檔案會被拷貝到 android_eclair/filesystem 目錄下。“PATH=$PATH:$ROOT_DIR/script” 該命令是把腳本的路徑告訴 Linux 系統，讀者根據自己的位置重新設置。

6、另外在 vendor 目錄下，我們創建了一個 dmatek 目錄，而在 dmatek 目錄中又包含了 system、bluetooth 和 wifi 三個目錄檔案。如下圖所示：

```
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair$ cd vendor/
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair/vendor$ ls
aosp  dmatek  htc  pv-open  qcom  sample
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair/vendor$ cd dmatek/
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair/vendor/dmatek$ ls
Android.mk  bluetooth  system  wifi
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair/vendor/dmatek$ cd system/
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair/vendor/dmatek/system$ ls
app  bin  etc  framework  lib  media  xbin
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair/vendor/dmatek/system$
```

圖 4-22

7、瞭解了 Android 2.1 原始程式碼的結構之後，進入到 android_eclair 目錄內，輸入 source build_android 命令設置編譯環境。如下圖所示：

```
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair$ ls
bionic    build_android  development  filesystem  Makefile  prebuilt  system
bootable  cts            device       frameworks  ndk       script    vendor
build     dalvik         external     hardware    packages  sdk
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair$ source build_android
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair$
```

輸入命令 echo $PATH 可以查看腳本的設置情況，如下：

```
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair$ echo $PATH
/usr/local/android/jdk1.6.0_14/bin:/usr/local/android/jdk1.6.0_14/jre/bin:/usr/local/arm/3.4.1/
bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/home/chenliang/bin
:/home/chenliang/HMI7000_110718/android_eclair/script
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair$
```

黃线部分就是我們 source 執行的結果。

8、輸入命令 build_android.sh(只有在執行 source build_android 命令的前提下，腳本 build_android.sh 才有效。)進行編譯 Android 原始程式碼，如下：

```
ach@dmatek-ubuntu:android_eclair$ build_android.sh

                Build android2.1 for Dmatek


[[[[[[[ Build android2.1 platform for dma6410 ]]]]]]]

make -j4

============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=2.1-update1
TARGET_PRODUCT=generic
TARGET_BUILD_VARIANT=eng
TARGET_SIMULATOR=
TARGET_BUILD_TYPE=release
TARGET_ARCH=arm
HOST_ARCH=x86
HOST_OS=linux
HOST_BUILD_TYPE=release
BUILD_ID=ECLAIR
============================================
```

圖 4-23

這個過程大約要花費 2~3 個小時，而具體的時間要根據讀者的 PC 配置來決定。

9、 經過漫長的編譯過程，最終我們會看到如下圖所示，就表示我們編譯成功了，如下：

```
[[[[[[[ Make ramdisk image for u-boot ]]]]]]]

Image Name:   ramdisk
Created:      Sun Sep 19 16:31:15 2010
Image Type:   ARM Linux RAMDisk Image (uncompressed)
Data Size:    159554 Bytes = 155.81 kB = 0.15 MB
Load Address: 0x50800000
Entry Point:  0x50800000

[[[[[[[ Transfer images to tftp server ]]]]]]]

ramdisk-uboot.img -> /home/ach/linux_system/android_eclair/filesystem
system.img -> /home/ach/linux_system/android_eclair/filesystem
userdata.img -> /home/ach/linux_system/android_eclair/filesystem

[[[[[[[ u-boot commands for fusing ]]]]]]]

tftp 50008000 ramdisk-uboot.img;nand erase 900000 100000;nand write 5000
8000 900000 100000
tftp 50008000 system.img;nand erase a00000 4300000;nand write.yaffs 5000
8000 a00000 46239c0
tftp 50008000 userdata.img;nand erase 9000000 7000000;nand write.yaffs 5
0008000 9000000 214200

cache erace: nand erase 4d00000 4300000

ok success !!!
ach@dmatek-ubuntu:android_eclair$
```

圖 4-24

10、 進入 android_eclair/filesystem 目錄，我們會看到 system.img、userdata.img 和 ramdisk- uboot.img 三個檔案，如下圖所示：

```
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair$ cd filesystem/
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair/filesystem$ ls
ramdisk-uboot.img  system.img  userdata.img
chenliang@dmatek-ubuntu:~/HMI7000_110718/android_eclair/filesystem$
```

圖 4-25

11、 讀者也許會提問，Android 編譯生成的檔不是在 out/target/product/generic 目錄下嗎？這裏需要向讀者解釋一下。我們在編譯 Android 原始碼的過程中使用了 build_android.sh 這個腳本檔。而在 build_android.sh 腳本內部我們對 out/target/product/generic 目錄下的 system.img、userdata.img 和 ramdisk.img 這三個檔進行了處理，首先使用命令 mkimage -A arm -O linux -T ramdisk -C none -a 0x50800000 –n "ramdisk" -d ramdisk.img ramdisk-uboot.img 把 ramdisk.img 轉換爲 ramdisk-uboot.img 檔案，然後再把 ramdisk-uboot.img 等三個檔拷貝到 filesystem 目錄下。如下：

```
function make_uboot_img()
{
        cd out/target/product/$SEC_PRODUCT
        echo
        echo '[[[[[[[ Make ramdisk image for u-boot ]]]]]]]'
        echo

        mkimage -A arm -O linux -T ramdisk -C none -a 0x50800000 -n "ramdisk"
 -d ramdisk.img ramdisk-uboot.img
        check_exit

        echo
        echo '[[[[[[[ Transfer images to tftp server ]]]]]]]'
        echo

        transfer_img ramdisk-uboot.img
        transfer_img system.img
        transfer_img userdata.img

        rm -f ramdisk-uboot.img
```

圖 4-26

12、 在 filesystem 目錄下的三個 img 檔就是 Android 的系統檔，直接燒寫到 HMI700-6410S 開發平台即可。

第五章
燒錄和啟動 Android 2.1 系統

5-1　燒寫準備工作

拿到一塊嶄新的開發平台，通常這個平台是個裸機（即：平台存儲設備中沒有任何程式），要想讓Android系統在開發平台上執行起來，必須給他裝載一些必要的程式，如u-boot、kernel等一些系統二進位檔。通常在這些開發平台中，掉電不丟失資料的存儲設備都採用Flash存儲體，HMI700-6410S這款開發平台採用的是Nand Flash和Nor Flash兩種存儲體，而在執行過程中主要選擇的是NAND FLASH。

Bootloader是開發平台進入作業系統之前執行的一段程式碼，主要用於完成硬體設置和作業系統啓動前的過渡，從而爲作業系統提供基本的執行環境，其功能類似於PC主機的BIOS程式。u-boot是嵌入式Linux系統中常用的一款Bootloader，由於它的功能比較強大，在當前比較流行。

Android系統的燒寫過程大致分爲以下3個部分。

1、燒寫u-boot

2、燒寫kernel

3、燒寫Android檔案系統

Android的檔案系統包含了3個鏡像檔，分別是：ramdisk-uboot.img、system.img、

userdata.img。

本章節主要介紹從SD卡燒寫u-boot，kernel以及Android檔案系統，因爲此方法方便快捷，所以被作爲主要的燒寫方法推薦給USER。

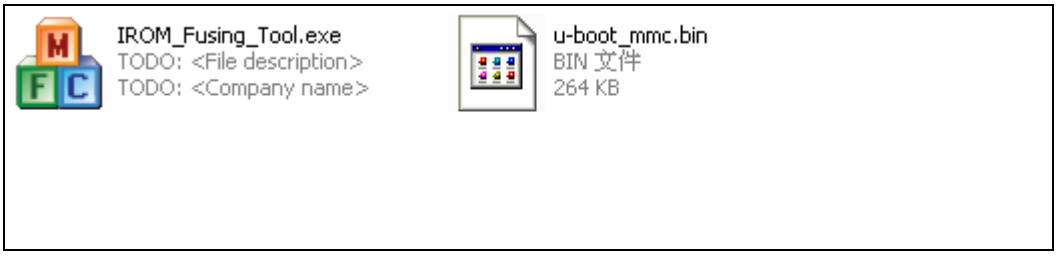
準備工作：5V電源，串列線一根，交叉網路線一根， TF卡（4GB以下）一張。

5-2 從 TF 卡燒寫檔案

使用 TF 卡來代替 JTAG 燒寫系統檔，可以節省大量的時間，從而可以大大的提高工作效率。

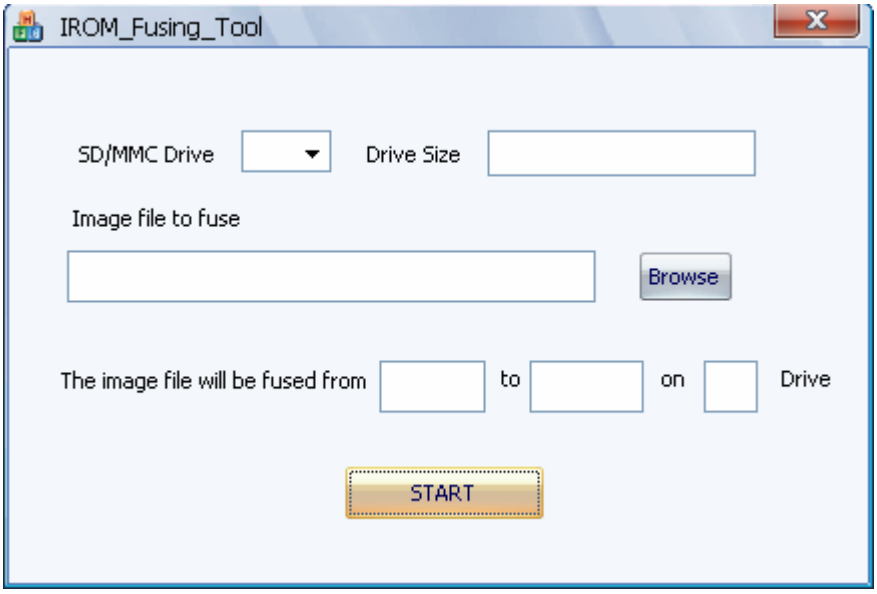
5-2.1 將 u-boot_mmc.bin 文件燒寫到 TF 卡

1、首先把 u-boot_mmc 目錄拷貝到 PC 主機上的任意目錄下，並確保裏面有 IROM_Fusing_Tool.exe 和 u-boot_mmc.bin 這兩個文件。如下圖所示：

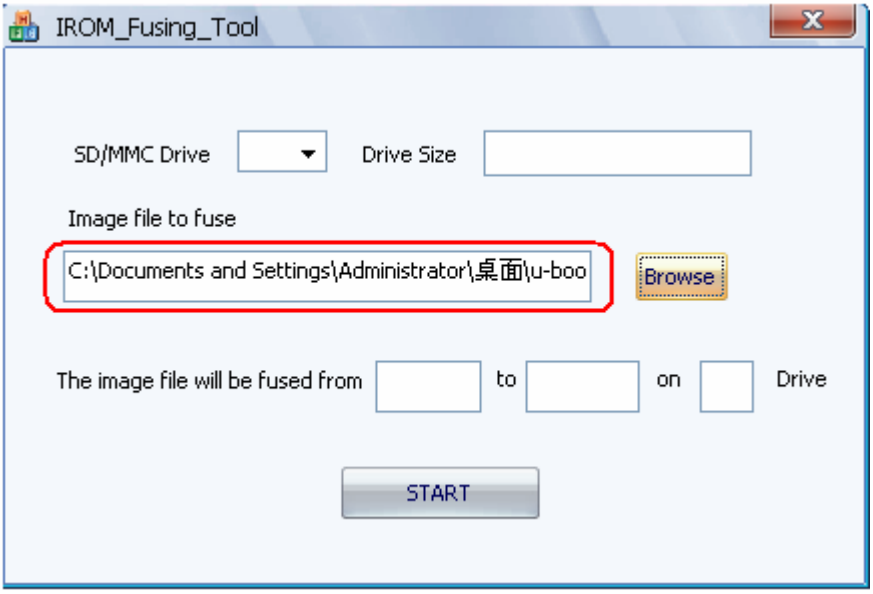


2、把 TF 卡連接到你的 PC 主機上，並且格式化爲 FAT32 格式。注意：最好選用 4GB 以下的 TF 卡。

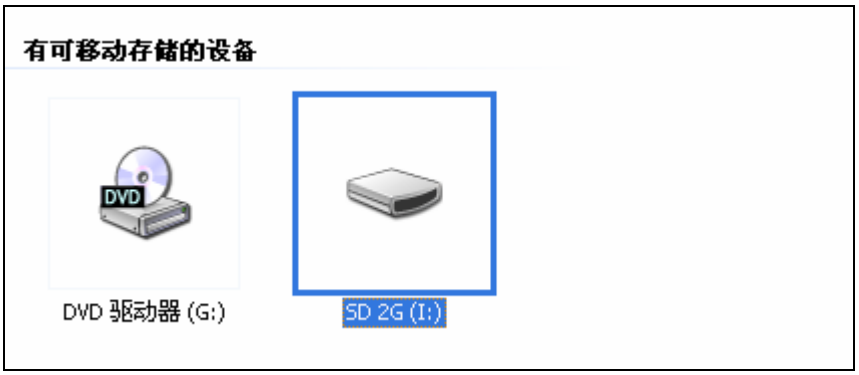
3、點兩下燒寫程式 IROM_Fusing_Tool.exe，進入燒寫介面。如下圖所示：

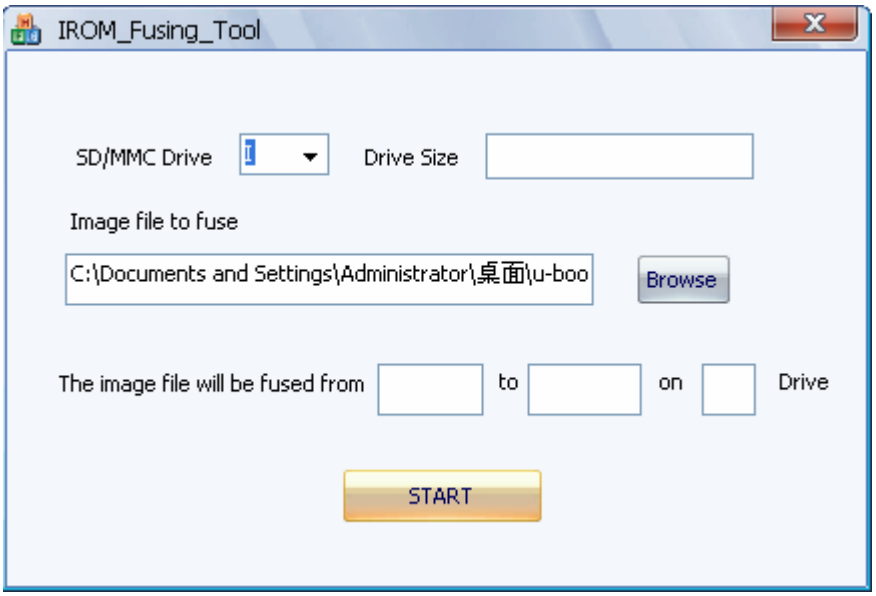


4、點選 Browse 按鈕，把 u-boot_mmc.bin 檔添加到該路徑下，如下圖所示：



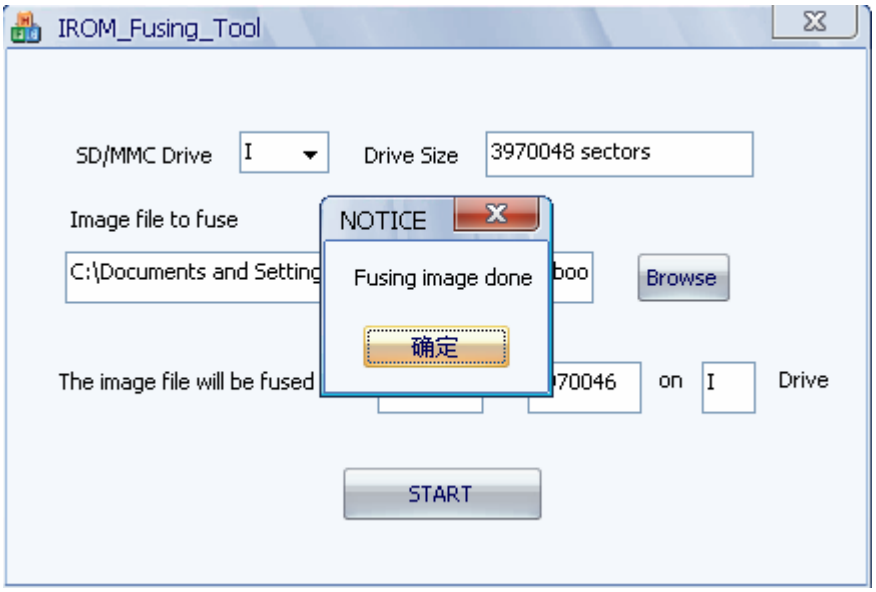
接著就選擇 SD/TF 卡所在的路徑，如我們的 SD/TF 卡是在 PC 主機上的可移動 I 磁碟目錄下，那麼我們就在 **SD/MMC Driver** 的路徑上選擇 I 磁碟。如下圖所示：





警告：在點選“START”按鈕前必須確認所選擇的 SD/MMC Drive 是真正的 SD 卡的磁碟，如設置成硬碟的話，將可能損壞硬碟上的資料！

5、在完成了以上操作之後，就可以點選 START 按鈕進行燒寫操作 ，把 u-boot_mmc.bin 文件燒寫到 TF 卡中。如果顯示 Fusing image done 按鈕圖示就說明燒寫成功：

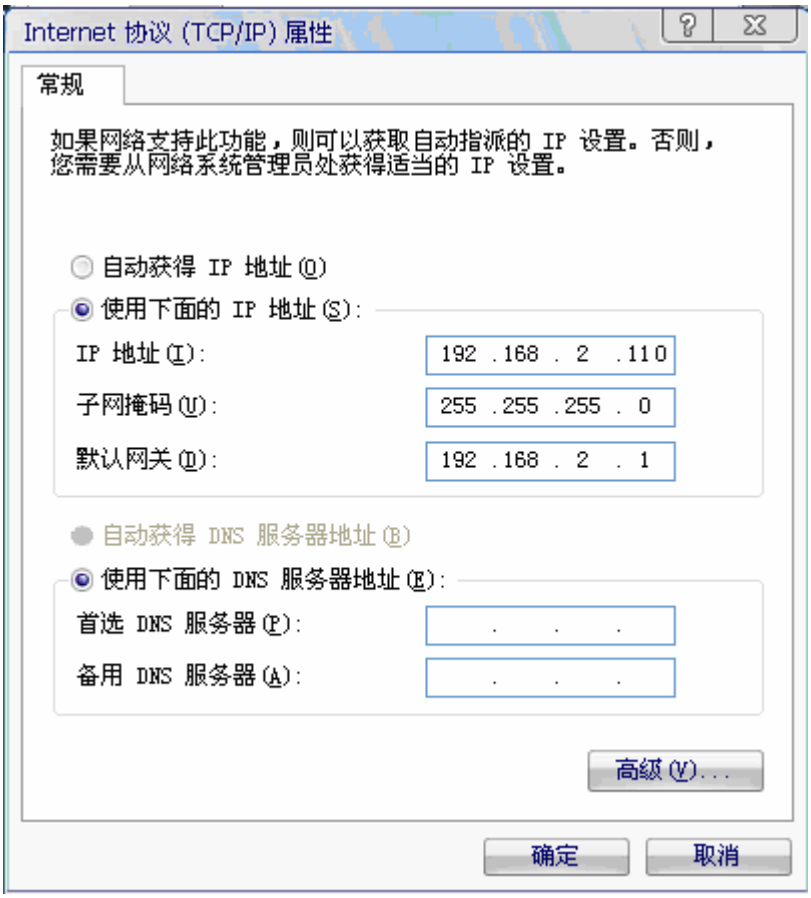


5-2.6　使用 TF 卡燒寫 u-boot、kernel 以及 Android 檔案

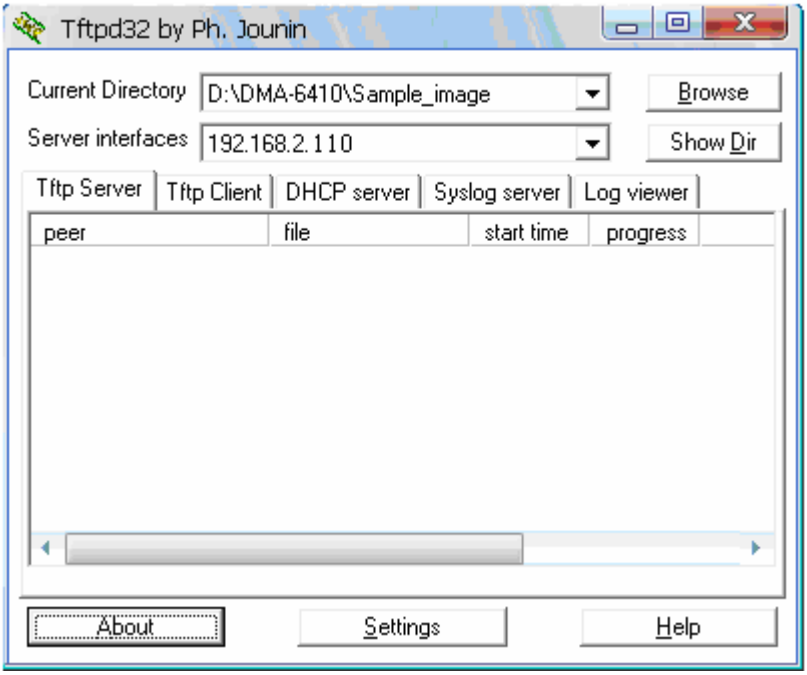
由於在 u-boot 環境下燒寫檔案檔要用到網路設備，所以在燒寫檔案檔之前，需要在 PC 端建立 TFTP 服務。如下：

在 PC 端建立 tftp 服務

（1） 由控制台修改區域聯機的 IP 位址。例如在下圖中，筆者將 PC 端的 IP 指定爲 192.168.2.110。

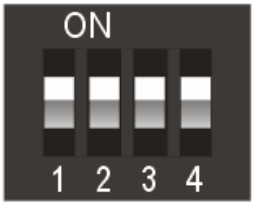


（2） 點兩下光碟內 tftpd32.exe，這個程式可以在 Microsoft Windows 系統上提供 TFTP 服務。啓動 Tftpd32.exe 後，它會將「Current Directory」下的目錄作爲 tftp 服務的根目錄，將 Sample_image 目錄作爲根目錄，如下圖所示。因爲在這個目錄內放置了稍後將要下載的 Linux Kernel (zImage)和 Android 根檔案系統(Android 由三個鏡像檔組成，分別是 ramdisk-uboot.img、system.img、userdata.img)。



（3） 使用交叉網路線連接 PC 和開發平台。請注意，開發平台只提供一個網路介面。

（4） 設置 TF 卡啓動跳線模式，跳線腳 J701 設置爲：1111→



（5） 啓動開發平台，用交叉網路線把主機和目標板（CON802）連接起來，在 u-boot 命令行裏輸入命令 pri，出現下圖內容：

```
SMDK6410 # pri
bootargs=console=ttySAC0,115200
bootcmd=nand read 50008000 600000 300000;nand read 50800000 900000 100000;bootm 50
008000 50800000
bootdelay=3
baudrate=115200
ethaddr=00:40:5c:26:0a:5b
ipaddr=192.168.2.144
serverip=192.168.2.110
gatewayip=192.168.2.1
netmask=255.255.255.0
u-boot=tftp 0x50008000 u-boot.bin;nand erase 0x0 0x40000;nand write 0x50008000 0x0
 0x40000
kernel=tftp 0x50008000 zImage;nand erase 0x600000  0x300000;nand write 0x50008000
0x600000 0x300000
ramdisk_uboot=tftp 0x50008000 ramdisk-uboot.img;nand erase 0x900000 0x100000;nand
write 0x50008000 0x900000 0x100000
system=tftp 0x50008000 system.img;nand erase 0xa00000  0x5000000;nand  write.yaffs
 0x50008000 0xa00000 ${filesize}
userdata=tftp 0x50008000 userdata.img;nand erase 0x9000000 0x7000000;nand write.ya
ffs 0x50008000 0x9000000 ${filesize}
stdin=serial
stdout=serial
stderr=serial

Environment size: 856/16380 bytes
SMDK6410 #
```

上圖框內內容爲開發平台當前網路配置資訊

（6） 若要對開發平台和 PC 進行網路配置，輸入如下命令：

#setenv ipaddr 192.168.2.144 (目標板 ip 位址要與主機 ip 保持在同一網段)

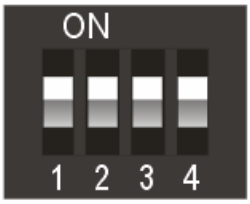
setenv serverip 192.168.2.110（與主機 ip 位址一致）

saveenv （保存設置資訊）

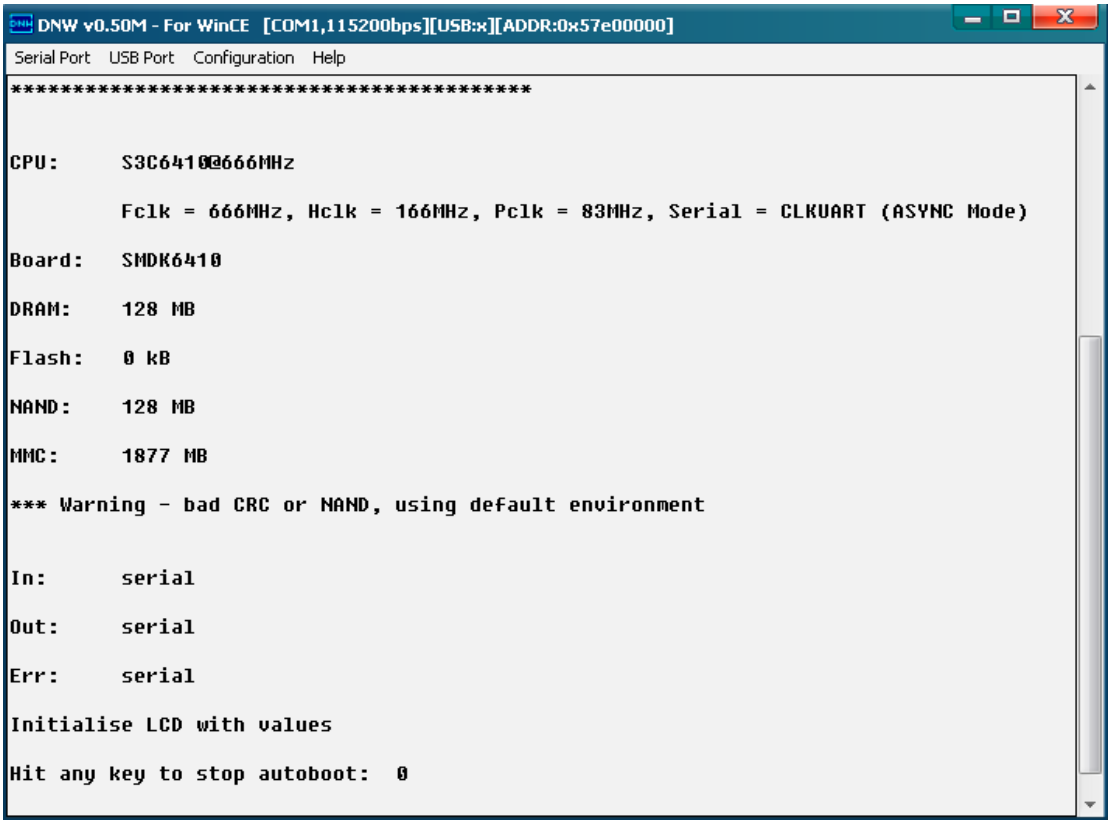
```
SMDK6410 # setenv ipaddr 192.168.2.144
SMDK6410 # setenv serverip 192.168.2.110
SMDK6410 # saveenv
Saving Environment to NAND...
Erasing Nand...Writing to Nand... done
SMDK6410 #
```

透過 tftp 協定燒寫檔案檔

（1） 以 HMI700-6410S 平台爲例，在確保 HMI700-6410S 平台電源已關閉情況下，按下圖進行跳線設置 J701，其設定方式爲將 OM4、OM3、OM2、OM1 設爲 1111(爲 1 時接上 ON)。如下：



（2） 正確跳線後，把已經準備好的 TF 卡插到 HMI700-6410S 平台的相應卡座內（TF 端），上電啓動系統，並透過串列埠除錯工具查看終端輸出資訊，如下圖顯示的是 1G 的 TF 卡啓動資訊。



（3） 在 u-boot 命令行輸入 pri 命令，查看燒寫命令，如下：

```
SMDK6410 # pri
bootargs=console=ttySAC0,115200
bootcmd=nand read 50008000 600000 300000;nand read 50800000 900000 100000;bootm 50
008000 50800000
bootdelay=3
baudrate=115200
ethaddr=00:40:5c:26:0a:5b
ipaddr=192.168.2.144
serverip=192.168.2.110
gatewayip=192.168.2.1
netmask=255.255.255.0
u-boot=tftp 0x50008000 u-boot.bin;nand erase 0x0 0x40000;nand write 0x50008000 0x0
 0x40000
kernel=tftp 0x50008000 zImage;nand erase 0x600000  0x300000;nand write 0x50008000
0x600000 0x300000
ramdisk_uboot=tftp 0x50008000 ramdisk-uboot.img;nand erase 0x900000 0x100000;nand
write 0x50008000 0x900000 0x100000
system=tftp 0x50008000 system.img;nand erase 0xa00000  0x5000000;nand  write.yaffs
 0x50008000 0xa00000 ${filesize}
userdata=tftp 0x50008000 userdata.img;nand erase 0x9000000 0x7000000;nand write.ya
ffs 0x50008000 0x9000000 ${filesize}
stdin=serial
stdout=serial
stderr=serial

Environment size: 856/16380 bytes
SMDK6410 #
```

（4） 輸入命令：tftp 0x50008000 u-boot.bin;nand erase 0x0 0x40000;nand write 0x50008000 0x0 0x40000，燒寫 u-boot(也可以使用擴充命令 run u-boot)。

```
SMDK6410 # tftp 0x50008000 u-boot.bin;nand erase 0x0 0x40000;nand write 0x50008000 0x0 0x40000
dm9000 i/o: 0x30000300, id: 0x90000a46
MAC: 00:00:40:00:5c:00
operating at 100M full duplex mode
TFTP from server 192.168.2.110; our IP address is 192.168.2.144
Filename 'u-boot.bin'.
Load address: 0x50008000
Loading: #################################################
done
Bytes transferred = 212992 (34000 hex)

NAND erase: device 0 offset 0x0, size 0x40000
Erasing at 0x20000 -- 100% complete.
OK

NAND write: device 0 offset 0x0, size 0x40000
 262144 bytes written: OK
SMDK6410 #
```

（5） 輸入命令：tftp 0x50008000 zImage;nand erase 0x600000　0x300000; nand write 0x50008000 0x600000 0x300000，燒寫核心(也可以使用擴充命令 run kernel)。

```
SMDK6410 # tftp 0x50008000 zImage;nand erase 0x600000  0x300000;nand write 0x50008000 0x600000 0x300000
dm9000 i/o: 0x30000300, id: 0x90000a46
MAC: 00:00:40:00:5c:00
operating at 100M full duplex mode
TFTP from server 192.168.2.110; our IP address is 192.168.2.144
Filename 'zImage'.
Load address: 0x50008000
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ########################################
done
Bytes transferred = 2530008 (269ad8 hex)

NAND erase: device 0 offset 0x600000, size 0x300000
Erasing at 0x8e0000 -- 100% complete.
OK

NAND write: device 0 offset 0x600000, size 0x300000
 3145728 bytes written: OK
SMDK6410 #
```

（6） 輸入命令：tftp 0x50008000 ramdisk-uboot.img;nand erase 0x900000 0x100000;nand write 0x50008000 0x900000 0x100000，燒寫 ramdisk_uboot.img 檔，(可以使用擴充命令 run ramdisk_uboot)。

```
SMDK6410 # tftp 0x50008000 ramdisk-uboot.img;nand erase 0x900000 0x100000;nand write 0x50008000 0x900000 0x100000
dm9000 i/o: 0x30000300, id: 0x90000a46
MAC: 00:00:40:00:5c:00
operating at 100M full duplex mode
TFTP from server 192.168.2.110; our IP address is 192.168.2.144
Filename 'ramdisk-uboot.img'.
Load address: 0x50008000
Loading: ################################
done
Bytes transferred = 159676 (26fbc hex)

NAND erase: device 0 offset 0x900000, size 0x100000
Erasing at 0x9e0000 -- 100% complete.
OK

NAND write: device 0 offset 0x900000, size 0x100000
 1048576 bytes written: OK
SMDK6410 #
```

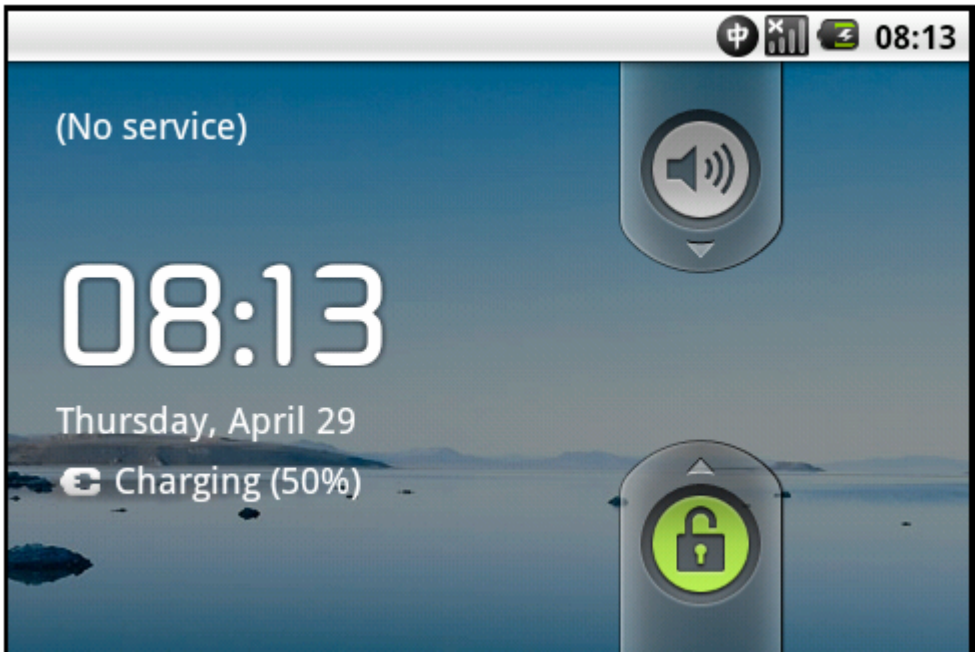
（7） 輸入命令：tftp 0x50008000 system.img;nand erase 0xa00000 0x5000000; nand write.yaffs 0x50008000 0xa00000 ${filesize}，燒寫 system.img 檔，也可以使用擴充命令 run system。

```
SMDK6410 # tftp 0x50008000 system.img;nand erase 0xa00000  0x5000000;nand  write.yaffs 0x50008000 0xa00000 ${filesize}
dm9000 i/o: 0x30000300, id: 0x90000a46
MAC: 00:00:40:00:5c:00
operating at 100M full duplex mode
TFTP from server 192.168.2.110; our IP address is 192.168.2.144
Filename 'system.img'.
Load address: 0x50008000
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
```

（8） 輸入命令：tftp 0x50008000 userdata.img;nand erase 0x9000000 0x7000000;nand write.yaffs 0x50008000 0x9000000 ${filesize}，燒寫 userdata.img 檔，也可以使用擴充命令 run userdata。

```
SMDK6410 # tftp 0x50008000 userdata.img;nand erase 0x9000000 0x7000000;nand write.yaffs 0x50008000 0x9000000 ${filesize}
dm9000 i/o: 0x30000300, id: 0x90000a46
MAC: 00:00:40:00:5c:00
operating at 100M full duplex mode
TFTP from server 192.168.2.110; our IP address is 192.168.2.144
Filename 'userdata.img'.
Load address: 0x50008000
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ###################################
done
Bytes transferred = 2179584 (214200 hex)

NAND erase: device 0 offset 0x9000000, size 0x7000000
Erasing at 0xffe0000 -- 100% complete.
OK

NAND write: device 0 offset 0x9000000, size 0x214200

Writing data at 0x9203800 -- 100% complete.
 2179584 bytes written: OK
SMDK6410 #
```

（9） 使用 TF 卡完成燒寫之後，可以直接按重啓鍵從 TF 卡啓動，也可以關閉電源，拔下 TF 卡，從 Nand Flash 跳線啓動（J701 跳線腳設置如下圖）。以上兩種方法都可以成功啓動 Android 系統。



第六章
Android 2.1 系統程式及 AP 測試

Android 系統本身自帶了很多應用程式，如鬧鐘、Music、Gallery 等，讀者也可以在 Android 系統上開發一些有用的測試程式。下面將講解 Android 2.1 系統中自帶的一些應用以及開發出來的一些應用測試程式。

6-1　Android 系統應用程式測試

6-1.1　APK 安裝器

APK 安裝器在 Android 系統中起到一個橋樑的作用，透過它可以把外部應用程式植入到 Android 系統內部，大大方便了對應用程式的除錯，所以 APK 安裝器在 Android 系統中具有舉足輕重的作用。下面就介紹一下如何使用 APK 安裝器來安裝應用程式。

點選桌面圖示 

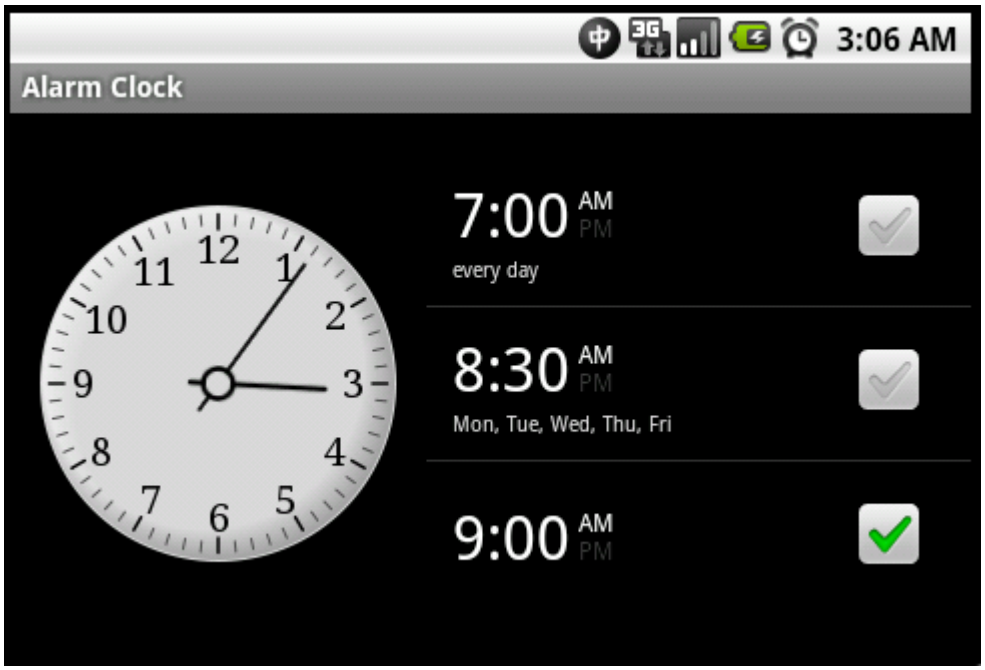
進入如下圖所示介面：

選中所需要安裝的 apk 檔進行安裝即可。

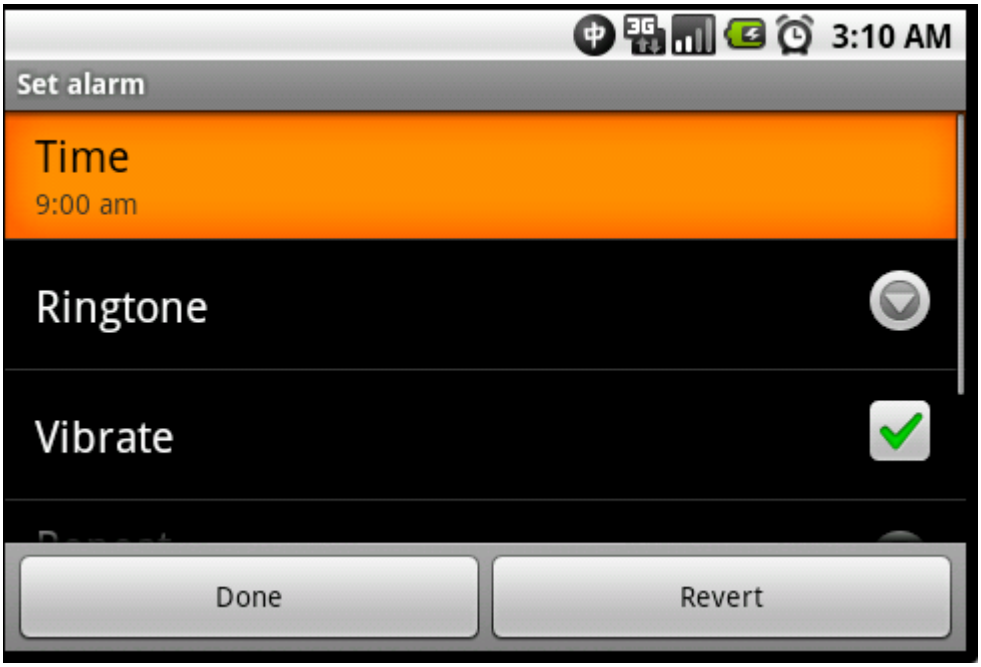
6-1.2　時鐘

點選桌面 

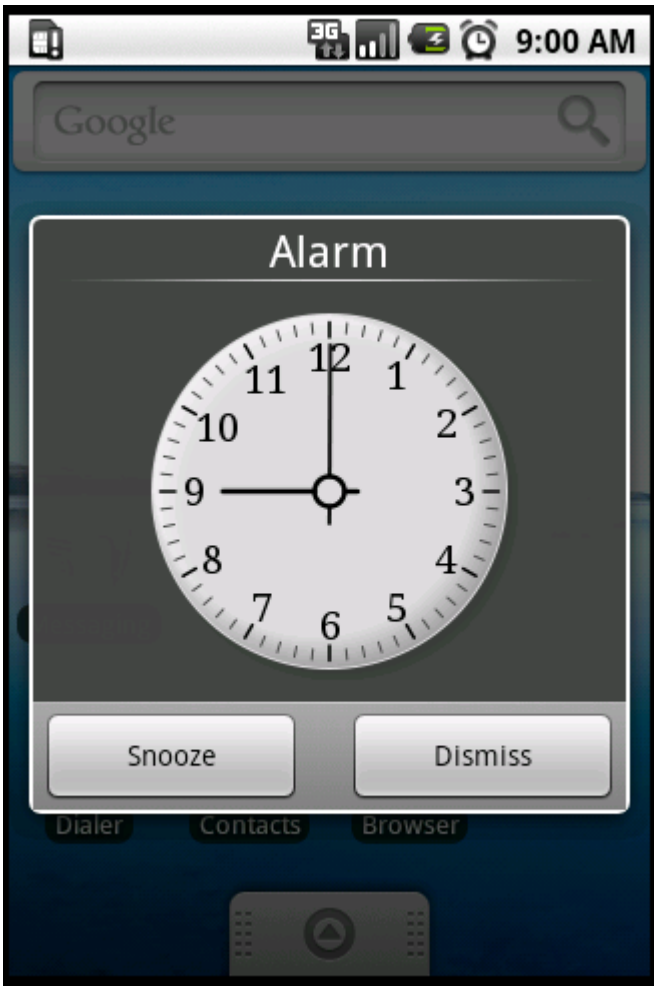
圖示，進入下圖所示介面：



可以設置三個鬧鐘，他們的時間可以不一樣。當右邊方框內的對鉤變成這種狀態  後表標設置鬧鐘成功。點選時間選項，將進入如下圖所示設置介面：



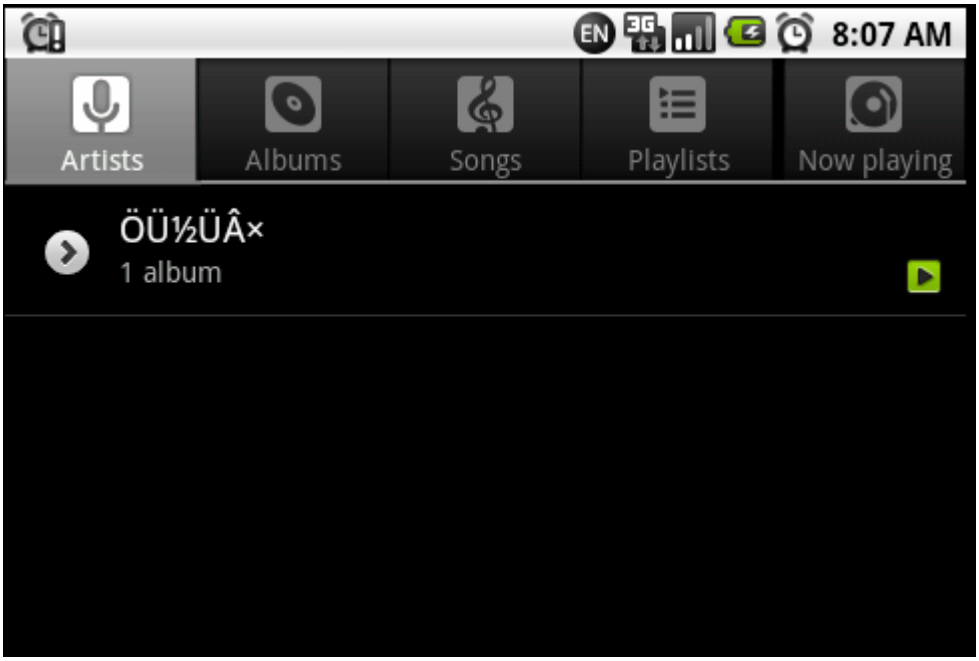
可以設置時間、鈴聲以及響鈴在一周的哪幾天響鈴等這些設置，根據不同需求進行不同的設置。當設置完成後返回即可。當到了所設定的鬧鐘時間時，Android 系統介面會出現如下圖所示介面：


9:00 AM
Google
Alarm
12
1
2
3
4
5
6
7
8
9
10
11
Snooze
Dismiss
Dialer
Contacts
Browser

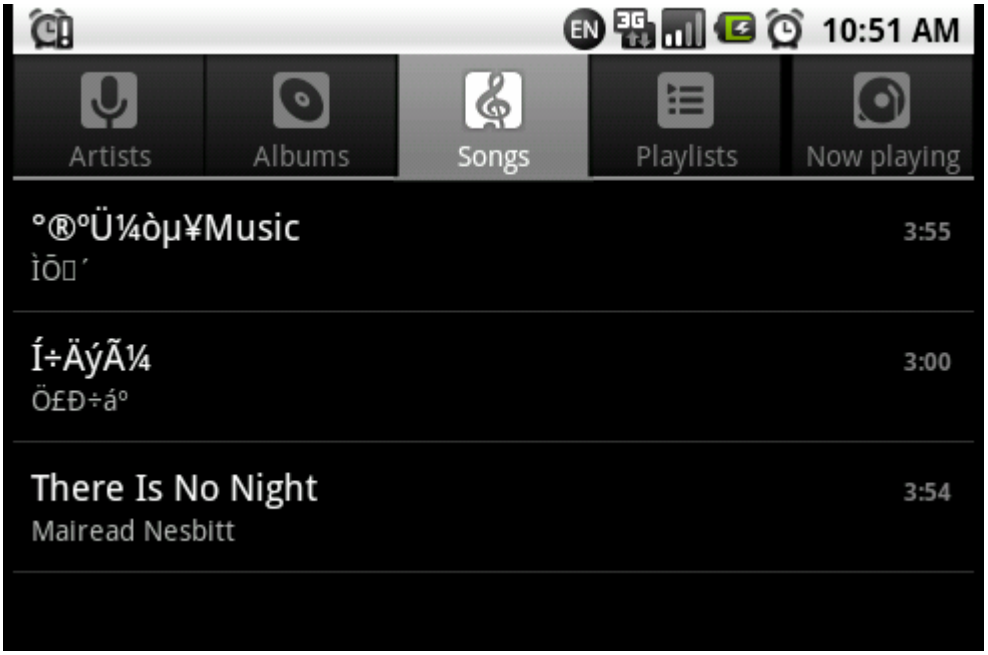
6-1.3 Music 音效測試

(1) 將 SD 卡插入卡槽內，點選螢幕上的 Music 圖示

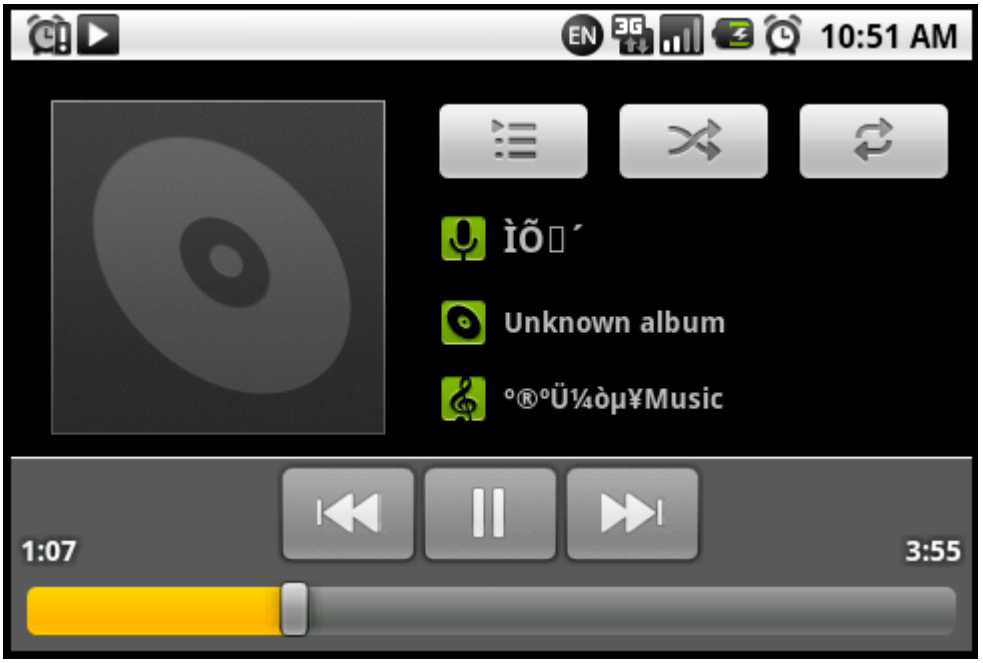
進入下圖所示介面：



(2) 點選 Songs 按鈕，系統將 TF 卡中的音效檔自動讀到當前目錄下，如下圖所示：

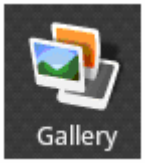


(3) 點選要播放的音效檔案名稱，進入播放介面，如下圖所示：

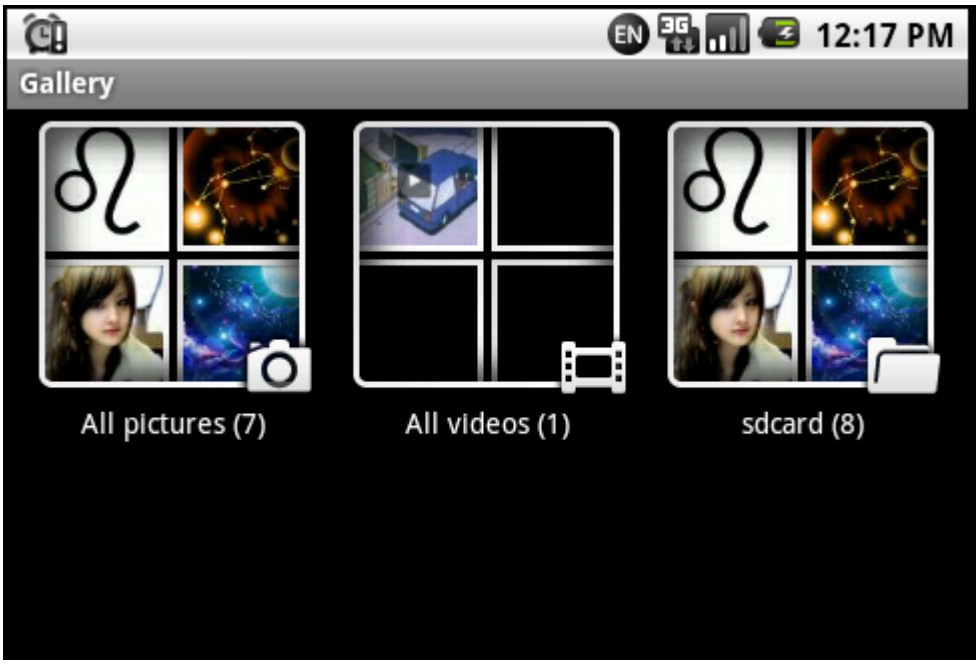


此時插入耳機即可聽到音樂。

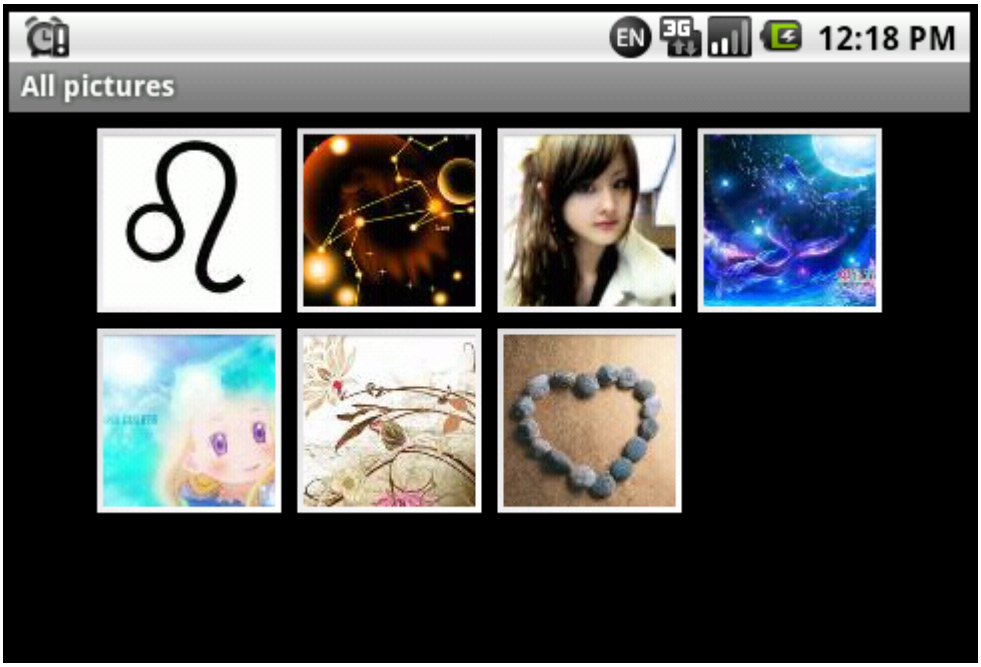
6-1.4　Gallery

點選桌面圖示 

，將進入下圖所示介面：



共有三個檔案夾，其中 All pictures 中顯示的是 TF 卡中所有的圖片，All videos 中顯示的是 TF 卡中所有的視訊檔，sdcard 中顯示的是 TF 卡中的所有的圖片和視訊檔。點選 All videos 可以播放視訊檔，點選 All pictures，進入下圖所示介面：



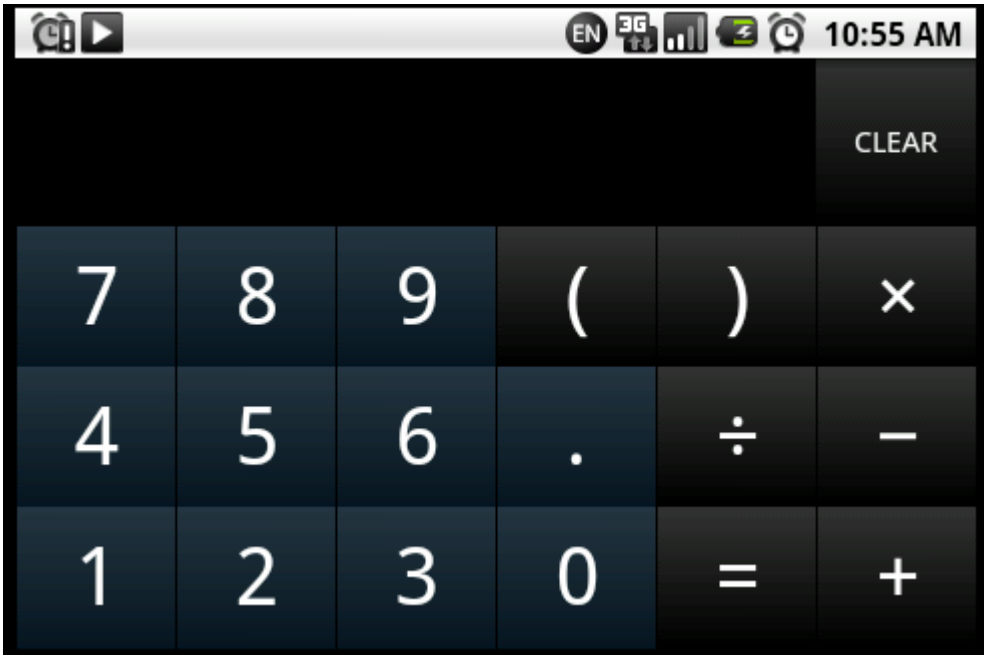
點選網格中的小圖片就可以開始瀏覽圖片，並可以進行一些設置，如下圖所示：

點選 Set as 可以將此圖片設置爲壁紙，點選 Delete 可以刪除此圖片。

6-1.5　Calculator

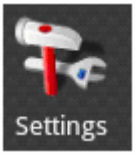
點選桌面 

圖示，進入如下圖所示介面：

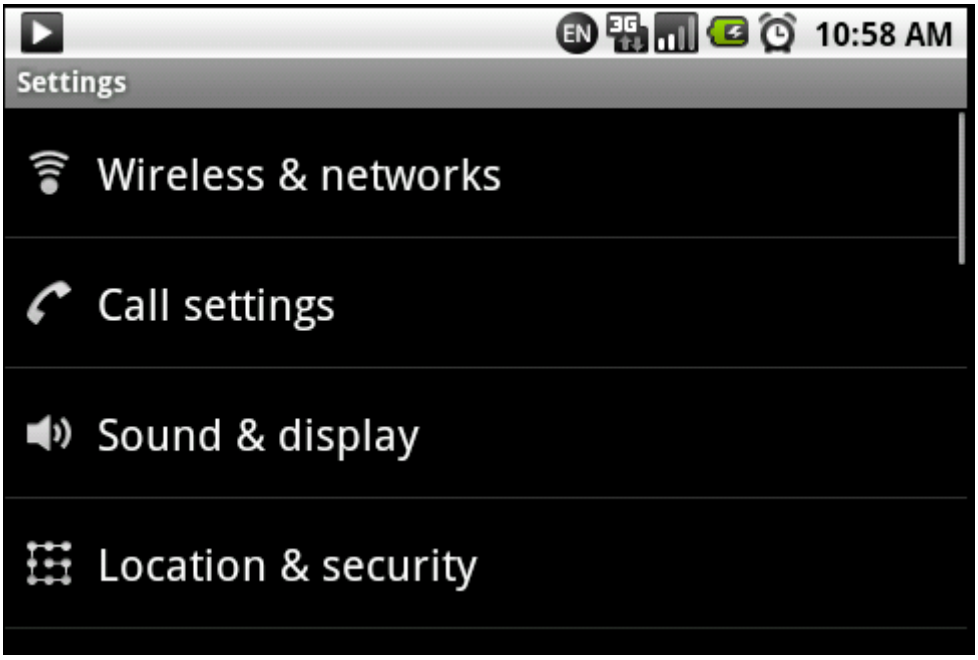


可進行簡單的計算。

6-1.6 Setting

(1) 點選桌面 

圖示，進入如下圖所示介面：



(2) 點選 Sound&display 選項，可以對系統的鈴聲音量以及音樂音量進行設置，可以設置背光以及螢幕的休眠時間等。點選 Applications 選項，可以卸載掉安裝在 Android 中的應用程式，點選 Date&time 選項時可以對系統的日期時間等進行設置。Setting 應用程式主要是對 Android 系統進行一些相關的設置，讀者如有興趣，可以獨自研究。

6-2　TF 卡自動掛載測試

(1) 進入 Android 命令終端，輸入命令：ls，顯示資訊如下：

```
# ls
sqlite_stmt_journals
config
cache
sdcard
d
etc
system
sys
sbin
proc
init.rc
init.goldfish.rc
init
default.prop
data
root
dev
#
```

然後輸入命令：ls sdcard/，查看 sdcard/目錄下的目錄和檔案。顯示資訊如下：

```
#  ls sdcard/
#
```

此時 sdcard 爲一空目錄。

(2) 將TF 卡插入卡槽(HSMMC)，系統將列印下圖所所示資訊：

```
# [  283.678224] mmc1: new SD card at address cefe
[  283.690338] mmcblk0: mmc1:cefe SD02G 1.89 GiB
[  283.692430]  mmcblk0: p1
[  285.477829] select 2000 (droid.bluetooth), adj 14, size 3765, to kill
[  285.478713] select 2042 (roid.alarmclock), adj 14, size 3814, to kill
[  285.485244] send sigkill to 2042 (roid.alarmclock), adj 14, size 3814
```

然後輸入命令：ls sdcard/。顯示如下資訊：

```
# ls sdcard
userdata.img
zImage
ramdisk-uboot.img
system.img
LOST.DIR
Android
鍋交噹鏽?mp3
123.mp4
test.3gp
RECYCLER
DCIM
p_large_zhsF_20f2000393a22d0b.jpg
200804201739410s.jpg
20080420174011169.jpg
untitled.bmp
p_large_fvgm_025c0000710c2d13.jpg
ex07_11_73240.amr
RICHAR~1.MP3
albumthumbs
澶忢奖.mp3
RecordTest.apk
paipai_dgxb_audio_2006_8_5_12_19_03.mp3
ImageFlipper.apk
DMA6410_TEST.apk
Music.apk
003544603.gif
#
```

出現如上資訊，說明 TF 卡自動掛載成功。系統會自動讀取 TF 卡中的檔，如音效檔等。

(3) 由於 Android 系統支援 TF 卡的熱插拔，所以我們可以隨時插拔 TF 卡。

6-3　DM9000 網路測試

(1) 用交叉網路線將平台網路埠 CON201 和 PC 主機網路埠連接起來，開啓平台電源，進入 Android 命令終端後，輸入命令：ifconfig eth0 up。啓動 DM9000 的網路埠。

(2) 設置 DM9000 的網路埠的 IP：ifconfig eth0 192.168.2.11(保證此 IP 位址與伺服器的位址在同一網段)。

(3) 用 ping 命令 ping 伺服器埠，看是否可以 Ping 通。具體過程如下圖：

```
# ifconfig eth0 up
# ifconfig eth0 192.168.2.11
# ping 192.168.2.110
PING 192.168.2.110 (192.168.2.110) 56(84) bytes of data.
64 bytes from 192.168.2.110: icmp_seq=1 ttl=64 time=1.75 ms
64 bytes from 192.168.2.110: icmp_seq=2 ttl=64 time=0.364 ms
64 bytes from 192.168.2.110: icmp_seq=3 ttl=64 time=0.358 ms
64 bytes from 192.168.2.110: icmp_seq=4 ttl=64 time=0.363 ms
64 bytes from 192.168.2.110: icmp_seq=5 ttl=64 time=0.366 ms
64 bytes from 192.168.2.110: icmp_seq=6 ttl=64 time=0.357 ms
64 bytes from 192.168.2.110: icmp_seq=7 ttl=64 time=0.372 ms
64 bytes from 192.168.2.110: icmp_seq=8 ttl=64 time=0.355 ms
64 bytes from 192.168.2.110: icmp_seq=9 ttl=64 time=0.355 ms
^C
--- 192.168.2.110 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8014ms
rtt min/avg/max/mdev = 0.355/0.516/1.756/0.438 ms
#
```

如上圖的測試結果，就表明 HMI700-6410S 平台的 DM9000 的網端是好的，DM9000 網端可以使用。按 Ctrl+C 鍵即可退出測試程式。

6-4 DM9000 網路瀏覽測試

開啓HMI700-6410S平台，並接上網路線，然後在串列埠終端輸入如下命令：

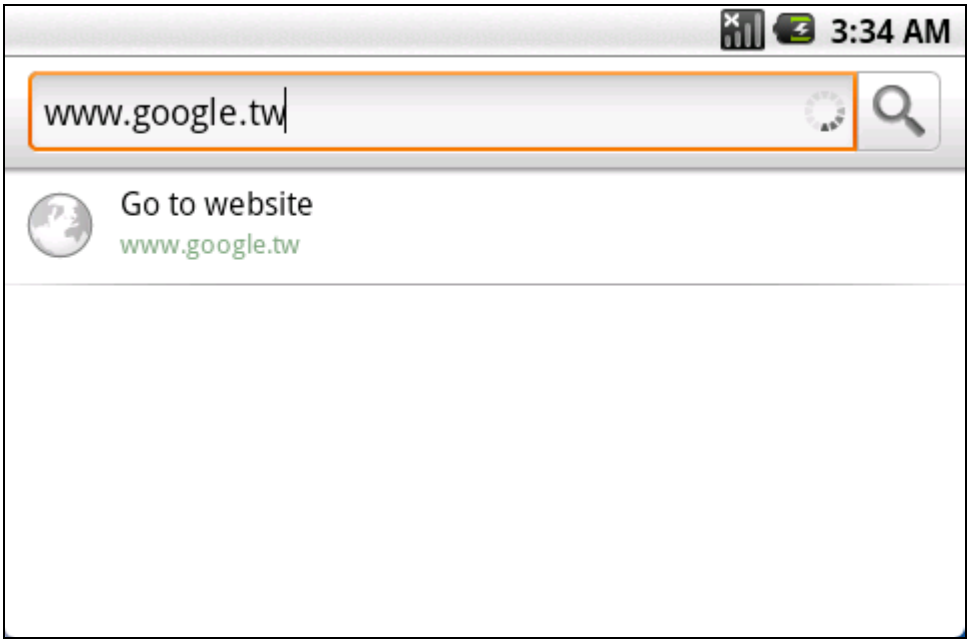
ifconfig eth0 up

netcfg eth0 dhcp

setprop net.dns0 2020.96.128.166//設置功能變數名稱伺服器，大陸爲86，台灣爲166

setprop net.dns1 202.96.134.133//設置功能變數名稱伺服器

在確認以上命令輸入正確後，直接可以在 google 搜索引擎內輸入網路位址，進行網路瀏覽。如下：



註：實現網路瀏覽的前提就是，你所使用的網路線是被允許網路瀏覽，否則，系統會找不到相應網路位址。

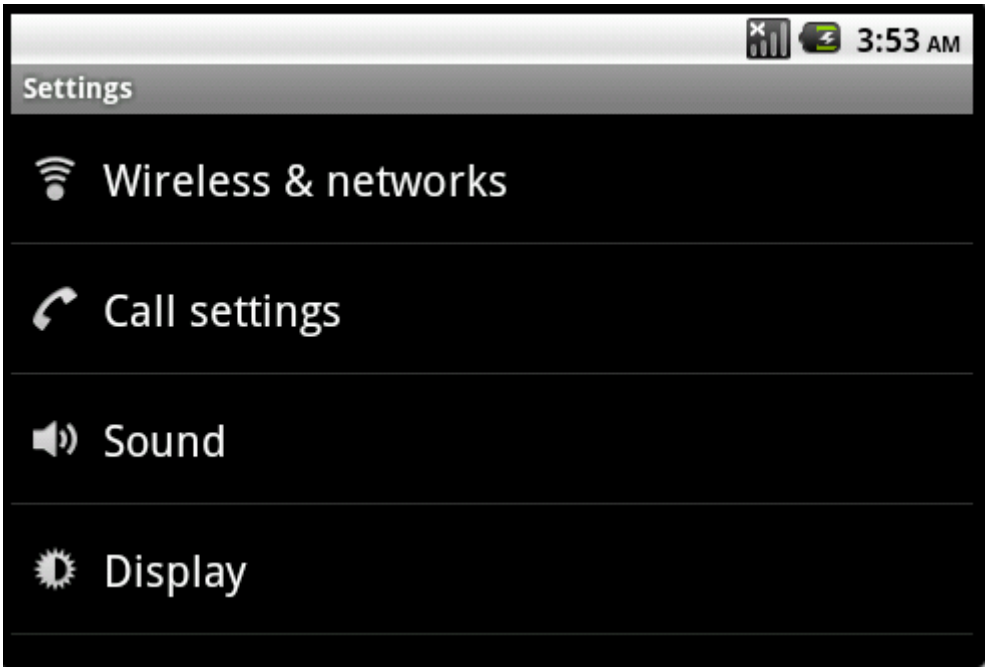
6-5　USB WIFI 無線上網瀏覽測試

爲了確保能夠實現 USB WIFI 無線網路瀏覽，在測試前必須開通相應的無線網路。然後再進行測試，操作如下：

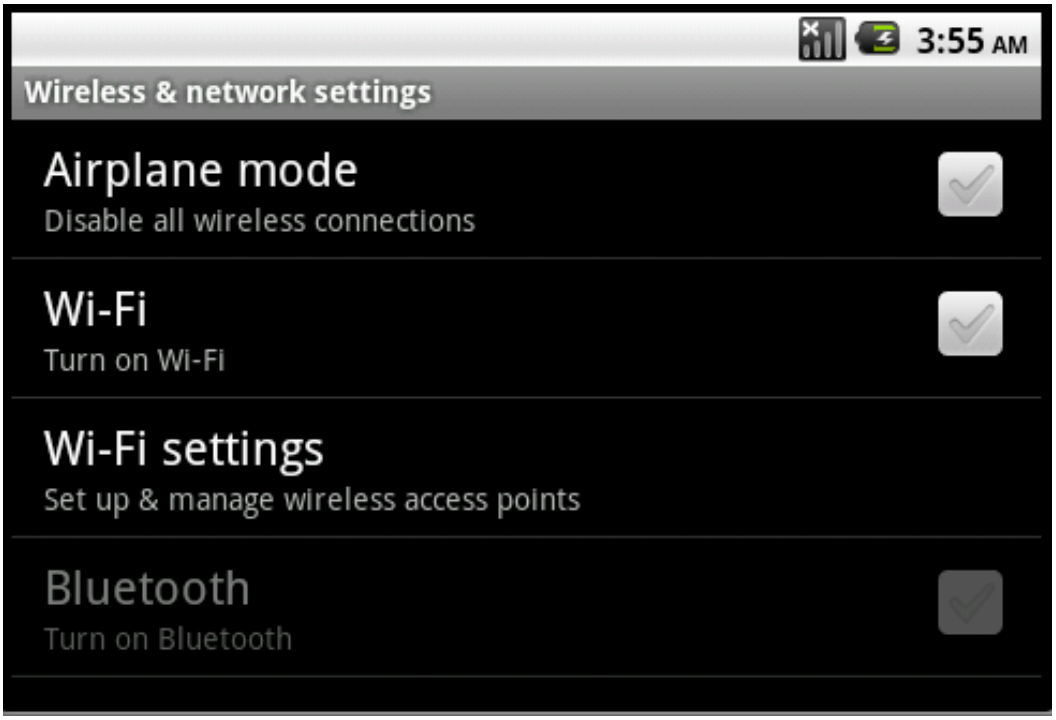
開啓 HMI700-6410S 開發平台，並在 USB HOST 埠接上 USB WIFI 模組（這裏我們使用的是 RT73 模組，同時還支援 RT3070 模組），然後在 Android 系統的 Setting 介面進行 WIFI 連接，如下圖所示：

1. 點擊“Settings”圖示 

進入到“Settings”介面，如圖：

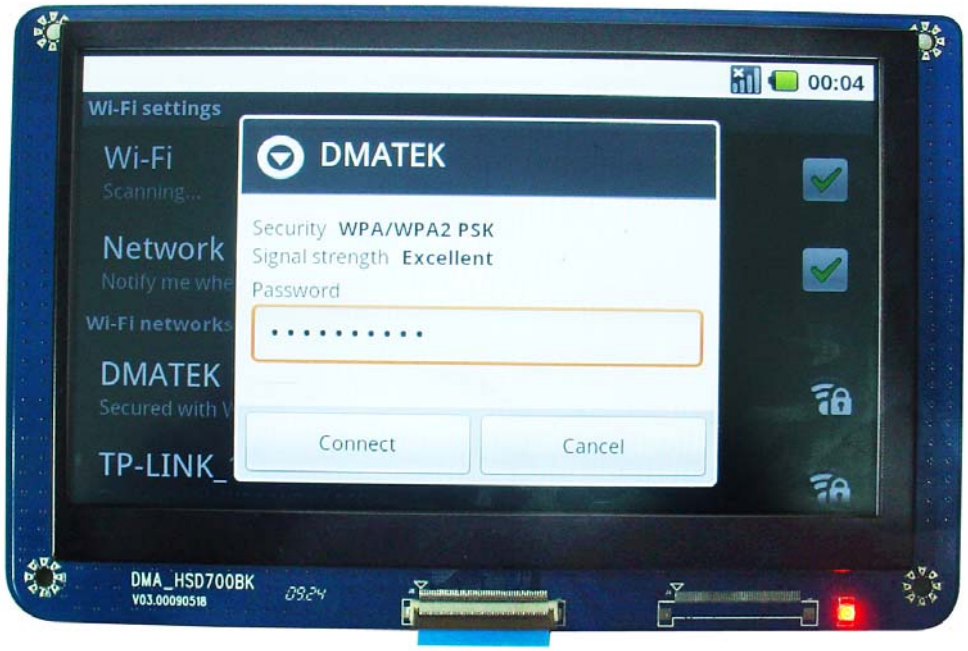


2. 點擊 Wireless&networks 項，我們會看到 WIFI 的設置介面，如圖：



3. 點擊 Wi-Fi 項連接 WIFI 或點擊 Wi-Fi settings 項進入到 Wi-Fi 設置介面連接 WIFI 都可以。下面是我們進入到 Wi-Fi settings 介面後連接顯示的圖示：

4. 在上圖中我們可以看到，WIFI 已經掃描到幾個無線路由設備，這裏我們選擇連接 DMATEK 這個網端。輸入密碼的圖示如下：

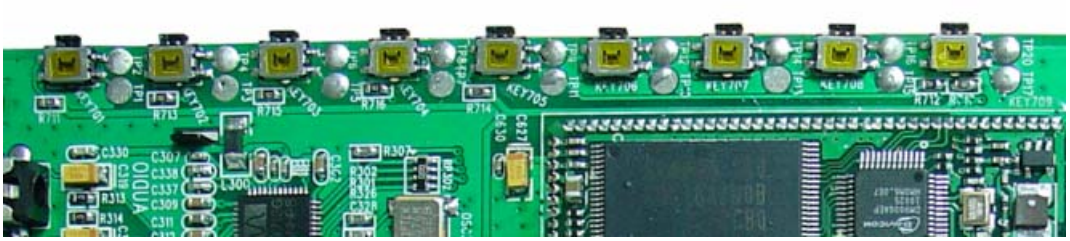


5. 輸入密碼後點擊“Connect”進行連接，數秒之後我們就會看到系統連接上網路的顯示，如下圖所示：

6. 這是我們可以打開 Google 自帶的搜索引擎看一下是否能夠流覽網頁，如下圖所示：

上圖顯示，WIFI 連接網路成功。

6-6 HMI700-6410S 按鍵佈局



按鍵	KEY1	KEY2	KEY3	KEY4	KEY5	KEY6	KEY7	KEY8	KEY9
功能	音量+	音量-	上	下	選項	返回	睡眠	確定	HOME

第七章
NDK 環境架構與編譯

Android平台的第三方應用程式均是依靠於Java的Dalvik特質虛擬機進行開發的，原生NDK可以讓開發者更加直接的接觸Android系統資源，並使用傳統的C或C++語言編寫程式，並在程式封包檔(.apks)中直接嵌入原生庫檔，讓java開發者方便的調用底層操作。NDK提供了一系列的工具，NDK集成了交叉編譯器，幫助開發者快速開發C或C++的動態庫，並能將so和java應用一起打包成apk。下面將講述編譯環境的搭建以及用NDK編譯so的整個過程。

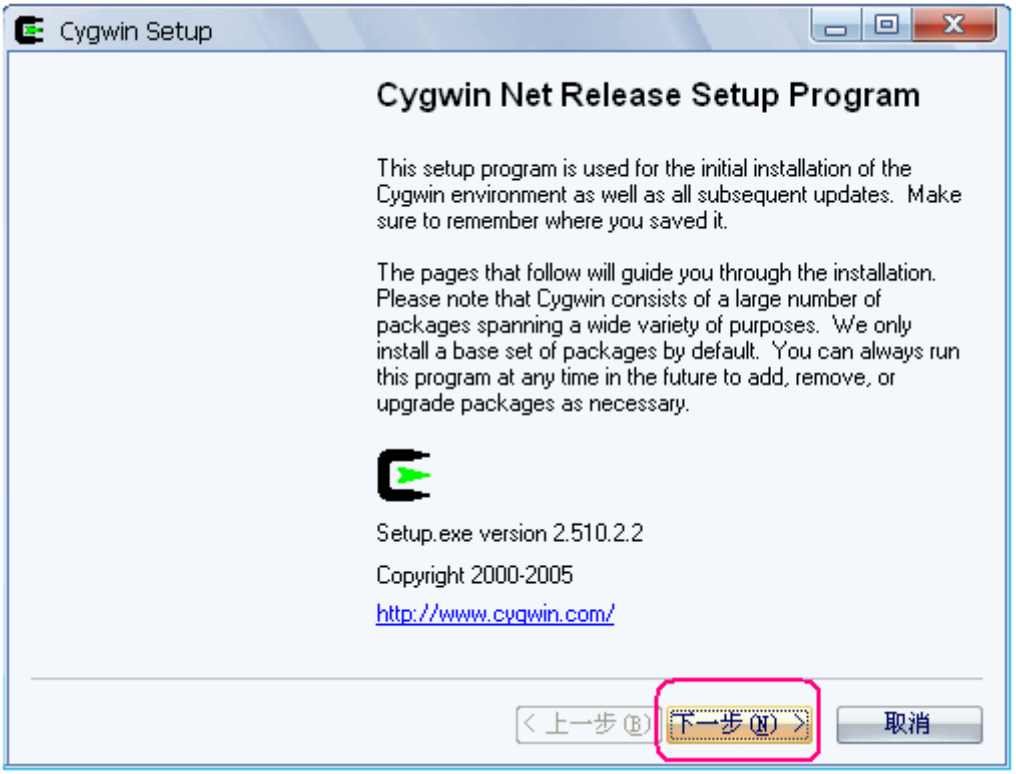
7-1 Windows 平台下的環境搭建与編譯

7-1.1 Cygwin 的安裝

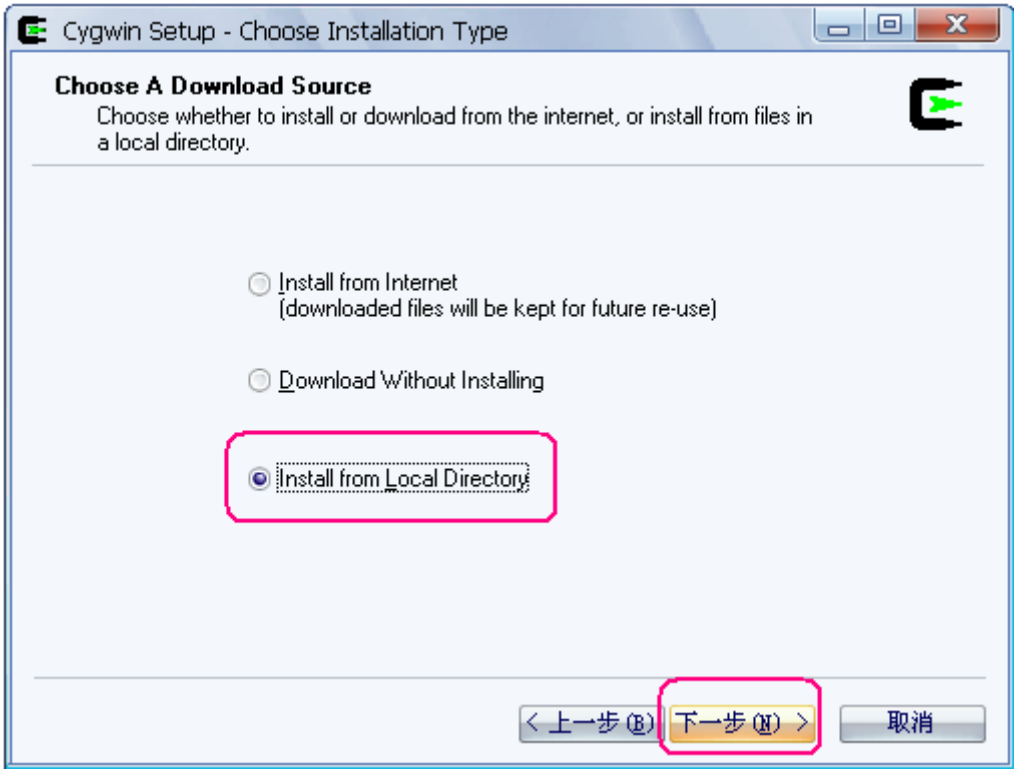
（1） NDK 編譯需要用到 Cygwin 中的 make 和 gcc，所以得先下載並安裝 Cygwin。

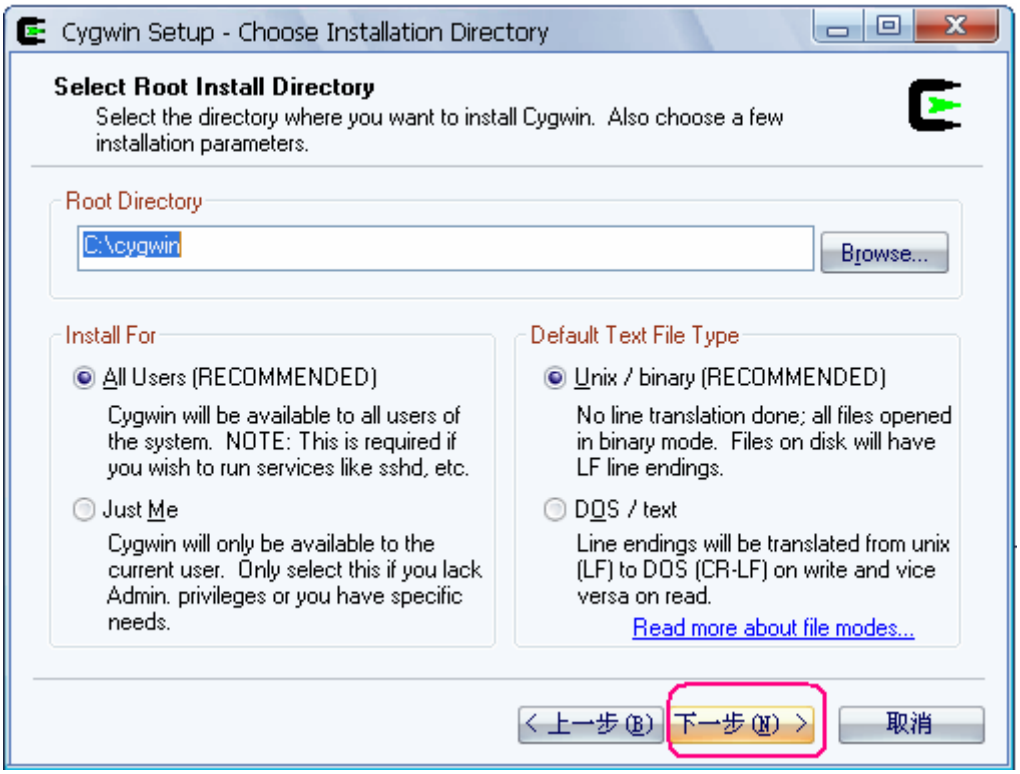
（2） 下載完成後，點選 

圖示，進入安裝介面。如下圖所示：

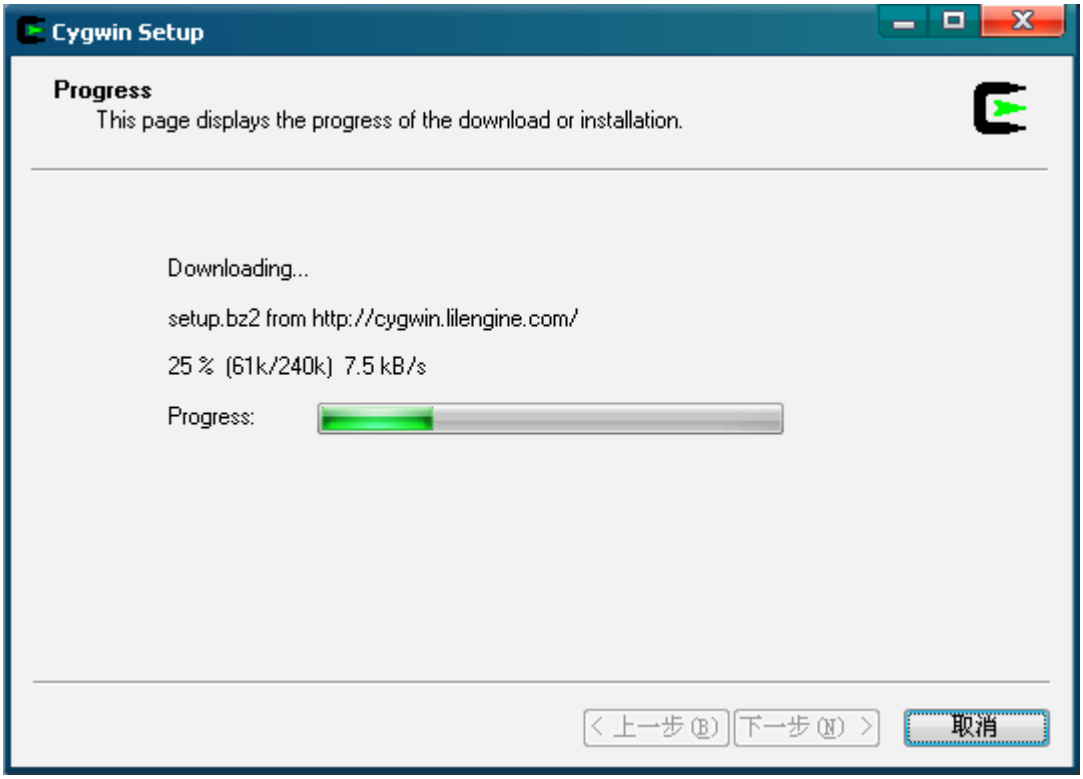


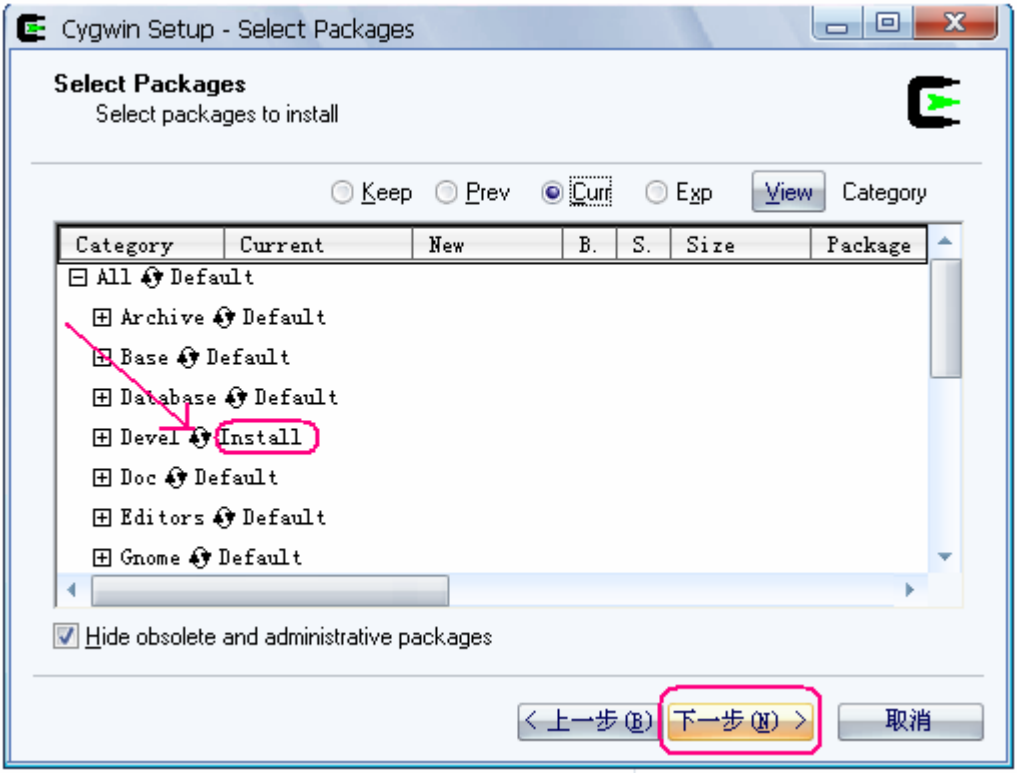
（3）　下面以圖解的方式來說明安裝步驟。



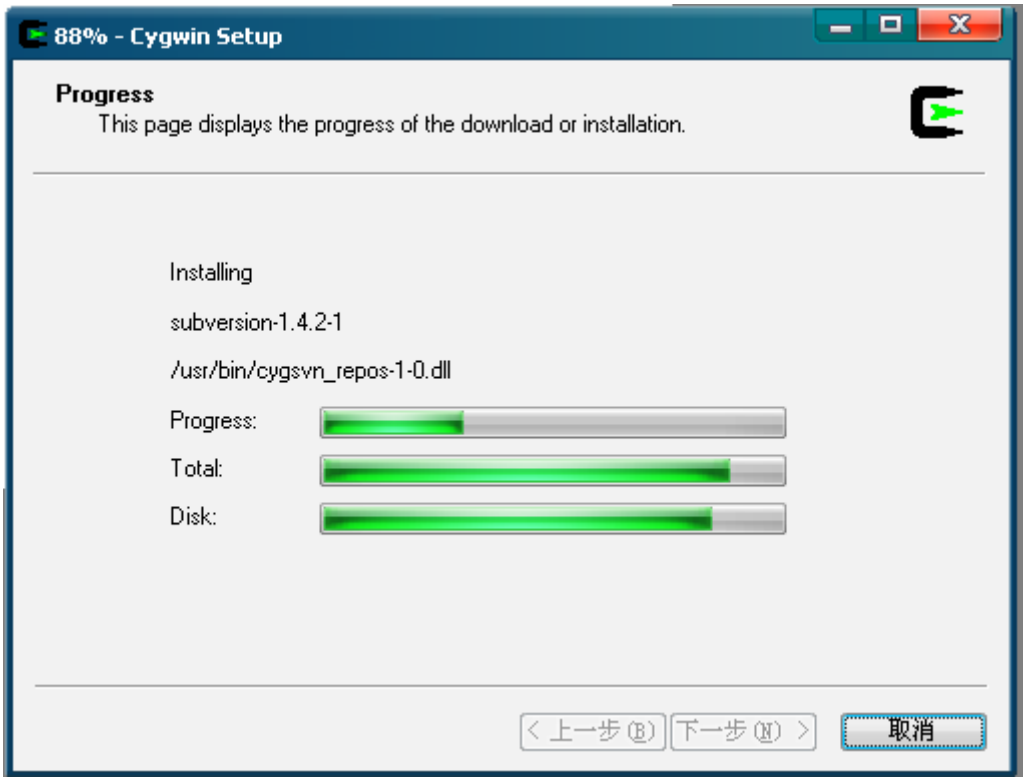

Cygwin Setup - Choose Installation Directory
Select Root Install Directory
Select the directory where you want to install Cygwin. Also choose a few installation parameters.
Root Directory
C:\cygwin
Browse...
Install For
All Users (RECOMMENDED)
Cygwin will be available to all users of the system. NOTE: This is required if you wish to run services like sshd, etc.
Just Me
Cygwin will only be available to the current user. Only select this if you lack Admin. privileges or you have specific needs.
Default Text File Type
Unix / binary (RECOMMENDED)
No line translation done; all files opened in binary mode. Files on disk will have LF line endings.
DOS / text
Line endings will be translated from unix (LF) to DOS (CR-LF) on write and vice versa on read.
Read more about file modes...
< 上一步 (B)
下一步 (N) >
取消


Cygwin Setup - Select Local Package Directory
Select Local Package Directory
Select a directory where you want Setup to store the installation files it downloads. The directory will be created if it does not already exist.
Local Package Directory
ndroid\LocalCygwinInstaller23.7MB\cygwin-instal
Browse...
< 上一步 (B)
下一步 (N) >
取消

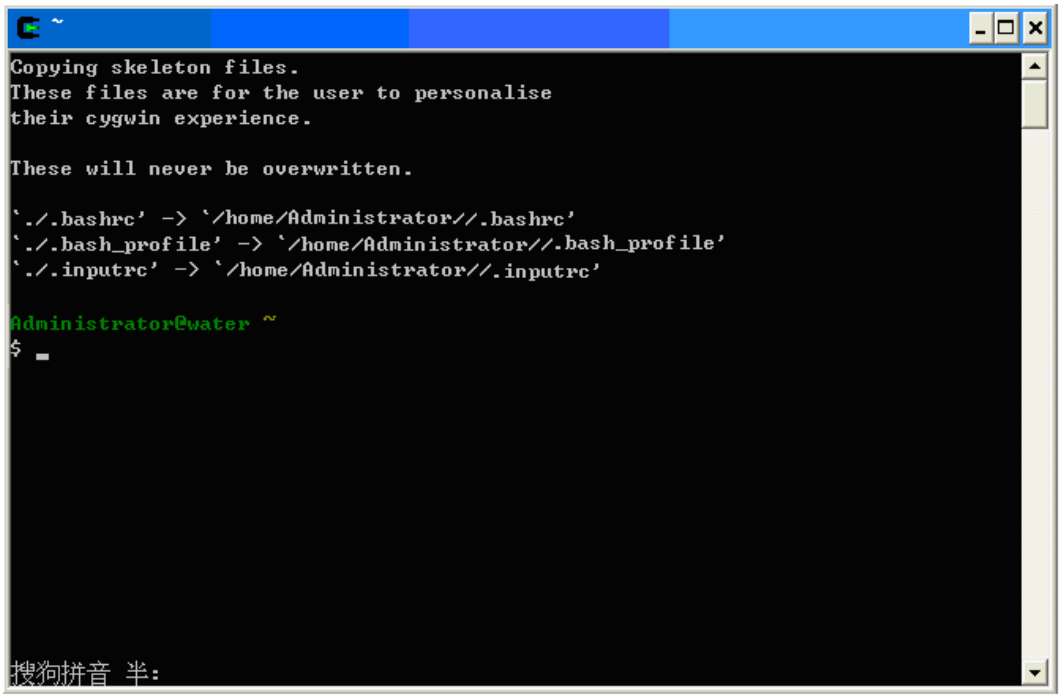

Cygwin Setup
Progress
This page displays the progress of the download or installation.
Downloading...
setup.bz2 from http://cygwin.lilengine.com/
25 % (61k/240k) 7.5 kB/s
Progress:
< 上一步 (B)
下一步 (N) >
取消



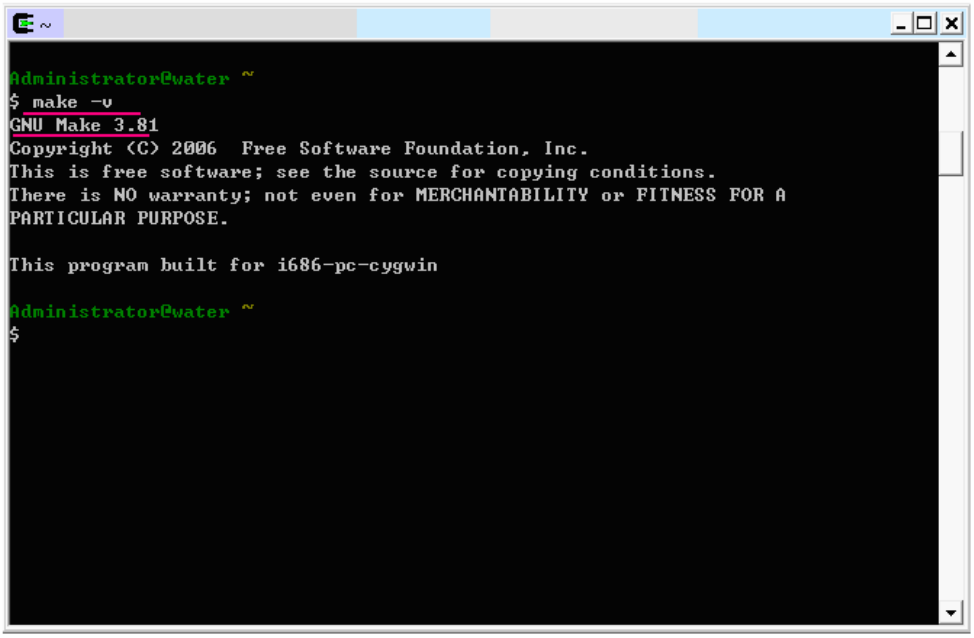
注意：NDK 需要的 make 和 gcc 在該節點下，點選紅色箭頭所指的迴圈箭頭圖示，將 Default 狀態切換成 Install 狀態。

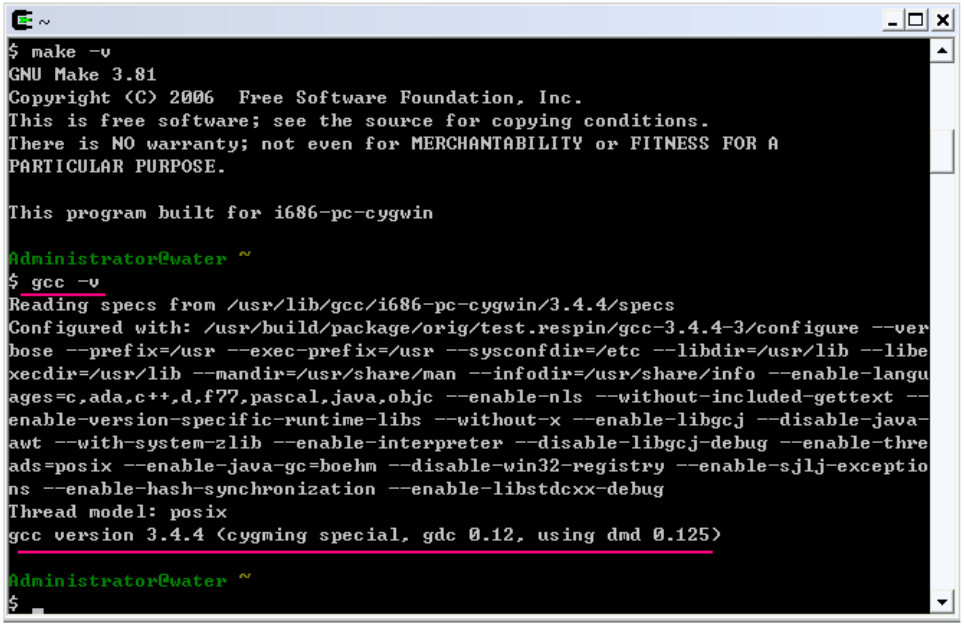

88% - Cygwin Setup
Progress
This page displays the progress of the download or installation.
Installing
subversion-1.4.2-1
/usr/bin/cygsvn_repos-1-0.dll
Progress:
Total:
Disk:
< 上一步 (B)
下一步 (N) >
取消

（4）　安裝成功後，執行 Cygwin。進入下圖所示介面：



（5） 輸入命令檢查 make，gcc 是否安裝成功，如下圖所示：



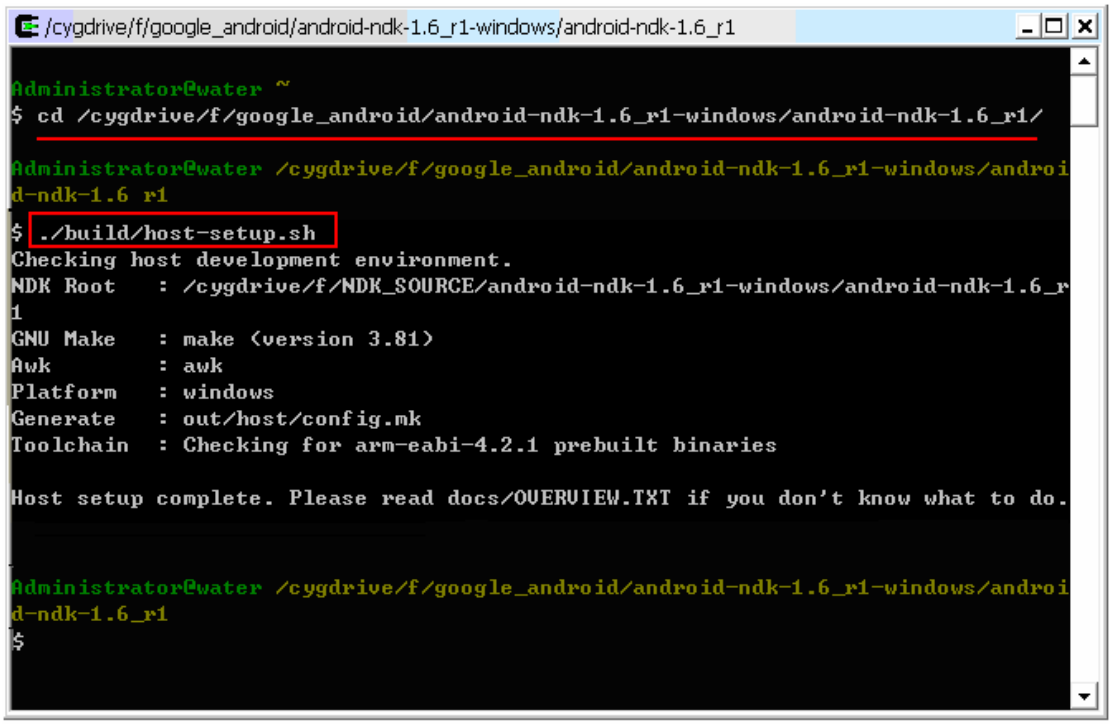


（6） 出現上述資訊，說明安裝成功。

7-1.2　設置環境變數

（1）　將下載的 NDK 壓縮包解壓(筆者解壓到的路徑爲：F:\google_android\android-ndk-1.6_r1-windows\android-ndk-1.6_r1)。

（2）　配置 NDK，打開 Cygwin，執行命令：cd /cygdrive/f/google_android/android-ndk-1.6_r1-windows/android-ndk-1.6_r1，進入到 NDK 存放的路徑，執行/build/host-setup.sh 腳本檔，如下圖所示則配置成功。

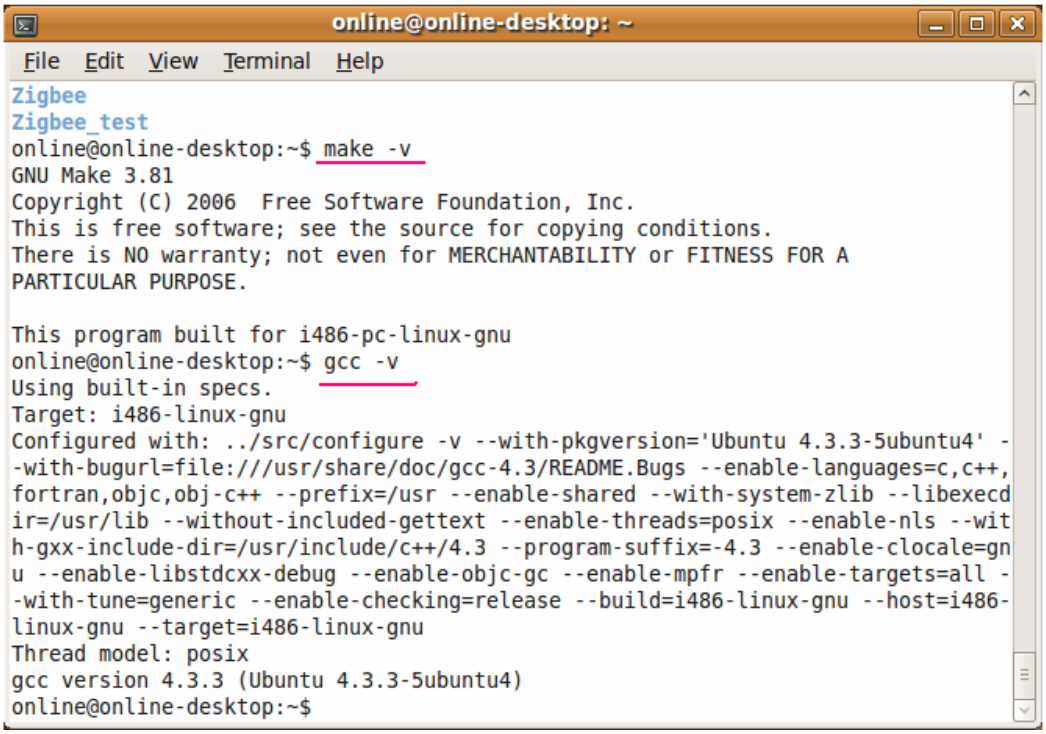


NDK 環境配置完成。

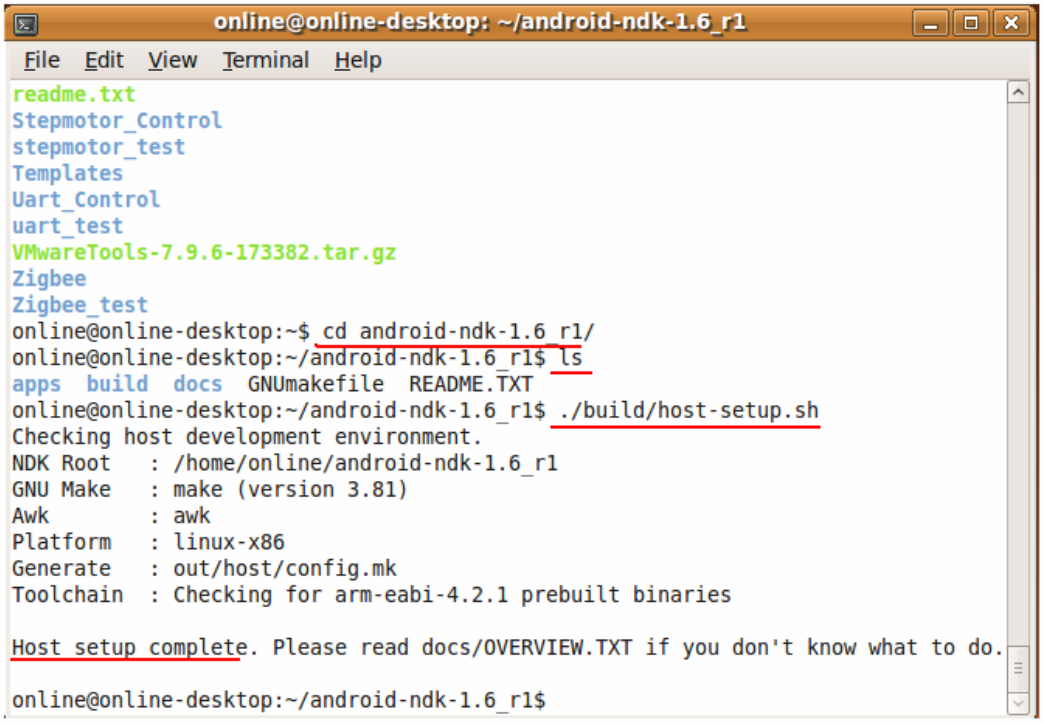
7-2 Linux 平台下的環境搭建與編譯

7-2.1 環境搭建

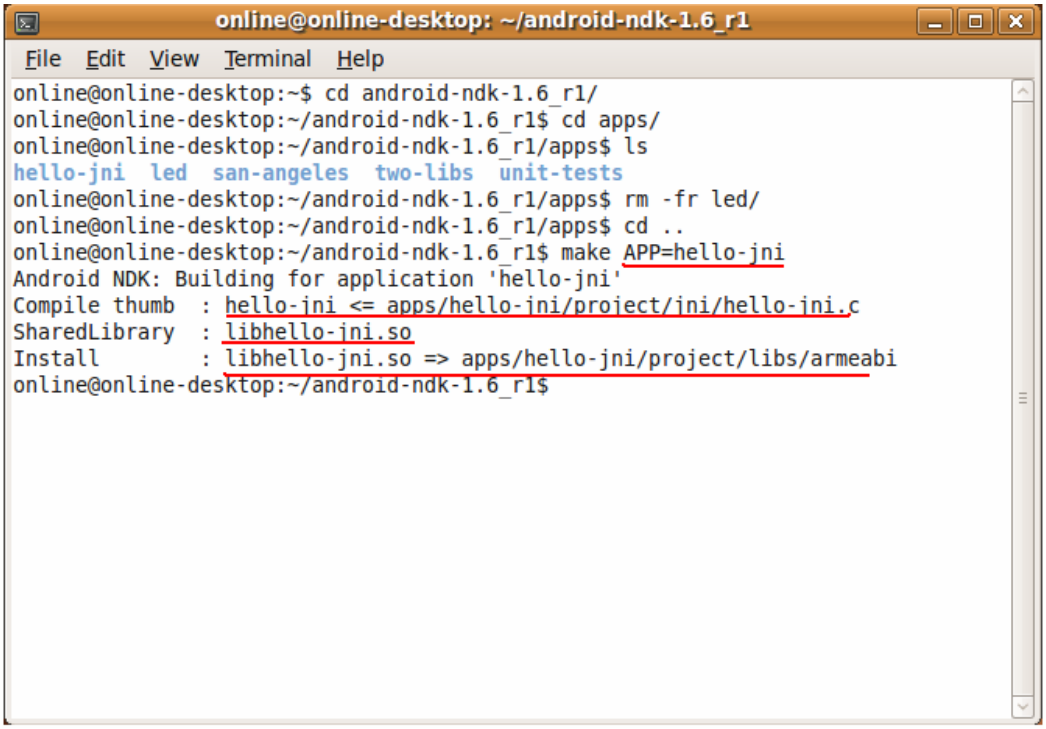
(1) Linux 環境下 make gcc 版本檢測，打開 Linux 命令終端，依次執行命令：make –v，gcc –v，如下圖所示：



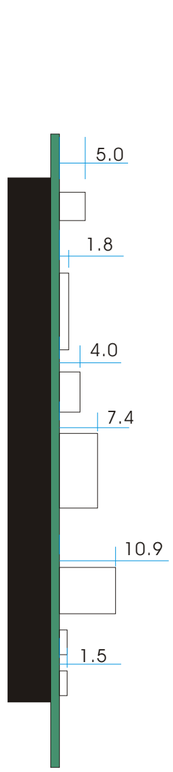
(2) 執行指令：unzip android-ndk-1.6_r1-linux-x86.zip，解壓 ndk 壓縮包。解壓後生成一個 android-ndk-1.6_r1 檔，進入到 android-ndk-1.6_r1/目錄下，執行命令：./build/host-setup.sh，如下圖所示：



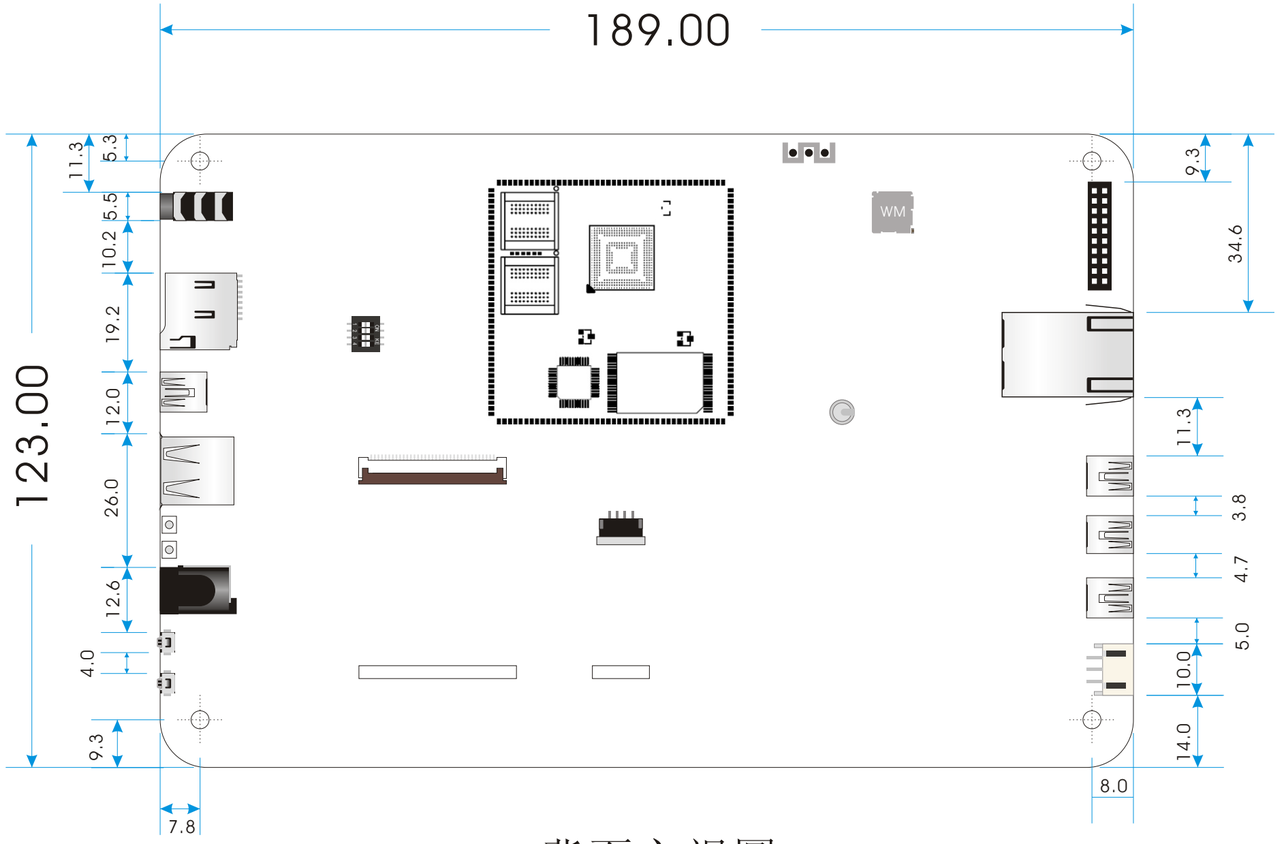
(3) 安裝後，可以編譯 ndk 中自帶的例子，來檢測環境是否搭建成功。執行命令：make APP=hello-jni，如下圖所示：



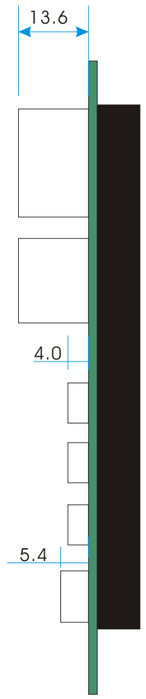
OK！環境搭建成功!



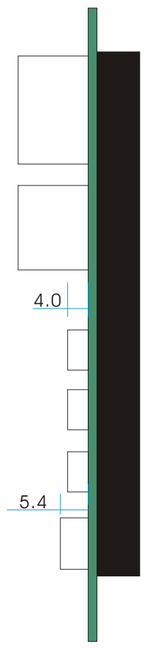
左視圖



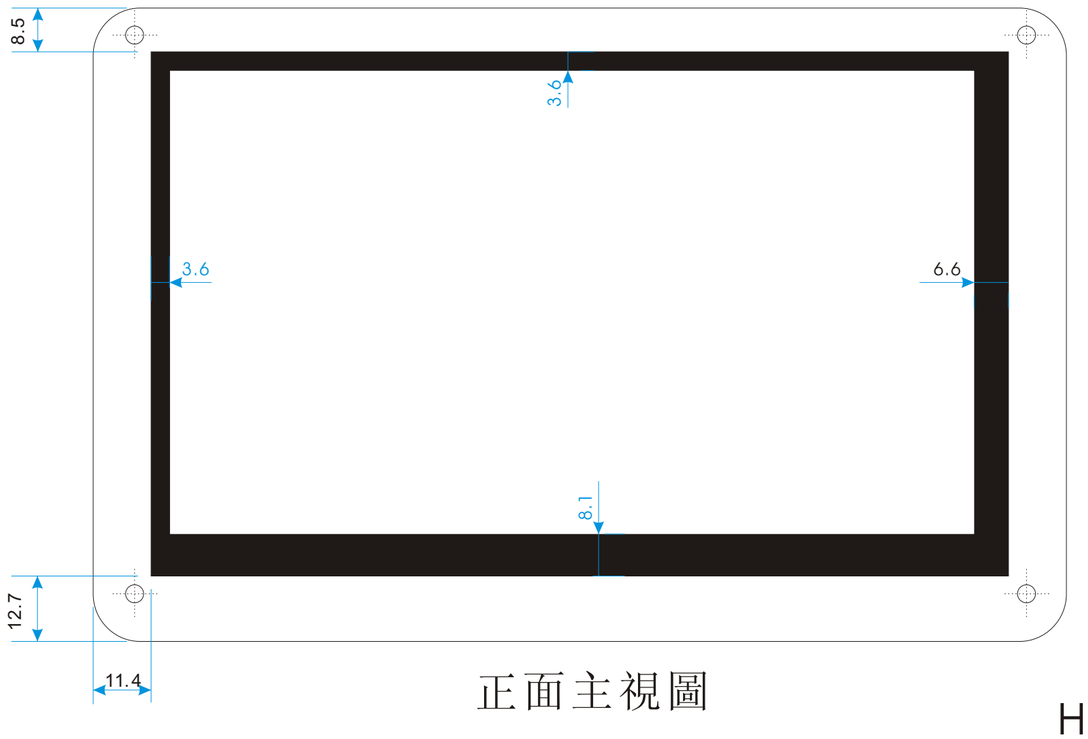
背面主視圖



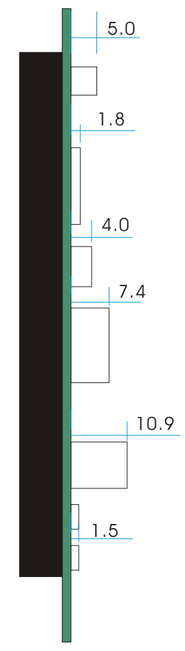
右視圖



左視圖



正面主視圖

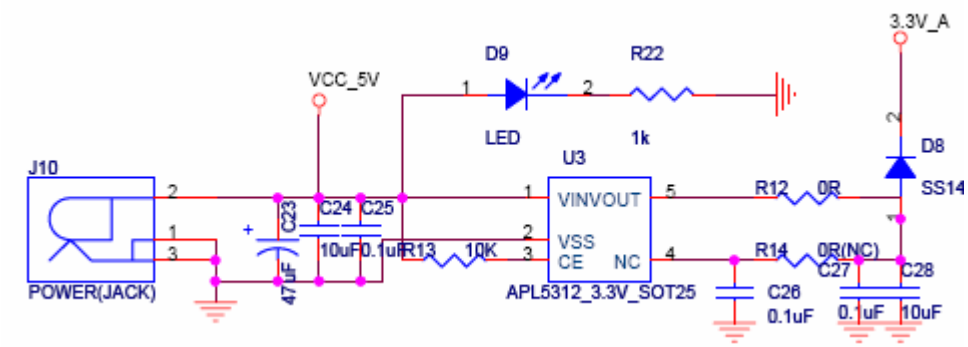


HMI700-6410S介面結構尺寸圖

附錄三 ZB2530-01 電路圖

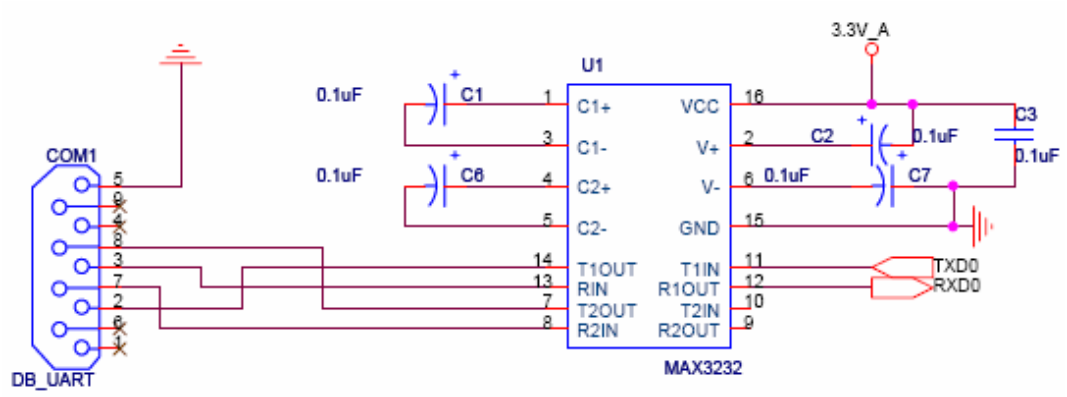
3-1 電源電路

電源電路圖如下：



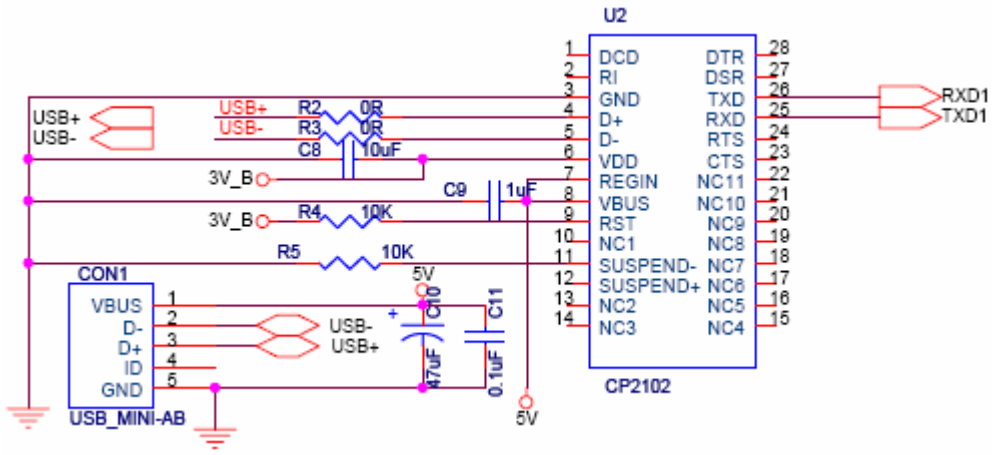
3-2 DB9 串列埠電路

DB 串列埠電路圖如下：



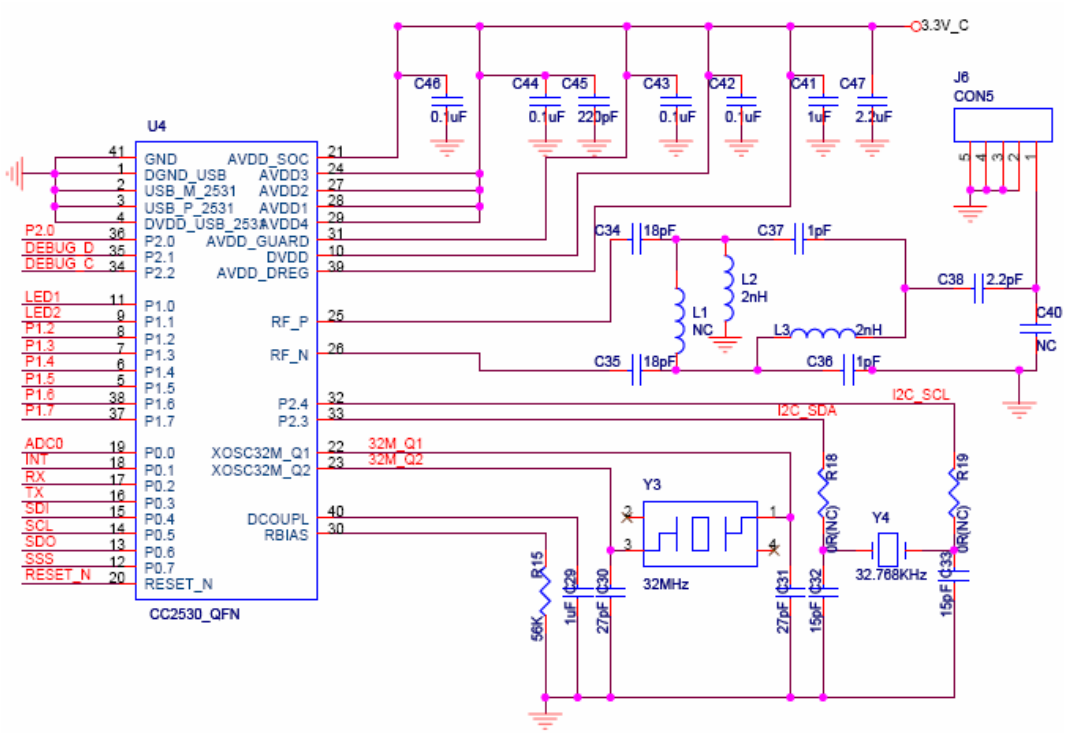
3-3 USB 串列埠電路

USB 串列埠電路圖如下：



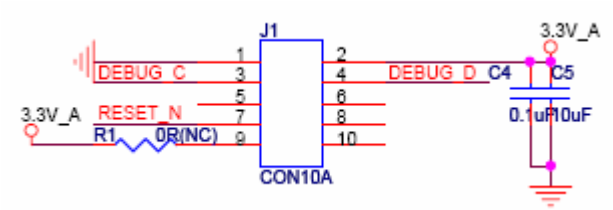
3-4 CPU 電路

CC2530 CPU電路圖如下：



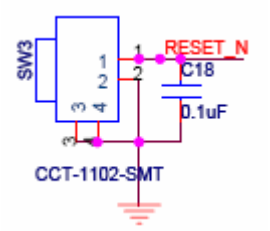
3-5 JTAG 電路

JTAG 電路圖如下：



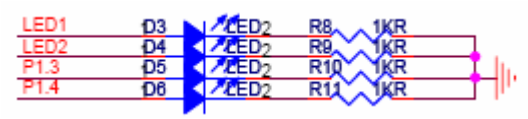
3-6 重置電路

複位電路圖如下：



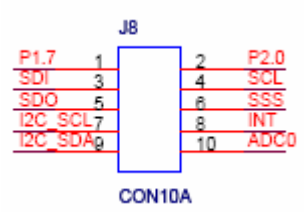
3-7 LED 指示電路

LED 指示燈電路圖如下：



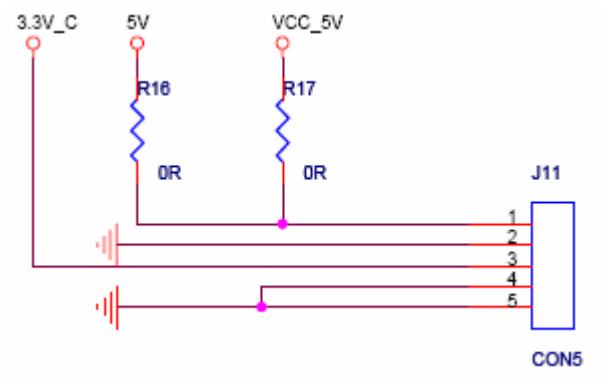
3-8 擴充埠 J8 介面電路

擴充埠 J8 介面電路圖如下：



3-9 擴充埠J11介面電路

擴充埠 J11 介面電路圖如下：



人體紅外感測器模組(B110)

功能特點:

1. 全自動感應：當有人進入其感應範圍則輸入高電平，人離開感應範圍則自動延時關閉高電平。輸出低電平。
2. 光敏控制(可選）：模組預留有位置，可設置光敏控制，白天或光線強時不感應。光敏控制爲可選功能,出廠時未安裝光敏電阻。如果需要，請另行購買光敏電阻自己安裝。
3. 兩種觸發方式：L 不可重複，H 可重複。**可跳線選擇，預設爲 H。**

   A.不可重複觸發方式：即感應輸出高電平後，延時時間一結束，輸出將自動從高電平變爲低電平。

   B.可重複觸發方式： 即感應輸出高電平後，在延時時間段內，如果有人體在其感應範圍內活動，其輸出將一直保持高電平，直到人離開後才延時將高電平變爲低電平(感應模組檢測到人體的每一次活動後會自動順延一個延時時間段，並且以最後一次活動的時間爲延時時間的起始點)。
4. 具有感應封鎖時間(預設設置：0.2 秒)：感應模組在每一次感應輸出後(高電平變爲低電平)，可以緊跟著設置一個封鎖時間，在此時間段內感應器不接收任何感應信號。此功能可以實現(感應輸出時間和封鎖時間)兩者的間隔工作，可應用於間隔探測產品；同時此功能可有效抑制負載切換過程中產生的各種干擾。
5. 工作電壓範圍寬：預設工作電壓 DC5V 至 20V
6. 微功耗：靜態電流 65 微安，特別適合乾電池供電的電器產品。
7. 輸出高電平信號：可方便與各類電路實現對接。

使用說明:

1. 感應模組通電後有一分鐘左右的初始化時間，在此時間模組會間隔地輸出0-3 次，一分鐘後進入待機狀態。
2. 應儘量避免燈光等干擾源近距離直射模組表面的透鏡，以免引進干擾信號產生誤動作；使用環境儘量避免流動的風，風也會對感應器造成干擾。
3. 感應模組採用雙元探頭，探頭的視窗爲長方形，雙元（A 元 B 元）位於較長方向的兩端，當人體從左到右或從右到左走過時,紅外光譜到達雙元的時間、距離有差值，差值越大，感應越靈敏，當人體從正面走向探頭或從上到下或從下到上方向走過時，雙元檢測不到紅外光譜距離的變化，無差值，因此感應不靈敏或不工作；所以安裝感應器時應使探頭雙元的方向與人體活動最多的方向儘量相平行，保證人體經過時先後被探頭雙元所感應。爲了增加感應角度範圍，本模組採用圓形透鏡，也使得探頭四面都感應，但左右兩側仍然比上下兩個方向感應範圍大、靈敏度強，安裝時仍須儘量按以上要求。

熱釋電感測器基本知識和使用中的常見問題

熱釋電紅外感測器是一種能檢測人或動物發射的紅外線而輸出電信號的感測器。早在 1938 年，有人提出過利用熱釋電效應探測紅外輻射，但並未受到重視，直到六十年代，隨著鐳射、紅外技術的迅速發展，才又推動了對熱釋電效應的研究和對熱釋電晶體的應用。熱釋電晶體已廣泛用於紅外光譜儀、紅外遙感以及熱輻射探測器，它可以作爲紅外鐳射的一種較理想的探測器。它目標正在被廣泛的應用到各種自動化控制裝置中。除了在我們熟知的樓道自動開關、防盜報警上得到應用外，在更多的領域應用前景看好。比如：在房間無人時會自動停機的空調機、飲水機。電視機能判斷無人觀看或觀眾已經睡覺後自動關機的機構。開

啓監視器或自動門鈴上的應用。結合攝影機或數碼照相機自動記錄動物或人的活動等等……。您可以根據自己的奇思妙想，結合其他電路開發出更加優秀的新產品。或自動化控制裝置。

熱釋電感測器基本知識

熱釋電效應同壓電效應類似，是指由於溫度的變化而引起晶體表面荷電的現象。熱釋電感測器是對溫度敏感的感測器。它由陶瓷氧化物或壓電晶體元件組成，在元件兩個表面做成電極，在感測器監測範圍內溫度有 ΔT 的變化時，熱釋電效應會在兩個電極上會產生電荷 ΔQ，即在兩電極之間產生一微弱的電壓 ΔV。由於它的輸出阻抗極高，在感測器中有一個場效應管進行阻抗變換。熱釋電效應所產生的電荷 ΔQ 會被空氣中的離子所結合而消失，即當環境溫度穩定不變時， ΔT=0，則感測器無輸出。當人體進入檢測區，因人體溫度與環境溫度有差別，產生 ΔT，則有 ΔT 輸出；若人體進入檢測區後不動，則溫度沒有變化，感測器也沒有輸出了。所以這種感測器檢測人體或者動物的活動傳感。 由實驗證明，感測器不加光學透鏡(也稱菲涅爾透鏡)，其檢測距離小於 2m，而加上光學透鏡後，其檢測距離可大於 7m。

使用中應注意以下幾點:

一、直流工作電壓必須符合我們要求的數值，過高和過低都會影響模組性能，而且要求電源必須經過良好的穩壓濾波，例如電腦 USB 電源、手機充電器電源、比較舊的 9V 的層疊電池都無法滿足模組工作要求，建議客戶用變壓器的電源並經過三端穩壓晶片穩壓後再通過 220UF 和 0.1UF 的電容濾波後供電。

二、除錯時人體儘量遠離感應區域，有時雖然人體不在模組的正前方，但是人體離模組太近時模組也能感應到造成一直有輸出，還有除錯時人體不要觸摸電路部分也會影響模組工作，比較科學的辦法是將輸出端接一個 LED 或者是萬用表，把模組用報紙蓋住，人離開這個房間，等 2 分鐘後看看模組是否還是一直有輸出？

三、模組不接負載時能正常工作，接上負載後工作紊亂，一種原因是因爲電源容量很小負載比較耗電，負載工作時引起的電壓波動導致模組誤動作，另一種原因是負載得電工作時會產生干擾，例如繼電器或者電磁鐵等感性負載會產生反向電動勢，315M 發射板工作時會有電磁輻射等都會影響模組。解決辦法如下：A、電源部分加電感濾波。B、採用負載和模組使用不同的電壓的方法，例如：負載使用 24V 工作電壓，模組使用 12V 工作電壓，其間用三端穩壓器隔離。C: 使用更大容量的電源。

四、人體感應模組只能工作在室內並且工作環境應該避免陽光、強烈燈光直接照射，如果工作環境有強大的射頻干擾，可以採用遮罩措施。若遇有強烈氣流干擾，關閉門窗或阻止對流。感應區儘量避免正對著發熱電器和物體以及容易被風吹動的雜物和衣物。

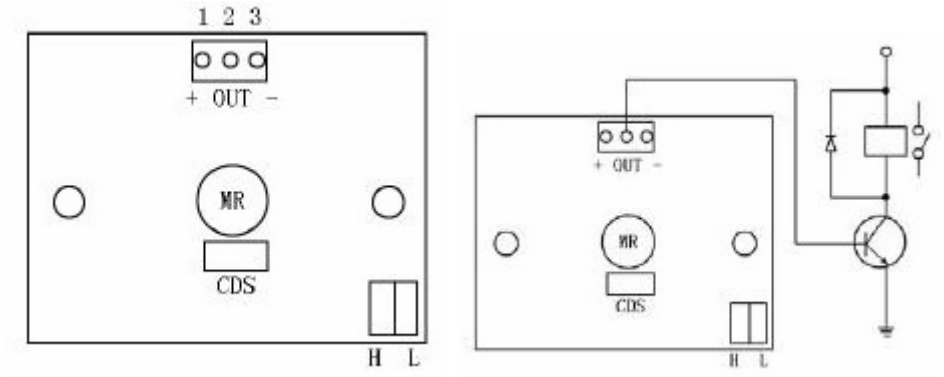
五、人體感應模組建議安裝在密封的盒裏，否則可能一直會有輸出信號。

六、如果要求人體感應模組的探測角度小於 90 度時，可以用不透明膠紙遮擋鏡片或裁剪縮小鏡片來實現。

七、人體感應模組採用雙元探頭，人體的手腳和頭部運動方向與感應靈敏度有著密切的聯繫，而且紅外模組的特性決定了無法精確控制感應距離。

八、模組中的探頭（PIR）可以裝焊在電路板的另一面。也可將探頭用雙芯遮罩線延長，長度應在 20 釐米以內爲好。

接繼電器的參考電路



1：正電源

2：高低電平輸出

3：電源負極

H：可重復觸發

L：不可重復觸發

CDS：光敏控制

技術參數:

1．工作電壓：DC5V 至 20V

2．靜態功耗：65 微安

3．電平輸出：高 3.3V，低 0V

4．延時時間：可調（0.3 秒~10 分鐘）

5．封鎖時間：0.2 秒

6．觸發方式：L 不可重復；H 可重復；預設值爲 H

7．感測範圍：小於 120 度錐角，7 米以內

8．工作溫度：-15~70℃

9．PCB 外形尺寸：32X24㎜，螺絲孔距 28㎜，螺絲孔徑 2㎜，感測透鏡尺寸

直徑 23㎜（預設）

應用範圍:

走廊、樓道、衛生間、地下室、倉庫、車庫等場所的自動照明、排氣扇的自動抽風及其他電器（白熾燈、熒光燈、蜂鳴器、自動門、電風扇、烘幹機和自動洗衣機），特別適用於企業、賓館、商場、庫房敏感區域或安全區域和報警系統，還可用於防盜等用途。

BMA023
Digital, triaxial acceleration sensor

Data sheet

Bosch Sensortec

BOSCH
Invented for life

BMA023 Data sheet

Order code	0 273 141 065
Package type	12-pin LGA
Data sheet version	1.1
Document number	BST-BMA023-DS000-01
Release date	07 April 2009
Notes	Specifications are subject to change without notice. Product photos and pictures are for illustration purposes only and may differ from the real product's appearance. Proprietary information – not for publication.

BMA023
Digital, triaxial ±2g/±4g/±8g acceleration sensor

Key features

- Three-axis accelerometer
- Small package — LGA package, Footprint 3mm x 3mm, height 0.90mm
- Digital interface — SPI (4-wire, 3-wire), I²C, interrupt pin
- Programmable functionality — g-range ±2g/±4g/±8g, bandwidth 25-1500Hz, internal acceleration evaluation for interrupt trigger, self-test
- Ultra-low power ASIC — Low current consumption, short wake-up time, advanced features for system power management
- RoHS compliant
- Halogen-free

Typical applications

- Menu scrolling
- Tap sensing functionality
- Gaming
- Pedometer, step-counting
- Drop detection for warranty logging
- Display profile switching
- Advanced system power management for mobile applications
- Shock detection

General description

The BMA023 is a triaxial, low-g acceleration sensor IC with digital output for consumer market applications. It allows measurements of acceleration in three perpendicular axes.

An evaluation circuitry converts the output of a three-channel micromechanical acceleration-sensing structure that works according to the differential capacitance principle.
Package and interface have been defined to match a multitude of hardware requirements. Since the sensor IC has small footprint and flat package it is attractive for mobile applications. The sensor IC can be programmed to optimize functionality, performance and power consumption in customer specific applications.

The BMA023 senses tilt, motion and shock vibration in cell phones, handhelds, computer peripherals, man-machine interfaces, virtual reality features and game controllers.

Contents

1. Specification

If not stated otherwise, the given values are maximum values over lifetime and full performance temperature/voltage range in the normal operation mode, and min./max. values represent 3-sigma limits.

Table 1: Operating range, output signal and mechanical specifications of BMA023

Parameter	Symbol	Condition	Min	Typ	Max	Units
OPERATING RANGE						
Acceleration range	g_{FS2g}	Switchable via serial digital interface	-2		2	g
	g_{FS4g}		-4		4	g
	g_{FS8g}		-8		8	g
Supply voltage analogue	V_{DD}		2.0		3.6	V
Supply voltage for digital I/O	V_{DDIO}	$V_{DDIO} \leq V_{DD}$	1.62		3.6	V
Supply current in normal mode	I_{DD}	Digital and analog		200	290	µA
Supply current in stand-by mode *	I_{DDsbm}	Digital and analog		1	2	µA
Operating temperature	T_A		-40		+85	°C
ACCELERATION OUTPUT SIGNAL						
Acceleration output resolution		Format: 2's complement			10	Bit
Sensitivity	S_{2g}	g-range ±2g, T_A=25°C	238	256	274	LSB/g
	S_{4g}	g-range ±4g, T_A=25°C	115	128	141	LSB/g
	S_{8g}	g-range ±8g, T_A=25°C	57	64	71	LSB/g
Sensitivity temperature drift	TCS	Over T_A		±0.03	±0.1	%/K
Zero-g offset	Off	T_A=25°C, calibrated	-75		+75	mg
Zero-g offset	Off	T_A=25°C , over lifetime	-180		+180	mg
Zero-g offset temperature drift	TCO	Over T_A	-6	1	+6	mg/K
Power supply rejection ratio	PSRR	Over V_{DD}			0.2	LSB/V

Parameter	Symbol	Condition	Min	Typ.	Max	Units
Bandwidth	bw	2nd order analog filter		1500		Hz
		Digital filter **		25, 50, 100, 190, 375, 750		Hz
Acceleration data refresh rate (all axes)	f_rate		2700	3000	3300	Hz
Nonlinearity	NL	Best fit straight line	-0.5		0.5	%FS
Output noise	n_{rms}	Rms		0.5		mg/√Hz
MECHANICAL CHARACTERISTICS						
Cross axis sensitivity	$\overline{S}$	Relative contribution between 3 axes			2	%
POWERING UP CHARACTERISTICS						
Wake-up time	t_{wu}	From standby		1	1.5	ms
Start-up time	t_{su}	From power-off		3		ms

Notes:

* For more details on the current consumption of the BMA023 during wake-up mode, please refer to chapter 7.3

** Please refer to chapter 3.1.3 for more detailed explanations

2. Maximum ratings

Table 2: Maximum ratings specified for the BMA023

Parameter	Condition	Min	Max	Units
Supply voltage	V_{DD} and V_{DDIO}	-0.3	4.25	V
Storage temperature range		-50	+150	°C
Mechanical shock	Duration ≤ 100µs		10,000	g
	Duration ≤ 1.0ms		2,000	g
	Free fall onto hard surfaces		1.5	m
ESD	HBM, at any pin		2	kV
	CDM		500	V

Note:
Stress above these limits may cause damage to the device. Exceeding the specified electrical limits may affect the device reliability or cause malfunction.

3. Global memory map

All operational registers are accessible through serial interface with a standard protocol:

Type of Register	Function of Register	Command	Volatile / non-volatile
Data Registers	– Chip identification, chip version – Acceleration data	Read Read	non-volatile (hard coded) volatile
Control Registers	– Activating self test, soft reset, switch to sleep mode etc.	Read / Write	volatile
Status Registers	– Interrupt status and self test status – Customer usable status bytes	Read Read / Write	Volatile volatile
Setting Register	– Functional settings (range, bandwidth) – Interrupt settings	Read / Write Read / Write	volatile volatile
EEPROM	– Default settings of functional and interrupt settings – Trimming values – Bosch Sensortec reserved memory	Write Protected Protected	non-volatile non-volatile non-volatile

Figure 1: Global memory map of BMA023

Memory Region	Register Address (hexadecimal)	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	type	Default setting
Default Settings	16h to 7Fh	reserved								reserved	NA
Operational Registers	15h	SPI4	enable_adv_INT	new_data_INT	latch_INT	shadow_dis	wake_up_pause		wake_up	control	1 0 0 0 0 0 0 0b
	14h	reserved			range<1:0>		bandwidth<2:0>			control	XXX 00 000b
	13h	customer_reserved 2 <7:0>								status	NA
	12h	customer_reserved 1 <7:0>								status	NA
	11h	any_motion_dur		HG_hyst<2:0>			LG_hyst<2:0>			settings	NA
	10h	any_motion_thres<7:0>								settings	NA
	0Fh	HG_dur<7:0>								settings	NA
	0Eh	HG_thres<7:0>								settings	NA
	0Dh	LG_dur<7:0>								settings	NA
	0Ch	LG_thres<7:0>								settings	NA
	0Bh	alert	any_motion	counter_HG		counter_LG		enable_HG	enable_LG	control	0
	0Ah	reserved	reset_INT	reserved	reserved	self_test_1	self_test_0	soft_reset	sleep	control	0
	09h	st_result	reserved		alert_phase	LG_latched	HG_latched	status_LG	status_HG	status	NA
	08h	unused								data	NA
	07h	acc_z<9:2> (msb)								data	NA
	06h	acc_z<1:0> (lsb)		unused					new_data_z	data	NA
	05h	acc_y<9:2> (msb)								data	NA
	04h	acc_y<1:0> (lsb)		unused					new_data_y	data	NA
	03h	acc_x<9:2> (msb)								data	NA
	02h	acc_x<1:0> (lsb)		unused					new_data_x	data	NA
	01h	al_version<3:0>				ml_version<3:0>				data	NA
	00h	unused					chip_id<2:0>			data	----- 010b

Important notes:

1) Bits 5, 6 and 7 of register addresses 14h do contain critical sensor individual calibration data which must not be changed or deleted by any means.

In order to properly modify addresses 14h for range and/or bandwidth selection using bits 0, 1, 2, 3 and 4, it is highly recommended to read-out the complete byte, perform bit-slicing and write back the complete byte with unchanged bits 5, 6 and 7.

Otherwise the reported acceleration data may show incorrect results.

2) Bit 7, bit 5 and bit 4 of register 0Ah should be left at a value of “0”.

3.1 Operational registers

3.1.1 SPI4

The SPI4 bit ((address 15h, bit 7) is used to select the correct SPI protocol (three-wire or four-wire, SPI-mode 3). The default value is SPI4=1 (four-wire SPI is default value). After power on reset or soft reset it is recommended to set the SPI4-bit to the correct value.
This first writing is possible because only CSB, SCK and SDI are required for a write sequence and the 3 bit timing diagrams are identical in three-wire and four-wire configuration.

Recommended procedure: Set SPI4 to the correct value (SPI4=0 for SPI three-wire, SPI4=1 for SPI four-wire (=default)) every time after power on reset or soft reset.

3.1.2 Range

These two bits (address 14h, bits 4 and 3) are used to select the full scale acceleration range.
Directly after changing the full scale range it takes 1/(2*bandwidth) to overwrite the data registers with filtered data according to the selected bandwidth.

Table 3: Settings of full scale range register

range<1:0>	Full scale acceleration range
00	+/- 2g
01	+/- 4g
10	+/- 8g
11	Not authorised code

Important note:
Please refer to the comment in chapter 3 of how to protect bits 5, 6 and 7 when modifying other bits of register 14h.

3.1.3 Bandwidth

These three bits (address 14h, bits 2-0) are used to setup the digital filtering of ADC output data to obtain the desired bandwidth. A second order analogue filter defines the max. bandwidth to 1.5kHz. Digital filters can be activated to reduce the bandwidth down to 25Hz in order to reduce signal noise. The digital filters are moving average filters of various length with a refresh rate of 3kHz.

Since the bandwidth is reduced by a digital filter for the factor ½ , ¼, ... of the analogue filter frequency of 1.5kHz the mean values of the bandwidth are slightly deviating from the rounded nominal values. Table 4 shows the corresponding data:

Table 4: Settings of bandwidth

bandwidth<2:0>	Nominal selected bandwidth [Hz]	Min.	Mean bandwidth[Hz]	Max.
000	25	-10%	23	+10%
001	50		47	
010	100		94	
011	190		188	
100	375		375	
101	750		750	
110	1500		1500	
111	Not authorised code	-	-	-

At wake-up from sleep mode to normal operation, the bandwidth is set to its maximum value and then reduced to bandwidth setting as soon as enough ADC samples are available to fill the whole digital filter.

Important note:
Please refer to the comment in chapter 3 of how to protect bits 5, 6 and 7 when modifying other bits of register 14h.

3.1.4 Wake_up

This bit (address 15h, bit 0) makes BMA023 automatically switching from sleep mode to normal mode after the delay defined by wake_up_pause (section 3.1.5). When the sensor IC goes from sleep to normal mode, it starts acceleration acquisition and performs interrupt verification (section 3.2). The sensor IC automatically switches back from normal to sleep mode again if no fulfilment of programmed interrupt criteria has been detected. The IC wakes-up for a minimum duration which depends on the number of required valid acceleration data to determine if an interrupt should be generated.

If a latched interrupt is generated, this can be used to wake-up a microprocessor. The sensor IC will wait for a reset_INT command and restart interrupt verification. BMA023 can not go back to sleep mode if reset_INT is not issued after a latched interrupt.

If a not-latched interrupt is generated, the device waits in the normal mode till the interrupt condition disappears. The minimum duration of interrupt activation is 330µs. If no interrupt is generated, the sensor IC goes to sleep mode for a defined time (wake_up_pause).

For more details on the wake-up functionality, please refer to chapter 7.3

3.1.5 Wake_up_pause

These bits (address 15h, bit 2 and 1) define the sleep phase duration between each automatic wake-up.

Table 5: Settings of wake_up_pause

wake_up_pause<1:0>	Sleep phase duration
00	20 ms
01	80 ms
10	320 ms
11	2560 ms

Note: The accuracy of the wake-up timer is about ±30%.

3.1.6 Shadow_dis

BMA023 provides the possibility to block the update of data MSB while LSB are read out. This avoids a potential mixing of LSB and MSB of successive conversion cycles. When this bit (address 15h, bit 3) is at 1, the blocking procedure for MSB is not realized and MSB only reading is possible.

3.2 Interrupt settings

Five different types of interrupts can be programmed. When the corresponding criterion becomes valid, the interrupt pin is triggered to a high level. All interrupt criteria are combined and drive the interrupt pad with a Boolean <OR> condition.

Interrupt generations may be disturbed by changes of control bits because some of these bits influence the interrupt calculation. As a consequence, no write sequence should occur when microprocessor is triggered by interrupt or the interrupt should be deactivated on the microprocessor side when write sequences are operated.

Interrupt criteria are using digital code coming from digital filter output. As a consequence all thresholds are scaled with range selection (section 3.1.3.2). Timings used for high acceleration and low acceleration debouncing are absolute values (1 LSB of HG_dur and LG_dur registers corresponds to 1 millisecond, timing accuracy is proportional to oscillator accuracy = +/-10%), thus it does not depend on selected bandwidth. Timings used for any motion interrupt and alert detection are proportional to bandwidth settings (section 3.1.3).

3.2.1 Enable_LG

This bit (address 0Bh, bit 0) enables the LG_thres criteria to generate an interrupt.

3.2.2 Enable_HG

This bit (address 0Bh, bit 1) enables the HG_thres criteria to generate an interrupt.

3.2.3 Enable_adv_INT

This bit (address 15h, bit 6) is used to disable advanced interrupt control bits (any_motion, alert). If enable_adv_INT=0, writing to these bits has no effect on sensor IC function.

3.2.4 Any_motion

This bit ((address 0Bh, bit 6)enables the any motion criteria to generate directly an interrupt. It can not be turned on simultaneously with alert. This bit can be masked by enable_ adv_INT, the value of this bit is ignored when enable_adv_INT=0 (section 3.2.3).

3.2.5 Alert

If this bit (address 0Bh, bit 7) is at 1, the any_motion criterion will set BMA023 into alert mode (section 3.2.9). This bit can be masked by enable_adv_INT, the value of this bit is ignored when enable_adv_INT=0 (section 3.2.3).

3.2.6 Latch_INT

If this bit (address 15h, bit 4) is at 1, interrupts are latched. The INT pad stays high until microprocessor detects it and writes reset_INT control bit to 1 (section 3.3.1). When this bit is at 0, interrupts are set and reset directly by BMA023 according to programmable criteria (sections 3.2.7 and 3.2.8).

3.2.7 LG_thres, LG_hyst, LG_dur, counter_LG

LG_thres (address 0C, bits 7-0 / low-g threshold) and LG_hyst (address 11h, bits 2-0 / low-g threshold hysteresis) can basically (although not recommended with BMA023) be used to detect a free fall. The threshold and duration codes define one criterion for interrupt generation when absolute value of acceleration is low for long enough duration.

Data format is unsigned integer.

LG_thres criterion_x is true if	$\|acc\_x\| \leq$ LG_thres / 255 * range
LG_thres interrupt is set if	(LG_thres criterion_x AND LG_thres criterion_y AND LG_thres criterion_z) AND interrupt counter = (LG_dur+1)
LG_thres criterion_x is false if	$\|acc\_x\| >$ (LG_thres + 32*LG_hyst) / 255 * range
LG_thres interrupt is reset if	NOT(LG_thres criterion_x AND LG_thres criterion_y AND LG_thres criterion_z)

LG_thres and LG_hyst codes must be chosen to have (LG_thres + 32*LG_hyst) < 511.

When LG_thres criterion becomes active, an interrupt counter is incremented by 1 LSB/ms. When the low-g interrupt counter value equals (LG_dur+1), an interrupt is generated.
Depending on counter_LG (address 0Bh, bit 3 and 2) register, the counter could also be reset or count down when LG_thres criterion is false.

Table 6: Description of debouncing counter counter_LG

counter_LG<1:0>	low acceleration interrupt counter status when LG_thres criteria is false
00	reset
01	Count down by 1 LSB/ms
10	Count down by 2 LSB/ms
11	Count down by 3 LSB/ms

If latch_INT=0, the interrupt is not a latched interrupt and then it is reset as soon as LG_thres criteria becomes false. When interrupt occurs, the interrupt counter is reset.

Remark: The LG_thres criteria is set with an AND condition on all three axes to be used for free fall detection. However, please note due to the relatively wide sensitivity tolerance of the BMA023 the absolute threshold values for low-g and high-g interrupt can only provide a rough estimation of the motion profile.

3.2.8 HG_thres, HG_hyst, HG_dur, counter_HG

HG_thres (address 0Eh, bits 7-0 / high-g threshold) and HG_hyst (address 11h, bits 5-3 / high-g threshold hysteresis) define the high-G level and its associated hysteresis. HG_dur (high-g threshold qualification duration) and counter_HG (address 0Bh, bits 5 and 4 / high-g counter down register) are used for debouncing the high-g criteria.
Threshold and duration codes define a criterion for interrupt generation when absolute value of acceleration is high for long enough duration.

The data format is unsigned integer.

HG_threshold criterion_x is true if	$\|acc\_x\| \geq HG\_thres / 255 * range$
HG_threshold interrupt is set if	(HG_thres criterion_x OR HG_thres criterion_y OR HG_thres criterion_z) AND interrupt counter = (HG_dur+1)
HG_threshold criterion_x is false if	$\|acc\_x\| < (HG\_thres - 32*HG\_hyst) / 255 * range$
HG_threshold interrupt is reset if	NOT(HG_thres criterion_x OR HG_thres criterion_y OR HG_thres criterion_z)

HG_thres and HG_hyst codes must be chosen to have (HG_thres - 32*HG_hyst) > 0.

When HG_thres criterion becomes active, a counter is incremented by 1 LSB/ms. When the high-g acceleration interrupt counter value equals (HG_dur+1), an interrupt is generated. Depending on counter_HG register value, the counter could also be reset or count down when HG_thres criterion is false.

Table 7: Description of debouncing counter_HG

counter_HG<1:0>	High acceleration interrupt counter status when HG_thres criterion is false
00	reset
01	Count down by 1 LSB/ms
10	Count down by 2 LSB/ms
11	Count down by 3 LSB/ms

If latch_INT=0, the interrupt is not a latched interrupt and then it is reset as soon as HG_thres criterion becomes false. When interrupt occurs, the interrupt counter is reset.

3.2.9 Any_motion_thres, any_motion_dur

For the evaluation using "any motion" criterion successive acceleration data from digital filter output are stored and moving differences for all axes are built. To calculate the difference the acceleration values of all axes at time t0 are compared to values at t0+3/(2*bandwidth). The difference of both values is equal to the difference of two successive moving averages (from three data points).
The differential value is compared to a global critical threshold any_motion_thres (address 10h, bits 7-0). Interrupt can be generated when the absolute value of measured difference is higher than the programmed threshold for long enough duration defined by any_motion_dur (address 11h, bits 7 and 6).

Any_motion_thres and any_motion_dur data are unsigned integer. Any_motion_thres LSB size corresponds to 15.6mg for +/- 2g range and scales with range selection (section 3.1.2).

Any motion criterion is valid if	$\|acc(t0)-acc(t0+3/(2*bandwidth))\| \geq$ any_motion_thres.
An interrupt is set if	(any motion criterion_x OR any motion criterion_y OR any motion criterion_z) for any_motion_dur consecutive times.
The any motion interrupt is reset if	NOT(any_motion criterion_x OR any_motion criterion_y OR any_motion criterion_z) for any_motion_dur consecutive times.

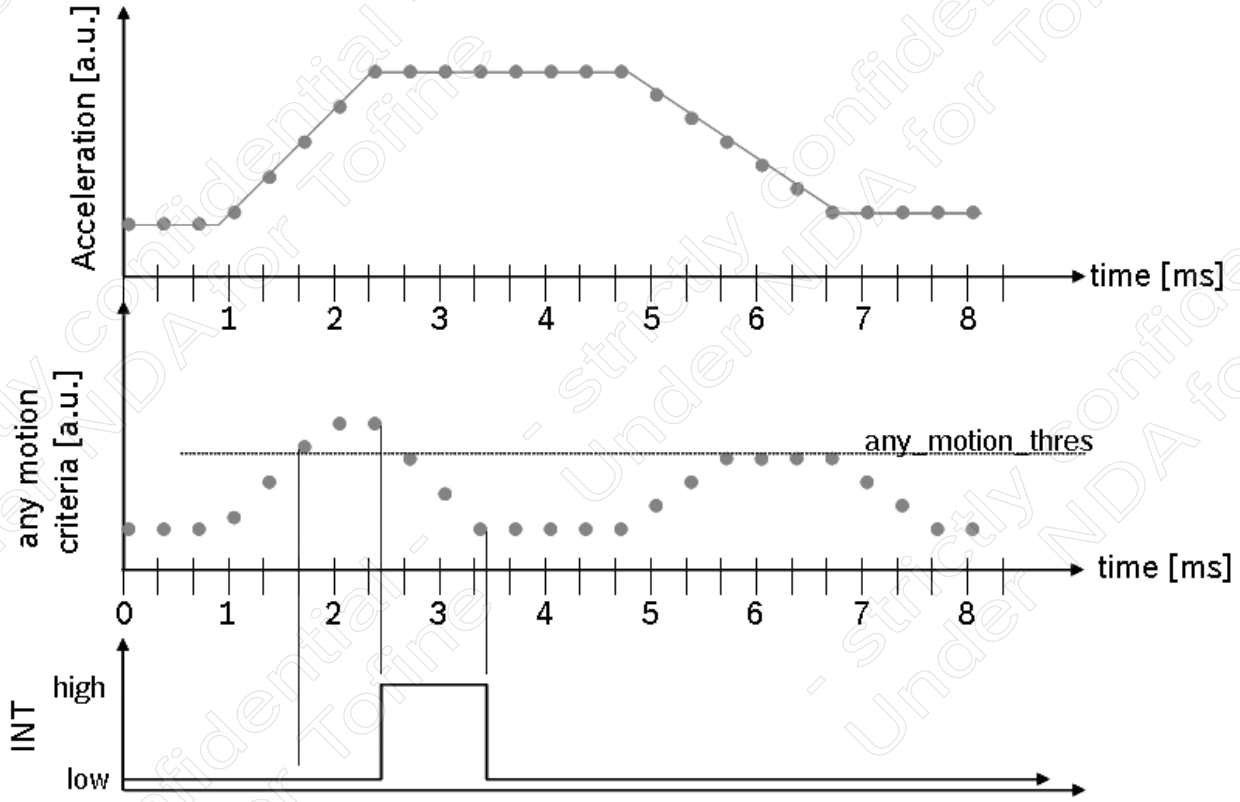
Table 8: any_motion_dur settings

any_motion_dur<1:0>	Number of required consecutive conditions to set or reset the any motion interrupt
00	1
01	3
10	5
11	7

Any_motion_dur is used to filter the motion profile and also to define a minimum interrupt duration because the reset condition is also filtered.

Any_motion_thres can be used to generate an any_motion interrupt or to put BMA023 in alert mode to preload the low-g or high-g threshold logic (enables reduction of reaction time in tumbling mode); this is selected by alert bit (section 3.2.5). These two modes (any_motion and alert) can not be turned on simultaneously.

Figure 2: Any motion criterion (middle graph) is determined from digital filter output (upper graph) and depends on bandwidth settings: for example for any_motion_dur=01b and bandwidth=110b (1.5kHz), we have 2*bandwidth=3ksamples/s which leads to reaction for interrupt activation of 3*333µs = 1ms and a minimum any motion interrupt duration of 3*333us = 1ms (see lower graph).
If lower bandwidth is selected i) the digitally filtered values (lower noise) are taken for the verification of the any motion criterion and ii) the time scale to evaluate the criterion is stretched. Thus adjusting the bandwidth, the any motion threshold, the any motion duration as well as the full scale range enables to tailor the sensitivity of the any motion algorithm.

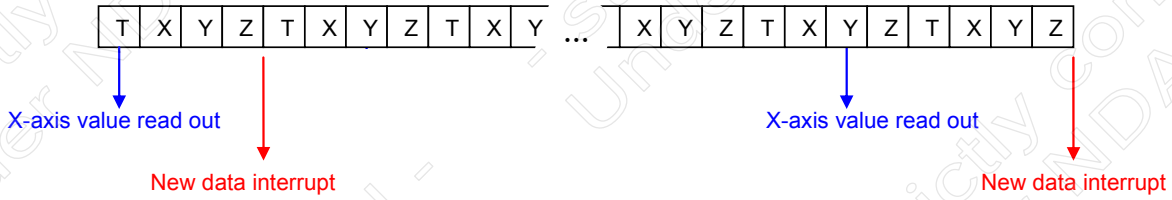


3.2.10 New_data_int

If this bit (address 15h, bit 5) is set to 1, an interrupt will be generated when all three axes acceleration values are new, i.e. BMA023 updated all acceleration values after latest serial read-out. Interrupt generated from new data detection is a latched one; microcontroller has to write reset_INT at 1 after interrupt has been detected high (section 3.3.1). This interrupt is also reset by any acceleration byte read procedure (read access to address 02h to 07h).
New data interrupt always occurs at the end of the Z-axis value update in the output register (3kHz rate). Following figure shows two examples of X-axis read out and the corresponding interrupt generation.

Figure 3: Explanation of new data interrupt.

left side - read out command of x-axis prior to next x-axis conversion
→ new data interrupt after completion of current conversion cycle after z-axis conversion
right side - read out of x-axis send after x-axis conversion
→ new data interrupt at the end of next period when x axis has been updated



Please refer to chapter 8.1 for more details.

Note: When using the I^2C interface for data transfer, the data read out phase can be longer than 330µs (depending on I^2C clock frequency and the amount of data transmitted). Starting a new data read out sequence may lead to the situation that the new_data_int may not be cleared right in time. This must be considered and taken care of properly.

3.3 Control registers

All single control bits are active at 1.

3.3.1 Reset_INT

This interrupt (address 0Ah, bit 6) is reset (interrupt pad goes to low) each time this bit is written to 1.

3.3.2 Selftest_0

The self-test command (address 0Ah, bit 2) uses electrostatic forces to move the MEMS common electrode. The result from selftest can be verified by reading st_result (section 3.4.1). During the self-test procedure no external change of the acceleration should be generated.

3.3.3 Selftest_1

This self test bit (address 0Ah, bit3) does not generate any electrostatic force in the MEMS element but is used to verify the interrupt function is working correctly and that microprocessor is able to react to the interrupts.

0g acceleration is emulated at ADC input and the user can detect the whole logic path for interrupt, including the PCB path integrity. The LG_thres register must be set to about 0.4g while LG_dur = 0 to generate a low-g interrupt

3.3.4 Soft_reset

BMA023 is reset each time this bit (address 0Ah, bit 1) is written to 1. The effect is identical to power-on reset. Control, status and image registers are reset to values stored in the default setting registers (see also memory map). After soft_reset or power-on reset BMA023 comes up in normal mode or wake-up mode. It is not possible to boot BMA023 to sleep mode.

No serial transaction should occur within 10µs after soft_reset command.

The soft_reset procedure may overwrite the SPI4 setting (section 3.1.1). Thus the correct interface configuration may have to be updated.

3.3.5 Sleep

This bit (address 0Ah, bit 0) turns the sensor IC in sleep mode. Control and image registers are not cleared.
When BMA023 is in sleep mode no operation can be performed but wake-up the sensor IC by setting sleep=0 or soft_reset. As a consequence all write and read operations are forbidden when the sensor IC is in sleep mode except command used to wake up the device or soft_reset command. After sleep mode removal, it takes 1ms to obtain stable acceleration values (>99% data integrity).

3.4 Status registers

3.4.1 St_result

This is the self test result bit (address 09h, bit 7). It can be used together with selftest_0 control bit (section 3.3.4). After selftest_0 has been set, self-test procedure starts. At the end selftest_0 is written to 0 and microcontroller can react by reading st_result bit. When st_result=1 the self test passed successfully.
The result of the st_result can be taken into account to evaluate the basic function of the sensor. Note: Evaluation of the st_result bit should only be understood as one part of a wider functionality test. It should not be taken into consideration as the only criterion.

3.4.2 Alert_phase

This status bit (address 09h, bit 4) is set when BMA023 has been set to alert mode (section 3.2.5) and an any motion criterion has been detected. During alert phase, HG_dur and LG_dur variables are decreased to have a smaller reaction time when HG_thres and LG_thres thresholds are crossed; the decrease rate is by 1 ms per ms.

The alert mode is reset when an interrupt generated due to a high threshold or a low threshold event or when both HG_dur and LG_dur variables are at 0. When alert is reset, HG_dur and LG_dur variables come back to their original values stored in image registers.

3.4.3 LG_latched, HG_latched

These status bits (address 09h, bit 3 and address 09h, bit 2) are set when the corresponding criteria have been issued. They are latched and thus only the microcontroller can reset them. When both high acceleration and low acceleration thresholds are enabled, these bits can be used by microprocessor to detect which criteria generated the interrupt.

3.4.4 Status_LG, status_HG

These status bits (address 09h, bit 1 and address 09h, bit 2) are set when the corresponding criteria have been issued; they are automatically reset by BMA023 when the criteria disappear.

3.4.5 Customer_reserved 1, customer_reserved 2

Both bytes (address 12h, bit 7-0 and address 13h, bit 7-0) can be used by customer. Writing or reading of these registers has no effect on the sensor IC functionality.

3.5 Data registers

3.5.1 Acc_x, acc_y, acc_z

Acceleration values are stored in the following registers to be read out through serial interface.
acc_x (02h, 7-6; 03h, 7-0)
acc_y (04h, 7-6; 05h, 7-0)
acc_z (06h, 7-6; 07h, 7-0)

The description of the digital signals acc_x, acc_y and acc_z is "2's complement".

From negative to positive accelerations, the following sequence for the ±2g measurement range can be observed (±4g and ±8g correspondingly):

-2.000g	:	10 0000 0000
-1.996g	:	10 0000 0001
...		
-0.004g	:	11 1111 1111
0.000g		:00 0000 0000
+0.004g	:	00 0000 0001
...		
+1.992g	:	01 1111 1110
+1.996g	:	01 1111 1111

Data is periodically updated (rate 3kHz) with values from the digital filter output. LSB acceleration bytes must be read first. After an acceleration LSB byte read access, the corresponding MSB byte update can optionally be blocked until it is also accessed for read. Thus, MSB / LSB mix from different samples can be avoided (section 3.1.6).
It is not possible to read-out only MSB bytes if shadow_dis=0, an LSB byte must first be read out. To be able to read out only MSB byte, shadow_dis must be written to 1.

new_data_* flags on bits 0 of acc_x (LSB), acc_y (LSB) and acc_z (LSB) can be used to detect if acceleration values have already been read out (section 3.5.3).
If systematic acceleration values read out is planned (for signal processing by the microcontroller), the interrupt pad can be programmed to flag the new data (section 3.2.10). Every time all three axes values have been updated, the interrupt goes high and microcontroller can read out data. With this method, microcontroller accesses are synchronized with internal sensor IC updates.

Synchronization of read-out sequence has several advantages:

- it enables a constant phase shift between acceleration conversion and its corresponding digital value read by microprocessor
- it reduces interface communication by avoiding over-sampling.
- potential noise due to serial interface activity perturbation would always be generated during a less critical phase of the conversion cycle. The maximum delay advised to start read out acceleration data is 20µs after INT high (window 0-80µs).

3.5.2 New_data_x, new_data_y, new_data_z

These bits (New_data_x (02h, 0), new_data_y (04h, 0), new_data_z (06h, 0)) are flags which are turned at 1 when acceleration registers have been updated. Reading acceleration data MSB or LSB registers turns the flags at 0. The flag value can be read by microprocessor.

3.5.3 Al_version, ml_version, chip_id

al_version (address 01h, bit 7-4) and ml_version (address 01h, bit 3-0) are used to identify the chip revision. These codes are programmed with metal layer.

chip_id (address 00h, bit 2-0) is used by customer to be able to recognize BMA023. This code is fixed to 010b.

4. Digital interface

BMA023 is capable to be adjusted to customer's specific hardware requirements. It provides three different digital interfaces (SPI 4-wire, SPI 3-wire, I²C) and an interrupt output pin.

The digital interface is used for regular reading of data registers (acceleration). For a complete read out of acceleration data two successive read cycles are required. The 10 bit coded data word is split into 8 MSB and 2 LSB. The most significant bit (MSB) is transferred first during address and data phases.
The serial interface is also used for verifying status registers or writing to control registers.

4.1 SPI

The SPI interfaces using three wire or four wire bus provide 16-bit protocols. Multiple read out is possible.
The communication is opened with a read/write control bit (R/W=0 for writing, R/W=1 for reading) followed by 7 address bits and at least 8 data bits (see figure 6 and figure 7). For a complete readout of 10 bit acceleration data from all axes the sensor IC provides the option to use an automatic incremented read command to read more than one byte (multiple read). This is activated when the serial enable pin CSB (chip select) stays active low after the read out of a data register. Thus, read out of data LSB will also cause read out of MSB if the CSB stays low for further 8 cycles of system clock.

The customer has the possibility to communicate with operational registers at addresses 00h-15h via SPI interface (chip identification Bytes, data Bytes, status and control registers with setting parameters). Access to the residual part of the memory map is locked (section 3.3.3). If the master addresses outside the range 00h-15h then SDI will go to tri-state enabling the communication of a second device on the same CSB and SDI line.
The CSB input has an internal 120kΩ pull-up resistor to V_{DDIO}.

4.1.1 Four-wire SPI interface

The 4-wire SPI is the default serial interface. The customer can easily activate the 3-wire SPI by writing a control bit (SPI4=0). The 4-wire SPI interface uses SCK (serial clock), CSB (chip select), SDI (serial data in) and SDO (serial data out).

CSB is active low. Data on SDI is latched by BMA023 at SCK rising edge and SDO is changed at SCK falling edge (SPI mode 3). Communication starts when CSB goes to low and stops when CSB goes to high; during these transitions on CSB, SCK must be high. While CSB=1, no SDI change is allowed when SCK=1.

Figure 4: Timing diagram for four-wire SPI interface

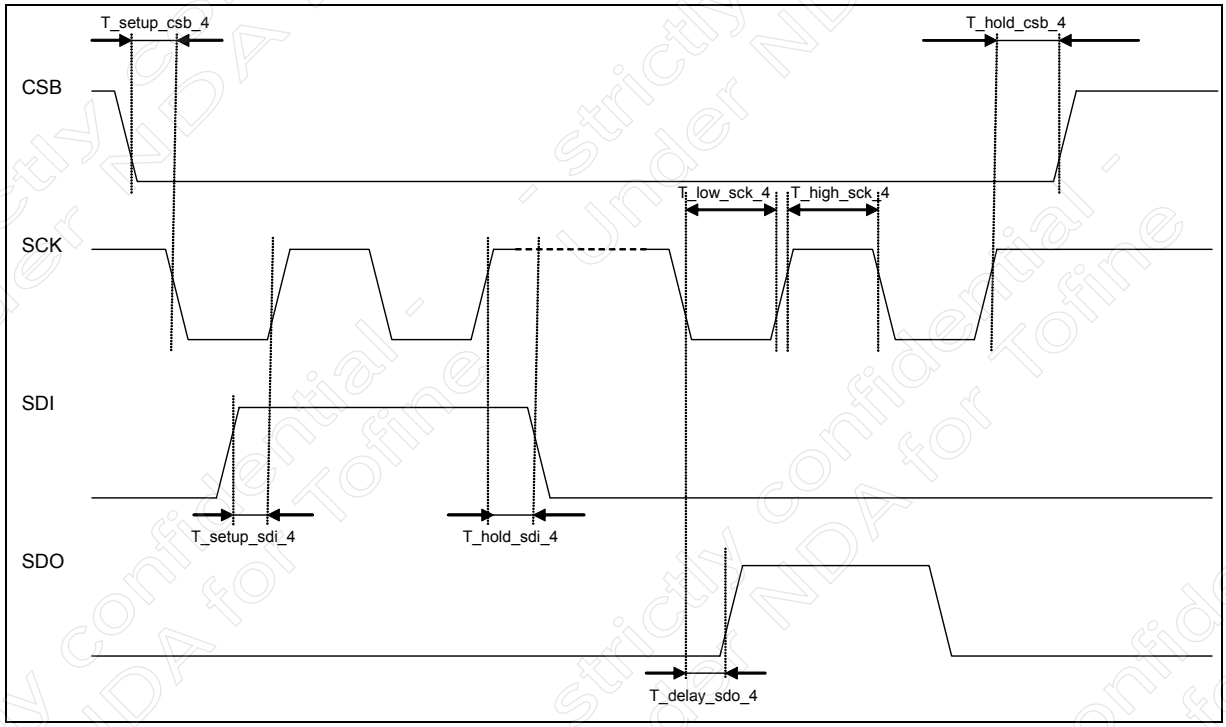


Figure 5: Four wire SPI bit transfer

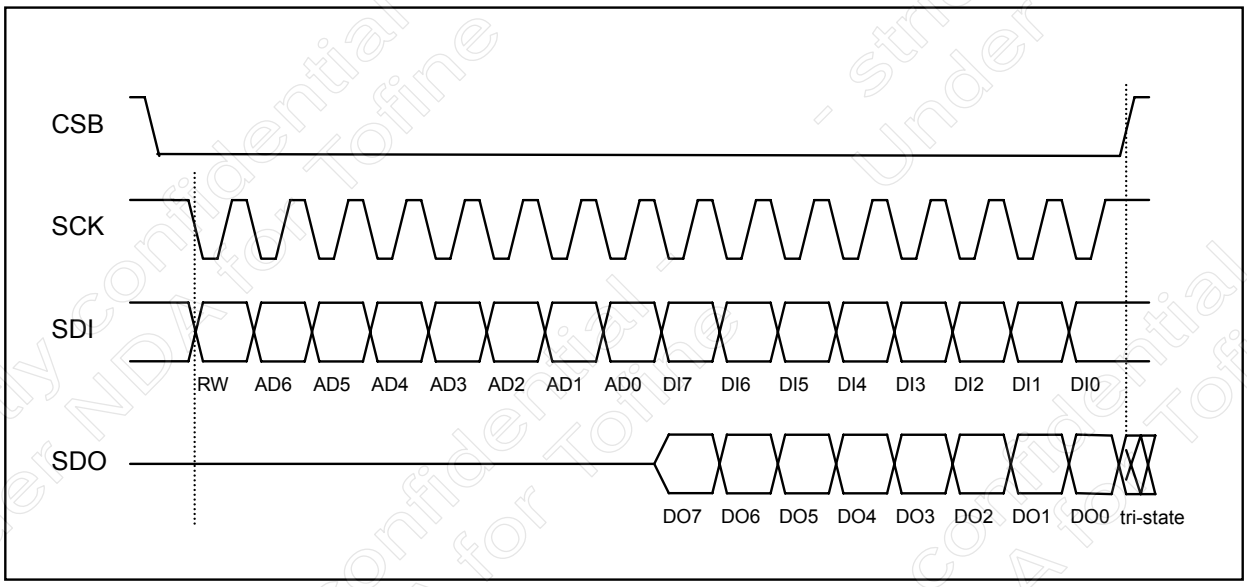


Table 9: Specification of four-wire SPI serial interface

Interface parameters :		**Conditions**	**Min.**	**Typ.**	**Max.**	**unit**
Input - low level	Vil_si	V_{DDIO}=1.62V to 3.6V			0.3*V_{DDIO}	V
Input - high level	Vih_si	V_{DDIO}=1.62V to 3.6V	0.7*V_{DDIO}			V
Output – low level	Vol_SDI	V_{DDIO}=1.8V, iol=3 mA			0.4	V
Output – high level	Voh_SDI	V_{DDIO}=1.8V, ioh=1mA	1.4			V
Load capacitor (on SDO)	Csdo_spi	For 10MHz SPI transfer			25	pF
CSB pull-up resistor	CSB_pull_up	Internal pull-up resistance to V_{DDIO}	70	120	190	kΩ
4-wire SPI timings :						
SPI clock input frequency	Fspi_4				10	MHz
SCK low pulse	Tlow_sck_4		5			ns
SCK high pulse	Thigh_sck_4		5			ns
SDI setup time	Tsetup_sdi_4		5			ns
SDI hold time	Thold_sdi_4		5			ns
SDO output delay	Tdelay_sdo_4				25	ns
CSB setup time	Tsetup_csb_4		5			ns
CSB hold time	Thold_csb_4		5			ns

Figure 6: When write is required, sequences of 2 bytes are necessary: 1 control byte to define the address to be written and the data byte.

	Control byte								Data byte								Control byte								Data byte								
Start	RW	Register adress (16h)								Data register - adress 1Eh								RW	Register adress (0Bh)								Data register - adress 02h								Stop
CSB = 0	0	0	0	1	0	1	1	0	X	X	X	X	X	X	X	X	0	0	0	0	1	0	1	1	X	X	X	X	X	X	X	X	CSB = 1

Figure 7: When read access is required, the sequence consists of 1 control byte to define first address to be read followed by data bytes. Addresses are automatically incremented as long as CSB stays active low.

	Control byte								Data byte								Data byte								Data byte								
Start	RW	Register adress (02h)								Data register - adress 02h								Data register - adress 03h								Data register - adress 04h								Stop
CSB = 0	1	0	0	0	0	0	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	CSB = 1

4.1.2 Three-wire SPI interface

3-wire SPI is not the default serial interface. The customer can easily activate the 3-wire SPI by setting a control bit (SPI4=0). The 3-wire SPI interface uses SCK (serial clock), CSB (chip select, active low) and SDA (serial data in/out). A maximum clock frequency up to 70MHz can be handled.

The protocol data acquisition by the sensor IC occurs at the rising edge of SCK. The output data provided by the sensor IC is synchronized also on the rising edges of SCK. The 3-wire read protocol needs one extra clock cycle between address byte and data output byte.

Figure 8: Timing diagram for three-wire SPI interface (SDI = SDA)

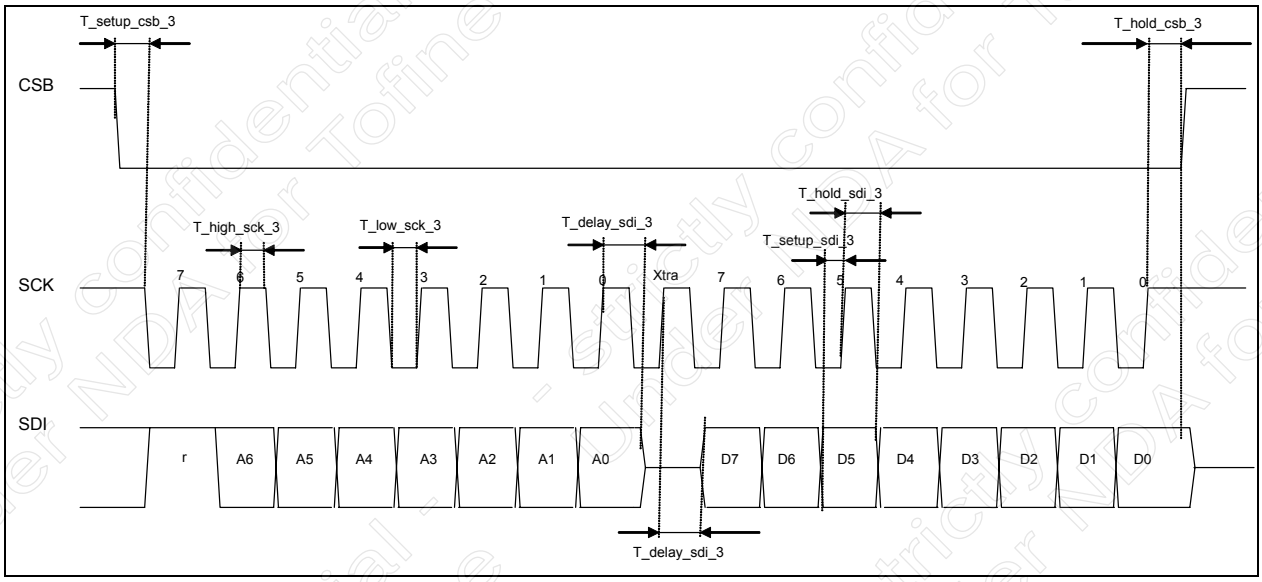


Table 10: Specification of three-wire SPI serial interface

		Conditions	Min.	Typ.	Max.	unit
Input - low level	Vil_si	V_{DDIO}=1.62V to 3.6V			0.3*V_{DDIO}	V
Input - high level	Vih_si	V_{DDIO}=1.62V to 3.6V	0.7*V_{DDIO}			V
Output – low level	Vol_SDI	V_{DDIO}=1.8V, iol=3 mA			0.4	V
Output – high level	Voh_SDI	V_{DDIO}=1.8V, ioh=1mA	1.4			V
CSB pull-up resistor	CSB_pull_up	Internal pull-up resistance to V_{DDIO}	70	120	190	kΩ
Load capacitor (on SDO)	Csdo_spi	for 70MHz SPI transfer			10	pF
3-wire SPI timings :						
SPI clock input frequency	Fspi_3				70	MHz
SCK low pulse	Tlow_sck_3		5			ns
SCK high pulse	Thigh_sck_3		5			ns
SDI setup time	Tsetup_sdi_3		3.8			ns
SDI hold time	Thold_sdi_3		2			ns
SDI output delay	Tdelay_sdi_3	when SDI is an output for read			10.5	ns
CSB setup time	Tsetup_csb_3		5			ns
CSB hold time	Thold_csb_3		5			ns

Figure 9: The three wire SPI write protocol is identical to four wire bus

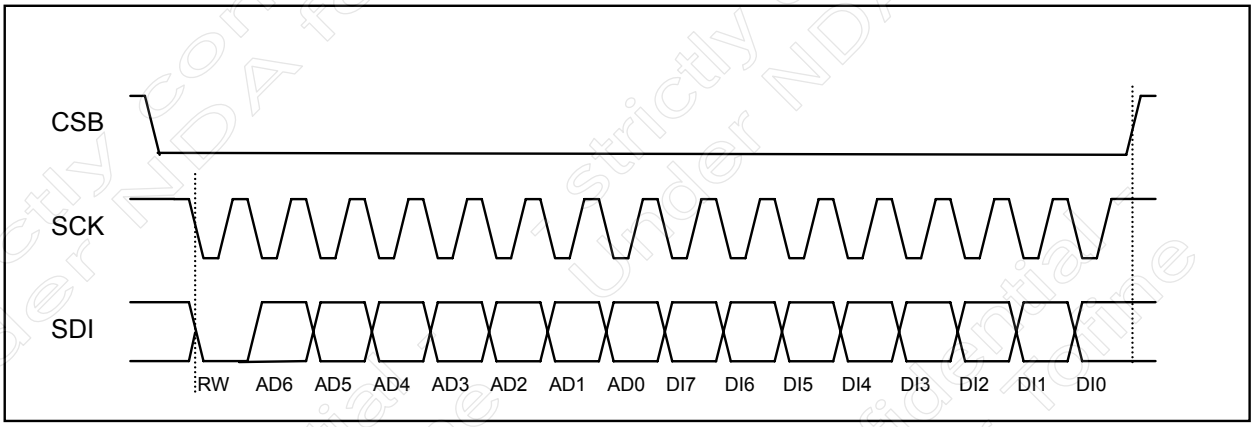
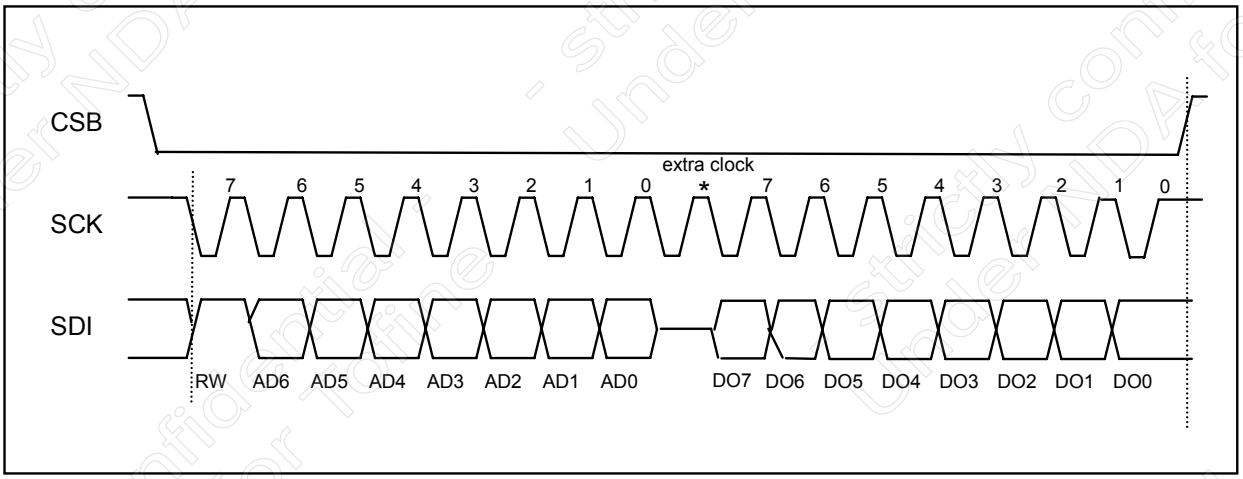


Figure 10: For three wire read protocol one extra clock between address byte and data out byte is required. Output data are changed on SDI (SDI=SDA) by SCK rising edge and should be latched by microprocessor during next SCK rising edge.



4.2 I²C interface

The I²C bus uses SCK (serial clock) and SDA (=SDI, serial data input/output). SDA is bidirectional with open drain; it must be connected externally to V_{DDIO} via a pull up resistor. CSB is not used and must be connected to V_{DDIO}.

Figure 11: Timing diagram for I²C interface (SDI=SDA)

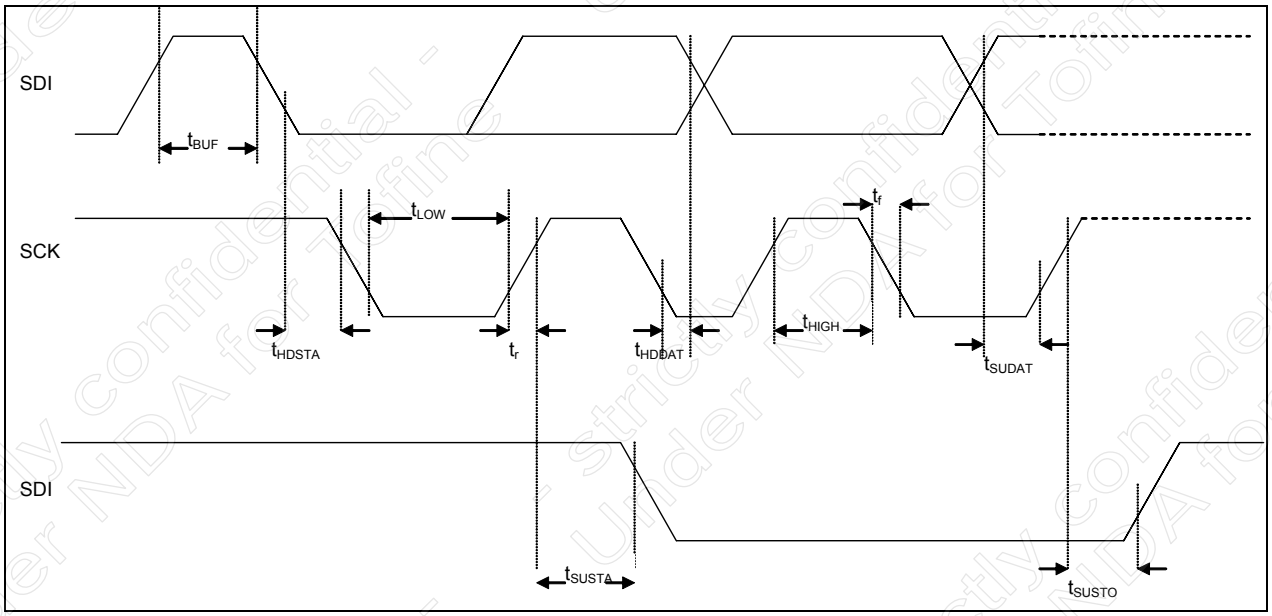


Table 11: Specification of I²C serial interface (SDI=SDA)

Interface parameters :		**Conditions**	**Min.**	**Typ.**	**Max.**	**unit**
Input - low level	Vil_si	V_{DDIO}=1.62V to 3.6V			0.3*V_{DDIO}	V
Input - high level	Vih_si	V_{DDIO}=1.62V to 3.6V	0.7*V_{DDIO}			V
Output – low level	Vol_SDI	V_{DDIO}=1.8V, iol=3 mA			0.4	V
Output – high level	Voh_SDI	V_{DDIO}=1.8V, ioh=1mA	1.4			V
I²C bus load capacitor	Cb	On SDI and SCK			100	pF
I²C timings :						
SCK frequency	FI²C				3.4	MHz
SCK low period	Tlow		160			ns
SCK high period	Thigh		60			ns
SDI setup time	Tsudat		10			ns
SDI hold time	Thddat		10		70	ns
Setup time for a repeated start condition	Tsusta		160			ns
Hold time for a start condition	Thdsta		160			ns
Setup time for a stop condition	Tsusto		160			ns
Time before a new transmission can start	Tbuf		100			ns

Start and stop conditions:

Data transfer begins by a falling edge on SDA when SCK is high (start condition (S) indicated by I²C bus master). Stop condition (P) is a rising edge on SDA when SCK is high (see figure 12).

Bit transfer:
One data bit is transferred during each SCK pulse. Data on SDA line must remain stable during high period of SCK pulse (see figure 13).

Acknowledge:
After start condition each byte of data transfer is followed by an acknowledge bit. The transmitter let the SDA line high (no pull down) and generates a high SCK pulse. If BMA023 has been addressed and data transfer has performed correctly it generates a low SDA level (active pull down). Then SDA line is let free enabling the next transfer (see figure 14).

Figure 12: Timing diagram for I²C start and stop condition (SDI=SDA)

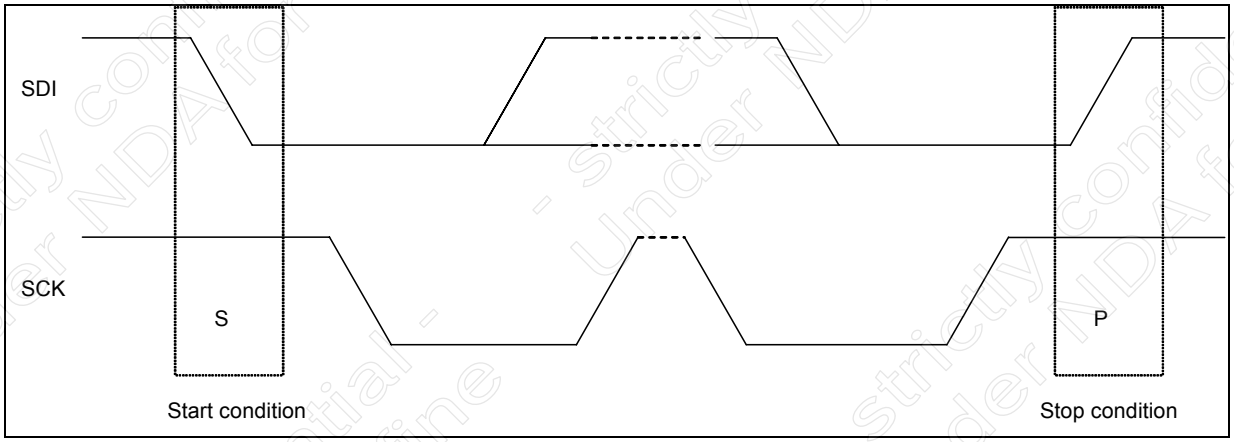


Figure 13: Timing diagram for one bit transfer with I²C interface (SDI=SDA)

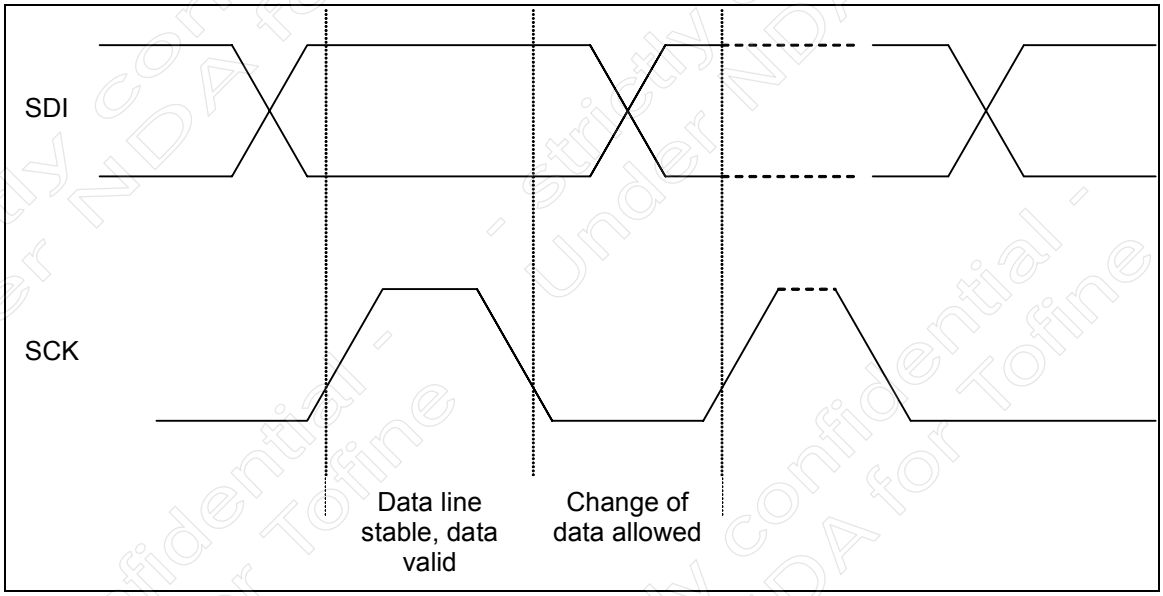
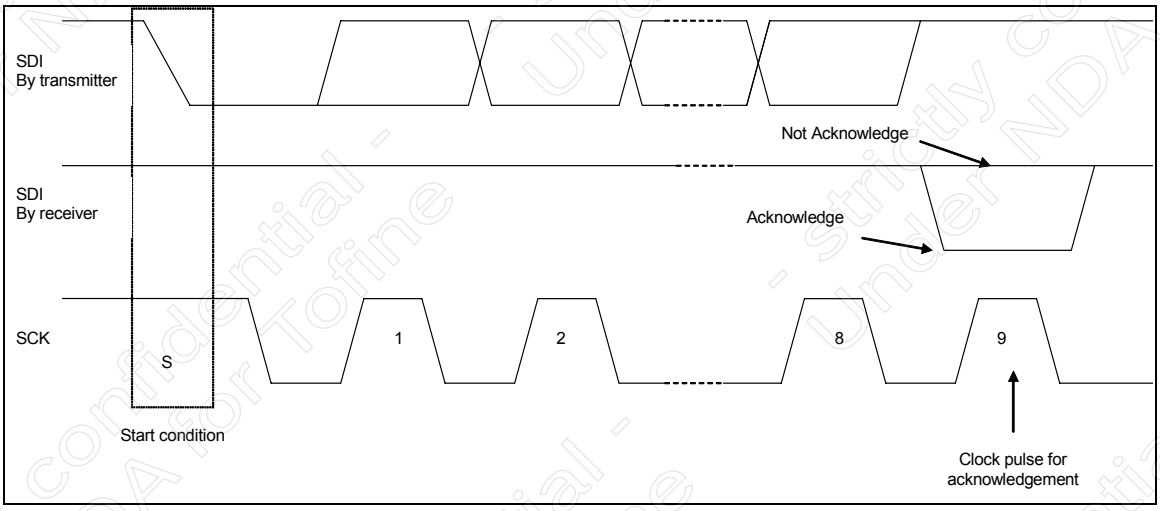


Figure 14: Timing diagram for I²C acknowledgement on SDI (SDI=SDA)



4.2.1 I²C protocol:

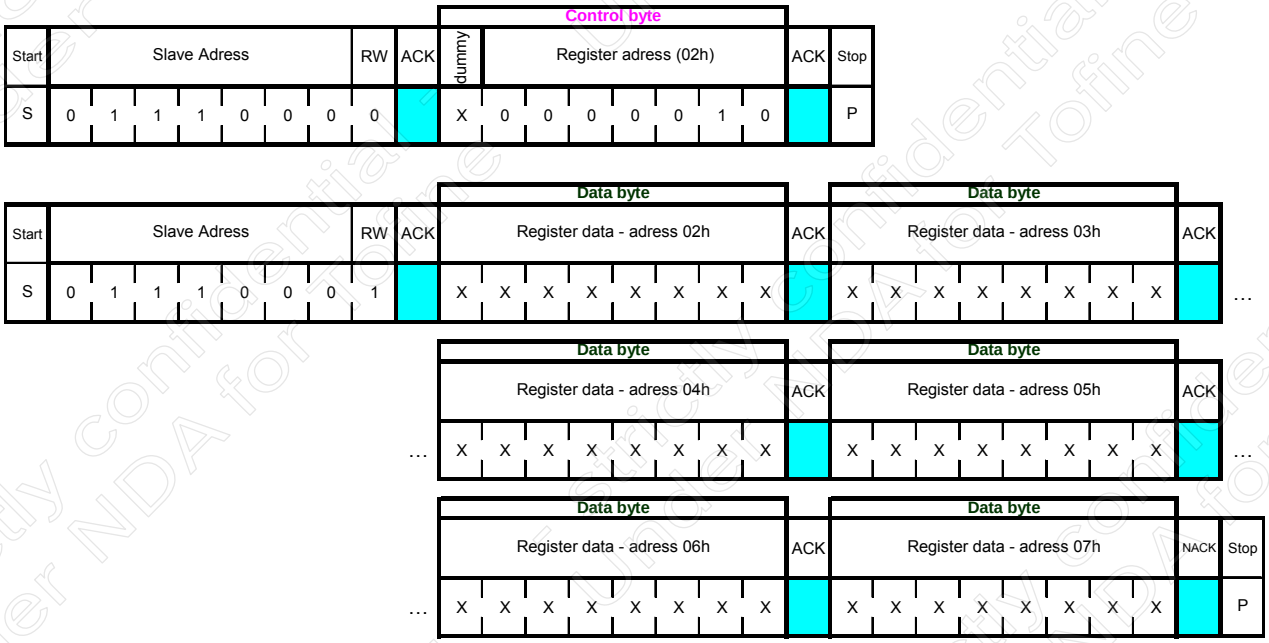
The BMA023 I²C slave address is coded on 7 bits (0111000b=38h) fixed by a metal option. Thus I²C write address is 01110000b (=70h), read address is 01110001b (=71h).
After a start condition, the slave address + RW bit must be send. If the slave address does not match with BMA023 there is no acknowledgement and the following data transfer will not affect the chip. If the slave address corresponds to BMA023 it will acknowledge (pull SDA down during 9th clock pulse) and data transfer is enabled. The 8th bit RW sets the chip in read or write mode, RW=1 for reading, RW=0 for writing.
After slave address and RW bit, the master sends 1 control byte: the 7-bit register address and one dummy bit.

When BMA023 is accessed in write mode, sequences of 2 bytes (= 1 control byte to define which address will be written and 1 data byte) must be sent:

Figure 15: I²C multiple write protocol

To be able to access registers in read mode, first address has to be send in write mode. Then a stop and a start conditions are issued and data bytes are transferred with automatic address increment:

Figure 16: I²C multiple read protocol. Address register is first written to BMA023, the RW=0 (lowest acceleration data located at address 02h). I²C transfer is stopped and restarted with RW=1, address is automatically incremented and the 6 bytes can be sequentially read out.



5. Package

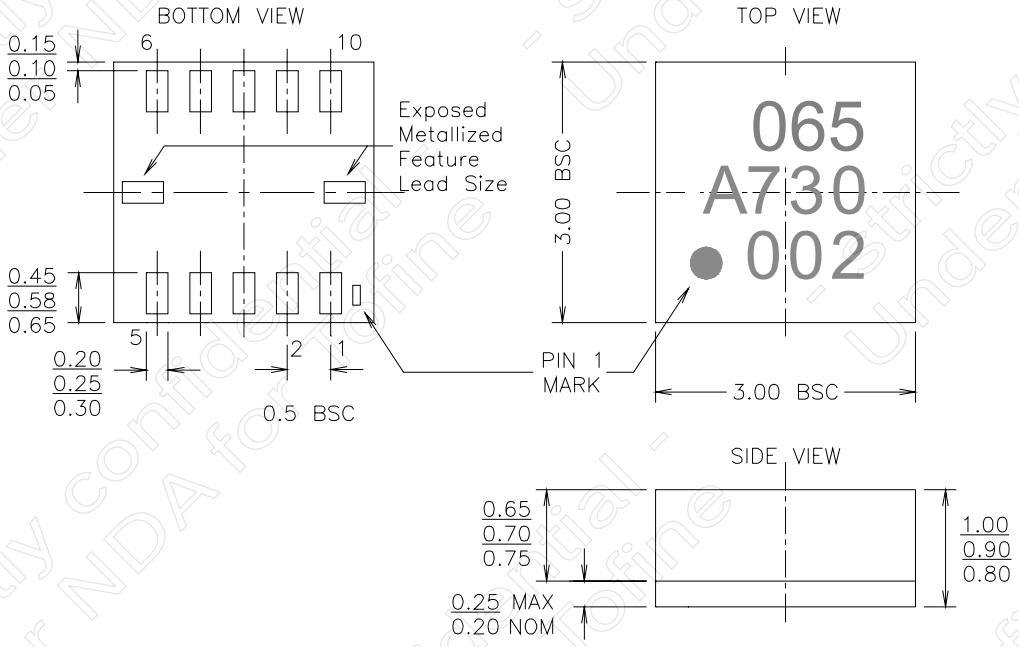
5.1 Outline dimensions

The BMA023 is packaged in a 3mm x 3mm x 0.9mm LGA package following JEDEC MO-229.

Basic outline geometry is based on:

- Mold package footprint 3mm x 3mm (tolerance ±0.1mm)
- Height 0.9mm
- No. of leads 12
  - 8 used for electrical connection
  - 2 not used / reserved
  - 2 additional metal features on front edges without electrical functionality (not available on first engineering samples)
- Lead pitch 0.5mm

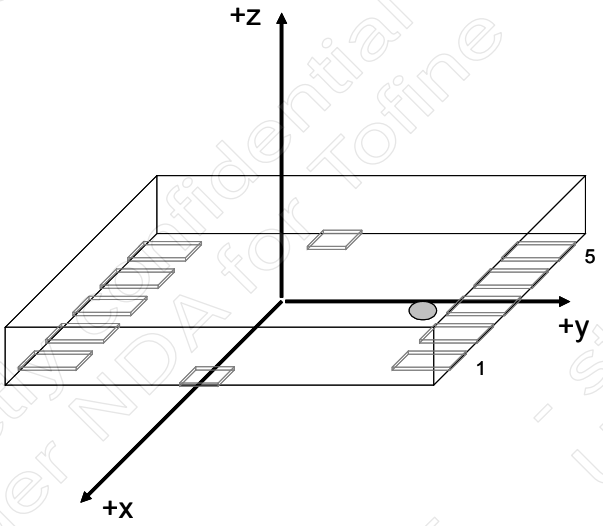
Figure 17: Top, bottom and side views of the 3mm x 3mm x 0.9mm LGA package outline drawing (dimensions in mm)



5.2 Axes orientation

The following diagram describes the orientation of the package with respect to the axes of acceleration measurement.

Figure 18: Axes orientation of the BMA023



5.3 Landing pattern recommendations

As for the design of the landing patterns, the following recommendations can be given:

Figure 19: Landing patterns for the BMA023 relative to the device pins, dimensions are in mm

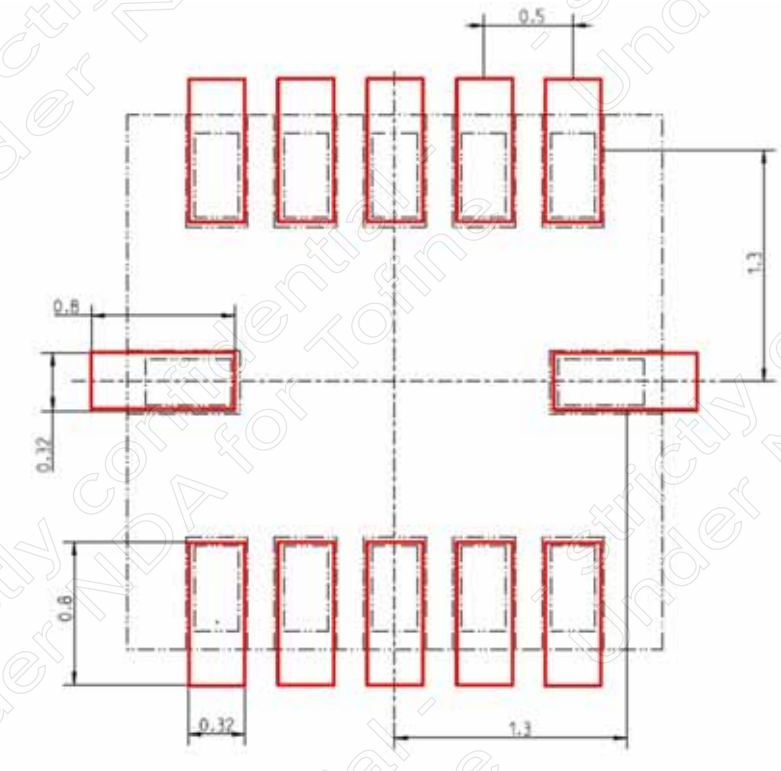
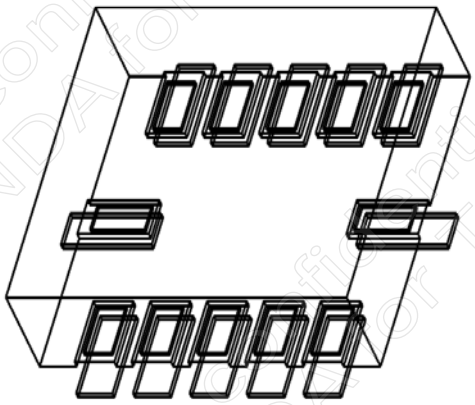


Figure 20: Perspective view of the BMA023 relative to the PCB landing pattern.



5.4 Moisture sensitivity level and soldering

The moisture sensitivity level (MSL) of the BMA023 sensor IC corresponds to JEDEC Level 1, see also

- IPC/JEDEC J-STD-020C "Joint Industry Standard: Moisture/Reflow Sensitivity Classification for Non-hermetic Solid State Surface Mount Devices"
- IPC/JEDEC J-STD-033A "Joint Industry Standard: Handling, Packing, Shipping and Use of Moisture/Reflow Sensitive Surface Mount Devices".

The sensor IC fulfils the lead-free soldering requirements of the above-mentioned IPC/JEDEC standard, i.e. reflow soldering with a peak temperature up to 260°C.

5.5 RoHS compliancy

The BMA023 sensor IC meets the requirements of the EC restriction of hazardous substances (RoHS) directive, see also
"Directive 2002/95/EC of the European Parliament and of the Council of 27 January 2003 on the restriction of the use of certain hazardous substances in electrical and electronic equipment".

5.6 Halogen content

Results of chemical analysis show that the BMA023 contains less than 900ppm (by weight) of Fluorine, Chlorine, Iodine and Bromine (i.e. <900ppm per each substance).

Therefore the BMA023 can be regarded as halogen-free.

For more details on the analysis results please contact your Bosch Sensortec representative.

5.7 Note on internal package structure

Within the scope of Bosch Sensortec's ambition to improve its products and secure the product supply while in mass production, Bosch Sensortec qualifies additional sources for the LGA package of the BMA023.

While Bosch Sensortec took care that all of the technical package parameters as described above are 100% identical for both sources, there can be differences in the chemical analysis and internal structural between the different package sources.

However, as secured by the extensive product qualification processes of Bosch Sensortec, this has no impact to the usage or to the quality of the BMA023 product.

6. Pin-out out and connection diagrams

Figure 21: Pin-out of the BMA023 (bottom view);

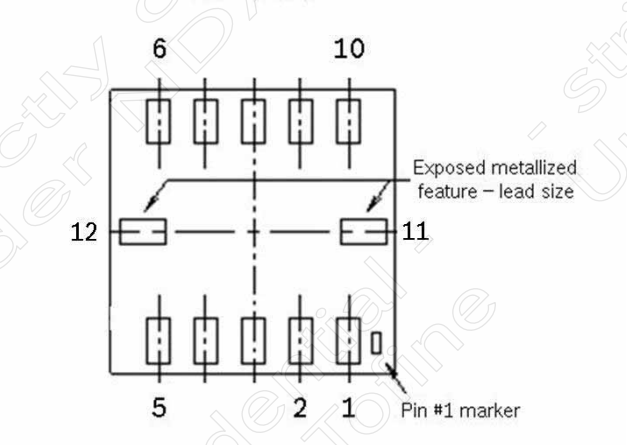


Table 12: Pin-out description of the BMA023

Pin No	Name	Type	Description	Connect to (in SPI 4w)	Connect to (in SPI 3w)	Connect to (in I²C)	Stand alone (without µC)
1	reserved		Do not connect	NC	NC	NC	NC
2	V_{DD}	Power	Analogue power supply	V_{DD}	V_{DD}	V_{DD}	V_{DD}
3	GND	Power	Ground	GND	GND	GND	GND
4	INT	Output	Interrupt	INT / NC	INT / NC	INT / NC	INT
5	CSB	Input	Chip select	CSB	CSB	V_{DDIO}	V_{DD}
6	SCK	Input	Serial clock	SCK	SCK	SCK	GND
7	SDO	Output	Serial data out	SDO	GND	GND	GND
8	SDI	Input / Output	Serial data in / out	SDI	SDA	SDA	GND
9	V_{DDIO}	Power	Digital interface power supply	V_{DDIO}	V_{DDIO}	V_{DDIO}	V_{DD}
10	reserved		Do not connect	NC	NC	NC	NC
11	reserved		Do not connect	NC	NC	NC	NC
12	reserved		Do not connect	NC	NC	NC	NC

Recommendation for decoupling: between GND and VDD (pin 1 or 2) a 22nF capacitor and between GND and IOVDD (pin 9) a 100nF capacitor should be connected.

Figure 22: Connection diagram for use with 4-wire SPI interface

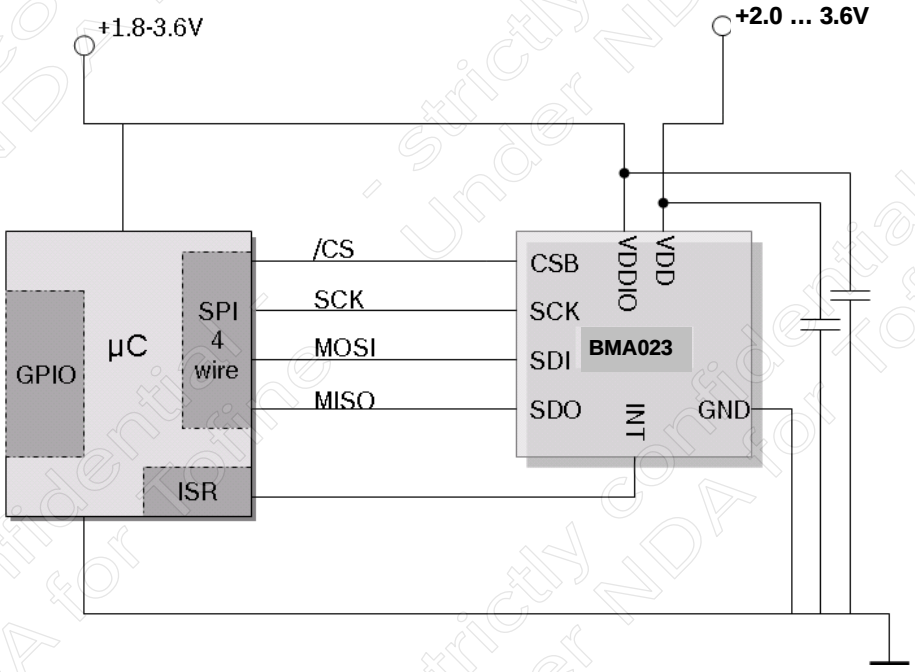


Figure 23: Connection diagram for use with 3-wire SPI interface

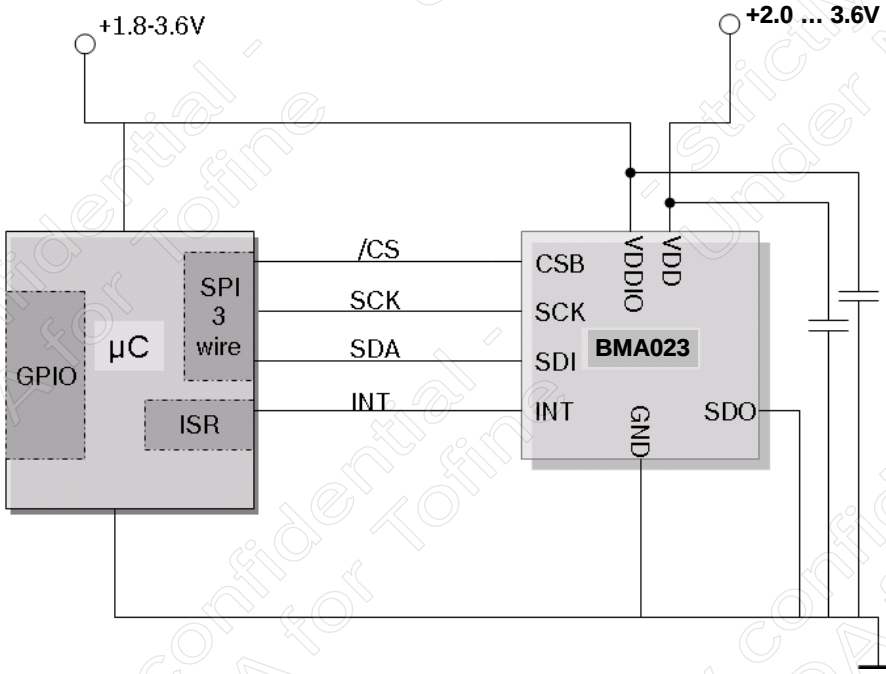
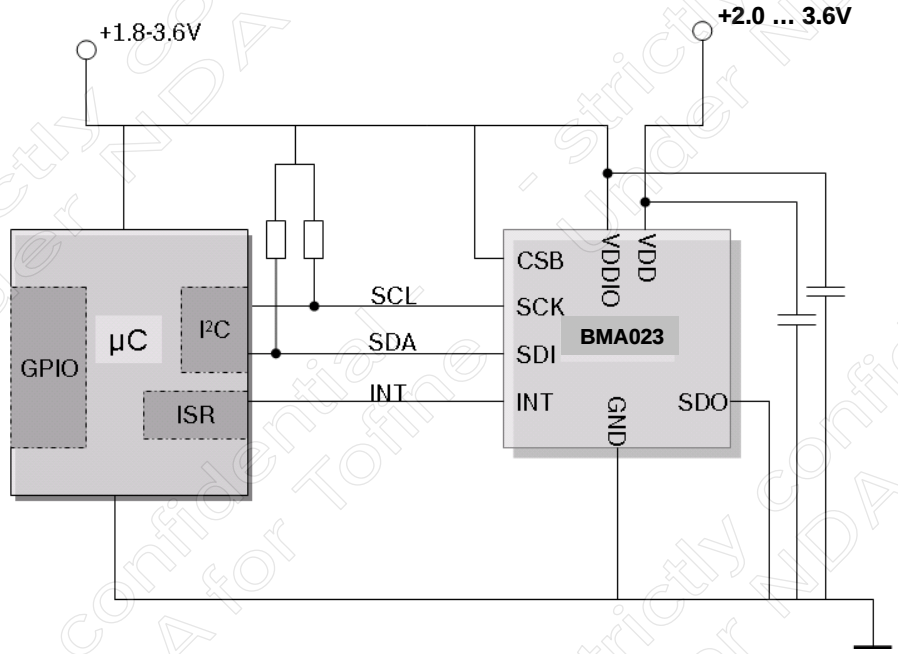


Figure 24: Connection diagram for use with I²C interface



7. Operation modes

7.1 Normal operational mode

In normal operational mode the sensor IC can be addressed via digital interface. Data and status registers can be read out and control registers can be read and changed. In parallel to normal operation the user has the option to activate several internal logic paths and set criteria to trigger the interrupt pin. BMA023 is designed to enable low current consumption of 200µA in operational mode.
A self-test procedure can be started in operational mode for testing of the complete signal evaluation path including the micro-machined sensor IC structure, the evaluation ASIC and the physical connection to the host system.

7.2 Sleep mode

Sleep mode is activated by setting a control bit. In sleep mode no communication with the sensor IC is possible – all read and write commands are forbidden. The recommended command to switch to operational mode is the wake-up call.

Wake-up time from sleep to operational mode is 1ms.

In case of a soft-reset, it is recommended to do this reset after having switched from sleep to operational mode. In this case the total typical wake-up and reset time at maximum bandwidth is "switching to operational mode = 1ms" and "time after soft reset until acceleration data is available = 1.3ms", i.e. 2.3ms in total.
In case a soft-reset is activated during sleep mode it can take up to 30msec. until normal operation has resumed.

The current consumption in sleep mode is 1µA.

7.3 Wake-up mode

In general BMA023 is attributed to low power applications and can contribute to the system power management.

- Current consumption 200µA operational
- Current consumption 1µA sleep mode
- Wake-up time 1ms
- Start-up time 3ms
- Data ready indicator to reduce unnecessary interface communication
- Wake-up mode to trigger a system wake-up (interrupt output to master) when motion detected
- Current consumption in wake-up mode

The BMA023 provides the possibility to wake up a system master when specific acceleration values are detected. Therefore the BMA023 stays in an ultra low power mode and periodically evaluates the acceleration data with respect to interrupt criteria defined by the user. An interrupt

output can be generated and trigger the system master. The wake-up mode is used for ultra-low power applications where inertial factors can be an indicator to change the activity mode of the system.

The following table shows values calculated for the average current consumption during the wake-up mode of the BMA023. The power consumption in wake-up mode is dependent on the duration of the interrupt algorithm (number of data acquisitions) and the bandwidth (for more details on setting of the bandwidth please refer to chapter 3.1.3).

Table 13: Average current consumption in self wake-up mode using high-g or low-g interrupt

	Current consumption during BMA023 wake-up mode [µA] (depending on bandwidth, calculated using typical values)						
Pause [ms]	(@ 1,500Hz)	(@ 750Hz)	(@375Hz)	(@190Hz)	(@100Hz)	(@50Hz)	(@25Hz)
20	16.3	21,8	31.8	48.4	71.6	102.9	134.7
80	5.1	6.6	9.7	15.4	25.0	42.3	68.4
360	1.9	2.3	3.0	4.4	6.9	12.0	21.3
2,560	1.1	1.2	1.3	1.5	1.9	2.6	4.1

Durations of the pause values can vary for about ±30% due to the accuracy of the ultra-low-power oscillator implemented within the sensor.

For estimating the typical current consumption in wake-up mode the following formula can be applied:

$$i\_self\_wake\_up = (i\_DD \cdot t\_active + i\_DDsbm \cdot wake\text{-}up\text{-}pause) / (t\_active + wake\text{-}up\text{-}pause)$$

With the approximation:

$$t\_active = 1ms + 0.333ms \cdot (4 \cdot 750 / bandwidth) + 0.333ms \cdot (1500 / bandwidth) \cdot n$$

With the following parameters:

i_DD	Current in normal mode
i_DDsm	Current in sleep mode
wake_up_pause	Setting of wake-up pause
n	number of data points in any-motion logic (n=0 for high-g threshold and low-g threshold interrupt, n=3 for any-motion logic)
Bandwidth	Setting of bandwidth (750-25 Hz), for 1500Hz $t\_active = 1ms + 0.333ms \cdot (1500/bandwidth) \cdot n$

So, the relevant parameters for power consumption in self-wake up mode are:

- the current consumption in normal mode
- the current consumption in sleep mode
- the self-wake up pause duration
- the bandwidth (ie. length of digital filter to be filled for one data point)
- the interrupt criteria (determines the duration of normal operation):
    - high-g and low-g criteria (ie. acquisition of one data point)
    - any-motion criterion (ie. four data points)

As some of these parameters have certain deviations from the typical value results of various example Monte Carlo Simulations on the current consumption are shown in figures 25, 26 and 27.

The graphs provide an indication on the expected current consumption for different settings.

Figure 25: Estimation of current consumption using Monte Carlo simulation, example #1: bandwidth 750Hz, 2560ms wake-up setting, any-motion interrupt

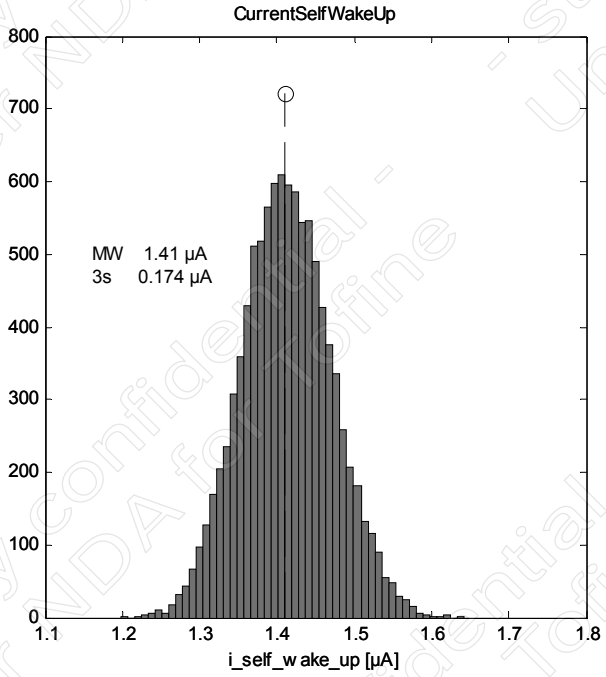


Figure 26: Estimation of current consumption using Monte Carlo simulation, example #2: bandwidth 47Hz, 2560ms wake-up setting, any-motion interrupt

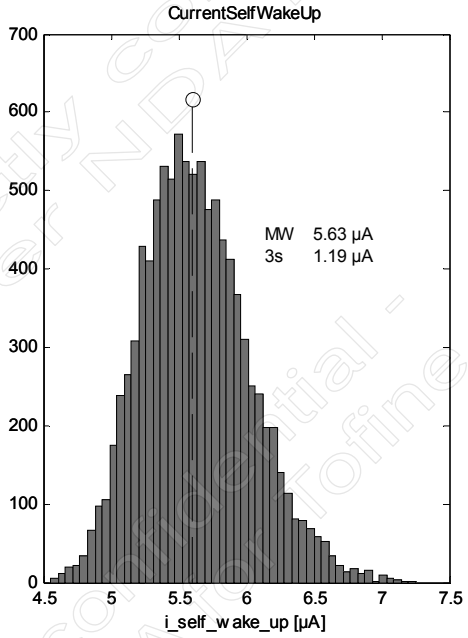
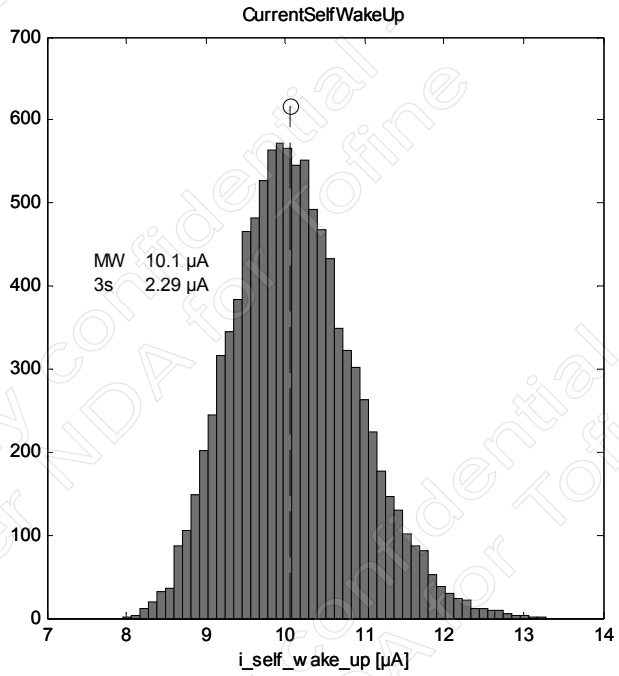


Figure 27: Estimation of current consumption using Monte Carlo simulation, example #3: Bandwidth 23Hz, 2560ms wake-up setting, any-motion interrupt



8. Data conversion

8.1 Acceleration data

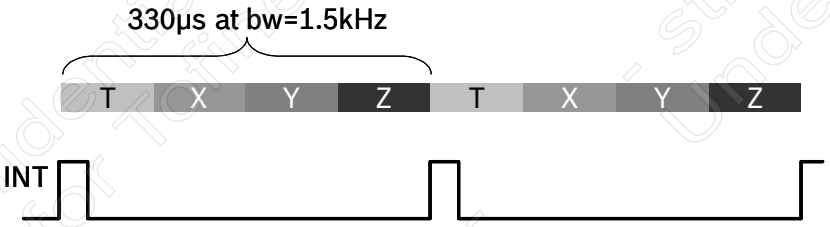
Acceleration data are converted by a 10bit ADC. The description of the digital signal is "2's complement". The 10 bit data are available as LSB (at lower register address) and MSB. It is possible to read out MSB only (8 bit) and LSB/MSB (16 bits with 10 data bits and 1 data ready bit) while LSB- and MSB-data are closely linked to avoid unintentional LSB/MSB mixing when read out and data conversion overlap accidentally (section 3.5.1).

The update rate of data registers is 3 kHz, independent of the digital filter. The acceleration data is filtered by a second order analog filter at 1.5 kHz. Additionally the data can be processed by digital averaging filters (moving average) to reduce the noise level (750Hz – 25Hz).
The transfer function of the mechanical element is designed to avoid resonance effects at frequencies below the bandwidth of the ASIC.

The availability of new data can be checked in two ways:

- Bit 0 from the LSB data registers is an indicator whether the data have already been read out or the data are new (Bit0=1) (section 3.5.2).
- The interrupt pin can be configured to indicate new data availability (not possible in parallel to internal interrupt logic). The synchronization of data acquisition and data read out enables the customer to avoid unnecessary interface traffic in order to reduce the system power consumption and the crosstalk between interface communication and data conversion. For a detailed explanation see Figure 3. (section 3.2.10)

Figure 28: Explanation of data ready interrupt: For a bandwidth of e.g. 1.5kHz the data refresh cycle takes 330µs to update all data registers. After the final conversion of z-axis the INT pad will be set high. New data can be read out via interface. The interrupt resets automatically after read out.



9. Internal logic functions

The sensor IC can inform the host system about specific conditions (e.g. new data ready flag or acceleration thresholds passed) by setting an interrupt pin high even if interface communication is not taking place. This feature can be used as “wake-up” indicator or “data ready flag” for instance.
The interrupt performance can be programmed by means of control bits. Thus the criteria to identify a special event can be tailored to a customer’s application and the sensor IC output can be defined specifically.

9.1 High-g logic

For indicating high-g events an upper threshold can be programmed. This logic can also be activated by a control bit. Threshold, duration and reset behaviour can be programmed.
See also section 3.2.8.

9.2 Any motion detection

The “any motion algorithm” can be used to detect changes of the acceleration. Thus it provides a relative evaluation of the acceleration signals. The criterion is kind of a gradient threshold of the acceleration over time. Thus one can distinguish between fast events with strong inertial dynamic (e.g. shock), instant changes of force balance (e.g. drop, tumbling) and even slight changes (e.g. touch of a mobile device).
Due to a high bandwidth and a fast response MEMS device the BMA023 is capable to detect shock situations. The “any motion interrupt” or a high-g criterion setting can be used to give a shock alert. The phase shift between onset of mechanical shock and interrupt output is defined by the mechanical transfer function of the chassis and internal mounting interfaces (e.g. PDA shell) and the data output rate of the sensor IC (currently 330µs, 100µs under consideration).

See also section 3.2.9.

9.3 Alert mode

Using the BMA023 it is possible to combine the “any motion criterion” with low-g and high-g interrupt logic to improve the reaction time.

See also sections 3.2.9 and 3.4.2.

10. Legal disclaimer

10.1 Engineering samples

Engineering Samples are marked with an asterisk (*) or (e). Samples may vary from the valid technical specifications of the product series contained in this data sheet. They are therefore not intended or fit for resale to third parties or for use in end products. Their sole purpose is internal client testing. The testing of an engineering sample may in no way replace the testing of a product series. Bosch Sensortec assumes no liability for the use of engineering samples. The Purchaser shall indemnify Bosch Sensortec from all claims arising from the use of engineering samples.

10.2 Product use

Bosch Sensortec products are developed for the consumer goods industry. They may only be used within the parameters of this product data sheet. They are not fit for use in life-sustaining or security sensitive systems. Security sensitive systems are those for which a malfunction is expected to lead to bodily harm or significant property damage. In addition, they are not fit for use in products which interact with motor vehicle systems.

The resale and/or use of products are at the purchaser's own risk and his own responsibility. The examination of fitness for the intended use is the sole responsibility of the Purchaser.

The purchaser shall indemnify Bosch Sensortec from all third party claims arising from any product use not covered by the parameters of this product data sheet or not approved by Bosch Sensortec and reimburse Bosch Sensortec for all costs in connection with such claims.

The purchaser must monitor the market for the purchased products, particularly with regard to product safety, and inform Bosch Sensortec without delay of all security relevant incidents.

10.3 Application examples and hints

With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Bosch Sensortec hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights or copyrights of any third party. The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. They are provided for illustrative purposes only and no evaluation regarding infringement of intellectual property rights or copyrights or regarding functionality, performance or error has been made.

11. Document history and modification

Rev. No	Chapter	Description of modification/changes	Date
1.0		Document creation	23-Feb-2009
1.1	1	Added comment on 3-sigma values	07-Apr-2009

Bosch Sensortec GmbH
Gerhard-Kindler-Strasse 8
72770 Reutlingen / Germany

contact@bosch-sensortec.com
www.bosch-sensortec.com

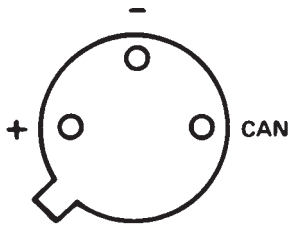
Modifications reserved | Printed in Germany
Specifications subject to change without notice
Document number: BST-BMA023-DS000-01
Version_1.1_042009

Two-Terminal IC Temperature Transducer

AD590

FEATURES
Linear Current Output: 1 μA/K
Wide Range: –55°C to +150°C
Probe Compatible Ceramic Sensor Package
Two Terminal Device: Voltage In/Current Out
Laser Trimmed to ±0.5°C Calibration Accuracy (AD590M)
Excellent Linearity: ±0.3°C Over Full Range (AD590M)
Wide Power Supply Range: +4 V to +30 V
Sensor Isolation from Case
Low Cost

PIN DESIGNATIONS



BOTTOM VIEW

PRODUCT DESCRIPTION

The AD590 is a two-terminal integrated circuit temperature transducer that produces an output current proportional to absolute temperature. For supply voltages between +4 V and +30 V the device acts as a high impedance, constant current regulator passing 1 μA/K. Laser trimming of the chip's thin-film resistors is used to calibrate the device to 298.2 μA output at 298.2K (+25°C).

The AD590 should be used in any temperature sensing application below +150°C in which conventional electrical temperature sensors are currently employed. The inherent low cost of a monolithic integrated circuit combined with the elimination of support circuitry makes the AD590 an attractive alternative for many temperature measurement situations. Linearization circuitry, precision voltage amplifiers, resistance measuring circuitry and cold junction compensation are not needed in applying the AD590.

In addition to temperature measurement, applications include temperature compensation or correction of discrete components, biasing proportional to absolute temperature, flow rate measurement, level detection of fluids and anemometry. The AD590 is available in chip form making, it suitable for hybrid circuits and fast temperature measurements in protected environments.

The AD590 is particularly useful in remote sensing applications. The device is insensitive to voltage drops over long lines due to its high impedance current output. Any well insulated twisted pair is sufficient for operation hundreds of feet from the receiving circuitry. The output characteristics also make the AD590 easy to multiplex: the current can be switched by a CMOS multiplexer or the supply voltage can be switched by a logic gate output.

PRODUCT HIGHLIGHTS

1. The AD590 is a calibrated two terminal temperature sensor requiring only a dc voltage supply (+4 V to +30 V). Costly transmitters, filters, lead wire compensation and linearization circuits are all unnecessary in applying the device.
2. State-of-the-art laser trimming at the wafer level in conjunction with extensive final testing ensures that AD590 units are easily interchangeable.
3. Superior interface rejection results from the output being a current rather than a voltage. In addition, power requirements are low (1.5 mWs @ 5 V @ +25°C.) These features make the AD590 easy to apply as a remote sensor.
4. The high output impedance (>10 MΩ) provides excellent rejection of supply voltage drift and ripple. For instance, changing the power supply from 5 V to 10 V results in only a 1 μA maximum current change, or 1°C equivalent error.
5. The AD590 is electrically durable: it will withstand a forward voltage up to 44 V and a reverse voltage of 20 V. Hence, supply irregularities or pin reversal will not damage the device.

REV. B

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 617/329-4700 World Wide Web Site: http://www.analog.com
Fax: 617/326-8703

AD590–SPECIFICATIONS (@ +25°C and V_S = +5 V unless otherwise noted)

Model	AD590J			AD590K			Units
	Min	Typ	Max	Min	Typ	Max	
ABSOLUTE MAXIMUM RATINGS							
Forward Voltage (E+ or E–)			+44			+44	Volts
Reverse Voltage (E+ to E–)			–20			–20	Volts
Breakdown Voltage (Case E+ or E–)			±200			±200	Volts
Rated Performance Temperature Range[1]	–55		+150	–55		+150	°C
Storage Temperature Range[1]	–65		+155	–65		+155	°C
Lead Temperature (Soldering, 10 sec)			+300			+300	°C
POWER SUPPLY							
Operating Voltage Range	+4		+30	+4		+30	Volts
OUTPUT							
Nominal Current Output @ +25°C (298.2K)		298.2			298.2		µA
Nominal Temperature Coefficient		1			1		µA/K
Calibration Error @ +25°C			**±5.0**			**±2.5**	°C
Absolute Error (Over Rated Performance Temperature Range)							
Without External Calibration Adjustment			**±10**			**±5.5**	°C
With +25°C Calibration Error Set to Zero			**±3.0**			**±2.0**	°C
Nonlinearity			**±1.5**			**±0.8**	°C
Repeatability[2]			±0.1			±0.1	°C
Long-Term Drift[3]			±0.1			±0.1	°C
Current Noise		40			40		pA/$\sqrt{\text{Hz}}$
Power Supply Rejection							
+4 V ≤ V_S ≤ +5 V		0.5			0.5		µA/V
+5 V ≤ V_S ≤ +15 V		0.2			0.2		µV/V
+15 V ≤ V_S ≤ +30 V		0.1			0.1		µA/V
Case Isolation to Either Lead		10^{10}			10^{10}		Ω
Effective Shunt Capacitance		100			100		pF
Electrical Turn-On Time		20			20		µs
Reverse Bias Leakage Current[4]							
(Reverse Voltage = 10 V)		10			10		pA
PACKAGE OPTIONS							
TO-52 (H-03A)		AD590JH			AD590KH		
Flatpack (F-2A)		AD590JF			AD590KF		

NOTES

[1]The AD590 has been used at –100°C and +200°C for short periods of measurement with no physical damage to the device. However, the absolute errors specified apply to only the rated performance temperature range.

[2]Maximum deviation between +25°C readings after temperature cycling between –55°C and +150°C; guaranteed not tested.

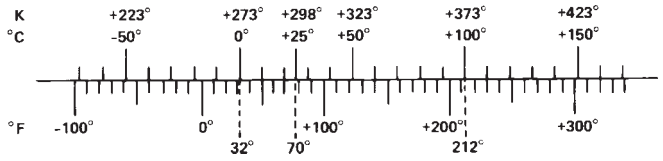
[3]Conditions: constant +5 V, constant +125°C; guaranteed, not tested.

[4]Leakage current doubles every 10°C.

Specifications subject to change without notice.

Specifications shown in **boldface** are tested on all production units at final electrical test. Results from those tests are used to calculate outgoing quality levels. All min and max specifications are guaranteed, although only those shown in **boldface** are tested on all production units.

Model	AD590L Min	AD590L Typ	AD590L Max	AD590M Min	AD590M Typ	AD590M Max	Units
ABSOLUTE MAXIMUM RATINGS							
Forward Voltage (E+ or E–)			+44			+44	Volts
Reverse Voltage (E+ to E–)			–20			–20	Volts
Breakdown Voltage (Case to E+ or E–)			±200			±200	Volts
Rated Performance Temperature Range[1]	–55		+150	–55		+150	°C
Storage Temperature Range[1]	–65		+155	–65		+155	°C
Lead Temperature (Soldering, 10 sec)			+300			+300	°C
POWER SUPPLY							
Operating Voltage Range	+4		+30	+4		+30	Volts
OUTPUT							
Nominal Current Output @ +25°C (298.2K)		298.2			298.2		µA
Nominal Temperature Coefficient		1			1		µA/K
Calibration Error @ +25°C			**±1.0**			**±0.5**	°C
Absolute Error (Over Rated Performance Temperature Range)							
Without External Calibration Adjustment			**±3.0**			**±1.7**	°C
With ±25°C Calibration Error Set to Zero			**±1.6**			**±1.0**	°C
Nonlinearity			**±0.4**			**±0.3**	°C
Repeatability[2]			±0.1			±0.1	°C
Long-Term Drift[3]			±0.1			±0.1	°C
Current Noise		40			40		pA/$\sqrt{\text{Hz}}$
Power Supply Rejection							
$+4\ V \le V_S \le +5\ V$		0.5			0.5		µA/V
$+5\ V \le V_S \le +15\ V$		0.2			0.2		µA/V
$+15\ V \le V_S \le +30\ V$		0.1			0.1		µA/V
Case Isolation to Either Lead		10^{10}			10^{10}		Ω
Effective Shunt Capacitance		100			100		pF
Electrical Turn-On Time		20			20		µs
Reverse Bias Leakage Current[4] (Reverse Voltage = 10 V)		10			10		pA
PACKAGE OPTIONS							
TO-52 (H-03A)		AD590LH			AD590MH		
Flatpack (F-2A)		AD590LF			AD590MF		



TEMPERATURE SCALE CONVERSION EQUATIONS

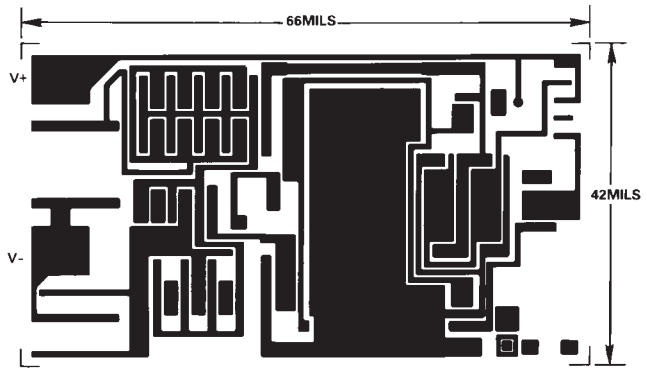
$$°C = \frac{5}{9}(°F - 32) \quad K = °C + 273.15$$

$$°F = \frac{9}{5}°C + 32 \qquad °R = °F + 459.7$$

The 590H has 60 µ inches of gold plating on its Kovar leads and Kovar header. A resistance welder is used to seal the nickel cap to the header. The AD590 chip is eutectically mounted to the header and ultrasonically bonded to with 1 MIL aluminum wire. Kovar composition: 53% iron nominal; 29% ±1% nickel; 17% ± 1% cobalt; 0.65% manganese max; 0.20% silicon max; 0.10% aluminum max; 0.10% magnesium max; 0.10% zirconium max; 0.10% titanium max; 0.06% carbon max.

The 590F is a ceramic package with gold plating on its Kovar leads, Kovar lid, and chip cavity. Solder of 80/20 Au/Sn composition is used for the 1.5 mil thick solder ring under the lid. The chip cavity has a nickel underlay between the metalization and the gold plating. The AD590 chip is eutectically mounted in the chip cavity at 410°C and ultrasonically bonded to with 1 mil aluminum wire. Note that the chip is in direct contact with the ceramic base, not the metal lid. When using the AD590 in die form, the chip substrate must be kept electrically isolated, (floating), for correct circuit operation.

METALIZATION DIAGRAM



THE AD590 IS AVAILABLE IN LASER-TRIMMED CHIP FORM; CONSULT THE CHIP CATALOG FOR DETAILS.

CIRCUIT DESCRIPTION[1]

The AD590 uses a fundamental property of the silicon transistors from which it is made to realize its temperature proportional characteristic: if two identical transistors are operated at a constant ratio of collector current densities, r, then the difference in their base-emitter voltage will be (kT/q)(In r). Since both k, Boltzman's constant and q, the charge of an electron, are constant, the resulting voltage is directly proportional to absolute temperature (PTAT).

In the AD590, this PTAT voltage is converted to a PTAT current by low temperature coefficient thin-film resistors. The total current of the device is then forced to be a multiple of this PTAT current. Referring to Figure 1, the schematic diagram of the AD590, Q8 and Q11 are the transistors that produce the PTAT voltage. R5 and R6 convert the voltage to current. Q10, whose collector current tracks the colletor currents in Q9 and Q11, supplies all the bias and substrate leakage current for the rest of the circuit, forcing the total current to be PTAT. R5 and R6 are laser trimmed on the wafer to calibrate the device at +25°C.

Figure 2 shows the typical V–I characteristic of the circuit at +25°C and the temperature extremes.

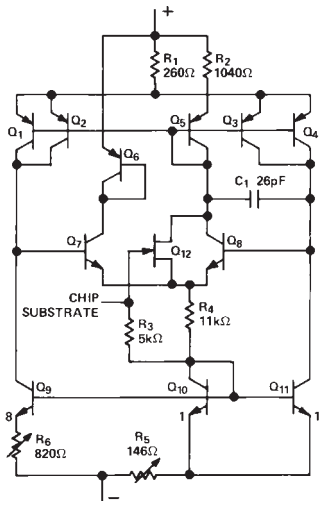


Figure 1. Schematic Diagram

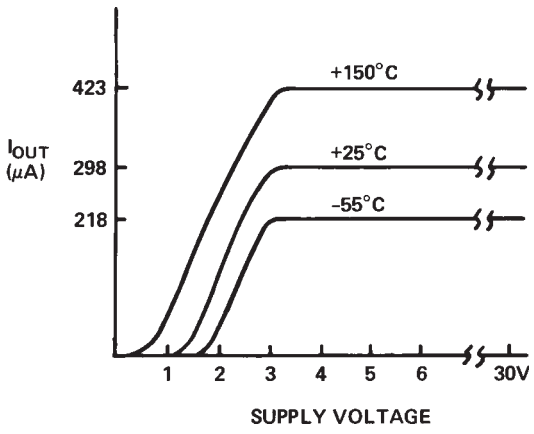


Figure 2. V–I Plot

[1]For a more detailed circuit description see M.P. Timko, "A Two-Terminal IC Temperature Transducer," IEEE J. Solid State Circuits, Vol. SC-11, p. 784-788, Dec. 1976.

EXPLANATION OF TEMPERATURE SENSOR SPECIFICATIONS

The way in which the AD590 is specified makes it easy to apply in a wide variety of different applications. It is important to understand the meaning of the various specifications and the effects of supply voltage and thermal environment on accuracy.

The AD590 is basically a PTAT (proportional to absolute temperature)[1] current regulator. That is, the output current is equal to a scale factor times the temperature of the sensor in degrees Kelvin. This scale factor is trimmed to 1 μA/K at the factory, by adjusting the indicated temperature (i.e., the output current) to agree with the actual temperature. This is done with 5 V across the device at a temperature within a few degrees of +25°C (298.2K). The device is then packaged and tested for accuracy over temperature.

CALIBRATION ERROR

At final factory test the difference between the indicated temperature and the actual temperature is called the calibration error. Since this is a scale factory error, its contribution to the total error of the device is PTAT. For example, the effect of the 1°C specified maximum error of the AD590L varies from 0.73°C at –55°C to 1.42°C at 150°C. Figure 3 shows how an exaggerated calibration error would vary from the ideal over temperature.

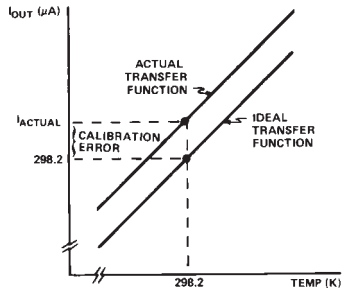


Figure 3. Calibration Error vs. Temperature

The calibration error is a primary contributor to maximum total error in all AD590 grades. However, since it is a scale factor error, it is particularly easy to trim. Figure 4 shows the most elementary way of accomplishing this. To trim this circuit the temperature of the AD590 is measured by a reference temperature sensor and R is trimmed so that V_T = 1 mV/K at that temperature. Note that when this error is trimmed out at one temperature, its effect is zero over the entire temperature range. In most applications there is a current-to-voltage conversion resistor (or, as with a current input ADC, a reference) that can be trimmed for scale factor adjustment.

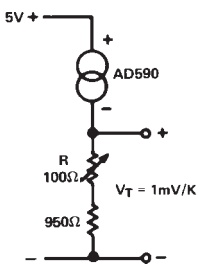


Figure 4. One Temperature Trim

[1]T(°C) = T(K) –273.2; Zero on the Kelvin scale is "absolute zero"; there is no lower temperature.

ERROR VERUS TEMPERATURE: WITH CALIBRATION ERROR TRIMMED OUT

Each AD590 is tested for error over the temperature range with the calibration error trimmed out. This specification could also be called the "variance from PTAT" since it is the maximum difference between the actual current over temperature and a PTAT multiplication of the actual current at 25°C. This error consists of a slope error and some curvature, mostly at the temperature extremes. Figure 5 shows a typical AD590K temperature curve before and after calibration error trimming.

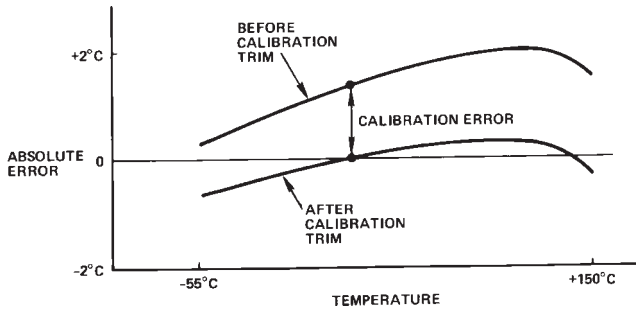


Figure 5. Effect to Scale Factor Trim on Accuracy

ERROR VERSUS TEMPERATURE: NO USER TRIMS

Using the AD590 by simply measuring the current, the total error is the "variance from PTAT" described above plus the effect of the calibration error over temperature. For example the AD590L maximum total error varies from 2.33°C at –55°C to 3.02°C at 150°C. For simplicity, only the large figure is shown on the specification page.

NONLINEARITY

Nonlinearity as it applies to the AD590 is the maximum deviation of current over temperature from a best-fit straight line. The nonlinearity of the AD590 over the –55°C to +150°C range is superior to all conventional electrical temperature sensors such as thermocouples. RTDs and thermistors. Figure 6 shows the nonlinearity of the typical AD590K from Figure 5.

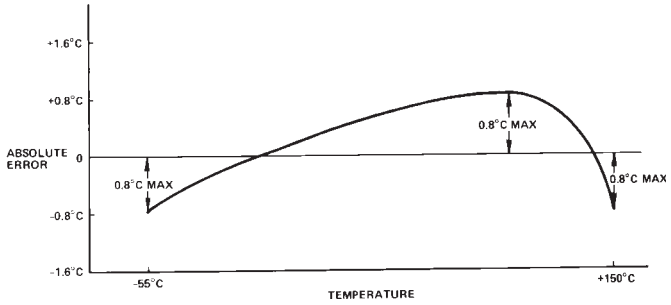


Figure 6. Nonlinearity

Figure 7A shows a circuit in which the nonlinearity is the major contributor to error over temperature. The circuit is trimmed by adjusting R_1 for a 0 V output with the AD590 at 0°C. R_2 is then adjusted for 10 V out with the sensor at 100°C. Other pairs of temperatures may be used with this procedure as long as they are measured accurately by a reference sensor. Note that for +15 V output (150°C) the V+ of the op amp must be greater than 17 V. Also note that V– should be at least –4 V: if V– is ground there is no voltage applied across the device.

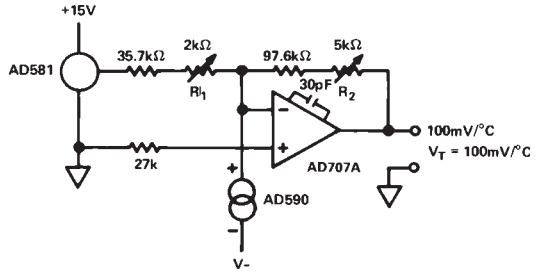


Figure 7A. Two Temperature Trim

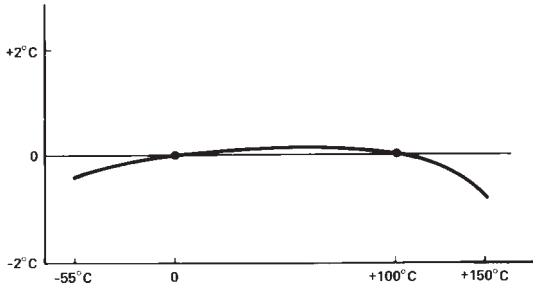


Figure 7B. Typical Two-Trim Accuracy

VOLTAGE AND THERMAL ENVIRONMENT EFFECTS

The power supply rejection specifications show the maximum expected change in output current versus input voltage changes. The insensitivity of the output to input voltage allows the use of unregulated supplies. It also means that hundreds of ohms of resistance (such as a CMOS multiplexer) can be tolerated in series with the device.

It is important to note that using a supply voltage other than 5 V does not change the PTAT nature of the AD590. In other words, this change is equivalent to a calibration error and can be removed by the scale factor trim (see previous page).

The AD590 specifications are guaranteed for use in a low thermal resistance environment with 5 V across the sensor. Large changes in the thermal resistance of the sensor's environment will change the amount of self-heating and result in changes in the output which are predictable but not necessarily desirable.

The thermal environment in which the AD590 is used determines two important characteristics: the effect of self heating and the response of the sensor with time.

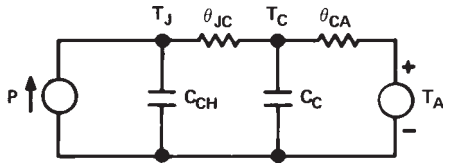


Figure 8. Thermal Circuit Model

Figure 8 is a model of the AD590 which demonstrates these characteristics. As an example, for the TO-52 package, θ_{JC} is the thermal resistance between the chip and the case, about 26°C/watt. θ_{CA} is the thermal resistance between the case and the surroundings and is determined by the characteristics of the thermal connection. Power source P represents the power dissipated on the chip. The rise of the junction temperature, T_J, above the ambient temperature T_A is:

$$T_J - T_A = P\,(\theta_{JC} + \theta_{CA}) \qquad \text{Equation 1}$$

Table I gives the sum of θ_{JC} and θ_{CA} for several common thermal media for both the "H" and "F" packages. The heatsink used was a common clip-on. Using Equation 1, the temperature rise of an AD590 "H" package in a stirred bath at +25°C, when driven with a 5 V supply, will be 0.06°C. However, for the same conditions in still air the temperature rise is 0.72°C. For a given supply voltage, the temperature rise varies with the current and is PTAT. Therefore, if an application circuit is trimmed with the sensor in the same thermal environment in which it will be used, the scale factor trim compensates for this effect over the entire temperature range.

Table I. Thermal Resistances

Medium	θ_{JC} + θ_{CA} (°C/Watt)		τ (sec)(Note 3)	
	H	F	H	F
Aluminum Block	30	10	0.6	0.1
Stirred Oil[1]	42	60	1.4	0.6
Moving Air[2]				
With Heat Sink	45	–	5.0	–
Without Heat Sink	115	190	13.5	10.0
Still Air				
With Heat Sink	191	–	108	–
Without Heat Sink	480	650	60	30

[1]Note: τ is dependent upon velocity of oil; average of several velocities listed above.
[2]Air velocity ≅ 9 ft./sec.
[3]The time constant is defined as the time required to reach 63.2% of an instantaneous temperature change.

The time response of the AD590 to a step change in temperature is determined by the thermal resistances and the thermal capacities of the chip, C_{CH}, and the case, C_C. C_{CH} is about 0.04 watt-sec/°C for the AD590. C_C varies with the measured medium since it includes anything that is in direct thermal contact with the case. In most cases, the single time constant exponential curve of Figure 9 is sufficient to describe the time response, T (t). Table I shows the effective time constant, τ, for several media.

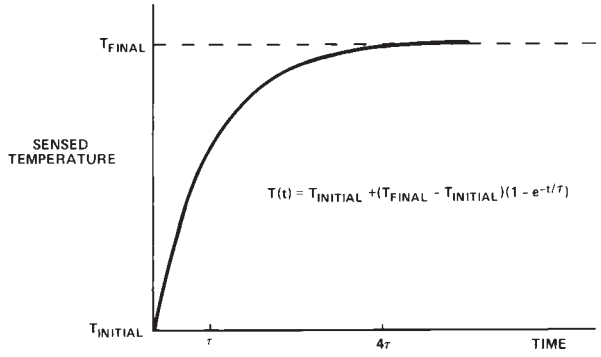


Figure 9. Time Response Curve

GENERAL APPLICATIONS

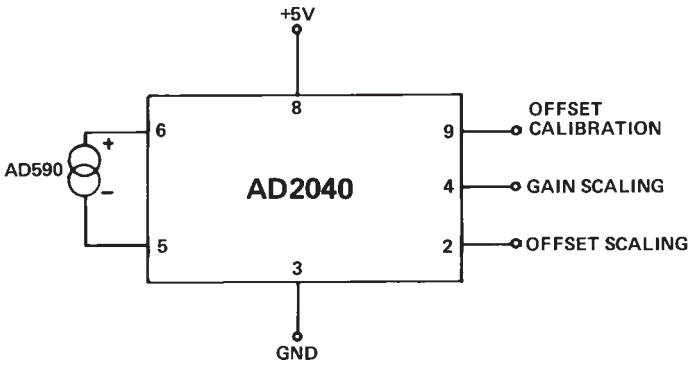


Figure 10. Variable Scale Display

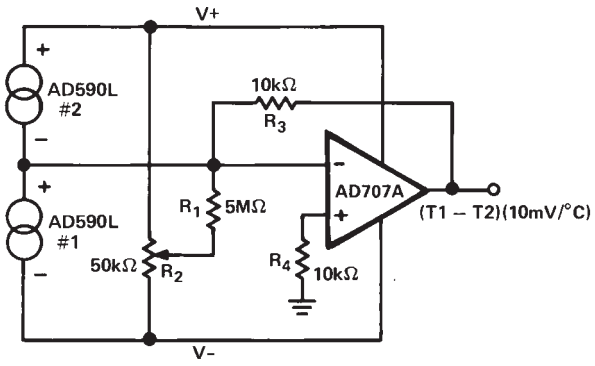


Figure 12. Differential Measurements

Figure 10 demonstrates the use of a low cost Digital Panel Meter for the display of temperature on either the Kelvin, Celsius or Fahrenheit scales. For Kelvin temperature Pins 9, 4 and 2 are grounded; and for Fahrenheit temperature Pins 4 and 2 are left open.

The above configuration yields a 3 digit display with 1°C or 1°F resolution, in addition to an absolute accuracy of ±2.0°C over the –55°C to +125°C temperature range if a one-temperature calibration is performed on an AD590K, L, or M.

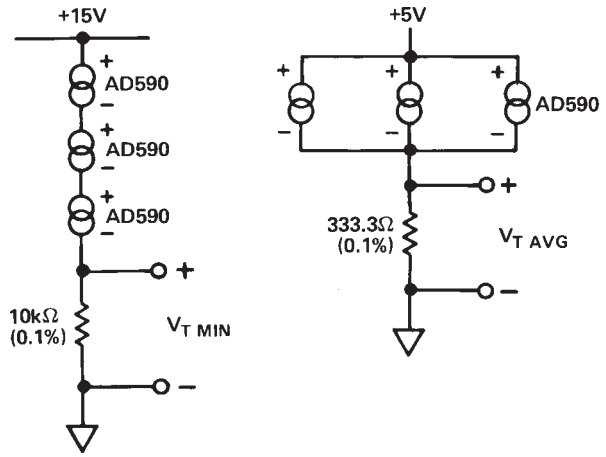


Figure 11. Series & Parallel Connection

Connecting several AD590 units in series as shown in Figure 11 allows the minimum of all the sensed temperatures to be indicated. In contrast, using the sensors in parallel yields the average of the sensed temperatures.

The circuit of Figure 12 demonstrates one method by which differential temperature measurements can be made. R1 and R2 can be used to trim the output of the op amp to indicate a desired temperature difference. For example, the inherent offset between the two devices can be trimmed in. If V+ and V– are radically different, then the difference in internal dissipation will cause a differential internal temperature rise. This effect can be used to measure the ambient thermal resistance seen by the sensors in applications such as fluid level detectors or anemometry.

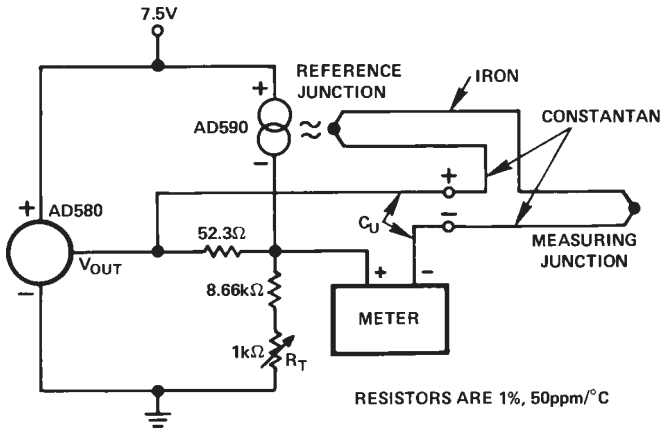


Figure 13. Cold Junction Compensation Circuit for Type J Thermocouple

Figure 13 is an example of a cold junction compensation circuit for a Type J Thermocouple using the AD590 to monitor the reference junction temperature. This circuit replaces an ice-bath as the thermocouple reference for ambient temperatures between +15°C and +35°C. The circuit is calibrated by adjusting R_T for a proper meter reading with the measuring junction at a known reference temperature and the circuit near +25°C. Using components with the TCs as specified in Figure 13, compensation accuracy will be within ±0.5°C for circuit temperatures between +15°C and +35°C. Other thermocouple types can be accommodated with different resistor values. Note that the TCs of the voltage reference and the resistors are the primary contributors to error.

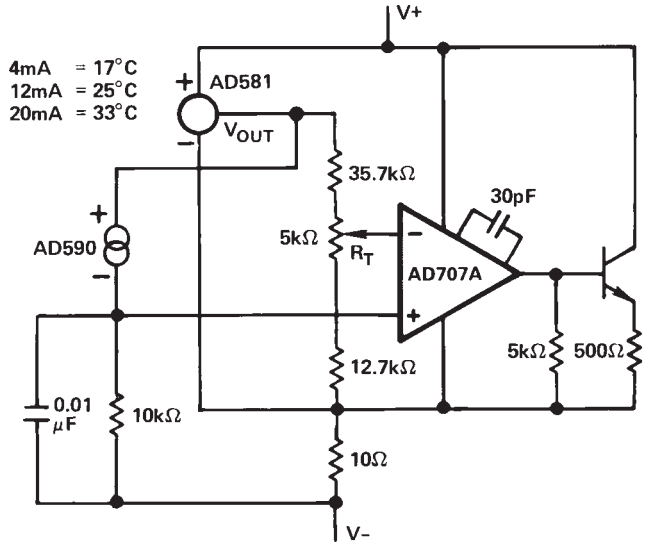


Figure 14. 4 mA-to-20 mA Current Transmitter

Figure 14 is an example of a current transmitter designed to be used with 40 V, 1 kΩ systems; it uses its full current range of 4 mA-to-20 mA for a narrow span of measured temperatures. In this example the 1 µA/K output of the AD590 is amplified to 1 mA/°C and offset so that 4 mA is equivalent to 17°C and 20 mA is equivalent to 33°C. R_T is trimmed for proper reading at an intermediate reference temperature. With a suitable choice of resistors, any temperature range within the operating limits of the AD590 may be chosen.

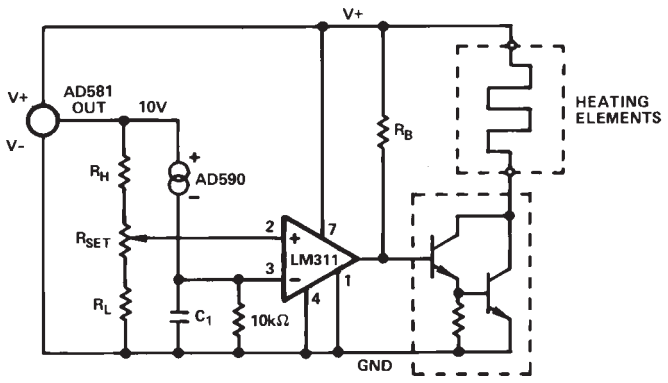


Figure 15. Simple Temperature Control Circuit

Figure 15 is an example of a variable temperature control circuit (thermostat) using the AD590. R_H and R_L are selected to set the high and low limits for R_{SET}. R_{SET} could be a simple pot, a calibrated multiturn pot or a switched resistive divider. Powering the AD590 from the 10 V reference isolates the AD590 from supply variations while maintaining a reasonable voltage (~7 V) across it. Capacitor C_1 is often needed to filter extraneous noise from remote sensors. R_B is determined by the β of the power transistor and the current requirements of the load.

Figure 16 shows the AD590 can be configured with an 8-bit DAC to produce a digitally controlled set point. This particular circuit operates from 0°C (all inputs high) to +51°C (all inputs

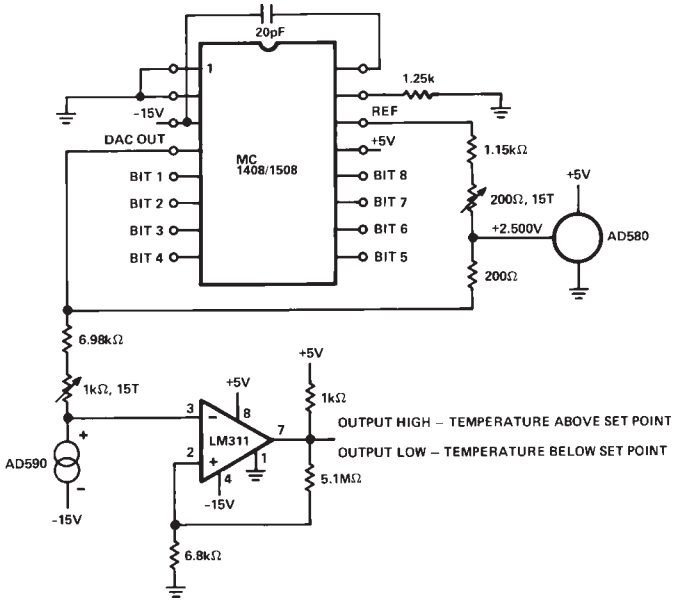


Figure 16. DAC Set Point

low) in 0.2°C steps. The comparator is shown with 1°C hysteresis which is usually necessary to guard-band for extraneous noise; omitting the 5.1 MΩ resistor results in no hysteresis.

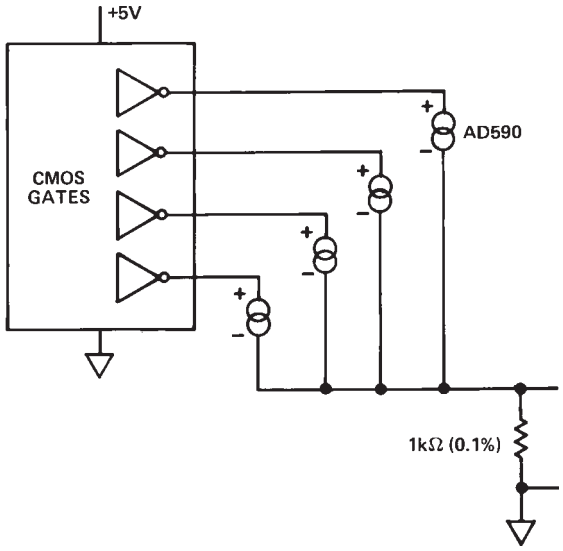


Figure 17. AD590 Driven from CMOS Logic

The voltage compliance and the reverse blocking characteristic of the AD590 allows it to be powered directly from +5 V CMOS logic. This permits easy multiplexing, switching or pulsing for minimum internal heat dissipation. In Figure 17 any AD590 connected to a logic high will pass a signal current through the current measuring circuitry while those connected to a logic zero will pass insignificant current. The outputs used to drive the AD590s may be employed for other purposes, but the additional capacitance due to the AD590 should be taken into account.

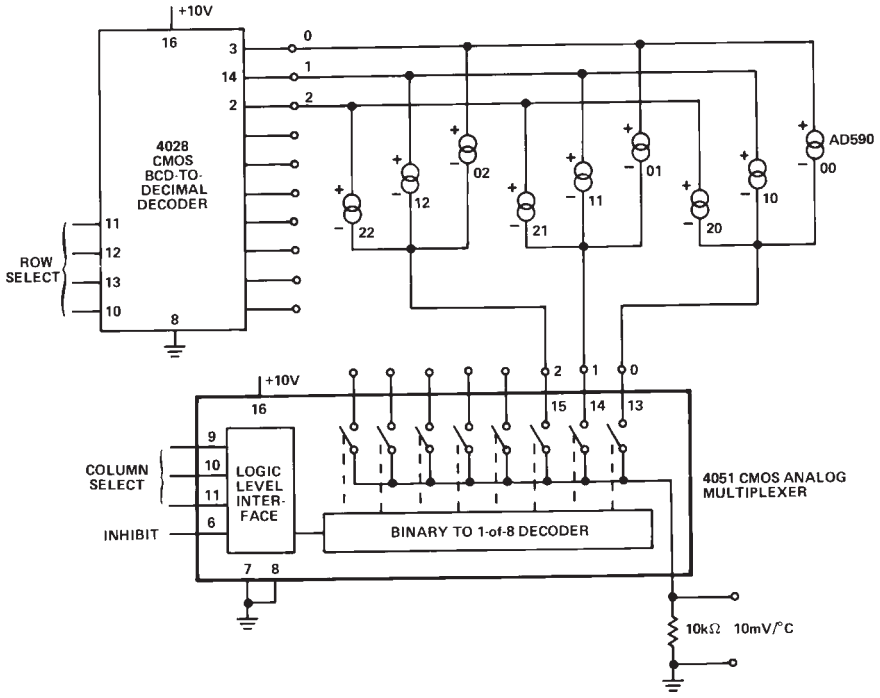


Figure 18. Matrix Multiplexer

CMOS Analog Multiplexers can also be used to switch AD590 current. Due to the AD590's current mode, the resistance of such switches is unimportant as long as 4 V is maintained across the transducer. Figure 18 shows a circuit which combines the principal demonstrated in Figure 17 with an 8-channel CMOS Multiplexer. The resulting circuit can select one of eighty sensors over only 18 wires with a 7-bit binary word. The inhibit input on the multiplexer turns all sensors off for minimum dissipation while idling.

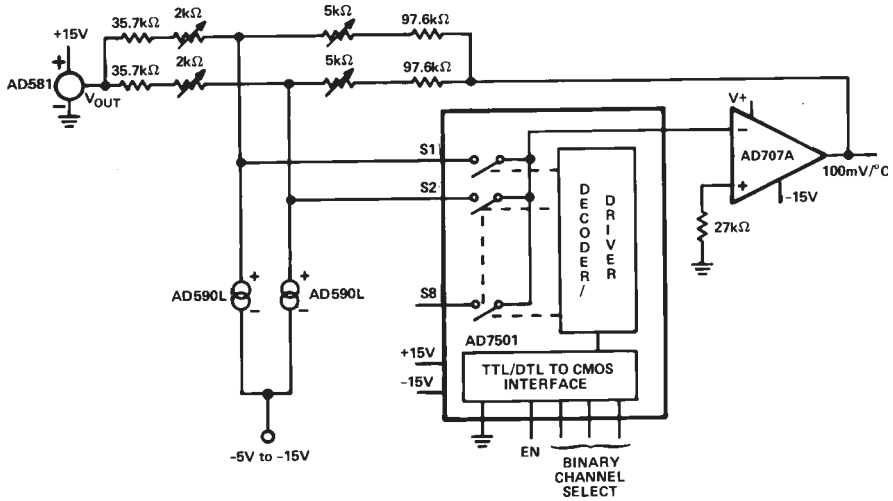


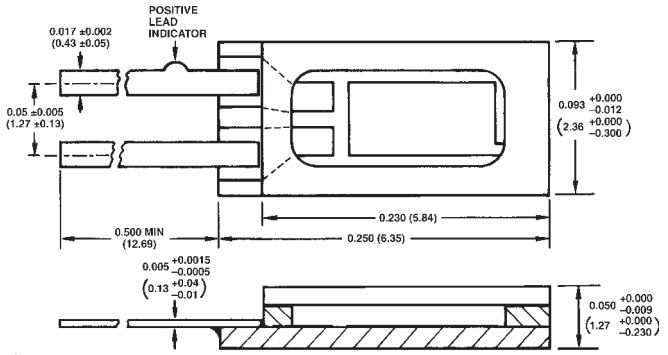
Figure 19. 8-Channel Multiplexer

Figure 19 demonstrates a method of multiplexing the AD590 in the two-trim mode (Figure 7). Additional AD590s and their associated resistors can be added to multiplex up to 8 channels of ±0.5°C absolute accuracy over the temperature range of –55°C to +125°C. The high temperature restriction of +125°C is due to the output range of the op amps; output to +150°C can be achieved by using a +20 V supply for the op amp.

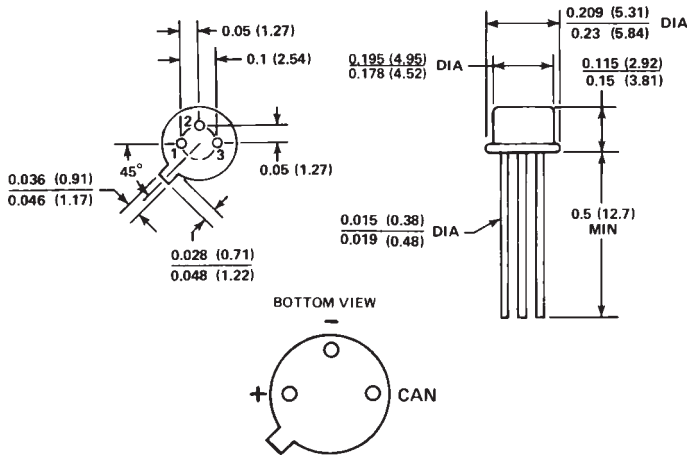
OUTLINE DIMENSIONS AND PIN DESIGNATIONS

Dimensions shown in inches and (mm).

FLATPACK PACKAGE: DESIGNATION "F''



TO-52 Package: Designation "H''



C426e–0–3/97

PRINTED IN U.S.A.

FIGARO

PRODUCT INFORMATION

TGS 822 - for the detection of Organic Solvent Vapors

Features:

* High sensitivity to organic solvent vapors such as ethanol
* High stability and reliability over a long period
* Long life and low cost
* Uses simple electrical circuit

Applications:

* Breath alcohol detectors
* Gas leak detectors/alarms
* Solvent detectors for factories, dry cleaners, and semiconductor industries

The sensing element of Figaro gas sensors is a tin dioxide (SnO_2) semiconductor which has low conductivity in clean air. In the presence of a detectable gas, the sensor's conductivity increases depending on the gas concentration in the air. A simple electrical circuit can convert the change in conductivity to an output signal which corresponds to the gas concentration.

The **TGS 822** has high sensitivity to the vapors of organic solvents as well as other volatile vapors. It also has sensitivity to a variety of combustible gases such as carbon monoxide, making it a good general purpose sensor. Also available with a ceramic base which is highly resistant to severe environments as high as 200°C (model# TGS 823).

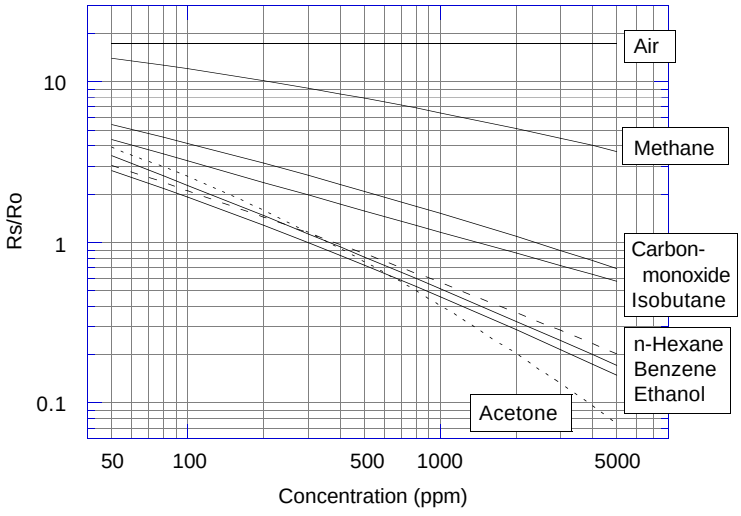
The figure below represents typical sensitivity characteristics, all data having been gathered at standard test conditions (see reverse side of this sheet). The Y-axis is indicated as sensor resistance ratio (Rs/Ro) which is defined as follows:

Rs = Sensor resistance of displayed gases at various concentrations
Ro = Sensor resistance in 300ppm ethanol

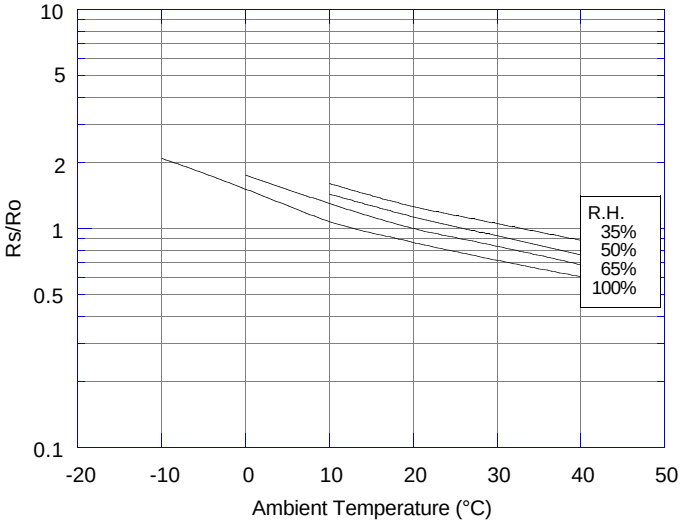
The figure below represents typical temperature and humidity dependency characteristics. Again, the Y-axis is indicated as sensor resistance ratio (Rs/Ro), defined as follows:

Rs = Sensor resistance at 300ppm of ethanol at various temperatures/humidities
Ro = Sensor resistance at 300ppm of ethanol at 20°C and 65% R.H.

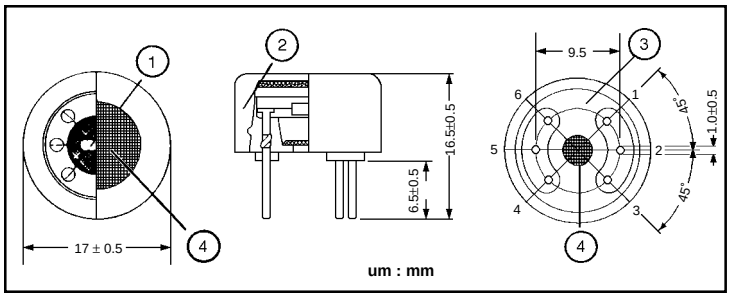
Sensitivity Characteristics:



Temperature/Humidity Dependency:



Structure and Dimensions:

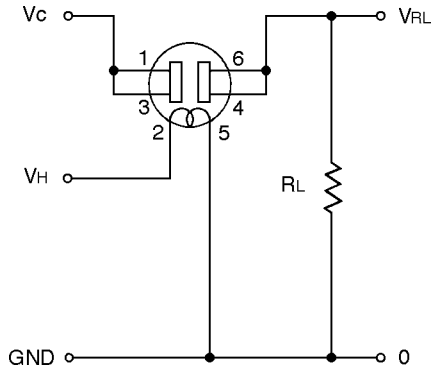


(1) Sensing Element:
SnO_2 is sintered to form a thick film on the surface of an alumina ceramic tube which contains an internal heater.
(2) Cap:
Nylon 66
(3) Sensor Base:
Nylon 66
(4) Flame Arrestor:
100 mesh SUS 316 double gauze

Pin Connection and Basic Measuring Circuit:

The numbers shown around the sensor symbol in the circuit diagram at the right correspond with the pin numbers shown in the sensor's structure drawing (above). When the sensor is connected as shown in the basic circuit, output across the Load Resistor (V_{RL}) increases as the sensor's resistance (Rs) decreases, depending on gas concentration.

Basic Measuring Circuit:



Standard Circuit Conditions:

Item	Symbol	Rated Values	Remarks
Heater Voltage	V_H	5.0±0.2V	AC or DC
Circuit Voltage	V_C	Max. 24V	AC or DC *PS 15mW
Load Resistance	R_L	Variable	*PS 15mW

Electrical Characteristics:

Item	Symbol	Condition	Specification
Sensor Resistance	Rs	Ethanol at 300ppm/Air	1k ~ 10k
Change Ratio of Sensor Resistance	Rs/Ro	$\frac{\text{Rs (Ethanol at 300ppm/Air)}}{\text{Rs (Ethanol at 50ppm/Air)}}$	0.40 ± 0.1
Heater Resistance	R_H	Room temperature	38.0 ± 3.0
Heater Power Consumption	P_H	VH=5.0V	660mW ± 55mW

Standard Test Conditions:

TGS 822 complies with the above electrical characteristics when the sensor is tested in standard conditions as specified below:

Test Gas Conditions: 20°±2°C, 65±5%R.H.
Circuit Conditions: V_C = 10.0±0.1V (AC or DC),
V_H = 5.0±0.05V (AC or DC),
R_L = 10.0kΩ±1%
Preheating period before testing: More than 7 days

Sensor Resistance (Rs) is calculated by the following formula:

$$Rs = \left(\frac{V_C}{V_{RL}} - 1\right) \times R_L$$

Power dissipation across sensor electrodes (Ps) is calculated by the following formula:

$$Ps = \frac{V_C^2 \times Rs}{(Rs + R_L)^2}$$

FIGARO USA, INC.
3703 West Lake Ave. Suite 203
Glenview, Illinois 60025
Phone: (847)-832-1701
Fax: (847)-832-1705
email: figarousa@figarosensor.com

ST-1KLA

ST-1KLAは、メタルキャップをハーメチックシーリングした、TO-18タイプの高感度シリコンフォトトランジスタです。屋外使用等厳しい条件下での信頼性が高められ、経時変化が少なく、耐久性、高信頼性の要求に答えます。

The ST-1KLA are high-sensitivity NPN silicon phototransistors mounted on durable, hermetically sealed TO-18 metal cans, providing years of reliable performance even under demanding conditions such as use outdoors.

ベース端子なし：ST-1KLA／Two leads (Collector, Emitter)：ST-1KLA

■特長　FEATURES

- ●TO-18レンズ付きキャンタイプ
- ●高信頼性

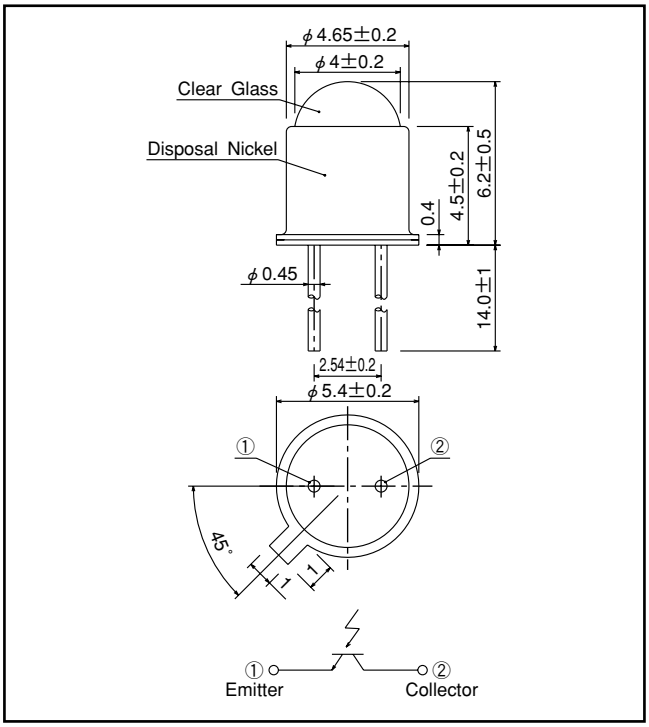
- ●TO-18 can type with lens
- ●High reliability

■用途　APPLICATIONS

- ●光電スイッチ
- ●産業機器

- ●Optical switches
- ●Industrial machines

■外形寸法　DIMENSIONS（Unit : mm）



■最大定格 MAXIMUM RATINGS

（Ta=25℃）

Item	Symbol	Rating	Unit
コレクタ・エミッタ間電圧　C-E voltage	V_{CEO}	40	V
エミッタ・コレクタ間電圧　E-C voltage	V_{ECO}	4	V
コレクタ電流　Collector current	I_C	50	mA
コレクタ損失　Collector power dissipation	P_C	150	mW
動作温度　Operating temp.	Topr.	−30～+100	℃
保存温度　Storage temp.	Tstg.	−50～+150	℃
半田付温度　Soldering temp.*1	Tsol.	260	℃

＊1. リード根元より2mm離れた所で5秒
For MAX. 5 seconds at the position of 2 mm from the resin edge

■電気的光学的特性　ELECTRO-OPTICAL　CHARACTERISTICS

（Ta=25℃）

Item		Symbol	Conditions	Min.	Typ.	Max.	Unit.
暗電流	Collecor dark current	I_{CEO}	V_{CEO}=10V		1	200	nA
光電流	Light current	I_L	V_{CE}=10V,E_V=200Lx*2	1.5	6.0	16	mA
コレクタ・エミッタ間飽和電圧	C-E saturation voltage	$V_{CE(sat)}$	I_C=2mA,E_V=2000Lx*2		0.2	0.4	V
応答時間 Switching speeds	立上り時間　Rise time	tr	V_{CC}=10V I_C=5mA R_L=100Ω		8		μsec.
	立下り時間　Fall time	tf			10		μsec.
分光感度	Spectral sensitivity	λ		500～1050			nm
ピーク感度波長	Peak wavelength	λp			880		nm
半値角	Half angle	Δθ			±15		deg.

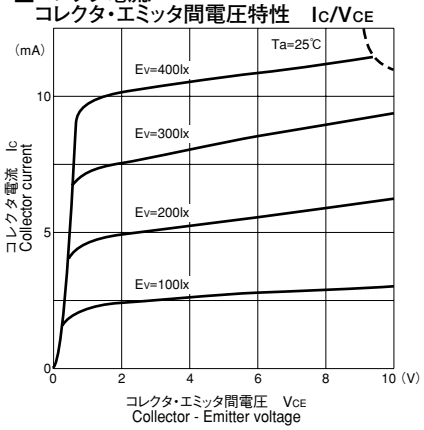
＊2. 色温度＝2856K標準タングステン電球
Color temp. = 2856K standard Tungsten lamp

フォトトランジスタ　Phototransistors

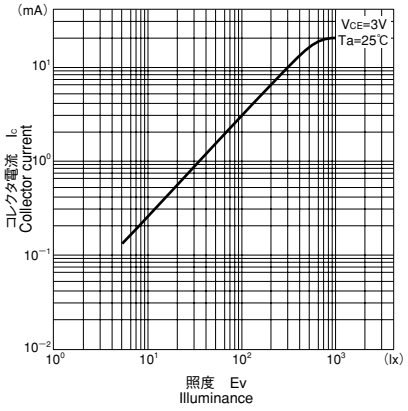
ST-1KLA

KODENSHI CORP.

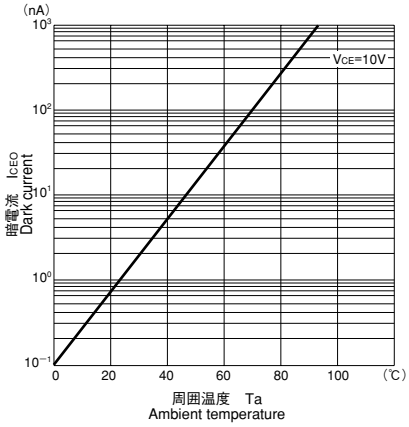
■コレクタ電流/
コレクタ・エミッタ間電圧特性　I_C/V_{CE}



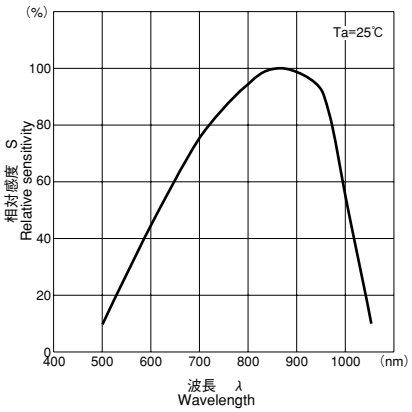
■コレクタ電流/照度特性　I_C/Ev



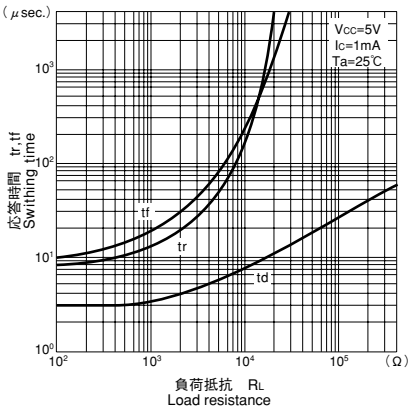
■暗電流/周囲温度特性　I_{CEO}/Ta



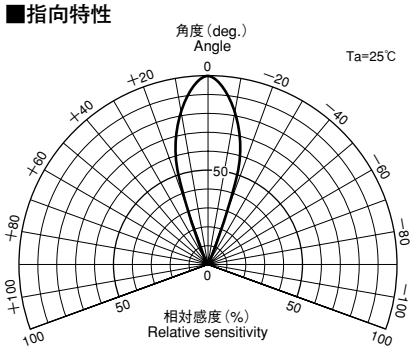
■分光感度特性



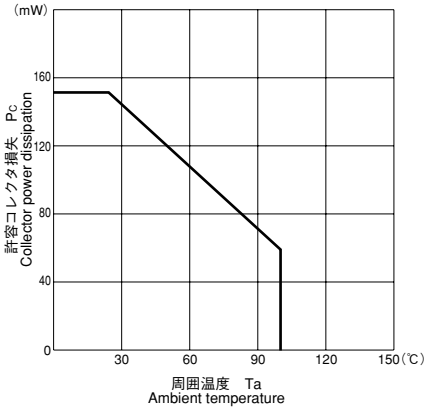
■応答時間/負荷抵抗特性　tr,tf/R_L ※1

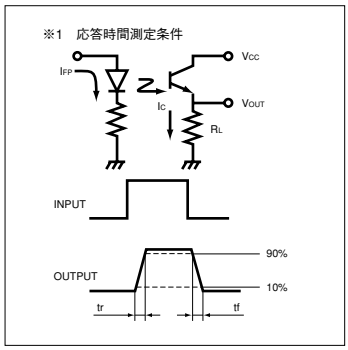


■指向特性



■許容コレクタ損失/周囲温度　P_C/Ta





光敏電阻感測器模組(B160)

1 光敏電阻 CDS

CDS 是以硫化鎘成份爲主的光導電元件，其具有歐姆性。你可以將它以電阻的方式來使用。

主要用途：

*路燈之自動點滅

*照相機之測光計

2 原理

圖 1 表示 CDS 的原理圖，此處 CDS 光導電體在兩電極間爲一個電阻體，其內有折疊狀的硫化鎘作爲光電導體的主要材料。若光線射入元件折疊狀的硫化鎘，則在其內部會產生光導電效應，而使電極 A，電極 B 兩端之電阻變小。

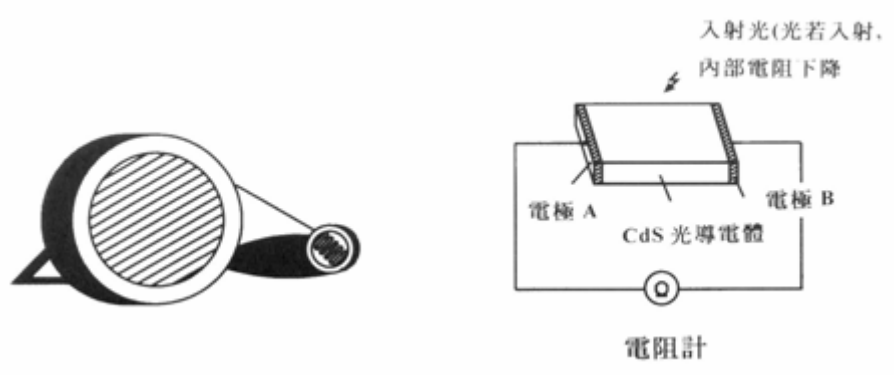


圖 1

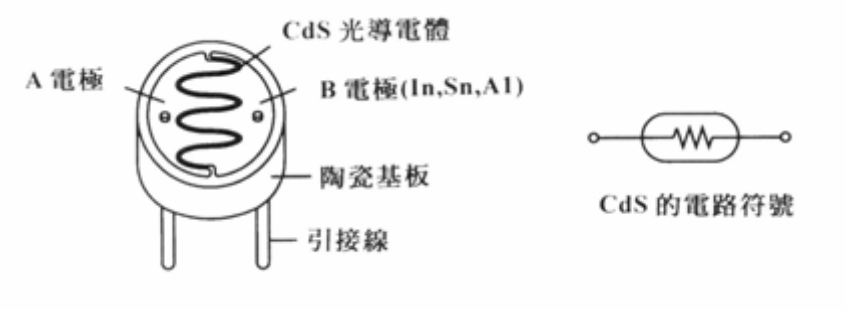


圖 2 CDS 的構造

圖 2 A 表示 CDS 之一般構造。其波浪線爲光導電體。而一般的電路符號如圖 2 B。

3 CDS 的特性

CDS 的基本特性對應入射光量，其內部電阻有變小的現象。而其電阻值隨照度的變化量而變化，其變化值的大小與構成元件素材的性質有關：

$$r = \frac{\log(R_{10} / R_{100})}{\log(100 / 10)}$$

R10 ：照度 10Lux 的 CDS 電阻

R100：照度 100Lux 的 CDS 電阻

CDS 的 r 特性一般 0.5–1，反光式的一般爲 0.6-0.7，而照相機的露出計常使用 r ≒1 的 CDS。

CDS 的響應特性非常慢，與光電晶體比較，相差甚遠，故 CDS 適用於靜態照度變化。

4 應用電路

電路分析(圖 3)

① CDS，R4，VR 構成信號拾取電路，VRI 可用來調整電路靈敏度。

② U4A 構成電壓隨藕器，做爲阻抗匹配用。

③ R5，R7，R8，U4B 構成第一級直流放大電路，其電壓放大倍率 AV1，

$$AV1 = -\frac{R7}{R5} = -\frac{33K}{10K}$$

$$AV2 = -\frac{R10}{R9} = -\frac{3.9K}{3.9K}$$

$$= -1$$

= -3.3

④ R9，R10，R1 1，U4C 構成第二級直流放大電路，其電壓放大倍率 AV2，

⑤ 電路總電壓增益 AV，

$AV = AV_1 \times AV_2$

= (-3.3)X(-1)

= 3.3

⑥ R6，VR2 構成零點調整電路

⑦ R12，D1 構成保護電路

⑧ U5A，U5B 將類比信號 AIN0 轉成數位信號轉出

⑨ R13，LED3 爲 CDS 的燈號顯示電路。當燈光愈亮時，LED3 亮，燈光愈暗，則 LED3 滅

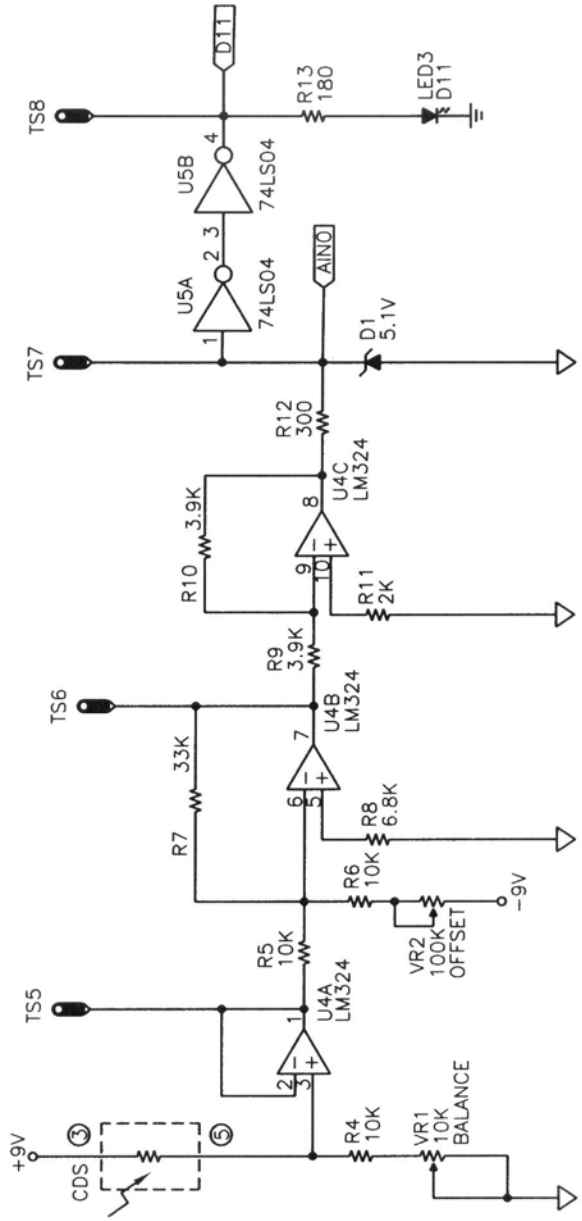

D11
R13
180
LED3
D11
TS8
U5B
74LS04
AIN0
U5A
74LS04
D1
5.1V
TS7
R12
300
U4C
LM324
3.9K
R10
R11
2K
R9
3.9K
TS6
U4B
LM324
33K
R7
R8
6.8K
R6
10K
-9V
VR2
100K
OFFSET
R5
10K
TS5
U4A
LM324
+9V
CDS
R4
10K
VR1
10K
BALANCE

圖 2-3

Features

- **Low-voltage and Standard-voltage Operation**
  - **2.7 (V_{CC} = 2.7V to 5.5V)**
  - **1.8 (V_{CC} = 1.8V to 5.5V)**
- **Internally Organized 128 x 8 (1K), 256 x 8 (2K), 512 x 8 (4K), 1024 x 8 (8K) or 2048 x 8 (16K)**
- **2-wire Serial Interface**
- **Schmitt Trigger, Filtered Inputs for Noise Suppression**
- **Bi-directional Data Transfer Protocol**
- **100 kHz (1.8V) and 400 kHz (2.5V, 2.7V, 5V) Compatibility**
- **Write Protect Pin for Hardware Data Protection**
- **8-byte Page (1K, 2K), 16-byte Page (4K, 8K, 16K) Write Modes**
- **Partial Page Writes are Allowed**
- **Self-timed Write Cycle (5 ms max)**
- **High-reliability**
  - **Endurance: 1 Million Write Cycles**
  - **Data Retention: 100 Years**
- **Automotive Grade, Extended Temperature and Lead-free/Halogen-free Devices Available**
- **8-lead PDIP, 8-lead JEDEC SOIC, 8-lead MAP, 5-lead SOT23, 8-lead TSSOP and 8-ball dBGA2™ Packages**

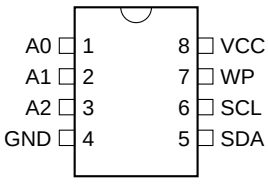
Description

The AT24C01A/02/04/08/16 provides 1024/2048/4096/8192/16384 bits of serial electrically erasable and programmable read-only memory (EEPROM) organized as 128/256/512/1024/2048 words of 8 bits each. The device is optimized for use in many industrial and commercial applications where low-power and low-voltage operation are essential. The AT24C01A/02/04/08/16 is available in space-saving 8-lead PDIP, 8-lead JEDEC SOIC, 8-lead MAP, 5-lead SOT23 (AT24C01A/AT24C02/AT24C04), 8-lead TSSOP and 8-ball dBGA2 packages and is accessed via a 2-wire serial interface.

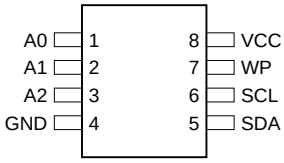
Pin Configurations

Pin Name	Function
A0 - A2	Address Inputs
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect
NC	No Connect
GND	Ground
VCC	Power Supply

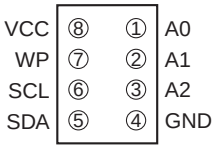
8-lead TSSOP



8-lead SOIC

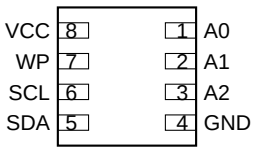


8-ball dBGA2



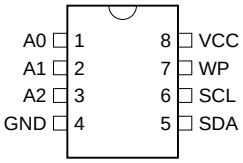
Bottom View

8-lead MAP

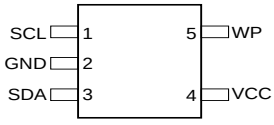


Bottom View

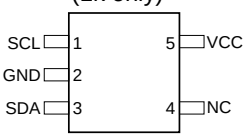
8-lead PDIP



5-lead SOT23



5-lead SOT23 Rotated (R)
(1k only)



2-wire Serial EEPROM

1K (128 x 8)

2K (256 x 8)

4K (512 x 8)

8K (1024 x 8)

16K (2048 x 8)

AT24C01A
AT24C02
AT24C04
AT24C08(1)
AT24C16(2)

Note: 1. This device is not recommended for new designs. Please refer to AT24C08A.

2. This device is not recommended for new designs. Please refer to AT24C16A.

0180Q–SEEPR–1/04

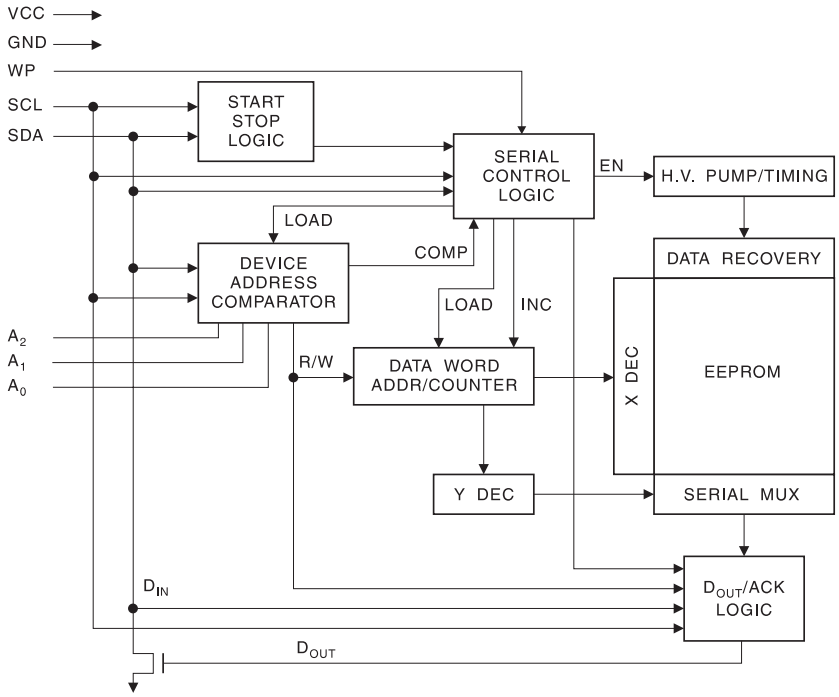
In addition, the entire family is available in 2.7V (2.7V to 5.5V) and 1.8V (1.8V to 5.5V) versions.

Absolute Maximum Ratings

Operating Temperature	-55° C to +125° C
Storage Temperature	-65° C to +150° C
Voltage on Any Pin with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.25V
DC Output Current	5.0 mA

*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Block Diagram



Pin Description

SERIAL CLOCK (SCL): The SCL input is used to positive edge clock data into each EEPROM device and negative edge clock data out of each device.

SERIAL DATA (SDA): The SDA pin is bi-directional for serial data transfer. This pin is open-drain driven and may be wire-ORed with any number of other open-drain or open-collector devices.

DEVICE/PAGE ADDRESSES (A2, A1, A0): The A2, A1 and A0 pins are device address inputs that are hard wired for the AT24C01A and the AT24C02. As many as eight 1K/2K devices may be addressed on a single bus system (device addressing is discussed in detail under the Device Addressing section).

The AT24C04 uses the A2 and A1 inputs for hard wire addressing and a total of four 4K devices may be addressed on a single bus system. The A0 pin is a no connect.

The AT24C08 only uses the A2 input for hardwire addressing and a total of two 8K devices may be addressed on a single bus system. The A0 and A1 pins are no connects.

The AT24C16 does not use the device address pins, which limits the number of devices on a single bus to one. The A0, A1 and A2 pins are no connects.

WRITE PROTECT (WP): The AT24C01A/02/04/16 has a Write Protect pin that provides hardware data protection. The Write Protect pin allows normal read/write operations when connected to ground (GND). When the Write Protect pin is connected to V_{CC}, the write protection feature is enabled and operates as shown in the following table.

WP Pin Status	Part of the Array Protected				
	24C01A	24C02	24C04	24C08[1]	24C16[2]
At V_{CC}	Full (1K) Array	Full (2K) Array	Full (4K) Array	Normal Read/ Write Operation	Upper Half (8K) Array
At GND	Normal Read/Write Operations				

Notes: 1. This device is not recommended for new designs. Please refer to AT24C08A.
2. This device is not recommended for new designs. Please refer to AT24C16A.

Memory Organization

AT24C01A, 1K SERIAL EEPROM: Internally organized with 16 pages of 8 bytes each, the 1K requires a 7-bit data word address for random word addressing.

AT24C02, 2K SERIAL EEPROM: Internally organized with 32 pages of 8 bytes each, the 2K requires an 8-bit data word address for random word addressing.

AT24C04, 4K SERIAL EEPROM: Internally organized with 32 pages of 16 bytes each, the 4K requires a 9-bit data word address for random word addressing.

AT24C08, 8K SERIAL EEPROM: Internally organized with 64 pages of 16 bytes each, the 8K requires a 10-bit data word address for random word addressing.

AT24C16, 16K SERIAL EEPROM: Internally organized with 128 pages of 16 bytes each, the 16K requires an 11-bit data word address for random word addressing.

Pin Capacitance[1]

Applicable over recommended operating range from T_A = 25° C, f = 1.0 MHz, V_{CC} = +1.8V.

Symbol	Test Condition	Max	Units	Conditions
$C_{I/O}$	Input/Output Capacitance (SDA)	8	pF	$V_{I/O}$ = 0V
C_{IN}	Input Capacitance (A_0, A_1, A_2, SCL)	6	pF	V_{IN} = 0V

Note: 1. This parameter is characterized and is not 100% tested.

DC Characteristics

Applicable over recommended operating range from: T_{AI} = -40° C to +85° C, V_{CC} = +1.8V to +5.5V, T_{AE} = -40° C to +125° C, V_{CC} = +1.8V to +5.5V (unless otherwise noted).

Symbol	Parameter	Test Condition	Min	Typ	Max	Units
V_{CC1}	Supply Voltage		1.8		5.5	V
V_{CC2}	Supply Voltage		2.5		5.5	V
V_{CC3}	Supply Voltage		2.7		5.5	V
V_{CC4}	Supply Voltage		4.5		5.5	V
I_{CC}	Supply Current V_{CC} = 5.0V	READ at 100 kHz		0.4	1.0	mA
I_{CC}	Supply Current V_{CC} = 5.0V	WRITE at 100 kHz		2.0	3.0	mA
I_{SB1}	Standby Current V_{CC} = 1.8V	V_{IN} = V_{CC} or V_{SS}		0.6	3.0	µA
I_{SB2}	Standby Current V_{CC} = 2.5V	V_{IN} = V_{CC} or V_{SS}		1.4	4.0	µA
I_{SB3}	Standby Current V_{CC} = 2.7V	V_{IN} = V_{CC} or V_{SS}		1.6	4.0	µA
I_{SB4}	Standby Current V_{CC} = 5.0V	V_{IN} = V_{CC} or V_{SS}		8.0	18.0	µA
I_{LI}	Input Leakage Current	V_{IN} = V_{CC} or V_{SS}		0.10	3.0	µA
I_{LO}	Output Leakage Current	V_{OUT} = V_{CC} or V_{SS}		0.05	3.0	µA
V_{IL}	Input Low Level[1]		-0.6		V_{CC} x 0.3	V
V_{IH}	Input High Level[1]		V_{CC} x 0.7		V_{CC} + 0.5	V
V_{OL2}	Output Low Level V_{CC} = 3.0V	I_{OL} = 2.1 mA			0.4	V
V_{OL1}	Output Low Level V_{CC} = 1.8V	I_{OL} = 0.15 mA			0.2	V

Note: 1. V_{IL} min and V_{IH} max are reference only and are not tested.

AC Characteristics

Applicable over recommended operating range from T_{AI} = -40° C to +85° C, T_{AE} = -40° C to +125° C, V_{CC} = +1.8V to +5.5V, CL = 1 TTL Gate and 100 pF (unless otherwise noted).

Symbol	Parameter	1.8-volt		2.5, 2.7, 5.0-volt		Units
		Min	Max	Min	Max	
f_{SCL}	Clock Frequency, SCL		100		400(1)	kHz
t_{LOW}	Clock Pulse Width Low	4.7		1.2		µs
t_{HIGH}	Clock Pulse Width High	4.0		0.6		µs
t_I	Noise Suppression Time(2)		100		50	ns
t_{AA}	Clock Low to Data Out Valid	0.1	4.5	0.1	0.9	µs
t_{BUF}	Time the bus must be free before a new transmission can start(2)	4.7		1.2		µs
$t_{HD.STA}$	Start Hold Time	4.0		0.6		µs
$t_{SU.STA}$	Start Setup Time	4.7		0.6		µs
$t_{HD.DAT}$	Data In Hold Time	0		0		µs
$t_{SU.DAT}$	Data In Setup Time	200		100		ns
t_R	Inputs Rise Time(2)		1.0		0.3	µs
t_F	Inputs Fall Time(2)		300		300	ns
$t_{SU.STO}$	Stop Setup Time	4.7		0.6		µs
t_{DH}	Data Out Hold Time	100		50		ns
t_{WR}	Write Cycle Time		5		5	ms
Endurance(1)	5.0V, 25° C, Byte Mode	1M		1M		Write Cycles

Note: 1. The 24C01A/02/04 bearing the process letter “D” on the package (the mark is located in the lower right corner on the top side of the package), guarantees 400 kHz (2.5 – 5.0V).

2. This parameter is characterized and is not 100% tested.

Device Operation

CLOCK and DATA TRANSITIONS: The SDA pin is normally pulled high with an external device. Data on the SDA pin may change only during SCL low time periods (refer to Data Validity timing diagram). Data changes during SCL high periods will indicate a start or stop condition as defined below.

START CONDITION: A high-to-low transition of SDA with SCL high is a start condition which must precede any other command (refer to Start and Stop Definition timing diagram).

STOP CONDITION: A low-to-high transition of SDA with SCL high is a stop condition. After a read sequence, the stop command will place the EEPROM in a standby power mode (refer to Start and Stop Definition timing diagram).

ACKNOWLEDGE: All addresses and data words are serially transmitted to and from the EEPROM in 8-bit words. The EEPROM sends a zero to acknowledge that it has received each word. This happens during the ninth clock cycle.

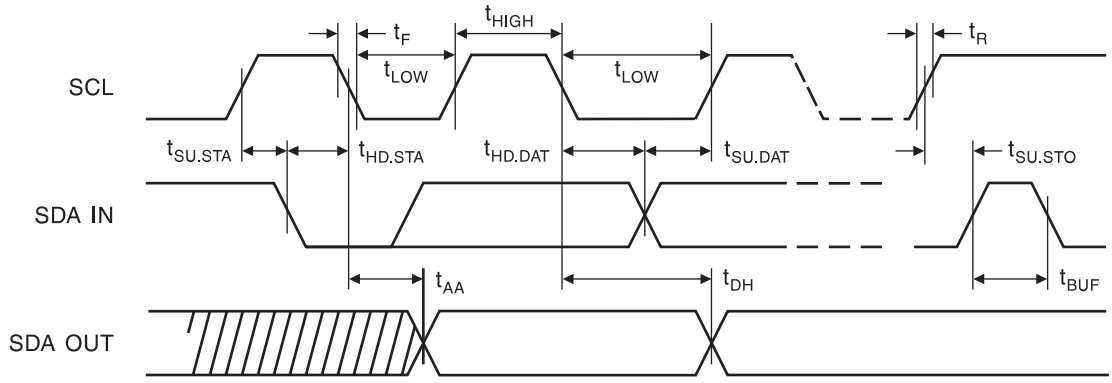
STANDBY MODE: The AT24C01A/02/04/08/16 features a low-power standby mode which is enabled: (a) upon power-up and (b) after the receipt of the STOP bit and the completion of any internal operations.

MEMORY RESET: After an interruption in protocol, power loss or system reset, any 2-wire part can be reset by following these steps:

1. Clock up to 9 cycles.
2. Look for SDA high in each cycle while SCL is high.
3. Create a start condition.

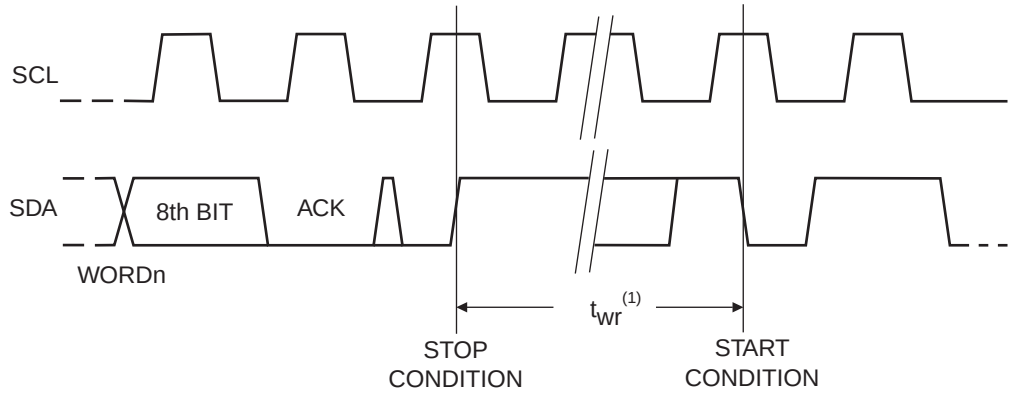
Bus Timing

SCL: Serial Clock, SDA: Serial Data I/O



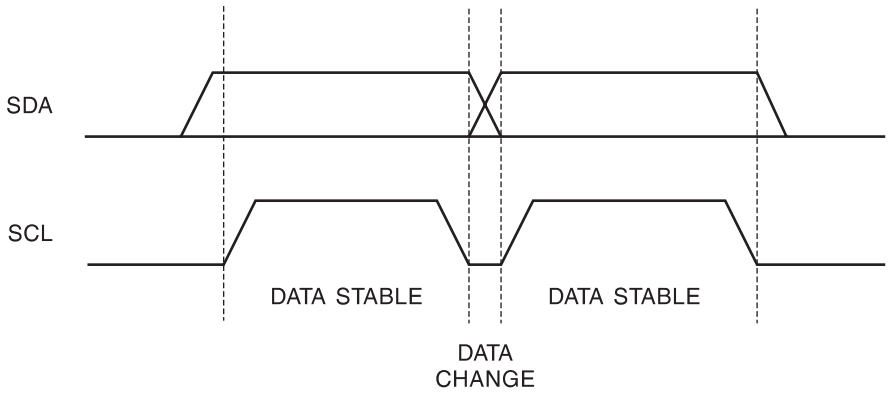
Write Cycle Timing

SCL: Serial Clock, SDA: Serial Data I/O

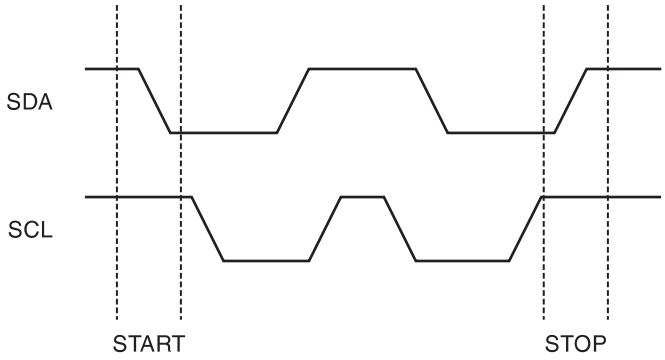


Note: 1. The write cycle time t_{WR} is the time from a valid stop condition of a write sequence to the end of the internal clear/write cycle.

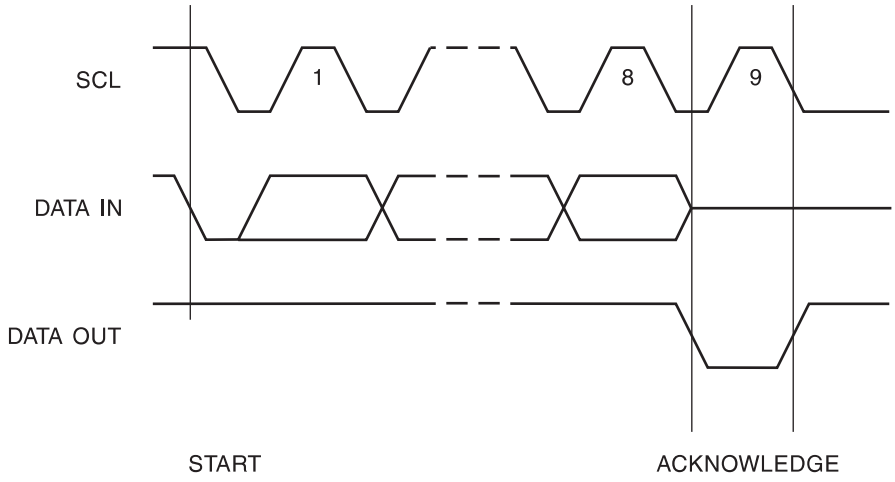
Data Validity



Start and Stop Definition



Output Acknowledge



Device Addressing

The 1K, 2K, 4K, 8K and 16K EEPROM devices all require an 8-bit device address word following a start condition to enable the chip for a read or write operation (refer to Figure 1).

The device address word consists of a mandatory one, zero sequence for the first four most significant bits as shown. This is common to all the EEPROM devices.

The next 3 bits are the A2, A1 and A0 device address bits for the 1K/2K EEPROM. These 3 bits must compare to their corresponding hard-wired input pins.

The 4K EEPROM only uses the A2 and A1 device address bits with the third bit being a memory page address bit. The two device address bits must compare to their corresponding hard-wired input pins. The A0 pin is no connect.

The 8K EEPROM only uses the A2 device address bit with the next 2 bits being for memory page addressing. The A2 bit must compare to its corresponding hard-wired input pin. The A1 and A0 pins are no connect.

The 16K does not use any device address bits but instead the 3 bits are used for memory page addressing. These page addressing bits on the 4K, 8K and 16K devices should be considered the most significant bits of the data word address which follows. The A0, A1 and A2 pins are no connect.

The eighth bit of the device address is the read/write operation select bit. A read operation is initiated if this bit is high and a write operation is initiated if this bit is low.

Upon a compare of the device address, the EEPROM will output a zero. If a compare is not made, the chip will return to a standby state.

Write Operations

BYTE WRITE: A write operation requires an 8-bit data word address following the device address word and acknowledgment. Upon receipt of this address, the EEPROM will again respond with a zero and then clock in the first 8-bit data word. Following receipt of the 8-bit data word, the EEPROM will output a zero and the addressing device, such as a microcontroller, must terminate the write sequence with a stop condition. At this time the EEPROM enters an internally timed write cycle, t_{WR}, to the nonvolatile memory. All inputs are disabled during this write cycle and the EEPROM will not respond until the write is complete (refer to Figure 2).

PAGE WRITE: The 1K/2K EEPROM is capable of an 8-byte page write, and the 4K, 8K and 16K devices are capable of 16-byte page writes.

A page write is initiated the same as a byte write, but the microcontroller does not send a stop condition after the first data word is clocked in. Instead, after the EEPROM acknowledges receipt of the first data word, the microcontroller can transmit up to seven (1K/2K) or fifteen (4K, 8K, 16K) more data words. The EEPROM will respond with a zero after each data word received. The microcontroller must terminate the page write sequence with a stop condition (refer to Figure 3).

The data word address lower three (1K/2K) or four (4K, 8K, 16K) bits are internally incremented following the receipt of each data word. The higher data word address bits are not incremented, retaining the memory page row location. When the word address, internally generated, reaches the page boundary, the following byte is placed at the beginning of the same page. If more than eight (1K/2K) or sixteen (4K, 8K, 16K) data words are transmitted to the EEPROM, the data word address will “roll over” and previous data will be overwritten.

ACKNOWLEDGE POLLING: Once the internally timed write cycle has started and the EEPROM inputs are disabled, acknowledge polling can be initiated. This involves sending a start condition followed by the device address word. The read/write bit is representative of the operation desired. Only if the internal write cycle has completed will the EEPROM respond with a zero allowing the read or write sequence to continue.

Read Operations

Read operations are initiated the same way as write operations with the exception that the read/write select bit in the device address word is set to one. There are three read operations: current address read, random address read and sequential read.

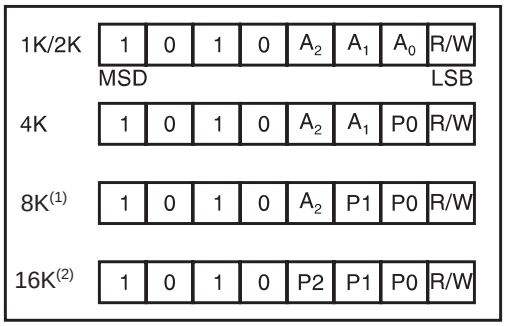
CURRENT ADDRESS READ: The internal data word address counter maintains the last address accessed during the last read or write operation, incremented by one. This address stays valid between operations as long as the chip power is maintained. The address "roll over" during read is from the last byte of the last memory page to the first byte of the first page. The address "roll over" during write is from the last byte of the current page to the first byte of the same page.

Once the device address with the read/write select bit set to one is clocked in and acknowledged by the EEPROM, the current address data word is serially clocked out. The microcontroller does not respond with an input zero but does generate a following stop condition (refer to Figure 4).

RANDOM READ: A random read requires a "dummy" byte write sequence to load in the data word address. Once the device address word and data word address are clocked in and acknowledged by the EEPROM, the microcontroller must generate another start condition. The microcontroller now initiates a current address read by sending a device address with the read/write select bit high. The EEPROM acknowledges the device address and serially clocks out the data word. The microcontroller does not respond with a zero but does generate a following stop condition (refer to Figure 5).

SEQUENTIAL READ: Sequential reads are initiated by either a current address read or a random address read. After the microcontroller receives a data word, it responds with an acknowledge. As long as the EEPROM receives an acknowledge, it will continue to increment the data word address and serially clock out sequential data words. When the memory address limit is reached, the data word address will "roll over" and the sequential read will continue. The sequential read operation is terminated when the microcontroller does not respond with a zero but does generate a following stop condition (refer to Figure 6).

Figure 1. Device Address



Notes: 1. This device is not recommended for new designs. Please refer to AT24C08A.
2. This device is not recommended for new designs. Please refer to AT24C16A.

Figure 2. Byte Write

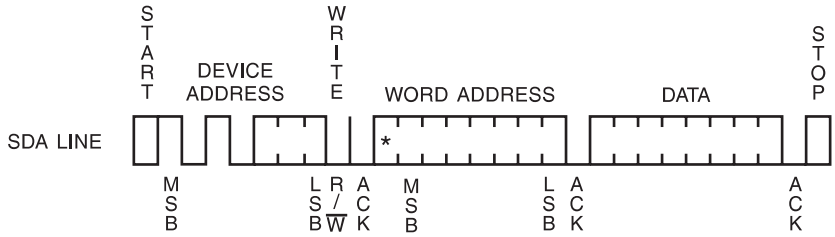
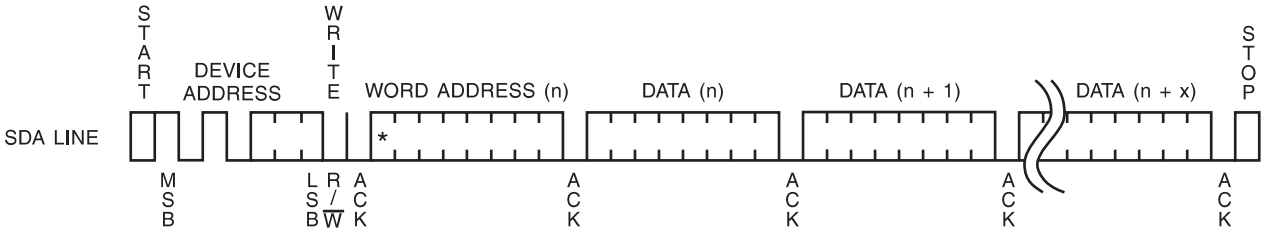


Figure 3. Page Write



(* = DON'T CARE bit for 1K)

Figure 4. Current Address Read

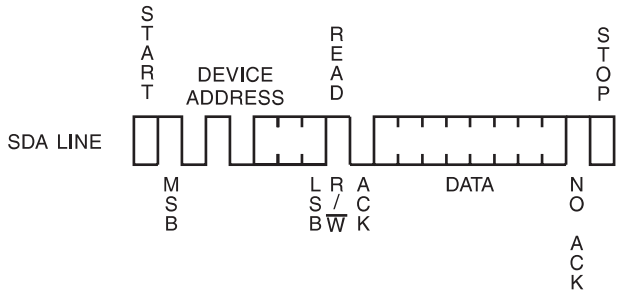
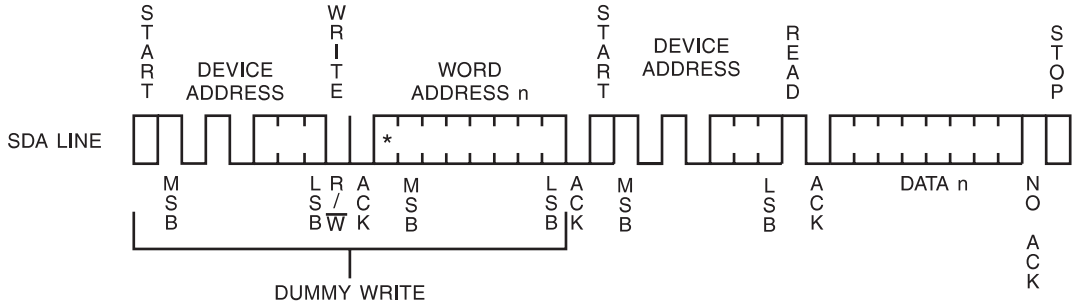
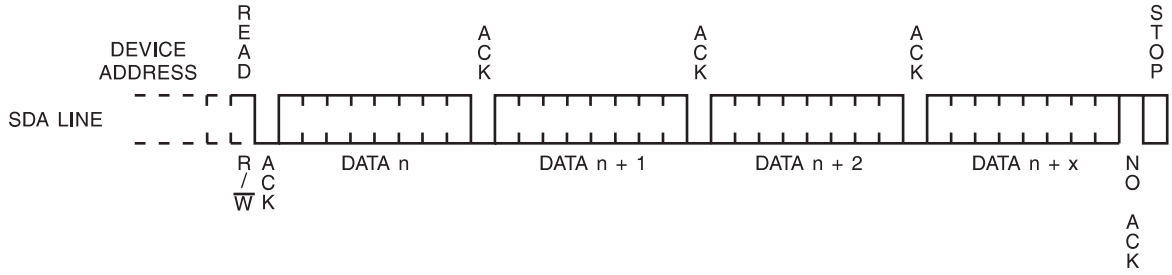


Figure 5. Random Read



(* = DON'T CARE bit for 1K)

Figure 6. Sequential Read



AT24C01A Ordering Information

Ordering Code	Package	Operation Range
AT24C01A-10PI-2.7 AT24C01A-10SI-2.7 AT24C01A-10TI-2.7 AT24C01AY1-10YI-2.7 AT24C01A-10TSI-2.7 AT24C01AR-10TSI-2.7 AT24C01AU3-10UI-2.7	8P3 8S1 8A2 8Y1 5TS1 5TS1 8U3-1	Industrial (-40° C to 85° C)
AT24C01A-10PI-1.8 AT24C01A-10SI-1.8 AT24C01A-10TI-1.8 AT24C01AY1-10YI-1.8 AT24C01A-10TSI-1.8 AT24C01AU3-10UI-1.8	8P3 8S1 8A2 8Y1 5TS1 8U3-1	Industrial (-40° C to 85° C)
AT24C01A-10SU-2.7 AT24C01A-10SU-1.8 AT24C01A-10TU-2.7 AT24C01A-10TU-1.8	8S1 8S1 8A2 8A2	Lead-free/Halogen-free/ Industrial Temperature (-40° C to 85° C)
AT24C01A-10SE-2.7	8S1	High Grade/Extended Temp (-40° C to 125° C)
AT24C01A-10SQ-2.7	8S1	Lead-free/Halogen-free/ High Grade/Extended Temp (-40° C to 125° C)

Note: For 2.7V devices used in the 4.5V to 5.5V range, please refer to performance values in the AC and DC characteristics table.

Package Type	
8P3	8-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP)
8S1	8-lead, 0.150" Wide, Plastic Gull Wing Small Outline (JEDEC SOIC)
8A2	8-lead, 0.170" Wide, Thin Shrink Small Outline Package (TSSOP)
8Y1	8-lead, 4.90 mm x 3.00 mm Body, Dual Footprint, Non-leaded, Miniature Array Package (MAP)
5TS1	5-lead, 2.90 mm x 1.60 mm Body, Plastic Thin Shrink Small Outline Package (SOT23)
8U3-1	8-ball, die Ball Grid Away Package (dBGA2)
Options	
-2.7	Low-voltage (2.7V to 5.5V)
-1.8	Low-voltage (1.8V to 5.5V)
R	Rotated Pinout

AT24C02 Ordering Information

Ordering Code	Package	Operation Range
AT24C02-10PI-2.7	8P3	Industrial (-40° C to 85° C)
AT24C02N-10SI-2.7	8S1	
AT24C02-10TI-2.7	8A2	
AT24C02Y1-10YI-2.7	8Y1	
AT24C02-10TSI-2.7	5TS1	
AT24C02U3-10UI-2.7	8U3-1	
AT24C02-10PI-1.8	8P3	Industrial (-40° C to 85° C)
AT24C02N-10SI-1.8	8S1	
AT24C02-10TI-1.8	8A2	
AT24C02Y1-10YI-1.8	8Y1	
AT24C02-10TSI-1.8	5TS1	
AT24C02U3-10UI-1.8	8U3-1	
AT24C02N-10SU-2.7	8S1	Lead-free/Halogen-free/ Industrial Temperature (-40° C to 85° C)
AT24C02N-10SU-1.8	8S1	
AT24C02-10TU-2.7	8A2	
AT24C02-10TU-1.8	8A2	
AT24C02N-10SE-2.7	8S1	High Grade/Extended Temp (-40° C to 125° C)
AT24C02N-10SQ-2.7	8S1	Lead-free/Halogen-free/ High Grade/Extended Temp (-40° C to 125° C)

Note: For 2.7V devices used in the 4.5V to 5.5V range, please refer to performance values in the AC and DC characteristics table.

Package Type	
8P3	8-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP)
8S1	8-lead, 0.150" Wide, Plastic Gull Wing Small Outline (JEDEC SOIC)
8A2	8-lead, 0.170" Wide, Thin Shrink Small Outline Package (TSSOP)
8Y1	8-lead, 4.90 mm x 3.00 mm Body, Dual Footprint, Non-leaded, Miniature Array Package (MAP)
5TS1	5-lead, 2.90 mm x 1.60 mm Body, Plastic Thin Shrink Small Outline Package (SOT23)
8U3-1	8-ball, die Ball Grid Away Package (dBGA2)
Options	
-2.7	Low-voltage (2.7V to 5.5V)
-1.8	Low-voltage (1.8V to 5.5V)

AT24C04 Ordering Information

Ordering Code	Package	Operation Range
AT24C04-10PI-2.7 AT24C04N-10SI-2.7 AT24C04-10TI-2.7 AT24C04Y1-10YI-2.7 AT24C04-10TSI-2.7 AT24C04U3-10UI-2.7	8P3 8S1 8A2 8Y1 5TS1 8U3-1	Industrial (-40° C to 85° C)
AT24C04-10PI-1.8 AT24C04N-10SI-1.8 AT24C04-10TI-1.8 AT24C04Y1-10YI-1.8 AT24C04-10TSI-1.8 AT24C04U3-10UI-1.8	8P3 8S1 8A2 8Y1 5TS1 8U3-1	Industrial (-40° C to 85° C)
AT24C04N-10SU-2.7 AT24C04N-10SU-1.8 AT24C04-10TU-2.7 AT24C04-10TU-1.8	8S1 8S1 8A2 8A2	Lead-free/Halogen-free/ Industrial Temperature (-40° C to 85° C)
AT24C04N-10SE-2.7	8S1	High Grade/Extended Temp (-40° C to 125° C)
AT24C04N-10SQ-2.7	8S1	Lead-free/Halogen-free/ High Grade/Extended Temp (-40° C to 125° C)

Note: For 2.7V devices used in the 4.5V to 5.5V range, please refer to performance values in the AC and DC characteristics table.

Package Type	
8P3	8-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP)
8S1	8-lead, 0.150" Wide, Plastic Gull Wing Small Outline (JEDEC SOIC)
8A2	8-lead, 0.170" Wide, Thin Shrink Small Outline Package (TSSOP)
8Y1	8-lead, 4.90 mm x 3.00 mm Body, Dual Footprint, Non-leaded, Miniature Array Package (MAP)
5TS1	5-lead, 2.90 mm x 1.60 mm Body, Plastic Thin Shrink Small Outline Package (SOT23)
8U3-1	8-ball, die Ball Grid Away Package (dBGA2)

Package Type	
Options	
-2.7	Low-voltage (2.7V to 5.5V)
-1.8	Low-voltage (1.8V to 5.5V)

AT24C08[2] Ordering Information

Ordering Code	Package	Operation Range
AT24C08-10PI-2.7 AT24C08N-10SI-2.7 AT24C08-10TI-2.7	8P3 8S1 8A2	Industrial (-40° C to 85° C)
AT24C08-10PI-1.8 AT24C08N-10SI-1.8 AT24C08-10TI-1.8	8P3 8S1 8A2	Industrial (-40° C to 85° C)

Notes: 1. For 2.7V devices used in the 4.5V to 5.5V range, please refer to performance values in the AC and DC characteristics table.
2. This device is not recommended for new designs. Please refer to AT24C08A.

Package Type	
8P3	8-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP)
8S1	8-lead, 0.150" Wide, Plastic Gull Wing Small Outline (JEDEC SOIC)
8A2	8-lead, 0.170" Wide, Thin Shrink Small Outline Package (TSSOP)
Options	
-2.7	Low-voltage (2.7V to 5.5V)
-1.8	Low-voltage (1.8V to 5.5V)

AT24C16[2] Ordering Information

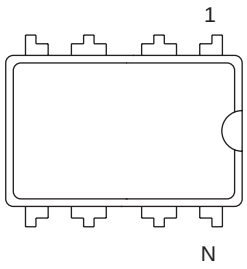
Ordering Code	Package	Operation Range
AT24C16-10PI-2.7 AT24C16N-10SI-2.7 AT24C16-10TI-2.7	8P3 8S1 8A2	Industrial (-40° C to 85° C)
AT24C16-10PI-1.8 AT24C16N-10SI-1.8 AT24C16-10TI-1.8	8P3 8S1 8A2	Industrial (-40° C to 85° C)

Notes: 1. For 2.7V devices used in the 4.5V to 5.5V range, please refer to performance values in the AC and DC characteristics table.
2. This device is not recommended for new designs. Please refer to AT24C16A.

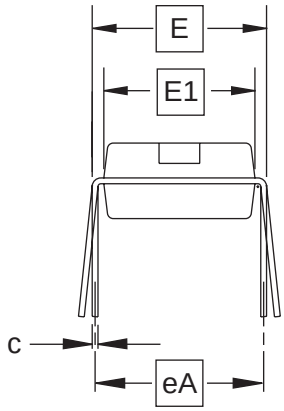
Package Type	
8P3	8-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP)
8S1	8-lead, 0.150" Wide, Plastic Gull Wing Small Outline (JEDEC SOIC)
8A2	8-lead, 0.170" Wide, Thin Shrink Small Outline Package (TSSOP)
Options	
-2.7	Low-voltage (2.7V to 5.5V)
-1.8	Low-voltage (1.8V to 5.5V)

Packaging Information

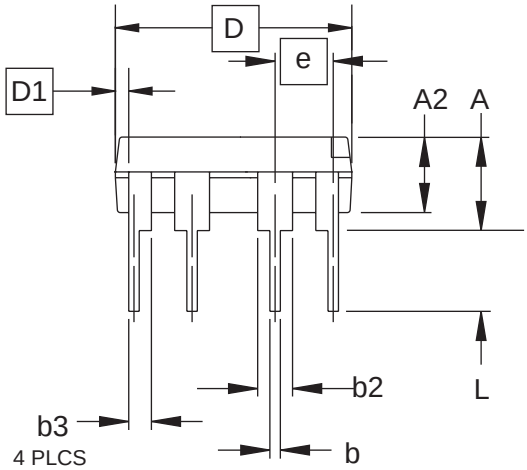
8P3 – PDIP



Top View



End View



Side View

COMMON DIMENSIONS
(Unit of Measure = inches)

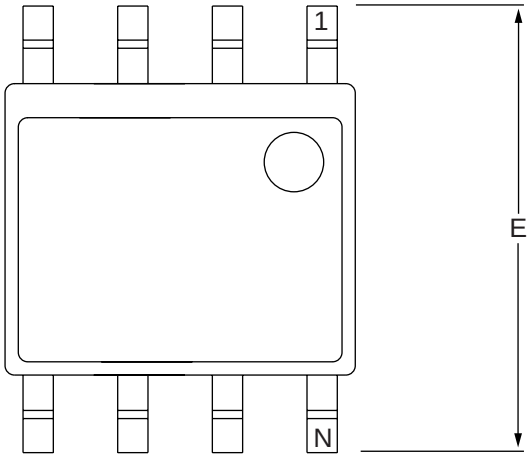
SYMBOL	MIN	NOM	MAX	NOTE
A			0.210	2
A2	0.115	0.130	0.195	
b	0.014	0.018	0.022	5
b2	0.045	0.060	0.070	6
b3	0.030	0.039	0.045	6
c	0.008	0.010	0.014	
D	0.355	0.365	0.400	3
D1	0.005			3
E	0.300	0.310	0.325	4
E1	0.240	0.250	0.280	3
e	0.100 BSC			
eA	0.300 BSC			4
L	0.115	0.130	0.150	2

Notes:
1. This drawing is for general information only; refer to JEDEC Drawing MS-001, Variation BA for additional information.
2. Dimensions A and L are measured with the package seated in JEDEC seating plane Gauge GS-3.
3. D, D1 and E1 dimensions do not include mold Flash or protrusions. Mold Flash or protrusions shall not exceed 0.010 inch.
4. E and eA measured with the leads constrained to be perpendicular to datum.
5. Pointed or rounded lead tips are preferred to ease insertion.
6. b2 and b3 maximum dimensions do not include Dambar protrusions. Dambar protrusions shall not exceed 0.010 (0.25 mm).

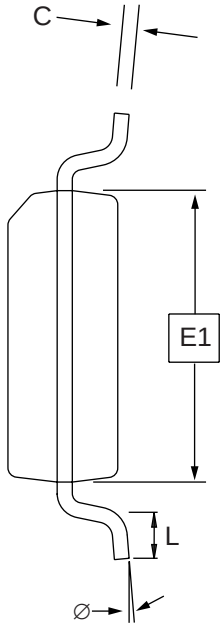
01/09/02

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**8P3**, 8-lead, 0.300" Wide Body, Plastic Dual In-line Package (PDIP)	8P3	B

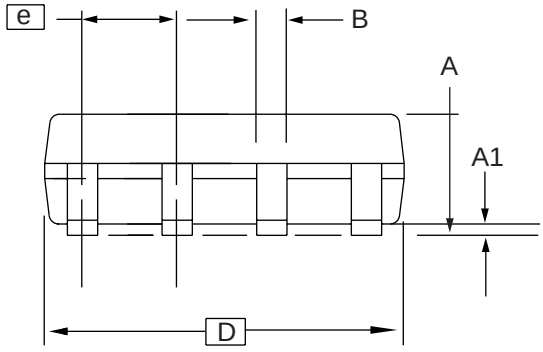
8S1 – JEDEC SOIC



Top View



End View



Side View

COMMON DIMENSIONS
(Unit of Measure = mm)

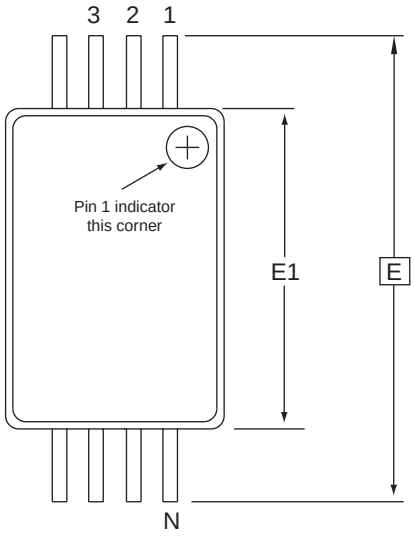
SYMBOL	MIN	NOM	MAX	NOTE
A	1.35	–	1.75	
A1	0.10	–	0.25	
b	0.31	–	0.51	
C	0.17	–	0.25	
D	4.80	–	5.00	
E1	3.81	–	3.99	
E	5.79	–	6.20	
e	1.27 BSC			
L	0.40	–	1.27	
∅	0°	–	8°	

Note: These drawings are for general information only. Refer to JEDEC Drawing MS-012, Variation AA for proper dimensions, tolerances, datums, etc.

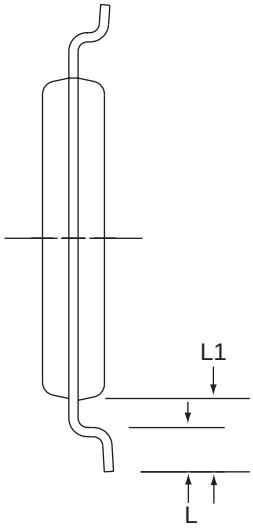
10/7/03

1150 E. Cheyenne Mtn. Blvd. Colorado Springs, CO 80906	**TITLE** **8S1**, 8-lead (0.150" Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC)	**DRAWING NO.** 8S1	**REV.** B

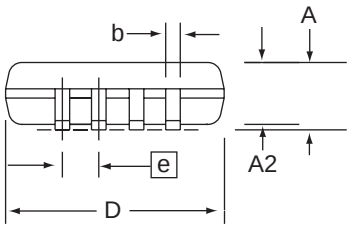
8A2 – TSSOP



Top View



End View



Side View

COMMON DIMENSIONS
(Unit of Measure = mm)

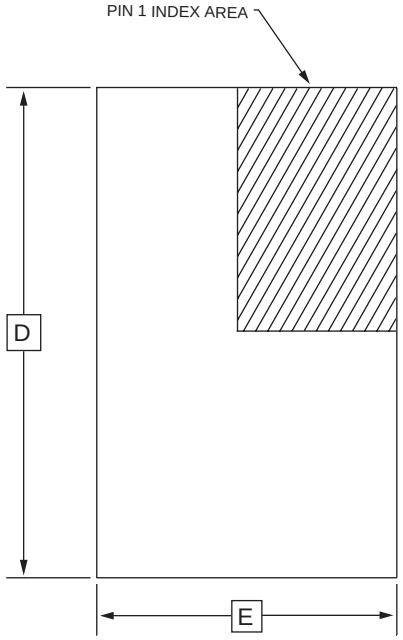
SYMBOL	MIN	NOM	MAX	NOTE
D	2.90	3.00	3.10	2, 5
E		6.40 BSC		
E1	4.30	4.40	4.50	3, 5
A	–	–	1.20	
A2	0.80	1.00	1.05	
b	0.19	–	0.30	4
e		0.65 BSC		
L	0.45	0.60	0.75	
L1		1.00 REF		

Notes:
1. This drawing is for general information only. Refer to JEDEC Drawing MO-153, Variation AA, for proper dimensions, tolerances, datums, etc.
2. Dimension D does not include mold Flash, protrusions or gate burrs. Mold Flash, protrusions and gate burrs shall not exceed 0.15 mm (0.006 in) per side.
3. Dimension E1 does not include inter-lead Flash or protrusions. Inter-lead Flash and protrusions shall not exceed 0.25 mm (0.010 in) per side.
4. Dimension b does not include Dambar protrusion. Allowable Dambar protrusion shall be 0.08 mm total in excess of the b dimension at maximum material condition. Dambar cannot be located on the lower radius of the foot. Minimum space between protrusion and adjacent lead is 0.07 mm.
5. Dimension D and E1 to be determined at Datum Plane H.

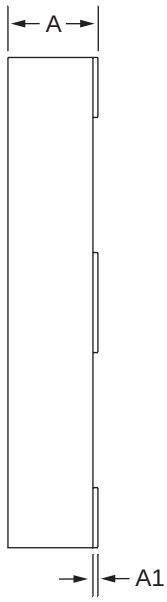
5/30/02

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**8A2**, 8-lead, 4.4 mm Body, Plastic Thin Shrink Small Outline Package (TSSOP)	8A2	B

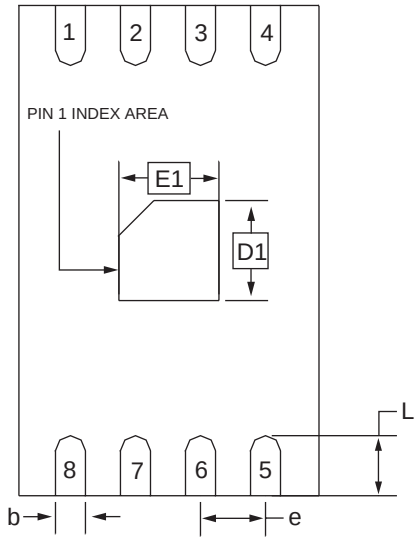
8Y1 – MAP



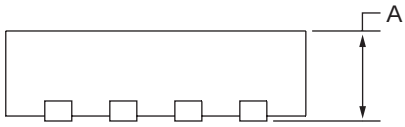
Top View



End View



Bottom View



Side View

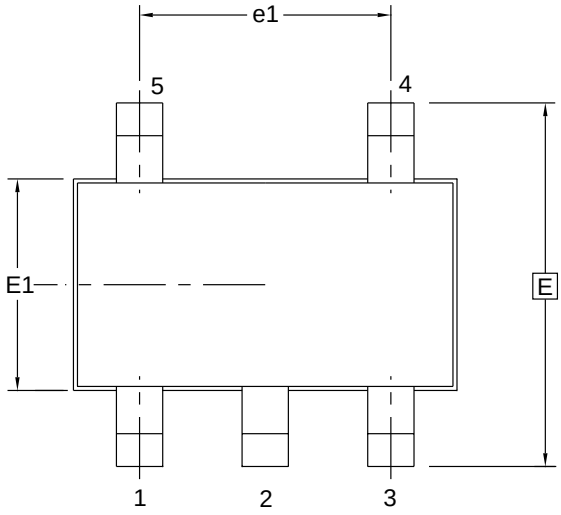
COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	0.90	
A1	0.00	–	0.05	
D	4.70	4.90	5.10	
E	2.80	3.00	3.20	
D1	0.85	1.00	1.15	
E1	0.85	1.00	1.15	
b	0.25	0.30	0.35	
e	0.65 TYP			
L	0.50	0.60	0.70	

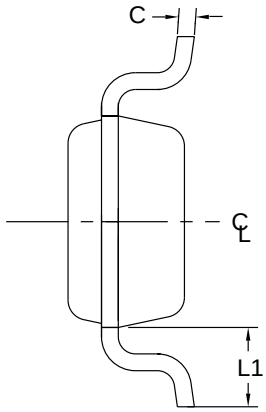
2/28/03

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**8Y1,** 8-lead (4.90 x 3.00 mm Body) MSOP Array Package (MAP) Y1	8Y1	C

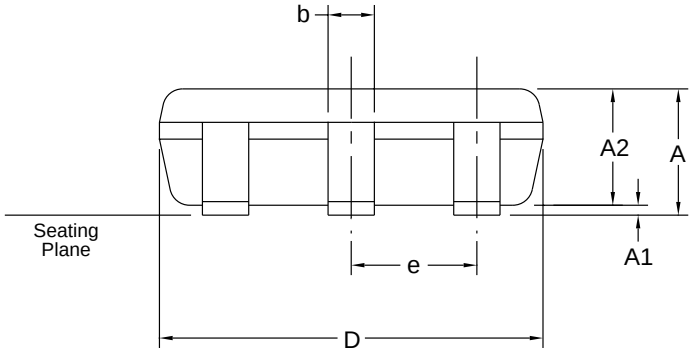
5TS1 – SOT23



Top View



End View



Side View

COMMON DIMENSIONS
(Unit of Measure = mm)

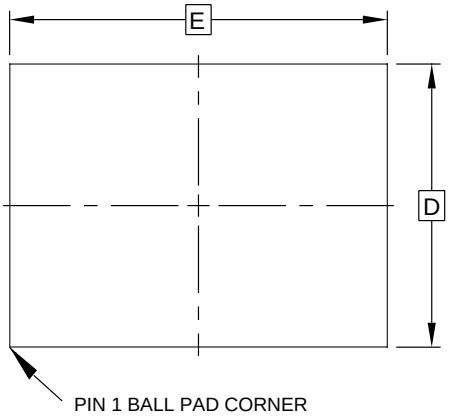
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.10	
A1	0.00	–	0.10	
A2	0.70	0.90	1.00	
c	0.08	–	0.20	4
D		2.90 BSC		2, 3
E		2.80 BSC		2, 3
E1		1.60 BSC		2, 3
L1		0.60 REF		
e		0.95 BSC		
e1		1.90 BSC		
b	0.30	–	0.50	4, 5

NOTES:
1. This drawing is for general information only. Refer to JEDEC Drawing MO-193, Variation AB, for additional information.
2. Dimension D does not include mold flash, protrusions, or gate burrs. Mold flash, protrusions, or gate burrs shall not exceed 0.15 mm per end. Dimension E1 does not include interlead flash or protrusion. Interlead flash or protrusion shall not exceed 0.15 mm per side.
3. The package top may be smaller than the package bottom. Dimensions D and E1 are determined at the outermost extremes of the plastic body exclusive of mold flash, tie bar burrs, gate burrs, and interlead flash, but including any mismatch between the top and bottom of the plastic body.
4. These dimensions apply to the flat section of the lead between 0.08 mm and 0.15 mm from the lead tip.
5. Dimension "b" does not include Dambar protrusion. Allowable Dambar protrusion shall be 0.08 mm total in excess of the "b" dimension at maximum material condition. The Dambar cannot be located on the lower radius of the foot. Minimum space between protrusion and an adjacent lead shall not be less than 0.07 mm.

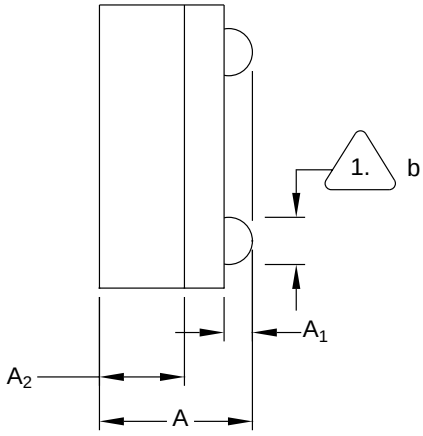
6/25/03

	TITLE	DRAWING NO.	REV.
1150 E. Cheyenne Mtn. Blvd. Colorado Springs, CO 80906	**5TS1,** 5-lead, 1.60 mm Body, Plastic Thin Shrink Small Outline Package (SHRINK SOT)	PO5TS1	A

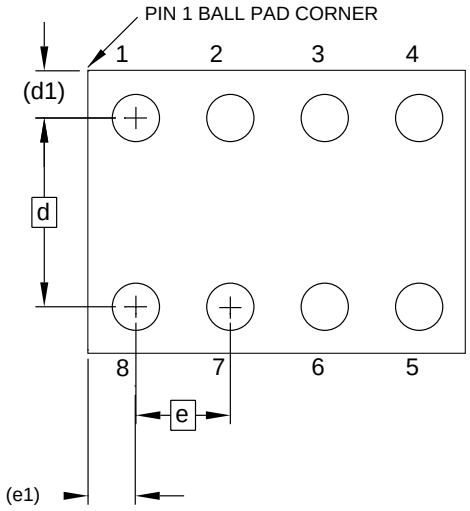
8U3-1 – dBGA2



Top View



Side View



Bottom View
8 SOLDER BALLS

1. Dimension 'b' is measured at the maximum solder ball diameter.

This drawing is for general information only.

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.71	0.81	0.91	
A1	0.10	0.15	0.20	
A2	0.40	0.45	0.50	
b	0.20	0.25	0.30	
D		1.50 BSC		
E		2.00 BSC		
e		0.50 BSC		
e1		0.25 REF		
d		1.00 BSC		
d1		0.25 REF		

6/24/03

1150 E. Cheyenne Mtn. Blvd. Colorado Springs, CO 80906	**TITLE** **8U3-1,** 8-ball, 1.50 x 2.00 mm Body, 0.50 mm pitch, Small Die Ball Grid Array Package (dBGA2)	**DRAWING NO.** PO8U3-1	**REV.** A

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

0180Q–SEEPR–1/04 xM

直流馬達感測器

1 直流馬達種類

直流馬達可分爲兩族，其一是功率額定超過 lhp 的整數馬力型馬達：其二是功率額定低於 lhp 之分數馬力型馬達。

分數馬達型馬達依各種方式來分類：

(一) 依激磁方式

(a) 永磁式
(b) 直串激式
(c) 分患激式
(d) 並激式
(e) 複激式

(二) 責務率(Duty Cycle)

(a) 連續運轉
(b) 間歇運轉

(三) 冷卻方式或密閉方式

(a) 風扇冷卻
(b) 無冷卻
(c) 密閉結構
(d) 開放結構

2 線繞馬達（Would-Field Mofors）

（一） 直串激式(Straight-SerieS)

直串激式馬達的線圈與馬達電樞串聯，起動時，由於馬達轉速較慢故反電動勢較小，所施與的電壓均跨於電樞和激磁上，故超動電流相當大。直串激式馬達往往使用於須要較大起動轉矩的地方，典型的直串激式馬達表示準及速度一轉矩曲線如圖 6-1。由圖得知轉矩幾乎和電流的平方成正比。

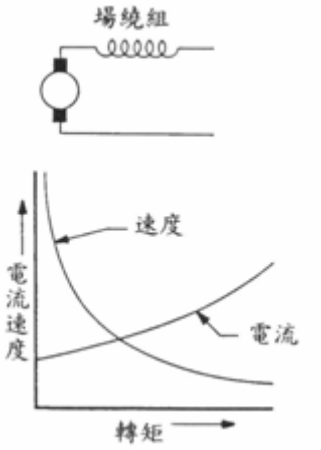


圖 6-1 直串激式馬達

（二）分串激式(Split-Series)

分串激式馬達和直串激式馬達非常相似，惟一的不同點是它具有兩組對立的場線圖。這種特性使它更方便地容許極性快速交換，因此它可以迅速地被改變運轉方向。其主要應用於飛機上之致動器。典型的分串激式馬達之電路符號與特性曲線，如圖 6-2。

（三）並激式馬達(Shunt)

並激式馬達的電樞和磁場線圈並聯連接，電樞電流由負載的狀況所決定。其主要應用於定速和變速的場合中。其轉矩一速度特性曲線在較高電流準位時呈非線性。如圖 6-3 。

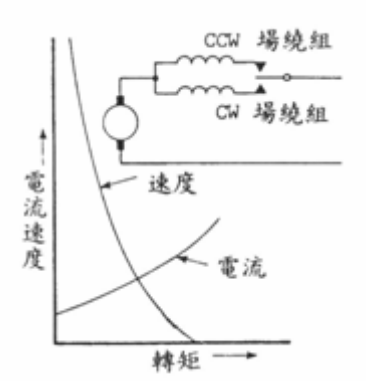


圖 6-2 分串激式馬達

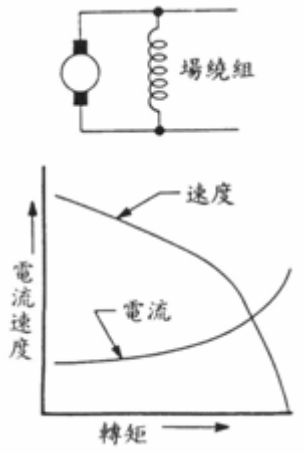


圖 6-3 典型的並激式馬達

（四）複激式(Compound)

複激式馬達同時具有串激和並激兩組場繞組。馬達的串激場繞組強度，並激場強度者，稱爲和複激式馬達。反之， 則稱爲加成差複激式馬達。通常，小型的複激式馬達具有較強的並激場和輔助馬達起動的較弱串激場。其特性曲線如圖 6-4 。

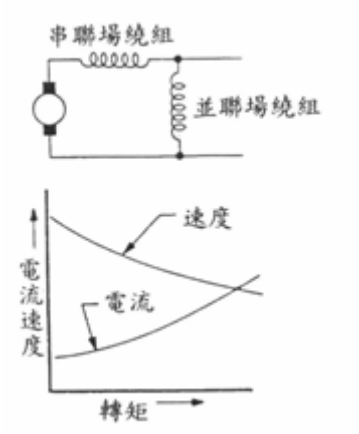


圖 6-4

3 直流馬達速度控制

原理：直流馬達的速度與流過其線圈之電流成正比。所以，我們設計一電晶體使它在“工作區”工作，由外部類比電壓來驅動之。如此，流過直流馬達的電樞電流便會隨之變大、變小，進而馬達的速度便會被控制。如圖 6-5 。

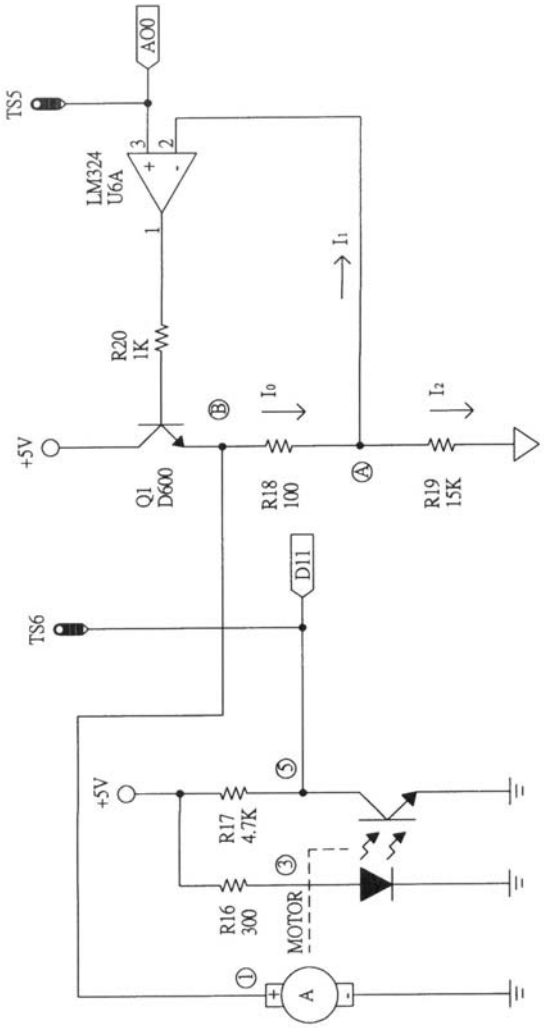


圖 6-5

電路說明：

(1) LM324，R20，Q1，R18，R19 構成一個線性電流(電壓)直流馬达驅動電路，

$$V_- = V+ \text{（OP 的虛接地）}$$

$$V_- = V_A$$

$$I_2 = I_0 \text{（}\because I_1 = 0\text{，OP 虛接地）}$$

$$I_2 = \frac{V_A}{R_{19}}$$

$$V_B = I_2 \times (R_{18} + R_{19})$$

$$= \frac{V_A}{R_{19}} (R_{18} + R_{19})$$

$$= V_A \left(1 + \frac{R_{18}}{R_{19}}\right)$$

(2) D14 是經馬達上的轉盤與光插斷器所產生的脈波信號，這樣可以量測馬達的速度、加速度、角度。

InvenSense Inc.
1197 Borregas Ave, Sunnyvale, CA 94089 U.S.A.
Tel: +1 (408) 988-7339 Fax: +1 (408) 988-8104
Website: www.invensense.com

Document Number: PS-ITG-3200A-00-01.4
Revision: 1.4
Release Date: 03/30/2010

ITG-3200
Product Specification
Revision 1.4

CONTENTS

1 Document Information

1.1 Revision History

Revision Date	Revision	Description
10/23/09	1.0	Initial Release
10/28/09	1.1	Edits for readability
02/12/2010	1.2	• Changed full-scale range and sensitivity scale factor (Sections 2, 3.1, 5.3, and 8.3) • Changed sensitivity scale factor variation over temperature (Section 3.1) • Changed total RMS noise spec (Section 3.1) • Added range for temperature sensor (Section 3.1) • Updated VDD Power-Supply Ramp Rate specification (Sections 3.2 and 4.4) • Added VLOGIC Voltage Range condition (Section 3.2) • Added VLOGIC Reference Voltage Ramp Rate specification (Sections 3.2 and 4.4) • Updated Start-Up Time for Register Read/Write specification (Section 3.2) • Updated Input logic levels for AD0 and CLKIN (Section 3.2) • Updated Level I_{OL} specifications for the I^2C interface (Section 3.3) • Updated Frequency Variation Over Temperature specification for internal clock source (Section 3.4) • Updated VLOGIC conditions for I^2C Characterization (Section 3.5) • Updated ESD specification (Section 3.6) • Added termination requirements for CLKIN if unused (Section 4.1) • Added recommended power-on procedure diagram (Section 4.4) • Changed DLPF_CFG setting 7 to reserved (Section 8.3) • Changed Reflow Specification description (Section 9.12) • Removed errata specifications
03/05/2010	1.3	• Updated temperature sensor linearity spec (Section 3.1) • Updated VDD Power-Supply Ramp Rate timing figure (Sections 3.2 and 4.4) • Updated VLOGIC Reference Voltage timing figure (Section 4.4) • Added default values to registers (all of Section 8) • Updated FS_SEL description (Section 8.3) • Updated package outline drawing and dimensions (Section 9.2) • Updated Reliability (Section 10.1 and 10.2) • Removed Environmental Compliance (Section 11)
03/30/2010	1.4	• Removed confidentiality mark

1.2 Purpose and Scope

This document is a preliminary product specification, providing a description, specifications, and design related information for the ITG-3200™. Electrical characteristics are based upon simulation results and limited characterization data of advanced samples only. Specifications are subject to change without notice. Final specifications will be updated based upon characterization of final silicon.

1.3 Product Overview

The ITG-3200 is the world's first single-chip, digital-output, 3-axis MEMS gyro IC optimized for gaming, 3D mice, and 3D remote control applications. The part features enhanced bias and sensitivity temperature stability, reducing the need for user calibration. Low frequency noise is lower than previous generation devices, simplifying application development and making for more-responsive remote controls.

The ITG-3200 features three 16-bit analog-to-digital converters (ADCs) for digitizing the gyro outputs, a user-selectable internal low-pass filter bandwidth, and a Fast-Mode I^2C (400kHz) interface. Additional features include an embedded temperature sensor and a 2% accurate internal oscillator. This breakthrough in gyroscope technology provides a dramatic 67% package size reduction, delivers a 50% power reduction, and has inherent cost advantages compared to competing multi-chip gyro solutions.

By leveraging its patented and volume-proven Nasiri-Fabrication platform, which integrates MEMS wafers with companion CMOS electronics through wafer-level bonding, InvenSense has driven the ITG-3200 package size down to a revolutionary footprint of 4x4x0.9mm (QFN), while providing the highest performance, lowest noise, and the lowest cost semiconductor packaging required for handheld consumer electronic devices. The part features a robust 10,000*g* shock tolerance, as required by portable consumer equipment.

For power supply flexibility, the ITG-3200 has a separate VLOGIC reference pin, in addition to its analog supply pin, VDD, which sets the logic levels of its I^2C interface. The VLOGIC voltage may be anywhere from 1.71V min to VDD max.

1.4 Applications

- Motion-enabled game controllers
- Motion-based portable gaming
- Motion-based 3D mice and 3D remote controls
- "No Touch" UI
- Health and sports monitoring

2 Features

The ITG-3200 triple-axis MEMS gyroscope includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyros) on one integrated circuit with a sensitivity of 14.375 LSBs per °/sec and a full-scale range of ±2000°/sec
- Three integrated 16-bit ADCs provide simultaneous sampling of gyros while requiring no external multiplexer
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Low frequency noise lower than previous generation devices, simplifying application development and making for more-responsive motion processing
- Digitally-programmable low-pass filter
- Low 6.5mA operating current consumption for long battery life
- Wide VDD supply voltage range of 2.1V to 3.6V
- Flexible VLOGIC reference voltage allows for I^2C interface voltages from 1.71V to VDD
- Standby current: 5µA
- Smallest and thinnest package for portable devices (4x4x0.9mm QFN)
- No high pass filter needed
- Turn on time: 50ms
- Digital-output temperature sensor
- Factory calibrated scale factor
- 10,000 *g* shock tolerant
- Fast Mode I^2C (400kHz) serial interface
- On-chip timing generator clock frequency is accurate to +/-2% over full temperature range
- Optional external clock inputs of 32.768kHz or 19.2MHz to synchronize with system clock
- MEMS structure hermetically sealed and bonded at wafer level
- RoHS and Green compliant

3 Electrical Characteristics

3.1 Sensor Specifications

Typical Operating Circuit of Section 4.2, VDD = 2.5V, VLOGIC = 1.71V to VDD, T_A=25°C.

Parameter	Conditions	Min	Typical	Max	Unit	Note
GYRO SENSITIVITY						
Full-Scale Range	FS_SEL=3		±2000		º/s	4
Gyro ADC Word Length			16		Bits	3
Sensitivity Scale Factor	FS_SEL=3		14.375		LSB/(º/s)	3
Sensitivity Scale Factor Tolerance	25°C	-6		+6	%	1
Sensitivity Scale Factor Variation Over Temperature			±10		%	2
Nonlinearity	Best fit straight line; 25°C		0.2		%	6
Cross-Axis Sensitivity			2		%	6
GYRO ZERO-RATE OUTPUT (ZRO)						
Initial ZRO Tolerance			±40		º/s	1
ZRO Variation Over Temperature	-40°C to +85°C		±40		º/s	2
Power-Supply Sensitivity (1-10Hz)	Sine wave, 100mVpp; VDD=2.2V		0.2		º/s	5
Power-Supply Sensitivity (10 - 250Hz)	Sine wave, 100mVpp; VDD=2.2V		0.2		º/s	5
Power-Supply Sensitivity (250Hz - 100kHz)	Sine wave, 100mVpp; VDD=2.2V		4		º/s	5
Linear Acceleration Sensitivity	Static		0.1		º/s/*g*	6
GYRO NOISE PERFORMANCE	**FS_SEL=3**					
Total RMS noise	100Hz LPF (DLPFCFG=2)		0.38		º/s-rms	1
Rate Noise Spectral Density	At 10Hz		0.03		º/s/√Hz	2
GYRO MECHANICAL FREQUENCIES						
X-Axis		30	33	36	kHz	1
Y-Axis		27	30	33	kHz	1
Z-Axis		24	27	30	kHz	1
Frequency Separation	Between any two axes	1.7			kHz	1
GYRO START-UP TIME	**DLPFCFG=0**					
ZRO Settling	to ±1º/s of Final		50		ms	6
TEMPERATURE SENSOR						
Range			-30 to +85		ºC	2
Sensitivity			280		LSB/ºC	2
Temperature Offset	35ºC		-13,200		LSB	1
Initial Accuracy	35ºC		TBD		°C	
Linearity	Best fit straight line (-30°C to +85°C)		±1		°C	2, 5
TEMPERATURE RANGE						
Specified Temperature Range		-40		85	ºC	

Notes:

1. Tested in production
2. Based on characterization of 30 pieces over temperature on evaluation board or in socket
3. Based on design, through modeling and simulation across PVT
4. Typical. Randomly selected part measured at room temperature on evaluation board or in socket
5. Based on characterization of 5 pieces over temperature
6. Tested on 5 parts at room temperature

3.2 Electrical Specifications

Typical Operating Circuit of Section 4.2, VDD = 2.5V, VLOGIC = 1.71V to VDD, T_A=25°C.

Parameters	Conditions	Min	Typical	Max	Units	Notes
VDD POWER SUPPLY						
Operating Voltage Range		2.1		3.6	V	2
Power-Supply Ramp Rate	Monotonic ramp. Ramp rate is 10% to 90% of the final value (see Figure in Section 4.4)	0		5	ms	2
Normal Operating Current			6.5		mA	1
Sleep Mode Current			5		µA	5
VLOGIC REFERENCE VOLTAGE						
Voltage Range	VLOGIC must be ≤VDD at all times	1.71		VDD	V	
VLOGIC Ramp Rate	Monotonic ramp. Ramp rate is 10% to 90% of the final value (see Figure in Section 4.4)			1	ms	6
Normal Operating Current			100		µA	
START-UP TIME FOR REGISTER READ/WRITE			20		ms	5
I^2C ADDRESS	AD0 = 0		1101000			6
	AD0 = 1		1101001			6
DIGITAL INPUTS (AD0, CLKIN)						
V_{IH}, High Level Input Voltage		0.9*VLOGIC			V	5
V_{IL}, Low Level Input Voltage				0.1*VLOGIC	V	5
C_I, Input Capacitance				5	pF	7
DIGITAL OUTPUT (INT)						
V_{OH}, High Level Output Voltage	OPEN=0, Rload=1MΩ	0.9*VLOGIC			V	2
V_{OL}, Low Level Output Voltage	OPEN=0, Rload=1MΩ			0.1*VLOGIC	V	2
$V_{OL.INT1}$, INT Low-Level Output Voltage	OPEN=1, 0.3mA sink current			0.1	V	2
Output Leakage Current	OPEN=1		100		nA	4
t_{INT}, INT Pulse Width	LATCH_INT_EN=0		50		µs	4

Notes:

1. Tested in production
2. Based on characterization of 30 pieces over temperature on evaluation board or in socket
4. Typical. Randomly selected part measured at room temperature on evaluation board or in socket
5. Based on characterization of 5 pieces over temperature
6. Guaranteed by design

3.3 Electrical Specifications, continued

Typical Operating Circuit of Section 4.2, VDD = 2.5V, VLOGIC = 1.71V to VDD, T_A=25°C.

Parameters	Conditions	Typical	Units	Notes
I^2C I/O (SCL, SDA)				
V_{IL}, LOW-Level Input Voltage		-0.5 to 0.3*VLOGIC	V	2
V_{IH}, HIGH-Level Input Voltage		0.7*VLOGIC to VLOGIC + 0.5V	V	2
V_{hys}, Hysteresis		0.1*VLOGIC	V	2
V_{OL1}, LOW-Level Output Voltage	3mA sink current	0 to 0.4	V	2
I_{OL}, LOW-Level Output Current	V_{OL} = 0.4V V_{OL} = 0.6V	3 6	mA mA	2 2
Output Leakage Current		100	nA	4
t_{of}, Output Fall Time from V_{IHmax} to V_{ILmax}	Cb bus cap. in pF	20+0.1Cb to 250	ns	2
C_I, Capacitance for Each I/O pin		10	pF	5

Notes:

2. Based on characterization of 5 pieces over temperature.
4. Typical. Randomly selected part measured at room temperature on evaluation board or in socket
5. Guaranteed by design

3.4 Electrical Specifications, continued

Typical Operating Circuit of Section 4.2, VDD = 2.5V, VLOGIC = 1.71V to VDD, T_A=25°C.

Parameters	Conditions	Min	Typical	Max	Units	Notes
INTERNAL CLOCK SOURCE	**CLKSEL=0, 1, 2, or 3**					
Sample Rate, Fast	DLPFCFG=0 SAMPLERATEDIV = 0		8		kHz	4
Sample Rate, Slow	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1		kHz	4
Clock Frequency Initial Tolerance	CLKSEL=0, 25°C	-2		+2	%	1
	CLKSEL=1,2,3; 25°C	-1		+1	%	1
Frequency Variation over Temperature	CLKSEL=0		-15 to +10		%	2
	CLKSEL=1,2,3		+/-1		%	2
PLL Settling Time	CLKSEL=1,2,3		1		ms	3
EXTERNAL 32.768kHz CLOCK	**CLKSEL=4**					
External Clock Frequency			32.768		kHz	3
External Clock Jitter	Cycle-to-cycle rms		1 to 2		µs	3
Sample Rate, Fast	DLPFCFG=0 SAMPLERATEDIV = 0		8.192		kHz	3
Sample Rate, Slow	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1.024		kHz	3
PLL Settling Time			1		ms	3
EXTERNAL 19.2MHz CLOCK	**CLKSEL=5**					
External Clock Frequency			19.2		MHz	3
Sample Rate, Fast	DLPFCFG=0 SAMPLERATEDIV = 0		8		kHz	3
Sample Rate, Slow	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1		kHz	3
PLL Settling Time			1		ms	3
Charge Pump Clock Frequency						
Frequency	1st Stage, 25°C		8.5		MHz	5
	2nd Stage, 25°C		68		MHz	5
	Over temperature		+/-15		%	5

Notes:

1. Tested in production
2. Based on characterization of 30 pieces over temperature on evaluation board or in socket
3. Based on design, through modeling and simulation across PVT
4. Typical. Randomly selected part measured at room temperature on evaluation board or in socket
5. Based on characterization of 5 pieces over temperature.

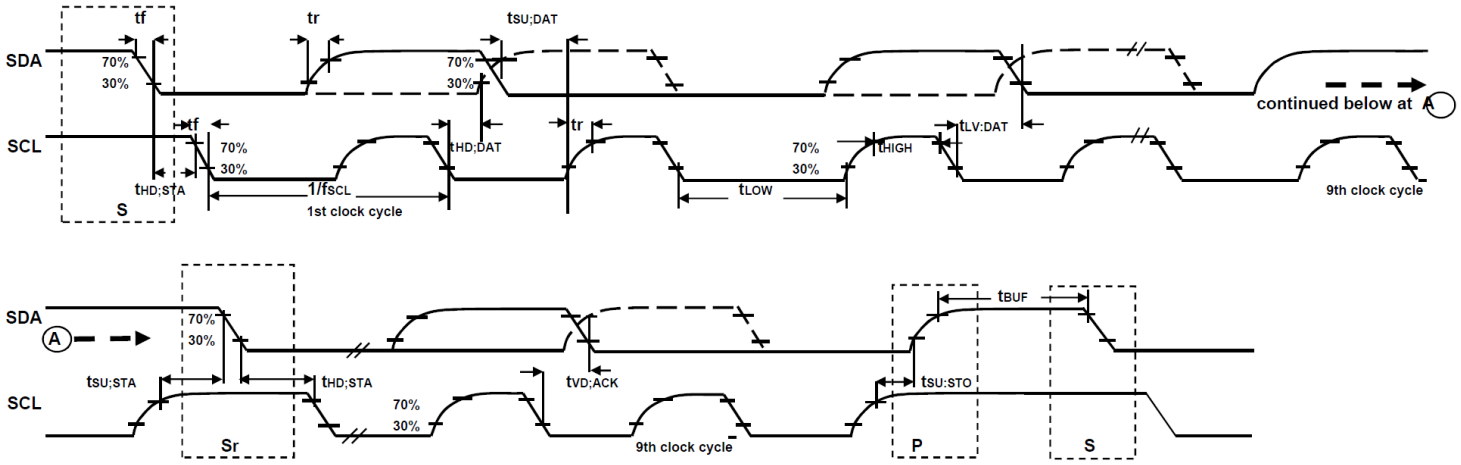
3.5 I²C Timing Characterization

Typical Operating Circuit of Section 4.2, VDD = 2.5V, VLOGIC = 1.8V±5%, 2.5V±5%, 3.0V±5%, or 3.3V±5%, T_A=25°C.

Parameters	Conditions	Min	Typical	Max	Units	Notes
I²C TIMING	**I²C FAST-MODE**					
f_{SCL}, SCL Clock Frequency		0		400	kHz	1
$t_{HD.STA}$, (Repeated) START Condition Hold Time		0.6			us	1
t_{LOW}, SCL Low Period		1.3			us	1
t_{HIGH}, SCL High Period		0.6			us	1
$t_{SU.STA}$, Repeated START Condition Setup Time		0.6			us	1
$t_{HD.DAT}$, SDA Data Hold Time		0			us	1
$t_{SU.DAT}$, SDA Data Setup Time		100			ns	1
t_r, SDA and SCL Rise Time	Cb bus cap. from 10 to 400pF	20+0.1Cb		300	ns	1
t_f, SDA and SCL Fall Time	Cb bus cap. from 10 to 400pF	20+0.1Cb		300	ns	1
$t_{SU.STO}$, STOP Condition Setup Time		0.6			us	1
t_{BUF}, Bus Free Time Between STOP and START Condition		1.3			us	1
C_b, Capacitive Load for each Bus Line				400	pF	2
$t_{VD.DAT}$, Data Valid Time				0.9	us	1
$t_{VD.ACK}$, Data Valid Acknowledge Time				0.9	us	1

Notes:

1. Based on characterization of 5 pieces over temperature on evaluation board or in socket
2. Guaranteed by design



I²C Bus Timing Diagram

3.6 Absolute Maximum Ratings

Stresses above those listed as "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these conditions is not implied. Exposure to the absolute maximum ratings conditions for extended periods may affect device reliability.

Absolute Maximum Ratings

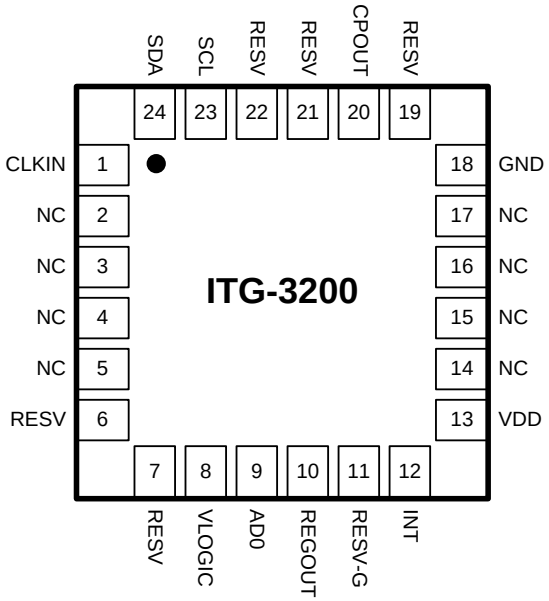
Parameter	Rating
Supply Voltage, VDD	-0.5V to +6V
VLOGIC Input Voltage Level	-0.5V to VDD + 0.5V
REGOUT	-0.5V to 2V
Input Voltage Level (CLKIN, AD0)	-0.5V to VDD + 0.5V
SCL, SDA, INT	-0.5V to VLOGIC + 0.5V
CPOUT (2.1V ≤ VDD ≤ 3.6V)	-0.5V to 30V
Acceleration (Any Axis, unpowered)	10,000*g* for 0.3ms
Operating Temperature Range	-40°C to +105°C
Storage Temperature Range	-40°C to +125°C
Electrostatic Discharge (ESD) Protection	1.5kV (HBM); 200V (MM)

4 Applications Information

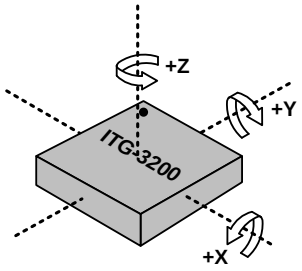
4.1 Pin Out and Signal Description

Number	Pin	Pin Description
1	CLKIN	Optional external reference clock input. Connect to GND if unused.
8	VLOGIC	Digital IO supply voltage. VLOGIC must be ≤ VDD at all times.
9	AD0	I^2C Slave Address LSB
10	REGOUT	Regulator filter capacitor connection
12	INT	Interrupt digital output (totem pole or open-drain)
13	VDD	Power supply voltage
18	GND	Power supply ground
11	RESV-G	Reserved - Connect to ground.
6, 7, 19, 21, 22	RESV	Reserved. Do not connect.
20	CPOUT	Charge pump capacitor connection
23	SCL	I^2C serial clock
24	SDA	I^2C serial data
2, 3, 4, 5, 14, 15, 16, 17	NC	Not internally connected. May be used for PCB trace routing.

Top View

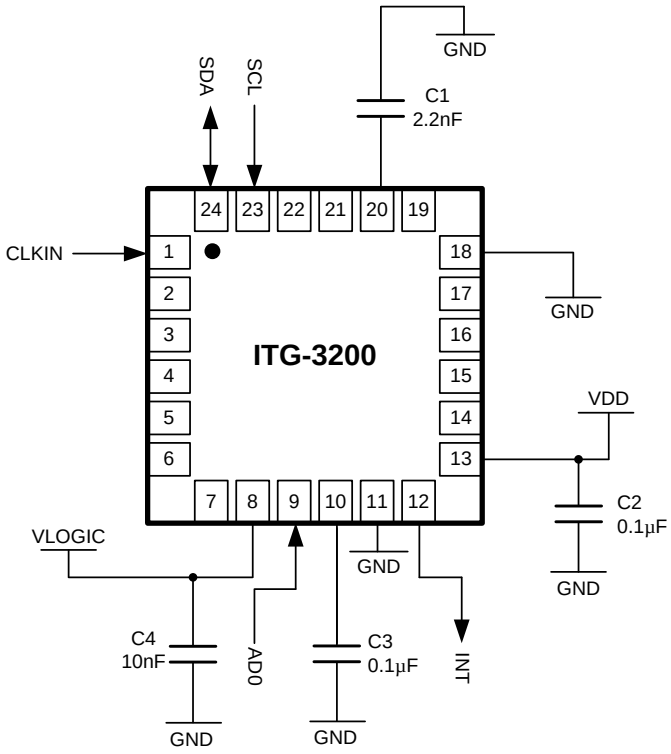


QFN Package
24-pin, 4mm x 4mm x 0.9mm



Orientation of Axes of Sensitivity and Polarity of Rotation

4.2 Typical Operating Circuit

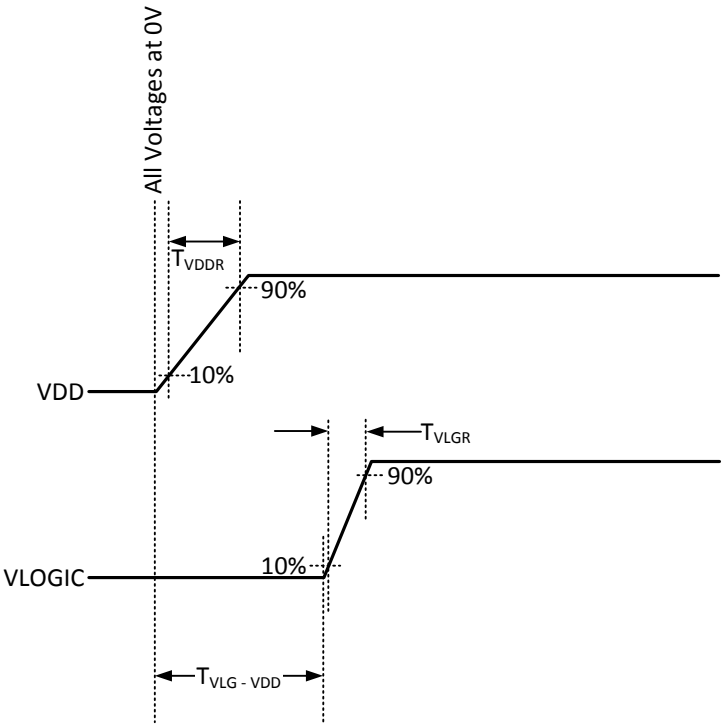


Typical Operating Circuit

4.3 Bill of Materials for External Components

Component	Label	Specification	Quantity
Charge Pump Capacitor	C1	Ceramic, X7R, 2.2nF ±10%, 50V	1
VDD Bypass Capacitor	C2	Ceramic, X7R, 0.1µF ±10%, 4V	1
Regulator Filter Capacitor	C3	Ceramic, X7R, 0.1µF ±10%, 2V	1
VLOGIC Bypass Capacitor	C4	Ceramic, X7R, 10nF ±10%, 4V	1

4.4 Recommended Power-On Procedure

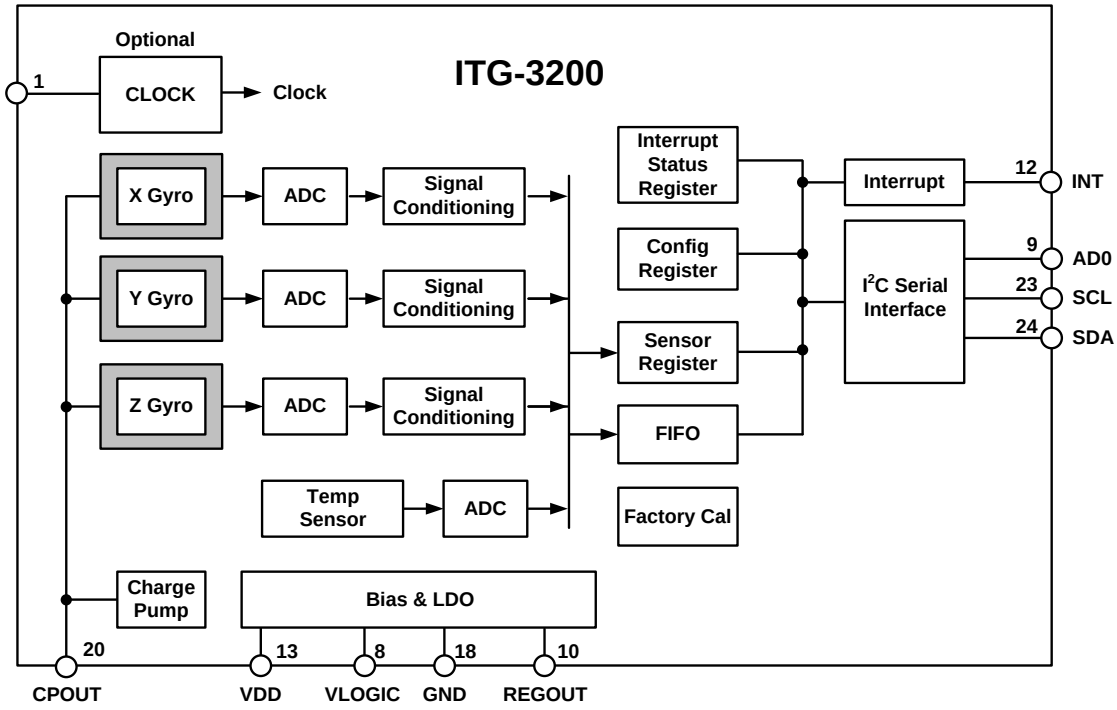


Power-Up Sequencing

1. T_{VDDR} is VDD rise time: Time for VDD to rise from 10% to 90% of its final value
2. T_{VDDR} is ≤5msec
3. T_{VLGR} is VLOGIC rise time: Time for VLOGIC to rise from 10% to 90% of its final value
4. T_{VLGR} is ≤1msec
5. $T_{VLG-VDD}$ is the delay from the start of VDD ramp to the start of VLOGIC rise
6. $T_{VLG-VDD}$ is 0 to 20msec but VLOGIC amplitude must always be ≤VDD amplitude
7. VDD and VLOGIC must be monotonic ramps

5 Functional Overview

5.1 Block Diagram



5.2 Overview

The ITG-3200 consists of the following key blocks and functions:

- Three-axis MEMS rate gyroscope sensors with individual 16-bit ADCs and signal conditioning
- I^2C serial communications interface
- Clocking
- Sensor Data Registers
- Interrupts
- Digital-Output Temperature Sensor
- Bias and LDO
- Charge Pump

5.3 Three-Axis MEMS Gyroscope with 16-bit ADCs and Signal Conditioning

The ITG-3200 consists of three independent vibratory MEMS gyroscopes, which detect rotational rate about the X (roll), Y (pitch), and Z (yaw) axes. When the gyros are rotated about any of the sense axes, the Coriolis Effect causes a deflection that is detected by a capacitive pickoff. The resulting signal is amplified, demodulated, and filtered to produce a voltage that is proportional to the angular rate. This voltage is digitized using individual on-chip 16-bit Analog-to-Digital Converters (ADCs) to sample each axis.

The full-scale range of the gyro sensors is preset to ±2000 degrees per second (°/s). The ADC output rate is programmable up to a maximum of 8,000 samples per second down to 3.9 samples per second, and user-selectable low-pass filters enable a wide range of cut-off frequencies.

5.4 I²C Serial Communications Interface

The ITG-3200 communicates to a system processor using the I²C serial interface, and the device always acts as a slave when communicating to the system processor. **The logic level for communications to the master is set by the voltage on the VLOGIC pin.** The LSB of the of the I²C slave address is set by pin 9 (AD0).

5.5 Clocking

The ITG-3200 has a flexible clocking scheme, allowing for a variety of internal or external clock sources for the internal synchronous circuitry. This synchronous circuitry includes the signal conditioning, ADCs, and various control circuits and registers. An on-chip PLL provides flexibility in the allowable inputs for generating this clock.

Allowable internal sources for generating the internal clock are:

- An internal relaxation oscillator (less accurate)
- Any of the X, Y, or Z gyros' MEMS oscillators (with an accuracy of ±2% over temperature)

Allowable external clocking sources are:

- 32.768kHz square wave
- 19.2MHz square wave

Which source to select for generating the internal synchronous clock depends on the availability of external sources and the requirements for clock accuracy. There are also start-up conditions to consider. When the ITG-3200 first starts up, the device operates off of its internal clock until programmed to operate from another source. This allows the user, for example, to wait for the MEMS oscillators to stabilize before they are selected as the clock source.

5.6 Sensor Data Registers

The sensor data registers contain the latest gyro and temperature data. They are read-only registers, and are accessed via the Serial Interface. Data from these registers may be read at any time, however, the interrupt function may be used to determine when new data is available.

5.7 Interrupts

Interrupt functionality is configured via the Interrupt Configuration register. Items that are configurable include the INT pin configuration, the interrupt latching and clearing method, and triggers for the interrupt. Items that can trigger an interrupt are (1) Clock generator locked to new reference oscillator (used when switching clock sources); and (2) new data is available to be read from the Data registers. The interrupt status can be read from the Interrupt Status register.

5.8 Digital-Output Temperature Sensor

An on-chip temperature sensor and ADC are used to measure the ITG-3200 die temperature. The readings from the ADC can be read from the Sensor Data registers.

5.9 Bias and LDO

The bias and LDO sections take in an unregulated VDD supply from 2.1V to 3.6V and generate the internal supply and the references voltages and currents required by the ITG-3200. The LDO output is bypassed by a capacitor at REGOUT. Additionally, the part has a VLOGIC reference voltage which sets the logic levels for its I²C interface.

5.10 Charge Pump

An on-board charge pump generates the high voltage (25V) required to drive the MEMS oscillators. Its output is bypassed by a capacitor at CPOUT.

6 Digital Interface

6.1 I²C Serial Interface

The internal registers and memory of the ITG-3200 can be accessed using I²C at up to 400kHz.

Serial Interface

Pin Number	Pin Name	Pin Description
8	VLOGIC	Digital IO supply voltage. VLOGIC must be ≤ VDD at all times.
9	AD0	I²C Slave Address LSB
23	SCL	I²C serial clock
24	SDA	I²C serial data

6.1.1 I²C Interface

I²C is a two wire interface comprised of the signals serial data (SDA) and serial clock (SCL). In general, the lines are open-drain and bi-directional. In a generalized I²C interface implementation, attached devices can be a master or a slave. The master device puts the slave address on the bus, and the slave device with the matching address acknowledges the master.

The ITG-3200 always operates as a slave device when communicating to the system processor, which thus acts as the master. SDA and SCL lines typically need pull-up resistors to VDD. The maximum bus speed is 400kHz.

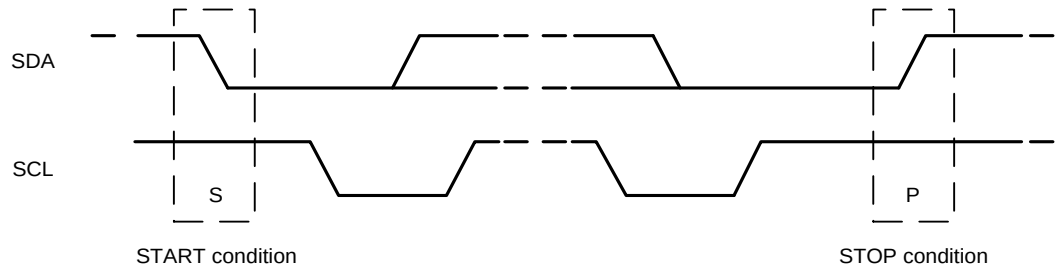
The slave address of the ITG-3200 devices is b110100X which is 7 bits long. The LSB bit of the 7 bit address is determined by the logic level on pin 9. This allows two ITG-3200 devices to be connected to the same I²C bus. When used in this configuration, the address of the one of the devices should be b1101000 (pin 9 is logic low) and the address of the other should be b1101001 (pin 9 is logic high). The I²C address is stored in register 0 (WHO_AM_I register).

I²C Communications Protocol

START (S) and STOP (P) Conditions

Communication on the I²C bus starts when the master puts the START condition (S) on the bus, which is defined as a HIGH-to-LOW transition of the SDA line while SCL line is HIGH (see figure below). The bus is considered to be busy until the master puts a STOP condition (P) on the bus, which is defined as a LOW to HIGH transition on the SDA line while SCL is HIGH (see figure below).

Additionally, the bus remains busy if a repeated START (Sr) is generated instead of a STOP condition.

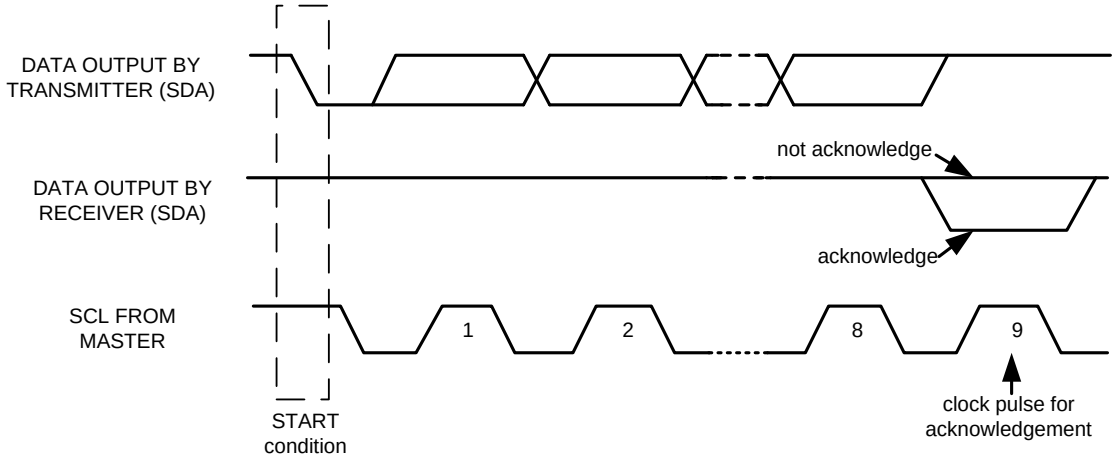


START and STOP Conditions

Data Format / Acknowledge

I²C data bytes are defined to be 8 bits long. There is no restriction to the number of bytes transmitted per data transfer. Each byte transferred must be followed by an acknowledge (ACK) signal. The clock for the acknowledge signal is generated by the master, while the receiver generates the actual acknowledge signal by pulling down SDA and holding it low during the HIGH portion of the acknowledge clock pulse.

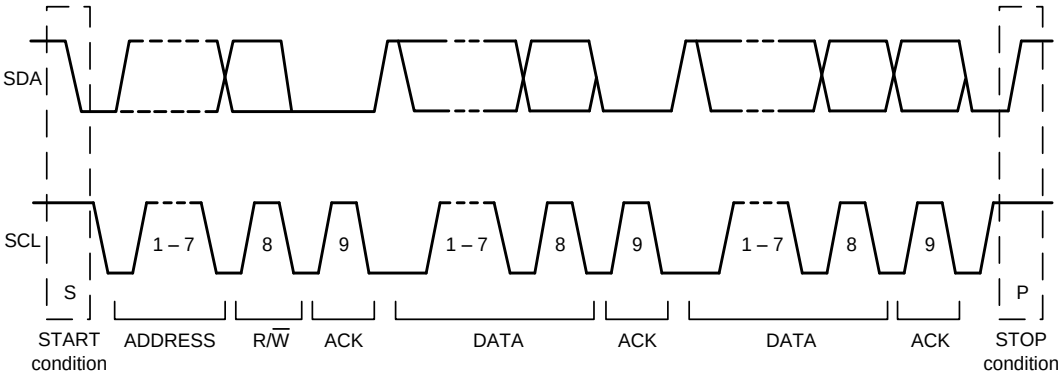
If a slave is busy and cannot transmit or receive another byte of data until some other task has been performed, it can hold SCL LOW, thus forcing the master into a wait state. Normal data transfer resumes when the slave is ready, and releases the clock line (see figure below).



Acknowledge on the I²C Bus

Communications

After beginning communications with the START condition (S), the master sends a 7-bit slave address followed by an 8th bit, the read/write bit. The read/write bit indicates whether the master is receiving data from or is writing to the slave device. Then, the master releases the SDA line and waits for the acknowledge signal (ACK) from the slave device. Each byte transferred must be followed by an acknowledge bit. To acknowledge, the slave device pulls the SDA line LOW and keeps it LOW for the high period of the SCL line. Data transmission is always terminated by the master with a STOP condition (P), thus freeing the communications line. However, the master can generate a repeated START condition (Sr), and address another slave without first generating a STOP condition (P). A LOW to HIGH transition on the SDA line while SCL is HIGH defines the stop condition. All SDA changes should take place when SCL is low, with the exception of start and stop conditions.



Complete I²C Data Transfer

To write the internal ITG-3200 device registers, the master transmits the start condition (S), followed by the I^2C address and the write bit (0). At the 9th clock cycle (when the clock is high), the ITG-3200 device acknowledges the transfer. Then the master puts the register address (RA) on the bus. After the ITG-3200 acknowledges the reception of the register address, the master puts the register data onto the bus. This is followed by the ACK signal, and data transfer may be concluded by the stop condition (P). To write multiple bytes after the last ACK signal, the master can continue outputting data rather than transmitting a stop signal. In this case, the ITG-3200 device automatically increments the register address and loads the data to the appropriate register. The following figures show single and two-byte write sequences.

Single-Byte Write Sequence

Master	S	AD+W		RA		DATA		P
Slave			ACK		ACK		ACK	

Burst Write Sequence

Master	S	AD+W		RA		DATA		DATA		P
Slave			ACK		ACK		ACK		ACK	

To read the internal ITG-3200 device registers, the master first transmits the start condition (S), followed by the I^2C address and the write bit (0). At the 9th clock cycle (when clock is high), the ITG acknowledges the transfer. The master then writes the register address that is going to be read. Upon receiving the ACK signal from the ITG-3200, the master transmits a start signal followed by the slave address and read bit. As a result, the ITG-3200 sends an ACK signal and the data. The communication ends with a not acknowledge (NACK) signal and a stop bit from master. The NACK condition is defined such that the SDA line remains high at the 9th clock cycle. To read multiple bytes of data, the master can output an acknowledge signal (ACK) instead of a not acknowledge (NACK) signal. In this case, the ITG-3200 automatically increments the register address and outputs data from the appropriate register. The following figures show single and two-byte read sequences.

Single-Byte Read Sequence

Master	S	AD+W		RA		S	AD+R			NACK	P
Slave			ACK		ACK			ACK	DATA		

Burst Read Sequence

Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

I²C Terms

Signal	Description
S	Start Condition: SDA goes from high to low while SCL is high
AD	Slave I²C address
W	Write bit (0)
R	Read bit (1)
ACK	Acknowledge: SDA line is low while the SCL line is high at the 9th clock cycle
NACK	Not-Acknowledge: SDA line stays high at the 9th clock cycle
RA	ITG-3200 internal register address
DATA	Transmit or received data
P	Stop condition: SDA going from low to high while SCL is high

7 Register Map

Addr Hex	Addr Decimal	Register Name	R/W	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	0	WHO_AM_I	R/W	-	ID						-
15	21	SMPLRT_DIV	R/W	SMPLRT_DIV							
16	22	DLPF_FS	R/W	-	-	-	FS_SEL		DLPF_CFG		
17	23	INT_CFG	R/W	ACTL	OPEN	LATCH_ INT_EN	INT_ ANYRD_ 2CLEAR	-	ITG_RDY _EN	-	RAW_ RDY_ EN
1A	26	INT_STATUS	R	-	-	-	-	-	ITG_RDY	-	RAW_ DATA_ RDY
1B	27	TEMP_OUT_H	R	TEMP_OUT_H							
1C	28	TEMP_OUT_L	R	TEMP_OUT_L							
1D	29	GYRO_XOUT_H	R	GYRO_XOUT_H							
1E	30	GYRO_XOUT_L	R	GYRO_XOUT_L							
1F	31	GYRO_YOUT_H	R	GYRO_YOUT_H							
20	32	GYRO_YOUT_L	R	GYRO_YOUT_L							
21	33	GYRO_ZOUT_H	R	GYRO_ZOUT_H							
22	34	GYRO_ZOUT_L	R	GYRO_ZOUT_L							
3E	62	PWR_MGM	R/W	H_RESET	SLEEP	STBY_XG	STBY_YG	STBY_ZG	CLK_SEL		

8 Register Description

This section details each register within the InvenSense ITG-3200 gyroscope. Note that any bit that is not defined should be set to zero in order to be compatible with future InvenSense devices.

The register space allows single-byte reads and writes, as well as burst reads and writes. When performing burst reads or writes, the memory pointer will increment until either (1) reading or writing is terminated by the master, or (2) the memory pointer reaches certain reserved registers between registers 33 and 60.

8.1 Register 0 – Who Am I

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	0	-	ID						-

Description:
This register is used to verify the identity of the device.
Parameters:
ID Contains the I^2C address of the device, which can also be changed by writing to this register.

The Power-On-Reset value of Bit6: Bit1 is 110 100.

8.2 Register 21 – Sample Rate Divider

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Default Value
15	21	SMPLRT_DIV								00h

Description:
This register determines the sample rate of the ITG-3200 gyros. The gyros outputs are sampled internally at either 1kHz or 8kHz, determined by the *DLPF_CFG* setting (see register 22). This sampling is then filtered digitally and delivered into the sensor registers after the number of cycles determined by this register. The sample rate is given by the following formula:

$F_{sample} = F_{internal} / (divider+1)$, where $F_{internal}$ is either 1kHz or 8kHz

As an example, if the internal sampling is at 1kHz, then setting this register to 7 would give the following:

$F_{sample} = 1kHz / (7 + 1) = 125Hz$, or 8ms per sample

Parameters:
SMPLRT_DIV Sample rate divider: 0 to 255

8.3 Register 22 – DLPF, Full Scale

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Default Value
16	22	-			FS_SEL		DLPF_CFG			00h

Description:

This register configures several parameters related to the sensor acquisition.

The *FS_SEL* parameter allows setting the full-scale range of the gyro sensors, as described in the table below. The power-on-reset value of *FS_SEL* is 00h. **Set to 03h for proper operation.**

FS_SEL

FS_SEL	Gyro Full-Scale Range
0	Reserved
1	Reserved
2	Reserved
3	±2000°/sec

The *DLPF_CFG* parameter sets the digital low pass filter configuration. It also determines the internal sampling rate used by the device as shown in the table below.

DLPF_CFG

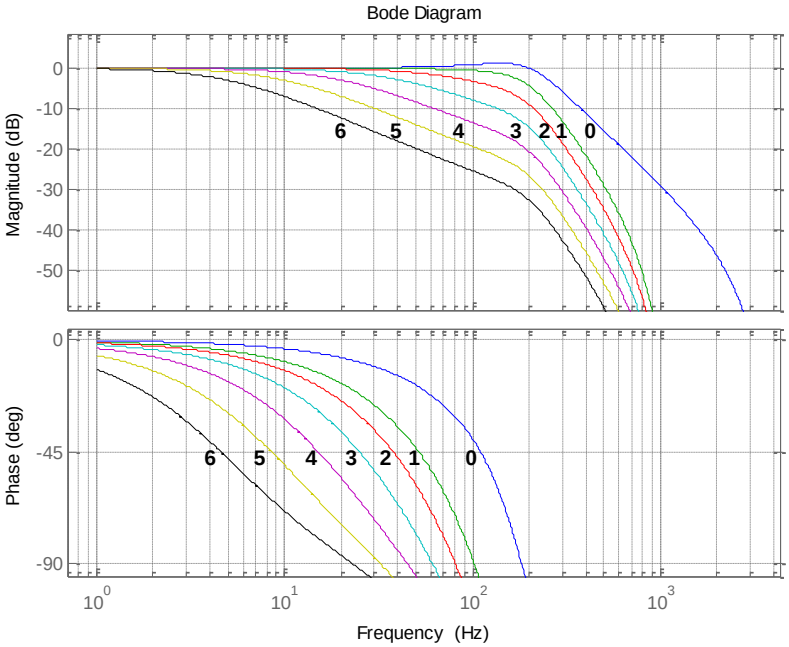
DLPF_CFG	Low Pass Filter Bandwidth	Internal Sample Rate
0	256Hz	8kHz
1	188Hz	1kHz
2	98Hz	1kHz
3	42Hz	1kHz
4	20Hz	1kHz
5	10Hz	1kHz
6	5Hz	1kHz
7	Reserved	Reserved

Parameters:

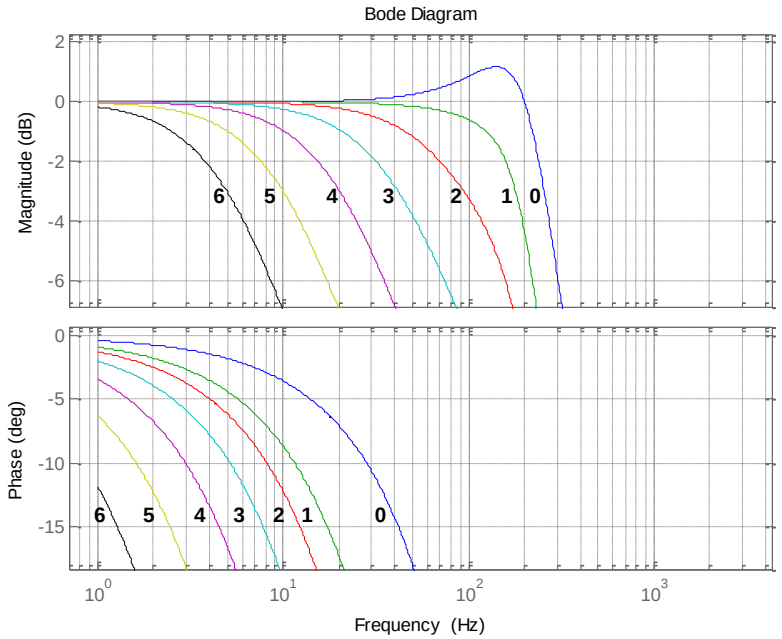
FS_SEL Full scale selection for gyro sensor data

DLPF_CFG Digital low pass filter configuration and internal sampling rate configuration

DLPF Characteristics: The gain and phase responses of the digital low pass filter settings (*DLPF_CFG*) are shown below:



Gain and Phase vs. Digital Filter Setting



Gain and Phase vs. Digital Filter Setting, Showing Passband Details

8.4 Register 23 – Interrupt Configuration

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Default Value
17	23	ACTL	OPEN	LATCH_ INT_EN	INT_ ANYRD_ 2CLEAR	0	ITG_RDY_ EN	0	RAW_ RDY_ EN	00h

Description:

This register configures the interrupt operation of the device. The interrupt output pin (INT) configuration can be set, the interrupt latching/clearing method can be set, and the triggers for the interrupt can be set.

Note that if the application requires reading every sample of data from the ITG-3200 part, it is best to enable the raw data ready interrupt (*RAW_RDY_EN*). This allows the application to know when new sample data is available.

Parameters:

ACTL	Logic level for INT output pin – 1=active low, 0=active high
OPEN	Drive type for INT output pin – 1=open drain, 0=push-pull
LATCH_INT_EN	Latch mode – 1=latch until interrupt is cleared, 0=50us pulse
INT_ANYRD_2CLEAR	Latch clear method – 1=any register read, 0=status register read only
ITG_RDY_EN	Enable interrupt when device is ready (PLL ready after changing clock source)
RAW_RDY_EN	Enable interrupt when data is available
0	Load zeros into Bits 1 and 3 of the Interrupt Configuration register.

8.5 Register 26 – Interrupt Status

Type: Read only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Default Value
1A	26	-	-	-	-	-	ITG_RDY	-	RAW_ DATA_ RDY	00h

Description:

This register is used to determine the status of the ITG-3200 interrupts. Whenever one of the interrupt sources is triggered, the corresponding bit will be set. The polarity of the interrupt pin (active high/low) and the latch type (pulse or latch) has no affect on these status bits.

Use the Interrupt Configuration register (23) to enable the interrupt triggers. If the interrupt is not enabled, the associated status bit will not get set.

In normal use, the *RAW_DATA_RDY* interrupt is used to determine when new sensor data is available in either the sensor registers (27 to 32).

Interrupt Status bits get cleared as determined by INT_ANYRD_2CLEAR in the interrupt configuration register (23).

Parameters:

ITG_RDY	PLL ready
RAW_DATA_RDY	Raw data is ready

8.6 Registers 27 to 34 – Sensor Registers

Type: Read only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1B	27	TEMP_OUT_H							
1C	28	TEMP_OUT_L							
1D	29	GYRO_XOUT_H							
1E	30	GYRO_XOUT_L							
1F	31	GYRO_YOUT_H							
20	32	GYRO_YOUT_L							
21	33	GYRO_ZOUT_H							
22	34	GYRO_ZOUT_L							

Description:
These registers contain the gyro and temperature sensor data for the ITG-3200 parts. At any time, these values can be read from the device; however it is best to use the interrupt function to determine when new data is available.

Parameters:
TEMP_OUT_H/L 16-bit temperature data (2's complement format)
GYRO_XOUT_H/L 16-bit X gyro output data (2's complement format)
GYRO_YOUT_H/L 16-bit Y gyro output data (2's complement format)
GYRO_ZOUT_H/L 16-bit Y gyro output data (2's complement format)

8.7 Register 62 – Power Management

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Default Value
3E	62	H_RESET	SLEEP	STBY _XG	STBY _YG	STBY _ZG	CLK_SEL			00h

Description:
This register is used to manage the power control, select the clock source, and to issue a master reset to the device.

Setting the *SLEEP* bit in the register puts the device into very low power sleep mode. In this mode, only the serial interface and internal registers remain active, allowing for a very low standby current. Clearing this bit puts the device back into normal mode. To save power, the individual standby selections for each of the gyros should be used if any gyro axis is not used by the application.

The *CLK_SEL* setting determines the device clock source as follows:

CLK_SEL

CLK_SEL	Clock Source
0	Internal oscillator
1	PLL with X Gyro reference
2	PLL with Y Gyro reference
3	PLL with Z Gyro reference
4	PLL with external 32.768kHz reference
5	PLL with external 19.2MHz reference
6	Reserved
7	Reserved

On power up, the ITG-3200 defaults to the internal oscillator. It is highly recommended that the device is configured to use one of the gyros (or an external clock) as the clock reference, due to the improved stability.

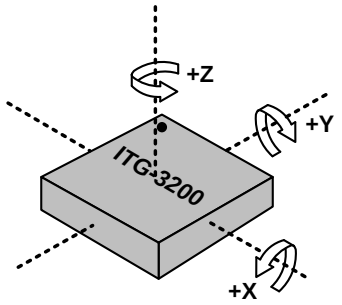
Parameters:

H_RESET	Reset device and internal registers to the power-up-default settings
SLEEP	Enable low power sleep mode
STBY_XG	Put gyro X in standby mode (1=standby, 0=normal)
STBY_YG	Put gyro Y in standby mode (1=standby, 0=normal)
STBY_ZG	Put gyro Z in standby mode (1=standby, 0=normal)
CLK_SEL	Select device clock source

9 Assembly

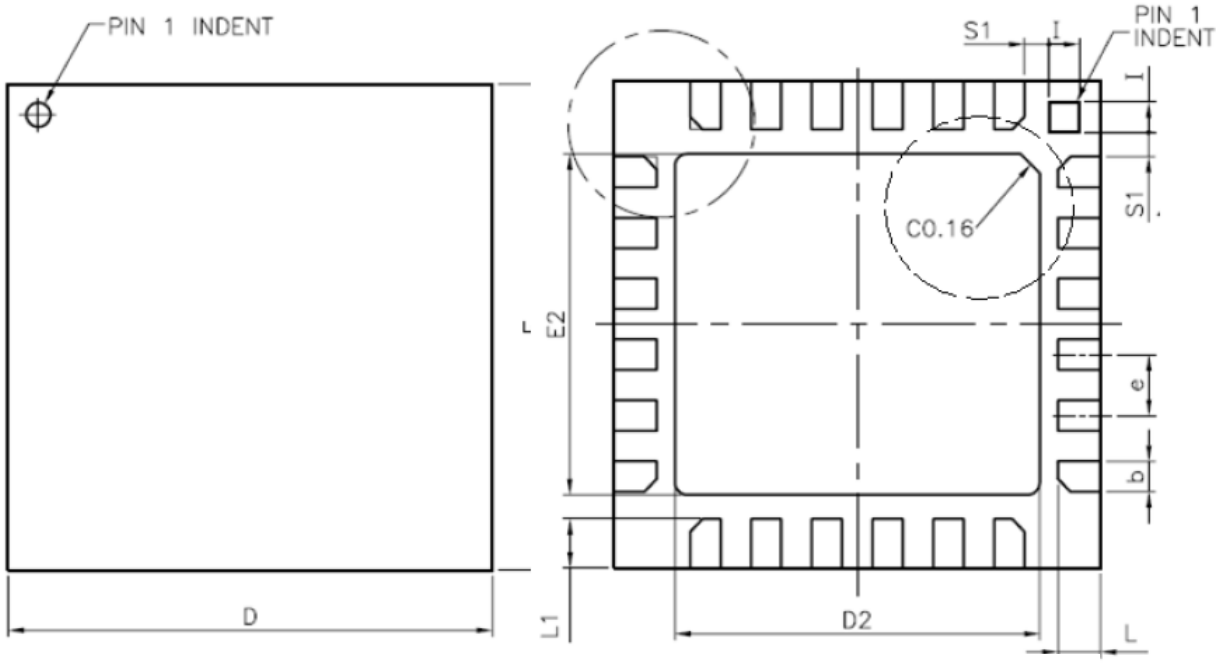
9.1 Orientation

The diagram below shows the orientation of the axes of sensitivity and the polarity of rotation.



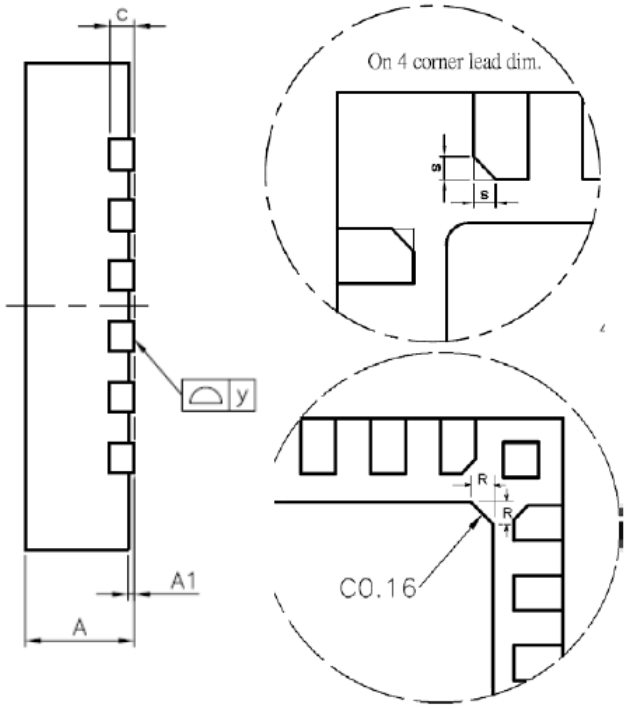
Orientation of Axes of Sensitivity and Polarity of Rotation

9.2 Package Dimensions



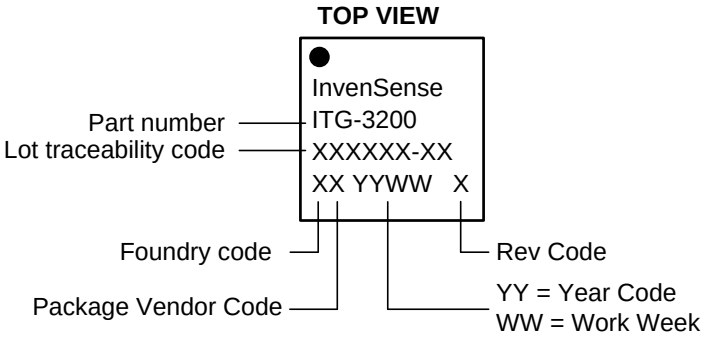
Top View **Bottom View**

SYMBOLS	DIMENSIONS IN MILLIMETERS		
	MIN	NOM	MAX
A	0.85	0.90	0.95
A1	0.00	0.02	0.05
b	0.18	0.25	0.30
c	---	0.20 REF.	---
D	3.90	4.00	4.10
D2	2.95	3.00	3.05
E	3.90	4.00	4.10
E2	2.75	2.80	2.85
e	---	0.50	---
L	0.30	0.35	0.40
L1	0.35	0.40	0.45
I	0.20	0.25	0.30
R	0.085	---	0.145
s	0.05	---	0.15
S1	0.15	0.20	0.25
y	0.00	---	0.075



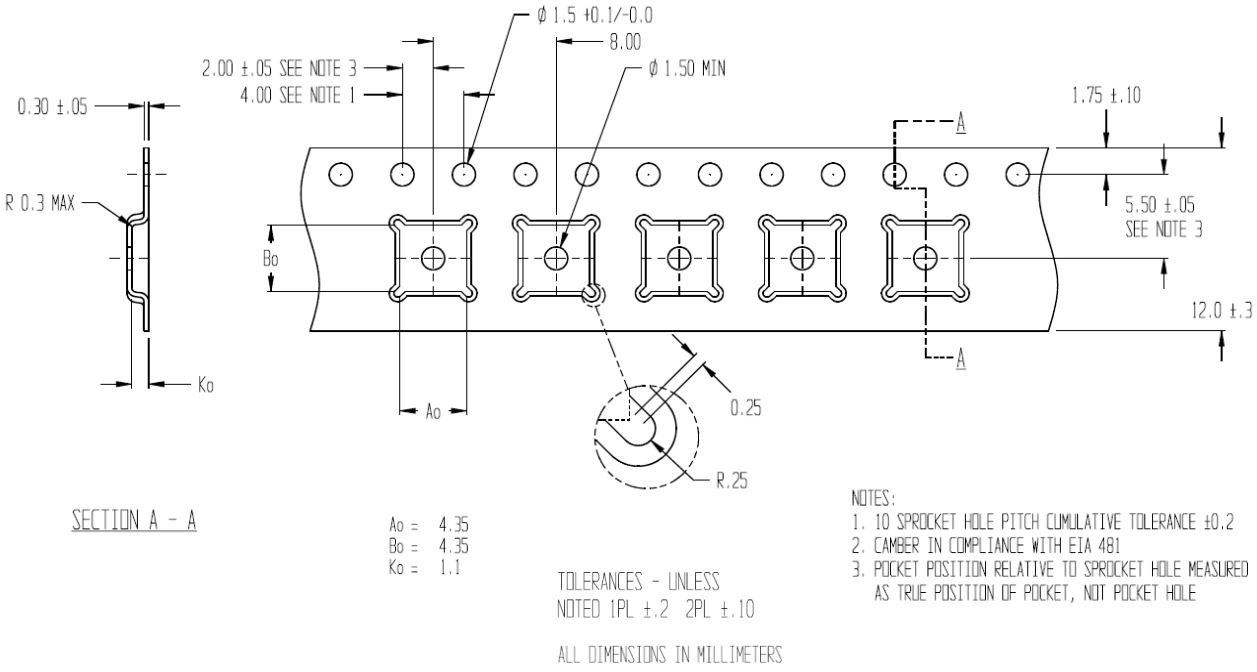
Package Dimensions

9.3 Package Marking Specification

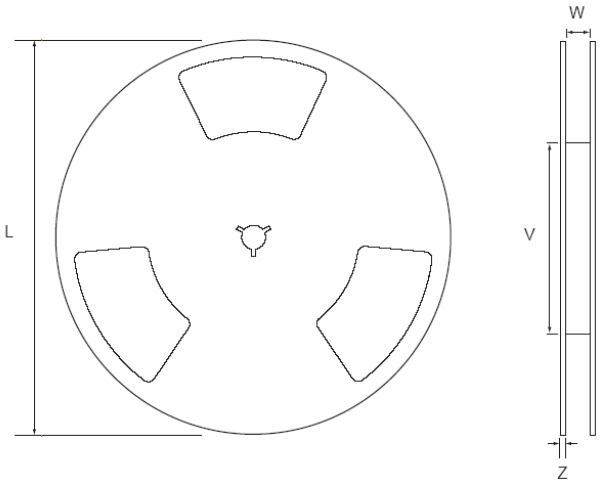


Package Marking Specification

9.4 Tape & Reel Specification



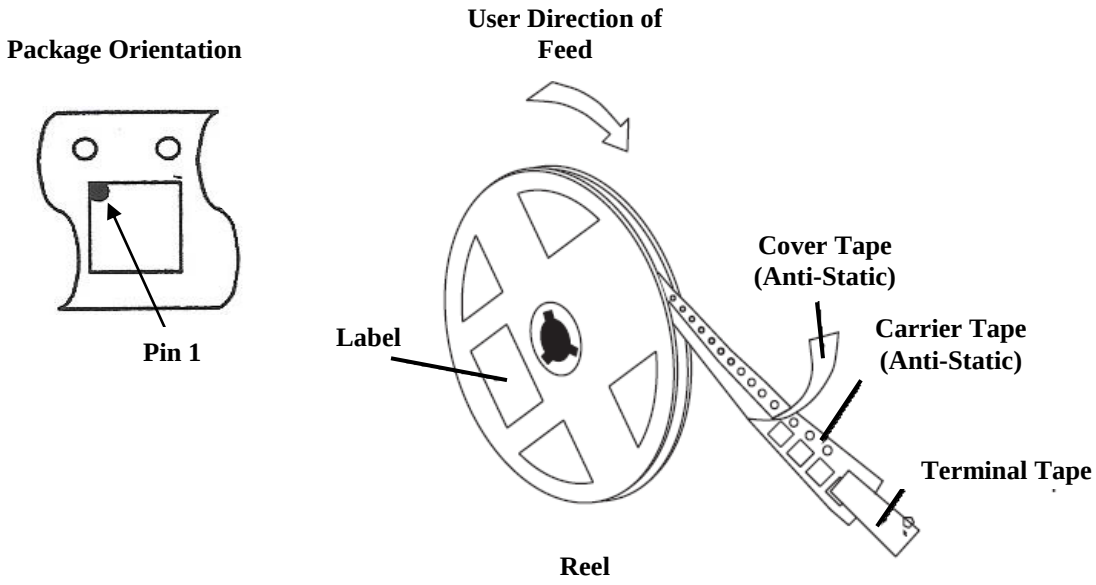
Tape Dimensions



Reel Outline Drawing

Reel Dimensions and Package Size

PKG SIZE	REEL (mm)			
	L	V	W	Z
4x4	330	100	13.2	2.2

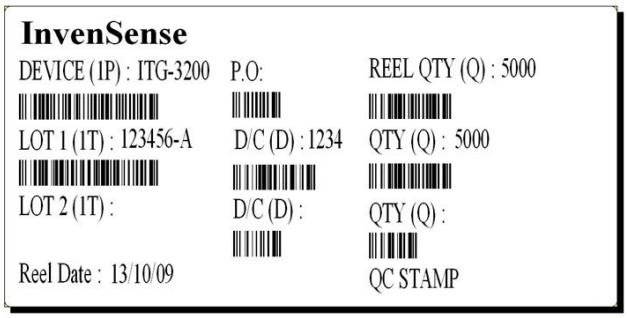


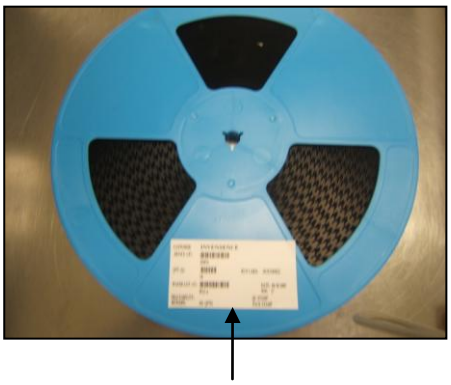
Tape and Reel Specification

Reel Specifications

Quantity Per Reel	5,000
Reels per Box	1
Boxes Per Carton (max)	3 full pizza boxes packed in the center of the carton, buffered by two empty pizza boxes (front and back).
Pcs/Carton (max)	15,000

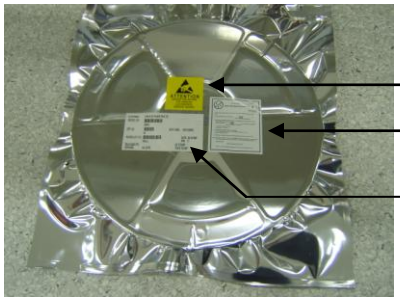
9.5 Label



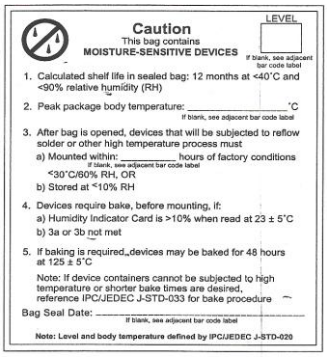


Location of Label

9.6 Packaging



Moisture Barrier Bag With Labels



Moisture-Sensitive Caution Label

Reel in Box

Box with Tape & Reel Label

9.7 Soldering Exposed Die Pad

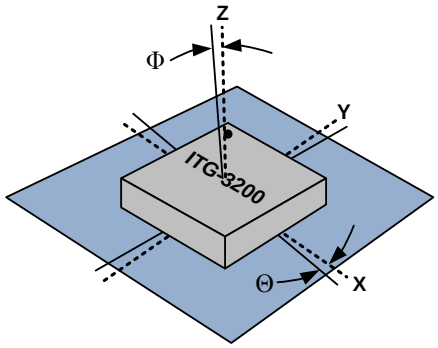
The ITG-3200 has very low active and standby current consumption. The exposed die pad is not required for heat sinking, and should not be soldered to the PCB since soldering to it contributes to performance changes due to package thermo-mechanical stress.

9.8 Component Placement

Testing indicates that there are no specific design considerations other than generally accepted industry design practices for component placement near the ITG-3200 multi-axis gyroscope to prevent noise coupling, and thermo-mechanical stress.

9.9 PCB Mounting and Cross-Axis Sensitivity

Orientation errors of the gyroscope mounted to the printed circuit board can cause cross-axis sensitivity in which one gyro responds to rotation about another axis, for example, the X-axis gyroscope responding to rotation about the Y or Z axes. The orientation mounting errors are illustrated in the figure below.



Package Gyro Axes (·····) Relative to PCB Axes (——) with Orientation Errors (Θ and Φ)

The table below shows the cross-axis sensitivity as a percentage of the specified gyroscope's sensitivity for a given orientation error.

Cross-Axis Sensitivity vs. Orientation Error

Orientation Error (θ or Φ)	Cross-Axis Sensitivity (sinθ or sinΦ)
0º	0%
0.5º	0.87%
1º	1.75%

The specification for cross-axis sensitivity in Section 3 includes the effect of the die orientation error with respect to the package.

9.10 MEMS Handling Instructions

MEMS (Micro Electro-Mechanical Systems) are a time-proven, robust technology used in hundreds of millions of consumer, automotive and industrial products. MEMS devices consist of microscopic moving mechanical structures. They differ from conventional IC products even though they can be found in similar packages. Therefore, MEMS devices require different handling precautions than conventional ICs prior to mounting onto printed circuit boards (PCBs).

The ITG-3200 gyroscope has a shock tolerance of 10,000*g*. InvenSense packages its gyroscopes as it deems proper for protection against normal handling and shipping. It recommends the following handling precautions to prevent potential damage.

- Individually packaged or trays of gyroscopes should not be dropped onto hard surfaces. Components placed in trays could be subject to *g*-forces in excess of 10,000*g* if dropped.
- Printed circuit boards that incorporate mounted gyroscopes should not be separated by manually snapping apart. This could also create *g*-forces in excess of 10,000*g*.

9.11 Gyroscope Surface Mount Guidelines

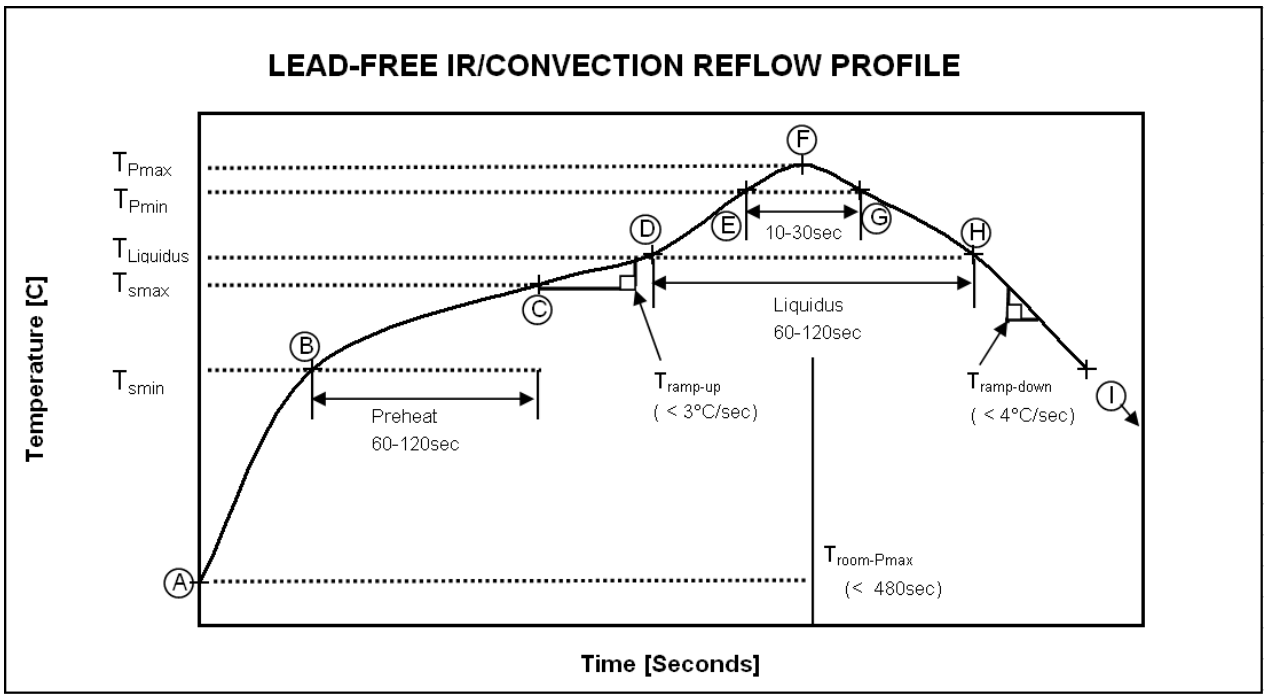
Any material used in the surface mount assembly process of the MEMS gyroscope should be free of restricted RoHS elements or compounds. Pb-free solders should be used for assembly.

In order to assure gyroscope performance, several industry standard guidelines need to be considered for surface mounting. These guidelines are for both printed circuit board (PCB) design and surface mount assembly and are available from packaging and assembly houses.

When using MEMS gyroscope components in plastic packages, package stress due to PCB mounting and assembly could affect the output offset and its value over a wide range of temperatures. This is caused by the mismatch between the Coefficient Temperature Expansion (CTE) of the package material and the PCB. Care must be taken to avoid package stress due to mounting.

9.12 Reflow Specification

The approved solder reflow curve shown in the figure below conforms to IPC/JEDEC J-STD-020D.01 (Moisture/Reflow Sensitivity Classification for Nonhermetic Solid State Surface Mount Devices) with a maximum peak temperature (Tc = 260°C). This is specified for component-supplier reliability qualification testing using lead-free solder for package thicknesses less than 1.6 mm. The reliability qualification pre-conditioning used by InvenSense incorporates three of these conforming reflow cycles. All temperatures refer to the topside of the QFN package, as measured on the package body surface. Customer solder-reflow processes should use the solder manufacturer's recommendations, making sure to never exceed the constraints listed in the table and figure below, as these represent the maximum tolerable ratings for the device. For optimum results, production solder reflow processes should use lower temperatures, reduced exposure times to high temperatures, and lower ramp-up and ramp-down rates than those listed below.



Approved IR/Convection Solder Reflow Curve

Temperature Set Points for IR / Convection Reflow Corresponding to Figure Above

Step	Setting	CONSTRAINTS		
		Temp (°C)	Time (sec)	Rate (°C/sec)
A	T_{room}	25		
B	T_{Smin}	150		
C	T_{Smax}	200	$60 < t_{BC} < 120$	
D	$T_{Liquidus}$	217		$r_{(TLiquidus-TPmax)} < 3$
E	T_{Pmin} [< TPmax-5°C, 250°C]	255		$r_{(TLiquidus-TPmax)} < 3$
F	T_{Pmax} [< TPmax, 260°C]	260	$t_{AF} < 480$	$r_{(TLiquidus-TPmax)} < 3$
G	T_{Pmin} [< TPmax-5°C, 250°C]	255	$t_{EG} < 30$	$r_{(TPmax-TLiquidus)} < 4$
H	$T_{Liquidus}$	217	$60 < t_{DH} < 120$	
I	T_{room}	25		

9.13 Storage Specifications

The storage specification of the ITG-3200 gyroscope conforms to IPC/JEDEC J-STD-020C Moisture Sensitivity Level (MSL) 3.

Storage Specifications for ITG-3200

Calculated shelf-life in moisture-sealed bag	12 months -- Storage conditions: <40°C and <90% RH
After opening moisture-sealed bag	168 hours -- Storage conditions: ambient ≤30°C at 60% RH

10 Reliability

10.1 Qualification Test Policy

InvenSense's products complete a Qualification Test Plan before being released to production. The Qualification Test Plan follows the JEDEC 47D Standards, "Stress-Test-Driven Qualification of Integrated Circuits," with the individual tests described below.

10.2 Qualification Test Plan

Accelerated Life Tests

TEST	Method/Condition	Lot Quantity	Sample / Lot	Acc / Reject Criteria
High Temperature Operating Life (HTOL/LFR)	JEDEC JESD22-A108C, Dynamic, 3.63V biased, Tj>125°C [read-points 168, 500, 1000 hours]	3	77	(0/1)
Steady-State Temperature Humidity Bias Life (1)	JEDEC JESD22-A101C, 85°C/85%RH [read-points 168, 500 hours], Information Only 1000 hours]	3	77	(0/1)
High Temperature Storage Life	JEDEC JESD22-A103C, Cond. A, 125°C Non-Bias Bake [read-points 168, 500, 1000 hours]	3	77	(0/1)

Device Component Level Tests

TEST	Method/Condition	Lot Quantity	Sample / Lot	Acc / Reject Criteria
ESD-HBM	JEDEC JESD22-A114F, Class 2 (1.5KV)	1	3	(0/1)
ESD-MM	JEDEC JESD22-A115-A, Class B (200V)	1	3	(0/1)
Latch Up	JEDEC JESD78B Level 2, 125C, +/- 100mA	1	6	(0/1)
Mechanical Shock	JEDEC JESD22-B104C, Mil-Std-883, method 2002, Cond. D, 10,000*g's*, 0.3ms, ±X,Y,Z – 6 directions, 5 times/direction	3	5	(0/1)
Vibration	JEDEC JESD22-B103B, Variable Frequency (random), Cond. B, 5-500Hz, X,Y,Z – 4 times/direction	3	5	(0/1)
Temperature Cycling (1)	JEDEC JESD22-A104D Condition N, -40°C to +85°C, Soak Mode 2, 100 cycles	3	77	(0/1)

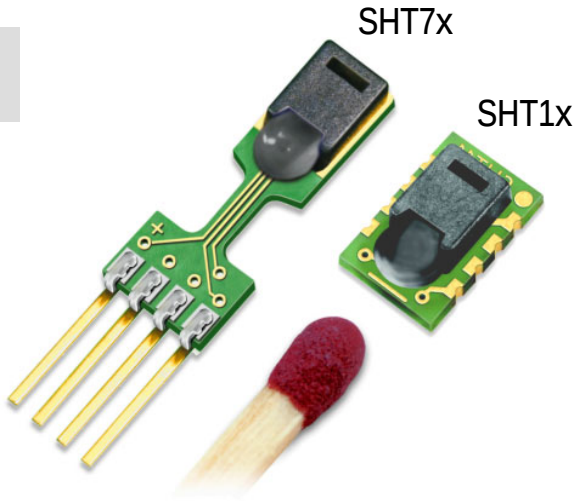
Board Level Tests

TEST	Method/Condition/	Lot Quantity	Sample / Lot	Acc / Reject Criteria
Board Mechanical Shock	JEDEC JESD22-B104C,Mil-Std-883, method 2002, Cond. D, 10,000*g's*, 0.3ms, +-X,Y,Z – 6 directions, 5 times/direction	1	5	(0/1)
Board T/C	JEDEC JESD22-A104D Condition N, -40°C to +85°C, Soak Mode 2, 100 cycles	1	40	(0/1)

(1) – Tests are preceded by MSL3 Preconditioning in accordance with JEDEC JESD22-A113F

SHT1x / SHT7x
Humidity & Temperature Sensor

Evaluation Kit Available



- Relative humidity and temperature sensors
- Dew point
- Fully calibrated, digital output
- Excellent long-term stability
- No external components required
- Ultra low power consumption
- Surface mountable or 4-pin fully interchangeable
- Small size
- Automatic power down

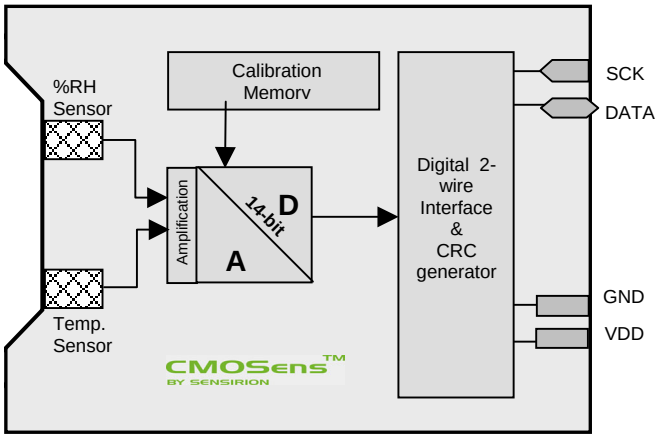
SHT1x / SHT7x Product Summary

The SHTxx is a single chip relative humidity and temperature multi sensor module comprising a calibrated digital output. Application of industrial CMOS processes with patented micro-machining (CMOSens® technology) ensures highest reliability and excellent long term stability. The device includes a capacitive polymer sensing element for relative humidity and a bandgap temperature sensor. Both are seamlessly coupled to a 14bit analog to digital converter and a serial interface circuit on the same chip. This results in superior signal quality, a fast response time and insensitivity to external disturbances (EMC) at a very competitive price.
Each SHTxx is individually calibrated in a precision humidity chamber with a chilled mirror hygrometer as reference. The calibration coefficients are programmed into the OTP memory. These coefficients are used internally during measurements to calibrate the signals from the sensors.
The 2-wire serial interface and internal voltage regulation allows easy and fast system integration. Its tiny size and low power consumption makes it the ultimate choice for even the most demanding applications.
The device is supplied in either a surface-mountable LCC (Leadless Chip Carrier) or as a pluggable 4-pin single-in-line type package. Customer specific packaging options may be available on request.

Applications

_ HVAC
_ Automotive
_ Consumer Goods
_ Weather Stations
_ Humidifiers
_ Dehumidifiers
_ Test & Measurement
_ Data Logging
_ Automation
_ White Goods
_ Medical

Block Diagram



Ordering Information

Part Number	Humidity accuracy [%RH]	Temperature accuracy [K] @ 25 °C	Package
SHT11	±3.0	±0.4	SMD (LCC)
SHT15	±2.0	±0.3	SMD (LCC)
SHT71	±3.0	±0.4	4-pin single-in-line
SHT75	±1.8	±0.3	4-pin single-in-line

1 Sensor Performance Specifications

Parameter	Conditions	Min.	Typ.	Max.	Units
Humidity					
Resolution (2)		0.5	0.03	0.03	%RH
		8	12	12	bit
Repeatability			±0.1		%RH
Accuracy (1) Uncertainty	linearized	see figure 1			
Interchangeability		Fully interchangeable			
Nonlinearity	raw data		±3		%RH
	linearized		<<1		%RH
Range		0		100	%RH
Response time	1/e (63%) slowly moving air		4		s
Hysteresis			±1		%RH
Long term stability	typical		< 0.5		%RH/yr
Temperature					
Resolution (2)		0.04	0.01	0.01	°C
		0.07	0.02	0.02	°F
		12	14	14	bit
Repeatability			±0.1		°C
			±0.2		°F
Accuracy		see figure 1			
Range		-40		123.8	°C
		-40		254.9	°F
Response Time	1/e (63%)	5		30	s

Table 1 Sensor Performance Specifications

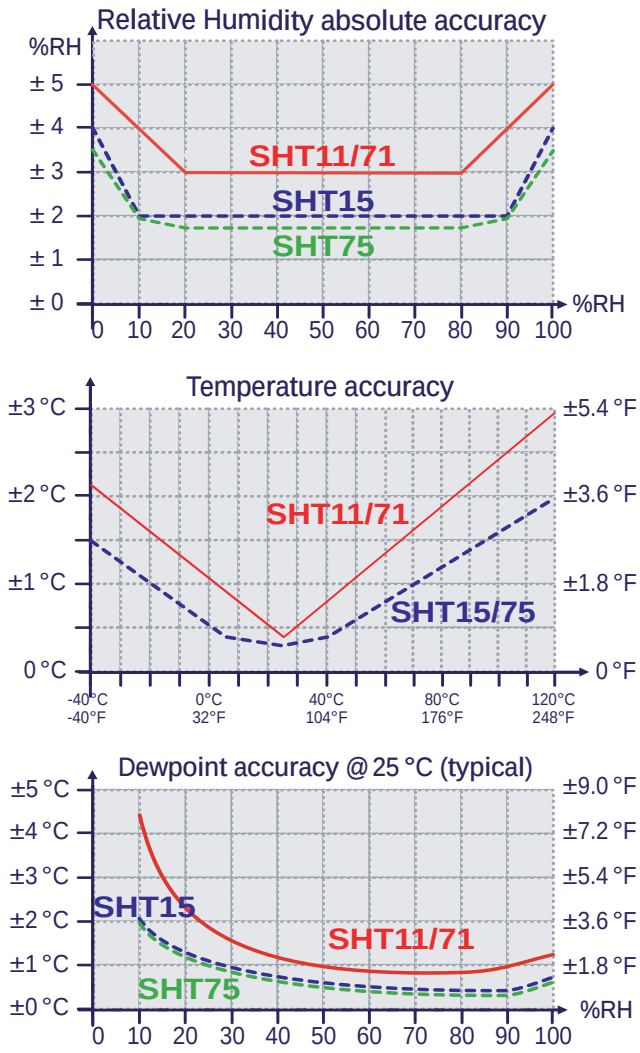


Figure 1 Rel. Humidity, Temperature and Dewpoint accuracies

2 Interface Specifications

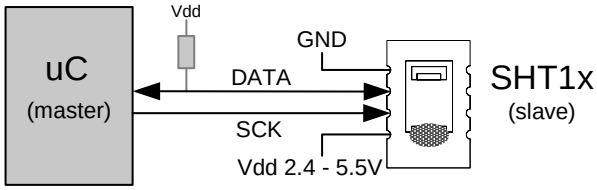


Figure 2 Typical application circuit

2.1 Power Pins

The SHTxx requires a voltage supply between 2.4 and 5.5 V. After powerup the device needs 11ms to reach its "sleep" state. No commands should be sent before that time.
Power supply pins (VDD, GND) may be decoupled with a 100 nF capacitor.

2.2 Serial Interface (Bidirectional 2-wire)

The serial interface of the SHTxx is optimized for sensor readout and power consumption and is not compatible with I²C interfaces, see FAQ for details.

2.2.1 Serial clock input (SCK)

The SCK is used to synchronize the communication between a microcontroller and the SHTxx. Since the interface consists of fully static logic there is no minimum SCK frequency.

2.2.2 Serial data (DATA)

The DATA tristate pin is used to transfer data in and out of the device. DATA **changes after the falling edge** and is **valid on the rising edge** of the serial clock SCK. During transmission the DATA line must remain stable while SCK is high. To avoid signal contention the microcontroller should only drive DATA low. An external pull-up resistor (e.g. 10 kΩ) is required to pull the signal high. (See Figure 2) Pull-up resistors are often included in I/O circuits of microcontrollers. See Table 5 for detailed IO characteristics.

(1) Each SHTxx is tested to be fully within RH accuracy specifications at 25 °C (77 °F) and 48 °C (118.4 °F)
(2) The default measurement resolution of 14bit (temperature) and 12bit (humidity) can be reduced to 12 and 8 bit through the status register.

2.2.3 Sending a command

To initiate a transmission, a "Transmission Start" sequence has to be issued. It consists of a lowering of the DATA line while SCK is high, followed by a low pulse on SCK and raising DATA again while SCK is still high.

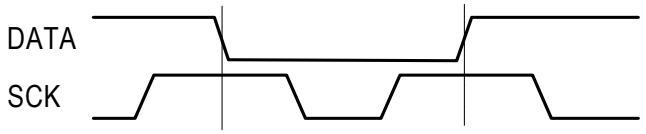


Figure 3 "Transmission Start" sequence

The subsequent command consists of three address bits (only "000" is currently supported) and five command bits. The SHTxx indicates the proper reception of a command by pulling the DATA pin low (ACK bit) after the falling edge of the 8th SCK clock. The DATA line is released (and goes high) after the falling edge of the 9th SCK clock.

Command	Code
Reserved	0000x
Measure Temperature	00011
Measure Humidity	00101
Read Status Register	00111
Write Status Register	00110
Reserved	0101x-1110x
Soft reset, resets the interface, clears the status register to default values wait minimum 11 ms before next command	11110

Table 2 SHTxx list of commands

2.2.4 Measurement sequence (RH and T)

After issuing a measurement command ('00000101' for RH, '00000011' for Temperature) the controller has to wait for the measurement to complete. This takes approximately 11/55/210 ms for a 8/12/14bit measurement. The exact time varies by up to ±15% with the speed of the internal oscillator. To signal the completion of a measurement, the SHTxx pulls down the data line and enters idle mode. The controller must wait for this "data ready" signal before restarting SCK to readout the data. Measurement data is stored until readout, therefore the controller can continue with other tasks and readout as convenient.

Two bytes of measurement data and one byte of CRC checksum will then be transmitted. The uC must acknowledge each byte by pulling the DATA line low. All values are MSB first, right justified. (e.g. the 5th SCK is MSB for a 12bit value, for a 8bit result the first byte is not used).
Communication terminates after the acknowledge bit of the CRC data. If CRC-8 checksum is not used the controller may terminate the communication after the measurement data LSB by keeping ack high.
The device automatically returns to sleep mode after the measurement and communication have ended.

Warning: To keep self heating below 0.1 °C the SHTxx should not be active for more than 10% of the time (e.g. max. 2 measurements / second for 12bit accuracy).

2.2.5 Connection reset sequence

If communication with the device is lost the following signal sequence will reset its serial interface:
While leaving DATA high, toggle SCK 9 or more times. This must be followed by a "Transmission Start" sequence preceding the next command. This sequence resets the interface only. The status register preserves its content.

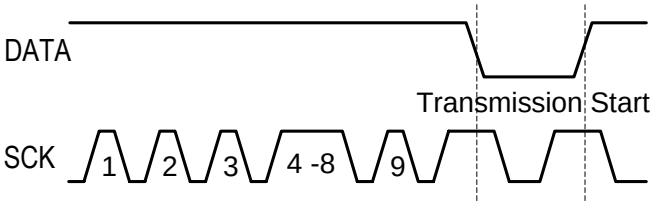


Figure 4 Connection reset sequence

2.2.6 CRC-8 Checksum calculation

The whole digital transmission is secured by a 8 bit checksum. It ensures that any wrong data can be detected and eliminated.
Please consult application note "CRC-8 Checksum Calculation" for information on how to calculate the CRC.

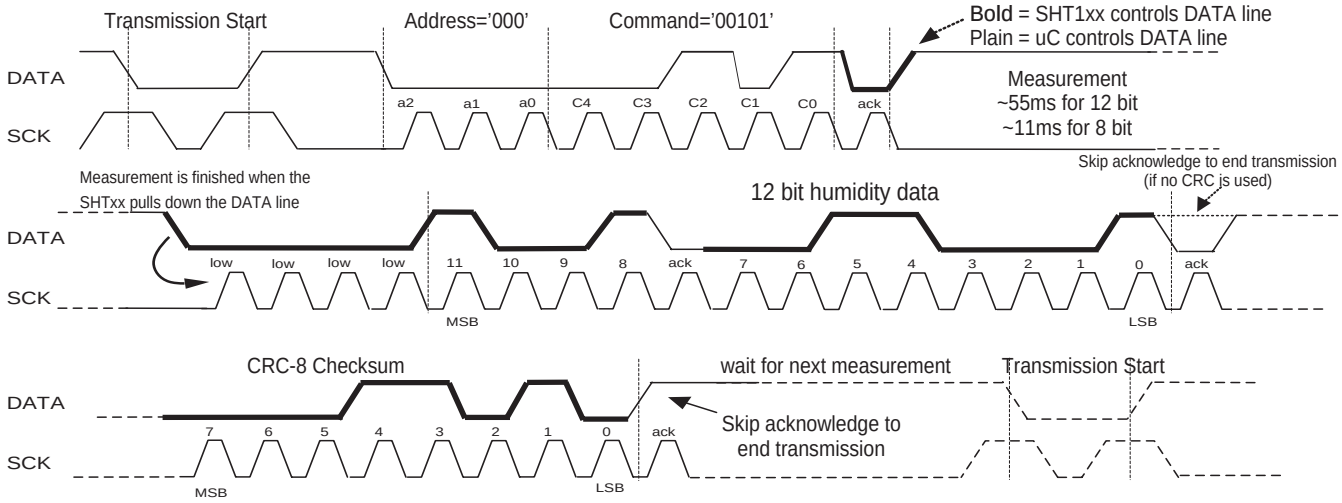


Figure 5 Example RH measurement sequence for value "0000'1001 ' 0011'0001"= 2353 = 75.79 %RH (without temperature compensation)

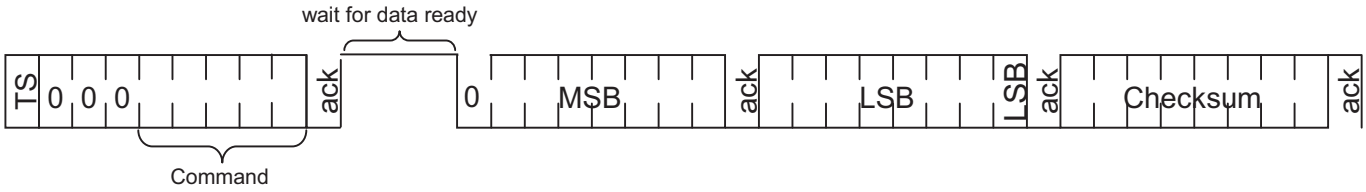


Figure 6 Overview of Measurement Sequence (TS = Transmission Start)

2.3 Status Register

Some of the advanced functions of the SHTxx are available through the status register. The following section gives a brief overview of these features. A more detailed description is available in the application note "Status Register"

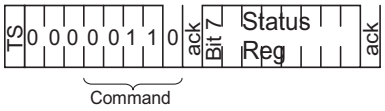


Figure 7 Status Register Write

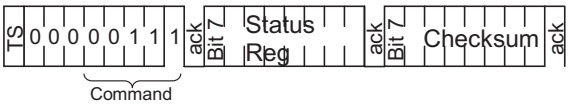


Figure 8 Status Register Read

Bit	Type	Description	Default	
7		reserved	0	
6	R	End of Battery (low voltage detection) '0' for Vdd > 2.47 '1' for Vdd < 2.47	X	No default value, bit is only updated after a measurement
5		reserved	0	
4		reserved	0	
3		For Testing only, do not use	0	
2	R/W	Heater	0	off
1	R/W	no reload from OTP	0	reload
0	R/W	'1' = 8bit RH / 12bit Temperature resolution '0' = 12bit RH / 14bit Temperature resolution	0	12bit RH 14bit Temp.

Table 3 Status Register Bits

2.3.1 Measurement resolution

The default measurement resolution of 14bit (temperature) and 12bit (humidity) can be reduced to 12 and 8bit. This is especially useful in high speed or extreme low power applications.

2.3.2 End of Battery

The "End of Battery" function detects VDD voltages below 2.47 V. Accuracy is ±0.05 V

2.3.3 Heater

An on chip heating element can be switched on. It will increase the temperature of the sensor by 5-15 °C (9-27 °F). Power consumption will increase by ~8 mA @ 5 V.
Applications:
By comparing temperature and humidity values before and after switching on the heater, proper functionality of both sensors can be verified.

- In high (>95 %RH) RH environments heating the sensor element will prevent condensation, improve response time and accuracy

Warning: While heated the SHTxx will show higher temperatures and a lower relative humidity than with no heating.

2.4 Electrical Characteristics[1]

VDD=5V, Temperature = 25 °C unless otherwise noted

Parameter	Conditions	Min.	Typ.	Max.	Units
Power supply DC		2.4	5	5.5	V
Supply current	measuring		550		µA
	average	2[2]	28[3]		µA
	sleep		0.3	1	µA
Low level output voltage		0		20%	Vdd
High level output voltage		75%		100%	Vdd
Low level input voltage	Negative going	0		20%	Vdd
High level input voltage	Positive going	80%		100%	Vdd
Input current on pads				1	µA
Output peak current	on			4	mA
	Tristated (off)		10		µA

Table 4 SHTxx DC Characteristics

	Parameter	Conditions	Min	Typ.	Max.	Unit
F_{SCK}	SCK frequency	VDD > 4.5 V			10	MHz
		VDD < 4.5 V			1	MHz
T_{RFO}	DATA fall time	Output load 5 pF	3.5	10	20	ns
		Output load 100 pF	30	40	200	ns
T_{CLx}	SCK hi/low time		100			ns
T_V	DATA valid time			250		ns
T_{SU}	DATA set up time		100			ns
T_{HO}	DATA hold time		0	10		ns
T_R/T_F	SCK rise/fall time			200		ns

Table 5 SHTxx I/O Signals Characteristics

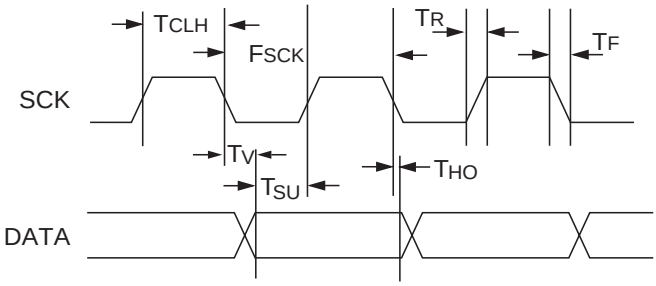


Figure 9 Timing Diagram

1) Parameters are periodically sampled and not 100% tested
(2) With one measurement of 8 bit accuracy without OTP reload per second
(3) With one measurement of 12bit accuracy per second

3 Converting Output to Physical Values

3.1 Relative Humidity

To compensate for the non-linearity of the humidity sensor and to obtain the full accuracy it is recommended to convert the readout with the following formula[1]:

$$RH_{linear} = c_1 + c_2 \bullet SO_{RH} + c_3 \bullet SO_{RH}^2$$

SO_{RH}	c_1	c_2	c_3
12 bit	-4	0.0405	$-2.8 * 10^{-6}$
8 bit	-4	0.648	$-7.2 * 10^{-4}$

Table 6 Humidity conversion coefficients

For simplified, less computation intense conversion formulas see application note “RH and Temperature Non-Linearity Compensation”.

The humidity sensor has no significant voltage dependency.

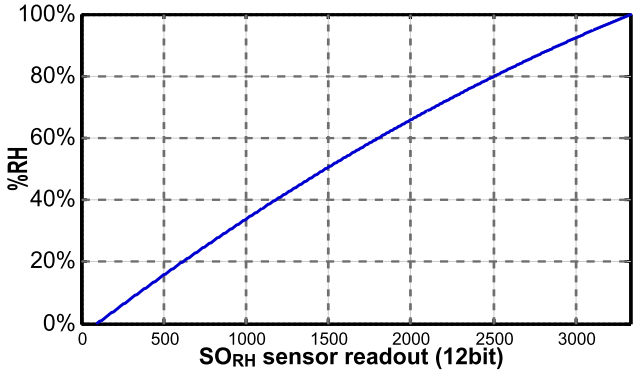


Figure 10 Conversion from SO_{RH} to relative humidity

3.1.1 Humidity Sensor RH/Temperature compensation

For temperatures significantly different from 25 °C (~77 °F) the temperature coefficient of the RH sensor should be considered:

$$RH_{true} = (T_{°C} - 25) \bullet (t_1 + t_2 \bullet SO_{RH}) + RH_{linear}$$

SO_{RH}	t_1	t_2
12 bit	0.01	0.00008
8 bit	0.01	0.00128

Table 7 Temperature compensation coefficients

This equals ~0.12 %RH / °C @ 50 %RH

3.2 Temperature

The bandgap PTAT (Proportional To Absolute Temperature) temperature sensor is very linear by design. Use the following formula to convert from digital readout to temperature:

$$Temperature = d_1 + d_2 \bullet SO_T$$

VDD	d_1 [°C]	d_1 [°F]
5V	-40.00	-40.00
4V	-39.75	-39.50
3.5V	-39.66	-39.35
3V	-39.60	-39.28
2.5V	-39.55	-39.23

SO_T	d_2 [°C]	d_2 [°F]
14bit	0.01	0.018
12bit	0.04	0.072

Table 8 Temperature conversion coefficients

For improved accuracies in extreme temperatures with more computation intense conversion formulas see application note “RH and Temperature Non-Linearity Compensation”.

3.3 Dewpoint

Since humidity and temperature are both measured on the same monolithic chip, the SHTxx allows superb dewpoint measurements. See application note “Dewpoint calculation” for more.

[1] Where SO_{RH} is the sensor output for relative humidity

4 Applications Information

4.1 Operating and Storage Conditions

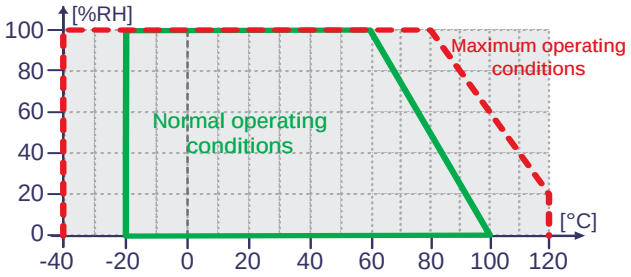


Figure 11 Recommended operating conditions

Conditions outside the recommended range may temporarily offset the RH signal up to ±3 %RH. After return to normal conditions it will slowly return towards calibration state by itself. See 4.3 "Reconditioning Procedure" to accelerate this process. Prolonged exposure to extreme conditions may accelerate ageing.

4.2 Exposure to Chemicals

Chemical vapors may interfere with the polymer layers used for capacitive humidity sensors. The diffusion of chemicals into the polymer may cause a shift in both offset and sensitivity. In a clean environment the contaminants will slowly outgas. The reconditioning procedure described below will accelerate this process. High levels of pollutants may cause permanent damage to the sensing polymer.

4.3 Reconditioning Procedure

The following reconditioning procedure will bring the sensor back to calibration state after exposure to extreme conditions or chemical vapors.
80-90 °C (176-194°F) at < 5 %RH for 24h (baking) followed by
20-30 °C (70-90°F) at > 74 %RH for 48h (re-hydration)

4.4 Temperature Effects

The relative humidity of a gas strongly depends on its temperature. It is therefore essential to keep humidity sensors at the same temperature as the air of which the relative humidity is to be measured.
If the SHTxx shares a PCB with electronic components that give off heat it should be mounted far away and below the heat source and the housing must remain well ventilated.
To reduce heat conduction copper layers between the SHT1x and the rest of the PCB should be minimized and a slit may be milled in between (see figure 13).

4.5 Membranes

A membrane may be used to prevent dirt from entering the housing and to protect the sensor. It will also reduce peak concentrations of chemical vapors. For optimal response times air volume behind the membrane must be kept to a minimum. For the SHT1x package Sensirion recommends the SF1 filter cap for optimal IP67 protection.

4.6 Light

The SHTxx is not light sensitive. Prolonged direct exposure to sunshine or strong UV radiation may age the housing.

4.7 Materials Used for Sealing / Mounting

Many materials absorb humidity and will act as a buffer, increasing response times and hysteresis. Materials in the vicinity of the sensor must therefore be carefully chosen. Recommended materials are: All Metals, LCP, POM (Delrin), PTFE (Teflon), PE, PEEK, PP, PB, PPS, PSU, PVDF, PVF
For sealing and gluing (use sparingly): High filled epoxy for electronic packaging (e.g. glob top, underfill), and Silicone. Outgassing of these materials may also contaminate the SHTxx (cf. 4.2). Store well ventilated after manufacturing or bake at 50°C for 24h to outgas contaminants before packing.

4.8 Wiring Considerations and Signal Integrity

Carrying the SCK and DATA signal parallel and in close proximity (e.g. in wires) for more than 10cm may result in cross talk and loss of communication. This may be resolved by routing VDD and/or GND between the two data signals.
Please see the application note "ESD, Latchup and EMC" for more information.
Power supply pins (VDD, GND) should be decoupled with a 100 nF capacitor if wires are used.

4.9 Qualifications

Extensive tests were performed in various environments. Please contact SENSIRION for detailed information.

Environment	Norm	Results(1)
Temperature Cycles	JESD22-A104-B -40 °C / 125 °C, 1000 cy	Within Specifications
HAST Pressure Cooker	JESD22-A110-B 2.3 bar 125 °C 85 %RH	Reversible shift by +2 %RH
High Temperature and Humidity	JESD22-A101-B 85 °C 85 %RH 1250h	Reversible shift by +2 %RH
Salt Atmosphere	DIN-50021ss	Within Spec.
Condensing Air	-	Within Spec.
Freezing cycles fully submerged	-20 / +90 °C, 100 cy 30min dwell time	Reversible shift by +2 %RH
Various Automotive Chemicals	DIN 72300-5	Within Specifications

Table 9 Qualification tests (excerpt)

4.10 ESD (Electrostatic Discharge)

ESD immunity is qualified according to MIL STD 883E, method 3015 (Human Body Model at ±2 kV)).
Latch-up immunity is provided at a force current of ±100 mA with T_{amb} = 80 °C according to JEDEC 17. See application note "ESD, Latchup and EMC" for more information.

(1) The temperature sensor passed all tests without any detectable drift. Package and electronics also passed 100%

5 Package Information

5.1 SHT1x (surface mountable)

Pin	Name	Comment
1	GND	Ground
2	DATA	Serial data, bidirectional
3	SCK	Serial clock, input
4	VDD	Supply 2.4 - 5.5 V
	NC	Remaining pins must be left unconnected

Table 10 SHT1x Pin Description

5.1.1 Package type

The SHT1x is supplied in a surface-mountable LCC (Leadless Chip Carrier) type package. The sensors housing consists of a Liquid Crystal Polymer (LCP) cap with epoxy glob top on a standard 0.8 mm FR4 substrate. The device is free of lead, Cd and Hg.
Device size is 7.42 x 4.88 x 2.5 mm (0.29 x 0.19 x 0.1 inch)
Weight 100 mg

The production date is printed onto the cap in white numbers in the form wwy. e.g. ”351” = week 35, 2001.

5.1.2 Delivery Conditions

The SHT1x are shipped in 12mm tape at 100pcs or 400pcs.. Reels are individually labelled with barcode and human readable labels. The Lot numbers allow full traceability through production, calibration and test.

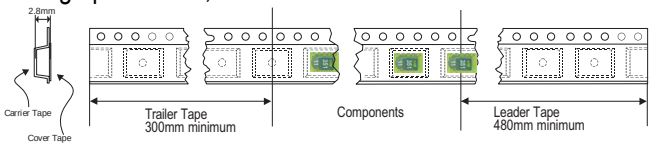


Figure 12 Tape configuration and unit orientation

5.1.3 Soldering Information

Standard reflow soldering ovens may be used. For details please see application note "soldering procedure".

For manual soldering contact time must be limited to 5 seconds at up to 350 °C.
After soldering the devices should be stored at >74 %RH for at least 24h to allow the polymer to rehydrate.
Please consult the application note “Soldering procedure” for more information.

5.1.4 Mounting Examples

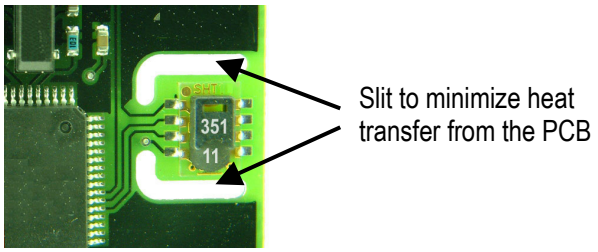


Figure 13 SHT1x PCB Mounting example

The SF1 membrane filter cap is available for optimal IP67 protection. When mounted through a housing the interior can be protected from the environment while still allowing high quality humidity measurements (see example below).

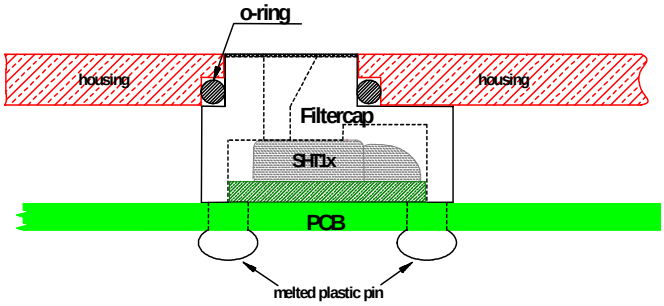


Figure 14 SF1 IP67 filter cap mounting example

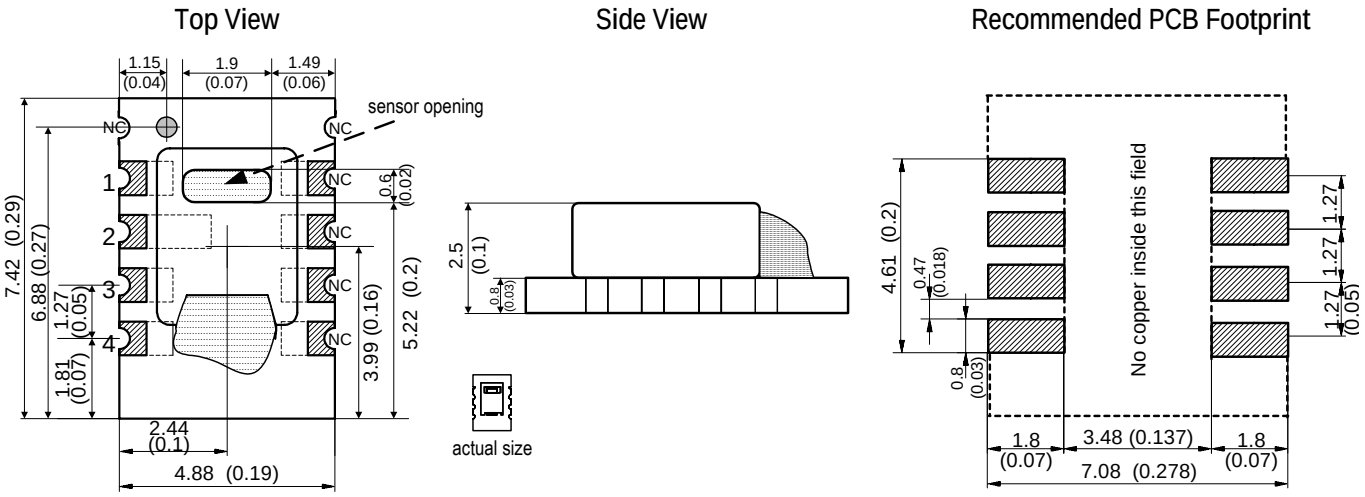


Figure 15 SHT1x drawing and footprint dimensions in mm (inch)

5.2 SHT7x (4-pin single-in-line)

Pin	Name	Comment
1	*SCK*	*Serial clock input*
2	*VDD*	*Supply 2.4 - 5.5 V*
3	*GND*	*Ground*
4	*DATA*	*Serial data bidirectional*

Table 11 SHT7x Pin Description

5.2.1 Package type[1]

The device is supplied in a single-in-line pin type package. The sensor housing consists of a Liquid Crystal Polymer (LCP) cap with epoxy glob top on a standard 0.6 mm FR4 substrate. The device is Cd and Hg free.
The sensor head is connected to the pins by a small bridge to minimize heat conduction and response times. The gold plated back side of the sensor head is connected to the GND pin.
A 100nF capacitor is mounted on the back side between VDD and GND.
All pins are gold plated to avoid corrosion. They can be soldered or mate with most 1.27 mm (0.05") sockets
e.g.: Preci-dip / Mill-Max 851-93-004-20-001 or similar
Total weight: 168 mg, weight of sensor head: 73 mg

The production date is printed onto the cap in white numbers in the form wwy. e.g. "351" = week 35, 2001.

5.2.2 Delivery Conditions

The SHT7x are shipped in 32 mm tape. These reeled parts in standard option are shipped with 500 units per 13 inch diameter reel. Reels are individually labelled with barcode and human readable labels.

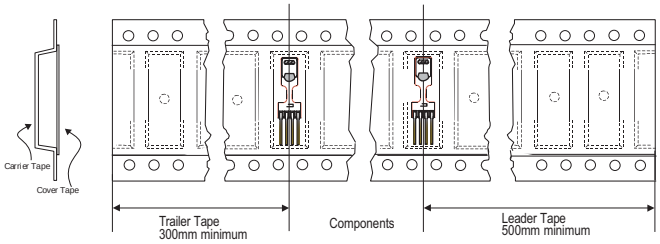


Figure 16 Tape configuration and unit orientation

5.2.3 Soldering Information[2]

Standard wave SHT7x soldering ovens may be used at maximum 235 °C for 20 seconds.
For manual soldering contact time must be limited to 5 seconds at up to 350 °C.
After wave soldering the devices should be stored at >74 %RH for at least 24 h to allow the polymer to rehydrate.
Please consult the application note "Soldering procedure" for more information.

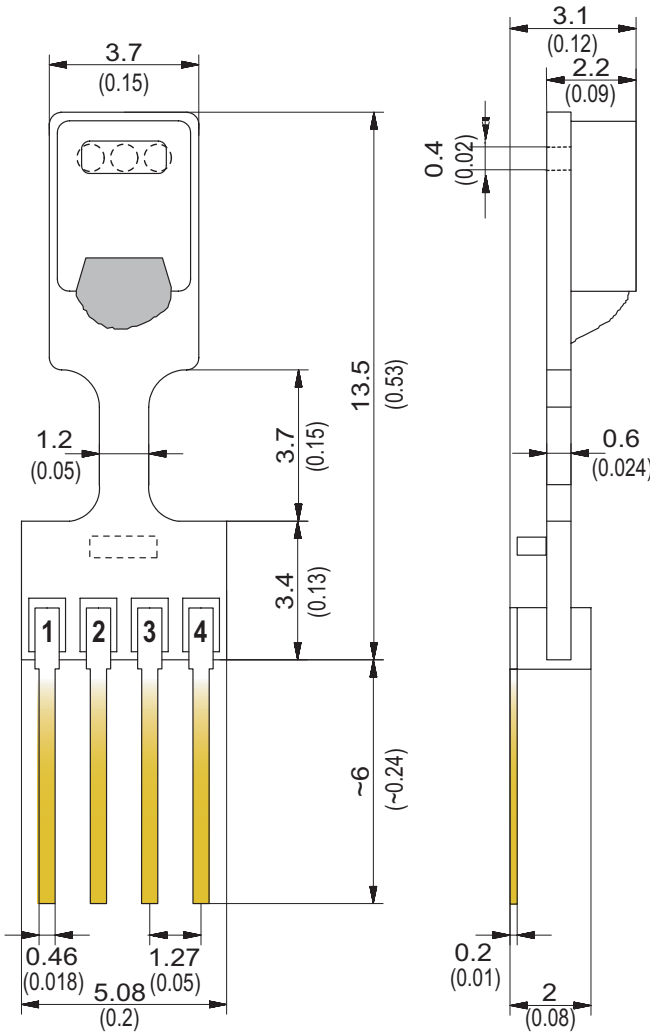


Figure 17 SHT7x dimensions in mm (inch)

[1] Other packaging options may be available on request.
[2] For maximum accuracy do not solder SHT75!

6 Revision history

Date	Version	Page(s)	Changes
February 2002	Preliminary	1-9	First public release
June 2002	Preliminary		Added SHT7x information
March 2003	Final v2.0	1-9	Major remake, added application information etc. Various small modifications
	V2.01	1-9	Typos, Graph labeling
July 2004	V2.02	1-9	Improved specifications, added SF1 information, improved wording

The latest version of this document and all application notes can be found at:
www.sensirion.com/en/download/humiditysensor/SHT1x_SHT7x.htm

7 Important Notices

7.1 Warning, personal injury

Do not use this product as safety or emergency stop devices or in any other application where failure of the product could result in personal injury. Failure to comply with these instructions could result in death or serious injury.

Should buyer purchase or use SENSIRION AG products for any such unintended or unauthorized application, Buyer shall indemnify and hold SENSIRION AG and its officers, employees, subsidiaries, affiliates and distributors harmless against all claims, costs, damages and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that SENSIRION AG was negligent regarding the design or manufacture of the part.

7.2 ESD Precautions

The inherent design of this component causes it to be sensitive to electrostatic discharge (ESD). To prevent ESD-induced damage and/or degradation, take normal ESD precautions when handling this product.
See application note “ESD, Latchup and EMC” for more information.

7.3 Warranty

SENSIRION AG makes no warranty, representation or guarantee regarding the suitability of its product for any particular purpose, nor does SENSIRION AG assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters can and do vary in different applications. All operating parameters, including “Typical” must be validated for each customer applications by customer’s technical experts.

SENSIRION AG reserves the right, without further notice, to change the product specifications and/or information in this document and to improve reliability, functions and design.

Headquarters and Sales Office

SENSIRION AG
Eggbühlstr. 14
P.O. Box
CH-8052 Zürich
Switzerland

Phone: + 41 (0)44 306 40 00
Fax: + 41 (0)44 306 40 30
e-mail: info@sensirion.com
http://www.sensirion.com/

Sensirion humidity sensors are available from:

find your local representative at:
www.sensirion.com/reps

DMATEK

ARM周邊配件系列

RFID射頻IC卡讀寫器

● 操作應用手冊

作業系統：WinCE 5.0 / 6.0　Android 2.1 / 2.2

配合平台：DMA-2440 / DMA-2440XP
DMA-NAV2443II / DMA-2443
DMA-NAV2450 / DMA-2450
DMA-6410 / DMA-6410L / DMA-6410XP
DMA-210U / DMA-210XP

目錄

目錄

第〇章　導 讀

0-1　如何開始

在本手冊中，筆者將於第一章中簡述RFID射頻IC卡讀寫器的特色，於第二章說明RFID的原理。第三章中將說明RFID讀寫器的命令碼和RFID卡片的規格介紹。第四章是Android系統上的RFID展示程式的使用介紹。第五章是WinCE系統的RFID展示程式的使用介紹。第六章是Android系統的程式Source講解。第七章是WinCE系統的程式Source講解。最後是聯繫方式。

0-2 光碟內容說明

在RFID射頻IC卡讀寫器的產品光碟中包含了許多實用的東西，讀者可以從產品光碟的README文件瞭解光碟的內容，主要包含以下的內容：

目錄名稱	主要檔描述
WinCE	WinCE 系統相關的資料
WinCE\Source	WinCE 系統 RFID 展示程式 Source，採用 VS2005 編寫
WinCE\RFID_COM1.exe	WinCE 系統 RFID 展示程式執行檔，對應 COM1 使用
WinCE\RFID_COM2.exe	WinCE 系統 RFID 展示程式執行檔，對應 COM2 使用
WinCE\RFID_COM3.exe	WinCE 系統 RFID 展示程式執行檔，對應 COM3 使用
WinCE\RFID_COM4.exe	WinCE 系統 RFID 展示程式執行檔，對應 COM4 使用
WinCE\RFID_43.dll	WinCE 系統 RFID 展示程式資源檔，對應 4.3 吋螢幕使用
WinCE\RFID_70.dll	WinCE 系統 RFID 展示程式資源檔，對應 7 吋螢幕使用
Android	Android 系統相關的資料
Android\RFID	Android 系統 RFID 程式 Source，採用 Eclipse 工具編寫
Android\RFID.apk	Android 系統 RFID 程式 APK 安裝執行檔。
PDF	手冊

表：DMATEK平台對RF500讀寫器的支援表。

平台	模组介面	Android系統	WinCE 6.0系統
DMA210-U	RF500 USB介面	√	—
	RF500 串列埠介面	√	√
DMA-6410XP	RF500 USB介面	—	—
	RF500 串列埠介面	√	√

註：√ 表示：支援；— 表示：暫不支援，升級後可支援；X 表示：不支援。

第一章　RFID 射頻 IC 卡讀寫器簡介

本手冊適用於長高公司的所有平台，作業系統為Android。

1-1　RFID 射頻 IC 卡讀寫器外觀

圖 RFID 射頻 IC 卡讀寫器外觀

1-2 RFID 射頻 IC 卡讀寫器詳細規格說明

RF系列非接觸式IC卡讀寫器是由主機、天線、串列介面等組成，透過RS232串列介面或USB介面能實現同PC主機或嵌入式設備的連接。目前該設備已廣泛地應用於門禁、考勤及高速公路、油站、停車場、公交等收費系統中。

- 適用於符合 MIFARE 標準的非接觸卡
- 支援 MIFARE 標準
- 工作頻率 13.56MHZ
- 以 106kbit/s 速率高速訪問射頻卡
- 資料加密和雙向驗證
- 防衝突，可同時讀取多張射頻卡
- 通信錯誤自動偵測
- 功能操作：讀、寫、初始化值、加值、減值、讀值和裝載密碼等
- 通訊介面：RS232 串列埠
- 串列傳輸速率：115200 bit/s
- 操作距離：平均 50 毫米
- 控制蜂鳴器鳴響功能
- 提供豐富的二次開發平台和應用範例

技術指標：

1. 支援 MIFARE 標準
2. 發射頻率：13.56MHz
3. 訪問卡速率：106Kbit/s
4. 介面功能：RS232 串列介面
5. 串列傳輸速率：115200bit/s
6. 工作電源：DC 5V±5%
7. 讀寫距離：平均爲 50mm（Mifare 標準卡讀寫距離）
8. 最大功耗：200mW

9. 環境溫度：0°C～50°C
10. 相對濕度：30%～95%
11. 重量：約 200 克

Mifare 1K 卡片特性介紹：

1. 8K 位元組 EEPROM
2. 分爲 16 個磁區，每個磁區包括 4 塊，每塊 16 個位元組，以塊爲存取單位
3. 用戶可自定義每個存儲塊的訪問條件
4. 每張卡有唯一序列號，爲 32 位
5. 具有防衝突機制，支援多卡操作
6. 非接觸傳送資料和無源(卡中無電源)
7. 至少 10 年資料儲存期
8. 至少 10 萬次擦寫
9. 讀寫距離：在 50-100mm 內(與天線形狀有關)
10. 工作頻率：13.56 MHZ
11. 通信速率：106kbit/s
12. 典型交易過程：<100 ms(不包括管理部分)
13. 溫度範圍：-20℃~50℃

第二章　RFID 原理概論

射頻識別技術（Radio-frequeney Identification，簡稱RFID）是無線電技術在自動識別領域應用中的具體運用。它透過射頻信號自動識別目標物件並獲取相關資料。近年來，隨著晶片技術、天線技術以及電腦技術的不斷發展，RFID系統的體積、功耗越來越小，同時成本越來越低、功能日趨靈活、操作快捷方便，加上其擅長多目標識別與運動目標識別，方便物品跟蹤和物流管理的突出特點，RFID系統日益廣泛的應用於各種生產生活場所，扮演著越來越重要的角色，被評爲「帶來了一個進化的無線市場」。

Radio-frequeney Identification（RFID）通稱「電子標籤技術」，作爲一種快速、即時、準確採集與處理資訊的高新技術和資訊標準化的基礎，被列爲本世紀十大重要技術之一。

RFID技術透過對實體物件（包括零售商品、物流單元、集裝箱、貨運包裝、生產零部件等）的唯一有效標識，被廣泛應用於生產、零售、物流、交通等各個行業。RFID技術已逐漸成爲企業提高物流供應鏈管理水準，降低成本，企業管理資訊化，參與國際經濟大循環，增強企業核心競爭力不可缺少的技術工具和手段。RFID技術的興起並不是因爲它是一項新技術，而是因爲這項技術已經開始成熟並逐漸具備了走向實際應用的能力。

2-1 RFID 特色與歷史

RFID 技術是從 20 世紀 90 年代興起的一項自動識別技術。它是透過磁場或電磁場，利用無線射頻方式進行非接觸雙向通信，以達到識別目的並交換資料，可識別高速運動物體並可同時識別多個目標。與傳統識別方式相比，RFID 技術無需直接接觸、無需光學可視、無需人工干預即可完成資訊輸入和處理，操作方便快捷。RFID 能廣泛用於生產、物流、交通、運輸、醫療、防僞、跟蹤、設備和資產管理等需要收集和處理資料的應用領域，被認爲是條碼標籤的未來替代品。

就發展的歷史而言，RFID 可算是「老兵新姿」，因爲 RFID 並不是一個嶄新的技術，RFID 在歷史上的首次應用可以追溯到第二次世界大戰期間(約 1940 年代)，其當時的功能是用於分辨出敵方飛機與我方飛機。目前世界上的飛安管制系統仍是以此爲概念。到了 1970 年代末期，美國政府透過 Los Alamos 科學實驗室將 RFID 技術轉移到民間。RFID 技術最先的商業應用是使用在牲畜身上。到了 1980 年代，美國與歐洲的幾家公司開始著手生產 RFID 標籤。在今日，RFID 技術已經被廣泛應用於各個領域，從門禁管制、牲畜管理，到物流管理，皆可以見到其蹤跡。

它的優勢及特點主要表現在：

I. **快速掃描**

條碼一次只能有一個條碼受到掃描；RFID 辨識器可同時辨識讀取數個 RFID 標籤。

II. **體積小型化、形狀多樣化**

RFID 在讀取上並不受尺寸大小與形狀限制，不需爲了讀取精確度而配合紙張的固定尺寸和印刷品質。此外，RFID 標籤更可往小型化與多樣形態發展，以應用於不同產品。

III. **抗污染能力和耐久性**

傳統條碼的載體是紙張，因此容易受到污染，但 RFID 對水、油和化學藥品等物質具有很強抵抗性。此外，由於條碼是附於塑膠袋或外包裝紙箱上，所以特別容易受到折損；RFID 標籤是將資料存在晶片中，因此可以免受

汙損。

IV. 可重複使用

現今的條碼印刷上去之後就無法更改，RFID 標籤則可以重複地新增、修改、刪除。RFID 標籤內儲存的資料，方便資訊的更新。

V. 穿透性和無螢幕障閱讀

在被覆蓋的情況下，RFID 能穿透紙張、木材和塑膠等非金屬或非透明的材質，並能進行穿透性通信。而條碼掃描機必須在近距離而且沒有物體阻擋的情況下，才可辨讀條碼。

VI. 資料的記憶容量大

一維條碼的容量是 50Bytes，二維條碼最大的容量可儲存 2 至 3000 字元，RFID 最大的容量則有數 MgeaBytes。隨著記憶載體的發展，資料容量也有不斷擴大的趨勢。未來物品所需攜帶的資料量會越來越大，對標籤所能擴充容量的需求也相應增加。

VII. 安全性

RFID 承載的是電子式資訊，其資料內容可經由密碼保護，使其內容不易被偽造及變造。近年來，RFID 因其所具備的遠距離讀取、高儲存量等特性而備受矚目。它不僅可以幫助一個企業大幅提高貨物、資訊管理的效率，還可以讓銷售企業和製造企業互聯，從而更加準確地接收回饋資訊，控制需求資訊，優化整個供應鏈。

在統一的標準平台上，RFID 標籤在整條供應鏈內任何時候都可提供產品的流向資訊，讓每個產品資訊有了共同的溝通語言。透過電腦互聯網就能實現物品的自動識別和資訊交換與共用，進而實現對物品的透明化管理，就能實現真正意義上的「物聯網」。

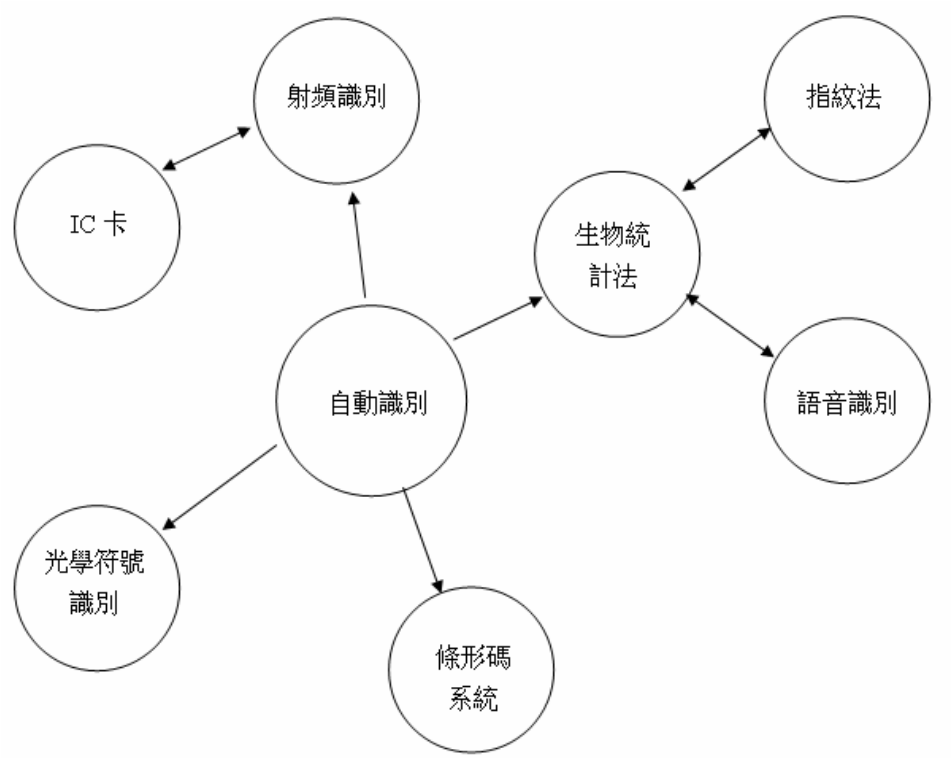


圖 2-1　自動識別方法綜合示意圖

2-2　RFID 的硬體組成

射頻識別 RFID 系統一般至少由以下兩部分構成：

1. 應答器(RF Code 或稱感應器/電子標籤)：

電子標籤由標籤天線和晶片組成，在實際應用上是放置在要被識別的物體上。標籤天線的作用是有效地發射和接收射頻載波，晶片中則儲存著電子標籤的資訊。根據電子標籤供電方式不同，可以分爲有源標籤(ActiveTag)和無源標籤(PassiveTag)。有源標籤內部裝設有電源(通常爲電池)，標籤接收和發送資料靠標籤自身的電源供電。無源標籤中則沒有電源，從電磁波中吸收能量啓動，然後進行通信。有源標籤可讀距離較大，但是壽命較短，而且需要維護；無源標籤可讀距離較

小，但壽命較長，不需要特別維護。本實驗中用的標籤是無源的。

2. 閱讀器(或稱讀寫器/Reader)：

可以是單讀取或可讀可寫的裝置，取決於所使用的結構和技術。

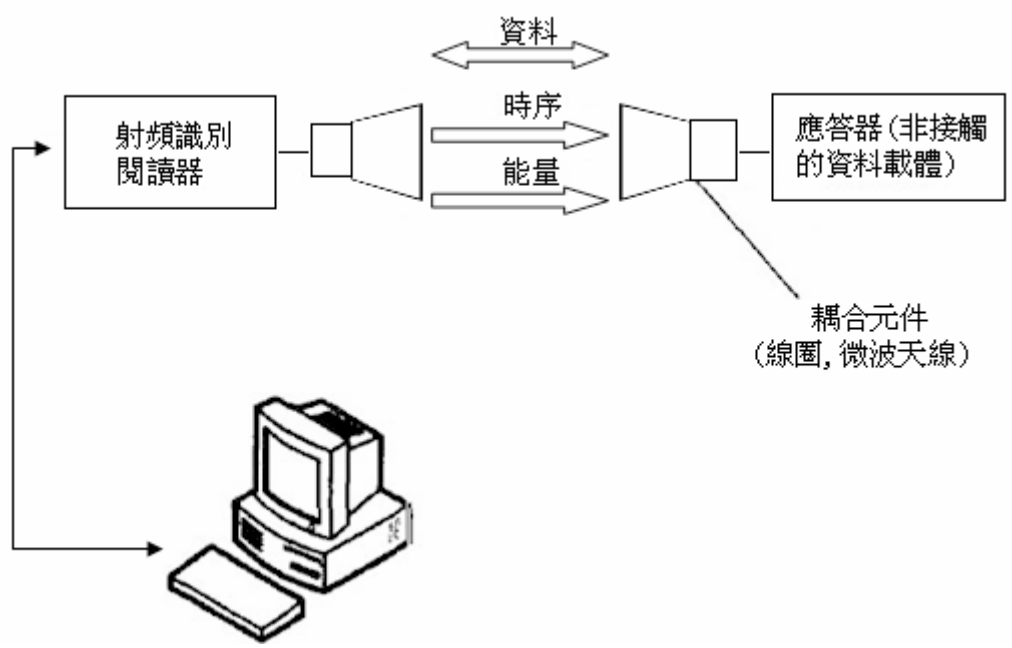


圖 2-2　閱讀器與應答器是各種射頻識別系統的基本組成部分

應答器中一般儲存有約定格式的編碼資料，用以唯一標識標籤所附著的物體。閱讀器透過天線發送出一定頻率的射頻信號，當應答器進入閱讀器工作區域時，其天線產生感應電流，應答器則由電磁波獲得能量而被啓動，並向讀寫器發送出自身編碼等資料資訊。閱讀器接收到來自應答器的載波信號，對接收的信號進行解調和解碼後送至電腦主機進行處理。電腦系統根據邏輯運算判斷該標籤的合法性，針對不同的設定做出相應的處理和控制，發出指令信號，應答器的資料解調部分從接收到的射頻脈衝中解調出資料並送到控制邏輯，控制邏輯接收指令完成儲存、發送資料或其他操作。

一台典型的閱讀器包含有高頻模組(發送器和接收器)，控制單元以及與應答器相連接的耦合元件，此外許多閱讀器還有附加的介面(如 RS232、RS485 等)，而應答

器往往由耦合元件以及微電子晶片組成。圖 2-3 爲 RFID 的工作原理框圖。

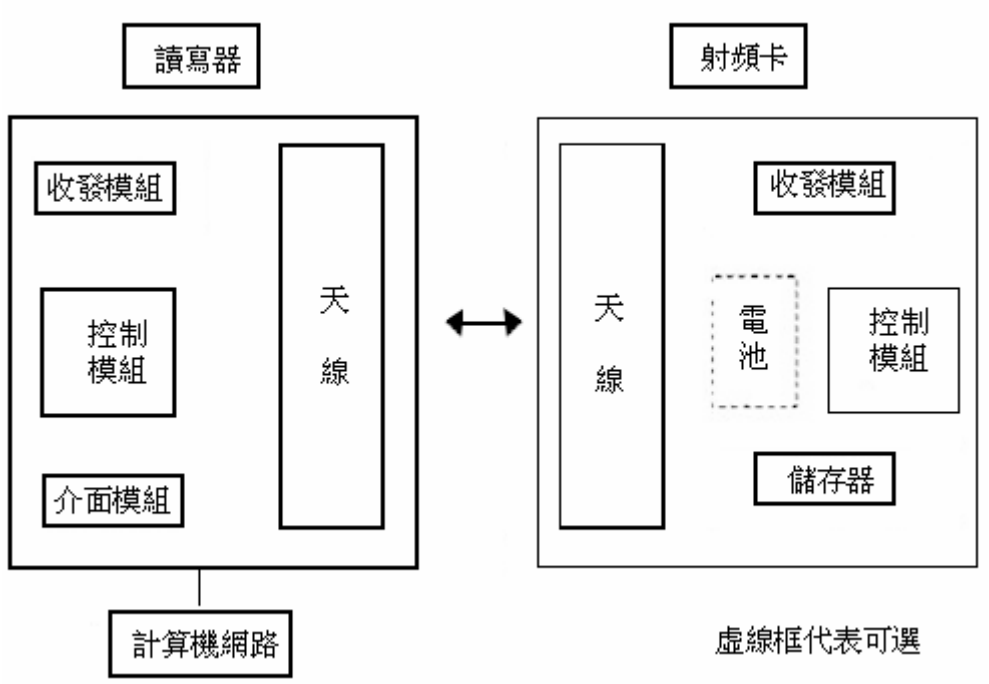


圖 2-3　RFID 的原理框圖

射頻識別技術依採用頻率不同可分爲低頻系統和高頻系統兩大類；根據電子標籤內是否裝有電池爲其供電，又可將其分爲有源系統和無源系統兩大類；從電子標籤內保存的資訊注入的方式可將其爲分積體電路固化式、現場有線改寫式和現場無線改寫式三大類；根據讀取電子標籤資料的技術實現手段，可將其分爲廣播發射式、倍頻式和反射調製式三大類。

RFID 低頻系統的工作頻率小於 30MHz，典型工作頻率有 25KHz、225KHz、13.56MHz。其特點是電子標籤的成本較低、標籤儲存的資料量較少、閱讀距離較短，子標籤外形多樣，閱讀天線方向性不強。

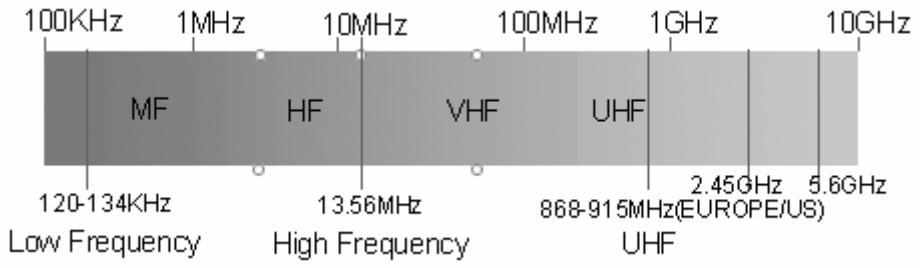


圖 2-4　RFID 系統應用頻段示意圖

RFID 高頻系統的工作頻率大於 40ONHIz，典型的工作頻段有 9I5MHz、2.45GHz 及 5.8GHz。其特點是電子標籤及閱讀器成本均較高、標籤內儲存的資料量較大、閱讀距離較遠，適應物體高速運動性能好、外形一般爲卡狀、閱讀天線及電子標籤天線均有較強的方向性。

2-3　射頻識別系統的兩種耦合方式

RFID 系統的耦合方式也因頻率高低而不同。當頻率低時，由於識別距離較近，系統採用電感藕合；而頻率高時，因其識別距離較遠，系統採用電磁反向散射耦合。

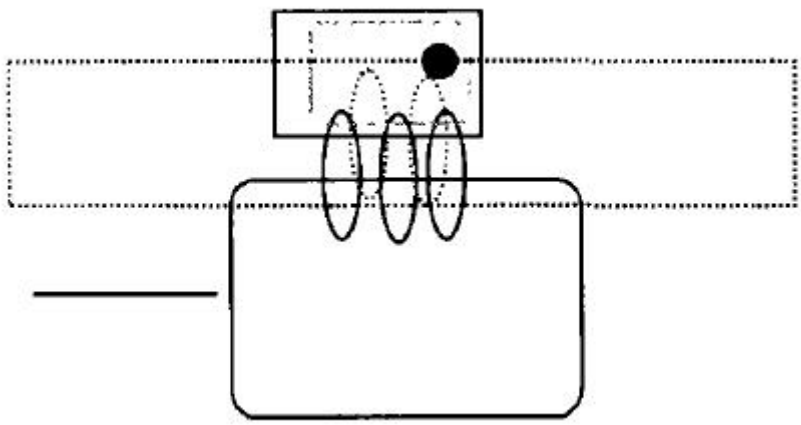


圖 2-5　電感耦合

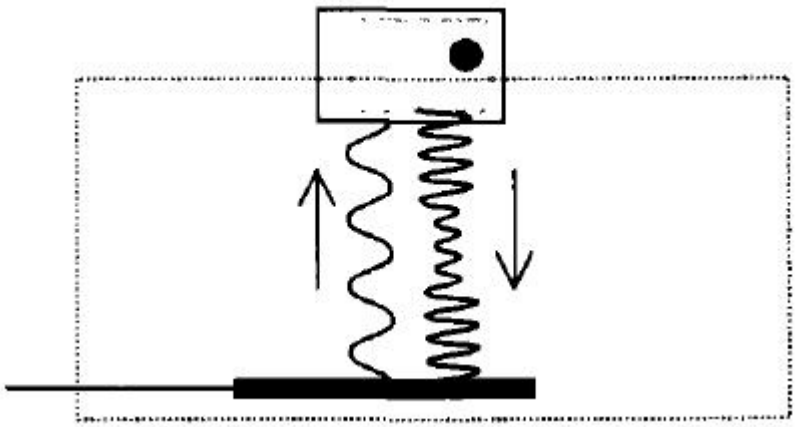


圖 2-6　電磁反向散射耦合

電感耦合透過空間高頻交變磁場實現祸合，依據的是電磁感應定律。電感耦合採用的天線形式一般都爲線圈，如圖 2-5 所示。耦合的實質是讀寫器天線線圈的交變磁力線穿過電子標籤天線的線圈，並在標籤天線的線圈中產生感應電壓。在耦合的過程中，利用的是讀寫器天線線圈產生的未輻射出去的交變磁能相當於天線的輻射近場情況。電感耦合方式適合於在較低頻率工作的近距離射頻識別系統。典型的工作頻率有：125kHz、225kHz 和 13.56MHZ。識別作用距離小於 lm，典型作用距離爲 10cm～20cm。

電磁反向散射耦合：發射出去的電磁波，碰到目標後反射，同時攜帶回目標資訊，依據的是電磁波的空間傳播規律。這種情況下，讀寫器天線與電子標籤天線是真正意義上的天線（有效地輻射出或接收到電磁波），如圖 2-6 所示。耦合的實質是讀寫器天線輻射出的電磁波照射到電子標籤天線後形成反射（如同雷達發現目標一樣）回波，反射回波再被讀寫器天線接收。耦合過程中，利用的是讀寫器天線輻射出的交變電磁能，相當於天線的輻射遠場情況。電磁反向散射耦合方式適合於在超高頻、微波頻段工作的遠距離射頻識別系統。典型的工作頻率有：433MHz、915MHz、2.45GHz、5.SGHz。識別作用距離大於 lm，典型作用距離爲 3m～10m。

2-4　研究現狀及發展趨勢

台灣的 RFID 應用中，最具代表性的傑作非悠游卡莫屬了，憑藉內部的 RFID 晶片與四周的線圈，凡是搭公車、捷運及停車場停車皆可用一張悠遊卡解決，未來甚至連計程車與路邊停車都以悠遊卡收費。交通大學科管所及電子所的 11 名研究生，以「IntelligencenIside-RFDI 無線辨識系統」專案，榮獲第二屆「台灣工業銀行創業大賽」首獎。他們的創業計畫是利用 RFID 在數位資訊、無線通訊傳輸及加密技術上的優點，應用到航空及貨運運輸、物流倉儲、保全系統、人員管理及動物晶片上。該隊出線的原因除因計畫周延、分析縝密、表達技巧熟穩外，更重要的是對新產品價格及市場優勢的掌握，顯示該計畫具有的商機。

另外中華、裕隆等台灣汽車業者早在幾年前就已經引入使用來做零件管理，由於汽車業組裝不但零件繁多，且步驟複雜，因此廠商運用 RFID 晶片標籤，來確認每一零件、步驟是否組裝完成。由於射頻識別的電波不會干擾到院內醫療儀器，在 SARS 期間，工研院與台北醫學大學更是將 RFID 充分運用醫學上。工研院在新竹東元醫院中裝置感應器，並且讓院內所有人都配有裝有 RFID 晶片的識別證，一旦查覺有人發病，二十分鐘內就可以清楚知道發病者和活動路徑；而台北醫學大學也使用配置 RFDI 晶片及體溫感測器的腕帶型標籤來掌握病患與員工的狀況及接觸史。未來，RFID 晶片更可以藉由 PHS 及 ADSL 傳送訊號以提供獨居老人居家生活照顧呢!而台灣部分高科技製造業者，如晶圓廠、封裝廠，也有以 RFID 追蹤晶圓及容易遺失物品的應用方式。

RFID 的發展趨勢：RFID 的應用範圍非常的廣泛，舉凡我們日常生活中的食、衣、住、行、育、樂所有牽涉到的物品，皆能使用 RFID。例如：

- 門禁管制：人員出入門禁監控、管制及上下班人事管理。
- 回收資產：棧板、貨櫃、台車、籠車等可回收容器管理。
- 貨物管理：航空運輸的行李識別、存貨、物流運輸管理。
- 物料處理：工廠的物料清點、物料控制系統。
- 廢物處理：垃圾回收處理、廢棄物管控系統。

- 醫療應用：醫院的病歷系統、危險或管制之生化物品管理。
- 交通運輸：高速公路的收費系統。
- 防盜應用：超市的防盜、圖書館或書店的防盜管理。
- 動物監控：畜牧動物管理、寵物識別、野生動物生態的追蹤。
- 自動控制：汽車、家電、電子業之組裝生產。
- 聯合票證：聯合多種用途的智慧型儲值卡、紅利積點卡。

RFID 應用分爲兩個層次，一是在閉環系統內部應用；二是在開放系統中應用。目前 RFID 的應用還主要是在閉環系統（比如企業或單位內部）內應用，使用獨立的一套設備、頻率、流程和編碼標準體系。因此其市場規模受到了極大的限制，但隨 RFID 技術的發展演進以及成本的降低，RFID 勢必向外部拓展，成爲一個開放式的資訊共用鏈條。未來幾年內 RFID 技術將主要以供應鏈的應用爲贏利的主體，從採購、倉儲、生產、包裝、卸載、流通加工、配送、銷售到服務，這些供應鏈上的業務流程和環節來增強企業對商流、物流、資訊和資金的掌控能力。RFID 在製造業、機場、汽車、醫藥、醫療保健這五大領域深具潛力。RFID 技術的發展趨勢必然會朝著標籤成本的降低，讀寫距離的提高，標籤存儲容量增大，處理時間縮短這些方向前進。

2-5　RFID 存在的問題

雖然 RFID 的應用相當廣泛，且具有極大的商機與市場，但 RFID 目前仍存在部份問題，例如：

I. 技術不成熟：RFID 是個充滿希望的行業，但它的推廣需要每一個環節的堅定與成熟，包括標準的制訂、晶片設計與製造技術、天線設計與製造技術、晶片封裝技術、讀寫設備開發與生產技術、系統集成和資料管理軟體平台、應用系統開發等。最具潛力的 UHF 頻段市場上能支援的設備並不多，射頻識別訊號容易被物體所阻斷，這些都是技術上的挑戰。

II. 成本居高不下：全球的零售業龍頭沃爾瑪（Wal-Mart Stores, Inc.）在 2003 年 6 月宣布導入 RFID 技術時，當時 RFID 標籤晶片的價格爲 50 美分，然而到了 2008 年第一季時，RFID 的採購價仍約爲 20 美分（以當時的匯率換算約新台幣 6.49 元），即使採購量達到百萬個，其價格仍爲 15 美分（以當時的匯率換算約新台幣 4.87 元）。沃爾瑪的 RFID 技術合作商 Tag-isties 公司曾經表示，一套 RFID 系統在美國定價（包括項目實施）一般在 250 萬美元左右，若是大型系統的價格還要更高。

目前業界對於 RFID 標籤市場都是以替換條碼的前提來估算，然而條碼系統相對於 RFID 標籤更是擁有低價位和基礎設置完善等既有優勢。目前條碼單價是 3 美分，但是根據美國市場研究機構 In-Stat/MDR 分析認爲未來幾年內，RFID 標籤的成本最低約在 15 美分而最高將超越 100 美元以上。此外，業界也估計 RFID 的年需求量至少必須達到 100 億個，才有讓採購價成本降到 5 美分。因此，目前 RFID 標籤本身成本即比條碼高，除非 RFID 標籤的經濟規模達到與條碼接近的程度，成本和基礎設施問題才能逐漸克服。

III. 標準不統一的問題：由於 RFID 缺乏統一技術標準，目前的技術還不能保證所有的標籤和閱讀器實現相容。因此，在希望採用 RFID 技術的倉儲系統中，也許需要不止一種閱讀器，以此來分別處理來自不同硬體提供商生產的不同標籤。

IV. 侵犯隱私權的問題：由於在非接觸的條件下，可以對標籤中的資料進行讀取，這引發了人們對 RFID 技術侵犯個人隱私權的爭議，其中又涉及到各個國家對隱私

權的法律保護不盡相同等問題。

V. 失業問題：採用射頻識別系統後，將接手原來由人工完成的工作並進一步取代人工作業，其衍生而來的的問題，將是許多的勞工面臨失去工作的危機。

VI. 安全保密問題：儘管 RFID 晶片的記憶體容量只能存儲 128 個字元的資訊，但還是容易被軟體病毒感染，這是由 2006 年於比薩如開的電腦學術會上一個歐洲的電腦研究團體指出的，大多數電腦安全專家對其是否感染病毒的可能性並不重視，但對 RFID 晶片試圖進入監控等安全領域仍存在頗多爭議。

第三章　如何開發程式

對於RFID最典型的應用就是使用讀寫器來讀寫RFID標籤中的資料。從而實現門禁，電子錢包等的功能。

在自己開發程式來對RFID卡片進行讀寫操作之前，需要先瞭解兩大方面的內容，一是瞭解RFID卡片的結構，這樣才知道資料是如何存儲的。二是瞭解使用什麼樣的命令來控制讀寫器進行讀寫操作。

3-1　Mifare 1K 射頻卡介紹

電氣特性：

- 容量爲 8K 位元 EEPROM
- 分爲 16 個磁區，每個磁區爲 4 塊，每塊 16 個位元組，以塊爲存取單位
- 每個磁區有獨立的一組密碼及訪問控制
- 每張卡有唯一序列號，爲 32 位
- 具有防衝突機制，支持多卡操作
- 無電源，自帶天線，內含加密控制邏輯和通訊邏輯電路
- 工作溫度：-20℃～50℃
- 工作頻率：13.56MHZ
- 通信速率：106KBPS
- 讀寫距離：可達 10cm(與讀寫器以及卡天線尺寸有關)

- 資料保存期爲 10 年，可改寫 10 萬次，讀不限次

工作原理：

卡片由一個捲繞天線和特定用途積體電路模組組成。其中，模組由一個高速(106KB 串列傳輸速率)的 RF 介面。一個控制單元和一個 8K 位元 E2PROM 組成。讀寫器向 MF1 卡發出一組固定頻率(13.56MHZ)的電磁波，卡片內有一個 LC 串聯諧振電路，其頻率與讀寫器發射的頻率相同，在電磁波的激勵下，LC 諧振電路產生共振，從而使諧振電容內有了電荷，在這個電容的另一端，接有一個單嚮導通的電子泵，將電容內的電荷送到模組存儲電容內儲存，當所積累的電荷達到 2V 以上時，此電容可作爲電源向模組電路提供工作電壓，將卡內資料發射出去或接收讀寫器的資料。

存儲結構：

存儲區分 4 塊（塊 0，塊 1，塊 2，塊 3），按塊號編址爲 0～63 共 64 塊。存儲區的分佈圖如下：

磁區號	序號	0 1 2 3 4 5	6 7 8 9	10 11 12 13 14 15	塊號
	0	廠商代碼區（固化不可更改）			0
	1	資料區 1			1
0	2	資料區 2			2
	3	A 密碼	存取控	B 密碼	3
	0	資料區 0			4
	1	資料區 1			5
1	2	資料區 2			6
	3	A 密碼	存取控	B 密碼	7
		⋮			
		資料區 0			
		資料區 1			
		資料區 2			
		A 密碼	存取控	B 密碼	
	0				60
	1	61			
	2	62			
	3	63			

廠商代碼區：第 0 磁區的塊 0（即絕對位址 0 塊）用於存放廠商代碼，已經固化，不可更改。

資料區：所有磁區都有 3 塊（每塊 16 個位元組）存儲資料。（磁區 0 只有兩個資料塊和一個制度廠商代碼塊）。

磁區控制塊（塊 3）：每個磁區都有一個磁區控制塊，包括：

- 密碼 A（0-5 位元組），讀取時始終返回 0。
- 存取控制區（6-9 位元組）。
- 密碼 B（10-15 位元組），讀取時，依據存取控制條件的不同，可以顯示。

控制屬性：

每個磁區的用戶密碼和存取控制條件都是獨立設置的，可以根據實際需要設定各自的密碼及存取控制。在存取控制中，每個塊都有三個控制位相對應，用以決定某資料塊或控制塊的讀寫條件，定義爲："CXy"，見表 1 所示。

其中 CX 代表每塊控制位號(C1～C3)，y 代表該磁區內某塊號。例如 C12 即爲當前磁區內塊 2 的第 1 控制位，以此類推。

表 1：控制位定義"CXy"

塊 0	C10	C20	C30	用戶資料塊，(0 區 0 塊除外)
塊 1	C11	C21	C31	用戶資料塊
塊 2	C12	C22	C32	用戶資料塊
塊 3	C13	C23	C33	密匙存取控制塊

表 2：三個控制位元在存取控制位元組中的位置 (注：" _b" 表示取反)

	bit 7	6	5	4	3	2	1	0
位元組 6	C23_b	C22_b	C21_b	C20_b	C13_b	C12_b	C11_b	C10_b
位元組 7	C13	C12	C11	C10	C33_b	C32_b	C31_b	C30_b
位元組 8	C33	C32	C31	C30	C23	C22	C21	C20
所屬塊	塊 3 控制位	塊 2 控制位	塊 1 控制位	塊 0 控制位	塊 3 控制位	塊 2 控制位	塊 1 控制位	塊 0 控制位

各磁區資料塊 0～塊 2 的三個控制位以正反兩種形式存在於塊 3 的存取控制位元組中，它決定了該塊的訪問許可權(例如進行減值及初始化值操作必須驗證 KEY A，進行加值操作必須驗證 KEY B，等等)。三個控制位元在存取控制位元組(6～9 位元組)中的許可權如下(陰影區的存取控制爲廠商初始值；位元組 9 爲備用位元組，預設值爲 69)：

表 3：資料塊的存取控制許可權(y=塊 0，塊 1，塊 2)

C1y C2y C3y	讀	寫	加值	減值，初始化
0 0 0	KeyA|B	KeyA|B	KeyA|B	KeyA|B
0 1 0	KeyA|B	Never	Never	Never
1 0 0	KeyA|B	KeyB	Never	Never
1 0 0	KeyA|B	KeyB	KeyB	KeyA|B
0 0 1	KeyA|B	Never	Never	KeyA|B
0 1 1	KeyB	KeyB	Never	Never
1 0 1	KeyB	Never	Never	Never
1 1 1	Never	Never	Never	Never

注釋：表 3 中，KeyA|B 表示密碼 A 或密碼 B，Never 表示任何條件下不能實現。

例如，某區塊的 3 個存取控制位 C1y，C2y，C3y=000 時(廠商預設的初始值，見陰影區)，驗證密碼 A 或密碼 B 正確後可讀出/可寫入/可加值/減值及初始化操作。該初始值主要供制卡和髮卡商檢測心片功能使用，確認所有讀寫/加密功能均正常(存

取控制初始值"ff078069"，請參考"步驟舉例"自行驗算)後，再依據使用需要和參照表 4 表 5 設置新的存取控制許可權值，進行用戶資料操作和修改新的用戶密碼。

再如當某區塊 0 的存取控制位 C10，C20，C30 的設置均=100 時，驗證密碼 A 或密碼 B 正確後可讀出其資料；只有驗證密碼 B 正確後才可允許改寫資料；不能進行加值，減值等操作。

控制塊（塊 3）的存取控制與資料塊（塊 0、1、2）不同，它的存取控制如下：

表 4：塊 3 的存取控制許可權(供髮卡商及用戶設置許可權時參考)

控制位設置值			密碼 A 許可權		存取控制許可權		密碼 B 許可權	
C13	C23	C33	讀	寫	讀	寫	讀	寫
0	0	0	Never	KeyA|B	KeyA|B	Never	KeyA|B	KeyA|B
0	1	0	Never	Never	KeyA|B	Never	KeyA|B	Never
1	0	0	Never	KeyB	KeyA|B	Never	Never	KeyB
1	1	0	Never	Never	KeyA|B	Never	Never	Never
0	0	1	Never	KeyA|B	KeyA|B	KeyA|B	KeyA|B	KeyA|B
0	1	1	Never	KeyB	KeyA|B	KeyB	Never	KeyB
1	0	1	Never	Never	KeyA|B	KeyB	Never	Never
1	1	1	Never	Never	KeyA|B	Never	Never	Never

例如：當塊 3 的存取控制位 C13 C23 C33 = 100 時，表示：密碼 A：不可讀(隱藏)，驗證 KEYB 正確後，可寫(或更改)；存取控制：驗證 KEYA 或 KEYB 正確後，可讀不可寫(防寫)；密碼 B：不可讀，驗證 KEYB 正確後可寫。

又如：當塊 3 的存取控制位 C13 C23 C33 = 110 或者 111 時，除存取控制值需要在驗證 KEYA 或 KEYB 正確後僅僅可讀外，其他如存取控制值的改寫，密碼 A，密碼 B 的讀寫許可權均被鎖死而無法訪問!

下面舉例說明各磁區塊 3 的控制位元如何設置資料區和控制區的訪問許可權。

以廠商預設控制碼：FF 07 80 69 爲例。

1、對"FF 07 80 69"值進行計算，該值定位于各區塊 3 的 6，7，8，9 四個位元組內，位元組 6=FF，位元組 7=07，位元組 8=80，位元組 9=69(預設值，

不予計算)。

2、轉換爲二進位碼

位元組 6 = FF = 1 1 1 1 1 1 1 1

位元組 7 = 07 = 0 0 0 0 0 1 1 1

位元組 8 = 80 = 1 0 0 0 0 0 0 0

3、把上面的二進位資料填入到上面的表二中

	bit 7	6	5	4	3	2	1	0
位元組 6	1	1	1	1	1	1	1	1
位元組 7	0	0	0	0	0	1	1	1
位元組 8	1	0	0	0	0	0	0	0
所屬塊	塊 3 控制位	塊 2 控制位	塊 1 控制位	塊 0 控制位	塊 3 控制位	塊 2 控制位	塊 1 控制位	塊 0 控制位

把_b 位取反之後成爲下表

	bit 7	6	5	4	3	2	1	0
位元組 6	0	0	0	0	0	0	0	0
位元組 7	0	0	0	0	1	0	0	0
位元組 8	1	0	0	0	0	0	0	0
所屬塊	塊 3 控制位	塊 2 控制位	塊 1 控制位	塊 0 控制位	塊 3 控制位	塊 2 控制位	塊 1 控制位	塊 0 控制位

注意表中字體的顏色，其中：紅色字體代表的是 1 控制位；藍色字體是 2 控制位元；黑色字體是 3 控制位元。

注意：最關鍵的一點，上面表中，字體顏色相同的兩組值，必須一樣！例如：位元組 6 的高 4 位元(藍色字體 0 0 0 0)等於位元組 8 的低 4 位元(藍色字體 0 0 0 0)；位元組 7 的高 4 位元(紅色字體 0 0 0 0)等於位元組 6 的低 4 位元(紅色字體 0 0 0 0)；位元組 8 的高 4 位元(黑色字體 1 0 0 0)等於位元組 7 的低 4 位元(黑色字體 1 0 0 0)；

只有當上面表中字體顏色相同的兩組值是相同的情況下，這樣的一組控制值（例如這裏的 FF 07 80）才是有效的。無效的控制值將會產生無法預知的效果，例如：導致相應磁區的資料塊不能讀寫等。所以當想要自己重新設定控制區的值的時候，一定要小心計算該值。

4、把上面表中的值，和表 2 進行比對，可以得出：

塊 0 控制碼 = C10 C20 C30 = 0 0 0
塊 1 控制碼 = C11 C21 C31 = 0 0 0
塊 2 控制碼 = C12 C22 C32 = 0 0 0
塊 3 控制碼 = C13 C23 C33 = 0 0 1

5、把得到的塊 0、1、2 的控制碼和表 3 進行比對，可以知道到，塊 0、1、2 的控制碼都是 0 0 0，他們的控制方式都是，在驗證密碼 A 或者密碼 B 之後，可以對資料區進行讀、寫、加值、減值等操作。

6、把得到的塊 3 的控制碼和表 4 進行比對，得到其控制方式爲：對密碼 A 沒有讀取許可權（讀的時候返回全 0），當驗證了密碼 A 或者密碼 B 之後，可以讀取控制區、和密碼 B，可以寫密碼 A、控制區、密碼 B。

下面舉例說明如何修改控制區的控制碼。

1、確定塊 0、1、2 的訪問許可權。

這裏爲了簡單起見，把對塊 0、1、2 的訪問許可權設置爲一樣，通過查找表 3，例如這裏把它們的訪問許可權都設定爲 1 1 0（即：在驗證了密碼 A 或者密碼 B 的情況下可以讀，只有在驗證了 B 密碼的情況下才可以寫）

2、確定塊 3 的訪問許可權。

同樣類似上面，通過查找表 4，例如這裏把它的訪問許可權設定爲 0 1 1（即：在驗證了密碼 B 的情況下才可以修改（寫）密碼 A、控制區、密碼 B

3、把上面的控制碼填入到表 2 的相應位置，如下：

	bit 7	6	5	4	3	2	1	0
位元組 6	–	–	–	–	–	–	–	–
位元組 7	0	1	1	1	–	–	–	–
位元組 8	1	0	0	0	1	1	1	1
所屬塊	塊 3 控制位	塊 2 控制位	塊 1 控制位	塊 0 控制位	塊 3 控制位	塊 2 控制位	塊 1 控制位	塊 0 控制位

拷貝資料到相同顏色的位置

	bit 7	6	5	4	3	2	1	0
位元組 6	1	1	1	1	0	1	1	1
位元組 7	0	1	1	1	1	0	0	0
位元組 8	1	0	0	0	1	1	1	1
所屬塊	塊 3 控制位	塊 2 控制位	塊 1 控制位	塊 0 控制位	塊 3 控制位	塊 2 控制位	塊 1 控制位	塊 0 控制位

把_b 位的值取反

	bit 7	6	5	4	3	2	1	0
位元組 6	0	0	0	0	1	0	0	0
位元組 7	0	1	1	1	0	1	1	1
位元組 8	1	0	0	0	1	1	1	1
所屬塊	塊 3 控制位	塊 2 控制位	塊 1 控制位	塊 0 控制位	塊 3 控制位	塊 2 控制位	塊 1 控制位	塊 0 控制位

得到位元組 6 = 0 0 0 0 1 0 0 0 = 08

得到位元組 7 = 0 1 1 1 0 1 1 1 = 77

得到位元組 8 = 1 0 0 0 1 1 1 1 = 8F

位元組 9 是保留位元，目前沒有使用到，可以爲任意值，這裏還是用預設的 69。

所以可以得出控制區的 4 位元組值爲：08 77 8F 69。同時配合新的密碼，即可完成控制方式的修改。例如，把塊 3 的值修改爲：00 11 22 33 44 55 08 77 8F 69 AA

BB CC DD EE FF。這樣就修改了密碼 A 爲：00 11 22 33 44 55；密碼 B 爲：AA BB CC DD EE FF。

下面是一個錯誤的控制區 4 位元組值分析。

筆者曾經在不小心的情況下向塊 3 寫入了：00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 這樣的資料。當寫入這組資料之後，控制區的 4 位元組爲：66 77 88 99。這是一個錯誤的控制碼，當寫入該控制碼之後，雖然可以使用上面的密碼來通過密碼的驗證，但是，在讀寫資料的時候會被阻止，會返回錯誤。下面分析這個控制碼爲什麼不對。

1、對"66 77 88 99"值進行計算，該值定位于各區塊 3 的 6，7，8，9 四個位元組內，位元組 6=66，位元組 7=77，位元組 8=88，位元組 9=99(保留位的值，可以不管)。

2、轉換爲二進位碼

位元組 6 = 66 = 0 1 1 0 0 1 1 0

位元組 7 = 77 = 0 1 1 1 0 1 1 1

位元組 8 = 88 = 1 0 0 0 1 0 0 0

3、把上面的二進位資料填入到上面的表二中

	bit 7	6	5	4	3	2	1	0
位元組 6	0	1	1	0	0	1	1	0
位元組 7	0	1	1	1	0	1	1	1
位元組 8	1	0	0	0	1	0	0	0
所屬塊	塊 3 控制位	塊 2 控制位	塊 1 控制位	塊 0 控制位	塊 3 控制位	塊 2 控制位	塊 1 控制位	塊 0 控制位

把_b 位取反之後成爲下表

	bit 7	6	5	4	3	2	1	0
位元組 6	1	0	0	1	1	0	0	1
位元組 7	0	1	1	1	1	0	0	0
位元組 8	1	0	0	0	0	0	0	0
所屬塊	塊 3 控制位	塊 2 控制位	塊 1 控制位	塊 0 控制位	塊 3 控制位	塊 2 控制位	塊 1 控制位	塊 0 控制位

4、根據前面講述的，需要顏色相同的兩組值必須完全相同才是有效的控制碼。從上面的表中可以看到。位元組 6 的藍色字體（1 0 0 1）不等於位元組 8 的藍色字體（0 0 0 0），所以這是一組無效的控制碼。無效的控制到導致的直接結果就是，在密碼驗證成功的情況下，也不能進行讀寫操作。

所以在修改控制區的控制碼的時候需要特別小心！

Mifare 1K 卡片每個磁區的預設密碼和控制碼均爲：FF FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF

3-2　RF500 讀寫器控制命令

3-2.1　命令集

- Reset：　復位 RF500 模組
- Status：　讀硬體版本號
- Loadkey：　改變存放於 RF500 內部模組的密碼
- Request：　尋卡
- Anticoll：　防衝突
- Select：　選擇卡片
- Authentication　校驗密碼
- Read：　讀卡片資料
- Write：　寫卡片資料
- Halt：　釋放卡片
- Beep：　執行蜂鳴器鳴叫

3-2.2　編程協議

規定：通訊串列傳輸速率 115200；通訊格式是：1 位元起始位元，8 位元資料位元，1 位元結束位，無奇偶校驗位。資料最大長度爲 32 位元組。

通訊資料之間的最大時間間隔爲 0.5 秒，若超時則認爲通訊失敗。

控制單元發給 RF500 的資料格式是：

0xAA	Command	Len	Data[0]…Data[n]	Bcc

RF500 發給控制單元的資料格式是：

0xBB	Status	Len	Data[0]…Data[n]	Bcc

註：

AA/BB	資料包頭（1 位元組）
Command	命令碼（1 位元組）
Status	指令調用的結果（1 位元組）
Len	資料長度（1 位元組）
Data[i]	資料
Bcc	異或校驗碼

Bcc 等於前面所有資料（從 0xAA 或者 0xBB 開始到 Data[n]的所有資料的異或值。（異或運算：相同爲 0，不同爲 1）

當 Len = 0 的時候，沒有 Data 位。

3-2.3 命令格式說明

1、Reset，復位 RF500 模組

控制單元⇒RF500

Command: 0x4E

Len: 2

Data[0]: 重定時間（單位 10 毫秒）

Data[1]: 0

RF500⇒控制單元

Status: OP_OK：成功 其他：錯誤

Len: 0

例如：

發送：AA 4E 02 0A 00 EC；復位 10*10 毫秒

返回：BB 00 00 BB

2、Status，讀硬體版本號

控制單元⇒RF500

Command: 0x4F

Len:　　0

RF500⇒控制單元

Status:　　OP_OK：成功　其他：錯誤

Len:　　18

Data[0]

：　　硬體版本號

Data[17]

例如：

發送：AA 4F 00 E5

返回：BB 00 12 4B 49 4E 47 52 46 2D 52 43 35 30 30 2D 56 32 2E 30 36 DE

3、Load_key，把密碼裝載的 RF500 中，次動作並不驗證密碼，只是把密碼裝載到 RF500 的 ARM 中

控制單元⇒RF500

Command:　　0x4C

Len:　　18

Data[0]:　　模式

Data[1]:　　磁區號

Data[2]:　　_TKey[0]

：　　：　傳輸密碼

Data[9]:　　_Tkey[7]

Data[10]:　　_Nkey[0]

：　　：　用戶卡密碼（KeyA 或者 KeyB）

Data[17]:　　_Nkey[7]

RF500⇒控制單元

Status:　　OP_OK：成功　其他：錯誤

Len:　　0

註：

模式：取值含義：

0–用KEYSET0 和 KEYA 驗證

1–用KEYSET1 和 KEYA 驗證

2–用KEYSET2 和 KEYA 驗證

4–用KEYSET0 和 KEYB 驗證

5–用KEYSET1 和 KEYB 驗證

6–用KEYSET2 和 KEYB 驗證

磁區號：取值範圍：0-15。

_TKey[0]-_TKey[7]：傳輸密碼，現已不用，可以爲任意值。

_Nkey[0]-_Nkey[7]：卡片密碼，即 KeyA 或者 KeyB，前 6 位有效，後 2 位可以爲任意值。

此操作將密碼下載到 RF500 中，並不與卡片通訊。傳輸密碼和用戶卡密碼爲 6 位元組，但通訊時按 8 位元組發送，最後兩個位元組爲任意值。傳輸密碼是爲相容老模組所用，現已不用，可以爲任意值。

例如：

發送：aa 4c 12 00 01 14 8a c5 e2 28 28 00 00 ff ff ff ff ff ff 40 f6 fa

//模式 0、磁區號 1，密碼 ff ff ff ff ff ff

返回：bb 00 00 bb

4、Request，尋卡請求，返回卡片類型

控制單元⇒RF500

Command： 0x41

Len： 1

Data[0]： 模式

RF500⇒控制單元

Status： OP_OK，OP_NOTAG，OP_BITCNT

Len： 2

Data[0]：　　TagType（低位元組）

Data[1]：　　TagType（高位元組）

註：

模式：

0：IDLE 模式，只有處於 IDLE 狀態的卡片才回應讀寫器的命令。

1：ALL 模式，處在 IDLE 狀態和 HALT 狀態的卡片都回應讀寫器的命令。

Tagtype：　返回卡片類型

Mifare std. 1k：　0x0004

UltraLight：　　0x0044

FM005：　　0x0005

Mifare std. 4k：　0x0002

SHC1122：　　0x3300

在重新選擇卡片時必須執行 request 操作。

例如：

發送：aa 41 01 01 eb

返回：bb 00 02 04 00 bd

5、Anticoll

控制單元⇒讀寫模組

Command:　　0x42

Len:　　5

Data[0]:　　系列號位元數，標準值爲 0（不考慮系列號）

Data[1]:

:　　卡片系列號

Data[4]:

讀寫模組⇒控制單元

Status:　　OP_OK, OP_NOTAG, OP_BITCNT, OP_SERNR

Len:　　4

Data[0]:

: 卡片系列號

Data[3]:

註：

此操作必須緊隨在 request 操作後執行。如果被選的卡片的系列號已知，可以不用執行此操作。

例如：

發送：aa 42 05 00 00 00 00 00 ed

返回：bb 00 04 82 8a 8d 5d 67 //返回 4 位元組卡號

6、Select，選擇卡片

控制單元⇒讀寫模組

Command: 0x43

Len: 4

Data[0]:

: 卡片序列號

Data[3]:

讀寫模組⇒控制單元

Status: OP_OK, OP_NOTAG, OP_BITCNT, OP_PARTY, CRC

Len: 1

Data[0]: 卡片容量值：0x08 或 0x88

註：

Data[0]-Data[3]：卡片序列號，由 **Anticoll** 命令返回

例如：

發送：aa 43 04 82 8a 8d 5d 35

返回：bb 00 01 08 b2

7、Authentication，校驗磁區密碼

控制單元⇒讀寫模組

Command:　0x44

Len:　2

Data[0]:　模式

Data[1]:　磁區號

讀寫模組⇒控制單元

Status:　OP_OK, OP_NOTAG, OP_AUTH, OP_PARITY,BITCNT

Len:　0

註：

模式：取值含義：

0–用KEYSET0 和 KEYA 驗證

1–用KEYSET1 和 KEYA 驗證

2–用KEYSET2 和 KEYA 驗證

4–用KEYSET0 和 KEYB 驗證

5–用KEYSET1 和 KEYB 驗證

6–用KEYSET2 和 KEYB 驗證

這裏的模式應該和Load_Key中的一致。

磁區號：取值範圍：0-15。

如果讀寫模組中的密碼與卡片中的密碼相匹配，則可以在接下來進行讀、寫等操作。

例如：

發送：aa 44 02 00 01 ed //校驗 1 磁區密碼

返回：bb 00 00 bb

8、Read，讀塊操作

控制單元⇒讀寫模組

Command:　0x46

Len: 1

Data[0]: 塊地址（0～63）

讀寫模組⇒控制單元

Status: OP_OK,OP_NOTAG,OP_NOTAUTH, OP_READ,
OP_PARITY,OP_CRC,OP_BYTECNT

Len: 16

Data[0]: 資料塊的第一位元組

:

Data[15]: 資料塊的最後一個位元組

註冊：

地址塊：範圍（0-63），磁區 0 的塊 0 = 0，磁區 15 的塊 3 = 63

例如：

發送：aa 46 01 04 e9 //讀 4 塊即 1 磁區 0 塊的資料

返回：BB 00 10 6E DE 22 7C EE 08 04 00 62 63 64 65 66 67 68 69 A7
//返回 16 位元組資料

9、Write，寫塊操作

控制單元⇒讀寫模組

Command: 0x47

Len: 17

Data[0]: 要寫入資料的塊位址（1～63）

Data[1]: 要寫入卡片中的第一個資料

:

Data[16]: 要寫入卡片中的最後一個資料

讀寫模組⇒控制單元

Status: OP_OK,OP_NOTAG,OP_BITCNT
OP_NOTAUTH,OP_WRITE,OP_CODE

Len:　　0

註：

由於第 0 塊已經由廠商固化卡片 ID，所以寫入資料塊範圍（1-63）。磁區 0 的塊 1 = 1，磁區 15 的塊 3 = 63

例如：

發送：aa 47 11 04 00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff f8

//向塊 4（1 磁區塊 0）寫入 16 位元組資料

返回：bb 00 00 bb

10、Halt，將操作後的卡片置於 halt 模式

控制單元⇒讀寫模組

Command:　　0x45

Len:　　0

讀寫模組⇒控制單元

Status:　　OK,CODE

Len:　　0

註：

將操作後的卡片置於 halt 模式。如果又要對卡片操作，必須重新執行 request 操作。

例如：

發送：aa 45 00 ef

返回：bb 00 00 bb

11、Beep，執行蜂鳴器鳴叫

控制單元⇒讀寫模組

Command:　　0x36

Len:　　2

Data[0]:　　蜂鳴時長（單位 10 毫秒）

Data[1]:　　0

讀寫模組⇒控制單元

Status: OP_OK

Len: 0

例如：

發送：aa 36 02 1e 00 80 //蜂鳴 30*10ms

返回：bb 00 00 bb

3-3 典型的卡片操作步驟

1、Reset 讀寫器模組

↓

2、LoadKey，裝載卡片密碼到讀寫器中

↓

3、Request，尋卡操作

↓

4、Anticoll，獲取卡片的序列號

↓

5、Select，使用卡片序列號選中卡片

↓

6、Authentication，驗證卡片密碼

↓

7、根據需要，執行 Read 或者 Write 操作

↓

8、操作完成之後，把卡片置爲 Halt 模式

第四章 RFID 射頻 IC 卡讀寫器使用說明 Android 系統

4-1　硬體連接方式

筆者在本手冊中以 DMA-210U 開發平台作示範，說明在 DMA-210U 開發平台上使用 RFID 讀寫器。

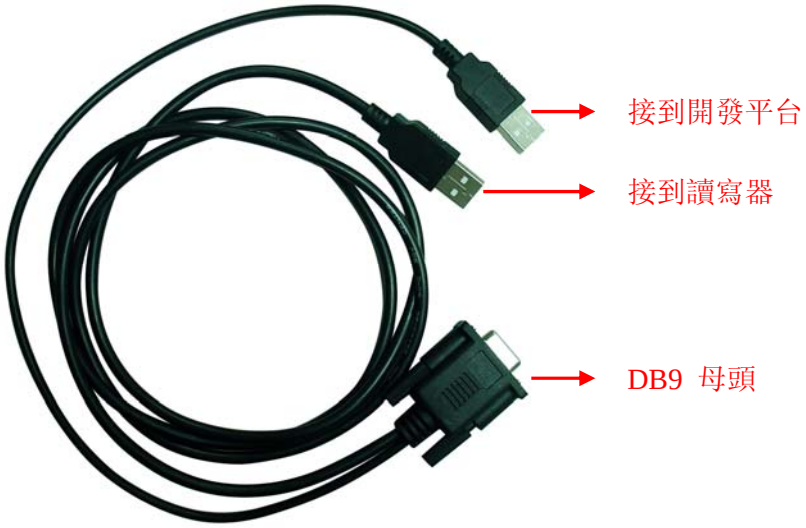
RF500 模組分爲 USB 介面和串列埠介面兩種資料連接方式。

說明： USB 介面和串列埠介面的兩種 RF500 模組外形結構是一模一樣的，區別它們的地方是看機身背面的型號貼紙。型號為 RF500S-U 為 USB 介面的；型號為 RF-500S 為串列埠介面的。

USB 介面，使用普通的 USB 連接線，把 RFID 和開發平台的 USB 介面連接即可。如下圖：


QC
PASSED
型 号: RF500S-U
批 号: K090807313
生产日期: 2009.08

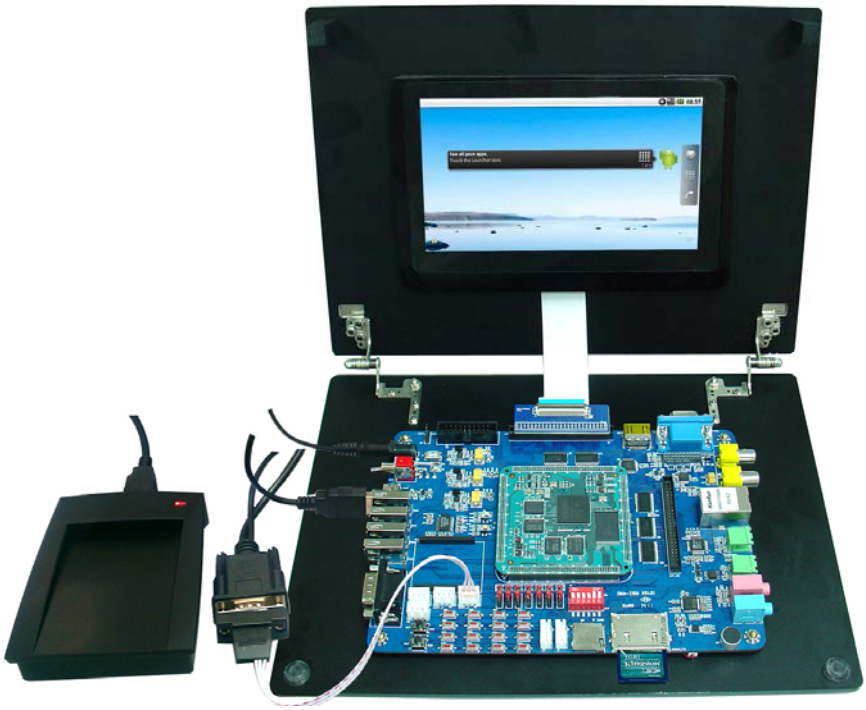
串列埠介面：其串列埠連接線如下圖所示：



RFID 的連接線

該 RFID 的連接線，是一條兩端都是 USB 公頭加一個 9PIN 串列埠的母頭。把串列埠接到 DMA-210U 的任意一個串列埠介面上，把離 DB9 頭比較近的那一端 USB 和開發平台的 USB 連接（用於取電），把離 DB9 頭比較遠的那一端 USB 和 RFID 模組連接（用於供電和傳送資料），切記這兩個 USB 埠不要接反了。

其連接圖如下所示：



本公司，其他平台的連接方式類似，注意 USB 的兩個埠不要接反即可。

4-2　RFID 射頻 IC 卡讀寫器使用方法

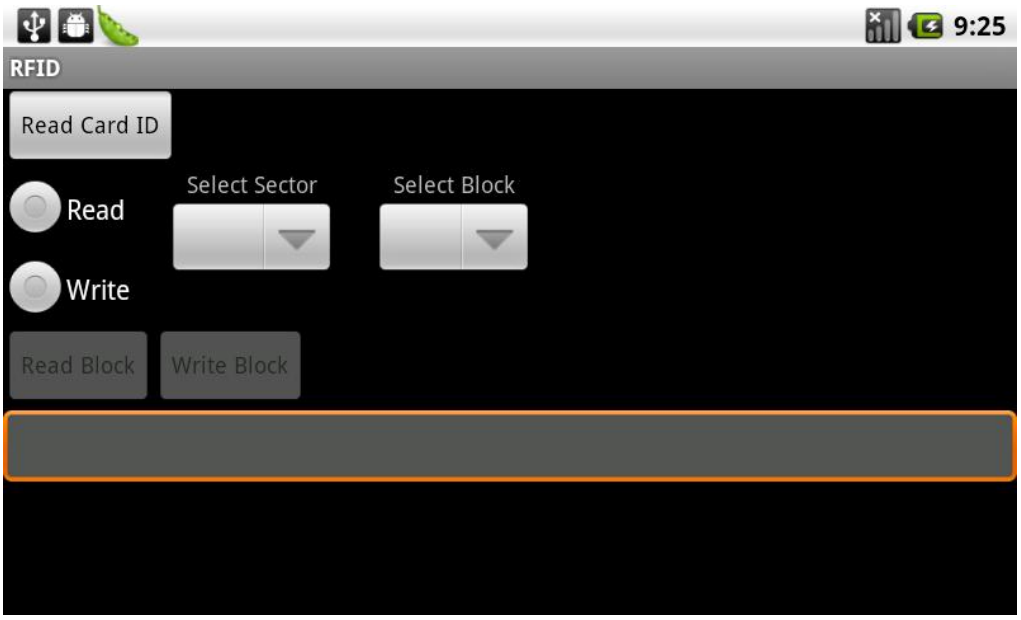
在依前述的說明，將 RFID 射頻 IC 卡讀寫器連接到本公式的嵌入式系統產品平台，就可以執行本公司提供的範例程式了。

以 DMA-210U 開發平台爲例。

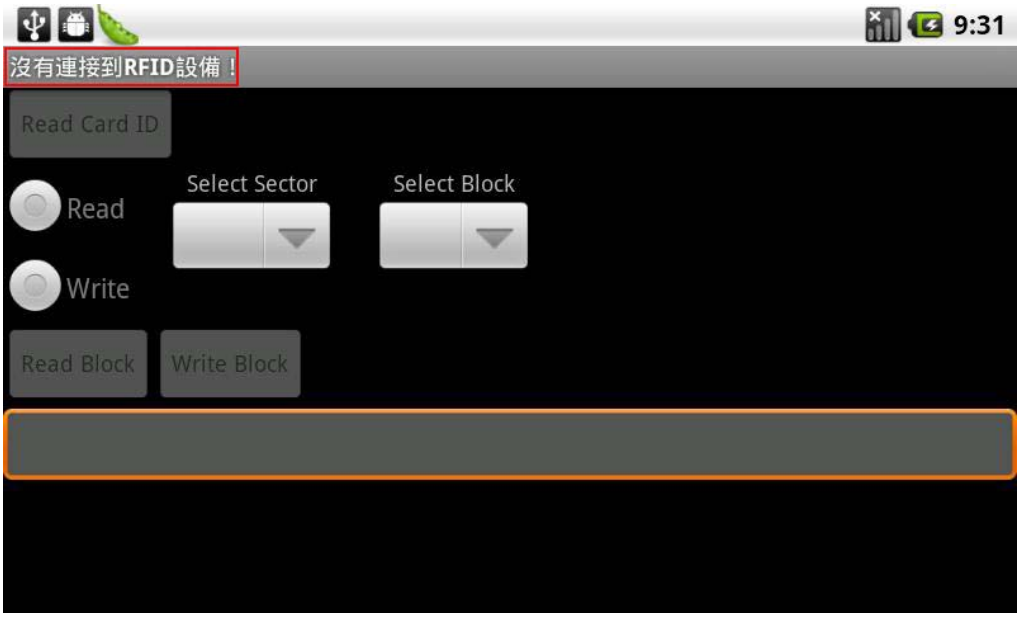
1、先安裝 RFID.apk 文件，然後點擊 

圖示，啓動程式。有如下介面：

程式啓動之後，會自動查找當前系統中是否有連接 RFID 硬體設備。成功連接到硬體設備之後，RFID 讀寫器會鳴叫，並且程式的相關按鈕變得可以點擊操作。如下圖：

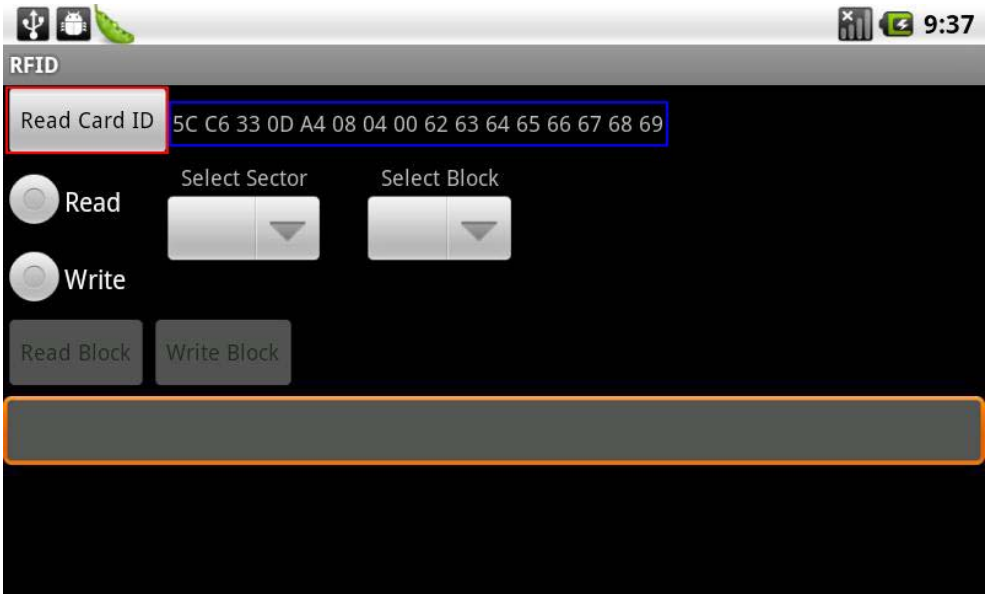


如果程式判斷沒有連接到硬體設備，此時會在程式的標題欄提示“沒有連接到 RFID 設備！”，如下圖：

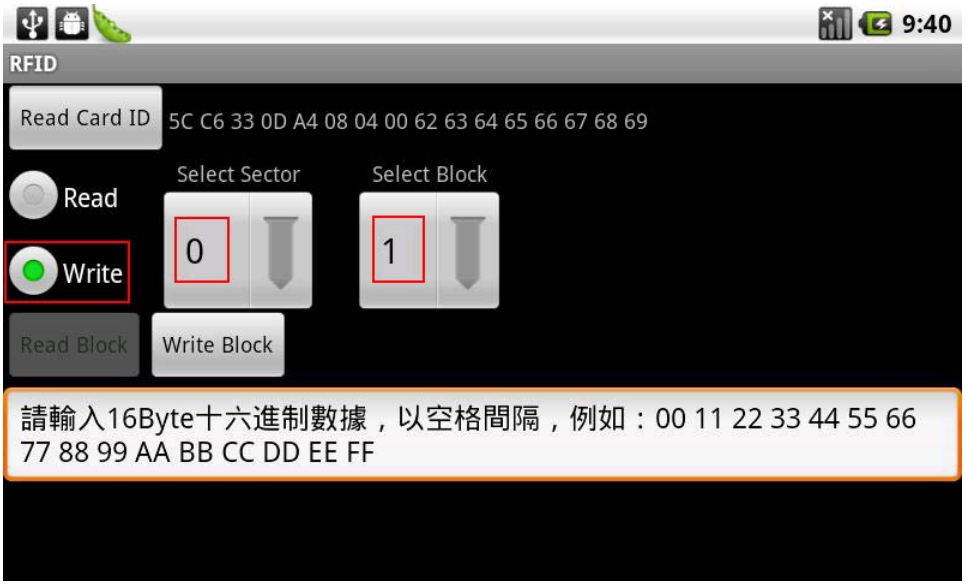


2、正常連接上 RFID 設備之後，請把 RFID 標籤，放入到讀寫器上面的凹槽中。如下圖：

3、點擊“Read Card ID”按鈕，可以讀取出卡片的唯一序列 ID，如下圖：



4、寫卡片測試，點擊“Write”單選按鈕，選擇爲“寫”卡片操作模式，此時在右側的 Sector 和 Block 的下拉清單中會有數位工選擇，如下圖所示：



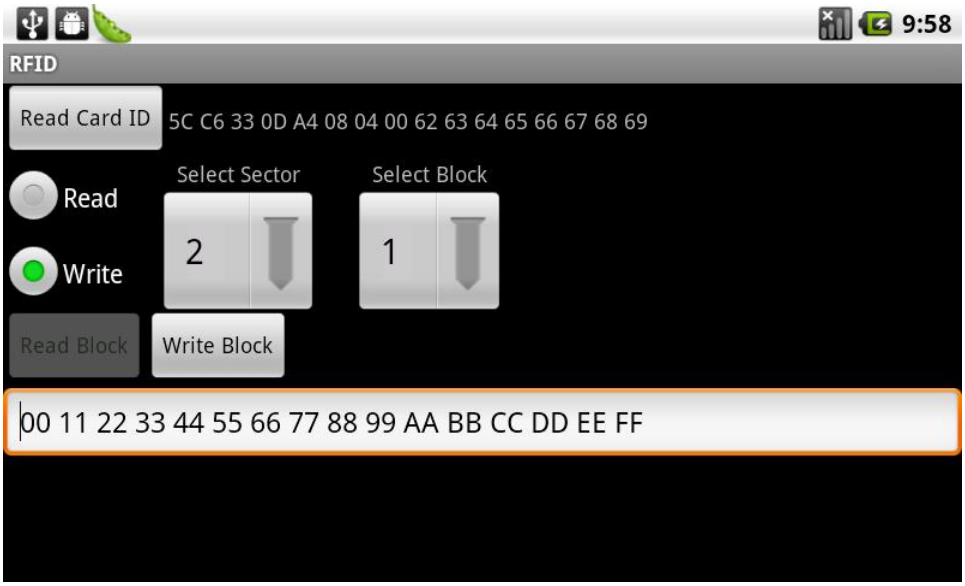
該程式是針對Mifare 1K卡片設計，該卡片的規格爲：一共分16個磁區（Sector），每個磁區有4個塊（Block），每個塊可以存儲16Byte的資料。其中第0磁區的第0塊是廠商固化的全球唯一ID，不可寫。每個磁區的第3塊，是這個磁區的密碼和存儲控制位元。

針對該卡片的這些規格，所以程式在設計的時候，在Sector的選擇上可以選擇範圍0-15，塊的選擇範圍0-2，當選擇的磁區是第0個磁區的時候，塊不能選擇0塊，因爲這個塊是已經被廠商固化的。另外，針對所有的磁區，都不允許讀寫塊3，因爲塊3存放了當前磁區的密碼和控制碼，任意的修改這個塊的資料之後，極易導致該磁區被鎖死（不能讀取資料也不能寫入資料），所以該程式不支援修改塊3的資料。

該程式使用的都是Mifare 1K卡片的預設密碼進行密碼驗證操作，所以，如果是被其他程式加密過的Mifare 1K卡片，在該程式中，可能不能正常的讀寫。

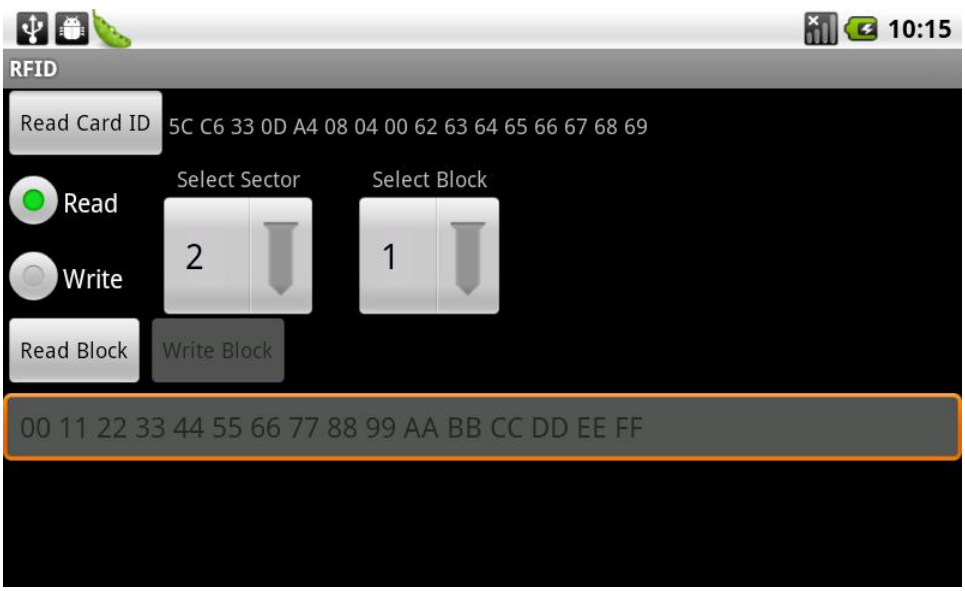
下面繼續進行寫卡片的操作驗證。選擇我們一個希望寫入的磁區號和塊號，例如：磁區2，塊1。

然後在下面的輸入框中，輸入要寫入的資料，注意這裏程式爲了便於查看的原因，要求，寫入16進制的16Byte資料，每個資料之間使用空格間隔。例如我們使用軟體提示使用的資料，注意把前面的中文相關字元消去。如下圖：



點擊“Write Block”按鈕，等待聽到蜂鳴器鳴叫，表示寫入資料成功。

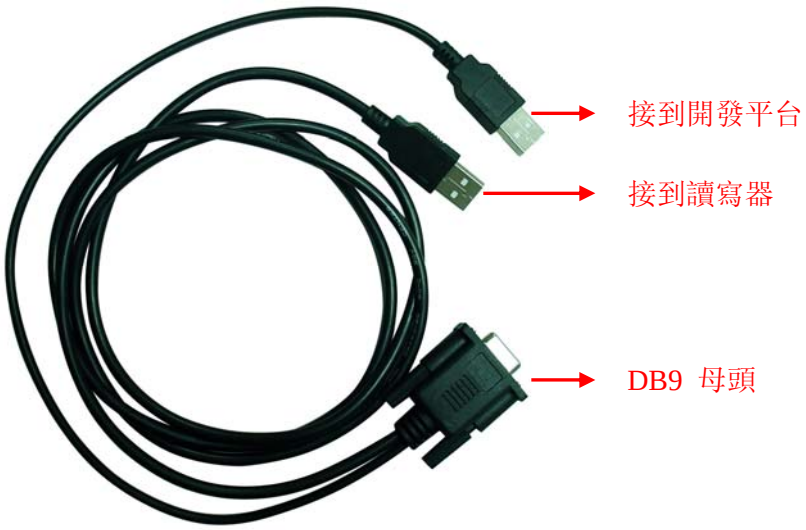
5、讀卡片測試，點擊“Read”單選按鈕，切換到讀卡片的模式，保持選擇剛才選擇的磁區 2 和塊 1，然後點擊“Read Block”按鈕。就會讀取出剛才我們寫入的資料。如下圖：



第五章 RFID 射頻 IC 卡讀寫器使用說明 WinCE 系統

5-1　硬體連接方式

筆者在本手冊中以 DMA-6410XP 教學平台作示範，說明如何連接 RFID 射頻 IC 卡讀寫器與 DMA-6410XP 教學平台，其連接方式如圖 5-1 並說明如下：



RFID 的連接線

該 RFID 的連接線，是一條兩端都是 USB 公頭加一個 9PIN 串列埠的母頭。把串列埠接到 XP 的任意一個串列埠介面上，把所在線比較短的那個 USB 埠接到板子的 USB 介面上，把長線的 USB 和 RFID 模組連接。切記這兩個 USB 埠不要接反了。

圖 5-1　RFID 射頻 IC 卡讀寫器安裝方式

本公司，其它平台的連接方式類似，注意 USB 的兩個埠不要接反即可。

5-2　RFID 射頻 IC 卡讀寫器使用方法

5-2.1　WinCE 範例

在依前述的說明，將 RFID 射頻 IC 卡讀寫器連接到本公式的嵌入式系統產品平台，就可以執行本公司提供的範例程式了。

以 DMA-6410XP 教學平台爲例，其 RFID 射頻 IC 卡讀寫器的範例程式需要以下四個項目：

1. RFID 目錄：在該目錄下存放 BMP 類型的圖片，這是範例程式添加使用者資料時所需要的圖檔。
2. RFID_COMX.exe：範例程式的執行檔，注意這裏提供了 RFID_COM1.exe、RFID_COM2.exe、RFID_COM3.exe、RFID_COM4.exe，四個執行檔，原因是考慮到針對不用的平台，可能連接的不同的串列埠，所以提供 4 個不同串列埠上的測試文件。請在執行的

時候選擇相應的文件執行。

3. RFID_43.dll：範例程式在 4.3” LCD 上執行時所需要的資源檔。
4. RFID_70.dll：範例程式在 7 ” LCD 上執行時所需要的資源檔。

在 DMA-6410XP 上，如果把串列埠接到 UART2 上，那麼請執行 RFID_COM2.exe 程式。

請將這四個項目複製到 SD 卡，插入開發平台後執行 RFID_COM2.exe 啓動範例程式，此時會開啓如圖 5-2 的視窗。

圖 5-2

5-3 範例實驗

本公司提供的 RFID 射頻 IC 卡讀寫器範例程式分成「卡片初始化」及「資料設置」兩個部份。以下將說明該範例程式的使用方法。

當開啓範例程式之後，首先會出現圖 5-2 的視窗，在視窗上的兩個按鈕分別提供了「卡片初始化」及「卡片資料設置」的功能。此時將 RFID 射頻 IC 卡放置在讀寫器上方，如果該卡片尚未初始化，則在視窗的「此卡狀態」會提示「此卡未初始化」，如圖 5-3 所示，且會有提示音告知使用者。請注意，RFID 射頻 IC 卡一定要先初始化才可以使用，筆者將在稍後說明初始化的步驟。

圖 5-3

如果該卡片已經初始化，但尚未設定資料，則在「此卡狀態」會顯示「此卡無效」的訊息，並在「此卡 ID」顯示該卡片的 ID 號，如圖 5-4 所示。筆者也將在本節稍後說明設置卡片資料的步驟。

圖 5-4

如果該卡片已經初始化，而且已經設定妥卡片內的資料時，則在「此卡狀態」會顯示「此卡有效」的訊息，並顯示該卡片的 ID 號與使用者的名稱(用戶名)，如圖 5-5 所示。此外，範例程式也會以提示音表示卡片讀取成功。

圖 5-5

1. 卡片初始化

RFID 射頻 IC 卡一定要先初始化才可以使用，因此若讀者將卡片置於讀寫器上時，範例程式出現該卡片未初始化的提示(圖 5-3)時，請讀者務必點擊範例程式的「初始化卡」鈕對該卡片進行初始化。

當點擊「初始化卡」鈕後，範例程式會開啓如圖 5-6 的視窗，在這個功能中提供了三個子功能，分別如下：

1. Read：自 RFID 標籤讀取 ID 號碼。
2. Write：將 ID 號碼寫入 RFID 標籤中。
3. Clean：清除 RFID 標籤內儲存的資訊。

以下將依次說明這三個子功能。

圖 5-6

I. 勾選「Read」並將卡片式的 RFID 標籤放置在讀寫器上方，如果該卡片已經被初始化過，則如圖 5-7 所示將會顯示該卡片的 ID 號碼，並且以蜂鳴聲提示；若該卡片尚未被初始化，則範例程式不會有任何動作反應。

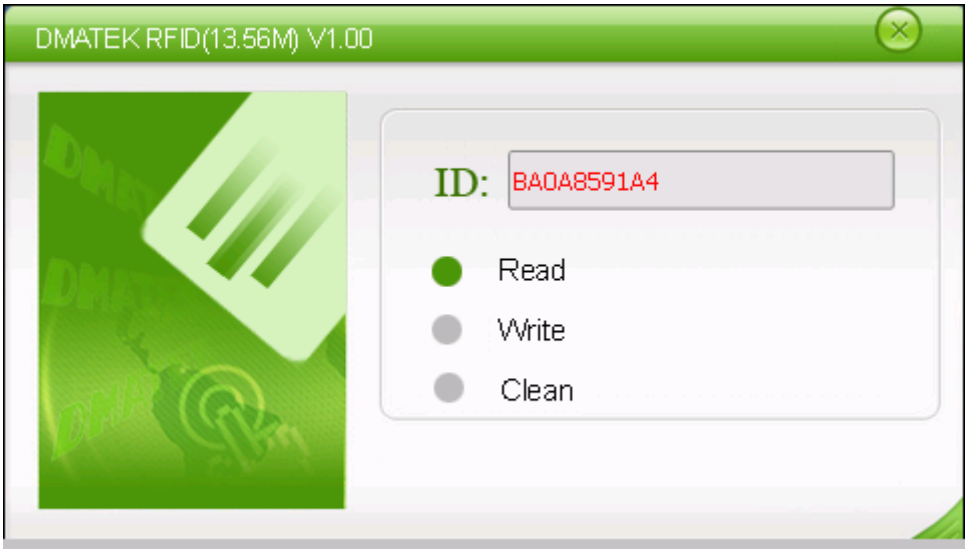


圖 5-7

II. 勾選「Write」並將尚未初始化的卡片式 RFID 標籤放置在讀寫器上方，則此時範例程式將會對卡片進行初始化並寫入一個 ID 號碼，如圖 5-8 所示，並以蜂鳴聲提示使用者成功完成初始化。當執行完這個動作之後，該卡片的狀態將會被標示爲「已初始化」。

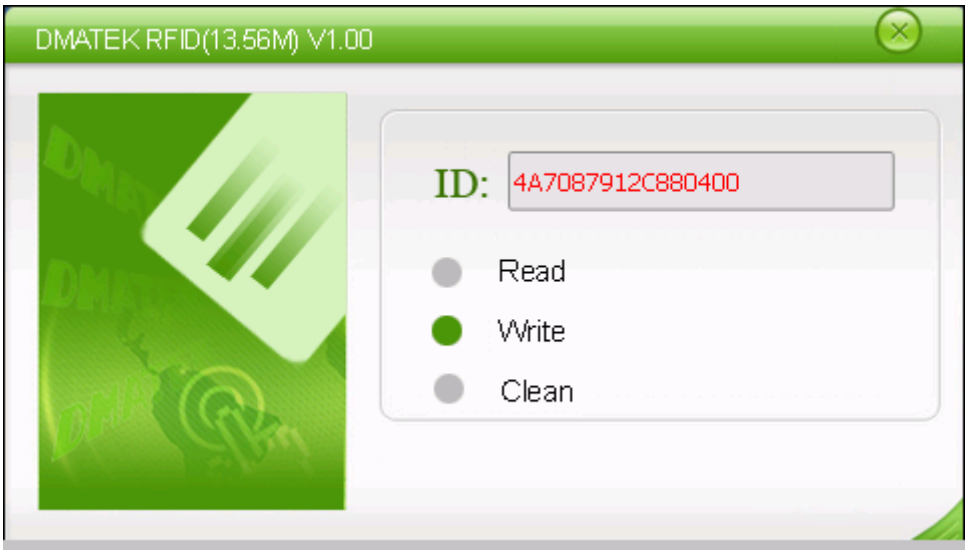


圖 5-8

III. 勾選「Clean」並將卡片式 RFID 標籤放置在讀寫器上方，讀寫器會將卡片內的資訊清除，並在清除成功時以蜂鳴聲提示使用者。當執行這個動作之後，該卡片的狀態將會標示爲「未初始化」。

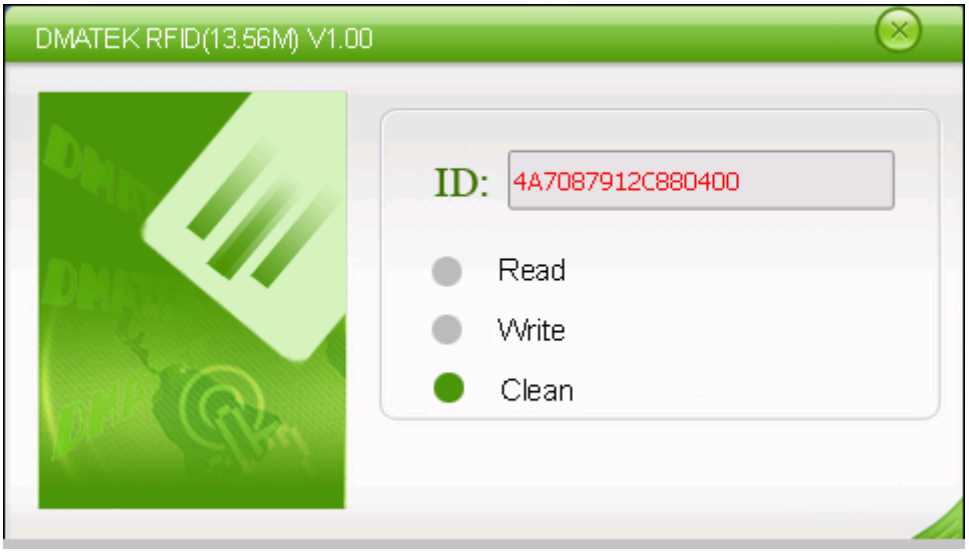


圖 5-9

2. 卡片資料設置

在圖 5-2 的視窗中按下「資料設置」鈕後，此時會開啓如圖 5-10 的程式畫面，此時可以對卡片進行使用者(用戶)資料的添加或刪除，預設的選項爲「添加用戶」:

圖 5-10

1.　添加使用者：

將卡片式 RFID 標籤放置到讀寫器上，在圖 5-10 的程式畫面上點擊「添加用戶」，然後輸入姓名、性別及照片等資料後再點擊「確認」鈕，此時會顯示「添加完畢」的訊息，並有提示音告知使用者。如圖 5-11 所示：

圖 5-11

2. 刪除用戶：

在圖 5-10 的程式畫面上點擊「刪除用戶」，然後將卡片式 RFID 標籤放置到讀寫器上並按下「確認」鈕，當卡片內的資料成功刪除後，程式畫面上會顯示「刪除完畢」的訊息，並有提示音告知使用者。如圖 5-12 所示：

圖 5-12

5-4　常見問題

Q： 將 RFID 射頻 IC 卡讀寫器連接到開發平台，執行 DEMO 程式爲什麼會出現「無法開啓串列埠」的錯誤？

A：這是說明目前該平台的系統中沒有這個串列埠或者說是該串列埠已經被占用。

Q： 執行 DEMO 程式時，可以開啓串列埠，但是無法出現 DEMO 程式的畫面？

A： 由於 WinCE 沒有「當前目錄」的概念，因此當 DEMO 程式從 RFCE 目錄抓取圖片資源時是採用「絕對路徑」的概念。所以若 DEMO 程式無法正確描繪它的畫面時，表示可能是路徑沒有對應。請確認將 5-2.1 節所說明的四個項目複製到 SD 卡內，由 SD 卡執行。另外，以 DMA-6410XP 教學平台爲例，在 WinCE 下讀取 SD 卡時的 label name 應該爲「SDMMC」，如圖 5-13 所示。

圖 5-13

第六章　RFID 應用程式分析 Android 系統

本章將簡要介紹Android系統上的RFID程式的編寫流程。

RF500模組分爲USB介面和串列埠介面。其中USB介面方式的模組，其內部使用了一顆PL2303 USB轉串列埠晶片。所以，雖然在硬體的連接方式上有所不一樣，但是，在開發程式的時候，都是使用讀寫串列埠的操作方式即可。

在進行程式開發前，請先參考第三章的相關內容，瞭解RF500讀寫器的命令碼和Mifare 1K卡片的規格。

本程式是針對Mifare 1K卡片設計，使用卡片預設的驗證密碼FF FF FF FF FF FF，只允許讀寫每個磁區的塊0、1、2，不允許讀寫塊3。

程式主要分3大部分的功能：程式啓動之後的自動尋找RFID模組硬體的功能。和找到模組硬體之後的讀寫卡片的功能。

6-1 如何自動尋找 RFID 模組硬體

其原理是：挨個遍曆串列埠，向串列埠發送模組的 Reset 命令，如果該串列埠上有這個模組，那麼會在這個串列埠中接收到相應的返回資料。那麼就可以確定該串列埠上連接了這個模組。否則，如果，所有的串列埠遍曆完畢之後還是沒有找到模組，那麼視爲當前系統中沒有連接硬體模組。

1、onCreate 函數

程式在啓動之後，會先執行到 onCreate 函數中，在該函數中，開始做了一些綁定相關控制項，和給一些按鈕添加點擊事件等。這都是一些普通的操作，在該函數的最後，有使用 StartConnect 函數來開始進行模組的自動檢測。

2、StartConnect 函數

該函數的作用就是依次打開串列埠，如果打開串列埠成功，那麼就啓動 DetectThread 執行緒，用於接收串列埠資料，並向該串列埠發送 RFID 模組 Reset 的命令。然後返回 true。如果所有的串列埠都遍曆完，那麼返回 false，表示當前系統沒有連接硬體模組。

3、DetectThread 執行緒

該執行緒用於接收串列埠資料，並分析其中接收到的資料，如果有發現收到 RFID 模組的 Reset 命令的返回值，並且該返回值是表示成功的，那麼該執行緒會把一個名爲 SharedData.ResetOK 表示置 True。該執行緒會在 1 秒鐘之後，自動退出，並且在退出的時候發送一個自定義的 *MESSAGE_CONNECT_DEVICE_OVER* 消息給主執行緒。

4、在主執行緒中處理 *MESSAGE_CONNECT_DEVICE_OVER* 消息

在主執行緒中當接收到 *MESSAGE_CONNECT_DEVICE_OVER* 消息之後，先會判斷 SharedData.ResetOK 標誌位元，是否爲 true；如果爲 True，那麼說明有找到 RFID 硬體，停止串列埠的遍曆。等待用戶的選擇來執行讀寫功能。如果表示位爲 False，那麼就要繼續讀取 StartConnect 來遍曆下一個串列埠，直到要麼 Reset 的標誌位元爲 True 或者 StartConnect 函數返回 False，表示，當前系統沒有連接硬體。

6-2　讀寫 RFID 卡片

因爲讀、寫 RFID 卡片在操作上極其類似，所以放到一起來說明。

當等待自動檢測到 RFID 模組之後，會開啓一個 ReceiveThread 執行緒。該執行緒用於接收串列埠資料，並解析其中的資料。

1、讀卡片資料

函數 m_Button_ReadID_Listener 和 m_Button_ReadBlock_Listener 均是讀卡片塊的資料，不同之處在於 m_Button_ReadID_Listener 是固定讀取卡片的 0 磁區的 0 塊的資料，這個資料是卡片的唯一 ID。m_Button_ReadID_Listener 是根據用戶的選擇來讀取指定塊的資料。

它們的操作流程都一樣。依次讀取 ResetRFID（Reset RFID 模組）、LoadKey（裝載密碼到讀寫器）、Request（尋卡操作）、Anticoll（獲取卡片序列號）、Select（使用序列號選中卡片）、Authentication（驗證密碼）、Read（讀指定塊的資料）這些函數，最後打讀取到的資料經處理之後，轉換爲字元顯示出來。

2、寫卡片資料

函數 m_Button_ReadBlock_Listener 展示了寫卡片的操作流程。同樣該函數類似 Read 一樣，會依次讀取 ResetRFID（Reset RFID 模組）、LoadKey（裝載密碼到讀寫器）、Request（尋卡操作）、Anticoll（獲取卡片序列號）、Select（使用序列號選中卡片）、Authentication（驗證密碼）、Write（寫指定塊的資料）這些函數。

3、其他幾個函數的說明

關於在讀寫中讀取的 ResetRFID（Reset RFID 模組）、LoadKey（裝載密碼到讀寫器）、Request（尋卡操作）、Anticoll（獲取卡片序列號）、Select（使用序列號選中卡片）、Authentication（驗證密碼）、Read（讀指定塊的資料）、Write（寫指定塊的資料）這些函數。它們的實現就是對相應命令碼的封裝，具體請參考第三章相關內容，結合 Source 自行理解。

第七章　RFID 應用程式分析 WinCE 系統

本章說明的 RFID 應用程式是採用 Visual Studio 2005 開發的，讀者若需要變更或重新編譯此應用程式請注意需要安裝開發平台的 WinCE SDK。

7-1　WinCE 實驗

7-1.1　WinCE 函數功能介紹

（1）　HANDLE CE_rf_init(HWND hWnd, long baud)

功能：初始化串列埠

參數：

long baud：通訊串列傳輸速率(115200)

返回：成功則返回設備描述符(≥0)

例：HANDLE icdev;

icdev= CE_rf_init (NULL,115200);/*串列傳輸速率 115200*/

（2）　BOOL CE_rf_get_status(HANDLE icdev, unsigned char *_Status)

功能：取得讀寫器的版本號

參數：unsigned char*_Status：保留暫無使用

返回：成功則返回 TRUE，否則返回 FALSE

例：BOOL st;

unsigned char status[32];

st = CE_rf_get_status(status);

（3） BOOL CE_rf_reset(HANDLE icdev, unsigned int _Msec)

功能：射頻頭重定

參數：HANDLE icdev

unsigned__int16_Msec：重定時間，0～500 毫秒有效

返回：成功則返回 TRUE，否則返回 FALSE

例：st= CE_rf_reset (icdev,60);

（4） BOOL CE_rf_select(HANDLE icdev,unsigned long _Snr,unsigned char *_Size)

功能：從多個卡中選取一個給定序列號的卡

參數：HANDLE icdev：設備描述符

unsigned long_Snr：卡序列號

unsigned char*_Size：保留暫無使用

返回：成功則返回 TRUE，否則返回 FALSE

例：項 BOOL st;

snr=239474;

st = CE_rf_select(icdev , snr, size);

（5） BOOL CE_rf_load_key(HANDLE icdev,unsigned char _Mode,unsigned char _SecNr,unsigned char *_NKey)

功能：向讀寫器的 RAM 裝入 16 進制密碼

參數：HANDLE icdev：rf_init()返回的設備描述符

unsigned char_Mode：驗證密碼的方式

取值如下：

0–用 KEYSET0 和 KEYA 驗證

1–用 KEYSET1 和 KEYA 驗證

2–用 KEYSET2 和 KEYA 驗證

4–用 KEYSET0 和 KEYB 驗證

5–用 KEYSET1 和 KEYB 驗證

6–用 KEYSET2 和 KEYB 驗證

unsigned char_SecNr：要驗證密碼的磁區號(0～15)

返回：成功則返回 TRUE， 否則返回 FALSE。

例：//key A

unsigned char tk[]="a0a1a2a3a4a5"/*等同於{0xa0_.0xa5}*/

/*裝入 1 磁區的 A 密碼|0 套，*/

if((rf_load_key_Hex(icdev,0,1,tk))!=0)

{

printf("Load key error!");

CE_rf_exit();

}

註：*_Nkey 是與卡中的密碼相對應的密碼

（6） BOOL CE_rf_anticoll(HANDLE icdev,unsigned char _Bcnt,unsigned long *_Snr)

功能：防止卡衝突；返回卡的序列號

參數：HANDLE icdev：設備描述符

unsigned char_Bcnt：預選卡所用的位數，標準值爲 0（不考慮系列號）

unsigned long*_Snr：返回的卡序列號地址

返回：成功則返回 0

例：BOOL st;

st=CE_rf_anticoll(0,snr);

註：request 指令之後應立即調用 anticoll，除非卡的序列號已知。

（7） BOOL CE_rf_authentication(HANDLE icdev,unsigned char _Mode,unsigned char _SecNr)

功能：驗證某一磁區密碼

unsigned char_Mode：驗證密碼的方式

取值如下：

0–用 KEYSET0 和 KEYA 驗證

1–用 KEYSET1 和 KEYA 驗證

2–用 KEYSET2 和 KEYA 驗證

4–用 KEYSET0 和 KEYB 驗證

5–用 KEYSET1 和 KEYB 驗證

6–用 KEYSET2 和 KEYB 驗證

unsigned char_SecNr：要驗證密碼的磁區號(0～15)

返回：成功則返回 0

例：BOOL st; st=CE_rf_authentication(icdev,0,0);

註：每張卡上有 A 密碼和 B 密碼，可根據實際需要確定是否使用 B 密碼，這由卡的存取控制位來決定。此外，讀寫器中可以存放三套密碼，可用 CE_rf_load_key()分別裝入，只有裝入後才能使用驗證密碼函數驗證

（8） BOOL CE_rf_beep(HANDLE icdev,unsigned short _Msec)

功能：蜂鳴

參數：unsigned int_Msec：蜂鳴時間的長短，單位是 10 毫秒

返回：成功則返回 TRUE，否則返回 FALSE

例：BOOL st;

st=CE_rf_beep(10);/*鳴叫 100 毫秒*/

（9） BOOL CE_rf_halt(HANDLE icdev)

功能：中止卡操作

參數：HANDLE icdev：設備描述符

返回：成功則返回 TRUE，否則返回 FALSE

例：BOOL st;

St = CE_rf_halt(icdev);

註：使用 CE_rf_card()函數時，如果模式選擇爲 0 則在對卡進行讀寫操作完畢後，必須執行 CE_rf_halt()，且只能當該卡離開並再次進入操作區域時，讀寫器才能夠再次操作它

（10）BOOL CE_rf_read(HANDLE icdev,unsigned char _Adr,unsigned char *_Data)

功能：讀數據，一次必須讀一個塊

參數：HANDLE icdev：設備描述符

unsigned char_Adr：塊地址(0～63)

unsigned char*_Data：讀出數據

返回：成功則返回 TRUE，否則返回 FALSE.

例：BOOL st;

static unsigned char data[16]

st = CE_rf_read(icdev,0,data);

（11）BOOL CE_rf_request(HANDLE icdev,unsigned char _Mode,unsigned __int16 *TagType)

功能：尋卡請求

參數：HANDLE icdev：設備描述符

unsigned char_Mode：尋卡模式，爲 0 或 1

Mode=0：IDLE 模式，一次只選一張卡

Mode=1：ALL 模式，一次可選多張卡

unsigned int*Tagtype：指向返回的卡類型值

返回：成功則返回 TRUE,否則返回 FALSE

例：BOOL st；

St = CE_rf_request(icdev,IDLE,tagtype);

註：對卡操作前先執行尋卡指令

（12）BOOL CE_rf_write(HANDLE icdev,unsigned char _Adr,unsigned char *_Data)

功能：寫資料，一次必須寫一個塊

參數：HANDLE icdev：設備描述符

unsigned char_Adr：塊地址(1～63)

unsigned char*_Data：寫入資料，塊長度爲 16 Byte

返回：成功則返回 TRUE，否則返回 FALSE

例：BOOL st；

static unsigned char data[16]

/*給 data 賦值*/

```
BOOL = CE_rf_write(icdev,1,data);/*寫入塊 1*/
```

7-1.2 WinCE 實驗程式

在 WinCE 系統的實驗程式區分成三個部份，分別爲 RFID、RFID_43 及 RFID-70 三個部份。其中 RFID 是實驗範例的主程式，而 RFID_43 與 RFID_70 則分別是對應於 4.3” LCD 及 7” LCD 所使用的資源檔。

1. 串列埠號指定

開啓 RFID 這個專案，在 SerCom.h 中找到 HANDLE CE_rf_init(HWND hWnd, log buad)的定義實作，然後再找到以下這個區段，其中的 L"COM2:"表示使用 COM 2 這個串列埠。

```
hLocal = CreateFile (L"COM2:", GENERIC_READ | GENERIC_WRITE,
                                    0, NULL, OPEN_EXISTING, 0, NULL);
```

2. 發送命令資訊模組程式碼：

```
DWORD WINAPI GetDevThreadFunc(LPVOID lpParam)
{
    TCHAR szwcsBuffer[17];
    unsigned char szCompare[] = "0123456789ABCDEF0123456789ABCDEF";

    CRFIDDlg* pDlg = (CRFIDDlg*)lpParam;
    unsigned __int16 TagType;
    unsigned long Snr;
    unsigned char Size;
    unsigned char key[7];
    unsigned char Sec = pDlg->m_devParameter.m_Sector;
    unsigned char szData[33];
    memset(szData,0,33);
```

```
    unsigned char temp[33];
    unsigned char szSave[33];
    BOOL bClean = FALSE;
    memset(temp, 0, 33);
    memset(szSave, 0, 33);
    memset(szwcsBuffer, 0, 17 * sizeof(TCHAR));
    for(int nKey = 0; nKey < 6; nKey++)
    {
        key[nKey] = 0xff;
    }
    WaitForSingleObject (g_hWriteEvent, INFINITE);
    while(TRUE)
    {
        Sleep(100);
        if(pDlg->current_page == PAGE_READ)
        {
        // 詳細實作請見程式碼。
        }
    }
    return 0
}
```

3.　讀設備返回資訊模組：

```
#define TEXTSIZE 256

DWORD WINAPI ReadThread (PVOID pArg) {
    HWND hWnd;
    DWORD cBytes, i;
    BYTE    szText_Rcv[TEXTSIZE], *pPtr;
```

```
    UINT nAllLen = 255;
    hWnd = (HWND)pArg;
    char szRst[256];
    WaitForSingleObject (g_hReadEvent, INFINITE);
    CRFIDDlg* pDlg = (CRFIDDlg*)pArg;
    RETAILMSG(DEBUG_MESSAGE, (TEXT("hou    readthread
++++++++++++++++++++\r\n")));
    while (TRUE)
    {
        pPtr = (BYTE *)szText_Rcv;

        for (i = 0; i < sizeof (szText)-sizeof(TCHAR); i++)
        {
            while (!ReadFile (pDlg->m_devParameter.m_icdev, pPtr, 1,
&cBytes, 0))
            {
                if (pDlg->m_devParameter.m_icdev ==
INVALID_HANDLE_VALUE)
                {
                    //MessageBox(NULL, L"", L"Open Serial Fail", MB_OK);
                    return 0;
                }
            }
            szDbg[i] = *pPtr;
            if(i == 0)
            {
                if(((BYTE)szDbg[i]) != ((BYTE)0xbb))
                {
                    break;
```

```
                    }
                }
                if(i == 1)
                {
                    if(((BYTE)szDbg[i]) != ((BYTE)0x0))
                    {
                        break;
                    }
                }
                if(i == 2)
                {
                    nAllLen = szDbg[i] + 4 - 1;
                }
                if(nAllLen == i)
                {
                    memset(&szDbg[(nAllLen + 1) * 2], 0, TEXTSIZE - ((nAllLen +
1) * 2));
                    mbstowcs(szText, (char*)szRst, (nAllLen + 1) * 2);

                    SetEvent(g_hMutexEvent);
                    break;
                }
            }

        }
        return 0;
}
```

4. 資源檔定義：

在 RFID_43 及 RFID_70 兩個專案中分別由 RFID_43.rc 及 RFID_70 兩個檔案來定義範例程式所使用的資源。以 RFID_43 的 RFID_43.rc 爲例，包括：

```
IDB_BG_READ_43              BITMAP                  "res\\ID2-bj.bmp"
IDB_BG_SET_43               BITMAP                  "res\\ID3.bmp"
IDB_BG_INITIZLIZE_43        BITMAP                  "res\\RFID.bmp"
IDB_CLOSE_UP_43             BITMAP                  "res\\RFID-cl2.bmp"
IDB_CLOSE_DOWN_43           BITMAP                  "res\\RFID-cl1 副本.bmp"
IDB_INITIZLIZE_UP_43        BITMAP                  "res\\ID3-jt1.bmp"
IDB_INITIZLIZE_DOWN_43      BITMAP                  "res\\3.bmp"
IDB_SET_UP_43               BITMAP                  "res\\ID3-jt1 副本.bmp"
IDB_SET_DOWN_43             BITMAP                  "res\\2.bmp"
IDB_LABBG_43                BITMAP                  "res\\IDbj.bmp"
IDB_INIT_ID_43              BITMAP                  "res\\init ID.bmp"
IDB_OK_UP_43                BITMAP                  "res\\ID3-qr1.bmp"
IDB_OK_DOWN_43              BITMAP                  "res\\212.bmp"
IDB_OK_INVALIDATION_43      BITMAP                  "res\\復件
ID3-qr1.bmp"
IDB_ADD_UP_43               BITMAP                  "res\\ID3-tj2.bmp"
IDB_ADD_DOWN_43             BITMAP                  "res\\ID3-tj1.bmp"
IDB_DELETE_UP_43            BITMAP                  "res\\ID3-sc2.bmp"
IDB_DELETE_DOWN_43          BITMAP                  "res\\ID3-sc1.bmp"
IDB_LAB_SET_43              BITMAP                  "res\\lab_set.bmp"
IDB_WRITE_UP_43             BITMAP                  "res\\RFID-w2.bmp"
IDB_WRITE_DOWN_43           BITMAP                  "res\\RFID-w1.bmp"
IDB_READ_UP_43              BITMAP                  "res\\RFID-r2.bmp"
IDB_READ_DOWN_43            BITMAP                  "res\\RFID-r1.bmp"
IDB_CLEAR_UP_43             BITMAP                  "res\\RFID-c2.bmp"
IDB_CLEAR_DOWN_43           BITMAP                  "res\\RFID-c1.bmp"
```

在 RFID 這個專案的 RFID.cpp 中則利用以下的實作來判斷 LCD 的尺寸大小：

```
int x = GetSystemMetrics(SM_CXSCREEN);
int y = GetSystemMetrics(SM_CYSCREEN);

if(x == 480 && y == 272) //4.3
{
        g_DllRes = LoadLibrary (_T("RFID_43.dll"));
        if (g_DllRes == NULL)
        {
                MessageBox(NULL, _T("need RFID_43.dll"), _T("DMATEK"),
MB_OK|MB_ICONERROR);
                return FALSE;
        }
        else
        {
                g_lcd = LCD_43;
        }
}
else if(x == 800 && y == 480) //7
{
        g_DllRes = LoadLibrary (_T("RFID_70.dll"));
        if (g_DllRes == NULL)
        {
                MessageBox(NULL, _T("need RFID_70.dll"), _T("DMATEK"),
MB_OK|MB_ICONERROR);
                return FALSE;
        }
        else
        {
```

```
                g_lcd = LCD_70;
            }
        }
        else
        {
            MessageBox(NULL, _T("unknown system"), _T("DMATEK"),
MB_OK|MB_ICONERROR);
            return FALSE;
        }
```

並在 RFIDDlg.cpp 中以這段程式碼分別實作 4.3” LCD 及 7” LCD 兩種尺寸大小的介面描繪：

```
CRFIDDlg::CRFIDDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CRFIDDlg::IDD, pParent)
{
    if(g_lcd == LCD_43)
    {
    // 請詳見程式碼實作
    }
    else if(g_lcd == LCD_70)
    {
    // 請詳見程式碼實作
    }
}
```

第八章　聯繫方式

若您有任何問題請使用下列方式與我們聯繫：

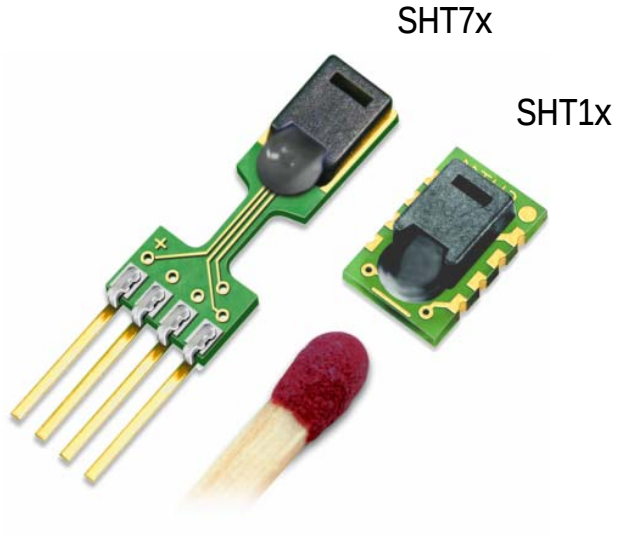
1、網路技術論壇（推薦）

http://www.dmatek.com.cn/bbs/Default.asp

2、其他方式：

公司名稱：	長高科技股份有限公司
公司電話：	886-4-22516589（Rep）
公司傳真：	886-4-22516836
公司郵箱：	Info@dmatek.com.tw

数字温湿度传感器
SHT 1x / SHT 7x
（请以英文为准，译文仅供参考）



_ 相对湿度和温度测量
_ 兼有露点
_ 全标定输出，无需标定即可互换使用
_ 卓越的长期稳定性
_ 两线制数字接口，无需额外部件
_ 基于请求式测量，因此低能耗
_ 表面贴片或 4 针引脚安装
_ 超小尺寸
_ 自动休眠
_ 超快响应时间

SHT1x / SHT7x 产品概述

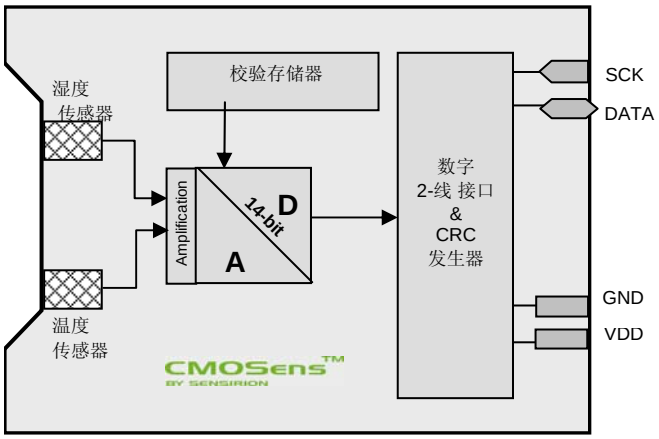
SHTxx 系列产品是一款高度集成的温湿度传感器芯片，提供全标定的数字输出。它采用专利的 CMOSens® 技术，确保产品具有极高的可靠性与卓越的长期稳定性。传感器包括一个电容性聚合体测湿敏感元件、一个用能隙材料制成的测温元件，并在同一芯片上，与 14 位的 A/D 转换器以及串行接口电路实现无缝连接。因此，该产品具有品质卓越、超快响应、抗干扰能力强、极高的性价比等优点。

每个传感器芯片都在极为精确的湿度腔室中进行标定，以镜面冷凝式湿度计为参照。校准系数以程序形式储存在 OTP 内存中，在标定的过程中使用。
两线制的串行接口与内部的电压调整，使外围系统集成变得快速而简单。微小的体积、极低的功耗，使其成为各类应用的首选。
产品提供表面贴片 LCC 或 4 针单排引脚封装。特殊封装形式可根据用户需求而提供。

应用领域

_ 暖通空调 HVAC
_ 汽车
_ 消费品
_ 气象站
_ 湿度调节器
_ 除湿器
_ 测试及检测设备
_ 数据记录器
_ 自动控制
_ 家电
_ 医疗

框图



订货信息

型号	测湿精度 [%RH]	测温精度 [℃]在 25℃	封装
SHT 10	±4.5	±0.5	SMD（LCC）
SHT 11	±3.0	±0.4	SMD（LCC）
SHT 15	±2.0	±0.3	SMD（LCC）
SHT 71	±3.0	±0.4	4-pin 单排直插
SHT 75	±1.8	±0.3	4-pin 单排直插

1 传感器性能说明

参数	条件	Min.	Typ.	Max.	单位
湿度					
分辨率 (2)		0.5	0.03	0.03	%RH
		8	12	12	Bit
重复性			±0.1		%RH
精度 (1) 不确定性	线性化	参见图 1			
互换性		可完全互换			
非线性度	原始数据		±3		%RH
	线性化		<<1		%RH
量程范围		0		100	%RH
响应时间	1/e (63%) 缓慢流动空气		4		S
迟滞			±1		%RH
长期稳定性	典型值		< 0.5		%RH/yr
温度					
分辨率 (2)		0.04	0.01	0.01	°C
		0.07	0.02	0.02	°F
		12	14	14	Bit
重复性			±0.1		°C
			±0.2		°F
精度		参见图 1			
量程范围		-40		123.8	°C
		-40		254.9	°F
响应时间	1/e (63%)	5		30	s

表 1 传感器性能说明

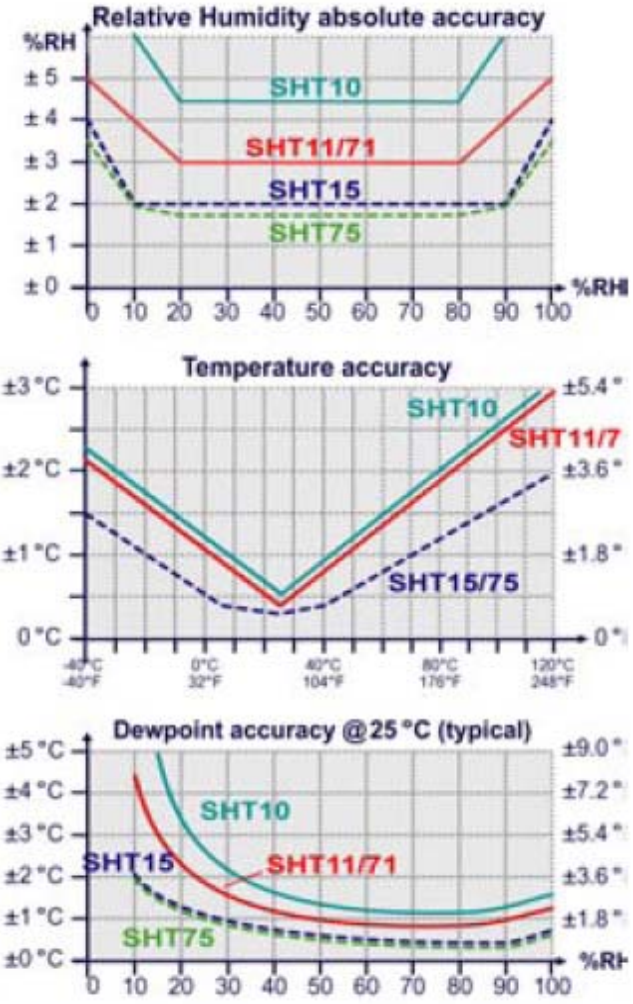


图 1 相对湿度、温度和露点的精度曲线

2 接口说明

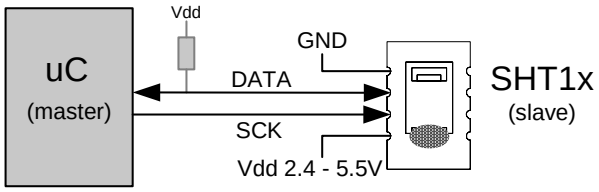


图 2 典型应用电路

2.1 电源引脚

SHTxx 的供电电压为 2.4~5.5V。传感器上电后，要等待 11ms 以越过“休眠”状态。在此期间无需发送任何指令。电源引脚（VDD，GND）之间可增加一个 100nF 的电容，用以去耦滤波。

2.2 串行接口 (两线双向)

SHTxx 的串行接口，在传感器信号的读取及电源损耗方面，都做了优化处理；但与 I^2C 接口不兼容，详情参见 FAQ .

2.2.1 串行时钟输入 (SCK)

SCK 用于微处理器与 SHTxx 之间的通讯同步。由于接口包含了完全静态逻辑，因而不存在最小 SCK 频率。

2.2.2 串行数据 (DATA)

DATA 三态门用于数据的读取。DATA 在 SCK 时钟下降沿之后改变状态，并仅在 SCK 时钟上升沿有效。数据传输期间，在 SCK 时钟高电平时，DATA 必须保持稳定。为避免信号冲突，微处理器应驱动 DATA 在低电平。需要一个外部的上拉电阻（例如：10kΩ）将信号提拉至高电平（参见图 2）。上拉电阻通常已包含在微处理器的 I/O 电路中。详细的 IO 特性，参见表 5。

(1) 每支 SHTxx 传感器在 25℃（77 °F）和 48℃（118.4 °F），均进行过全量程 RH 精度标定。

(2) 默认的测量精度为 14bit（温度）和 12bit（湿度），通过状态寄存器可分别降至 12bit 和 8bit。

2.2.3 发送命令

用一组“ 启动传输”时序，来表示数据传输的初始化。它包括：当 SCK 时钟高电平时 DATA 翻转为低电平，紧接着 SCK 变为低电平，随后是在 SCK 时钟高电平时 DATA 翻转为高电平。

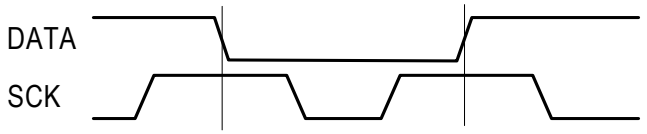


图 3 "启动传输" 时序

后续命令包含三个地址位（目前只支持“000”），和五个命令位。SHTxx 会以下述方式表示已正确地接收到指令：在第 8 个 SCK 时钟的下降沿之后，将 DATA 下拉为低电平（ACK 位）。在第 9 个 SCK 时钟的下降沿之后，释放 DATA（恢复高电平）。

命令	代码
预留	0000x
温度测量	00011
湿度测量	00101
读状态寄存器	00111
写状态寄存器	00110
预留	0101x-1110x
软复位，复位接口、清空状态寄存器,，即清空为默认值 下一次命令前等待至少 11ms	11110

表 2 命令集

2.2.4 测量时序(RH 和 T)

发布一组测量命令（‘00000101’表示相对湿度 RH，‘00000011’表示温度 T）后，控制器要等待测量结束。这个过程需要大约 11/55/210ms，分别对应 8/12/14bit 测量。确切的时间随内部晶振速度，最多有±15%变化。SHTxx 通过下拉 DATA 至低电平并进入空闲模式，表示测量的结束。控制器在再次触发 SCK 时钟前，必须等待这个“数据备妥”信号来读出数据。检测数据可以先被存储，这样控制器可以继续执行其它任务在需要时再读出数据。

接着传输 2 个字节的测量数据和 1 个字节的 CRC 奇偶校验。uC 需要通过下拉 DATA 为低电平，以确认每个字节。所有的数据从 MSB 开始，右值有效（例如：对于 12bit 数据，从第 5 个 SCK 时钟起算作 MSB；而对于 8bit 数据，首字节则无意义）。用 CRC 数据的确认位，表明通讯结束。如果不使用 CRC-8 校验，控制器可以在测量值 LSB 后，通过保持确认位 ack 高电平，来中止通讯。在测量和通讯结束后，SHTxx 自动转入休眠模式。

警告：为保证自身温升低于 0.1℃，SHTxx 的激活时间不要超过 15%（例如，对应 12bit 精度测量，每秒最多进行 3 次测量）。

2.2.5 通讯复位时序

如果与 SHTxx 通讯中断，下列信号时序可以复位串口：

当 DATA 保持高电平时，触发 SCK 时钟 9 次或更多。在下一次指令前，发送一个“传输启动”时序。这些时序只复位串口，状态寄存器内容仍然保留。

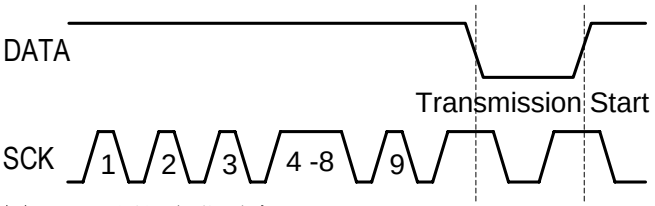


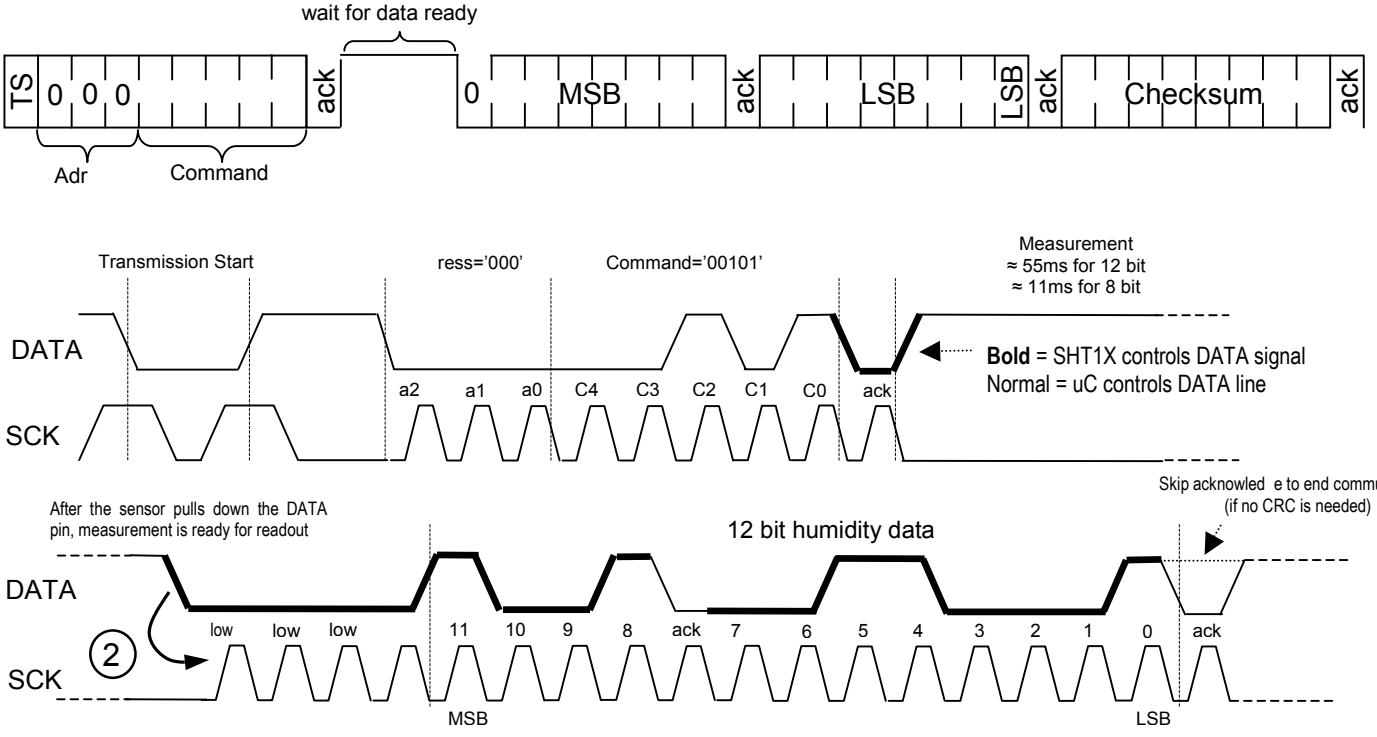
图 4 通讯复位时序

2.2.6 CRC-8 校验

数字信号的整个传输过程由 8bit 校验来确保。任何错误数据将被检测到并清除。

详情可参阅应用说明“CRC-8 校验”。

Table 2 list of commands



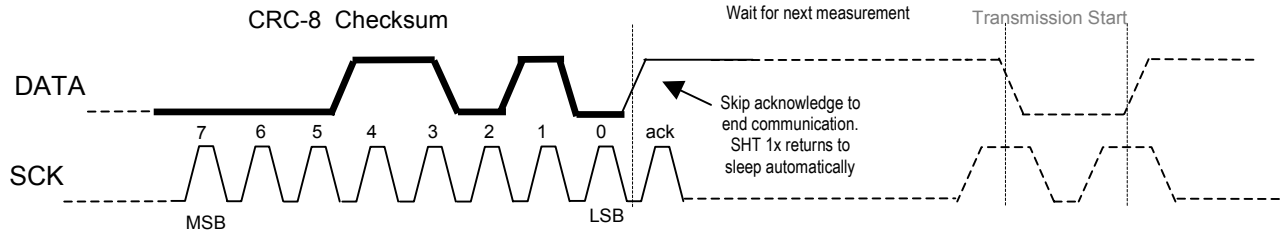


图 6 测量时序概览 (TS = 启动传输) Example RH measurement sequence for value "0000'1001 ' 0011'0001"= 2353 = 75.79%RH

2.3 状态寄存器

SHTxx 的某些高级功能可以通过状态寄存器实现。下面的章节概括介绍了这些功能。

详情可参阅应用说明"状态寄存器"。

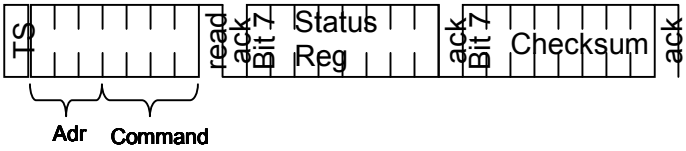


图 7 状态寄存器读

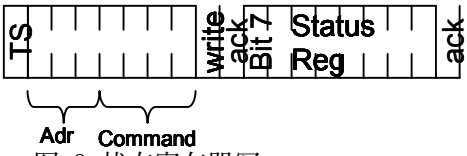


图 8 状态寄存器写

Bit	类型	说明	默认值	
7		预留	0	
6	R	电量不足 (低电压检测) '0'对应 Vdd > 2.47 '1'对应 Vdd < 2.47	X	无默认值, 此位仅在测量结束后更新
5		预留	0	
4		预留	0	
3		仅供测试, 不使用	0	
2	R/W	加热	0	关
1	R/W	不从 OTP 加载	0	加载
0	R/W	'1'= 8bit RH / 12bit T 分辨率 '0'=12bit RH / 14bit T 分辨率	0	12bit RH 14bit T

表 3 状态寄存器位

2.3.1 测量分辨率

默认的测量分辨率分别为 14bit（温度）、12bit（湿度），也可分别降至 12bit 和 8bit。通常在高速或超低功耗的应用中采用该功能。

2.3.2 电量不足

"电量不足"功能可监测到 Vdd 电压低于 2.47V 的状态。精度为±0.05V。

2.3.3 加热元件

芯片上集成了一个可通断的加热元件。接通后，可将 SHTxx 的温度提高大约 5℃（9℉）。功耗增加 8mA @ 5V。

应用于：

比较加热前后的温度和湿度值，可以综合验证两个传感器元件的性能。

- 在高湿 (>95 %RH) 环境中，加热传感器可预防结露，同时缩短响应时间，提高精度。

警告: 加热 SHTxx 后温度升高、相对湿度降低，较之加热前，示值略有差异。

2.4 电气特性(1)

VDD=5V，T = 25℃，除非特殊标注

参数	条件	Min.	Typ.	Max.	单位
供电 DC		2.4	5	5.5	V
供电电流	测量		550		µA
	平均	2(2)	28(3)		µA
	休眠		0.3	1	µA
低电平输出电压		0		20%	Vdd
高电平输出电压		75%		100%	Vdd
低电平输入电压	下降沿	0		20%	Vdd
高电平输入电压	上升沿	80%		100%	Vdd
焊盘上的输入电流				1	µA
输出峰值电流	on			4	mA
	三态门 (off)		10		µA

表 4 SHTxx DC 特性

	参数	条件	Min	Typ.	Ma	单位
F_{SCK}	SCK 频率	VDD > 4.5 V			10	MHz
		VDD < 4.5 V			1	MHz
T_{RFO}	DATA 下降时间	输出负载 5 pF	3.5	10	20	ns
		输出负载 100 pF	30	40	200	ns
T_{CLx}	SCK 高/低时间		100			ns
T_V	DATA 有效时间			250		ns
T_{SU}	DATA 设定时间		100			ns
T_{HO}	DATA 保持时间		0	10		ns
T_R/T_F	SCK 升/降时间			200		ns

表 5 SHTxx I/O 信号特性

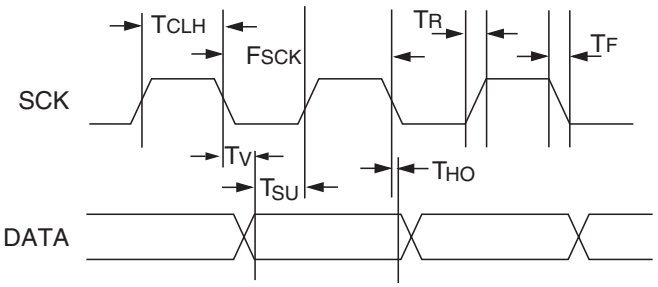


图 9 时序图

1) 试样参数周期抽检但非 100% 检测
(2) 每秒进行一次 8bit 精度的测量，不加载 OTP
(3) 每秒进行一次 12bit 精度的测量

3 输出转换为物理量

3.1 相对湿度

为了补偿湿度传感器的非线性以获取准确数据，建议使用如下公式 [1]修正读数：

$$RH_{linear} = c_1 + c_2 \bullet SO_{RH} + c_3 \bullet SO_{RH}^2$$

SO_{RH}	c_1	c_2	c_3
12 bit	-4	0.0405	$-2.8 * 10^{-6}$
8 bit	-4	0.648	$-7.2 * 10^{-4}$

表 6 湿度转换系数

简化的修正算法，可参阅应用说明“相对湿度与温度的非线性补偿”。
湿度传感器对电压基本上没有依赖性。

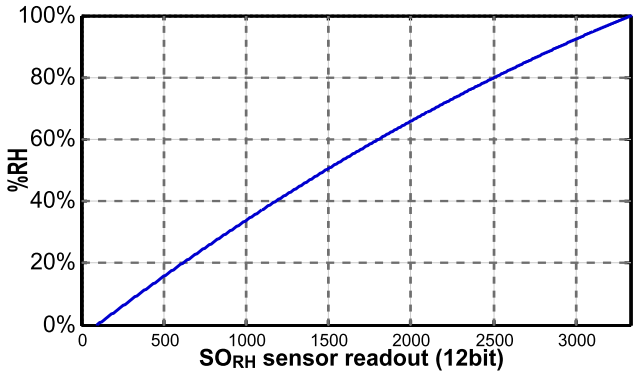


图 10 从 SO_{RH} 转换到相对湿度

3.1.1 相对湿度对于温度依赖性的补偿

由于实际温度与测试参考温度25℃ (~77℉)的显著不同， 应考虑湿度传感器的温度修正系数：

$$RH_{true} = (T_{°C} - 25) \bullet (t_1 + t_2 \bullet SO_{RH}) + RH_{linear}$$

SO_{RH}	t_1	t_2
12 bit	0.01	0.00008
8 bit	0.01	0.00128

表 7 温度补偿系数

相当于 ~0.12 %RH /℃ @ 50 %RH

3.2 温度

由能隙材料 PTAT (正比于绝对温度) 研发的温度传感器具有极好的线性。可用如下公式将数字输出转换为温度值：

$$Temperature = d_1 + d_2 \bullet SO_T$$

VDD	d_1 [℃]	d_1 [℉]
5V	-40.00	-40.00
4V	-39.75	-39.50
3.5V	-39.66	-39.35
3V	-39.60	-39.28
2.5V	-39.55	-39.23

SO_T	d_2 [℃]	d_2 [℉]
14bit	0.01	0.018
12bit	0.04	0.072

表 8 温度转换系数

在极端工作条件下测量温度时，可使用进一步的补偿算法以获取高精度。可参阅应用说明“相对湿度与温度的非线性补偿”。

3.3 露点

由于湿度与温度经由同一块芯片测量，SHTxx 系列产品可以同时实现高质量的露点测量。可参阅应用说明“露点计算”。

[1] SO_{RH} 表示传感器的相对湿度输出

4 应用信息

4.1 工作与贮存条件

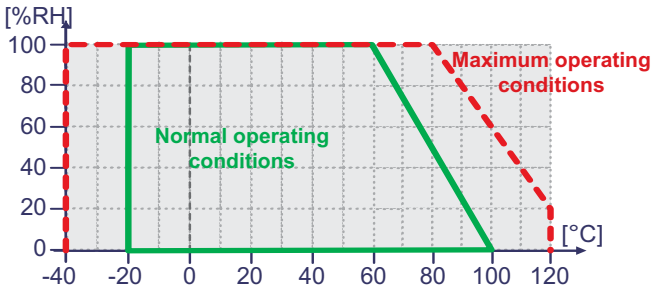


图 11 建议的工作条件

超出建议的工作范围可能导致最大 3%的 RH 信号暂时性漂移。当恢复到正常工作条件后，传感器会缓慢自恢复到校正状态。可参阅 4.3 小节的“恢复处理”以加速恢复进程。在非正常条件下的长时间使用，会加速产品的老化。

4.2 暴露在化学物质中

用于湿度测量的聚合物会受到化学蒸汽的侵蚀，化学物质在聚合物中的扩散可能导致测量元件精度的漂移与灵敏性下降。在纯净的环境中，污染物将缓慢释放。下文所述的恢复处理将加速实现这一过程。强化学污染可能导致传感器的彻底损坏。

4.3 恢复处理

暴露在极端工作条件或化学蒸汽中的传感器，可通过如下处理，使其恢复到校准状态。

在 80-90℃ (176-194℉) 和< 5%RH 的湿度条件下保持 24 小时（烘干）； 随后

在 20-30℃ (70-90℉) 和>74%RH 的湿度条件下保持 48 小时（重新水合）

4.4 温度影响

气体的相对湿度，在很大程度上依赖于温度。因此在测量湿度时，应尽可能保证湿度传感器在同一温度下工作。

如果与释放热量的电子元件共用一个印刷线路板，在安装时应尽可能将SHTxx远离电子元件，并安装在热源下方，同时保持外壳的良好通风。

为降低热传导，SHT1x 与印刷电路板其它部分的铜镀层应尽可能最小，并在两者之间留出一道缝隙 (参见图 13) 。

4.5 薄膜

薄膜可防止灰尘进入，以保护传感器。同时会减少化学蒸汽的浓度。为获取最佳的响应时间，薄膜内的空气量必须保持在最少。对于 SHT1X 封装系列，盛世瑞恩推荐使用 SF1 型过滤罩，以达到最优的 IP67 保护等级。

4.6 光线

SHTxx 对光线不敏感。但长时间暴露在太阳光下或强烈的紫外线辐射中，会使外壳老化。

4.7 用于密封和安装的材质

许多材质吸收湿气并将充当缓冲器的角色，这会加大响应时间和迟滞。因此传感器周边的材质应谨慎选用。推荐使用的材料有：所有的金属, LCP, POM (Delrin), PTFE (Teflon), PE, PEEK，PP, PB, PPS, PSU, PVDF, PVF

用于密封和粘合的材质（保守推荐）：推荐使用充满环氧树脂的方法进行电子元件的封装，或是硅树脂。这些材料释放的气体也有可能污染SHTxx(见4.2)。加工后应将传感器置于通风良好处，或在50℃的环境中干燥24小时，以使其在封装前将污染气体释放。

4.8 接线注意事项与信号完整性

使SCK和DATA信号线平行并尽可能使间距超过10cm（如使用导线），否则将导致串扰和信号丢失。解决方法是在两个信号线之间放置VDD和/或GND。

详情可参阅应用说明“ESD、latch-up和EMC”。

如使用导线，应在电源引脚（VDD，GND）之间可增加一个 100nF 的电容，用以去耦滤波。

4.9 产品资质

本品已经通过了大量的多种环境测试。

环境	标准	结果(1)
温度循环	JESD22-A104-B -40℃ / 125℃, 1000cy	符合本手册
高温蒸煮	JESD22-A110-B 2.3bar 125℃@5%RH	+2 %RH 的可逆漂移
食盐蒸气	DIN-50021ss	符合本手册
冷凝空气	-	符合本手册
冷冻循环 完全浸没	-20 / +90℃, 100cy 30min 驻留时间	+2 %RH 的可逆漂移
多种汽车化学品	DIN 72300-5	符合本手册
香烟烟幕	相当于在中型汽车上 15 年	符合本手册

表 9 品质测试 (摘录)

4.10 ESD (静电释放)

ESD静电释放符合MIL STD 883E method 3015标准。电路闭锁测试依据JEDEC 17标准，满足强制电流在±100 mA，环境温度T_{amb} = 80℃条件下不闭锁。详情可参阅应用说明“ESD、latch-up、EMC”。

(1) 温度传感器通过了所有的测试，没有任何漂移。 亦 100%通过包装及电子测试。

5 包装信息

5.1 SHT1x (表面贴装)

Pin	名称	注释
1	GND	地
2	DATA	串行数据, 双向
3	SCK	串行时钟, 输入
4	VDD	供电 2.4 - 5.5 V
	NC	剩余引脚请勿连结

表 10 SHT1x 引脚说明

5.1.1 包装类型

SHT1x 采用表面贴装LCC（无铅芯片载体）包装方式。液晶聚合物环氧包覆外壳，标准0.8 mm FR4衬底。不含铅、铬、汞。
尺寸：7.62×5.08×2.5 mm
重量：100 毫克

生产日期用白色数字标识于传感器顶部，格式为 wwy. e.g.” 351” = week 35, 2001.

5.1.2 运输条件

SHT1x 置于标准 IC 管中运输，每管 80 片，或以 12mm 胶带卷装。胶盘以条形码或可读标签做单独标记。

图 12 胶带结构和单片包装

5.1.3 焊接信息

使用标准的回流焊炉，在最高 235℃的温度条件下不超过 30 秒。

手动焊接，在最高 350℃的温度条件下接触时间须少于 5 秒。
焊接后，将传感器在>74%RH的环境下存放至少48小时，以保证聚合物的重新水合。
详情可参阅应用说明“焊接规程”。

5.1.4 安装举例

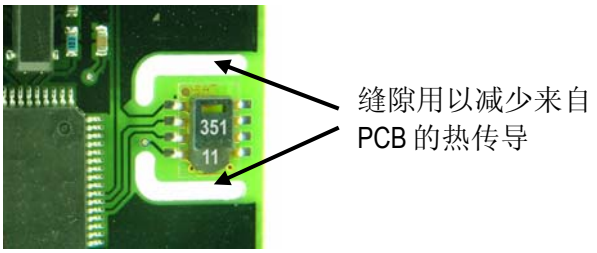


图 13 SHT1x PCB 安装举例

使用 SF1 型薄膜过滤罩可以达到 IP67 的保护等级。使用外壳包装，可以保护内部不受环境影响，从而保证高精度的湿度测量。

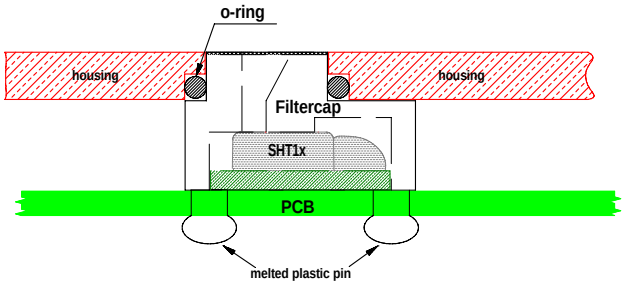


图 14 SHT1x 安装举例

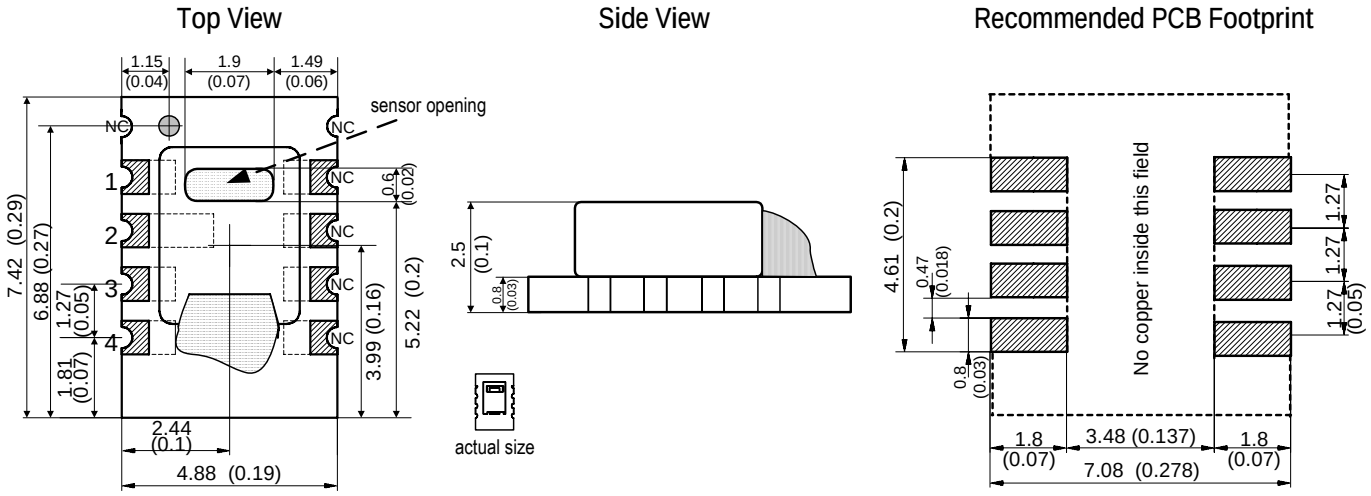


图 15 SHT1x 尺寸图和引脚尺寸 mm (inch)

5.2 SHT7x (4-pin 单排引脚)

Pin	名称	注释
1	*SCK*	*串行时钟, 输入*
2	*VDD*	*供电 2.4 - 5.5 V*
3	*GND*	*地*
4	*DATA*	串行数据, 双向

表 11 SHT7x 引脚说明

5.2.1 包装类型[1]

SHT7x采用4针的单排引脚形式包装。液晶聚合物环氧包覆外壳，标准0.8 mm FR4衬底。不含铬、汞。

传感器头部通过小桥接器实现与引脚的连接，以降低热传导及响应时间。传感器头部背面的镀金板与GND引脚相连。

在背面VDD与GND之间安装了一个100nF的电容。

所有引脚均镀金处理，以防腐蚀。可焊接使用，也可与1.27 mm (0.05")的插槽匹配。

例如： Preci-dip / Mil-Max 851-93-004-20-001或类似产品。

总重量：168 mg, 传感器头部重量：73 mg

生产日期用白色数字标识于传感器顶部，格式为wwy. e.g. "351" = week 35, 2001.

5.2.2 运输条件

SHT7x 以 32mm 胶带卷装运输。每个直径为 13 英寸的标准胶盘可装 500 片。胶盘以条形码或可读标签做单独标记。

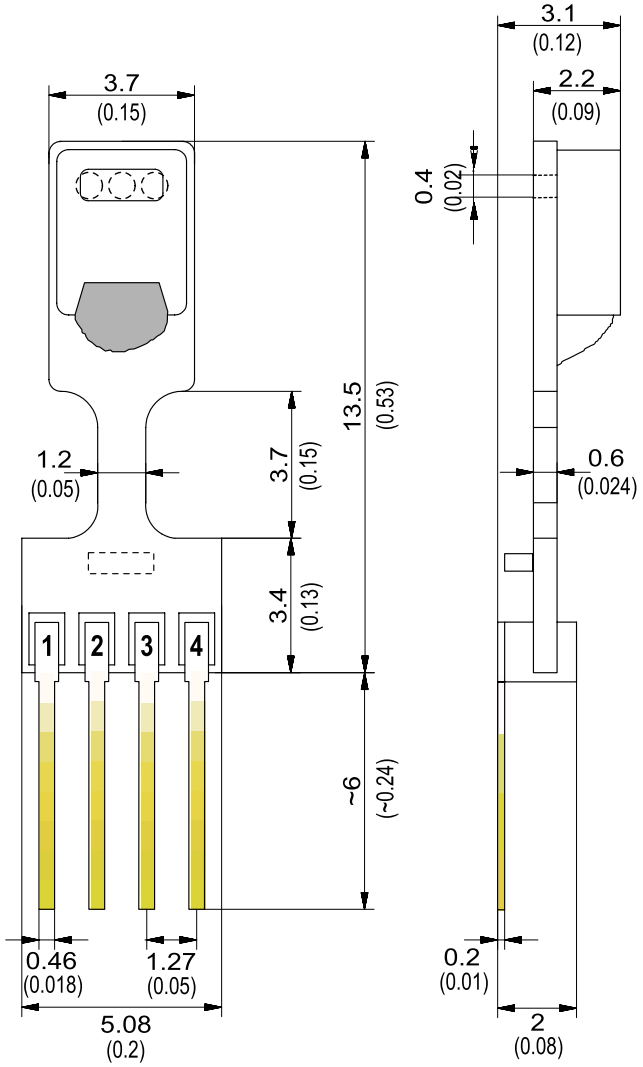


图 17 SHT7x 尺寸 mm (inch)

图 16 胶带结构和单片包装

5.2.3 焊接信息

使用标准的波动焊炉，在最高 250℃的温度条件下不超过 30 秒。

手动焊接，在最高 350℃的温度条件下接触时间须少于 5 秒。

焊接后，将传感器在>74%RH的环境下存放至少48小时，以保证聚合物的重新水合。

详情可参阅应用说明"焊接规程"。

[1] 可根据特殊需求提供其它包装方式。

6 版本

日期	版本	页数(s)	变更
2002.02	初稿	1-9	首次发布
2002.06	初稿		增加了 SHT7x 的内容
2003.05	定稿 v2.0	1-9	大调整，增加了应用说明部分 多方面的小改动
	V2.01	1-9	打字稿，加入曲线标注
2004.06	V2.02	1-9	改进了说明书，加入 SF1 信息，改进了某些用词
2004.05	V2.03	1-2	加入 SHT10 信息
2004.05	V2.04	1-9	修改公司信息

7 注意事项

7.1 警告，人身伤害

勿将本产品作为安全保护或急停设备，以及其它由于本产品故障导致人身伤害的应用中。
如不遵从此建议，可能导致死亡和严重的人身伤害。

7.2 ESD 静电释放的预防

由于元件的固有设计，导致其对静电的敏感性。为防止静电导入的伤害或者降低产品性能，在应用本产品时，请采取必要的防静电措施。
详情可参阅应用说明“ESD、latch-up、EMC”。

7.3 品质保证

SENSIRION AG对其产品应用在那些特殊的应用场合不做任何的保证、担保、或是书面陈述。同时SENSIRION AG对其产品应用到产品或是电路中的可靠性也不做任何承诺。

TGS813　可燃气体检测用

特点：

- 对较宽范围的可燃气体都灵敏的普敏气体传感器
- 对甲烷、丙烷、异丁烷 的灵敏度很高
- 长寿命、低成本
- 以简单电路即可使用

应用：

- 家庭用气体泄漏报警器
- 工业用可燃气体报警器
- 便携式气体检知器

费加罗气体传感器的气敏素子，使用在清洁空气中电导率低的二氧化锡(SnO2)。当存在检知对象气体时，传感器的电导率随空气中气体浓度增加而增大。使用简单的电路即可将电导率的变化，转换为与该气体浓度相对应的输出信号。

TGS813 传感器对甲烷、丙烷、丁烷的灵敏度高，对天然气、液化气的监视也很理想。这种传感器可检知多种可燃气体，所以是对各种应用方式都很优越的低成本传感器。此外，还有能耐 200℃苛刻气氛的陶瓷底座传感器（型号为 TGS816）。

下图是典型的灵敏度特性，全部是在标准试验条件下得出的结果。（请看背面）

纵坐标以传感器电阻比（Rs/Ro）表示，Rs，Ro 的定义如下：

Rs ＝不同浓度气体中的电阻值

Ro ＝1000ppm 甲烷中的电阻值

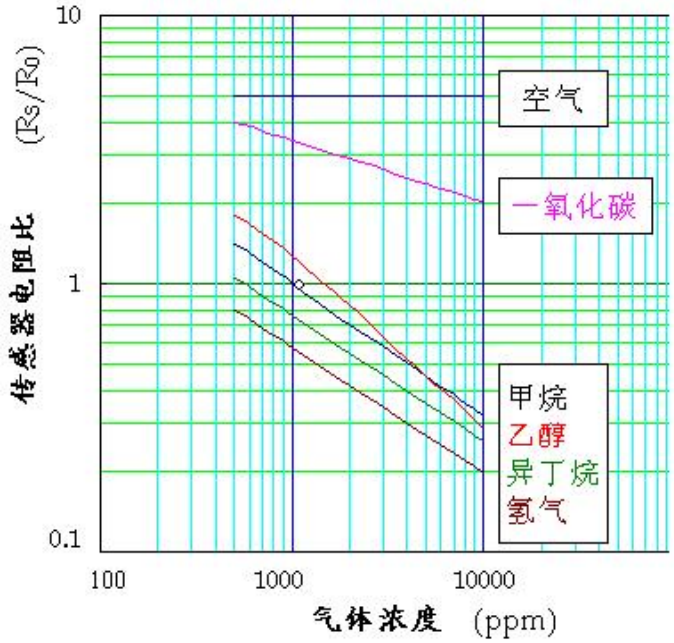
下图为受温度、湿度影响的典型曲线。

图中纵坐标也以传感器电阻比（Rs/Ro）表示，这里的 Rs，Ro 定义如下：

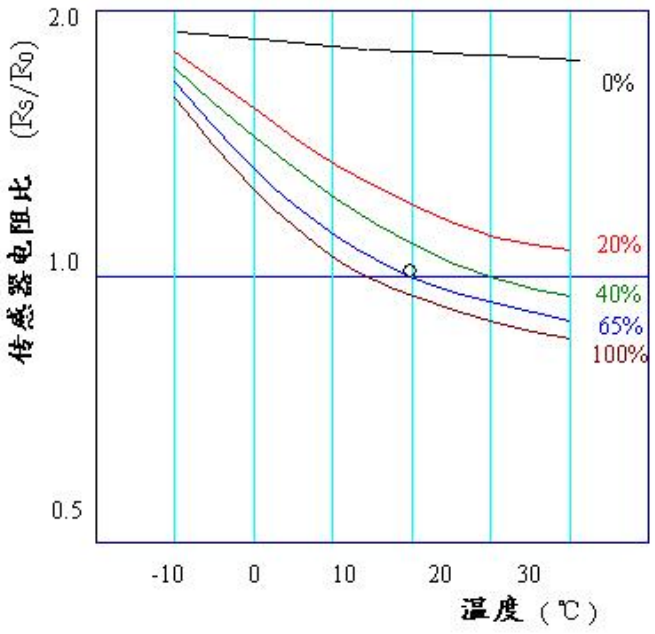
Rs=含 1000ppm 甲烷、各种温/湿度下的电阻值

Ro=含 1000ppm 甲烷、20℃65%R.H.下的电阻值

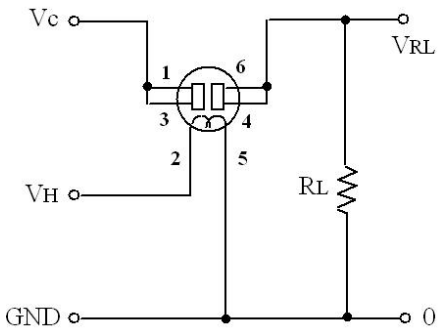
灵敏度特性：



温/湿度的影响：



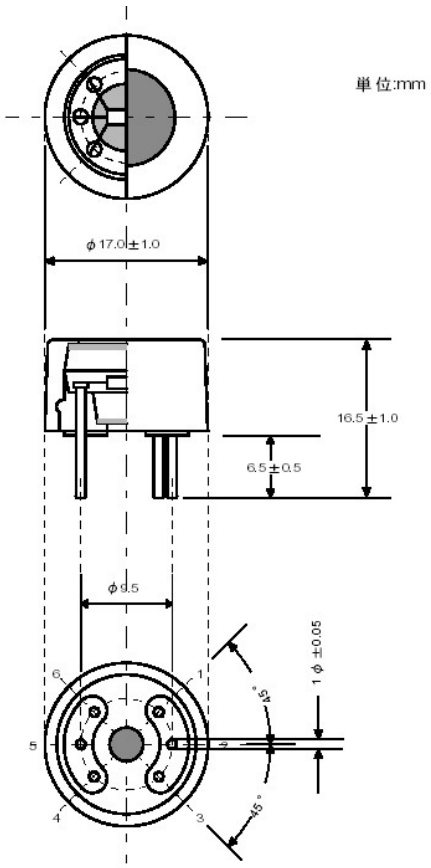
FIGARO



规格：

型　　号			TGS813	
素子类型			8 系列	
标准封装			塑料、SUS 双重金属网	
对象气体			可燃性气体	
检测范围			500～10,000 ppm	
标准回路条件	加热器电压	VH	5.0±0.2V　DC/AC	
	回路电压	VC	MAX　24V	Ps≦15mW
	负载电阻	RL	可变	Ps≦15mW
标准试验条件下的电学特性	加热器电阻	RH	30±3.0　Ω（室温）	
	加热器功耗	PH	835±90mW VH＝5.0V	
	传感器电阻	Rs	甲烷 1000ppm 中　5～15KΩ	
	灵敏度(Rs 的变化率)		0.6±0.05	Rs(CH4:3000ppm) Rs(CH4:1000ppm)
标准试验条件	试验气体条件		20±2℃，　65±5%RH	
	回路条件		VC＝10.0±0.1V　DC/AC VH＝5.0±0.05V　DC/AC RL=4.0　KΩ±1%	
	预热时间		7 天以上	

结构及尺寸：



功耗（Ps）值可用下式计算：

$$Ps = \frac{Vc^2 \times Rs}{(Rs + RL)^2}$$

传感器电阻（Rs），可用下式计算：

$$Rs = \left(\frac{Vc}{VRL} - 1\right) \times RL$$

管脚连接　1 或 3：传感器
　　4 或 6：传感器
　　2　　：加热器
　　5　　：加热器

为提高性能，本规格书将不事先预告而变更。

深圳市新世联科技有限公司

联系人：胡保定
手机：15018532664
邮箱：s201@apollounion.com

座机：0755-83680810-818（分机）
地址：深圳市深南中路2066号华能大厦712室

TC72

Digital Temperature Sensor with SPI™ Interface

Features

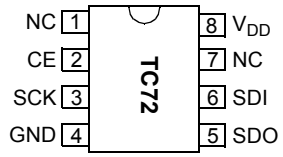
- Temperature-to-Digital Converter
- SPI™ Compatible Interface
- 10-Bit Resolution (0.25°C/Bit)
- ±2°C (max.) Accuracy from -40°C to +85°C
- ±3°C (max.) Accuracy from -55°C to +125°C
- 2.65V to 5.5V Operating Range
- Low Power Consumption:
  - 250 µA (typ.) Continuous Temperature Conversion Mode
  - 1 µA (max.) Shutdown Mode
- Power Saving One-Shot Temperature Measurement
- Industry Standard 8-Pin MSOP Package
- Space Saving 8-Pin DFN (3x3 mm) Package

Typical Applications

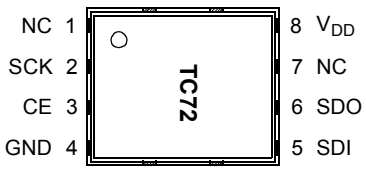
- Personal Computers and Servers
- Hard Disk Drives and Other PC Peripherals
- Entertainment Systems
- Office Equipment
- Datacom Equipment
- Mobile Phones
- General Purpose Temperature Monitoring

Package Types

MSOP



DFN



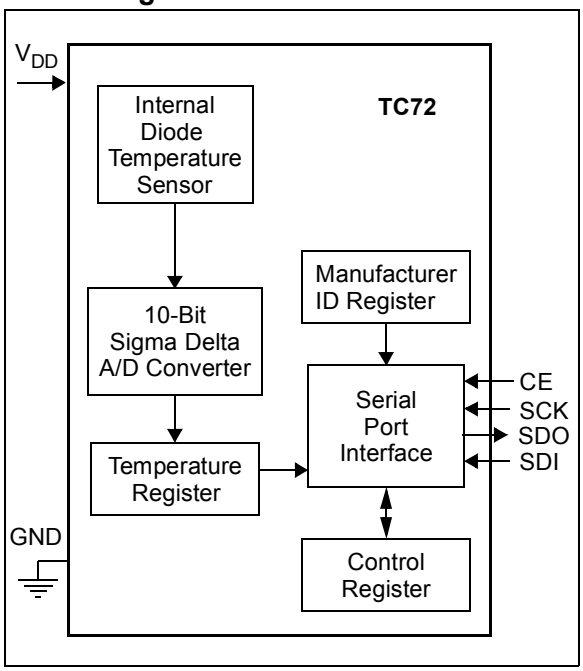
General Description

The TC72 is a digital temperature sensor capable of reading temperatures from -55°C to +125°C. This sensor features a serial interface that allows communication with a host controller or other peripherals. The TC72 interface is compatible with the SPI protocol. The TC72 does not require any additional external components. However, it is recommended that a decoupling capacitor of 0.01 µF to 0.1 µF be provided between the V_{DD} and GND pins.

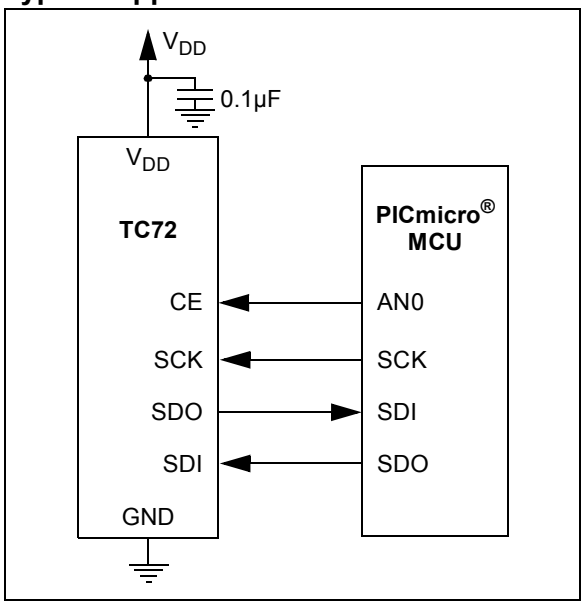
The TC72 can be used either in a Continuous Temperature Conversion mode or a One-Shot Conversion mode. The Continuous Conversion mode measures the temperature approximately every 150 ms and stores the data in the temperature registers. In contrast, the One-Shot mode performs a single temperature measurement and returns to the power saving shutdown mode.

The TC72 features high temperature accuracy, ease-of-use and is the ideal solution for implementing thermal management in a variety of systems. The device is available in both 8-pin MSOP and 8-pin DFN space-saving packages. The TC72 also features a shutdown mode for low power operation.

Block Diagram



Typical Application



1.0 ELECTRICAL CHARACTERISTICS

1.1 Maximum Ratings†

V_{DD}.....6.0V

All inputs and outputs w.r.t. GND ...-0.3V to V_{DD} +0.3V

Storage temperature .....-65°C to +150°C

Ambient temp. with power applied .....-55°C to +125°C

Junction Temperature ..... 150°C

ESD protection on all pins:

Human Body Model (HBM).....> 4 kV

Man Machine Model (MM).....> 400V

Latch-Up Current at each pin .....±200 mA

Maximum Power Dissipation.....250 mW

† Notice: Stresses above those listed under "Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

PIN FUNCTION TABLE

Name	Function
NC	No Internal Connection
CE	Chip Enable Input, the device is selected when this input is high
SCK	Serial Clock Input
GND	Ground
SDO	Serial Data Output
SDI	Serial Data Input
NC	No Internal Connection
V_{DD}	Power Supply

DC CHARACTERISTICS

Electrical Specifications: Unless otherwise noted, all parameters apply at V_{DD} = 2.65V to 5.5V, T_A = -55°C to +125°C.

Parameters	Sym	Min	Typ	Max	Units	Conditions
Power Supply						
Operating Voltage Range	V_{DD}	2.65	—	5.5	V	**Note 1**
Operating Current: Normal Mode, ADC Active	I_{DD-CON}	—	250	400	µA	Continuous temp. conversion mode (Shutdown Bit = '0')
Shut-Down Supply Current	I_{SHD}	—	0.1	1.0	µA	Shutdown Mode (Shutdown Bit = '1')
Temperature Sensor and Analog-to-Digital Converter						
Temperature Accuracy (**Note 1)**	T_{ACY}	-2.0	—	+2.0	°C	-40°C < T_A < +85°C
		-3.0	—	+3.0		-55°C < T_A < +125°C
Resolution		—	10	—	Bits	**Note 4**
ADC Conversion Time	t_{CONV}	—	150	200	ms	
Digital Input / Output						
High Level Input Voltage	V_{IH}	0.7 V_{DD}	—	—	V	
Low Level Input Voltage	V_{IL}	—	—	0.2 V_{DD}	V	
High Level Output Voltage	V_{OH}	0.7 V_{DD}	—	—	V	I_{OH} = 1 mA
Low Level Output Voltage	V_{OL}	—	—	0.2 V_{DD}	V	I_{OL} = 4 mA
Input Resistance	R_{IN}	1.0	—	—	MΩ	
Pin Capacitance	C_{IN}	—	15	—	pF	
	C_{OUT}	—	50	—		

Note 1: The TC72-2.8MXX, TC72-3.3MXX and TC72-5.0MXX will operate from a supply voltage of 2.65V to 5.5V. However, the TC72-2.8MXX, TC72-3.3MXX and TC72-5.0MXX are tested and specified at the nominal operating voltages of 2.8V, 3.3V and 5.0V respectively. As V_{DD} varies from the nominal operating value, the accuracy may be degraded. Refer to Figure 2-5 and Figure 2-6.

2: Measured with a load of C_L = 50 pF on the SDO output pin of the TC72.

3: All time measurements are measured with respect to the 50% point of the signal, except for the SCK rise and fall times. The rise and fall times are defined as the 10% to 90% transition time.

4: Resolution = Temperature Range/No. of Bits = (+127°C – -128°C) / (2^{10}) = 256/1024 = 0.25°C/Bit

DC CHARACTERISTICS (CONTINUED)

Electrical Specifications: Unless otherwise noted, all parameters apply at V_{DD} = 2.65V to 5.5V, T_A = -55°C to +125°C.						
Parameters	**Sym**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
Serial Port AC Timing (Note 2, 3)						
Clock Frequency	f_{CLK}	DC	—	7.5	MHz	
SCK Low Time	t_{CL}	65	—	—	ns	
SCK High Time	t_{CH}	65	—	—	ns	
CE to SCK Setup	t_{CC}	400	—	—	ns	
SCK to Data Out Valid	t_{CDD}	—	—	55	ns	
CE to Output Tri-state	t_{CDZ}	—	—	40	ns	
SCK to Data Hold Time	t_{CDH}	35	—	—	ns	
Data to SCK Set-up Time	t_{DC}	35	—	—	ns	
SCK to CE Hold Time	t_{CCH}	100	—	—	ns	
SCK Rise Time	t_R	—	—	200	ns	
SCK Fall Time	t_F	—	—	200	ns	
CE Inactive Time	t_{CWH}	400	—	—	ns	
Thermal Package Resistance						
Thermal Resistance, MSOP-8	θ_{JA}	—	206	—	°C/W	
Thermal Resistance, DFN-8	θ_{JA}	—	60.5	—	°C/W	

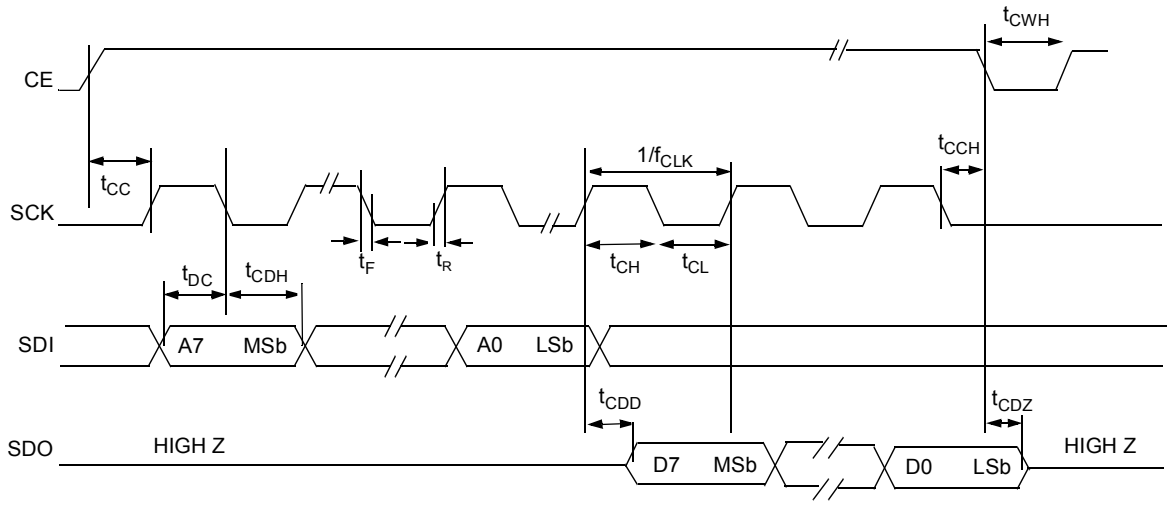
Note 1: The TC72-2.8MXX, TC72-3.3MXX and TC72-5.0MXX will operate from a supply voltage of 2.65V to 5.5V. However, the TC72-2.8MXX, TC72-3.3MXX and TC72-5.0MXX are tested and specified at the nominal operating voltages of 2.8V, 3.3V and 5.0V respectively. As V_{DD} varies from the nominal operating value, the accuracy may be degraded. Refer to Figure 2-5 and Figure 2-6.

2: Measured with a load of C_L = 50 pF on the SDO output pin of the TC72.

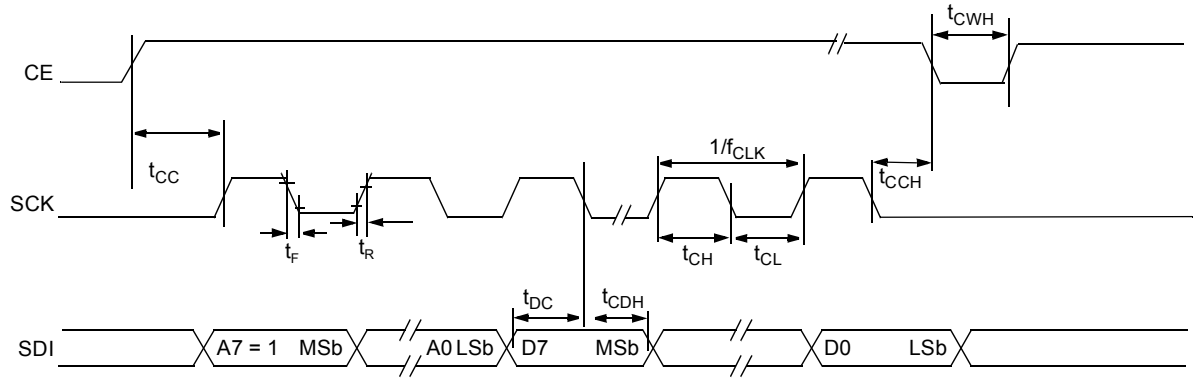
3: All time measurements are measured with respect to the 50% point of the signal, except for the SCK rise and fall times. The rise and fall times are defined as the 10% to 90% transition time.

4: Resolution = Temperature Range/No. of Bits = (+127°C – -128°C) / (2^{10}) = 256/1024 = 0.25°C/Bit

SPI READ DATA TRANSFER
(CP = 0, data shifted on rising edge of SCK, data clocked on falling edge of SCK, A7 = 0)



SPI WRITE DATA TRANSFER
(CP = 0, data shifted on rising edge of SCK, data clocked on falling edge of SCK, A7 = 1)



Note: The timing diagram is drawn with CP = 0. The TC72 also functions with CP = 1; however, the edges of SCK are reversed as defined in Table 3-3 and Figure 3-2.

FIGURE 1-1: *Serial Port Timing Diagrams.*

2.0 TYPICAL PERFORMANCE CURVES

Note:	The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore outside the warranted range.

Note: Unless otherwise indicated, all parameters apply at V_{DD} = 2.65V to 5.5V, TA = -55°C to +125°C.

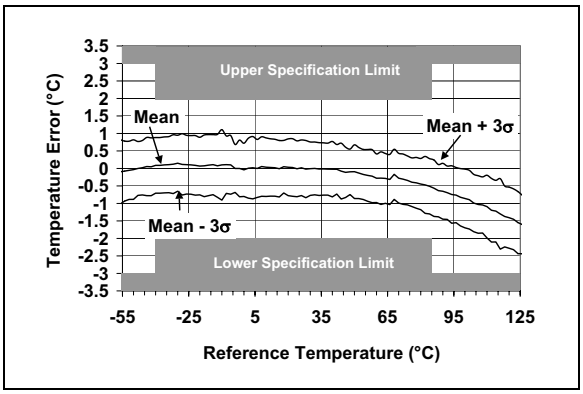


FIGURE 2-1: *Accuracy vs. Temperature (TC72-X.XMXX).*

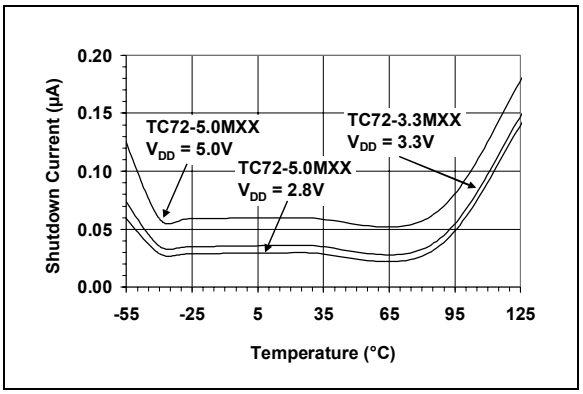


FIGURE 2-4: *Shutdown Current vs. Temperature.*

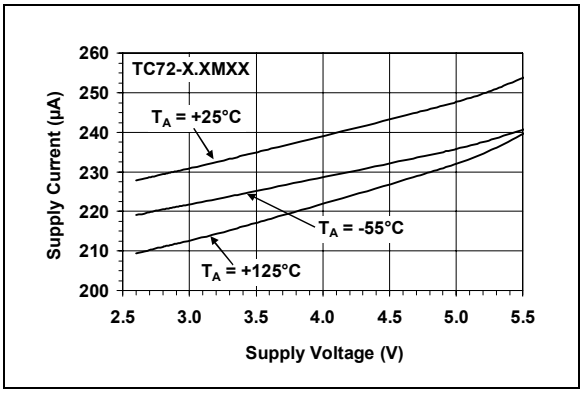


FIGURE 2-2: *Supply Current vs. Supply Voltage.*

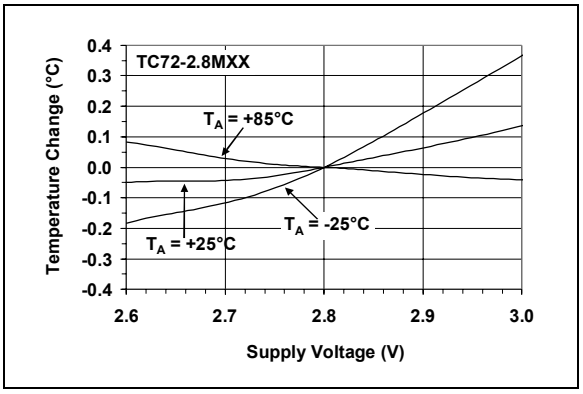


FIGURE 2-5: *Temperature Accuracy vs. Supply Voltage (TC72-2.8MXX).*

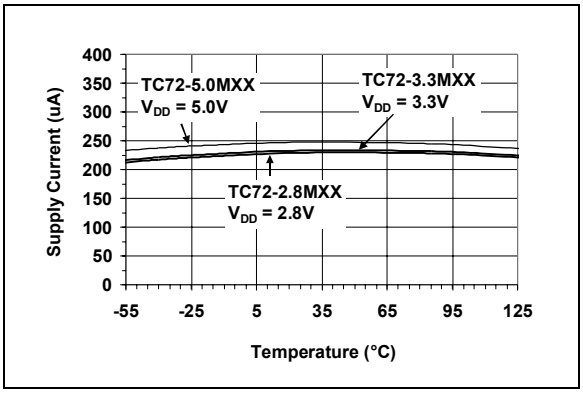


FIGURE 2-3: *Supply Current vs. Temperature.*

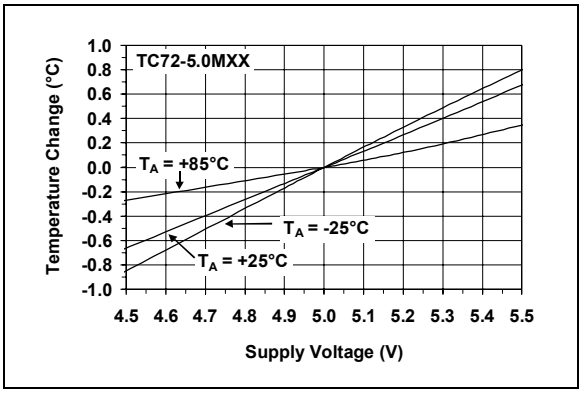


FIGURE 2-6: *Temperature Accuracy vs. Supply Voltage (TC72-5.0MXX).*

Note: Unless otherwise indicated, all parameters apply at V_{DD} = 2.65V to 5.5V, TA = -55°C to +125°C.

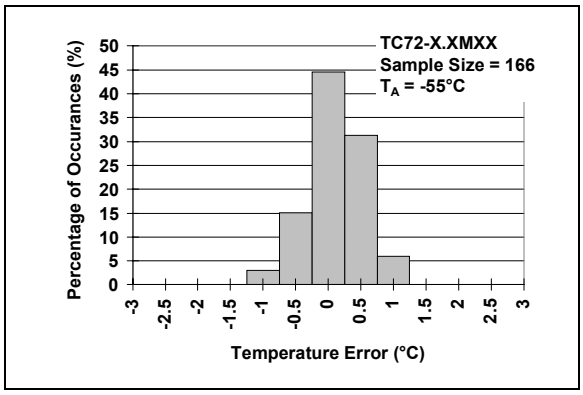


FIGURE 2-7: *Histogram of Temperature Accuracy at -55 Degrees C.*

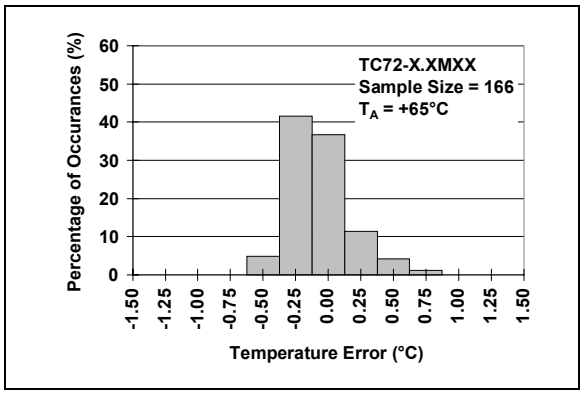


FIGURE 2-10: *Histogram of Temperature Accuracy at +65 Degrees C.*

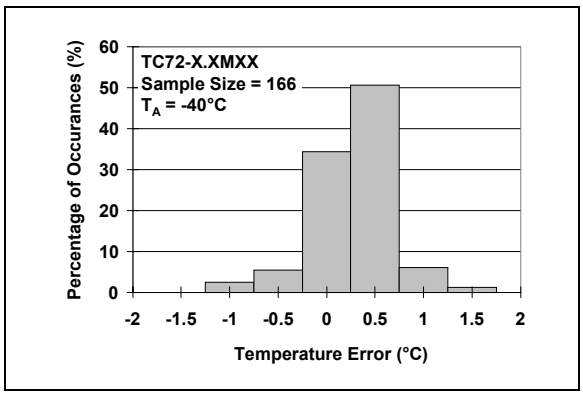


FIGURE 2-8: *Histogram of Temperature Accuracy at -40 Degrees C.*

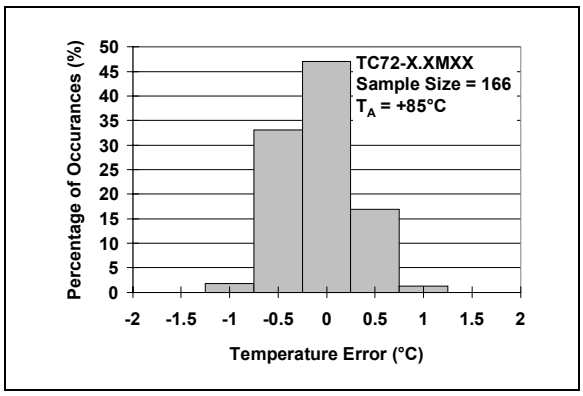


FIGURE 2-11: *Histogram of Temperature Accuracy at +85 Degrees C.*

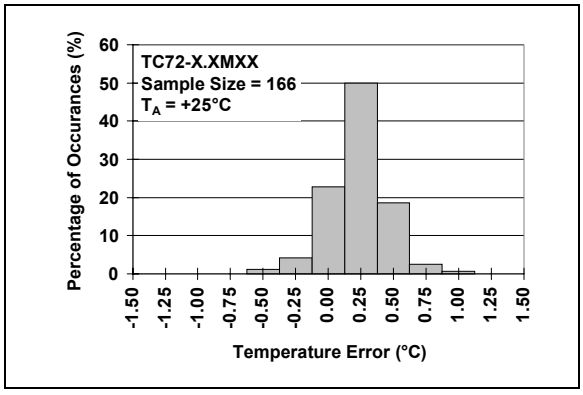


FIGURE 2-9: *Histogram of Temperature Accuracy at +25 Degrees C.*

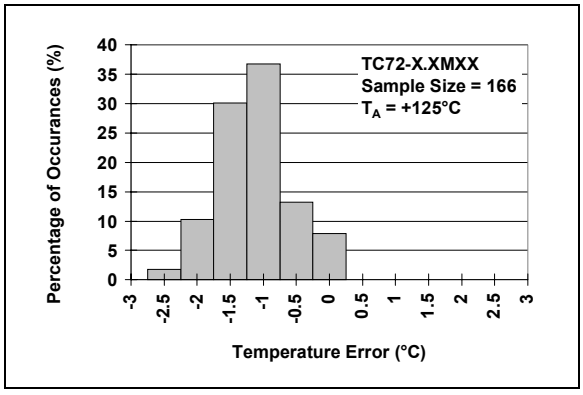


FIGURE 2-12: *Histogram of Temperature Accuracy at +125 Degrees C.*

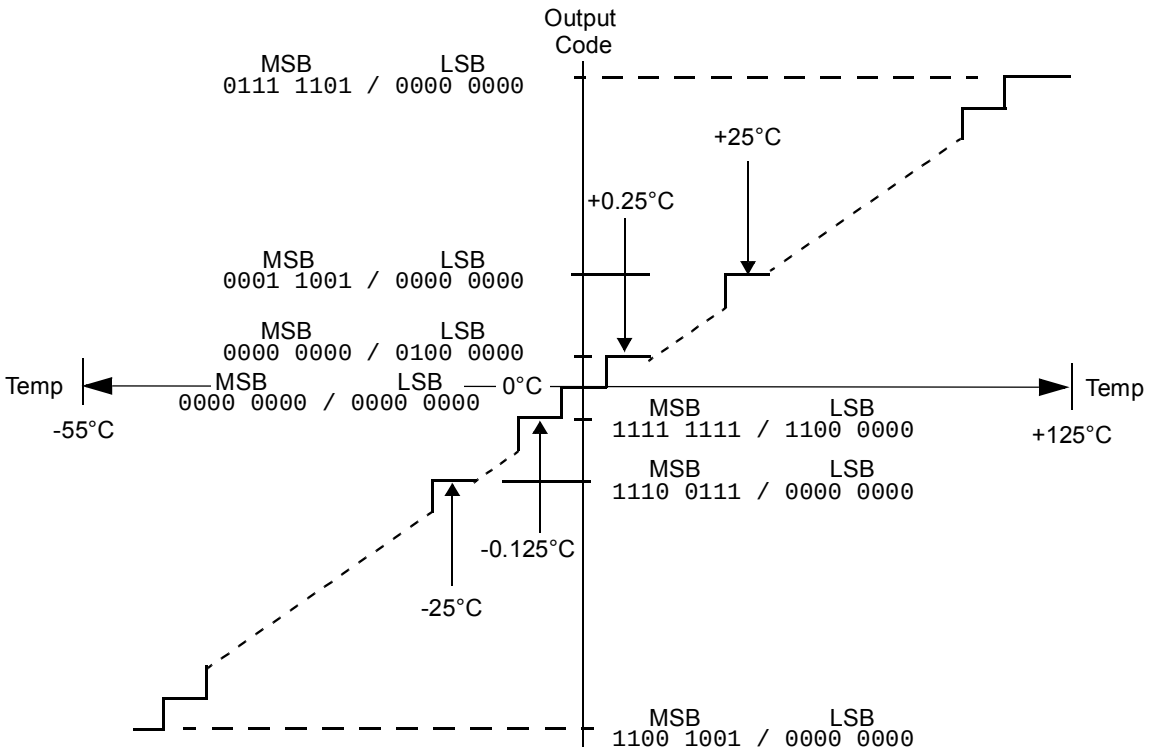
3.0 FUNCTIONAL DESCRIPTION

The TC72 consists of a band-gap type temperature sensor, a 10-bit Sigma Delta Analog-to-Digital Converter (ADC), an internal conversion oscillator and a double buffer digital output port. The 10-bit ADC is scaled from -128°C to +127°C; therefore, the resolution is 0.25°C per bit. The ambient temperature operating range of the TC72 is specified from -55°C to +125°C.

This device features a four-wire serial interface that is fully compatible with the SPI specification and, therefore, allows simple communications with common microcontrollers and processors. The TC72 can be used either in a Continuous Temperature Conversion mode or a One-Shot Conversion mode. The TC72 temperature measurements are performed in the background and, therefore, reading the temperature via the serial I/O lines does not affect the measurement in progress.

The Continuous Conversion mode measures the temperature approximately every 150 ms and stores the data in the temperature registers. The TC72 has an internal clock generator that controls the automatic temperature conversion sequence. The automatic temperature sampling operation is repeated indefinitely until the TC72 is placed in a shutdown mode by a write operation to the Control register. The TC72 will remain in the shutdown mode until the shutdown bit in the Control register is reset.

In contrast, the One-Shot mode performs a single temperature measurement and returns to the power-saving shut down mode. This mode is especially useful for low power applications.



Note: The ADC converter is scaled from -128°C to -127°C, but the operating range of the TC72 is specified from -55°C to +125°C.

FIGURE 3-1: *Temperature-To-Digital Transfer Function (Non-linear Scale).*

3.1 Temperature Data Format

Temperature data is represented by a 10-bit two's complement word with a resolution of 0.25°C per bit. The temperature data is stored in the Temperature registers in a two's complement format. The ADC converter is scaled from -128°C to +127°C, but the operating range of the TC72 is specified from -55°C to +125°C.

Example:

Temperature = 41.5°C

MSB Temperature Register= 00101001b
= $2^5 + 2^3 + 2^0$
= 32 + 8 + 1 = 41

LSB Temperature Register = 10000000b = 2^{-1} = 0.5

TABLE 3-1: TC72 TEMPERATURE OUTPUT DATA

Temperature	Binary MSB / LSB	Hex
+125°C	0111 1101/0000 0000	7D00
+25°C	0001 1001/0000 0000	1900
+0.5°C	0000 0000/1000 0000	0080
+0.25°C	0000 0000/0100 0000	0040
0°C	0000 0000/0000 0000	0000
-0.25°C	1111 1111/1100 0000	FFC0
-25°C	1110 0111/0000 0000	E700
-55°C	1100 1001/0000 0000	C900

TABLE 3-2: TEMPERATURE REGISTER

D7	D6	D5	D4	D3	D2	D1	D0	Address/ Register
Sign	2^6	2^5	2^4	2^3	2^3	2^1	2^0	02H Temp. MSB
2^{-1}	2^{-2}	0	0	0	0	0	0	01H Temp. LSB

3.2 Power-Up And Power-Down

The TC72 is in the low power consumption shutdown mode at power-up. The Continuous Temperature Conversion mode is selected by performing a Write operation to the Control register, as described in Section 4.0, "Internal Register Structure".

A supply voltage lower than 1.6V (typical) is considered a power-down state for the TC72. If the supply voltage drops below the 1.6V threshold, the internal registers are reset to the power-up default state.

3.3 Serial Bus Interface

The serial interface consists of the Chip Enable (CE), Serial Clock (SCK), Serial Data Input (SDI) and Serial Data Output (SDO) signals. The TC72 operates as a slave and is compatible with the SPI bus specifications. The serial interface is designed to be compatible with the Microchip PICmicro® family of microcontrollers.

The CE input is used to select the TC72 when multiple devices are connected to the serial clock and data lines. The CE is active-high, and data is written to or read from the device, when CE is equal to a logic high voltage. The SCK input is disabled when CE is low. The rising edge of the CE line initiates a read or write operation, while the falling edge of CE completes a read or write operation.

The SCK input is provided by the external microcontroller and is used to synchronize the data on the SDI and SDO lines. The SDI input writes data into the TC72's Control register, while the SDO outputs the temperature data from the Temperature register and the status of Shutdown bit of the Control register.

The TC72 has the capability to function with either an active-high or low SCK input. The SCK inactive state is detected when the CE signal goes high, while the polarity of the clock input (CP) determines whether the data is clocked and shifted on either the rising or falling edge of the system clock, as shown in Figure 3-2. Table 3-3 gives the appropriate clock edge used to transfer data into and out of the registers. Each data bit is transferred at each clock pulse, and the data bits are clocked in groups of eight bits, as shown in Figure 3-3.

The address byte is transferred first, followed by the data. A7, the MSb of the address, determines whether a read or write operation will occur. If A7 = '0', one or more read cycles will occur; otherwise, if A7 = '1', one or more write cycles will occur.

Data can be transferred either in a single byte or a multi-byte packet, as shown in Figure 3-3. In the 3-byte packet, the data sequence consists of the MSb temperature data, LSb temperature data, followed by the Control register data. The multi-byte read feature is initiated by writing the highest address of the desired packet to registers. The TC72 will automatically send the register addressed and all of the lower address registers, as long as the Chip Enable pin is held active.

TABLE 3-3: OPERATIONAL MODES

Mode	CE	SCK (Note 1)	SDI	SDO
Disable	L	Input Disabled	Input Disabled	High Z
Write (A7 = 1)	H	CP=1, Data Shifted on Falling Edge, Data Clocked on Rising Edge	Data Bit Latch	High Z
		CP=0, Data Shifted on Rising Edge, Data Clocked on Falling Edge		
Read (A7 = 0)	H	CP=1, Data Shifted on Falling Edge, Data Clocked on Rising Edge	X	Next data bit shift, **Note 2**
		CP=0, Data Shifted on Rising Edge, Data Clocked on Falling Edge		

Note 1: CP is the Clock Polarity of the microcontroller system clock. If the inactive state of SCK is logic level high, CP is equal to '1'; otherwise, if the inactive state of SCK is low, CP is equal to '0'.

2: During a Read operation, SDO remains at a high impedance (High Z) level until the eight bits of data begin to be shifted out of the Temperature register.

3.4 Read Operation

The temperature and control register data is outputted from the TC72 using the CE, SCK and SDO lines. Figure 3-3 shows a timing diagram of the read operation. Communication is initiated by the chip enable (CE) going high. The SDO line remains at the voltage level of the LSb bit that is outputted and goes to the tri-state level when the CE line goes to a logic low level.

3.5 Write Operation

Data is clocked into the Control register in order to enable the TC72's power saving shutdown mode. The write operation is shown in Figure 3-3 and is accomplished using the CE, SCK and SDI line.

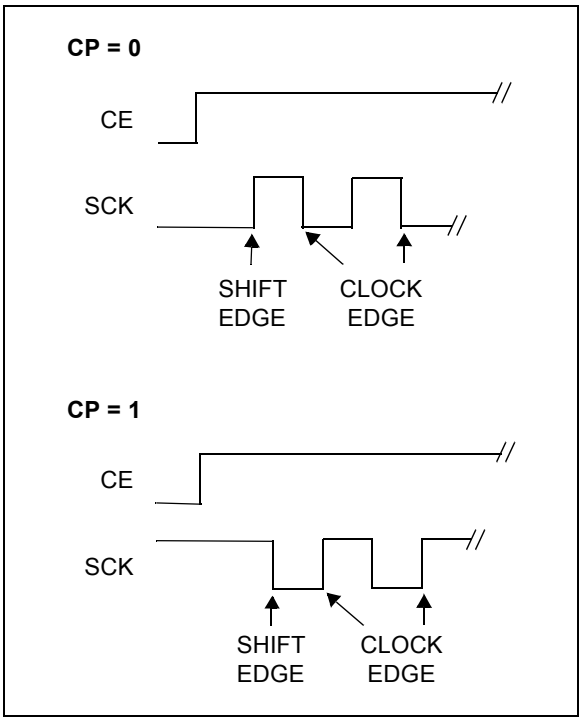


FIGURE 3-2: *Serial Clock Polarity (CP) Operation.*

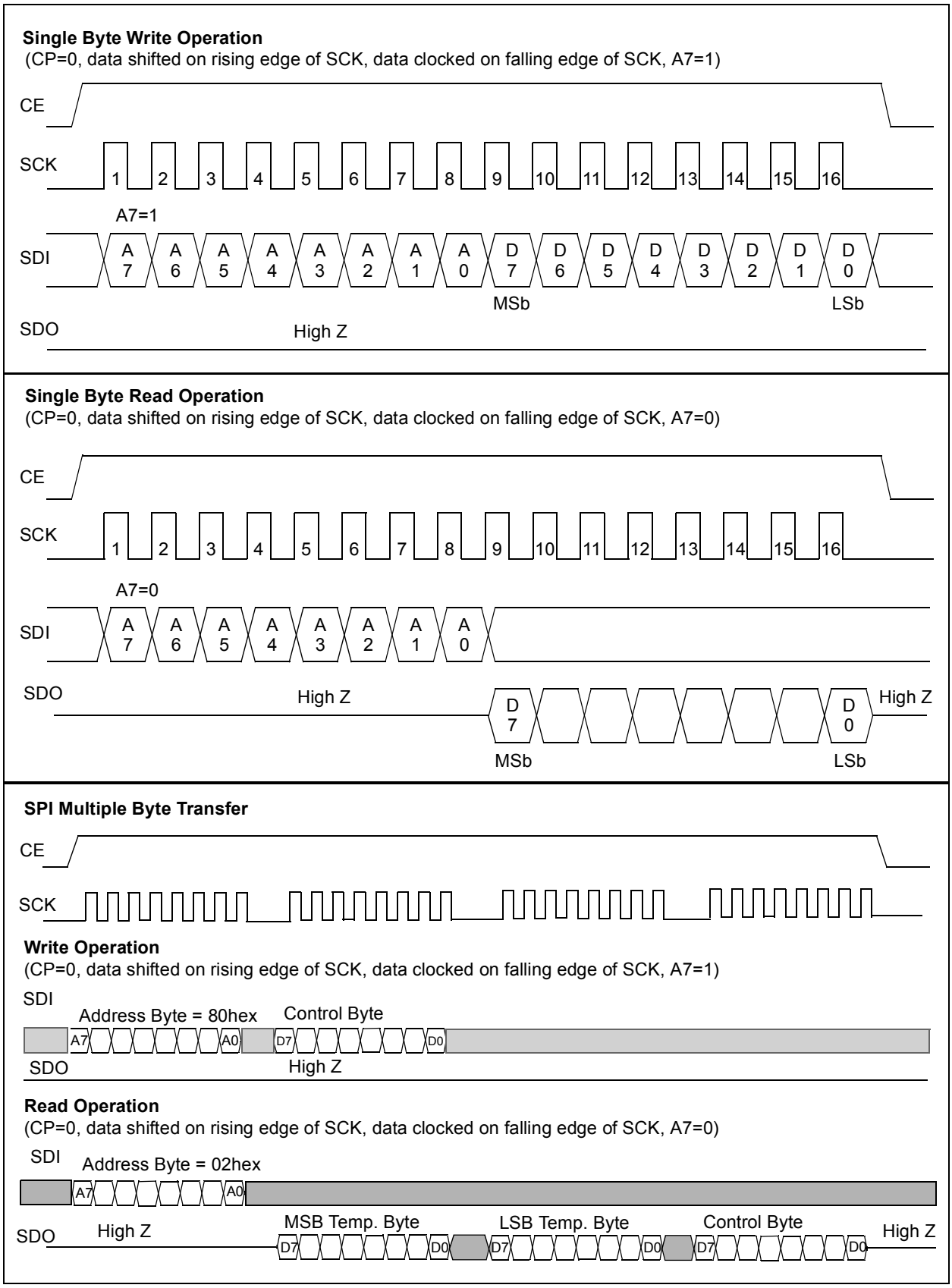


FIGURE 3-3: *Serial Interface Timing Diagrams (CP=0).*

4.0 INTERNAL REGISTER STRUCTURE

The TC72 registers are listed below.

TABLE 4-1: REGISTERS FOR TC72

Register	Read Address	Write Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR/BOR
Control	00hex	80hex	0	0	0	One-Shot (OS)	0	1	0	Shutdown (SHDN)	05hex
LSB Temperature	01hex	N/A	T1	T0	0	0	0	0	0	0	00hex
MSB Temperature	02hex	N/A	T9	T8	T7	T6	T5	T4	T3	T2	00hex
Manufacturer ID	03hex	N/A	0	1	0	1	0	1	0	0	54hex

4.1 Control Register

The Control register is both a read and a write register that is used to select either the Shutdown, Continuous or One-Shot Conversion operating mode. The Temperature Conversion mode selection logic is shown in Table 4-2. The Shutdown (SHDN) bit is stored in bit 0 of the Control register. If SHDN is equal to '1', the TC72 will go into the power-saving shutdown mode. If SHDN is equal to '0', the TC72 will perform a temperature conversion approximately every 150 ms.

At power-up, the SHDN bit is set to '1'. Thus, the TC72 is in the shutdown operating mode at startup. The Continuous Temperature Conversion mode is selected by writing a '0' to the SHDN bit of the Control register.

The Shutdown mode can be used to minimize the power consumption of the TC72 when active temperature monitoring is not required. The shutdown mode disables the temperature conversion circuitry; however, the serial I/O communication port remains active. A temperature conversion will be initialized by a Write operation to the Control register to select either the Continuous Temperature Conversion or the One-Shot operating mode. The temperature data will be available in the MSB and LSB Temperature registers approximately 150 ms after the Control register Write operation.

The One-Shot mode is selected by writing a '1' into bit 4 of the Control register. The One-Shot mode performs a single temperature measurement and returns to the power-saving shutdown mode. After completion of the temperature conversion, the One-Shot bit (OS) is reset to '0' (i.e. "OFF"). The user must set the One-Shot bit to '1' to initiate another temperature conversion.

Bits 1, 3, 5, 6 and 7 of the Control register are not used by the TC72. Bit 2 is set to a logic '1'. Any write operation to these bit locations will have no affect on the operation of the TC72.

4.2 Temperature Register

The Temperature register is a read-only register and contains a 10-bit two's complement representation of the temperature measurement. Bit 0 through Bit 5 of the LSB Temperature register are always set to a logic '0'.

At Power-On Reset (POR) or a Brown-Out Reset (BOR) low voltage occurrence, the temperature register is reset to all zeroes, which corresponds to a temperature value of 0°C. A V_{DD} power supply less than 1.6V is considered a reset event and will reset the Temperature register to the power-up state.

4.3 Manufacturer ID Register

The Manufacturer Identification (ID) register is a read-only register used to identify the temperature sensor as a Microchip component.

TABLE 4-2: CONTROL REGISTER TEMPERATURE CONVERSION MODE SELECTION

Operational Mode	One-Shot (OS) Bit 4	Shutdown (SHDN) Bit 0
Continuous Temperature Conversion	0	0
Shutdown	0	1
Continuous Temperature Conversion (One-Shot Command is ignored if SHDN = '0')	1	0
One-Shot	1	1

5.0 APPLICATIONS INFORMATION

The TC72 does not require any additional components in order to measure temperature; however, it is recommended that a decoupling capacitor of 0.1mF to 1mF be provided between the V_{DD} and GND pins. Although the current consumption of the TC72 is modest (250 mA, typical), the TC72 contains an on chip data acquisition with internal digital switching circuitry. Thus, it is considered good design practice to use an external decoupling capacitor with the sensor. A high frequency ceramic capacitor should be used and be located as close as possible to the IC power pins in order to provide effective noise protection to the TC72.

The TC72 measures temperature by monitoring the voltage of a diode located on the IC die. The IC pins of the TC72 provide a low impedance thermal path between the die and the PCB, allowing the TC72 to effectively monitor the temperature of the PCB board. The thermal path between the ambient air is not as efficient because the plastic IC housing package functions as a thermal insulator. Thus the ambient air temperature (assuming that a large temperature gradient exists between the air and PCB) has only a small effect on the temperature measured by the TC72.

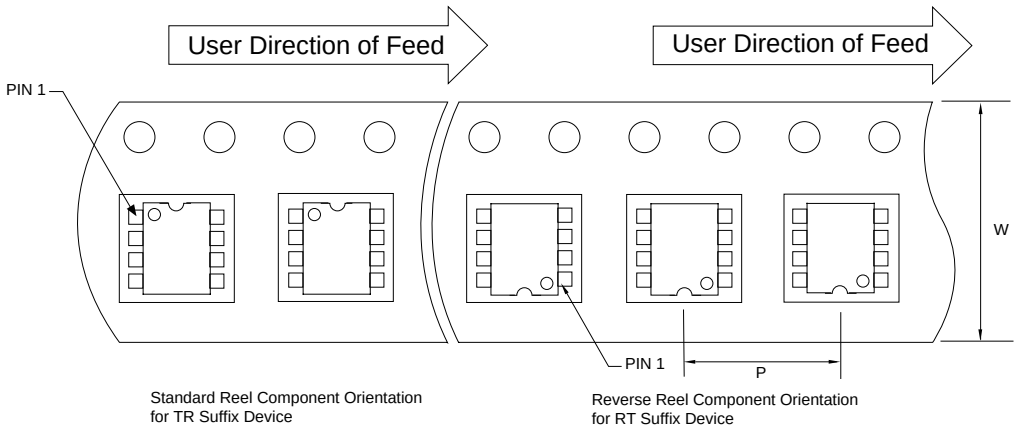
Note that the exposed metal center pad on the bottom of the DFN package is connected to the silicon substrate. The center pad should be connected to either the PCB ground plane or treated as a “No Connect” pin. The mechanical dimensions of the center pad are given in Section 6.0, “Packaging Information”, of this datasheet.

A potential for self-heating errors can exist if the TC72 SPI communication lines are heavily loaded. Typically, the self-heating error is negligible because of the relatively small current consumption of the TC72. A temperature accuracy error of approximately 0.5°C will result from self-heating if the SPI communication pins sink/source the maximum current specified for the TC72. Thus to maximize the temperature accuracy, the output loading of the SPI signals should be minimized.

6.0 PACKAGING INFORMATION

6.1 Taping Form

Component Taping Orientation for 8-Pin MSOP Devices



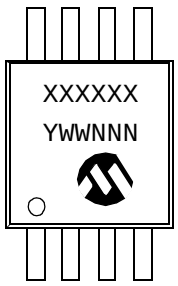
Carrier Tape, Number of Components Per Reel and Reel Size

Package	Carrier Width (W)	Pitch (P)	Part Per Full Reel	Reel Size
8-Pin MSOP	12 mm	8 mm	2500	13 in

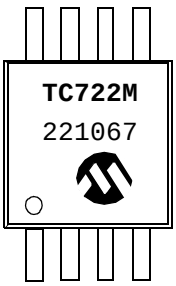
Tape and Reel information for the 8-Lead DFN package will be available TBD.

6.2 Package Marking Information

8-Lead MSOP



Example:



8-Lead DFN

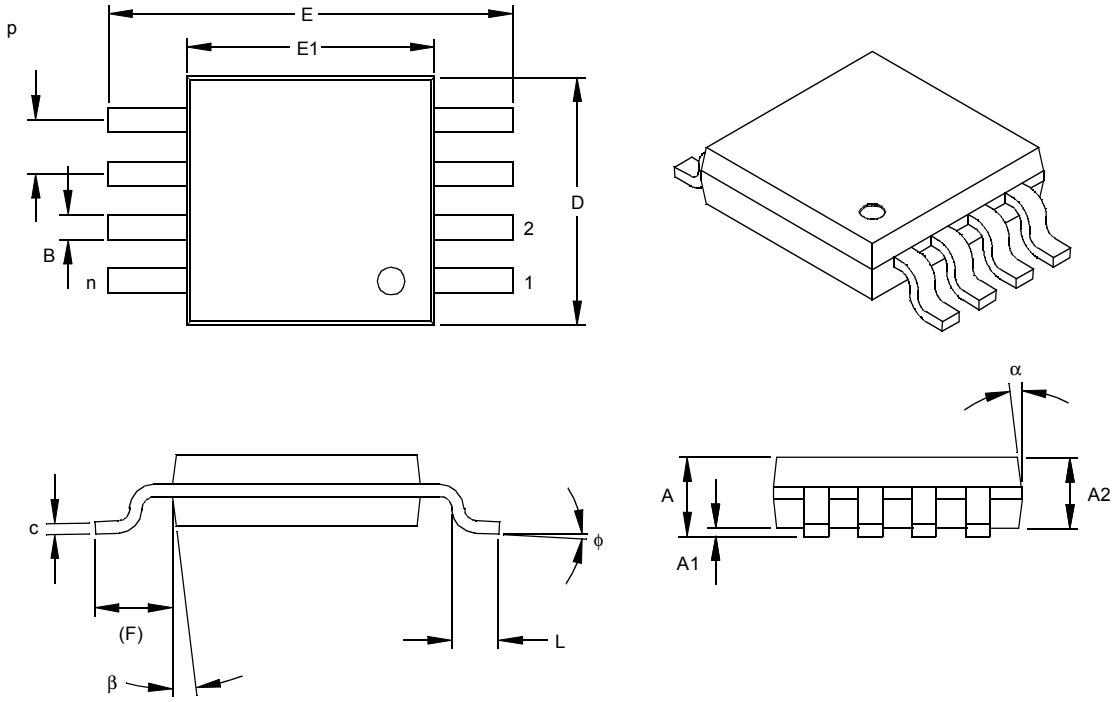
Example:

Legend:	XX...X	Customer specific information*
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line thus limiting the number of available characters for customer specific information.

***** Standard OTP marking consists of Microchip part number, year code, week code, and traceability code.

8-Lead Plastic Micro Small Outline Package (MS) (MSOP)



Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		8				8
Pitch	p		.026			0.65	
Overall Height	A			.044			1.18
Molded Package Thickness	A2	.030	.034	.038	0.76	0.86	0.97
Standoff §	A1	.002		.006	0.05		0.15
Overall Width	E	.184	.193	.200	4.67	4.90	.5.08
Molded Package Width	E1	.114	.118	.122	2.90	3.00	3.10
Overall Length	D	.114	.118	.122	2.90	3.00	3.10
Foot Length	L	.016	.022	.028	0.40	0.55	0.70
Footprint (Reference)	F	.035	.037	.039	0.90	0.95	1.00
Foot Angle	φ	0		6	0		6
Lead Thickness	c	.004	.006	.008	0.10	0.15	0.20
Lead Width	B	.010	.012	.016	0.25	0.30	0.40
Mold Draft Angle Top	α		7			7	
Mold Draft Angle Bottom	β		7			7	

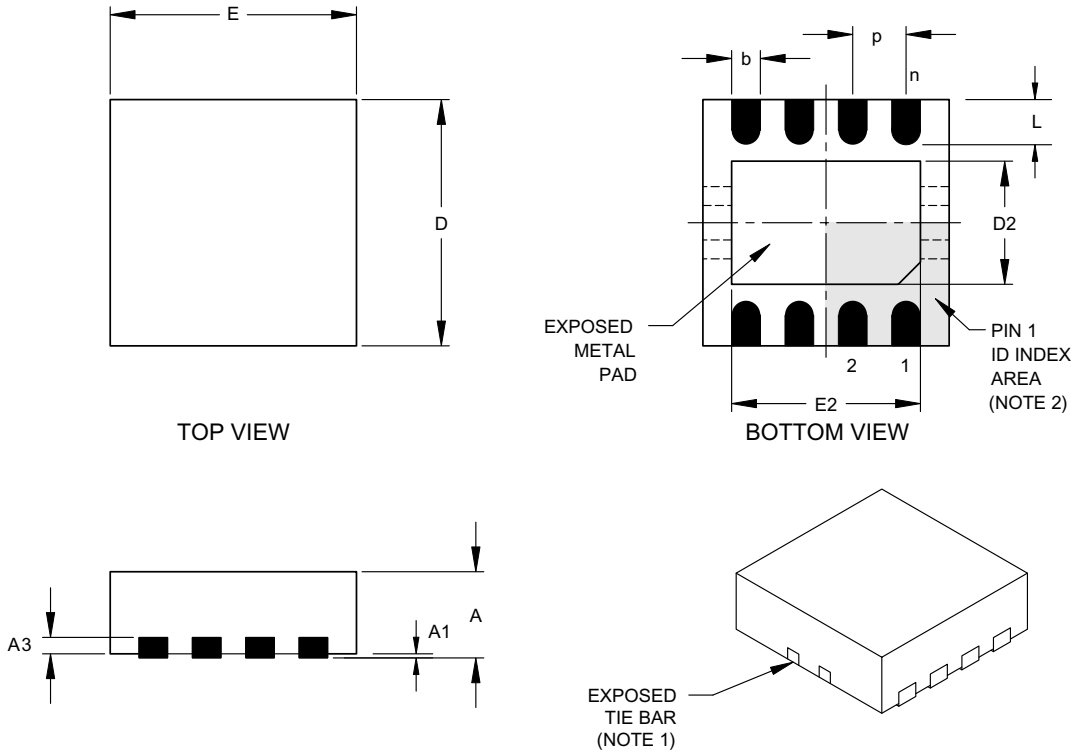
*Controlling Parameter
§ Significant Characteristic

Notes:

Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.

Drawing No. C04-111

8-Lead Plastic Dual Flat Pack, No Lead (MF) 3x3x1 mm Body (DFN)



Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		8			8	
Pitch	p		.026 BSC			0.65 BSC	
Overall Height	A	.031	.035	.039	0.80	0.90	1.00
Standoff	A1	.000	.001	.002	0.00	0.02	0.05
Lead Thickness	A3		.008 REF.			0.20 REF.	
Overall Length	E		.118 BSC			3.00 BSC	
Exposed Pad Length (Note 4)	E2	.055		.096	1.39		2.45
Overall Width	D		.118 BSC			3.00 BSC	
Exposed Pad Width (Note 4)	D2	.047		.069	1.20		1.75
Lead Width	b	.007	.010	.015	0.23	0.26	0.37
Lead Length	L	.012	.019	.022	0.30	0.48	0.55

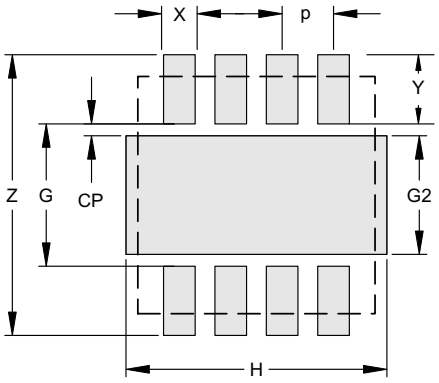
*Controlling Parameter

Notes:

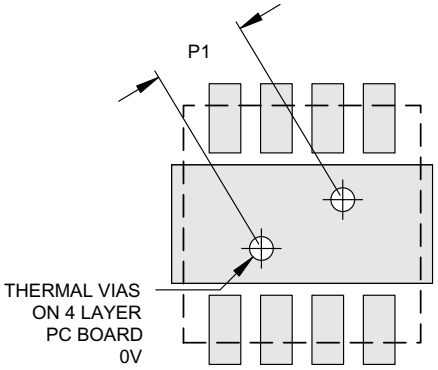
1. Package may have one or more exposed tie bars at ends.
2. Pin 1 visual index feature may vary, but must be located within the hatched area.
3. Dimensions D and E do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
4. Exposed pad dimensions vary with paddle size.
5. JEDEC equivalent: Pending

Drawing No. C04-062

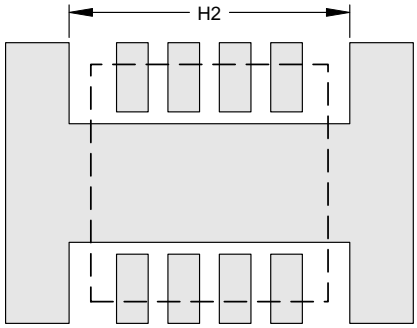
8-Lead Plastic Dual Flat Pack, No Lead (MF) 3x3x1 mm Body (DFN)



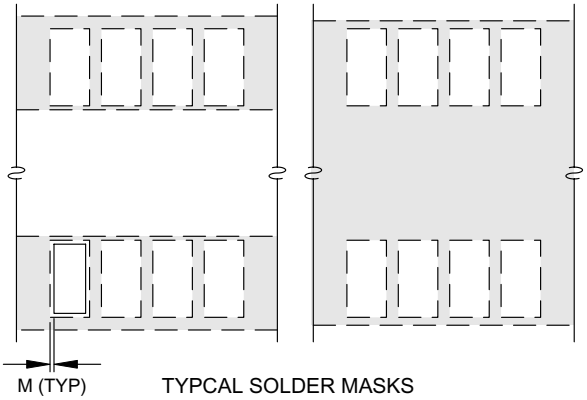
DFN LAND PATTERN



OPTIONAL FOUR LAYER
PCB LAND PATTERN



OPTIONAL TWO LAYER
PCB LAND PATTERN



TYPCAL SOLDER MASKS

Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Terminal Pitch	p		.026 BSC			0.65 BSC	
Terminal Land Pattern O.D.	Z	.134		.157	3.40		4.00
Terminal Land Pattern I.D.	A1	.057		.060	1.45		1.53
Exposed Pad Clearance	CP	.006			0.15		
Interior Lead Clearance	Z	.071			1.80		
Terminal Land Width	X	.014		.017	0.35		0.42
Terminal Land Length	Y	.033		.035	0.85		0.88
Exposed Pad Length	H	.130			3.30		
Optional Exposed Pad Length	H2	.130			3.30		
Exposed Pad Width (Note 1)	G2	.057		.059	1.45		1.50
Termal Via Pitch	P1		.047			1.20	
Thermal Via Diameter	V		.012			0.30	
Minimum Solder Mask Clearance	M	.002			0.05		

*Controlling Parameter

Notes:

1. Exposed pad dimensions vary with paddle size.

Drawing No. C04-2062

ON-LINE SUPPORT

Microchip provides on-line support on the Microchip World Wide Web site.

The web site is used by Microchip as a means to make files and information easily available to customers. To view the site, the user must have access to the Internet and a web browser, such as Netscape® or Microsoft® Internet Explorer. Files are also available for FTP download from our FTP site.

Connecting to the Microchip Internet Web Site

The Microchip web site is available at the following URL:

www.microchip.com

The file transfer site is available by using an FTP service to connect to:

ftp://ftp.microchip.com

The web site and file transfer site provide a variety of services. Users may download files for the latest Development Tools, Data Sheets, Application Notes, User's Guides, Articles and Sample Programs. A variety of Microchip specific business information is also available, including listings of Microchip sales offices, distributors and factory representatives. Other data available for consideration is:

- Latest Microchip Press Releases
- Technical Support Section with Frequently Asked Questions
- Design Tips
- Device Errata
- Job Postings
- Microchip Consultant Program Member Listing
- Links to other useful web sites related to Microchip Products
- Conferences for products, Development Systems, technical information and more
- Listing of seminars and events

SYSTEMS INFORMATION AND UPGRADE HOT LINE

The Systems Information and Upgrade Line provides system users a listing of the latest versions of all of Microchip's development systems software products. Plus, this line provides information on how customers can receive the most current upgrade kits.The Hot Line Numbers are:

1-800-755-2345 for U.S. and most of Canada, and

1-480-792-7302 for the rest of the world.

092002

READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this document.

To: Technical Publications Manager Total Pages Sent _____

RE: Reader Response

From: Name _____

Company _____

Address _____

City / State / ZIP / Country _____

Telephone: (_____) _____ - _____ FAX: (_____) _____ - _____

Application (optional):

Would you like a reply? ___Y ___N

Device: **TC72** Literature Number: **DS21734A**

Questions:

1. What are the best features of this document?

2. How does this document meet your hardware and software development needs?

3. Do you find the organization of this document easy to follow? If not, why?

4. What additions to the document do you think would enhance the structure and subject?

5. What deletions from the document could be made without affecting the overall usefulness?

6. Is there any incorrect or misleading information (what and where)?

7. How would you improve this document?

PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PART NO.	-X.X	X	XX
Device	Voltage Range	Temperature Range	Package

Device:	TC72:	Digital Temperature Sensor w/SPI Interface
Voltage Range:	2.8	= Accuracy Optimized for 2.8V
	3.3	= Accuracy Optimized for 3.3V
	5.0	= Accuracy Optimized for 5.0V
Temperature Range:	M	= -55°C to +125°C
Package:	MF	= Dual, Flat, No Lead (DFN) (3x3mm), 8-lead
	MFTR	= Dual, Flat, No Lead (DFN) (3x3mm), 8-lead (Tape and Reel)
	UA	= Plastic Micro Small Outline (MSOP), 8-lead
	UATR	= Plastic Micro Small Outline (MSOP), 8-lead (Tape and Reel)

Examples:

a) TC72-2.8MUA: Digital Temperature Sensor, 2.8V, 8LD MSOP package.

b) TC72-2.8MUATR: Digital Temperature Sensor, 2.8V, 8LD MSOP (tape and reel) package.

c) TC72-2.8MMF: Digital Temperature Sensor, 2.8V, 8LD DFN package.

d) TC72-3.3MUA: Digital Temperature Sensor, 3.3V, 8LD MSOP package.

e) TC72-3.3MMF: Digital Temperature Sensor, 3.3V, 8LD DFN package.

f) TC72-5.0MUA: Digital Temperature Sensor, 5.0V, 8LD MSOP package.

g) TC72-5.0MMF: Digital Temperature Sensor, 5.0V, 8LD DFN package.

h) TC72-5.0MMFTR: Digital Temperature Sensor, 5.0V, 8LD DFN (tape and reel) package.

Sales and Support

Data Sheets

Products supported by a preliminary Data Sheet may have an errata sheet describing minor operational differences and recommended workarounds. To determine if an errata sheet exists for a particular device, please contact one of the following:

1. Your local Microchip sales office
2. The Microchip Corporate Literature Center U.S. FAX: (480) 792-7277
3. The Microchip Worldwide Site (www.microchip.com)

Please specify which device, revision of silicon and Data Sheet (include Literature #) you are using.

New Customer Notification System

Register on our web site (www.microchip.com/cn) to receive the most current information on our products.

NOTES:

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, KEELOQ, MPLAB, PIC, PICmicro, PICSTART and PRO MATE are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, microID, MXDEV, MXLAB, PICMASTER, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

dsPIC, dsPICDEM.net, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, PICC, PICDEM, PICDEM.net, rfPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

Printed on recycled paper.

DNV Certification, Inc. USA
ANSI-RAB QMS ACCREDITED
DNV
DNV MSC The Netherlands Accredited by the RvA
ISO 9001 / QS-9000 REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, non-volatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-4338

Atlanta
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo
2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing
Microchip Technology Consulting (Shanghai) Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu
Microchip Technology Consulting (Shanghai) Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-86766200 Fax: 86-28-86766599

China - Fuzhou
Microchip Technology Consulting (Shanghai) Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai
Microchip Technology Consulting (Shanghai) Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen
Microchip Technology Consulting (Shanghai) Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

China - Hong Kong SAR
Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-6334-8870 Fax: 65-6334-8850

Taiwan
Microchip Technology (Barbados) Inc.,
Taiwan Branch
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Austria
Microchip Technology Austria GmbH
Durisolstrasse 2
A-4600 Wels
Austria
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

Denmark
Microchip Technology Nordic ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Microchip Technology GmbH
Steinheilstrasse 10
D-85737 Ismaning, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom
Microchip Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

08/01/02

TC72

带 SPI 接口的数字温度传感器

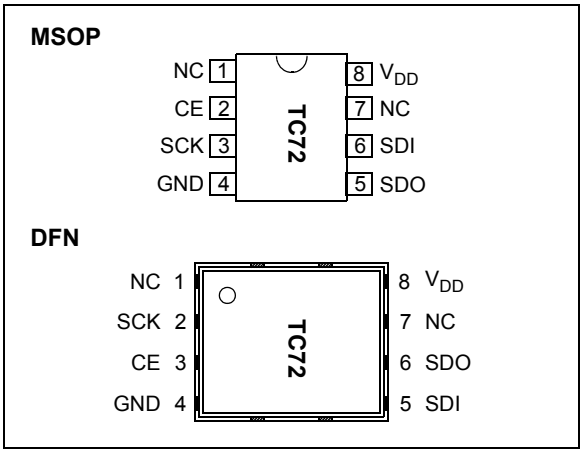
特性

- 温度 / 数字转换器
- SPI 兼容接口
- 10 位分辨率 （0.25°C/ 位）
- -40°C 至 +85°C 的精度为 ±2°C （最大值）
- -55°C 至 +125°C 的精度为 ±3°C （最大值）
- 2.65V 至 5.5V 工作电压范围
- 低功耗：
  - 连续温度转换模式下，消耗的电流为 250 µA （典型值）
  - 关断模式时，消耗的电流为 1 µA （最大值）
- 节能的单次温度测量
- 工业标准 8 引脚 MSOP 封装
- 节省空间的 8 引脚 DFN （3x3 mm）封装

典型应用

- 个人电脑和服务器
- 硬盘驱动器和其他 PC 外设
- 娱乐系统
- 办公设备
- 数据通信设备
- 移动电话
- 通用温度监测设备

封装类型



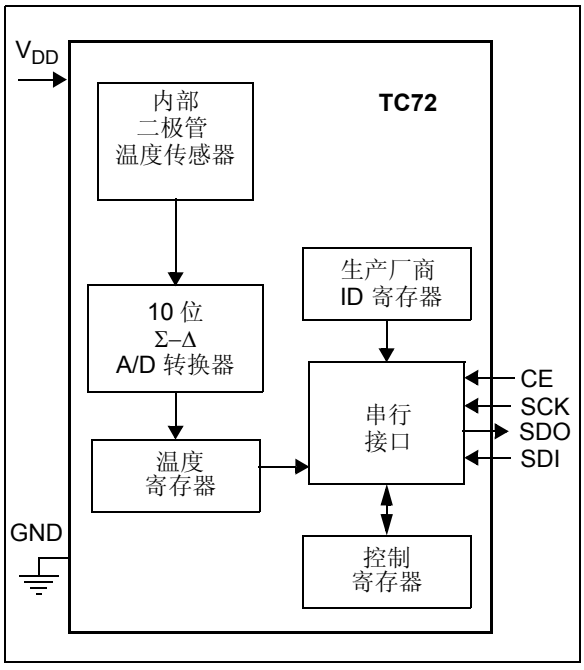
概述

TC72 数字温度传感器能够读取从 -55°C 至 +125°C 范围内的温度。该传感器具有的串行接口能够与主控制器或其他外设通信。TC72 接口兼容 SPI 协议。TC72 不需要额外的外部元器件。但是，推荐在 V_{DD} 和 GND 引脚间连接 0.01 µF 至 0.1 µF 的去耦电容。

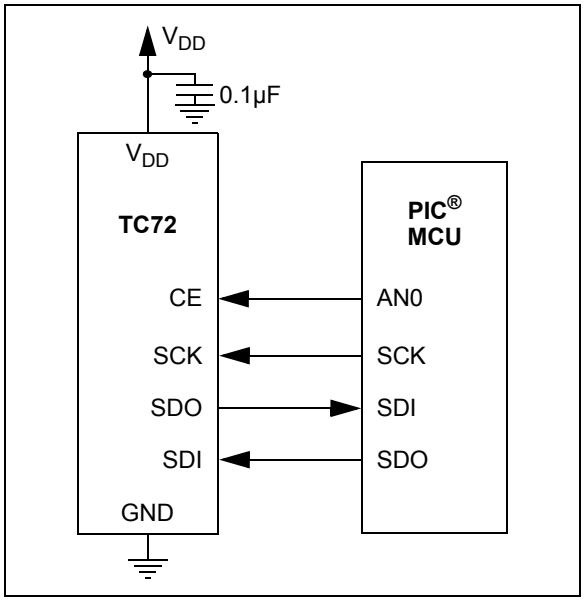
TC72 可使用于连续温度转换模式或单次转换模式。连续温度转换模式每隔约 150 ms 测量温度并将数据存储在温度寄存器中。相反，单次模式只进行单次温度测量，然后返回到节能的关断模式。

TC72 具有很高的温度精度，易于使用，是很多系统中温度管理的理想方案。器件提供 8 引脚 MSOP 和节省空间的 8 引脚 DFN 封装。TC72 还具有低功耗的关断模式。

框图



典型应用



1.0 电气特性

1.1 绝对最大值 †

V_{DD}.....6.0V
所有输入和输出引脚相对 GND 的电压
.....-0.3V 至 V_{DD} +0.3V
存储温度.....-65°C 至 +150°C
通电时的环境温度 .....-55°C 至 +125°C
结温.....150°C
所有引脚上的 ESD 保护：
人体模型 （HBM）.....> 4 kV
机器模型 （MM）.....> 400V
每个引脚的闭锁电流 .....±200 mA
最大功耗.....250 mW

† 注： 如果器件的工作条件超过 “绝对最大值”列出的范围，就可能会对器件造成永久性损坏。上述值仅为运行条件极大值，我们建议不要使器件在该规范规定的范围以外运行。器件长时间工作在最大额定值条件下，其稳定性会受到影响。

引脚功能表

名称	功能
NC	无内部连接
CE	芯片使能输入。当该输入为高电平时，选定器件。
SCK	串行时钟输入
GND	地
SDO	串行数据输出
SDI	串行数据输入
NC	无内部连接
V_{DD}	电源

直流特性

电气规范： 除非另有说明，否则所有参数均适用于 V_{DD} = 2.65V 至 5.5V 且 T_A = -55°C 至 +125°C 的条件。

参数	符号	最小值	典型值	最大值	单位	条件
供电电压						
工作电压范围	V_{DD}	2.65	—	5.5	V	**注 1**
工作电流： 正常模式，ADC 启动	I_{DD-CON}	—	250	400	µA	连续温度转换模式 （关断位 = “0”）
关断供电电流	I_{SHD}	—	0.1	1.0	µA	关断模式 （关断位 = “1”）
温度传感器和模拟 / 数字转换器						
温度精度 **（注 1）**	T_{ACY}	-2.0	—	+2.0	°C	-40°C < T_A < +85°C
		-3.0	—	+3.0		-55°C < T_A < +125°C
分辨率		—	10	—	位	**注 4**
ADC 转换时间	t_{CONV}	—	150	200	ms	
数字输入 / 输出						
高电平输入电压	V_{IH}	0.7 V_{DD}	—	—	V	
低电平输入电压	V_{IL}	—	—	0.2 V_{DD}	V	
高电平输出电压	V_{OH}	0.7 V_{DD}	—	—	V	I_{OH} = 1 mA
低高电平输出电压	V_{OL}	—	—	0.2 V_{DD}	V	I_{OL} = 4 mA
输入电阻	R_{IN}	1.0	—	—	MΩ	
引脚电容	C_{IN}	—	15	—	pF	
	C_{OUT}	—	50	—		

注 1： TC72-2.8MXX、 TC72-3.3MXX 和 TC72-5.0MXX 可工作在 2.65V 至 5.5V 的供电电压范围内。但是， TC72-2.8MXX、TC72-3.3MXX 和 TC72-5.0MXX 却是分别在 2.8V、3.3V 和 5.0V 的标称工作电压下测试和规定的。如果 V_{DD} 电压与标称工作电压不同，则精度可能会降低。请参见图 2-5 和图 2-6。

2： 在 TC72 的 SDO 输出引脚接 C_L = 50 pF 的负载时测得。

3： 除 SCK 上升和下降时间外，所有时间测量的值均以信号的中心点为基准。上升和下降时间定义为信号在满幅的 10% 至 90% 之间变化的时间。

4： 分辨率 = 温度范围 / 位数 = （+127°C – -128°C) / (2^{10}) = 256/1024 = 0.25°C/ 位

直流特性 （续）

电气规范：除非另有说明，否则所有参数均适用于 V_{DD} = 2.65V 至 5.5V 且 T_A = -55°C 至 +125°C 的条件。						
参数	**符号**	**最小值**	**典型值**	**最大值**	**单位**	**条件**
串行端口交流时序 （注 2 和 3）						
时钟频率	f_{CLK}	DC	—	7.5	MHz	
SCK 低电平时间	t_{CL}	65	—	—	ns	
SCK 高电平时间	t_{CH}	65	—	—	ns	
CE 至 SCK 建立时间	t_{CC}	400	—	—	ns	
SCK 至数据输出有效的时间	t_{CDD}	—	—	55	ns	
CE 至输出三态的时间	t_{CDZ}	—	—	40	ns	
SCK 至数据保持的时间	t_{CDH}	35	—	—	ns	
数据输出有效至 SCK 稳定的时间	t_{DC}	35	—	—	ns	
SCK 至 CE 保持的时间	t_{CCH}	100	—	—	ns	
SCK 上升时间	t_R	—	—	200	ns	
SCK 下降时间	t_F	—	—	200	ns	
CE 无效时间	t_{CWH}	400	—	—	ns	
封装热阻						
热阻，MSOP-8	θ_{JA}	—	206	—	°C/W	
热阻，DFN-8	θ_{JA}	—	60.5	—	°C/W	

注 **1：** TC72-2.8MXX、 TC72-3.3MXX 和 TC72-5.0MXX 可工作在 2.65V 至 5.5V 的供电电压范围内。但是， TC72-2.8MXX、TC72-3.3MXX 和 TC72-5.0MXX 却是分别在 2.8V、3.3V 和 5.0V 的标称工作电压下测试和规定的。如果 V_{DD} 电压与标称工作电压不同，则精度可能会降低。请参见图 2-5 和图 2-6。

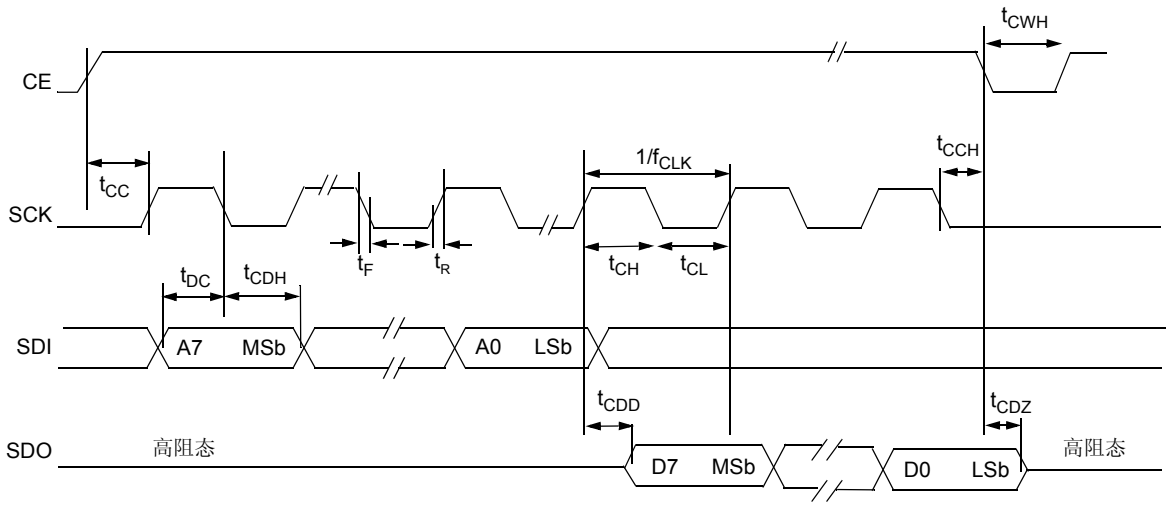
2： 在 TC72 的 SDO 输出引脚接 C_L = 50 pF 的负载时测得。

3： 除 SCK 上升和下降时间外，所有时间测量的值均以信号的中心点为基准。上升和下降时间定义为信号在满幅的 10% 至 90% 之间变化的时间。

4： 分辨率 = 温度范围 / 位数 = （+127°C – -128°C) / (2^{10}) = 256/1024 = 0.25°C/ 位

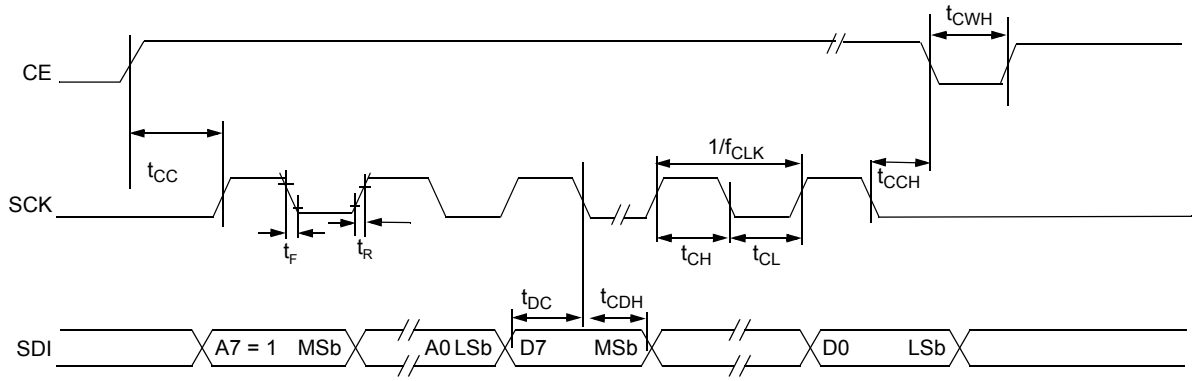
SPI 读数据传输

（CP = 0，数据在 SCK 的上升沿从传感器移出，在 SCK 的下降沿移入单片机，A7 = 0）



SPI 写数据传输

（CP = 0，数据在 SCK 的上升沿移出传感器，在 SCK 的下降沿移入单片机，A7 = 1）



注： 此时序图是在 CP = 0 时绘制的。TC72 也可在 CP = 1 时工作；但是，SCK 的边沿与表 3-3 和图 3-2 的定义相反。

图 1-1： *串行端口时序图*

2.0 典型性能曲线

注：	以下图表来自有限数量样本的统计结果，仅供参考。所列出的性能特性未经测试，不做任何担保。一些图表中列出的数据可能超出规定的工作范围 （例如，超出了规定的电源电压范围），因而不在担保范围内。

注：除非另有说明，否则所有参数均适用于 V_{DD} = 2.65V 至 5.5V 且 TA = -55°C 至 +125°C 的条件。

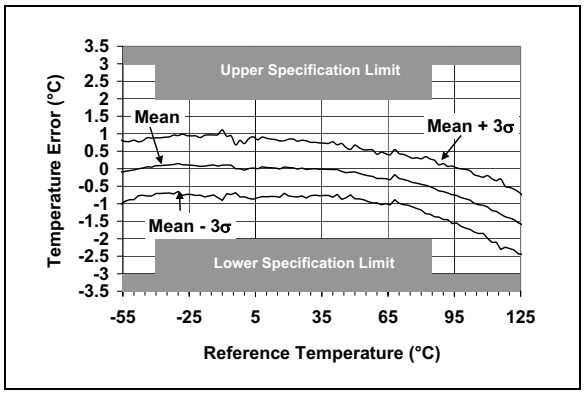


图 2-1： *精度——温度曲线 （TC72-X.XMXX）*

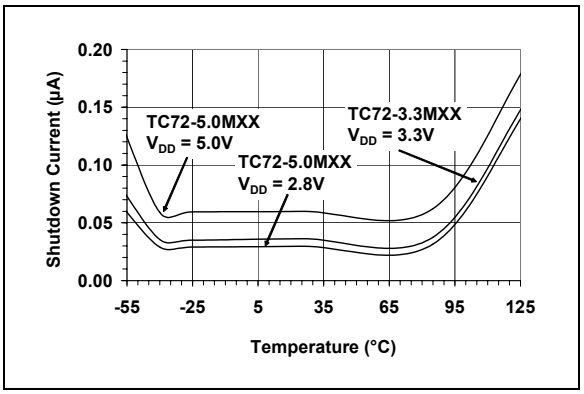


图 2-4： *关断电流——温度曲线*

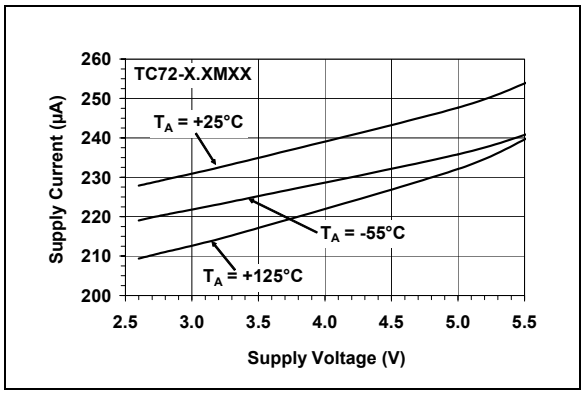


图 2-2： *供电电流——供电电压曲线*

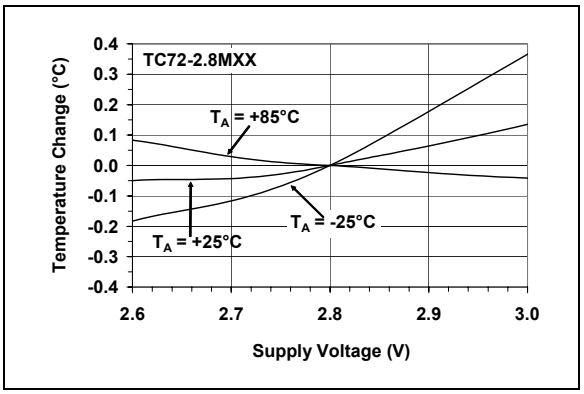


图 2-5： *温度精度——供电电压曲线 （TC72-2.8MXX）*

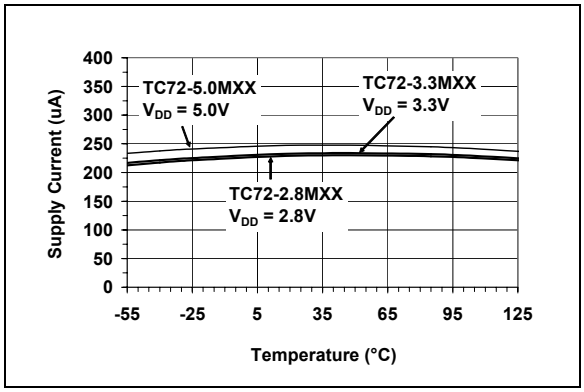


图 2-3： *供电电流——温度曲线*

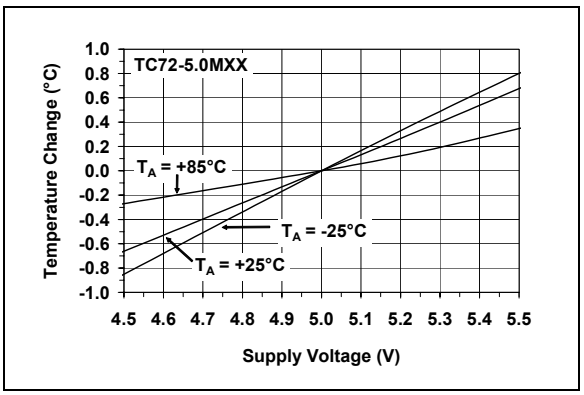


图 2-6： *温度精度——供电电压曲线 （TC72-5.0MXX）*

注：除非另有说明，否则所有参数均适用于 V_{DD} = 2.65V 至 5.5V 且 TA = -55°C 至 +125°C 的条件。

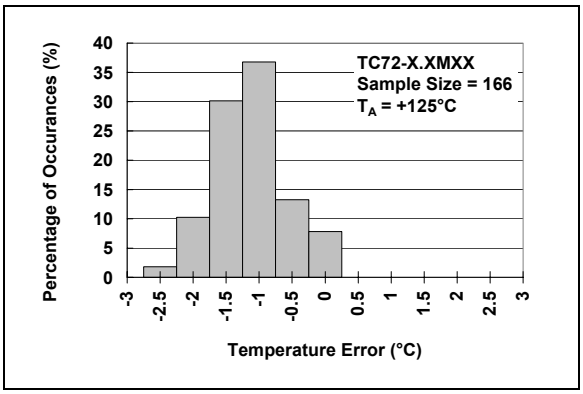


***图 2-7：** -55 摄氏度时温度精度柱状图*

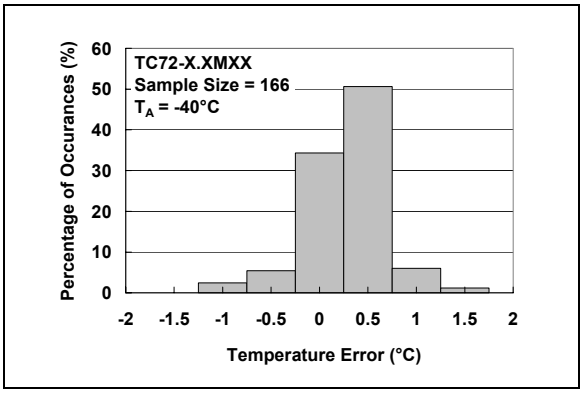


***图 2-8：** -40 摄氏度时温度精度柱状图*

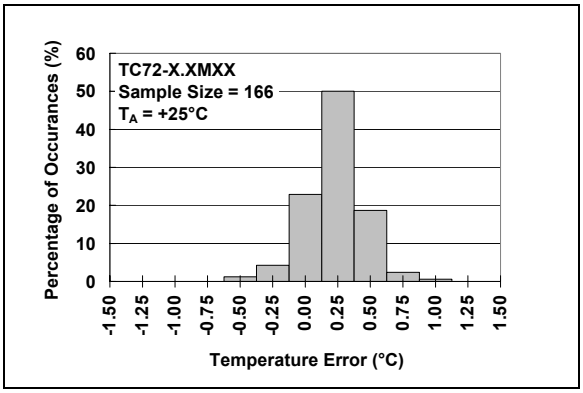


***图 2-9：** +25 摄氏度时温度精度柱状图*

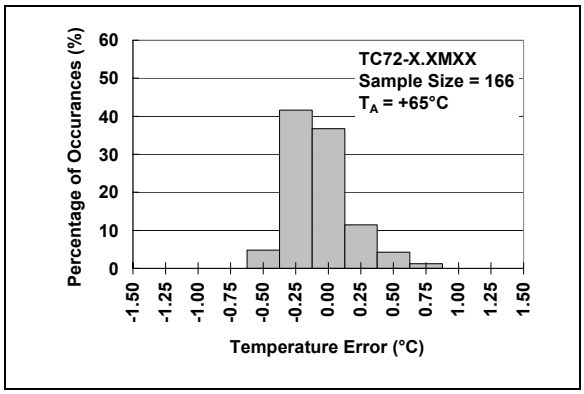


***图 2-10：** +65 摄氏度时温度精度柱状图*

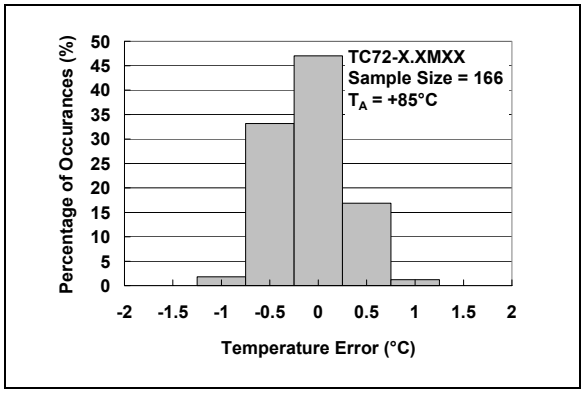


***图 2-11：** +85 摄氏度时温度精度柱状图*

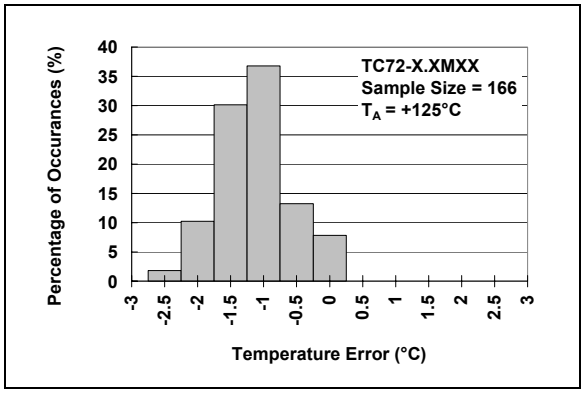


***图 2-12：** +125 摄氏度时温度精度柱状图*

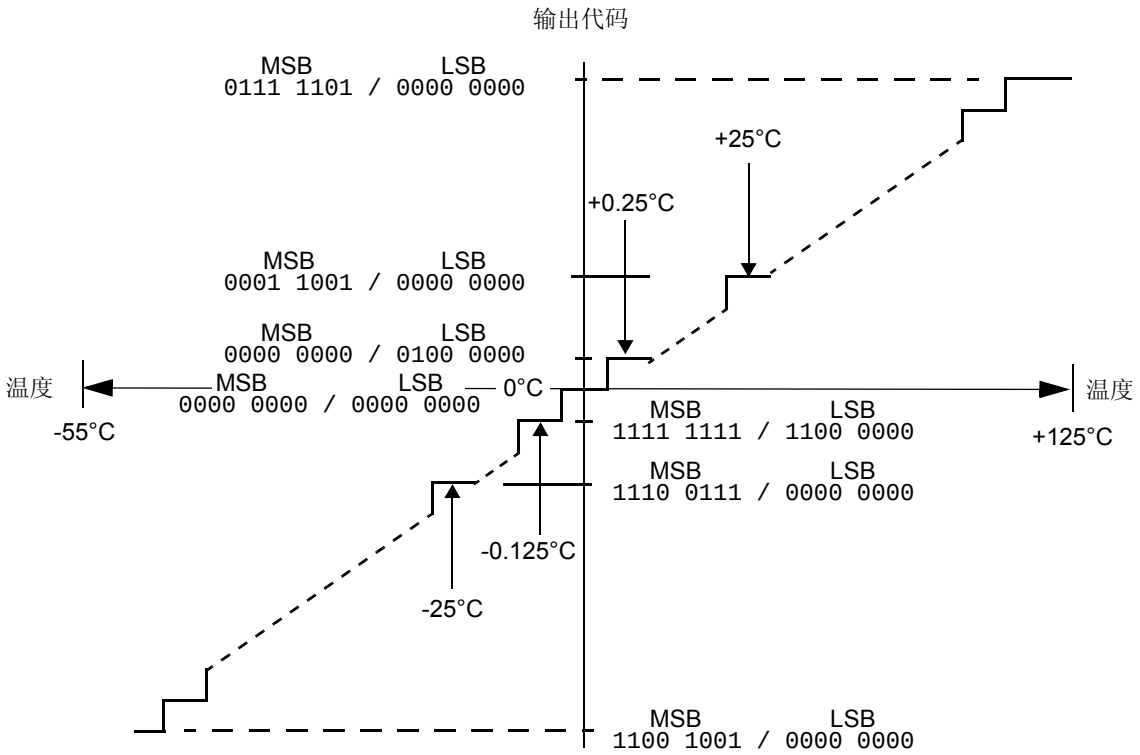
3.0 功能描述

TC72 包含一个带隙式温度传感器、一个 10 位 Σ–Δ 模 / 数转换器（Analog-to-Digital Converter， ADC）、一个内部转换振荡器和一个双缓冲数字输出端口。10 位 ADC 的量程为 -128°C 至 +127°C，因此分辨率为 0.25°C/ 位。TC72 工作的环境温度范围规定为 -55°C 至 +125°C。

器件具有的四线串行接口完全兼容 SPI 规范，可用于与普通单片机和处理器进行简单通信。TC72 可工作于连续温度转换模式或单次转换模式。TC72 温度测量是在后台进行的，因此，通过串行 I/O 线读取温度值并不影响温度测量过程。

连续转换模式每隔约 150 ms 测量一次温度，并将数据存储在温度寄存器中。 TC72 内部的时钟发生器控制自动温度转换序列。自动温度采样操作将无限重复，直至通过对控制寄存器的写操作使 TC72 处于关断模式。TC72 一直处于关断模式直到控制寄存器的关断位复位。

相反，单次转换模式仅执行一次温度测量，然后回到节能的关断模式。对于低功耗应用，这种模式特别有用。



注： ADC 转换器的量程为 -128°C 至 -127°C，但是 TC72 的工作范围规定为 -55°C 至 +125°C。

图 3-1： *温度—数字传递函数 （非线性比例）*

3.1 温度数据格式

使用 10 位二进制补码数字字的格式来表示温度值，分辨率为 0.25°C/ 位。温度数据以二进制补码的格式存储在温度寄存器中。 ADC 转换器的量程为 -128°C 至 +127°C，但是 TC72 的工作范围规定为 -55°C 至 +125°C。

示例：

温度 = 41.5°C

MSB 温度寄存器 = 00101001b
= $2^5 + 2^3 + 2^0$
= 32 + 8 + 1 = 41

LSB 温度寄存器 = 10000000b = 2^{-1} = 0.5

表 3-1： TC72 温度输出数据

温度	二进制 MSB / LSB	十六进制
+125°C	0111 1101/0000 0000	7D00
+25°C	0001 1001/0000 0000	1900
+0.5°C	0000 0000/1000 0000	0080
+0.25°C	0000 0000/0100 0000	0040
0°C	0000 0000/0000 0000	0000
-0.25°C	1111 1111/1100 0000	FFC0
-25°C	1110 0111/0000 0000	E700
-55°C	1100 1001/0000 0000	C900

表 3-2： 温度寄存器

D7	D6	D5	D4	D3	D2	D1	D0	地址 / 寄存器
符号	2^6	2^5	2^4	2^3	2^3	2^1	2^0	02H 温度 MSB
2^{-1}	2^{-2}	0	0	0	0	0	0	01H 温度 LSB

3.2 上电和掉电

上电后，TC72 处于低功耗关断模式。通过对控制寄存器执行写操作来选择连续温度转换模式，如第 4.0 节“内部寄存器结构”所述。

对于 TC72，供电电压低于 1.6V （典型值）通常被认为是掉电状态。若供电电压低于 1.6V 门限值，内部寄存器将复位成上电时的默认状态。

3.3 串行总线接口

串行接口包括片选 （Chip Enable， CE）、串行时钟（Serial Clock， SCK）、串行数据输入（Serial Data Input， SDI）以及串行数据输出（Serial Data Output，SDO ）信号线。TC72 作为从器件工作，它符合 SPI 总线规范，这一串行接口设计成兼容 Microchip PIC® 系列单片机。

CE 输入用于在有多个器件连接到串行时钟和数据线时选择 TC72。 CE 为高电平有效，当 CE 等于逻辑高电平时，数据可以写入器件或从器件读出。 CE 为低电平时， SCK 输入被禁止。 CE 线的上升沿启动读或写操作，而 CE 的下降沿结束读或写操作。

SCK 输入由外部单片机提供，用于同步 SDI 和 SDO 线的数据。SDI 输入向 TC72 的控制寄存器写入数据，而 SDO 从温度寄存器中输出温度数据和控制寄存器关断位的状态。

TC72 具有能在 SCK 输入信号为有效高电平或低电平时工作的能力。当 CE 信号变成高电平时，检测到 SCK 的无效状态，而时钟输入 （CP） 的极性决定数据是在系统时钟的上升沿或下降沿移入或移出，如图 3-2 所示。表 3-3 给出了用于传送数据到寄存器和从寄存器移出数据的相应时钟边沿。每个时钟脉冲传送一位数据，数据位以 8 位为一组传送，如图 3-3 所示。

先发送地址字节，随后为数据。地址的最高位 A7 决定要进行读操作还是写操作。如果 A7 = “0”，将进行一个或多个读操作；否则 A7 = “1”，进行一个或多个写操作。

数据可以采用单字节或多字节包的方式进行传送，如图 3-3 所示。在 3 字节包中，数据序列包括温度数据的 MSb、温度数据的 LSb 和紧接着的控制寄存器数据。通过向寄存器写入所需数据包的最高地址来启动多字节读功能。在芯片使能引脚保持有效的时间内， TC72 将自动发送寻址的寄存器和所有低地址寄存器的数据。

表 3-3：　　工作模式

模式	CE	SCK（注 1）	SDI	SDO
禁止	低电平	输入禁止	输入禁止	高阻态
写 （A7 = 1）	高电平	CP=1，数据在下降沿移出，数据在时钟上升沿移入	数据位锁存	高阻态
		CP=0，数据在上升沿移出，数据在时钟下降沿移入		
读 （A7 = 0）	高电平	CP=1，数据在下降沿移出，数据在时钟上升沿移入	X	下一个数据位移出 **注 2**
		CP=0，数据在上升沿移出，数据在时钟下降沿移入		

注　1： CP 为单片机系统时钟的时钟极性。如果 SCK 的无效状态为逻辑高电平，则 CP 等于 “1”；否则，若 SCK 的无效状态为逻辑低电平，则 CP 等于 “0”。

2： 在读操作期间，SDO 保持高阻态直到 8 个数据位开始从温度寄存器移出。

3.4　读操作

TC72 使用 CE、 SCK 和 SDO 线从温度和控制寄存器输出数据。图 3-3 显示了读操作的时序图。芯片使能（CE）变为高电平启动通信。SDO 线保持为输出 LSb 位的电压，当 CE 线变为逻辑低电平时，该线变为三态。

3.5　写操作

数据移入控制寄存器，以使能 TC72 的节能关断模式。写操作如图 3-3 所示，通过使用 CE、SCK 和 SDI 线来完成。

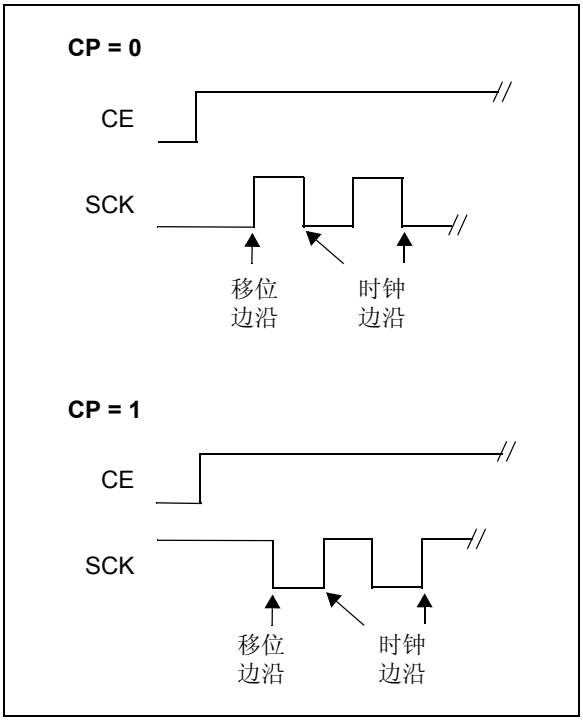


图 3-2：　　串行时钟极性 （CP）操作

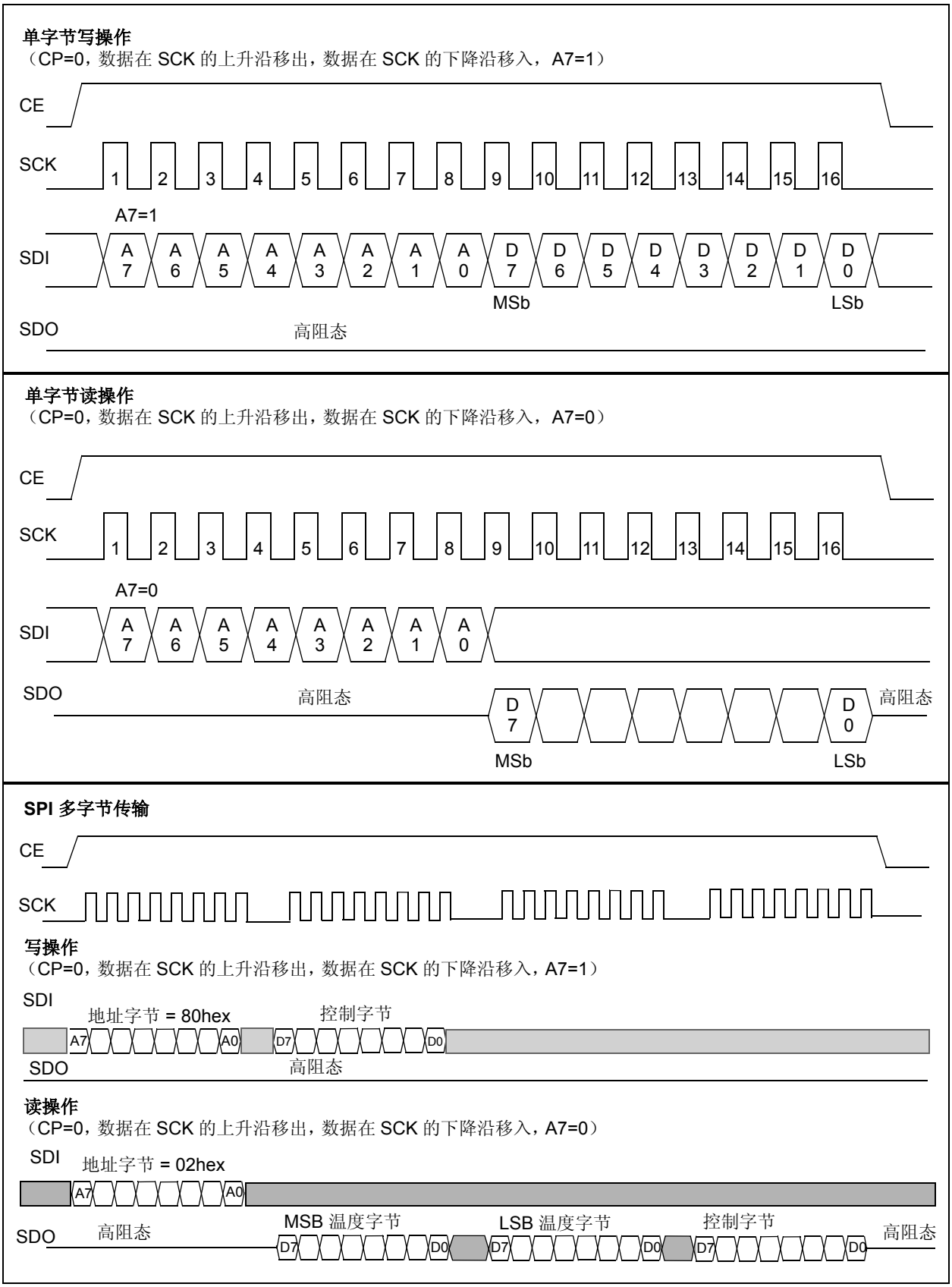


图 3-3： *串行接口时序图（CP=0）*

4.0 内部寄存器结构

TC72 的寄存器由下表列出。

表 4-1： TC72 的寄存器

寄存器	读地址	写地址	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR/BOR 时的值
控制	00hex	80hex	0	0	0	单次（OS）	0	1	0	关断（SHDN）	05hex
LSB 温度	01hex	N/A	T1	T0	0	0	0	0	0	0	00hex
MSB 温度	02hex	N/A	T9	T8	T7	T6	T5	T4	T3	T2	00hex
生产厂商 ID	03hex	N/A	0	1	0	1	0	1	0	0	54hex

4.1 控制寄存器

控制寄存器为可读写寄存器，用于选择关断模式、连续转换模式或单次转换模式。温度转换模式选择逻辑如表 4-2 所示。关断（SHDN）位存储在控制寄存器的 bit 0 。若 SHDN 等于“1”，则 TC72 将进入节能的关断模式。若 SHDN 等于 “0”，则 TC72 每隔约 150 ms 进行一次温度转换。

上电时， SHDN 位设置为 “1”。因此，TC72 最初 处于关断工作模式。通过在控制寄存器的 SHDN 位写入“0”，选择连续温度转换模式。

在不需要有源温度监测的场合，可使用关断模式来使 TC72 的功耗最小。关断模式禁止温度转换电路，但串行 I/O 通信端口仍然有效。通过对控制寄存器执行写操作来选择连续温度转换模式或单次工作模式，从而启动温度转换。控制寄存器写操作后经过约 150 ms， MSB 和 LSB 温度寄存器将包含温度数据。

向控制寄存器的 bit 4 写入 “1”将选择单次转换模式。单次转换模式只执行一次温度测量，然后返回节能的关断模式。完成温度转换后， 单次转换 （OS）位复位为“0”（即“OFF”）。用户必须设置单次转换位为“1”来启动另外一次温度转换。

TC72 控制器寄存器的 Bit 1、3、5、6 和 7 未使用。Bit 2 设置成逻辑“1”。对这些位的写操作不会影响 TC72 的正常操作。

4.2 温度寄存器

温度寄存器为只读寄存器，保存以 10 位二进制补码表示的温度测量值。LSB 温度寄存器的 Bit 0 至 Bit 5 始终设置成 “0”。

在上电复位（POR）或出现可使器件掉电复位 （BOR）的低电压时，温度寄存器复位成全零，其表示的温度值为 0°C。V_{DD} 供电电压低于 1.6V 被认为是复位事件，它会使温度寄存器复位成上电时的状态。

4.3 生产厂商 ID 寄存器

生产厂商标识（ID）寄存器为只读寄存器，用于表明温度传感器为 Microchip 的器件。

表 4-2： 控制寄存器的温度转换模式选择

工作模式	单次 （OS） Bit 4	关断 （SHDN） Bit 0
连续温度转换	0	0
关断	0	1
连续温度转换 （ 若 SHDN = “0”，则忽略单次转换命令）	1	0
单次转换	1	1

5.0 应用信息

TC72 不需要任何其他元器件就可以测量温度。然而，推荐在 V_{DD} 和 GND 引脚之间接一个 0.1mF 至 1mF 的去耦电容。虽然 TC72 的电流消耗并不大（250 mA，典型值），但其中却包含有片内数据采集和内部数字开关电路。因此，在传感器外部使用去耦电容是一个很好的设计技巧。应使用高频陶瓷电容，并使其尽可能靠近芯片电源引脚放置，以便为 TC72 提供有效的抗噪声保护。

TC72 通过监控 IC 管芯中二极管两端的电压来测量温度。管芯和印刷电路板（Printed Circuit Board，PCB）之间的低阻抗热传导通道由 TC72 的封装引脚提供，因而 TC72 能有效监控印刷电路板的温度。然而，其与周围空气交换热量的通道不是很有效，因为器件塑封的作用等同于热绝缘体。因此，周围空气温度（假如在空气和 PCB 之间存在很大的温差）只对 TC72 温度测量产生很小的影响。

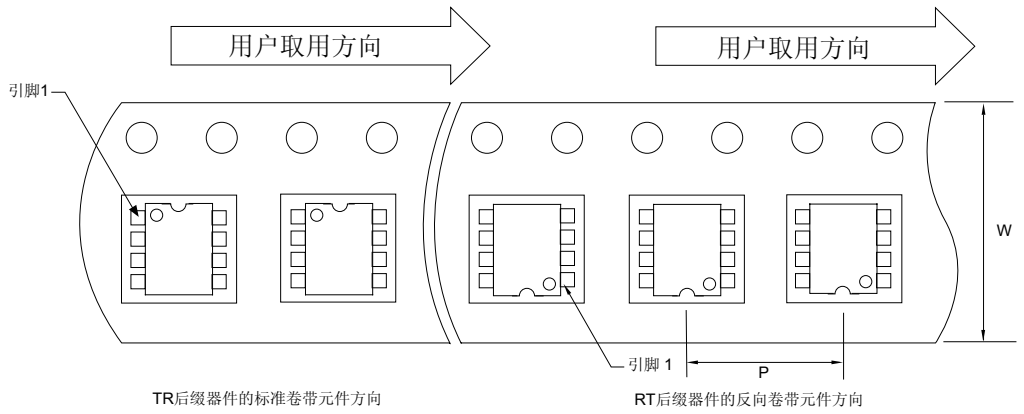
需要留意 DFN 封装底部外露的金属中心焊盘连接到芯片的衬底， 中心焊盘应连接到 PCB 地平面，或者作为“无连接”引脚。中心焊盘的机械尺寸由本数据手册第 6.0 节“封装信息”给出。

如果 TC72 SPI通信线的负载过大，可能存在自热误差。通常来说，由于 TC72 的电流消耗相对较小，自热误差是可以忽略的。但是，如果 SPI 通信引脚的拉或灌电流为 TC72 的最大电流规范值，则可能导致大约 0.5°C 的温度精度误差。因此，为了获得最大的温度精度， SPI 引脚的输出负载应当最小。

6.0 封装信息

6.1 卷带式

8引脚MSOP器件的元件卷带方向



TR后缀器件的标准卷带元件方向　　RT后缀器件的反向卷带元件方向

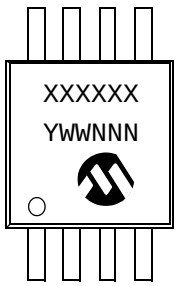
包装卷带，每卷带元件数目和卷带尺寸

封装	卷带宽度（W）	间距（P）	每卷带器件数	卷带尺寸
8引脚 MSOP	12 mm	8 mm	2500	13 in

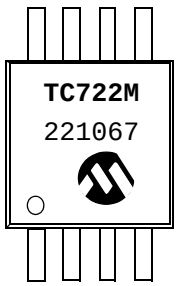
8 引脚 DFN 封装的卷带信息暂时无法提供。

6.2 封装标识信息

8 引脚 MSOP



示例：



8 引脚 DFN

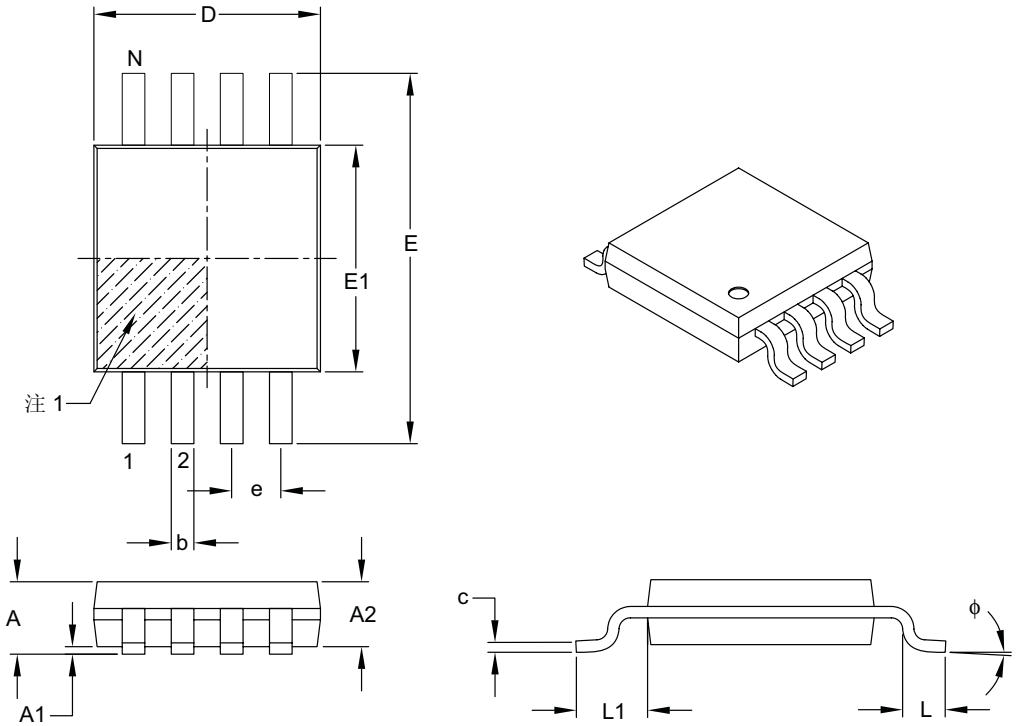
示例：

图注：	XX...X	客户指定信息 *
	YY	年份代码 （日历年的最后两位数字）
	WW	星期代码 （一月第一个星期的代码为 “01”）
	NNN	以字母数字排序的追踪代码
注：	Microchip 元器件编号如果无法在同一行内完整标注，将换行标出，因此会限制客户指定信息的可用字符数。	

***** 标准 OTP 标识包含 Microchip 部件号、年份代码、星期代码和追踪代码。

8 引脚塑封微型小外形封装 （MS）（MSOP）

注: 最新的封装图，请至 http://www.microchip.com/packaging 查看 Microchip 封装规范。



		单位	毫米	
	尺寸范围	最小	正常	最大
引脚数	N	8		
引脚间距	e	0.65 BSC		
总高度	A	–	–	1.10
塑模封装厚度	A2	0.75	0.85	0.95
悬空间隙	A1	0.00	–	0.15
总宽度	E	4.90 BSC		
塑模封装宽度	E1	3.00 BSC		
总长度	D	3.00 BSC		
底脚长度	L	0.40	0.60	0.80
引脚投影长度	L1	0.95 REF		
底脚倾角	ϕ	0°	–	8°
引脚厚度	c	0.08	–	0.23
引脚宽度	b	0.22	–	0.40

注：

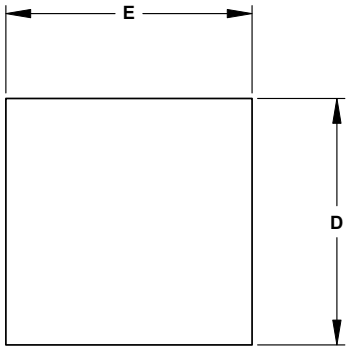
1. 引脚1定位标记可能会有变化，但一定位于阴影区域内。
2. 尺寸D和E1不包括塑模的毛边或突起。塑模每侧的毛边或突起不得超过0.15 mm。
3. 尺寸和公差遵循ASME Y14.5M。

BSC： 基本尺寸。理论精确值，不含公差。
REF： 参考尺寸。通常也不包含公差，仅供参考。

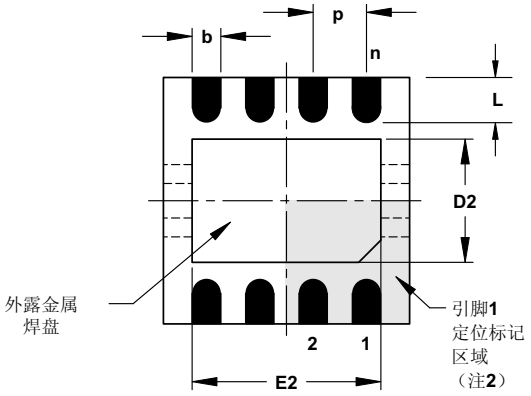
Microchip Technology 图号 C04-111B

8 引脚塑封双列扁平无引脚封装 （MF）——主体 3x3x1 mm （DFN）

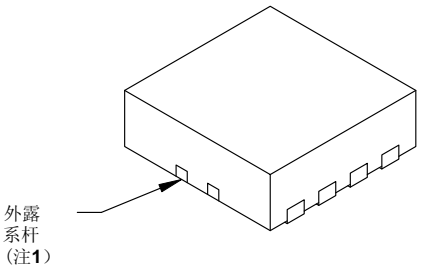
注: 最新的封装图，请至 http://www.microchip.com/packaging 查看 Microchip 封装规范。



俯视图



仰视图



	单位	英寸			毫米*		
	尺寸范围	最小	正常	最大	最小	正常	最大
引脚数	n		8			8	
引脚间距	p		.026 BSC			0.65 BSC	
总高度	A	.031	.035	.039	0.80	0.90	1.00
悬空间隙	A1	.000	.001	.002	0.00	0.02	0.05
引脚厚度	A3		.008 REF.			0.20 REF.	
总长度	E		.118 BSC			3.00 BSC	
外露焊盘长度 （注4）	E2	.055		.096	1.39		2.45
总宽度	D		.118 BSC			3.00 BSC	
外露焊盘宽度 （注4）	D2	.047		.069	1.20		1.75
引脚宽度	b	.007	.010	.015	0.23	0.26	0.37
引脚长度	L	.012	.019	.022	0.30	0.48	0.55

*控制参数

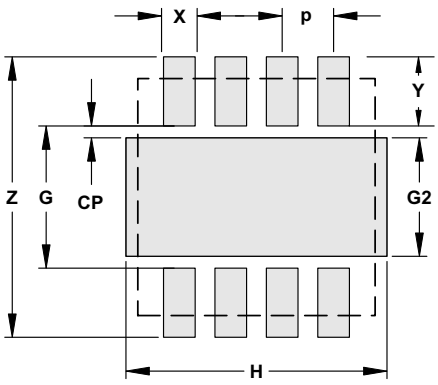
注：

1. 封装两端可能有一个或多个外露的系杆。
2. 引脚1 定位标记可能会有变化，但必须位于阴影区域内。
3. 尺寸D 和 E 不包括塑模毛边或突起。塑模每侧的毛边或突起不得超过0.010英寸（0.254mm）。
4. 外露焊盘的尺寸会随晶座尺寸变化。
5. 等同于JEDEC 号：待定

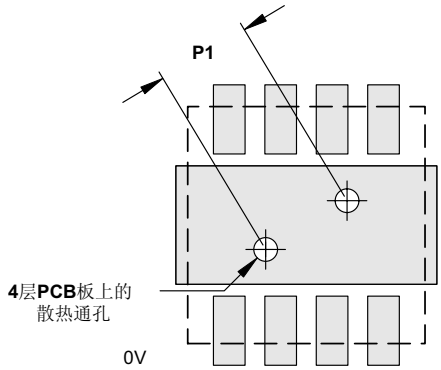
图号：C04-062

8 引脚塑封双列扁平无引脚 （MF）封装——主体 3x3x1 mm （DFN）

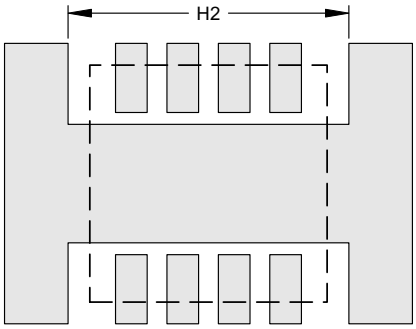
注: 最新的封装图，请至 http://www.microchip.com/packaging 查看 Microchip 封装规范。



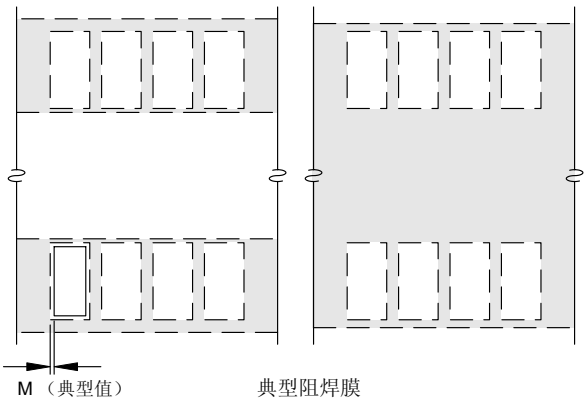
DFN 焊盘布局



可选的4层PCB焊盘布局



可选的双层PCB焊盘布局



典型阻焊膜

单位		英寸			毫米*		
尺寸范围		最小	正常	最大	最小	正常	最大
引脚端间距	p		.026 BSC			0.65 BSC	
引脚端焊盘布局O.D.	Z	.134		.157	3.40		4.00
引脚端焊盘布局I.D.	A1	.057		.060	1.45		1.53
外露的焊盘间距	CP	.006			0.15		
内部的引脚间距	Z	.071			1.80		
引脚端焊盘宽度	X	.014		.017	0.35		0.42
引脚端焊盘长度	Y	.033		.035	0.85		0.88
外露的焊盘长度	H	.130			3.30		
可选的外露焊盘长度	H2	.130			3.30		
外露的焊盘宽度 （注1）	G2	.057		.059	1.45		1.50
散热通孔的间距	P1		.047			1.20	
散热通孔的直径	V		.012			0.30	
最小阻焊膜间距	M	.002			0.05		

*控制参数

注：

1. 外露焊盘的尺寸会随晶座尺寸变化。

图号：C04-2062

MICROCHIP 网站

Microchip 网站（www.microchip.com）为客户提供在线支持。客户可通过该网站方便地获取文件和信息。只要使用常用的因特网浏览器即可访问。网站提供以下信息：

- **产品支持**——数据手册和勘误表、应用笔记和样本程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及存档软件
- **一般技术支持**——常见问题（FAQ）、技术支持请求、在线讨论组以及 Microchip 顾问计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

变更通知客户服务

Microchip 的变更通知客户服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请登录 Microchip 网站 www.microchip.com，点击“变更通知客户（Customer Change Notification）”服务后按照注册说明完成注册。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师（FAE）
- 技术支持

客户应联系其代理商、代表或应用工程师（FAE）寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过http://support.microchip.com获得网上技术支持。

读者反馈表

我们努力为您提供最佳文档，以确保您能够成功使用 Microchip 产品。如果您对文档的组织、条理性、主题及其他有助于提高文档质量的方面有任何意见或建议，请填写本反馈表并传真给我公司 TRC 经理，传真号码为 86-21-5407-5066。

请填写以下信息，并从下面各方面提出您对本文档的意见。

致: TRC 经理　　总页数 _____

关于: 读者反馈

发自: 姓名 _____

公司 _____

地址 _____

国家 / 省份 / 城市 / 邮编 _____

电话: （_____）_____　　传真:（_____）_____

应用（选填）:

您希望收到回复吗？是____ 否____

器件: TC72　　文献编号: DS21743A_CN

问题:

1. 本文档中哪些部分最有特色？

2. 本文档是否满足了您的软硬件开发要求？如何满足的？

3. 您认为本文档的组织结构便于理解吗？如果不便于理解，那么问题何在？

4. 您认为本文档应该添加哪些内容以改善其结构和主题？

5. 您认为本文档中可以删减哪些内容，而又不会影响整体使用效果？

6. 本文档中是否存在错误或误导信息？如果存在，请指出是什么信息及其具体页数。

7. 您认为本文档还有哪些方面有待改进？

产品标识体系

欲订货，或获取价格、交货等信息，请与我公司生产厂或销售办事处联系。

部件编号	-X.X	X	XX
器件	电压范围	温度范围	封装

器件：	TC72：	带 SPI 接口的数字温度传感器
电压范围：	2.8	= 针对 2.8V 优化的精度
	3.3	= 针对 3.3V 优化的精度
	5.0	= 针对 5.0V 优化的精度
温度范围：	M	= -55°C 至 +125°C
封装：	MF	= 8 引脚双列扁平无引脚 （DFN）（3x3mm）
	MFTR	= 8 引脚双列扁平无引脚 （DFN）（3x3mm）（卷带式）
	UA	= 8 引脚塑封微型小外形 （MSOP）
	UATR	= 8 引脚塑封微型小外形 （MSOP）（卷带式）

示例：

a) TC72-2.8MUA：数字温度传感器， 2.8V， 8 引脚 MSOP 封装。

b) TC72-2.8MUATR：数字温度传感器，2.8V，8 引脚 MSOP （卷带式）封装。

c) TC72-2.8MMF：数字温度传感器， 2.8V，8 引脚 DFN 封装。

d) TC72-3.3MUA：数字温度传感器， 3.3V，8 引脚 MSOP 封装。

e) TC72-3.3MMF：数字温度传感器， 3.3V，8 引脚 DFN 封装。

f) TC72-5.0MUA：数字温度传感器， 5.0V，8 引脚 MSOP 封装。

g) TC72-5.0MMF：数字温度传感器， 5.0V，8 引脚 DFN 封装。

h) TC72-5.0MMFTR：数字温度传感器，5.0V，8 引脚 DFN （卷带式）封装。

销售和支持

数据手册

初级数据手册中述及的产品可能带有一份勘误表，描述了运行中的细小差别以及建议的变通方法。要了解某一器件是否存在勘误表，请通过以下方式之一联系我们：

1. 当地的 Microchip 销售办事处
2. Microchip 网站：www.microchip.com

请说明器件名称，以及您所使用的芯片和数据手册 （包括文献编号）的版本。

客户通知系统

在我们的网站 （www.microchip.com）上注册，获取产品最新信息。

注：

请注意以下有关 Microchip 器件代码保护功能的要点：

- tMicrochip 的产品均达到 Microchip 数据手册中所述的技术指标。
- tMicrochip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- t 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- tMicrochip 愿与那些注重代码完整性的客户合作。
- tMicrochip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2002 认证。公司在 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品方面的质量体系流程均符合 ISO/TS-16949:2002。此外，Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://support.microchip.com
网址：www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

科科莫 Kokomo
Kokomo, IN
Tel: 1-765-864-8360
Fax: 1-765-864-8387

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

圣克拉拉 Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

加拿大多伦多 Toronto
Mississauga, Ontario,
Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 北京
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

中国 – 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 – 香港特别行政区
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 – 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 – 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 – 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 – 深圳
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

中国 – 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 – 厦门
Tel: 86-592-238-8138
Fax: 86-592-238-8130

中国 – 西安
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

中国 – 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

台湾地区 – 高雄
Tel: 886-7-536-4818
Fax: 886-7-536-4803

台湾地区 – 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

台湾地区 – 新竹
Tel: 886-3-572-9526
Fax: 886-3-572-6459

亚太地区

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

印度 India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

日本 Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

韩国 Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 Denmark-Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

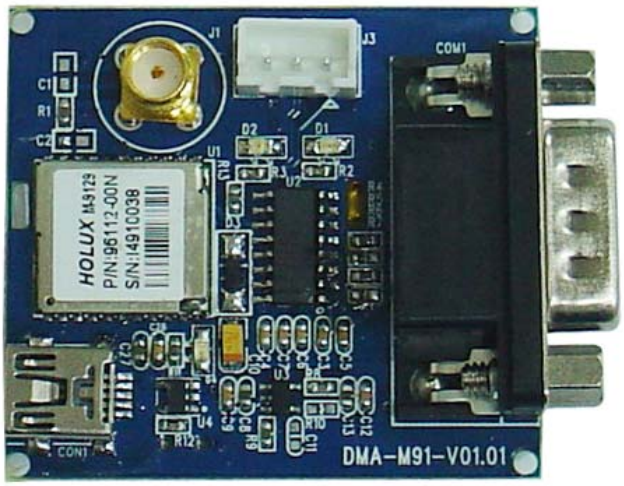
西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

英国 UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/02/08

第一章 M-91S 模組介紹

1-1 產品外觀



M-91S 模組外觀圖

1-2 產品特色

M-91S 是一種根據低耗電 Mediatek GPS 解決方案設計的超小型 13 x 15 x 2.2 mm GPS 引擎模組。它對於導航應用提供高達 -159dBm 的絕佳靈敏度與快速的第一次定位時間。**M-91S** 拉出 DTYPE 及 3P RS232 信號，非常方便於在外接 GPS 設備的平台使用。

- 精巧外型：13 * 15 *2.2 mm
- 符合 RoHS／WEEE 規定
- 高靈敏性 159dBm
- 可搜尋多達 32 個衛星頻道
- 快速位置修正
- 低耗電量
- 可使用 RTCM-in
- 內建 WAAS／EGNOS／MSAS 解調器
- 支援 NMEA0183 V 3.01 資料通訊協定
- 定位服務的即時導航
- 適用於汽車導航、船隻導航、艦隊管理、AVL 和定位服務、自動導航、個人導航或旅遊裝置、追蹤裝置／系統和地圖裝置應用

1-3 規格特性

- GPS 模組焊於擴充板上，由排針及 DTYPE 拉出信號，方便介接排針信號或是 RS232 DTYPE 實作介接，在不同的平台上使用有其方便性。
- 採用 Mediatek GPS 晶片組
- 頻率： L1, 1575.42MHz
- C/A 碼： 1.023MHZ 晶片速率
- 頻道： 查看搜尋中的 32 個頻道
- 靈敏度：-159dBm
- 電源(操作電流)

  取得定位：65 mA@3.3V

  追　　蹤：< 35 mA@3.3V

  DC 輸入範圍：VCC 3.0~5.0V; VBAT 3.0~5.0V
- 運算程式： ARM7EJ-S
- 處理速度：48 MHZ
- 內建快閃記憶體 4Mb
- 更新速率： 1HZ
- 數據： WGS84 (預設) 共 219 筆數據
- 1PPS： 啓用（1Hz 脈波 10%工作週期）
- 作業系統：WinCE 5.0 / WinCE 6.0 / Android 2.X
- 操作溫度： -40℃ to + 85℃
- 存放溫度： -40℃ to + 85℃
- 操作濕度： 5% to 95% 無壓縮條件下
- 尺寸：13×15×2.2 公釐
- 重量：2g

1-4 位置精準度

- 位置：Without aid: 3.0 M 2D-RMS
  DGPS(WAAS, EGNOS, MSAS, RTCM): 2.5 M
- 速率：Without aid: 0.1 M /sec
- 時間：0.1 微秒。GPS 同步時間
  (依照 MTK 晶片規格)

1-5 定位時間

- 熱開機： 1 秒，平均
- 暖開機： 33 秒，平均
- 冷開機： 36 秒，平均
- 重新抓取：< 1 秒，平均

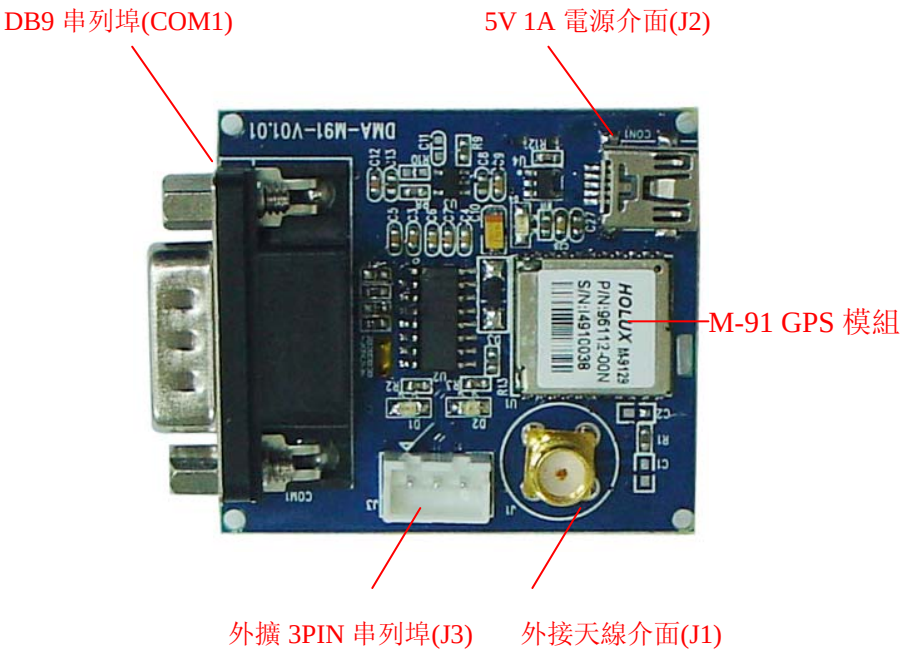
1-6 輸出規格及介面

- 通訊協定：NMEA0183(v3.1)
- 傳輸速率： 4800 bps (預設)
  資料位元： 8
  同位位元： N
  停止位元： 1
- 輸出格式：GGA、GLL、GSA、GSV、RMC、VTG
- 選用傳輸速率：9600/…/57600 bps
- 介面：CMOS 2.8 V 等級

1-7 M-91S 轉接板信號定義

M-91S 模組主要是用在 DMA-6410、DMA-6410XP、DMA-210U、DMA-210XP 等平台上，但是平台上沒有直接做 M-91S 的封裝，而是把 M-91S 做在一塊轉接板上，用戶需要用到時，可以透過 DTYPE 雙母線插在平台的相應介面上，這樣既方便有靈活。

轉接板如下圖所示：

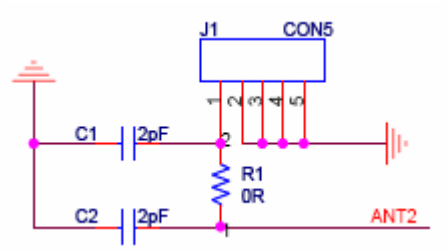


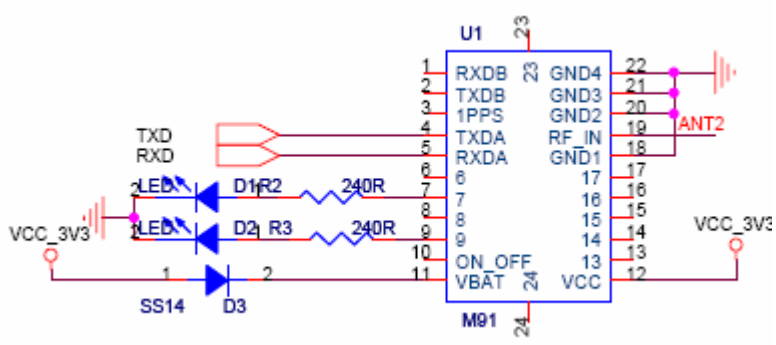
說明：

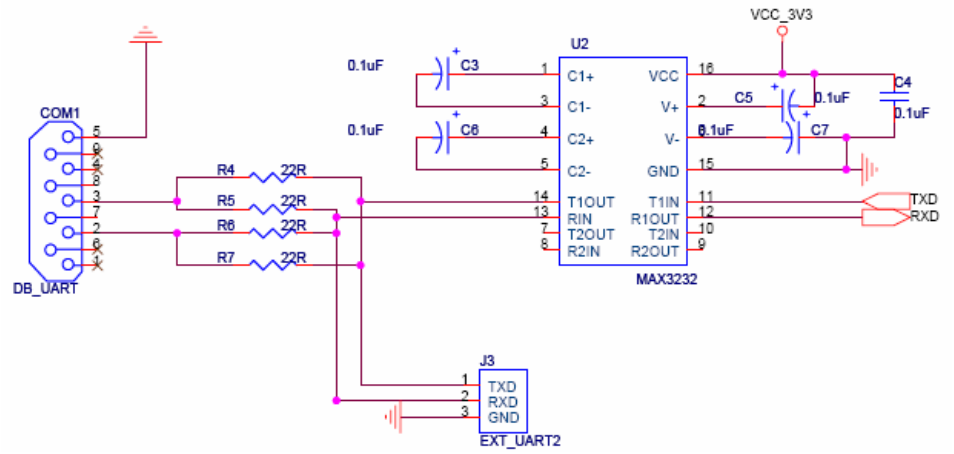
1、當使用者在室內，無法收到更多更好的 GPS 信號時，可以外接天線延伸接收模組至適當位置，以收到最佳信號。外接天線請接在如上圖所示的 J1 上。外接天線如下圖所示：

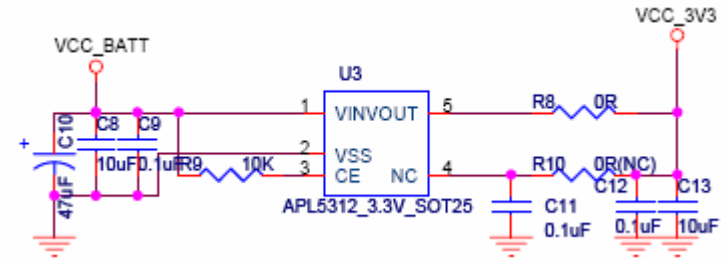
1-8 M-91S 電路圖

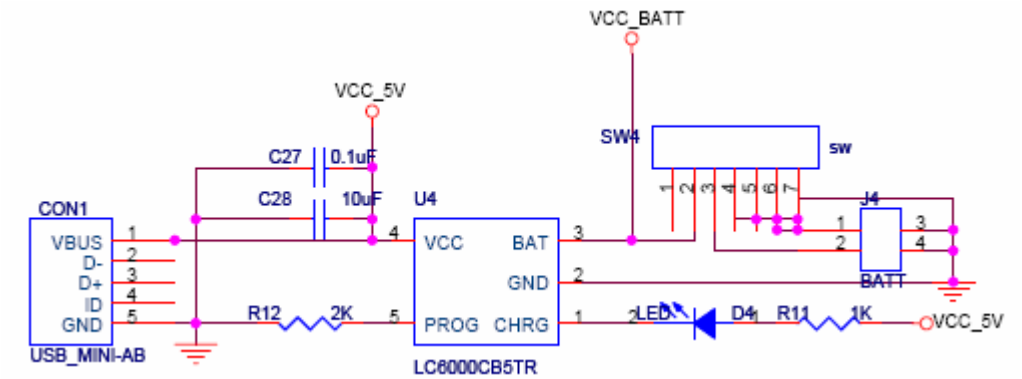
電路圖如下：






COM1
DB_UART
R4 22R
R5 22R
R6 22R
R7 22R
U2
C1+
C1-
C2+
C2-
VCC
V+
V-
GND
T1OUT
RIN
T2OUT
R2IN
T1IN
R1OUT
T2IN
R2OUT
MAX3232
C3 0.1uF
C6 0.1uF
C5 0.1uF
C7 0.1uF
C4 0.1uF
VCC_3V3
TXD
RXD
J3
TXD
RXD
GND
EXT_UART2


VCC_BATT
VCC_3V3
U3
VINVOUT
VSS
CE
NC
APL5312_3.3V_SOT25
C10 47uF
C8 10uF
C9 0.1uF
R9 10K
R8 0R
R10 0R(NC)
C11 0.1uF
C12 0.1uF
C13 10uF


VCC_BATT
VCC_5V
SW4
sw
CON1
VBUS
D-
D+
ID
GND
USB_MINI-AB
C27 0.1uF
C28 10uF
U4
VCC
BAT
GND
PROG
CHRG
LC6000CB5TR
R12 2K
LED
D4
R11 1K
J4
BATT
VCC_5V

第二章 GPS 原理與應用

2-1 GPS 定位原理

2-1.1 GPS 的組成

GPS（Global Positioning System）即全球定位系統，是由美國建立的一個衛星導航定位系統，利用該系統，用戶可以在全球範圍內實現全天候、連續、即時的三維導航定位和測速；另外，利用該系統，用戶還能夠進行高精度的時間傳遞和高精度的精密定位。

GPS 計畫始於 1973 年，已於 1994 年進入完全運行狀態。GPS 的整個系統由空間部分、地面控制部分和用戶部分所組成：

空間部分

GPS 的空間部分是由 24 顆 GPS 工作衛星所組成，這些 GPS 工作衛星共同組成了 GPS 衛星星座，其中 21 顆爲可用于導航的衛星，3 顆爲活動的備用衛星。這 24 顆衛星分佈在 6 個傾角爲 55°的軌道上繞地球運行。衛星的運行週期約爲 12 恒星時。每顆 GPS 工作衛星都發出用於導航定位的信號。GPS 用戶正是利用這些信號來進行工作的。

控制部分

GPS 的控制部分由分佈在全球的由若干個跟蹤站所組成的監控系統所構成，根據其作用的不同，這些跟蹤站又被分爲主控站、監控站和注入站。主控站有一個，位於美國克羅拉多（Colorado）的法爾孔（Falcon）空軍基地，它的作用是根據各監控站對 GPS 的觀測資料，計算出衛星的星曆和衛星鐘的改正參數等，並將這些資料透過注入站注入到衛星中去；同時，它還對衛星進行控制，向衛星發佈指令，當工作衛星出現故障時，調度備用衛星，替代失效的工作衛星工作；另外，主控站也具有監控站的功能。監控站有五個，除了主控站外，其他四個分別位於夏威夷（Hawaii）、阿松森群島（Ascencion）、迭哥伽西亞（Diego Garcia）、卡瓦加蘭（Kwajalein），監控站的作用是接收衛星信號，監測衛星的工作狀態；注入站有三個，它們分別位於阿松森群島（Ascencion）、迭哥伽西亞（Diego Garcia）、卡瓦加蘭（Kwajalein），注入站的作用是將主控站計算出的衛星星曆和衛星鐘的改正數等注入到衛星中去。

用戶部分

GPS 的用戶部分由 GPS 接收機、資料處理軟體及相應的用戶設備如電腦氣象儀器等所組成。它的作用是接收 GPS 衛星所發出的信號，利用這些信號進行導航定位等工作。 以上這三個部分共同組成了一個完整的 GPS 系統。

2-1.2 GPS 信號

GPS 衛星發射兩種頻率的載波信號，即頻率爲 1575.42MHz 的 L1 載波和頻率爲 1227.60HMz 的 L2 載波，它們的頻率分別是基本頻率 10.23MHz 的 154 倍和 120 倍，它們的波長分別爲 19.03cm 和 24.42cm。在 L1 和 L2 上又分別調製著多種信號，這些信號主要有：

C/A 碼

C/A 碼又被稱爲粗捕獲碼，它被調製在 L1 載波上，是 1MHz 的僞隨機雜訊碼（PRN 碼），其碼長爲 1023 位（週期爲 1ms）。由於每顆衛星的 C/A 碼都不一樣，因此，我們經常用它們的 PRN 號來區分它們。C/A 碼是普通用戶用以測定測站到衛星間的距離的一種主要的信號。

P 碼

P 碼又被稱爲精碼，它被調製在 L1 和 L2 載波上，是 10MHz 的僞隨機雜訊碼，其週期爲七天。在實施 AS 時，P 碼與 W 碼進行模二相加生成保密的 Y 碼，此時，一般用戶無法利用 P 碼來進行導航定位。

導航資訊

導航資訊被調製在 L1 載波上，其信號頻率爲 50Hz，包含有 GPS 衛星的軌道參數、衛星鐘改正數和其他一些系統參數。用戶一般需要利用此導航資訊來計算某一時刻 GPS 衛星在地球軌道上的位置，導航資訊也被稱爲廣播星曆。

SPS 和 PPS 是 GPS 系統針對不同用戶提供兩種不同類型的服務。一種是標準定位服務(SPSStandard Positioning Service)，另一種是精密定位服務(PPSPrecision Positioning Service)。這兩種不同類型的服務分別由兩種不同的子系統提供，標準定位服務由標準定位子系統(SPSStandard Positioning System)提供，精密定位服務則由精密定位子系統(PPSPrecision Positioning System)提供。

SPS 主要面向全世界的民用用戶。

PPS 主要面向美國及其盟國的軍事部門以及民用的特許用戶。

在 GPS 定位中，經常採用下列觀測值中的一種或幾種進行資料處理，以確定出待定點的座標或待定點之間的基線向量：

L1 載波相位觀測值

L2 載波相位觀測值（半波或全波）

調製在 L1 上的 C/A 碼僞距

調製在 L1 上的 P 碼僞距

調製在 L2 上的 P 碼僞距

L1 上的多普勒頻移

L2 上的多普勒頻移

實際上，在進行 GPS 定位時，除了大量地使用上面的觀測值進行資料處理以外，還經常使用由上面的觀測值透過某些組合而形成的一些特殊觀測值，如寬巷觀測值（Wide-Lane）、窄巷觀測值（Narrow-Lane）、消除電離層延遲的觀測值（Ion-Free）來進行資料處理。

2-1.3 GPS 的誤差

我們在利用 GPS 進行定位時，會受到各種各樣因素的影響。影響 GPS 定位精度的因素可分爲以下四大類：

一、與 GPS 衛星有關的因素

SA

美國政府從其國家利益出發，透過降低廣播星曆精度（ 技術）、在 GPS 基準信號中加入高頻抖動（ 技術）等方法，人爲降低普通用戶利用 GPS 進行導航定位時的精度。

衛星星曆誤差

在進行 GPS 定位時，計算在某時刻 GPS 衛星位置所需的衛星軌道參數是透過各種類型的星曆[7]提供的，但不論採用哪種類型的星曆，所計算出的衛星位置都會與其真實位置有所差異，這就是所謂的星曆誤差。

衛星鐘差

衛星鐘差是 GPS 衛星上所安裝的原子鐘的鐘面時與 GPS 標準時間之間的誤差。

衛星信號發射天線相位中心偏差

衛星信號發射天線相位中心偏差是 GPS 衛星上信號發射天線的標稱相位中心與其真實相位中心之間的差異。

二、與傳播途徑有關的因素

電離層延遲

由於地球周圍的電離層對電磁波的折射效應，使得 GPS 信號的傳播速度發生變化，這種變化稱爲電離層延遲。電磁波所受電離層折射的影響與電磁波的頻率以及電磁波傳播途徑上電子總含量有關。

對流層延遲

由於地球周圍的對流層對電磁波的折射效應，使得 GPS 信號的傳播速度發生變化，這種變化稱爲對流層延遲。電磁波所受對流層折射的影響與電磁波傳播途徑上的溫度、濕度和氣壓有關。

多路徑效應

由於接收機周圍環境的影響，使得接收機所接收到的衛星信號中還包含有各種反射和折射信號的影響，這就是所謂的多路徑效應。

三、與接收機有關的因素

接收機鐘差

接收機鐘差是 GPS 接收機所使用的鐘的鐘面時與 GPS 標準時之間的差異。

接收機天線相位中心偏差

接收機天線相位中心偏差是 GPS 接收機天線的標稱相位中心與其真實的相位中心之間的差異。

接收機軟體和硬體造成的誤差

在進行 GPS 定位時，定位結果還會受到諸如處理與控制軟體和硬體等的影響。

四、其他

GPS 控制部分人爲或電腦造成的影響

由於 GPS 控制部分的問題或用戶在進行資料處理時引入的誤差等。

資料處理軟體的影響

資料處理軟體的演算法不完善對定位結果的影響。

2-1.4 GPS 定位方法

GPS 定位的方法是多種多樣的，用戶可以根據不同的用途採用不同的定位方法。GPS 定位方法可依據不同的分類標準，作如下劃分：

一、根據定位所採用的觀測值

僞距定位

僞距定位所採用的觀測值爲 GPS 僞距觀測值，所採用的僞距觀測值既可以是 C/A 碼僞距，也可以是 P 碼僞距。僞距定位的優點是資料處理簡單，對定位條件的要求低，不存在整周模糊度的問題，可以非常容易地實現即時定位；其缺點是觀測值精度低，C/A 碼僞距觀測值的精度一般爲 3 米，而 P 碼僞距觀測值的精度一般也在 30 個釐米左右，從而導致定位成果精度低，另外，若採用精度較高的 P 碼僞距觀測值，還存在 AS 的問題。

載波相位定位

載波相位定位所採用的觀測值爲 GPS 的載波相位觀測值，即 L1、L2 或它們的某種線性組合。載波相位定位的優點是觀測值的精度高，一般優於 2 個毫米；其缺點是資料處理過程複雜，存在整周模糊度的問題。

二、根據定位的模式

絕對定位

絕對定位又稱爲單點定位，這是一種採用一台接收機進行定位的模式，它所確定的是接收機天線的絕對座標。這種定位模式的特點是作業方式簡單，可以單機作業。絕對定位一般用於導航和精度要求不高的應用中。

相對定位

相對定位又稱爲差分定位，這種定位模式採用兩台以上的接收機，同時對一組相同的衛星進行觀測，以確定接收機天線間的相互位置關係。

三、根據獲取定位結果的時間

即時定位

即時定位是根據接收機觀測到的資料，即時地解算出接收機天線所在的位置。

非即時定位

非即時定位又稱後處理定位，它是透過對接收機接收到的資料進行後處理以進行定位得方法。

四、根據定位時接收機的運動狀態

動態定位

所謂動態定位，就是在進行 GPS 定位時，認爲接收機的天線在整個觀測過程中的位置是變化的。也就是說，在資料處理時，將接收機天線的位置作爲一個隨時間的改變而改變的量。動態定位又分爲 Kinematic 和 Dynamic 兩類。

靜態定位

所謂靜態定位，就是在進行 GPS 定位時，認爲接收機的天線在整個觀測過程中的位置是保持不變的。也就是說，在資料處理時，將接收機天線的位置作爲一個不隨時間的改變而改變的量。在測量中，靜態定位一般用於高精度的測量定位，其具體觀測模式多台接收機在不同的測站上進行靜止同步觀測，時間由幾分鐘、幾小時甚至數十小時不等。

2-2 GPS 的應用

2-2.1 GPS 的應用範圍

1、GPS 應用於測量

GPS 技術給測繪界帶來了一場革命。利用載波相位差分技術（RTK），在即時處理兩個觀測站的載波相位的基礎上，可以達到釐米級的精度。與傳統的手工測量手段相比，GPS 技術有著巨大的優勢： 測量精度高；操作簡便，儀器體積小，便於攜帶；全天候操作；觀測點之間無須通視；測量結果統一在 WGS84 座標下，資訊自動接收、存儲，減少繁瑣的中間處理環節。 當前，GPS 技術已廣泛應用於大地測量、資源勘查、地殼運動、地籍測量等領域。

2、GPS 應用於交通

計程車、租車服務、物流配送等行業利用 GPS 技術對車輛進行跟蹤、調度管理，合理分佈車輛，以最快的速度回應用戶的乘車或送請求，降低能源消耗，節省運行成本。 GPS 在車輛導航方面發揮了重要的角色，在城市中建立數位化交通電臺，即時發播城市交通資訊，車載設備透過 GPS 進行精確定位，結合電子地圖以及即時的交通狀況，自動匹配最優路徑，並實行車輛的自主導航。 民航運輸透過 GPS 接收設備，使駕駛員著陸時能準確對準跑道，同時還能使飛機緊湊排列，提高機場利用率，引導飛機安全進離場。

3、GPS 應用於救援

利用 GPS 定位技術，可對火警、救護、員警進行應急調遣，提高緊急事件處理部門對火災、犯罪現場、交通事故、交通堵塞等緊急事件的回應效率。特種車輛（如運鈔車）等，可對突發事件進行報警、定位，將損失降到最低。 有了 GPS 的幫助，救援人員就可在人跡罕至、條件惡劣的大海、山野、沙漠，對失蹤人員實施有效的搜索、拯救。裝有 GPS 裝置的漁船，在發生險情時，可及時定位、報警，使之能更快更即使地獲得救援。

4、GPS 應用於農業

當前，發達國家已開始把 GPS 技術引入農業生產，即所謂的"精准農業耕作"。該方法利用 GPS 進行農田資訊定位獲取，包括產量監測、土樣採集等，電腦系統透過對資料的分析處理，決策出農田地塊的管理措施，把產量和土壤狀態資訊裝入帶有 GPS 設備的噴施器中，從而精確地給農田地塊施肥、噴藥。透過實施精准耕作，可在儘量不減產的情況下，降低農業生產成本，有效避免資源浪費，降低因施肥除蟲對環境造成的污染。

5、GPS 應用於娛樂消遣

隨著 GPS 接收機的小型化以及價格的降低，GPS 逐漸走進了人們的日常生活，成爲人們旅遊、探險的好幫手。透過 GPS，人們可以在陌生的城市裏迅速地找到目的地，並且可以最優的路徑行駛；野營者攜帶 GPS 接收機，可快捷地找到合適的野營地點，不必擔心迷路；甚至一些高檔的電子遊戲，也使用了 GPS 仿真技術。

2-2.2 應用劃分

A、 按其作用分：

◆ **GPS 應用於導航**

主要是爲船舶、汽車、飛機等運動物體進行定位導航。例如：

1.船舶遠洋導航和進港引水

2.飛機航路引導和進場降落

3.汽車自主導航

4.地面車輛跟蹤和城市智慧交通管理

5.緊急救生

6.個人旅遊及野外探險

7.個人通訊終端(與行動電話、PDA、電子地圖等集成一體)

◆ **GPS 應用于授時校頻**

1.電力、郵電、通訊等網路的時間同步

2.準確時間的授入

3.準確頻率的授入

◆ **GPS 應用于高精度測量**

1.各種等級的大地測量、控制測量

2.道路和各種線路放樣

3.水下地形測量

4.地殼形變測量、大壩和大型建築物變形監測

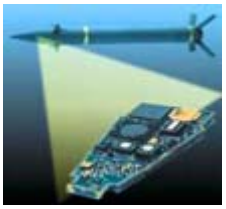
5.GIS 應用

6.工程機械（輪胎吊、推土機等）控制

7.精細農業

B、按其應用領域分：

a. 軍事用途

		
美國海軍核潛艇	配備 GPS 的士兵	GPS 導航的艦載飛彈

b. 民間用途

		
車輛導航管理	對航空器的定位及導航	農業監控

		
聖母峰的隆起研究	土地測量工程	森林調查、資源管理
		
生態研究給 動物戴的 GPS 項圈	配備 GPS 的巡警	自助旅遊

如人們所說：“GPS 的應用，僅受人們的想像力制約。”GPS 問世以來，已充分顯示了其在導航，定位領域的霸主地位。許多領域也由於 GPS 的出現而產生革命性變化。目前，幾乎全世界所有需要導航，定位的用戶，都被 GPS 的高精度、全天候、全球覆蓋、方便靈活和優質價廉所吸引。

我國的 GPS 應用發展勢頭迅猛，短短幾年，GPS 在我國的應用已從少數科研單位和軍用部門迅速擴展到各個民用領域，GPS 的廣泛應用改變人們的工作方式，提高了工作效率，帶來了巨大的經濟效益。可以說，GPS 在我國的應用前景是無限的。

ACR122S Serial NFC Reader

Technical Specifications

Table of Contents

1.0. Introduction

The ACR122S is the serial interface (RS-232) extension of the ACR122 Series, which is a family of NFC contactless smart card readers/writers. Developed based on the 13.56 MHz RFID technology and the ISO/IEC 18092 NFC standard, it supports not only Mifare and ISO 14443 Type A and B cards but also FeliCa and NFC tags.

ACR122S is equipped with a buzzer and two LEDs as well for rich user interaction. It also supports anti-collision and selective card polling, allowing smooth operation even when multiple cards are presented. Moreover, it is equipped with a built-in SAM slot to secure the overall contactless operation.

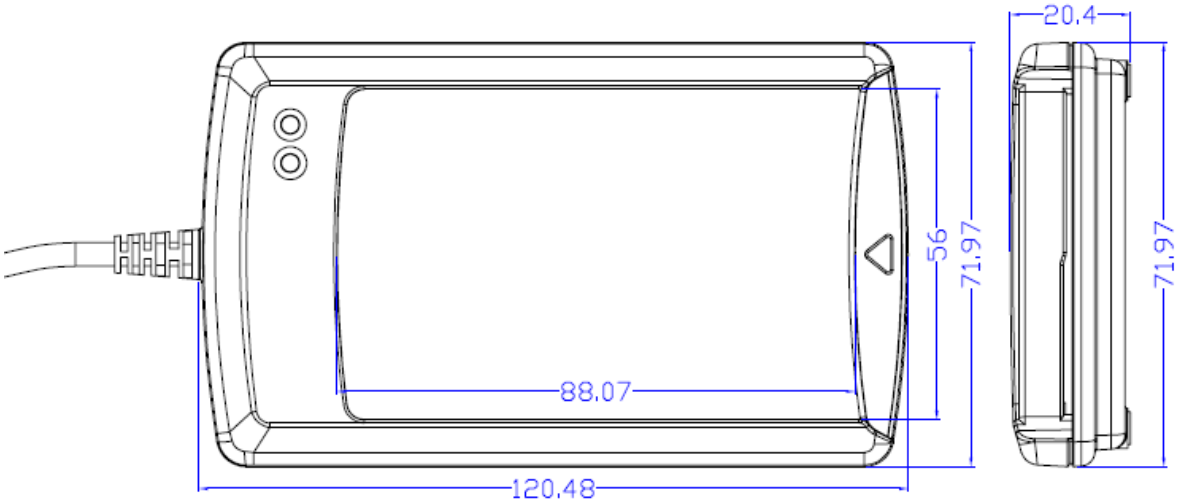
2.0. Features

- Serial Interface. Baud Rate = 9600 bps (default) or 115200 bps, 8-N-1. Initial Baud Rate is determined by the existence of R12. A command is also provided for changing the baud rate while the reader is running.
- USB interface for power supply
- CCID-like frame format (Binary format)
- Read/write speed up to 424 kbps
- Built-in antenna for contactless tag access, with card reading distance of up to 50 mm (depending on tag type)
- Supports ISO 14443 Part 4 Type A and B, Mifare, FeliCa and all 4 types of NFC (ISO/IEC 18092) tags
- Supports new Mifare Ultralight C and Mifare Plus SL1 (4 Byte UID) and SL3
- Supports all 3 modes of NFC: reader, card emulation and peer-to-peer modes
- Built-in anti-collision feature (at least 1 card is detected when multiple cards are presented)
- Selective card polling capability (especially useful when multiple cards are presented)
- SAM socket supports ISO 7816 T=0 cards
- User-controllable bi-color LED
- User-controllable buzzer
- OEM PCBA module version (*Upon request*)
- RS485 interface for data transmission (*Upon request*)
- PS/2 or DC power adaptor for power supply (*Upon request*)
- Relay (*Upon request*)
- Additional 2 SAM slots (*Upon request*)

3.0. Typical Applications

- Automatic Fare Collection
- Vending Machines
- Public Transportation Terminals
- Customer Loyalty
- Time and Attendance
- Contactless Public Phones
- Network Access Control
- Physical Access Control

4.0. Technical Specifications



Serial Interface

Power source ..... From USB interface
Speed ..... 9.6 kbps, 115.2 kbps
Supply Voltage ..... Regulated 5 V DC
Supply Current ..... 200 mA (maximum); 50 mA (standby); 100 mA (normal)

Contactless Smart Card Interface

Standard ..... Mifare, ISO 14443-4 Type A & B, FeliCa, ISO/IEC 18092 NFC
Operating Frequency ..... 13.56 MHz
Smart Card Read / Write Speed.... ..... 106, 212, 424 kbps

SAM Card Interface

Standard ..... ISO 7816
Protocol ..... T=0 protocol
Operating Frequency ..... 4 MHz
Smart Card Read / Write Speed ..... 9600 – 115200 bps

Physical Specifications

Dimensions ..... 120.5 mm (L) x 72.0 mm (W) x 20.4 mm (H)
Color ..... Metallic Blue
Operating Distance ..... Up to 50 mm (depending on tag type)

Built-In Peripherals

User controllable bi-color LED ..... Bi-color LED, Red and Green
Buzzer ..... Monotone

Operating Conditions

Temperature ..... 0 - 50° C
Humidity ..... 10% - 80%

Cable Connector

Length ..... 1.5 m (DB9 + USB)

Certifications/Compliance

CE, FCC, VCCI, RoHS Compliant

Device Driver Operating System Support

Windows ® 98, ME, 2000, Server 2003, XP, Vista, Server 2008, Server 2008 R2, 7
Windows CE ® 5.0
Linux, Mac

ACR122S Serial NFC Reader

Application Programming Interface

Table of Contents

Figures

1.0. Introduction

This manual describes the use of ACR122S interface software to facilitate application development with the ACR122S reader. This interface software is supplied in the form of 32-bit and 64-bit DLL (Dynamic Link Library) which can be programmed using popular development tools like Java, Delphi, Visual Basic, Visual C++, Visual C# and Visual Basic .Net.

ACR122S can be connected to the PC via the RS232 interface.

The architecture of the ACR122S library can be visualized as the following diagram:

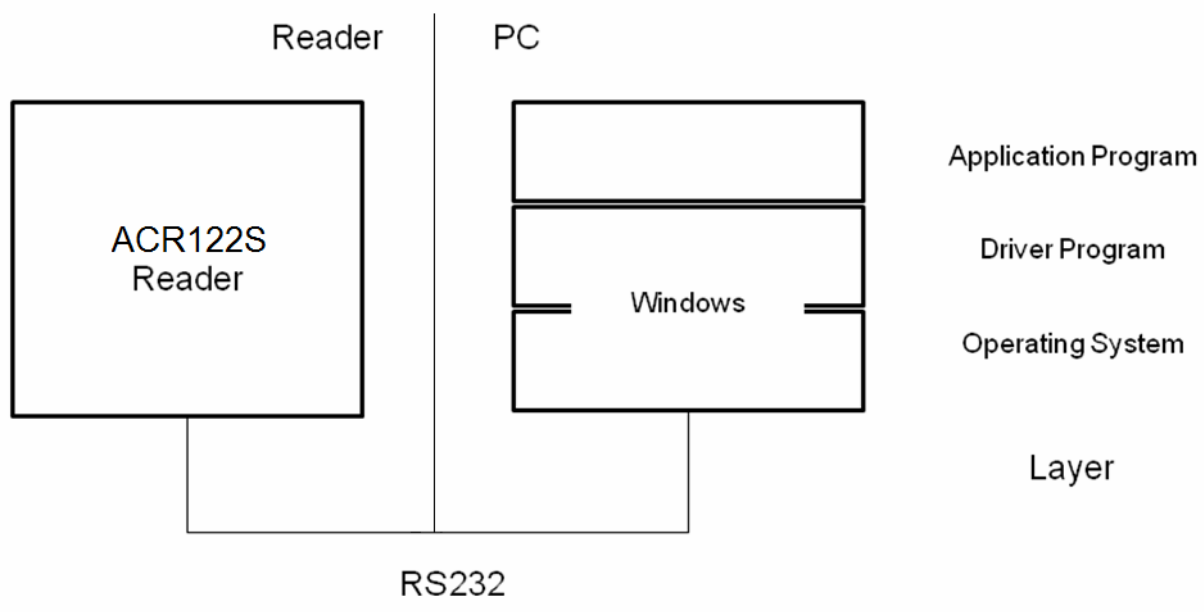


Figure 1: ACR122S Library Architecture

2.0. ACR122S API

2.1. Overview

The ACR122S DLL is a set of high-level functions provided for the application software to use. It supplies a consistent API (Application Programming Interface) for the application to operate on the ACR122S and the corresponding presented card. The DLL communicates with the ACR122S via the communication port facilities provided by the operating system.

The ACR122S API defines a common way of accessing the ACR122S. Application programs invoke the ACR122S through the interface functions and perform operations on the presented card.

The header file ACR122.h is available for the program developer, which contains all the function prototypes and macros described below.

2.2. Reader

2.2.1. Define Documentation

2.2.1.1. ACR122_GetFirmwareVersion and ACR122_GetFirmwareVersionA

Format:

```
#define ACR122_GetFirmwareVersion ACR122_GetFirmwareVersionA
```

Define Description:

`ACR122_GetFirmwareVersion` will be mapped to `ACR122_GetFirmwareVersionW()` function if UNICODE is defined.

Otherwise, it will be mapped to `ACR122_GetFirmwareVersionA()` function.

2.2.1.2. ACR122_Open and ACR122_OpenA

Format:

```
#define ACR122_Open ACR122_OpenA
```

Define Description:

`ACR122_Open` will be mapped to `ACR122_OpenW()` function if UNICODE is defined.

Otherwise, it will be mapped to `ACR122_OpenA()` function.

2.2.2. Function Documentation

2.2.2.1. ACR122_OpenA

Format:

```
DWORD WINAPI ACR122_OpenA ( LPCSTR   portName,
                            LPHANDLE phReader
                          )
```

Function Description: Open reader (ANSI).

This function opens a reader and returns a handle value as reference.

Parameters	Description
`[in] portName`	Port name. "\\.\COM1" means that the reader is connected to COM1 in Windows.
`[out] phReader`	Pointer to the `HANDLE` variable.

Function Description: Open reader (ANSI) (cont.)

Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

2.2.2.2. ACR122_OpenW

Format:

```
DWORD WINAPI ACR122_OpenW ( LPCWSTR  portName,
                            LPHANDLE phReader
                          )
```

Function Description: Open reader (Unicode).

This function opens a reader and returns a handle value as reference.

Parameters	Description	
[in] portName	Port name. "\\.\COM1" means that the reader is connected to COM1 in Windows.	
[out] phReader	Pointer to the HANDLE variable.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

2.2.2.3. ACR122_Close

Format:

```
DWORD WINAPI ACR122_Close ( HANDLE hReader
                          )
```

Function Description: Close reader.

This function closes the reader and releases the resources.

Parameters	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

2.2.2.4. ACR122_GetNumSlots

Format:

```
DWORD WINAPI ACR122_GetNumSlots ( HANDLE  hReader,
                                  LPDWORD pNumSlots
                                )
```

Function Description: Get number of slots.

This function retrieves the number of slots.

Parameters	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[out] pNumSlots	Pointer to a DWORD variable in which the number of slots is returned.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

2.2.2.5. ACR122_GetBaudRate

Format:

```
DWORD WINAPI ACR122_GetBaudRate ( HANDLE  hReader,
                                  LPDWORD pBaudRate
                                )
```

Function Description: Get baud rate.

This function retrieves the baud rate of reader.

Parameters	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[out] pBaudRate	Pointer to a DWORD variable in which the baud rate is returned.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

2.2.2.6. ACR122_SetBaudRate

Format:

```
DWORD WINAPI ACR122_SetBaudRate ( HANDLE hReader,
                                  DWORD baudRate
                                )
```

Function Description: Set baud rate.

This function sets the communication baud rate of reader. Actually, the reader supports 9600 bps and 115200 bps.

Parameters	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[in] baudRate	Baud rate must be 9600 bps or 115200 bps.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

2.2.2.7. ACR122_GetTimeouts

Format:

```
DWORD WINAPI ACR122_GetTimeouts ( HANDLE          hReader,
                                  PACR122_TIMEOUTS pTimeouts
                                )
```

Function Description: Get timeouts.

This function retrieves the timeout parameters for status and response operations of the reader.

Parameters	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[out] pTimeouts	Pointer to a ACR122_TIMEOUTS structure in which the timeout information is returned.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

Note: For PACR122_TIMEOUTS, please see **_ACR122_TIMEOUTS Struct Reference**

2.2.2.8. ACR122_SetTimeouts

Format:

```
DWORD WINAPI ACR122_SetTimeouts ( HANDLE                hReader,
                                  const PACR122_TIMEOUTS pTimeouts
                                )
```

Function Description: Set timeouts.

This function sets the timeout parameters for status and response operations on the reader.

Parameters	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[in] pTimeouts	Pointer to a ACR122_TIMEOUTS structure that contains the new timeout values	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

Note: For PACR122_TIMEOUTS, please see **_ACR122_TIMEOUTS Struct Reference**

2.2.2.9. ACR122_GetFirmwareVersionA

Format:

```
DWORD WINAPI ACR122_GetFirmwareVersionA ( HANDLE  hReader,
                                          DWORD   slotNum,
                                          LPSTR   firmwareVersion,
                                          LPDWORD pFirmwareVersionLen
                                        )
```

Function Description: Get firmware version (ANSI).

This function retrieves the firmware version in ANSI string of the slot.

Parameters	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[in] slotNum	Slot number.	
[out] firmwareVersion	A pointer to the buffer that receives the firmware version returned from the reader.	
[in,out] pFirmwareVersionLen	The length in number of bytes of the firmware version parameter and receives the actual number of bytes received from the reader.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

2.2.2.10. ACR122_GetFirmwareVersionW

Format:

```
DWORD WINAPI ACR122_GetFirmwareVersionW ( HANDLE  hReader,
                                          DWORD   slotNum,
                                          LPWSTR  firmwareVersion,
                                          LPDWORD pFirmwareVersionLen
                                        )
```

Function Description: Get firmware version (Unicode).

This function retrieves the firmware version in Unicode string of the slot.

Parameters	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[in] slotNum	Slot number.	
[out] firmwareVersion	A pointer to the buffer that receives the firmware version returned from the reader.	
[in,out] pFirmwareVersionLen	The length in number of bytes of the firmware version parameter and receives the actual number of bytes received from the reader.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

2.3. LED

2.3.1. Function Documentation

2.3.1.1. ACR122_SetLedStatesWithBeep

Format:

```
DWORD WINAPI ACR122_SetLedStatesWithBeep ( HANDLE              hReader,
                                           PACR122_LED_CONTROL controls,
                                           DWORD               numControls,
                                           DWORD               t1,
                                           DWORD               t2,
                                           DWORD               numTimes,
                                           DWORD               buzzerMode
                                         )
```

Function Description: Set LED states with beep.

This function controls LED0, LED1 and buzzer operation on the reader.

<table>
<tr><th>Parameters</th><th colspan="2">Description</th></tr>
<tr><td>[in] hReader</td><td colspan="2">A reference value returned from ACR122_Open() function.</td></tr>
<tr><td>[in] controls</td><td colspan="2">A pointer to the array of ACR122_LED_CONTROL data structure.</td></tr>
<tr><td>[in] numControls</td><td colspan="2">Number of controls must be 2.</td></tr>
<tr><td>[in] t1</td><td colspan="2">T1 in milliseconds. The value must be from 0 to 25500.</td></tr>
<tr><td>[in] t2</td><td colspan="2">T2 in milliseconds. The value must be from 0 to 25500</td></tr>
<tr><td>[in] numTimes</td><td colspan="2">Number of times. The values must be from 0 to 255.</td></tr>
<tr><td>[in] buzzerMode</td><td colspan="2">A bitmask of buzzer mode. Possible values may be combined with the OR operation.
<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>ACR122_BUZZER_MODE_OFF</td><td>The buzzer will not turn on.</td></tr><tr><td>ACR122_BUZZER_MODE_ON_T1</td><td>The buzzer will turn on during T1 duration.</td></tr><tr><td>ACR122_BUZZER_MODE_ON_T2</td><td>The buzzer will turn on during T2 duration.</td></tr></table></td></tr>
<tr><td rowspan="2">Return Value</td><td>ERROR_SUCCESS</td><td>The operation completed successfully.</td></tr>
<tr><td>Failure</td><td>An error code. See Windows API error codes and ACR122 error codes.</td></tr>
</table>

Note: For PACR122_LED_CONTROL, please see **_ACR122_LED_CONTROL Struct Reference**

2.4. Card

2.4.1. Function Documentation

2.4.1.1. ACR122_DirectTransmit

Format:

```
DWORD WINAPI ACR122_DirectTransmit ( HANDLE       hReader,
                                     const LPBYTE sendBuffer,
                                     DWORD        sendBufferLen,
                                     LPBYTE       recvBuffer,
                                     LPDWORD      pRecvBufferLen
                                   )
```

Function Description: Direct transmit TAG command.

This function sends TAG command and receives response from the reader.

Parameters	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[in] sendBuffer	A pointer to the actual data to be written to the card.	
[in] sendBufferLen	The length in number of bytes of the sendBuffer parameter.	
[in] recvBuffer	A pointer to any data returned from the card.	
[in,out] pRecvBufferLen	The length in number of bytes of the recvBuffer parameter and receives the actual number of bytes received from the card.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

2.4.1.2. ACR122_ExchangeApdu

Format:

```
DWORD WINAPI ACR122_ExchangeApdu ( HANDLE       hReader,
                                   DWORD        slotNum,
                                   const LPBYTE sendBuffer,
                                   DWORD        sendBufferLen,
                                   LPBYTE       recvBuffer,
                                   LPDWORD      pRecvBufferLen
                                 )
```

Function Description: Exchange APDU.

This function sends APDU command and receives APDU response from the card.

Parameters	Description	
`[in] hReader`	A reference value returned from `ACR122_Open()` function.	
`[in] slotNum`	Slot number.	
`[in] sendBuffer`	A pointer to the actual data to be written to the card.	
`[in] sendBufferLen`	The length in number of bytes of the `sendBuffer` parameter.	
`[out] recvBuffer`	A pointer to any data returned from the card.	
`[in,out] pRecvBufferLen`	The length in number of bytes of the `recvBuffer` parameter and receives the actual number of bytes received from the card.	
Return Value	`ERROR_SUCCESS`	The operation completed successfully.
	`Failure`	An error code. See Windows API error codes and ACR122 error codes.

2.4.1.3. ACR122_PowerOffIcc

Format:

```
DWORD WINAPI ACR122_PowerOffIcc ( HANDLE hReader,
                                  DWORD  slotNum
                                )
```

Function Description: Power off ICC in slot.

This function powers off the card in the slot.

Parameters	Description	
`[in] hReader`	A reference value returned from `ACR122_Open()` function.	
`[in] slotNum`	Slot number.	
Return Value	`ERROR_SUCCESS`	The operation completed successfully.
	`Failure`	An error code. See Windows API error codes and ACR122 error codes.

2.4.1.4. ACR122_PowerOnIcc

Format:

```
DWORD WINAPI ACR122_PowerOnIcc ( HANDLE  hReader,
                                 DWORD   slotNum,
                                 LPBYTE  atr,
                                 LPDWORD pAtrLen
                               )
```

Function Description: Power on ICC in slot.

This function powers on the card in the slot and returns the ATR string from the card.

Parameters	Description	
`[in] hReader`	A reference value returned from `ACR122_Open()` function.	
`[in] slotNum`	Slot number.	
`[out] atr`	A pointer to the buffer that receives the ATR string returned from the card.	
`[in,out] pAtrLen`	The length in number of bytes of the atr parameter and receives the actual number of bytes received from the card.	
Return Value	`ERROR_SUCCESS`	The operation completed successfully.
	`Failure`	An error code. See Windows API error codes and ACR122 error codes.

Appendix A. Data Structures

Appendix A.1. _ACR122_TIMEOUTS Struct Reference

This data structure is used in **`ACR122_GetTimeouts()`** and **`ACR122_SetTimeouts()`** function.

- **`DWORD _ACR122_TIMEOUTS::numResponseRetries`**

  Number of response retries.

  Default is 1.

- **`DWORD _ACR122_TIMEOUTS::numStatusRetries`**

  Number of status retries.

  Default is 1.

- **`DWORD _ACR122_TIMEOUTS::responseTimeout`**

  Response timeout in milliseconds.

  Default is 10000 ms

- **`DWORD _ACR122_TIMEOUTS::statusTimeout`**

  Status timeout in milliseconds.

  Default is 2000 ms.

Appendix A.2. _ACR122_LED_CONTROL Struct Reference

This data structure is used in **`ACR122_SetLedStatesWithBeep()`** function.

- **`BOOL _ACR122_LED_CONTROL::blinkEnabled`**

  Enable blink.

  Set to `TRUE` to enable blink. Otherwise, set to `FALSE`.

- **`DWORD _ACR122_LED_CONTROL::finalState`**

  Final state.

  Possible values are `ACR122_LED_STATE_OFF` and `ACR122_LED_STATE_ON`.

- **`DWORD _ACR122_LED_CONTROL::initialBlinkingState`**

  Initial blinking state.

  Possible values are `ACR122_LED_STATE_OFF` and `ACR122_LED_STATE_ON`.

- **`BOOL _ACR122_LED_CONTROL::updateEnabled`**

  Enable update.

  Set to `TRUE` to update the state. Otherwise, set to `FALSE` to keep the state unchanged.

Appendix B. Error Codes Returned by High-Level APIs

- **ACR122_ERROR_NO_MORE_HANDLES ((DWORD) 0x20000001L)**

  The handle is invalid.

- **ACR122_ERROR_UNKNOWN_STATUS ((DWORD) 0x20000002L)**

  Reader unknown error.

- **ACR122_ERROR_OPERATION_FAILURE ((DWORD) 0x20000003L)**

  Operation failed.

- **ACR122_ERROR_OPERATION_TIMEOUT ((DWORD) 0x20000004L)**

  Timeout operation.

- **ACR122_ERROR_INVALID_CHECKSUM ((DWORD) 0x20000005L)**

  Checksum calculation error

- **ACR122_ERROR_INVALID_PARAMETER ((DWORD) 0x20000006L)**

  Incorrect parameter input.

ACR122S Serial NFC Reader

Communication Protocol

Table of Contents

Figures

Tables

1.0. Introduction

The ACR122S is a contactless smart card reader/writer used for accessing ISO 14443-4 Type A and B, Mifare, ISO 18092 or NFC, and FeliCa tags using the Serial Interface. This document will discuss the command set in implementing a smart card application using the ACR122S.

1.1. Serial Interface

The ACR122S is connected to a Host through the RS232C Serial Interface at 9600 bps, 8-N-1.

Pin	Signal	Function
1	VCC	+5V power supply for the reader (Max 200mA, Normal 100mA)
2	TXD	The signal from the reader to the host
3	RXD	The signal from the host to the reader
4	GND	Reference voltage level for power supply

Table 1: PIN Configuration

1.2. Bi-Color LED

A user-controllable bi-color LED with red and green color is provided.

- The green color LED will be blinking if the “Card Interface” is not connected.
- The green color LED will be turned on if the “Card Interface” is connected.
- The green color LED will be flashing if the “Card Interface” is operating.
- The red color LED is controlled by the application only.

1.3. Buzzer

A user-controllable buzzer with a default state of OFF is provided.

1.4. SAM Interface

One SAM socket is provided.

1.5. Built-In Antenna

A 3-turns symmetric loop antenna, center-tapped, is provided.

- Estimated size is 60 mm x 48 mm.
- Loop inductance is approximately 1.6 uH to 2.5 uH.
- Operating distance for different tags is approximately up to 50 mm (depends on the tag).
- Only one tag can be accessed at any one time.

2.0. Communication between the Host and the Contactless Interface, SAM and Peripherals

The Contactless interface and peripherals are accessed through the use of pseudo-APDUs.

The SAM interface is accessed through the use of standard APDUs.

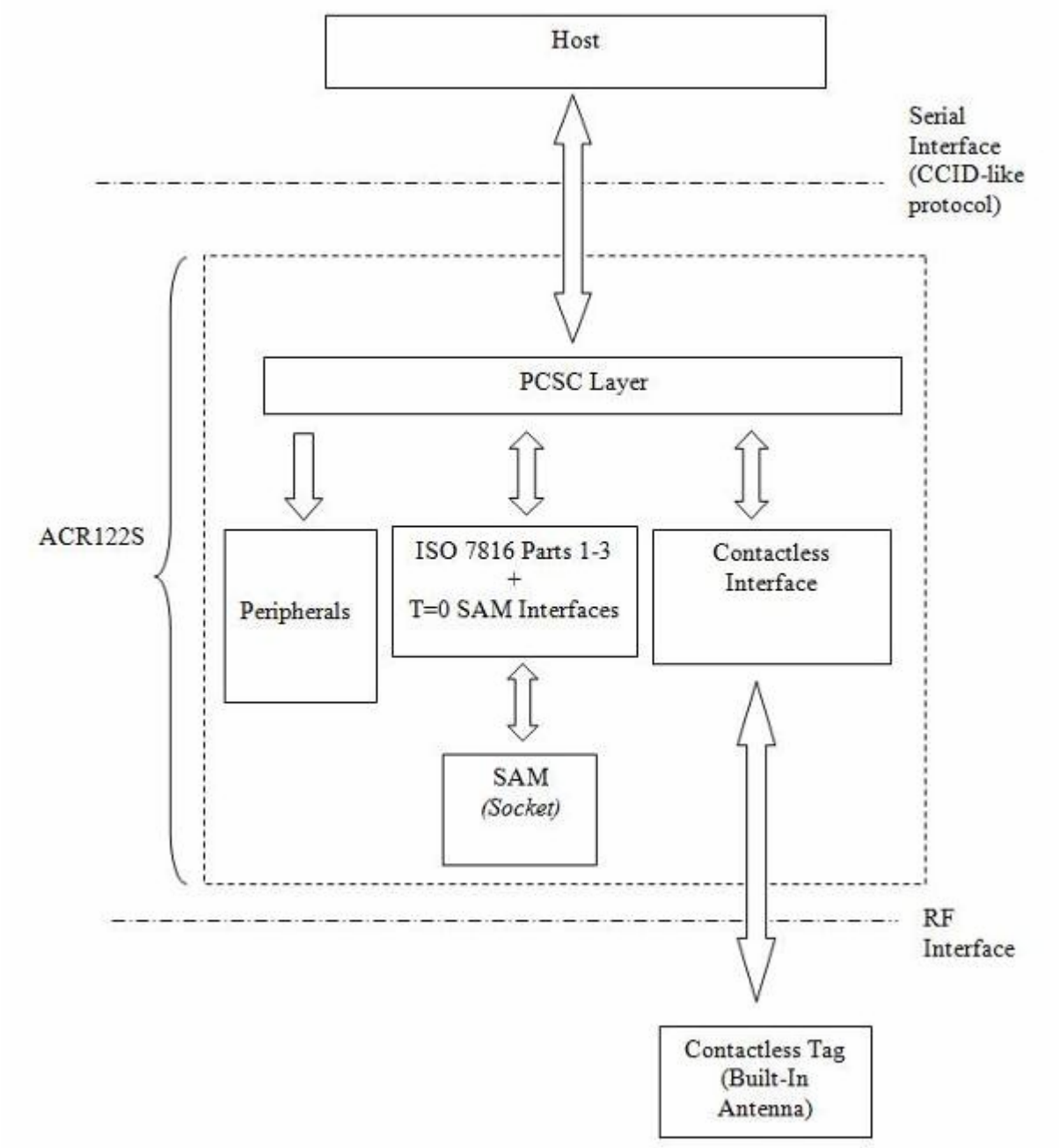


Figure 1: Communication Flowchart of ACR122S

3.0. Serial Interface (CCID-Like Frame Format)

Communication setting: 9600 bps, 8-N-1

The communication protocol between the Host and ACR122S is very similar to the CCID protocol.

STX (0x02)	Bulk-OUT Header	APDU Command Or Parameters	Checksum	ETX (0x03)
1 Byte	10 Bytes	M Bytes (If applicable)	1 Byte	1 Byte

Table 2: ACR122S Command Frame Format

STX (0x02)	Status	Checksum	ETX (0x03)
1 Byte	1 Byte	1 Byte	1 Byte

Table 3: ACR122S Status Frame Format

STX (0x02)	Bulk-IN Header	APDU Response Or abData	Checksum	ETX (0x03)
1 Byte	10 Bytes	N Bytes (If applicable)	1 Byte	1 Byte

Table 4: ACR122S Response Frame Format

```
Checksum = XOR {Bulk-OUT Header, APDU Command or Parameters}
Checksum = XOR {Bulk-IN Header, APDU Response or abData}
```

In general, we would make use of three types of `Bulk-OUT` Header:

- `HOST_to_RDR_IccPowerOn`: To activate the SAM interface. The ATR of the SAM will be returned if available.
- `HOST_to_RDR_IccPowerOff`: To deactivate the SAM interface.
- `HOST_to_RDR_XfrBlock`: To exchange APDUs between the Host and ACR122S.

The SAM interface must be activated in order to use the Contactless interface and Peripherals. In short, all the APDUs are exchanged through the SAM Interface.

Similarly, two types of `Bulk-IN` Header are used:

- `RDR_to_HOST_DataBlock`: In response to the "`HOST_to_RDR_IccPowerOn`" and "`HOST_to_RDR_XfrBlock`" Frames.
- `RDR_to_HOST_SlotStatus`: In response to the "`HOST_to_RDR_IccPowerOff`" Frame.

```
RDR = ACR122S; HOST = Host Controller
HOST_to_RDR = Host Controller -> ACR122S
RDR_to_HOST = ACR122S -> Host Controller
```

3.1. Protocol Flow Examples

1) Activate a SAM

	HOST		RDR
1. HOST sends a frame	→	02 62 00 00 00 00 00 01 01 00 00 [Checksum] 03	
2. RDR sends back a positive status frame immediately		02 00 00 03 (positive status frame)	←
3. RDR sends back the response of the command		.. after some processing delay .. 02 80 0D 00 00 00 00 01 00 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [Checksum] 03	←

2) Activate a SAM (Incorrect Checksum, HOST)

	HOST		RDR
1. HOST sends a corrupted frame	→	02 62 00 00 00 00 00 01 01 00 00 [Incorrect Checksum] 03	
2. RDR sends back a negative status frame immediately		02 FF FF 03 (negative status frame)	←
3. HOST sends the frame again.	→	02 62 00 00 00 00 00 01 01 00 00 [Checksum] 03	
4. RDR sends back a positive status frame immediately		02 00 00 03 (positive status frame)	←
5. RDR sends back the response of the command		.. after some processing delay .. 02 80 0D 00 00 00 00 01 00 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [Checksum] 03	←

3) Activate a SAM (Incorrect Checksum, RDR)

	HOST		RDR
1. HOST sends a frame	→	02 62 00 00 00 00 00 01 01 00 00 [Checksum] 03	
2. RDR sends back a positive status frame immediately		02 00 00 03 (positive status frame) .. after some processing delay ..	←
3. RDR sends back the response (corrupted) of the command	→	02 80 0D 00 00 00 00 01 00 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [Incorrect Checksum] 03	←
4. HOST sends a NAK frame to get the response again.		02 00 00 00 00 00 00 00 00 00 00 03 (NAK)	
5. RDR sends back the response of the command		02 80 0D 00 00 00 00 01 00 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [Checksum] 03	←

Note:

If the frame sent by the HOST is correctly received by the RDR, a positive status frame = {02 00 00 03} will be sent to the HOST immediately to inform the HOST the frame is correctly received. The HOST has to wait for the response of the command. The RDR will not receive any more frames while the command is being processed.

In case of errors, a negative status frame will be sent to the HOST to indicate the frame is either corrupted or incorrectly formatted.

- CheckSum Error Frame = {02 FF FF 03}

- Length Error Frame = {02 FE FE 03}. The length "dDwLength" is greater than 0x0105 bytes.

- ETX Error Frame = {02 FD FD 03}. The last byte is not equal to ETX "0x03".

The NAK Frame is only used by the HOST to get the last response.

{02 00 00 00 00 00 00 00 00 00 00 00 03} // 11 zeros

4.0. SAM Interface

The ACR122S comes with a SAM interface.

4.1. To Activate the SAM Interface

STX (0x02)	Bulk-OUT Header (HOST_to_RDR_IccPowerOn)	Parameters	Checksum	ETX (0x03)
1 Byte	10 Bytes	0 Byte	1 Byte	1 Byte

Table 5: ACR122S Command Frame Format

Offset	Field	Size	Value	Description
0	bMessageType	1	62h	
1	dDwLength <LSB .. MSB>	4	00000000h	Message-specific data length
5	bSlot	1	00-FFh	Identifies the slot number for this command. Default=00h
6	bSeq	1	00-FFh	Sequence number for command
7	bPowerSelect	1	00h, 01h, 02h, or 03h	Voltage that is applied to the ICC 00h – Automatic Voltage Selection 01h – 5.0 volts 02h – 3.0 volts 03h – 1.8 volts
8	abRFU	2		Reserved for Future Use

Table 6: HOST_to_RDR_IccPowerOn Format

STX (0x02)	Bulk-IN Header (RDR_to_HOST_DataBlock)	abData	Checksum	ETX (0x03)
1 Byte	10 Bytes	N Bytes (ATR)	1 Byte	1 Byte

Table 7: ACR122S Response Frame Format

Offset	Field	Size	Value	Description
0	bMessageType	1	80h	Indicates that a data block is being sent from the ACR122S
1	dwLength <LSB .. MSB>	4	N	Size of abData field. (N Bytes)
5	bSlot	1	Same as Bulk-OUT	Identifies the slot number for this command
6	bSeq	1	Same as Bulk-OUT	Sequence number for corresponding command
7	bStatus	1		
8	bError	1		
9	bChainParameter	1		

Table 8: RDR_to_HOST_DataBlock Format

Example: To activate the slot 0 (default), sequence number = 1, 5V card.

HOST -> 02 62 00 00 00 00 00 01 01 00 00 [Checksum] 03

RDR -> 02 00 00 03

RDR -> 02 80 0D 00 00 00 00 01 00 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [Checksum] 03

The ATR = 3B 2A 00 80 65 24 B0 00 02 00 82; SW1 SW2 = 90 00

4.2. To Deactivate the SAM Interface

STX (0x02)	Bulk-OUT Header (HOST_to_RDR_IccPowerOff)	Parameters	Checksum	ETX (0x03)
1 Byte	10 Bytes	0 Byte	1 Byte	1 Byte

Table 9: ACR122S Command Frame Format

Offset	Field	Size	Value	Description
0	bMessageType	1	63h	
1	dDwLength <LSB .. MSB>	4	00000000h	Message-specific data length
5	bSlot	1	00-FFh	Identifies the slot number for this command. Default=00h
6	bSeq	1	00-FFh	Sequence number for command
7	abRFU	3		Reserved for Future Use

Table 10: HOST_to_RDR_IccPowerOff Format

STX (0x02)	Bulk-IN Header (RDR_to_HOST_SlotStatus)	abData	Checksum	ETX (0x03)
1 Byte	10 Bytes	0 Byte	1 Byte	1 Byte

Table 11: ACR122S Response Frame Format

Offset	Field	Size	Value	Description
0	bMessageType	1	81h	Indicates that a data block is being sent from the ACR122S
1	dwLength <LSB .. MSB>	4	0	Size of abData field. (0 Bytes)
5	bSlot	1	Same as Bulk-OUT	Identifies the slot number for this command
6	bSeq	1	Same as Bulk-OUT	Sequence number for corresponding command
7	bStatus	1		
8	bError	1		
9	bClockStatus	1		

Table 12: RDR_to_HOST_DataBlock Format

Example: To deactivate the slot 0 (default), sequence number = 2.

```
HOST -> 02 63 00 00 00 00 00 02 00 00 00 [Checksum] 03
RDR -> 02 00 00 03
RDR -> 02 81 00 00 00 00 00 02 00 00 00 [Checksum] 03
```

4.3. To Do Data Exchange Through the SAM Interface

STX (0x02)	Bulk-OUT Header (HOST_to_RDR_XfrBlock)	Parameters	Checksum	ETX (0x03)
1 Byte	10 Bytes	M Byte	1 Byte	1 Byte

Table 13: ACR122S Command Frame Format

Offset	Field	Size	Value	Description
0	bMessageType	1	6Fh	
1	dDwLength <LSB .. MSB>	4	M	Message-specific data length
5	bSlot	1	00-FFh	Identifies the slot number for this command. Default=00h
6	bSeq	1	00-FFh	Sequence number for command
7	bBWI	1	00-FFh	Used to extend the Block Waiting Timeout.
8	wLevelParameter	2	0000h	

Table 14: HOST_to_RDR_XfrBlock Format

STX (0x02)	Bulk-IN Header (RDR_to_HOST_DataBlock)	abData	Checksum	ETX (0x03)
1 Byte	10 Bytes	N Bytes (ATR)	1 Byte	1 Byte

Table 15: ACR122S Response Frame Format

Offset	Field	Size	Value	Description
0	bMessageType	1	80h	Indicates that a data block is being sent from the ACR122S
1	dwLength <LSB .. MSB>	4	N	Size of abData field. (N Bytes)
5	bSlot	1	Same as Bulk-OUT	Identifies the slot number for this command
6	bSeq	1	Same as Bulk-OUT	Sequence number for corresponding command
7	bStatus	1		
8	bError	1		
9	bChainParameter	1		

Table 16: RDR_to_HOST_DataBlock Format

Example: To send an APDU "80 84 00 00 08" to the slot 0 (default), sequence number = 3.

HOST -> 02 6F 05 00 00 00 00 03 00 00 00 80 84 00 00 08 [Checksum] 03

RDR -> 02 00 00 03

RDR -> 02 80 0A 00 00 00 00 03 00 00 00 E3 51 B0 FC 88 AA 2D 18 90 00 [Checksum] 03

Response = E3 51 B0 FC 88 AA 2D 18; SW1 SW2 = 90 00

5.0. Pseudo-APDUs for Contactless Interface and Peripherals Control

ACR122S comes with two primitive commands for this purpose. <Class 0xFF>

5.1. Direct Transmit

To send a Pseudo-APDU (TAG Commands), and the length of the Response Data will be returned.

Command	Class	INS	P1	P2	Lc	Data In	
Direct Transmit	0xFF	0x00	0x00	0x00	Number of Bytes to send	TAG Command	Data

Table 17: Direct Transmit Command Format (Length of the Tag Command + 5 Bytes)

Lc: Number of Bytes to Send (1 Byte)

Maximum 255 bytes

Data In: TAG Command

The data to be sent to the Tag.

Item	Command	Data			Meaning
1	D4 40	Tg		[DataOut[]]	Tag Exchange Data
2	D4 4A	MaxTg	BrTy	[InitiatorData[]]	Tag Polling

Table 18: Direct Transmit Response Format (Tag Response + Data + 2 Bytes)

Tg:

A byte containing the logical number of the relevant target. This byte also contains the *More Information* (MI) bit (bit 6). When the MI bit is set to 1, this indicates that the host controller wants to send more data which is all the data contained in the **DataOUT[]**array. This bit is only valid for a TPE target.

DataOut:

An array of raw data (from 0 up to 262 bytes) to be sent to the target by the contactless chip.

MaxTg:

Maximum number of targets to be initialized by the contactless chip. The chip is capable of handling 2 targets maximum at once, so this field should not exceed 0x02.

Brty:

Baud rate and the modulation type to be used during the initialization

-0x00: 106 kbps type A (ISO/IEC14443 Type A),

-0x01: 212kbps (FeliCa polling),

-0x02: 424kbps (FeliCa polling),

-0x03: 106kbps type B (ISO/IEC 14443-3B),

-0x04: 106kbps Innovision Jewel tag.

InitiatorData[]:

An array of data to be used during the initialization of the target(s). Depending on the Baud Rate specified, the content of this field is different.

-106kbps type A:

The field is optional and is present only when the host controller wants to initialize a target with a known UID.

In that case, `InitiatorData[]` contains the UID of the card (or part of it). The UID must include the cascade tag CT if it is cascaded level 2 or 3.

Cascade Level 1

UID1	UID2	UID3	UID4

Cascade Level 2

UID1	UID2	UID3	UID4	UID5	UID6	UID7

Cascade Level 3

UID1	UID2	UID3	UID4	UID5	UID6	UID7	UID8	UID9	UID10

-106kbps type B:

In this case, `InitiatorData[]` is formatted as following:

AFI (1byte)	[Polling Method]

AFI:

The AFI (Application Family Identifier) parameter represents the type of application targeted by the device IC and is used to preselect the PICCs before the ATQB.

This field is mandatory.

Polling Method:

This field is optional. It indicates the approach to be used in the ISO/IEC 14443-3B initialization:

- If bit 0 = 1: Probabilistic approach (option 1) in the ISO/IEC 14443-3B initialization,
- If bit 0 = 0: Timeslot approach (option 2) in the ISO/IEC 14443-3B initialization,
- If this field is absent, the timeslot approach will be used.

212/424kbps:

In that case, this field is mandatory and contains the complete pay load information that should be used in the polling request command (5bypes, length bytes is excluded)

106kbps InnoVision Jewel tag:

This field is not used.

Data Out: TAG Response

TAG Response returned by the reader.

Response	Data Out				
Result	D5 41	Status	[DataIn[]]		SW1 SW2
	D5 4B	NbTg	[TargetData1[]]	[TargetData2[]]	

Table 19: Direct Transmit Response Format

Status:

A byte indicating if the process has been terminated successfully or not. When in either DEP or ISO/IEC 14443-4 PCD mode, this byte also indicates if *NAD (Node Address)* is used and if the transfer of data is not completed with bit *More Information*.

DataIn:

An array of raw data (from 0 up to 262 bytes) received by the contactless chip.

NbTg:

The number of initialized Targets (minimum 0, maximum 2 targets).

TargetDatai[]:

The “i” in TargetDatai[] refers to “1” or “2”. This contains the information about the detected targets and depends on the baud rate selected. The following information is given for one target, it is repeated for each target initialized (**NbTg** times).

-**106 kbps Type A:**

Tg	SENS_RES10 (2 bytes)	SEL_RES (1 byte)	NFCIDLength (1 byte)	NFCID1[] (NFCIDLength bytes)	[ATS[]] (ATSLength bytes11))

-**106 kbps Type B:**

Tg	ATQB Response (12 bytes)	ATTRIB_RES Length (1 byte)	ATTTRIB_RES[] (ATTRIB_RES Length)

-212/424 kbps:

Tg	POL_RES length	0x01 (response code)	NFCID2t	Pad	SYST_CODE (optional)
1 byte	1 byte	1 byte	8 bytes	8 bytes	2 bytes

POL_RES (18 or 20 bytes)

-106 kbps Innovision Jewel tag:

Tg	SENS_RES (2 bytes)	JEWELID[] (4 bytes)

Data Out: SW1 SW2

Status Code returned by the reader.

Results	SW1	SW2	Meaning
Success	90	00	The operation is completed successfully.
Error	63	00	The operation is failed.
Time Out Error	63	01	The TAG does not response.
Checksum Error	63	27	The checksum of the Response is wrong.
Parameter Error	63	7F	The TAG Command is wrong.

Table 20: Status Code

5.2. Pseudo-APDU for Changing the Communication Speed

This APDU is used to change the baud rate.

Command	Class	INS	P1	P2	Lc
Baud Rate Control	0xFF	0x00	0x44	New Baud Rate	0x00

Table 21: Baud Rate Control Command Format (9 Bytes)

P2: New Baud Rate

0x00: Set the new baud rate to 9600 bps.

0x01: Set the new baud rate to 115200 bps.

Data Out: SW1 SW2.

Results	SW1	SW2	Meaning
Success	90	Current Baud Rate	The operation is completed successfully.
Error	63	00	The operation is failed.

Table 22: Status Code

SW2: Current Baud Rate

0x00: The current baud rate is 9600 bps.

0x01: The current baud rate is 115200 bps.

Note:

After the communication speed is changed successfully, the program has to adjust its communication speed so as to continue the rest of the data exchanges.

The initial communication speed is determined by the existence of R12 (0 ohm).

- With R12 = 115200 bps
- Without R12 = 9600 bps (default)

Example 1: To initialize a FeliCa Tag (Tag Polling)

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 09 D4 4A 01 01 00 FF FF 01 00"

In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the Tag Command = "09"
Tag Command (InListPassiveTarget 212Kbps) = "D4 4A 01 01"
Tag Command (System Code Request) = "00 FF FF 01 00"

To send an APDU to the slot 0 (default), sequence number = 1.

```
HOST -> 02 6F 0E 00 00 00 00 01 00 00 00
FF 00 00 00 09 D4 4A 01 01 00 FF FF 01 00
[Checksum] 03

RDR -> 02 00 00 03

RDR -> 02 81 1A 00 00 00 00 01 00 00 00
D5 4B 01 01 14 01 01 01 05 01 86 04 02 02 03 00
4B 02 4F 49 8A 8A 80 08 90 00
[Checksum] 03
```

The APDU Response is
"D5 4B 01 01 14 01 01 01 05 01 86 04 02 02 03 00 4B 02 4F 49 8A 8A 80 08 90 00"

In which,
Response returned by the contactless chip = "D5 4B 01 01 14 01 01 01 05 01 86 04 02 02 03 00 4B 02 4F 49 8A 8A 80 08"
NFCID2t of the FeliCa Tag = "01 01 05 01 86 04 02 02"
Status Code returned by the reader = "90 00"

Example 2: To write 16 bytes data to the FeliCa Tag (Tag Write)

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 23 D4 40 01 20 08 01 01 05 01 86 04 02 02 01 09 01 01 80 00 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA"

In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the Tag Command = "23"
Tag Command (InDataExchange) = "D4 40 01"
Tag Command (Write Data) = "20 08 01 01 05 01 86 04 02 02 01 09 01 01 80 00 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA".

To send an APDU to the slot 0 (default), sequence number = 2.

```
HOST -> 02 6F 28 00 00 00 00 02 00 00 00
FF 00 00 00 00 23 D4 40 01 20 08 01 01 05 01 86
04 02 02 01 09 01 01 80 00 00 AA 55 AA 55 AA 55
AA 55 AA 55 AA 55 AA
[Checksum] 03

RDR -> 02 00 00 03

RDR  -> 02 81 11 00 00 00 00 02 00 00 00
D5 41 00 0C 09 01 01 05 01 86 04 02 02 00 00 90 00
[Checksum] 03
```

The APDU Response would be "D5 41 00 0C 09 01 01 05 01 86 04 02 02 00 00 90 00"

In which,
Response returned by the contactless chip = "D5 41"
Response returned by the FeliCa Tag = "00 0C 09 01 01 05 01 86 04 02 02 00 00"
Status Code returned by the reader = "90 00"

Example 3: To read 16 bytes data from the FeliCa Tag (Tag Write)

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 13 D4 40 01 10 06 01 01 05 01 86 04 02 02 01 09 01 01 80 00"

In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the Tag Command = "13"
Tag Command (InDataExchange) = "D4 40 01"
Tag Command (Read Data) = "10 06 01 01 05 01 86 04 02 02 01 09 01 01 80 00"

To send an APDU to the slot 0 (default), sequence number = 3.

```
HOST -> 02 6F 18 00 00 00 00 03 00 00 00
FF 00 00 00 13 D4 40 01 10 06 01 01 05 01 86 04
02 02 01 09 01 01 80 00 FF
[Checksum] 03

RDR -> 02 00 00 03

RDR  -> 02 81 22 00 00 00 00 03 00 00 00
D5 41 00 1D 07 01 01 05 01 86 04 02 02 00 00 01 00
AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 90 00
[Checksum] 03
```

The APDU Response would be
"D5 41 00 1D 07 01 01 05 01 86 04 02 02 00 00 01 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 90 00"

In which,
Response returned by the contactless chip = "D5 41"
Response returned by the FeliCa Tag =

"00 1D 07 01 01 05 01 86 04 02 02 00 00 01 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA"
Status Code returned by the reader = "90 00"

Example 4: To initialize an ISO 14443-4 Type B Tag (Tag Polling)

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 05 D4 4A 01 03 00"

In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the Tag Command = "05"
Tag Command (InListPassiveTarget Type B 106Kbps) = "D4 4A 01 03 00"

To send an APDU to the slot 0 (default), sequence number = 4.

```
HOST -> 02 6F 0A 00 00 00 00 04 00 00 00
FF 00 00 00 05 D4 4A 01 03 00
[Checksum] 03

RDR -> 02 00 00 03
RDR  -> 02 81 14 00 00 00 00 04 00 00 00
D5 41 01 01 50 00 01 32 F4 00 00 00 00 33 81 81 01 21
90 00 [Checksum] 03
```

The APDU Response is
"D5 4B 01 01 50 00 01 32 F4 00 00 00 00 33 81 81 01 21 90 00"

In which,
Response returned by the contactless chip = "D5 4B 01 01"
ATQB of the Type B Tag = "50 00 01 32 F4 00 00 00 00 33 81 81"
CRC-B = "01 21"
Status Code returned by the reader = "90 00"

Example 5: To send an APDU to an ISO 14443-4 Type B Tag (Data Exchange)

Step 1: Issue a "Direct Transmit" APDU.

The USER APDU Command should be "00 84 00 00 08"
The Composed APDU Command should be "FF 00 00 00 08 D4 40 01 00 84 00 00 08"

In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the Tag Command = "08"
Tag Command (InDataExchange) = "D4 40 01"
Tag Command (Get Challenge) = "00 84 00 00 08"

To send an APDU to the slot 0 (default), sequence number = 5.

```
HOST -> 02 6F 0D 00 00 00 00 05 00 00 00
FF 00 00 00 08 D4 40 01 00 84 00 00 08
[Checksum] 03
```

```
RDR -> 02 00 00 03

RDR  -> 02 81 0F 00 00 00 00 05 00 00 00
D5 41 00 01 02 03 04 05 06 07 08 90 00 90 00
[Checksum] 03
```

The APDU Response is "D5 41 00 0B 01 02 03 04 05 06 07 08 90 00"

In which,
Response returned by the contactless chip = "D5 41 00"
Response from the Type B Tag = "01 02 03 04 05 06 07 08 90 00"
Status Code returned by the reader = "90 00"

5.3. Get the Firmware Version of the Reader

To derive the firmware version of the reader.

Command	Class	INS	P1	P2	Le
Get Response	0xFF	0x00	0x48	0x00	0x00

Table 23: Get Firmware Version Command Format (5 Bytes)

Le: Number of Bytes to Retrieve (1 Byte)

Maximum 255 bytes

Response	Data Out
Result	Firmware Version

Table 24: Get Firmware Version Response Format (10 bytes)

E.g. Response = 41 43 52 31 32 32 53 31 30 30 (Hex) = ACR122S100 (ASCII)

5.4. Pseudo-APDU for Bi-Color LED and Buzzer Control

This APDU is used to control the states of the Bi-Color LED and Buzzer.

Command	Class	INS	P1	P2	Lc	Data In (4 Bytes)
Bi-Color LED and Buzzer Control	0xFF	0x00	0x40	LED State Control	0x04	Blinking Duration Control

Table 25: Bi-Color LED and Buzzer Control Command Format (9 Bytes)

P2: LED State Control

CMD	Item	Description
Bit 0	Final Red LED State	1 = On; 0 = Off
Bit 1	Final Green LED State	1 = On; 0 = Off
Bit 2	Red LED State Mask	1 = Update the State 0 = No change
Bit 3	Green LED State Mask	1 = Update the State 0 = No change
Bit 4	Initial Red LED Blinking State	1 = On; 0 = Off
Bit 5	Initial Green LED Blinking State	1 = On; 0 = Off

Bit 6	Red LED Blinking Mask	1 = Blink 0 = Not Blink
Bit 7	Green LED Blinking Mask	1 = Blink 0 = Not Blink

Table 26: Bi-Color LED and Buzzer Control Format (1 Byte)

Data In: Blinking Duration Control

Byte 0	Byte 1	Byte 2	Byte 3
T1 Duration Initial Blinking State (Unit = 100ms)	T2 Duration Toggle Blinking State (Unit = 100ms)	Number of repetition	Link to Buzzer

Table 27: Bi-Color LED Blinking Duration Control Format (4 Bytes)

Byte 3: Link to Buzzer. Control the buzzer state during the LED Blinking

0x00: The buzzer will not turn on

0x01: The buzzer will turn on during the T1 Duration

0x02: The buzzer will turn on during the T2 Duration

0x03: The buzzer will turn on during the T1 and T2 Duration

Data Out: SW1 SW2. Status Code returned by the reader.

Results	SW1	SW2	Meaning
Success	90	Current LED State	The operation is completed successfully.
Error	63	00	The operation is failed.

Table 28: Status Code

Status	Item	Description
Bit 0	Current Red LED	1 = On; 0 = Off
Bit 1	Current Green LED	1 = On; 0 = Off
Bits 2 – 7	Reserved	

Table 29: Current LED State (1 Byte)

Note:

1. The LED State operation will be performed after the LED Blinking operation is completed.
2. The LED will not be changed if the corresponding LED Mask is not enabled.
3. The LED will not be blinking if the corresponding LED Blinking Mask is not enabled. Also, the number of repetition must be greater than zero.
4. T1 and T2 duration parameters are used for controlling the duty cycle of LED blinking and Buzzer Turn-On duration. For example, if T1=1 and T2=1, the duty cycle = 50%. Duty Cycle = T1 / (T1 + T2).
5. To control the buzzer only, just set the P2 "LED State Control" to zero.
6. The make the buzzer operating, the "number of repetition" must be greater than zero.
7. To control the LED only, just set the parameter "Link to Buzzer" to zero.

Example 1: To read the existing LED State

// Assume both Red and Green LEDs are OFF initially //

// Not link to the buzzer //

APDU = "FF 00 40 00 04 00 00 00 00"

Response = "90 00". RED and Green LEDs are OFF.

Example 2: To turn on RED and Green Color LEDs

// Assume both Red and Green LEDs are OFF initially //

// Not link to the buzzer //

APDU = "FF 00 40 0F 04 00 00 00 00"

Response = "90 03". RED and Green LEDs are ON,

To turn off both RED and Green LEDs, APDU = "FF 00 40 0C 04 00 00 00 00"

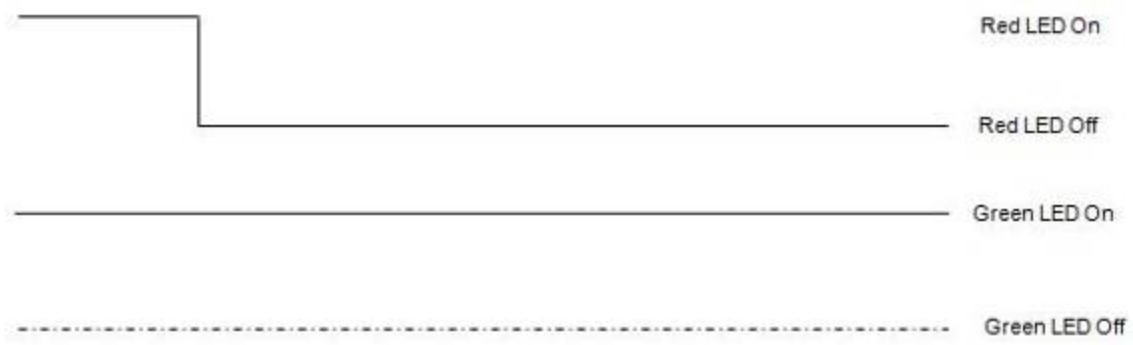
Example 3: To turn off the RED Color LED only, and leave the Green Color LED unchanged.

// Assume both Red and Green LEDs are ON initially //

// Not link to the buzzer //

APDU = "FF 00 40 04 04 00 00 00 00"

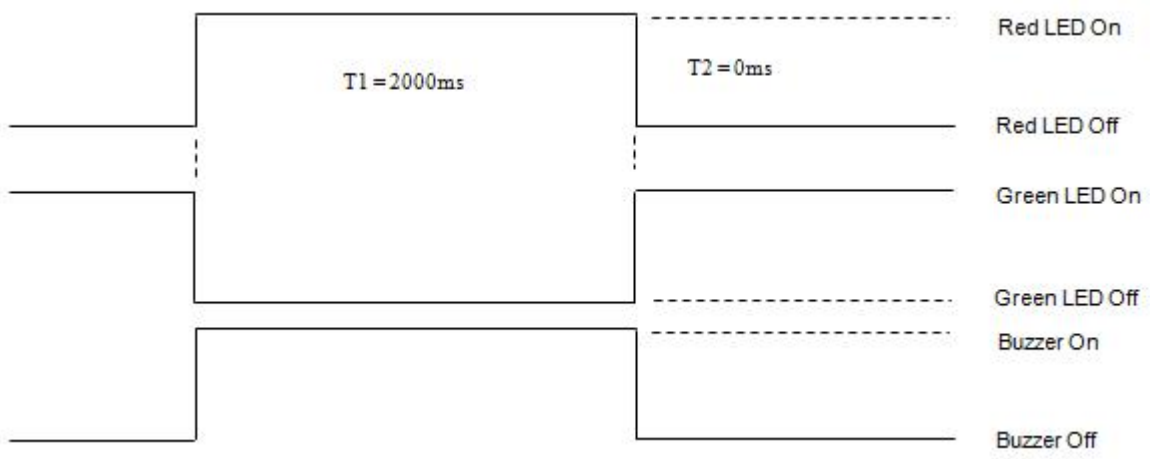
Response = "90 02". Green LED is not changed (ON); Red LED is OFF,



Example 4: To turn on the Red LED for 2 sec. After that, resume to the initial state

// Assume the Red LED is initially OFF, while the Green LED is initially ON. //

// The Red LED and buzzer will turn on during the T1 duration, while the Green LED will turn off during the T1 duration. //



1Hz = 1000ms Time Interval = 500ms ON + 500 ms OFF

T1 Duration = 2000ms = 0x14

T2 Duration = 0ms = 0x00

Number of repetition = 0x01

Link to Buzzer = 0x01

APDU = "FF 00 40 50 04 14 00 01 01"

Response = "90 02"

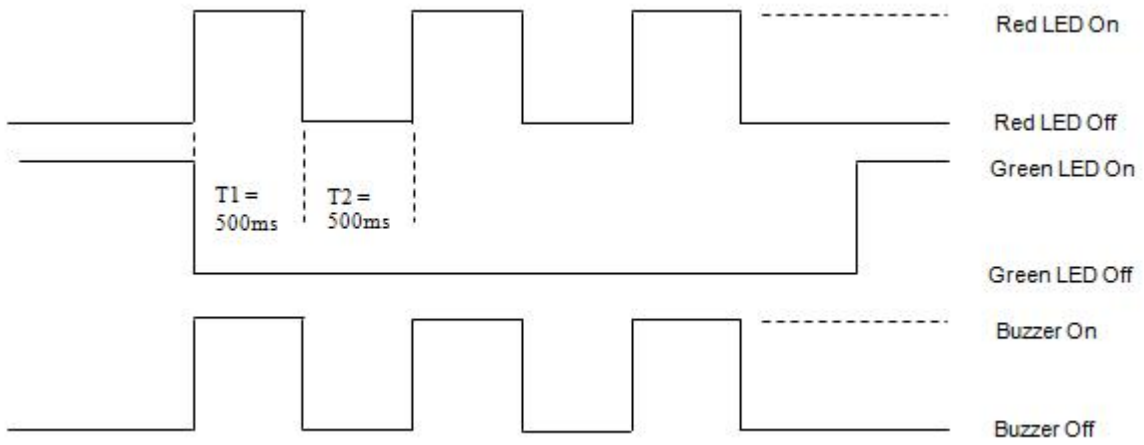
Example 5: To blink the Red LED of 1Hz for 3 times. After that, resume to initial state

// Assume the Red LED is initially OFF, while the Green LED is initially ON. //

// The Initial Red LED Blinking State is ON. Only the Red LED will be blinking.

// The buzzer will turn on during the T1 duration, while the Green LED will turn off during both the T1 and T2 duration.

// After the blinking, the Green LED will turn ON. The Red LED will resume to the initial state after the blinking //



1Hz = 1000ms Time Interval = 500ms ON + 500 ms OFF

T1 Duration = 500ms = 0x05

T2 Duration = 500ms = 0x05

Number of repetition = 0x03

Link to Buzzer = 0x01

APDU = "FF 00 40 50 04 05 05 03 01"

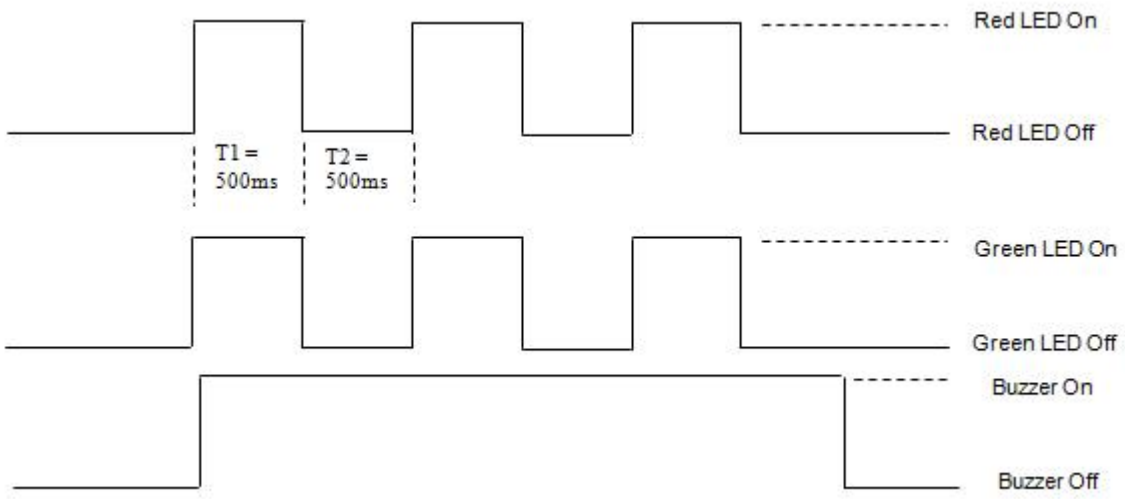
Response = "90 02"

Example 6: To blink the Red and Green LEDs of 1Hz for 3 times

// Assume both the Red and Green LEDs are initially OFF. //

// Both Initial Red and Green Blinking States are ON //

// The buzzer will turn on during both the T1 and T2 duration//



1Hz = 1000ms Time Interval = 500ms ON + 500ms OFF

T1 Duration = 500ms = 0x05

T2 Duration = 500ms = 0x05

Number of repetition = 0x03

Link to Buzzer = 0x03

APDU = “FF 00 40 F0 04 05 05 03 03”

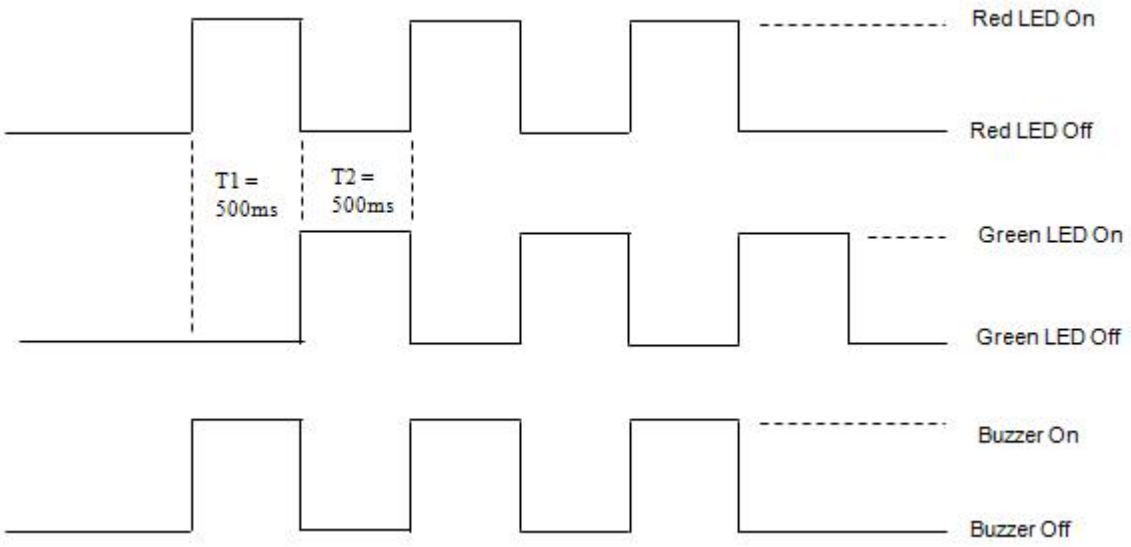
Response = “90 00”

Example 7: To blink the Red and Green LED in turn of 1Hz for 3 times

// Assume both Red and Green LEDs are initially OFF. //

// The Initial Red Blinking State is ON; The Initial Green Blinking States is OFF //

// The buzzer will turn on during the T1 duration//



1Hz = 1000ms Time Interval = 500ms ON + 500ms OFF

T1 Duration = 500ms = 0x05

T2 Duration = 500ms = 0x05

Number of repetition = 0x03

Link to Buzzer = 0x01

APDU = "FF 00 40 D0 04 05 05 03 01"

Response = "90 00"

5.5. Pseudo-APDU for Topaz512 and Jewel96

Note: This section only applies to ACR122S with firmware version 1.03.

This APDU is used to Write-with-erase (8 Bytes), Write-no-erase (8 Bytes), Read (8 Bytes) and Read Segment.

Command	Class	INS	P1	P2	Lc	Data In
Direct Transmit	0xFF	0x00	0x00	0x00	Number of Bytes to send	TAG Command

Table 30: Topaz512 and Jewel96 Command Format

Lc: Number of Bytes to Send (1 Byte)

Maximum 255 bytes

Data In: TAG Command

The data to be sent to the Tag.

Response	Data Out	
Result	TAG Response	SW1 SW2

Table 31: Direct Transmit Response Format (Response Length + 2 Bytes)

Data Out: TAG Response

TAG Response returned by the reader.

Data Out: SW1 SW2

Status Code returned by the reader.

Results	SW1	SW2	Meaning
Success	90	00	The operation is completed successfully.
Error	63	00	The operation is failed.

Table 32: Status Code

Example 1: To Write-with-erase (8 Bytes) a Topaz512/Jewel96 Tag

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 0D D4 40 01 54 05 01 23 45 67 89 AB CD EF"

#In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the Tag Command = "0D"
Tag Command (InDataExchange) = "D4 40 01"
Tag Command (Write-with-erase 8Bytes) = "54 "
Tag Address (00~3F (hex)) = "05"
Tag Data = "01 23 45 67 89 AB CD EF "

To send an APDU to the slot 0 (default), sequence number = 1.

```
HOST -> 02 6F 12 00 00 00 00 01 00 00 00
          FF 00 00 00 0D D4 40 01 54 05 01 23 45 67 89 AB CD EF
          [Checksum] 03

RDR -> 02 00 00 03

RDR  -> 02 80 0D 00 00 00 00 01 01 00 00
          D5 09 05 01 23 45 67 89 AB CD EF 90 00
          [Checksum] 03
```

The APDU Response is "D5 09 05 01 23 45 67 89 AB CD EF 90 00"

#In which,
Response returned by the contactless chip = "D5 09 05 01 23 45 67 89 AB CD EF 90 00"
Write Tag Address = "05 "
Write Tag 8Bytes Data = "01 23 45 67 89 AB CD EF "
Status Code returned by the reader = "90 00"

Example 2: To Write-no-erase (8 Bytes) a Topaz512/Jewel96 Tag

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 0D D4 40 01 1B 05 FF FF FF FF FF FF FF FF"

#In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the Tag Command = "0D"
Tag Command (InDataExchange) = "D4 40 01"
Tag Command (Write-no-erase 8Bytes) = "1B "
Tag Address (00~3F (hex)) = "05 "
Tag Data = "FF FF FF FF FF FF FF FF "

To send an APDU to the slot 0 (default), sequence number = 1.

```
HOST -> 02 6F 12 00 00 00 00 01 00 00 00
          FF 00 00 00 0D D4 40 01 1B 05 FF FF FF FF FF FF FF FF
          [Checksum] 03

RDR -> 02 00 00 03

RDR  -> 02 80 0D 00 00 00 00 01 01 00 00
          D5 09 05 FF FF FF FF FF FF FF FF 90 00
          [Checksum] 03
```

The APDU Response is "D5 09 05 FF FF FF FF FF FF FF FF 90 00"

#In which,
Response returned by the contactless chip = "D5 09 05 FF FF FF FF FF FF FF FF 90 00"
Write Tag Address = "05 "
Write Tag 8Bytes Data = "FF FF FF FF FF FF FF FF "
Status Code returned by the reader = "90 00"

Example 3: To Read 8 Bytes a Topaz512/Jewel96 Tag

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 0D D4 40 01 02 05 00 00 00 00 00 00 00 00"

#In which,
Direct Transmit APDU = "FF 00 00 00"

Length of the Tag Command = "0D"
Tag Command (InDataExchange) = "D4 40 01"
Tag Command (Read 8Bytes) = "02 "
Tag Address (00~3F (hex)) = "05 "
Tag Data = "00 00 00 00 00 00 00 00"

To send an APDU to the slot 0 (default), sequence number = 1.

```
HOST -> 02 6F 12 00 00 00 00 01 00 00 00
          FF 00 00 00 0D D4 40 01 02 05 00 00 00 00 00 00 00 00
          [Checksum] 03

RDR -> 02 00 00 03

RDR  -> 02 80 0D 00 00 00 00 01 01 00 00
          D5 09 05 01 23 45 67 89 AB CD EF 90 00
          [Checksum] 03
```

The APDU Response is "D5 09 05 01 23 45 67 89 AB CD EF 90 00"

#In which,
Response returned by the contactless chip = "D5 09 05 01 23 45 67 89 AB CD EF 90 00"
Read Tag Address = "05 "
Read Tag 8Bytes Data = "01 23 45 67 89 AB CD EF "
Status Code returned by the reader = "90 00"

Example 4: To Read Segment a Topaz512/Jewel96 Tag

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 0D D4 40 01 10 00 00 00 00 00 00 00 00 00"

#In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the Tag Command = "0D"
Tag Command (InDataExchange) = "D4 40 01"
Tag Command (Read Segment) = "10 "
Tag Address (00/10/20/30) = "00 "(Block 0)
Tag Data = "00 00 00 00 00 00 00 00"

To send an APDU to the slot 0 (default), sequence number = 1.

```
HOST ->  02 6F 12 00 00 00 00 01 00 00 00
          FF 00 00 00 0D D4 40 01 10 00 00 00 00 00 00 00 00 00
          [Checksum] 03

RDR -> 02 00 00 03

RDR -> 02 80 0D 00 00 00 00 01 01 00 00
          D5 41 00 ... <128 bytes data> ... 90 00
          [Checksum] 03
```

The APDU Response is "D5 41 00 … <128 bytes data> … 90 00"

#In which,
Response returned by the contactless chip = "D5 41 00 … <128 bytes data> … 90 00"
Read Tag Segment Data = "<128 bytes data> "
Status Code returned by the reader = "90 00"

Example 5: To Write Multi-Data at Topaz/Jewel Tag

Note that this function only can write at the segment 0.

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 36 D4 40 01 58 20 30 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47"

#In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the Tag Command = "36"
Tag Command (InDataExchange) = "D4 40 01"
Tag Command (Write Multi-Data) = "58 "

Tag Address = "20 (0 0100 000) "(Block 4, Byte-0) (refer to below Fig. 2)

*Tag Data = "*00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47*"*

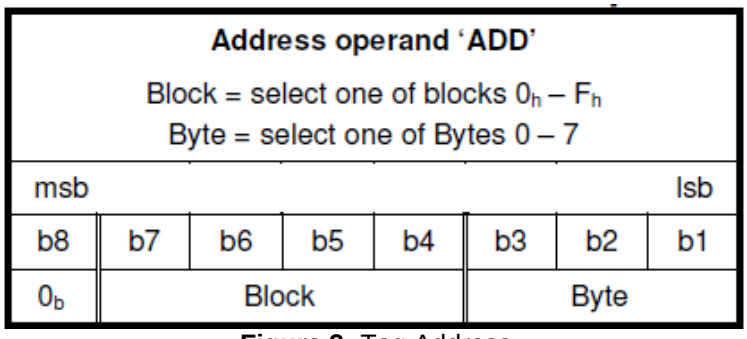


Figure 2: Tag Address

To send an APDU to the slot 0 (default), sequence number = 1.

```
HOST -> 02 6F 3B 00 00 00 00 01 00 00 00
          FF 00 00 00 36 D4 40 01 58 20 30 00 01 02 03 04 05 06 07 08 09 10
          11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
          33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
          [Checksum] 03

RDR  -> 02 00 00 03
```

```
RDR  -> 02 80 05 00 00 00 00 01 01 00 00
        D5 41 00 90 00
        [Checksum] 03
```

The APDU Response is "D5 41 00 90 00"

#In which,
Response returned by the contactless chip = "D5 41 00 90 00"
Write Tag Data = "00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 "
Status Code returned by the reader = "90 00"
If Status Code returned by the reader = "63 00" that means this operation is not complete.

Example 6: To Write Multi-8 bytes Data at Topaz512/Jewel96 Tag

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 36 D4 40 01 5A 04 30 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47"

#In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the Tag Command = "36"
Tag Command (InDataExchange) = "D4 40 01"

Tag Command (Write Multi-Data) = "5A"
Tag Address (Block No.00-3F) = "04 "(Block No. 4) (refer to below Fig. 3)

Tag Data = "00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47"

Address operand 'ADD8'

Block = select one of the 8-Byte blocks 00_h – $3F_h$

msb							lsb
b8	b7	b6	b5	b4	b3	b2	b1
0_b	0_b	Global Block					

Figure 3: Tag Address

To send an APDU to the slot 0 (default), sequence number = 1.

```
HOST -> 02 6F 3B 00 00 00 00 01 00 00 00
FF 00 00 00 36 D4 40 01 5A 04 30 00 01 02 03 04 05 06 07 08 09 10 11 12 13
14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
39 40 41 42 43 44 45 46 47
[Checksum] 03
```

```
RDR  -> 02 00 00 03

RDR  -> 02 80 04 00 00 00 00 01 01 00 00
D5 09 90 00
[Checksum] 03
```

The APDU Response is "D5 09 90 00"

#In which,
Response returned by the contactless chip = "D5 09 90 00"
Write Tag Data = "00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 "
Status Code returned by the reader = "90 00"
If Status Code returned by the reader = "63 00" that mean this operation is not complete.

5.6. Basic Program Flow for ISO 14443-4 Type A and B Tags

Typical sequence may be:

- Scanning the tags in the field (Polling) with the correct parameter (Type A or B)
- Change the Baud Rate (optional for Type A tags only)
- Perform any T=CL command
- Deselect the tag

Step 1) **Polling** for the ISO14443-4 Type A Tag, 106 kbps

```
HOST -> 02 6F 09 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST -> FF 00 00 00 04 D4 4A 01 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 4B 01 01 00 08 28 04 85 82 2F A0 07 77 F7 80 02 47 65 90 00
[Checksum] 03
```

In which,

```
Number of Tag found = [01];     Target number = 01
SENS_RES = 00 08;               SEL_RES = 28,
Length of the UID = 4;          UID = 85 82 2F A0
ATS = 07 77 F7 80 02 47 65
Operation Finished = 90 00
```

Or

Step 1) **Polling** for the ISO14443-4 Type B Tag, 106 kbps

```
HOST -> 02 6F 0A 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST -> FF 00 00 00 05 D4 4A 01 03 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 14 00 00 00 00 01 01 00 00
RDR -> D5 4B 01 01 50 00 01 32 F4 00 00 00 00 33 81 81 01 21 90 00
[Checksum] 03
```

In which,

```
Number of Tag found = [01];   Target number = 01
ATQB = 50 00 01 32 F4 00 00 00 00 33 81 81
ATTRIB_RES Length = 01;       ATTRIB_RES = 21
Operation Finished = 90 00
```

Step 2) **Change the default Baud Rate to other Baud Rate (optional)**

```
HOST -> 02 6F 0A 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST -> FF 00 00 00 05 D4 4E 01 02 02 [Checksum] 03 // Change to Baud Rate
424 kbps
Or
HOST -> FF 00 00 00 05 D4 4E 01 01 01 [Checksum] 03 // Change to Baud Rate
212 kbps
```

```
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 4F [00] 90 00 [Checksum] 03
```
Please check the maximum baud rate supported by the tags. Only Type A tags are supported.

Step 3) **Perform T=CL command, Get Challenge APDU = 00 84 00 00 08**
```
HOST -> 02 6F 0D 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST -> FF 00 00 00 08 D4 40 01 00 84 00 00 08 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 0F 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 62 89 99 ED C0 57 69 2B 90 00 90 00 [Checksum] 03
In which, Response Data = 62 89 99 ED C0 57 69 2B 90 00
```

Step 4) **Deselect the Tag**
```
HOST -> 02 6F 08 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST -> FF 00 00 00 03 D4 44 01 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03
```

Step 5) **Turn off the Antenna Power (optional)**
```
HOST -> 02 6F 09 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST -> FF 00 00 00 04 D4 32 01 00
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 04 00 00 00 00 01 01 00 00
RDR -> D5 33 90 00 [Checksum] 03
```

Please refer to the Tag specification for more detailed information.

5.7. Basic Program Flow for MIFARE Applications

Typical sequence may be:

- Scanning the tags in the field (Polling)
- Authentication
- Read / Write the memory of the tag
- Halt the tag (optional)

Step 1) **Polling** for the MIFARE 1K/4K Tags, 106 kbps
```
HOST -> 02 6F 09 00 00 00 00 01 00 00 00 FF 00 00 00 04 D4 4A 01 00
[Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 0E 00 00 00 00 01 01 00 00
RDR -> D5 4B 01 01 00 02 18 04 F6 8E 2A 99 90 00 [Checksum] 03
```

In which,
```
Number of Tag found = [01];          Target number = 01
SENS_RES = 00 02;                    SEL_RES = 18,
Length of the UID = 4;               UID = F6 8E 2A 99
Operation Finished = 90 00
```

Tip: The tag type can be determined by recognizing the SEL_RES.
SEL_RES of some common tag types.

00 = MIFARE Ultralight
08 = MIFARE 1K
09 = MIFARE MINI
18 = MIFARE 4K
20 = MIFARE DESFIRE
28 = JCOP30

98 = Gemplus MPCOS

Step 2) **KEY A Authentication,** Block **04**, KEY = FF FF FF FF FF FF, UID = F6 8E 2A 99

```
HOST -> 02 6F 14 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 0F D4 40 01 60 04 FF FF FF FF FF FF F6 8E 2A 99
[Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03
```

Tip: If the authentication failed, the error code [XX] will be returned.
[00] = Valid, other = Error. Please refer to Error Codes Table for more details.

*Tip: For **KEY B Authentication***

```
HOST -> 02 6F 14 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 0F D4 40 01 61 04 FF FF FF FF FF FF F6 8E 2A 99
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03
```

Step 3) **Read** the content of Block **04**

```
HOST -> 02 6F 0A 00 00 00 00 01 00 00 00 FF 00 00 00 05 D4 40 01 30 04
[Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 90 00
[Checksum] 03
In which, Block Data = 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
```

Step 4) **Update** the content of Block **04**

```
HOST -> 02 6F 14 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 15 D4 40 01 A0 04 01 02 03 04 05 06 07 08 09 0A 0B 0C
0D 0E 0F 10  [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03
```

Step 5) **Halt the tag** (optional)

```
HOST -> 02 6F 08 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 03 D4 44 01 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 45 [00] 90 00 [Checksum] 03
```

5.7.1. How to Handle Value Blocks of MIFARE 1K/4K Tag?

The value blocks are used for performing electronic purse functions, e.g. Increment, Decrement, Restore, Transfer, etc. The value blocks have a fixed data format which permits error detection and correction and a backup management.

Byte Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	Value				$\overline{\text{Value}}$				Value				Adr	$\overline{\text{Adr}}$	Adr	$\overline{\text{Adr}}$

Value: A signed 4-Byte value. The lowest significant byte off a value is stored in the lowest address byte. Negative values are stored in standard 2's complement format.

Adr: 1-Byte address, which can be used to save the storage address of a block. (optional)

E.g. `Value 100 (decimal) = 64 (Hex)`, assume `Block = 0x05`
The formatted value block = `64 00 00 00 9B FF FF FF 64 00 00 00 05 FA 05 FA`

Step 1) **Update** the content of Block **05 with a value 100 (dec)**
```
HOST -> 02 6F 1A 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 15 D4 40 01 A0 05 64 00 00 00 9B FF FF FF 64 00 00 00
05 FA 05 FA [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03
```

Step 2) **Increment** the value of Block **05 by 1 (dec)**
```
HOST -> 02 6F 0E 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 09 D4 40 01 C1 05 01 00 00 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03
```

*Tip: **Decrement** the value of Block **05 by 1 (dec)***
```
HOST -> FF 00 00 00 09 D4 40 01 C0 05 01 00 00 00
```

Step 3) **Transfer** the prior calculated value of Block **05 (dec)**
```
HOST -> 02 6F 0A 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 B0 05 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03
```

*Tip: **Restore** the value of Block **05** (cancel the prior Increment or Decrement operation)*
```
HOST -> FF 00 00 00 05 D4 40 01 C2 05
```

Step 4) **Read** the content of Block **05**
```
HOST -> 02 6F 0A 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 30 05 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 65 00 00 00 9A FF FF FF 65 00 00 00 05 FA 05 FA 90 00
[Checksum] 03
```

In which, the value = `101` (dec)

Step 5) **Copy** the value of Block **05** to Block **06 (dec)**
```
HOST -> 02 6F 0A 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 C2 05 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 B0 06 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03
```

Step 6) **Read** the content of Block **06**
```
HOST -> 02 6F 0A 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 30 06 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
```

```
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 65 00 00 00 9A FF FF FF 65 00 00 00 05 FA 05 FA 90 00
[Checksum] 03
```

In which, the value = 101 (dec). The Adr "05 FA 05 FA" tells us the value is copied from Block 05.

Please refer to the MIFARE specification for more detailed information.

MIFARE 1K Memory Map

Sectors (Total 16 sectors. Each sector consists of 4 consecutive blocks)	**Data Blocks** (3 blocks, 16 bytes per block)	**Trailer Block** (1 block, 16 bytes)
Sector 0	0x00 ~ 0x02	0x03
Sector 1	0x04 ~ 0x06	0x07
..		
..		
Sector 14	0x38 ~ 0x0A	0x3B
Sector 15	0x3C ~ 0x3E	0x3F

1K Bytes

MIFARE 4K Memory Map

Sectors (Total 32 sectors. Each sector consists of 4 consecutive blocks)	**Data Blocks** (3 blocks, 16 bytes per block)	**Trailer Block** (1 block, 16 bytes)
Sector 0	0x00 ~ 0x02	0x03
Sector 1	0x04 ~ 0x06	0x07
..		
..		
Sector 30	0x78 ~ 0x7A	0x7B
Sector 31	0x7C ~ 0x7E	0x7F

2K Bytes

Sectors (Total 8 sectors. Each sector consists of 16 consecutive blocks)	**Data Blocks** (15 blocks, 16 bytes per block)	**Trailer Block** (1 block, 16 bytes)
Sector 32	0x80 ~ 0x8E	0x8F
Sector 33	0x90 ~ 0x9E	0x9F
..		
..		
Sector 38	0xE0 ~ 0xEE	0xEF
Sector 39	0xF0 ~ 0xFE	0xFF

2K Bytes

Tip: Once the authentication is done, all the data blocks of the same sector are free to access. For example, once the data block 0x04 is successfully authenticated (Sector 1), the data blocks 0x04 ~ 0x07 are free to access.

5.7.2. How to Access MIFARE Ultralight Tags?

Typical sequence may be:

- Scanning the tags in the field (Polling)
- Read / Write the memory of the tag
- Halt the tag (optional)

Step 1) **Polling** for the MIFARE Ultralight Tags, 106 kbps

```
HOST -> 02 6F 09 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 04 D4 4A 01 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 11 00 00 00 00 01 01 00 00
RDR -> D5 4B 01 01 00 44 00 07 04 6E 0C A1 BF 02 84 90 00 [Checksum] 03
```

In which,

```
Number of Tag found = [01];        Target number = 01
SENS_RES = 00 44;                  SEL_RES = 00,
Length of the UID = 7;             UID = 04 6E 0C A1 BF 02 84
Operation Finished = 90 00
```

Step 2) **Read** the content of Page **04**

```
HOST -> 02 6F 09 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 30 04 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 90 00
[Checksum] 03
```

In which, Block Data = 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

Tip: 4 consecutive Pages will be retrieved. Pages 4, 5, 6 and 7 will be retrieved. Each data page consists of 4 bytes.

Step 3) **Update** the content of Page **04 with** the data "AA BB CC DD"

```
HOST -> 02 6F 0E 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 09 D4 40 01 A2 04 AA BB CC DD [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03
```

Or

Step 3) **Write (MIFARE compatible Write)** the content of Page **04 with** the data "AA BB CC DD"

```
HOST -> 02 6F 0E 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 15 D4 40 01 A0 04 AA BB CC DD 00 00 00 00 00 00 00 00
00 00 00 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03
```

Tip: This command is implemented to accommodate the established MIFARE 1K/4K infrastructure. We have to assemble the data into a 16 bytes frame. The first 4 bytes are for data, the rest of the bytes (12 ZEROS) are for padding. Only the page 4 (4 bytes) is updated even through 16 byte are sent to the reader.

Step 4) **Read** the content of Page **04 again**

```
HOST -> 02 6F 0E 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 30 04 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] AA BB CC DD 05 06 07 08 09 10 11 12 13 14 15 16 90 00
```

```
[Checksum] 03
In which, Block Data = AA BB CC DD 05 06 07 08 09 10 11 12 13 14 15 16
```

Tip: Only the page 4 is updated. Pages 5, 6 and 7 remain the same.

Step 5) **Halt the tag** (optional)

```
HOST -> 02 6F 0E 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 03 D4 44 01 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 45 [00] 90 00 [Checksum] 03
```

Please refer to the MIFARE Ultralight specification for more detailed information.

MIFARE Ultralight Memory Map

Byte Number	0	1	2	3	Page
Serial Number	SN0	SN1	SN2	BCC0	0
Serial Number	SN3	SN4	SN5	SN6	1
Internal / Lock	BCC1	Internal	Lock0	Lock1	2
OTP	OPT0	OPT1	OTP2	OTP3	3
Data read/write	Data0	Data1	Data2	Data3	4
Data read/write	Data4	Data5	Data6	Data7	5
Data read/write	Data8	Data9	Data10	Data11	6
Data read/write	Data12	Data13	Data14	Data15	7
Data read/write	Data16	Data17	Data18	Data19	8
Data read/write	Data20	Data21	Data22	Data23	9
Data read/write	Data24	Data25	Data26	Data27	10
Data read/write	Data28	Data29	Data30	Data31	11
Data read/write	Data32	Data33	Data34	Data35	12
Data read/write	Data36	Data37	Data38	Data39	13
Data read/write	Data40	Data41	Data42	Data43	14
Data read/write	Data44	Data45	Data46	Data47	15

512 bits
Or
64 Bytes

5.7.3. How to Access MIFARE Ultralight C Tags?

Typical sequence may be:

- Scanning the tags in the field (Polling)
- Authentication
- Read / Write the memory of the tag
- Halt the tag (optional)

Step 1) **Polling** for the MIFARE Ultralight C Tags, 106 kbps

```
HOST -> 02 6F 09 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 04 D4 4A 01 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 11 00 00 00 00 01 01 00 00
RDR -> D5 4B 01 01 00 44 00 07 04 6E 0C A1 BF 02 84 90 00 [Checksum] 03
```

In which,

```
Number of Tag found = [01];
SENS_RES = 00 44;
Length of the UID = 7;
Operation Finished = 90 00
```

```
Target number = 01
SEL_RES = 00,
UID = 04 6E 0C A1 BF 02 84
```

Step 2) **3DES Authentication**

```
HOST -> 02 6F 09 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 04 D4 42 1A 00 10 03
RDR -> 02 00 00 03 (Waiting the Tag)
```

```
RDR -> 02 80 0E 00 00 00 00 01 01 00 00
RDR -> D5 43 [00] 04 77 64 89 99 74 24 67 90 00 [Checksum] 03
```

In which, 3DES challenge from the card = [04 77 64 89 99 74 24 67];
h = 90 00

```
HOST -> 02 6F 18 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 13 D4 42 AF 88 68 45 07 65 86 99 67 00 53 77 56 98 65 49 67 [Checksum] 03
```

In which, 3DES reply to the card = [88 68 45 07 65 86 99 67 00 53 77 56 98 65 49 67];

```
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 0E 00 00 00 00 01 01 00 00
RDR -> D5 43 [00] 00 06 78 53 80 68 89 61 24 90 00 [Checksum] 03
```

In which, 3DES reply from the card = [06 78 53 80 68 89 61 24];
Operation Finished = 90 00

Tip: The 3DES reply from the card should be checked to make sure the card is legitimate.

Step 3) **Read** the content of Page **04**

```
HOST -> 02 6F 09 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 30 04 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 90 00 [Checksum] 03
```

In which, Block Data = 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

Tip: 4 consecutive Pages will be retrieved. Pages 4, 5, 6 and 7 will be retrieved. Each data page consists of 4 bytes.

Step 4) **Update** the content of Page **04 with** the data "AA BB CC DD"

```
HOST -> 02 6F 0E 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 09 D4 40 01 A2 04 AA BB CC DD [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03
```

Or

Step 4) **Write (MIFARE compatible Write)** the content of Page **04 with** the data "AA BB CC DD"

```
HOST -> 02 6F 0E 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 15 D4 40 01 A0 04 AA BB CC DD 00 00 00 00 00 00 00 00 00 00 00 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03
```

Tip: This command is implemented to accommodate the established MIFARE 1K/4K infrastructure. We have to assemble the data into a 16 bytes frame. The first 4 bytes are for data, the rest of the bytes (12 ZEROS) are for padding. Only the page 4 (4 bytes) is updated even through 16 byte are sent to the reader.

Step 5) **Read** the content of Page **04 again**

```
HOST -> 02 6F 0E 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 30 04 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
```

```
RDR -> D5 41 [00] AA BB CC DD 05 06 07 08 09 10 11 12 13 14 15 16 90 00
[Checksum] 03
```

In which, `Block Data` = `AA BB CC DD 05 06 07 08 09 10 11 12 13 14 15 16`

Tip: Only the page 4 is updated. Pages 5, 6 and 7 remain the same.

Step 6) **Halt the tag** (optional)

```
HOST -> 02 6F 0E 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 03 D4 44 01 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 45 [00] 90 00 [Checksum] 03
```

Please refer to the MIFARE Ultralight C specification for more detailed information.

MIFARE Ultralight C Memory Map

Byte Number	0	1	2	3	Page
Serial Number	SN0	SN1	SN2	BCC0	0
Serial Number	SN3	SN4	SN5	SN6	1
Internal / Lock	BCC1	Internal	Lock	Lock	2
OTP	OTP0	OTP1	OTP2	OTP3	3
Data read/write	Data0	Data1	Data2	Data3	4
Data read/write	Data4	Data5	Data6	Data7	5
Data read/write	Data8	Data9	Data10	Data11	6
Data read/write	Data12	Data13	Data14	Data15	7
Data read/write	Data16	Data17	Data18	Data19	8
Data read/write	Data20	Data21	Data22	Data23	9
Data read/write	Data24	Data25	Data26	Data27	10
Data read/write	Data28	Data29	Data30	Data31	11
Data read/write	Data32	Data33	Data34	Data35	12
Data read/write	Data36	Data37	Data38	Data39	13
Data read/write	Data40	Data41	Data42	Data43	14
Data read/write	Data44	Data45	Data46	Data47	15
Data read/write	Data48	Data49	Data50	Data51	16
Data read/write	Data52	Data53	Data54	Data55	17
Data read/write	Data56	Data57	Data58	Data59	18
Data read/write	Data60	Data61	Data62	Data63	19
Data read/write	Data64	Data65	Data66	Data67	20
Data read/write	Data68	Data69	Data70	Data71	21
Data read/write	Data72	Data73	Data74	Data75	22
Data read/write	Data76	Data77	Data78	Data79	23
Data read/write	Data80	Data81	Data82	Data83	24
Data read/write	Data84	Data85	Data86	Data87	25
Data read/write	Data88	Data89	Data90	Data91	26
Data read/write	Data92	Data93	Data94	Data95	27
Data read/write	Data96	Data97	Data98	Data99	28
Data read/write	Data100	Data101	Data102	Data103	29
Data read/write	Data104	Data105	Data106	Data107	30
Data read/write	Data108	Data109	Data110	Data111	31
Data read/write	Data112	Data113	Data114	Data115	32
Data read/write	Data116	Data117	Data118	Data119	33
Data read/write	Data120	Data121	Data122	Data123	34
Data read/write	Data124	Data125	Data126	Data127	35
Data read/write	Data128	Data129	Data130	Data131	36
Data read/write	Data132	Data133	Data134	Data135	37
Data read/write	Data136	Data137	Data138	Data139	38
Data read/write	Data140	Data141	Data142	Data143	39
Lock	Lock	Lock	-	-	40
16 bit counter	16 bit counter	16 bit counter	-	-	41
Authentication configuration	Authentication configuration	Authentication configuration	Authentication configuration	Authentication configuration	42
Authentication configuration	Authentication configuration	Authentication configuration	Authentication configuration	Authentication configuration	43
Authentication key	Authentication key	Authentication key	Authentication key	Authentication key	44
Authentication key	Authentication key	Authentication key	Authentication key	Authentication key	45
Authentication key	Authentication key	Authentication key	Authentication key	Authentication key	46
Authentication key	Authentication key	Authentication key	Authentication key	Authentication key	47

792 bits Or 198 Bytes

5.8. Basic Program Flow for FeliCa Applications

Step 0. Start the application. The first thing to do is to activate the "SAM Interface". The ATR of the SAM (if a SAM is inserted) or a Pseudo-ATR "3B 00" (if no SAM is inserted) will be returned. In other words, the SAM always exists from the view of the application.

Step 1. The second thing to do is to change the operating parameters of the PN531. Set the Retry Time to one.

Step 2. Poll a FeliCa Tag by sending "Direct Transmit" and "Get Response" APDUs (Tag Polling).

Step 3. If no tag is found, go back to Step 2 until a FeliCa Tag is found.

Step 4. Access the FeliCa Tag by sending APDUs (Tag Read or Write).

Step 5. If there is no any operation with the FeliCa Tag, then go back to Step 2 to poll the other FeliCa Tag.

..

Step N. Deactivate the "SAM Interface". Shut down the application.

Note:

1. The default Retry Time of the Tag command "InListPassiveTarget" is infinity. Send the APDU "FF 00 00 00 06 D4 32 05 00 00 00" to change the Retry Time to one.
2. It is recommended to turn off the Antenna if there is no contactless access. APDU for turning on the Antenna Power = APDU "FF 00 00 00 04 D4 32 01 03"

   APDU for turning off the Antenna Power = APDU "FF 00 00 00 04 D4 32 01 02"

5.9. Basic Program Flow for NFC Forum Type 1 Tag Applications

E.g. Jewel and Topaz Tags

Typical sequence may be:
- Scanning the tags in the field (Polling)
- Read / Update the memory of the tag
- Deselect the tag

Step 1) **Polling** for the Jewel or Topaz Tag, 106 kbps

```
HOST -> 02 6F 09 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST -> FF 00 00 00 04 D4 4A 01 04 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 0C 00 00 00 00 01 01 00 00
RDR -> D5 4B 01 01 0C 00 B5 3E 21 00 90 00 [Checksum] 03
```

In which, `Number of Tag found = [01];` `Target number = 01`
`ATQA_RES = 0C 00;` `UID = B5 3E 21 00`
`Operation Finished = 90 00`

Step 2) Read the memory address 08 (Block 1: Byte-0)

```
HOST -> 02 6F 0A 00 00 00 00 01 00 00 00 FF 00 00 00 05 D4 40 01 01 08
[Checksum] 03
RDR -> 02 00 00 03 02 80 06 00 00 00 00 01 01 00 00 D5 41 [00] 18 90 00
[Checksum] 03
```

In which, Response Data = 18

Tip: To read all the memory content of the tag starting from the memory address 00

```
HOST -> 02 6F 09 00 00 00 00 01 00 00 00 FF 00 00 00 04 D4 40 01 00
[Checksum] 03
RDR -> 02 00 00 03 02 80 7F 00 00 00 00 01 01 00 00 D5 41 00 11 48
```

```
RDR -> show all data … 90 00 [Checksum] 03
```

Step 3) Update the memory address 08(Block 1: Byte-0)with the data FF

```
HOST -> 2 6F 0B 00 00 00 00 01 00 00 00 FF 00 00 00 06 D4 40 01 53 08 FF
[Checksum] 03
RDR -> 02 00 00 03 02 80 05 00 00 00 00 01 01 00 00 D5 41 [00] FF 90 00
[Checksum] 03
```

In which, Response Data = FF

Tip: To update more than one memory content of the tag starting form the memory address 08(Block 1: Byte-0)

```
HOST -> 02 6F 0D 00 00 00 00 01 00 00 00 FF 00 00 00 08 D4 40 01 58 08 02
AA BB [Checksum] 03
RDR -> 02 00 00 03 02 80 06 00 00 00 00 01 01 00 00 D5 41 [00] 90 00
[Checksum] 03
```

In which, Command = 58; Starting memory address = 08;
Number of write content = 02; Memory content = AA, BB;

Step 4) Deselect the Tag

```
HOST -> 02 6F 08 00 00 00 00 01 00 00 00 FF 00 00 00 03 D4 44 01 [Checksum]
03
RDR -> 02 00 00 03 02 80 05 00 00 00 00 01 01 00 00 D5 45 [00] 90 00
[Checksum] 03
```

Appendix A. Topaz

EEPROM Memory Map										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
UID	0	UID-0	UID-1	UID-2	UID-3	UID-4	UID-5	UID-6		Locked
Data	1	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Yes
Data	2	Data8	Data9	Data10	Data11	Data12	Data13	Data14	Data15	Yes
Data	3	Data16	Data17	Data18	Data19	Data20	Data21	Data22	Data23	Yes
Data	4	Data24	Data25	Data26	Data27	Data28	Data29	Data30	Data31	Yes
Data	5	Data32	Data33	Data34	Data35	Data36	Data37	Data38	Data39	Yes
Data	6	Data40	Data41	Data42	Data43	Data44	Data45	Data46	Data47	Yes
Data	7	Data48	Data49	Data50	Data51	Data52	Data53	Data54	Data55	Yes
Data	8	Data56	Data57	Data58	Data59	Data60	Data61	Data62	Data63	Yes
Data	9	Data64	Data65	Data66	Data67	Data68	Data69	Data70	Data71	Yes
Data	A	Data72	Data73	Data74	Data75	Data76	Data77	Data78	Data79	Yes
Data	B	Data80	Data81	Data82	Data83	Data84	Data85	Data86	Data87	Yes
Data	C	Data88	Data89	Data90	Data91	Data92	Data93	Data94	Data95	Yes
Reserved	D									
Lock/Reserved	E	LOCK-0	LOCK-1	OTP-0	OTP-1	OTP-2	OTP-3	OTP-4	OTP-5	

Reserved for internal use
User Block Lock & Status
OTP bits

Appendix B. Topaz512

EEPROM Memory Map (Segment0)										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
UID	00	UID-0	UID-1	UID-2	UID-3	UID-4	UID-5	25_h		Locked
Data	01	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Yes
Data	02	Data8	Data9	Data10	Data11	Data12	Data13	Data14	Data15	Yes
Data	03	Data16	Data17	Data18	Data19	Data20	Data21	Data22	Data23	Yes
Data	04	Data24	Data25	Data26	Data27	Data28	Data29	Data30	Data31	Yes
Data	05	Data32	Data33	Data34	Data35	Data36	Data37	Data38	Data39	Yes
Data	06	Data40	Data41	Data42	Data43	Data44	Data45	Data46	Data47	Yes
Data	07	Data48	Data49	Data50	Data51	Data52	Data53	Data54	Data55	Yes
Data	08	Data56	Data57	Data58	Data59	Data60	Data61	Data62	Data63	Yes
Data	09	Data64	Data65	Data66	Data67	Data68	Data69	Data70	Data71	Yes
Data	0A	Data72	Data73	Data74	Data75	Data76	Data77	Data78	Data79	Yes
Data	0B	Data80	Data81	Data82	Data83	Data84	Data85	Data86	Data87	Yes
Data	0C	Data88	Data89	Data90	Data91	Data92	Data93	Data94	Data95	Yes
Reserved	0D									N/A
Lock/OTP	0E	LOCK-0	LOCK-1	OTP-0	OTP-1	OTP-2	OTP-3	OTP-4	OTP-5	N/A
Lock/OTP	0F	OTP-6	OTP-7	LOCK-2	LOCK-3	LOCK-4	LOCK-5	LOCK-6	LOCK-7	N/A

EEPROM Memory Map (Segment1)										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
Data	10	Data96	Data97	Data98	Data99	Data100	Data101	Data102	Data103	Yes
Data	11	Data104	Data105	Data106	Data107	Data108	Data109	Data110	Data111	Yes
Data	12	Data112	Data113	Data114	Data115	Data116	Data117	Data118	Data119	Yes
Data	13	Data120	Data121	Data122	Data123	Data124	Data125	Data126	Data127	Yes
Data	14	Data128	Data129	Data130	Data131	Data132	Data133	Data134	Data135	Yes
Data	15	Data136	Data137	Data138	Data139	Data140	Data141	Data142	Data143	Yes
Data	16	Data144	Data145	Data146	Data147	Data148	Data149	Data150	Data151	Yes
Data	17	Data152	Data153	Data154	Data155	Data156	Data157	Data158	Data159	Yes
Data	18	Data160	Data161	Data162	Data163	Data164	Data165	Data166	Data167	Yes
Data	19	Data168	Data169	Data170	Data171	Data172	Data173	Data174	Data175	Yes
Data	1A	Data176	Data177	Data178	Data179	Data180	Data181	Data182	Data183	Yes
Data	1B	Data184	Data185	Data186	Data187	Data188	Data189	Data190	Data191	Yes
Data	1C	Data192	Data193	Data194	Data195	Data196	Data197	Data198	Data199	Yes
Data	1D	Data200	Data201	Data202	Data203	Data204	Data205	Data206	Data207	Yes
Data	1E	Data208	Data209	Data210	Data211	Data212	Data213	Data214	Data215	Yes
Data	1F	Data216	Data217	Data218	Data219	Data220	Data221	Data222	Data223	Yes

EEPROM Memory Map (Segment2)										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
Data	20	Data224	Data225	Data226	Data227	Data228	Data229	Data230	Data231	Yes
Data	21	Data232	Data233	Data234	Data235	Data236	Data237	Data238	Data239	Yes
Data	22	Data240	Data241	Data242	Data243	Data244	Data245	Data246	Data247	Yes
Data	23	Data248	Data249	Data250	Data251	Data252	Data253	Data254	Data255	Yes
Data	24	Data256	Data257	Data258	Data259	Data260	Data261	Data262	Data263	Yes
Data	25	Data264	Data265	Data266	Data267	Data268	Data269	Data270	Data271	Yes
Data	26	Data272	Data273	Data274	Data275	Data276	Data277	Data278	Data279	Yes
Data	27	Data280	Data281	Data282	Data283	Data284	Data285	Data286	Data287	Yes
Data	28	Data288	Data289	Data290	Data291	Data292	Data293	Data294	Data295	Yes
Data	29	Data296	Data297	Data298	Data299	Data300	Data301	Data302	Data303	Yes
Data	2A	Data304	Data305	Data306	Data307	Data308	Data309	Data310	Data311	Yes
Data	2B	Data312	Data313	Data314	Data315	Data316	Data317	Data318	Data319	Yes
Data	2C	Data320	Data321	Data322	Data323	Data324	Data325	Data326	Data327	Yes
Data	2D	Data328	Data329	Data330	Data331	Data332	Data333	Data334	Data335	Yes
Data	2E	Data336	Data337	Data338	Data339	Data340	Data341	Data342	Data343	Yes
Data	2F	Data344	Data345	Data346	Data347	Data348	Data349	Data350	Data351	Yes

EEPROM Memory Map (Segment3)										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
Data	30	Data352	Data353	Data354	Data355	Data356	Data357	Data358	Data359	Yes
Data	31	Data360	Data361	Data362	Data363	Data364	Data365	Data366	Data367	Yes
Data	32	Data368	Data369	Data370	Data371	Data372	Data373	Data374	Data375	Yes
Data	33	Data376	Data377	Data378	Data379	Data380	Data381	Data382	Data383	Yes
Data	34	Data384	Data385	Data386	Data387	Data388	Data389	Data390	Data391	Yes
Data	35	Data392	Data393	Data394	Data395	Data396	Data397	Data398	Data399	Yes
Data	36	Data400	Data401	Data402	Data403	Data404	Data405	Data406	Data407	Yes
Data	37	Data408	Data409	Data410	Data411	Data412	Data413	Data414	Data415	Yes
Data	38	Data416	Data417	Data418	Data419	Data420	Data421	Data422	Data423	Yes
Data	39	Data424	Data425	Data426	Data427	Data428	Data429	Data430	Data431	Yes
Data	3A	Data432	Data433	Data434	Data435	Data436	Data437	Data438	Data439	Yes
Data	3B	Data440	Data441	Data442	Data443	Data444	Data445	Data446	Data447	Yes
Data	3C	Data448	Data449	Data450	Data451	Data452	Data453	Data454	Data455	Yes
Data	3D	Data456	Data457	Data458	Data459	Data460	Data461	Data462	Data463	Yes
Data	3E	Data464	Data465	Data466	Data467	Data468	Data469	Data470	Data471	Yes
Data	3F	Data472	Data473	Data474	Data475	Data476	Data477	Data478	Data479	Yes

Reserved for internal use

User Block Lock & Status

OTP bits

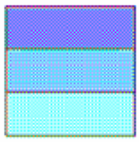
Appendix C. Jewel64

EEPROM Memory Map (Segment0)										
Type	Block No.	Byte-0 **(LSB)**	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 **(MSB)**	Lockable
UID	0	UID-0	UID-1	UID-2	UID-3	UID-4	UID-5	25_h		Locked
Data	1	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Yes
Data	2	Data8	Data9	Data10	Data11	Data12	Data13	Data14	Data15	Yes
Data	3	Data16	Data17	Data18	Data19	Data20	Data21	Data22	Data23	Yes
Data	4	Data24	Data25	Data26	Data27	Data28	Data29	Data30	Data31	Yes
Data	5	Data32	Data33	Data34	Data35	Data36	Data37	Data38	Data39	Yes
Data	6	Data40	Data41	Data42	Data43	Data44	Data45	Data46	Data47	Yes
Data	7	Data48	Data49	Data50	Data51	Data52	Data53	Data54	Data55	Yes
Reserved	8	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	N/A
Reserved	9	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	N/A
Reserved	A	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	N/A
Reserved	B	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	N/A
Data	C	Data56	Data57	Data58	Data59	Data60	Data61	Data62	Data63	Yes
Reserved	D	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	N/A
Lock/OTP	E	LOCK-0	LOCK-1	OTP-0	OTP-1	OTP-2	OTP-3	OTP-4	OTP-5	N/A

Reserved for internal use
User Block Lock & Status
OTP bits

Appendix D. Jewel96

EEPROM Memory Map										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
UID	0	UID-0	UID-1	UID-2	UID-3	UID-4	UID-5	UID-6		Locked
Data	1	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Yes
Data	2	Data8	Data9	Data10	Data11	Data12	Data13	Data14	Data15	Yes
Data	3	Data16	Data17	Data18	Data19	Data20	Data21	Data22	Data23	Yes
Data	4	Data24	Data25	Data26	Data27	Data28	Data29	Data30	Data31	Yes
Data	5	Data32	Data33	Data34	Data35	Data36	Data37	Data38	Data39	Yes
Data	6	Data40	Data41	Data42	Data43	Data44	Data45	Data46	Data47	Yes
Data	7	Data48	Data49	Data50	Data51	Data52	Data53	Data54	Data55	Yes
Data	8	Data56	Data57	Data58	Data59	Data60	Data61	Data62	Data63	Yes
Data	9	Data64	Data65	Data66	Data67	Data68	Data69	Data70	Data71	Yes
Data	A	Data72	Data73	Data74	Data75	Data76	Data77	Data78	Data79	Yes
Data	B	Data80	Data81	Data82	Data83	Data84	Data85	Data86	Data87	Yes
Data	C	Data88	Data89	Data90	Data91	Data92	Data93	Data94	Data95	Yes
Reserved	D									
Lock/Reserved	E	LOCK-0	LOCK-1	OTP-0	OTP-1	OTP-2	OTP-3	OTP-4	OTP-5	



Reserved for internal use
User Block Lock & Status
OTP bits

Appendix E. ACR122 Error Codes

Error	Error Code
No Error	**0x00**
Time Out, the target has not answered	0x01
A CRC error has been detected by the contactless UART	0x02
A Parity error has been detected by the contactless UART	0x03
During a MIFARE anti-collision/select operation, an erroneous Bit Count has been detected	0x04
Framing error during MIFARE operation	0x05
An abnormal bit-collision has been detected during bit wise anti-collision at 106 kbps	0x06
Communication buffer size insufficient	0x07
RF Buffer overflow has been detected by the contactless UART (bit BufferOvfl of the register CL_ERROR)	0x08
In active communication mode, the RF field has not been switched on in time by the counterpart (as defined in NFCIP-1 standard)	0x0A
RF Protocol error (cf. reference [4], description of the CL_ERROR register)	0x0B
Temperature error: the internal temperature sensor has detected overheating, and therefore has automatically switched off the antenna drivers	0x0D
Internal buffer overflow	0x0E
Invalid parameter (range, format, …)	0x10
DEP Protocol: The chip configured in target mode does not support the command received from the initiator (the command received is not one of the following: ATR_REQ, WUP_REQ, PSL_REQ, DEP_REQ, DSL_REQ, RLS_REQ, ref. [1]).	0x12
DEP Protocol / Mifare / ISO/IEC 14443-4: The data format does not match to the specification. Depending on the RF protocol used, it can be: • Bad length of RF received frame, • Incorrect value of PCB or PFB, • Invalid or unexpected RF received frame, • NAD or DID incoherence.	0x13
Mifare: Authentication error	0x14
ISO/IEC 14443-3: UID Check byte is wrong	0x23
DEP Protocol: Invalid device state, the system is in a state which does not allow the operation	0x25
Operation not allowed in this configuration (host controller interface)	0x26
This command is not acceptable due to the current context of the chip (Initiator vs. Target, unknown target number, Target not in the good state, …)	0x27
The chip configured as target has been released by its initiator	0x29
ISO/IEC 14443-3B only: the ID of the card does not match, meaning that the expected card has been exchanged with another one.	0x2A
ISO/IEC 14443-3B only: the card previously activated has disappeared.	0x2B
Mismatch between the NFCID3 initiator and the NFCID3 target in DEP 212/424 kbps passive.	0x2C
An over-current event has been detected	0x2D
NAD missing in DEP frame	0x2E

FIGARO

PRODUCT INFORMATION

TGS 2602 - for the detection of Air Contaminants

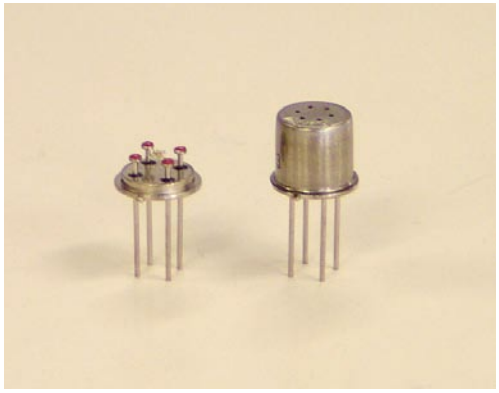
Features:

* High sensitivity to VOCs and odorous gases
* Low power consumption
* High sensitivity to gaseous air contaminants
* Long life
* Uses simple electrical circuit
* Small size

Applications:

* Air cleaners
* Ventilation control
* Air quality monitors
* VOC monitors
* Odor monitors

The sensing element is comprised of a metal oxide semiconductor layer formed on the alumina substrate of a sensing chip together with an integrated heater. In the presence of detectable gas, sensor conductivity increases depending on gas concentration in the air. A simple electrical circuit can convert the change in conductivity to an output signal which corresponds to the gas concentration.



The **TGS 2602** has high sensitivity to low concentrations of odorous gases such as ammonia and H_2S generated from waste materials in office and home environments. The sensor also has high sensitivity to low concentrations of VOCs such as toluene emitted from wood finishing and construction products. Figaro also offers a microprocessor (FIC02667) which contains special software for handling the sensor's signal for appliance control applications.

Due to miniaturization of the sensing chip, TGS 2602 requires a heater current of only 42mA and the device is housed in a standard TO-5 package.

The figure below represents typical sensitivity characteristics, all data having been gathered at standard test conditions (see reverse side of this sheet). The Y-axis is indicated as sensor resistance ratio (Rs/Ro) which is defined as follows:

Rs = Sensor resistance in displayed gases at various concentrations

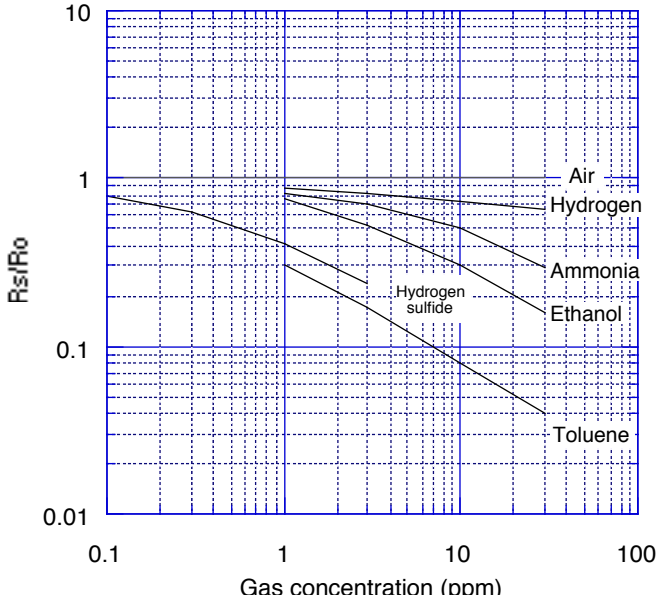
Ro = Sensor resistance in fresh air

The figure below represents typical temperature and humidity dependency characteristics. Again, the Y-axis is indicated as sensor resistance ratio (Rs/Ro), defined as follows:

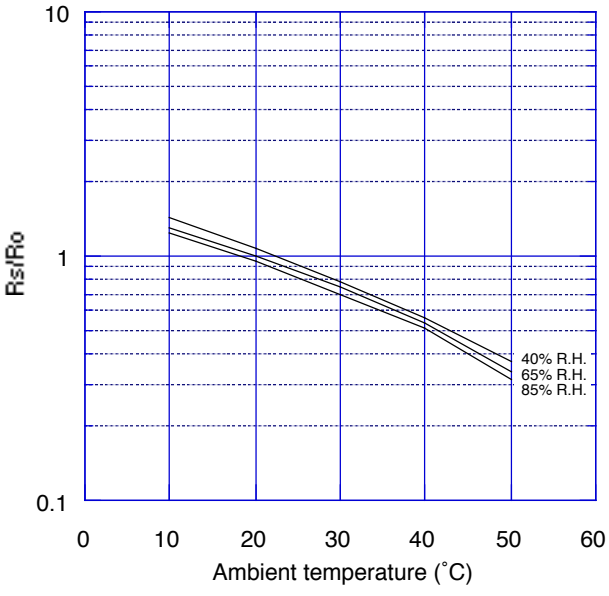
Rs = Sensor resistance in fresh air at various temperatures/humidities

Ro = Sensor resistance in fresh air at 20°C and 65% R.H.

Sensitivity Characteristics:

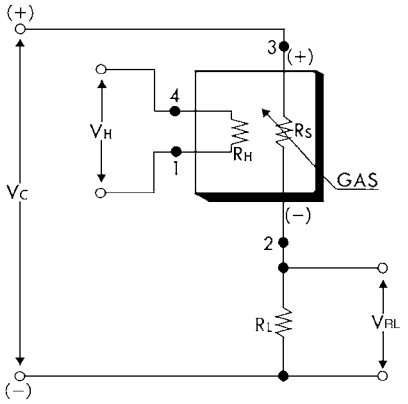


Temperature/Humidity Dependency:



Basic Measuring Circuit:

The sensor requires two voltage inputs: heater voltage (V_H) and circuit voltage (V_C). The heater voltage (V_H) is applied to the integrated heater in order to maintain the sensing element at a specific temperature which is optimal for sensing. Circuit voltage (V_C) is applied to allow measurement of voltage (Vout) across a load resistor (R_L) which is connected in series with the sensor.
DC voltage is required for the circuit voltage since the sensor has a polarity. A common power supply circuit can be used for both V_C and V_H to fulfill the sensor's electrical requirements. The value of the load resistor (R_L) should be chosen to optimize the alarm threshold value, keeping power consumption (P_S) of the semiconductor below a limit of 15mW. Power consumption (P_S) will be highest when the value of Rs is equal to R_L on exposure to gas.



Specifications:

Model number			TGS 2602-B00	
Sensing element type			D1	
Standard package			TO-5 metal can	
Target gases			Air contaminants	
Typical detection range			1 ~ 30 ppm of EtOH	
Standard circuit conditions	Heater voltage	V_H	5.0±0.2V DC/AC	
	Circuit voltage	V_C	5.0±0.2V DC	Ps≤15mW
	Load resistance	R_L	Variable	0.45kΩ min.
Electrical characteristics under standard test conditions	Heater resistance	R_H	approx. 59Ω at room temp.	
	Heater current	I_H	56±5mA	
	Heater power consumption	P_H	280mW (typical)	
	Sensor resistance	Rs	10k~100kΩ in air	
	Sensitivity (change ratio of Rs)		0.15~0.5	Rs (10ppm of EtOH) / Rs (air)
Standard test conditions	Test gas conditions		normal air at 20±2˚C, 65±5%RH	
	Circuit conditions		V_C = 5.0±0.01V DC V_H = 5.0±0.05V DC	
	Conditioning period before test		7 days	

The value of power consumption (P_S) can be calculated by utilizing the following formula:

$$P_S = \frac{(V_C - Vout)^2}{R_S}$$

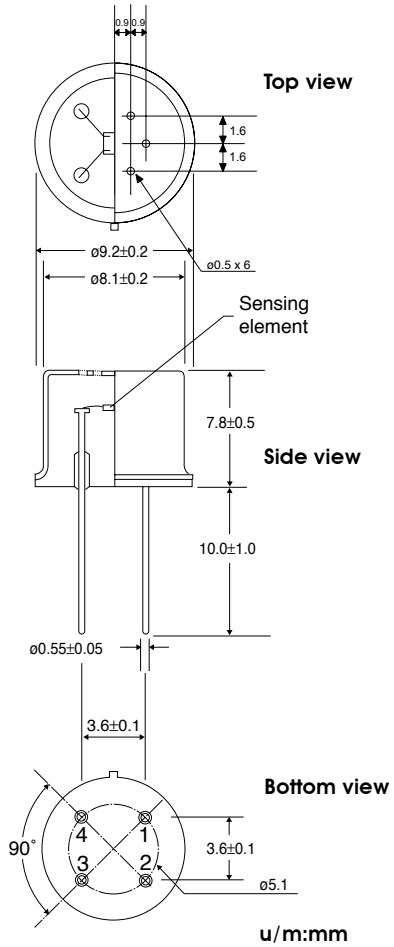
Sensor resistance (Rs) is calculated with a measured value of Vout by using the following formula:

$$R_S = \frac{V_C \times R_L}{Vout} - R_L$$

For information on warranty, please refer to Standard Terms and Conditions of Sale of Figaro USA Inc. All sensor characteristics shown in this brochure represent typical characteristics. Actual characteristics vary from sensor to sensor. The only characteristics warranted are those in the Specification table above.

REV: 01/05

Structure and Dimensions:



Pin connection:
1: Heater
2: Sensor electrode (-)
3: Sensor electrode (+)
4: Heater

FIGARO USA, INC.
121 S. Wilke Rd. Suite 300
Arlington Heights, IL 60005
Phone: (847)-832-1701
Fax: (847)-832-1705
e-mail: figarousa@figarosensor.com

TGS2602 空气污染、臭味检测用

特点：

- 低功耗
- 对 VOC、氨气、硫化氢有高灵敏度
- 长寿命、低成本
- 可利用简单电路

敏感素子由集成的加热器以及在氧化铝基板上形成的金属氧化物半导体构成。当可检知的气体存在时，空气中该气体的浓度越高，传感器的电导率就越高。使用简单的电路就可以将这种电导率的变化变换为与气体浓度对应的输出信号。

TGS2602 不仅对香烟的烟雾或烹调臭味有很高的灵敏度，而且对硫化氢、VOC、氨气有高灵敏度。这种传感器是利用相对值检知来实现更接近人类感觉的控制，即以空气清洁的时候为基准，通过传感器电阻值比空气清洁时变化了多少来检测空气的污染程度。

下图是典型的灵敏度特性，全部是在标准试验条件下得出的结果。（请看背面）

纵轴以传感器电阻比 Rs/Rs(Air)表示，Rs、Rs(Air)的定义如下：

Rs　　＝各种浓度气体中的传感器电阻值

Rs(Air)＝清洁大气中的传感器电阻值

应用：

- 空气清新机、换气扇控制
- 脱臭器控制
- 室内空气监视器

下图是典型的香烟烟雾灵感度特性。

香烟的根数是约 10 平米的房间吸烟情况下的数值。

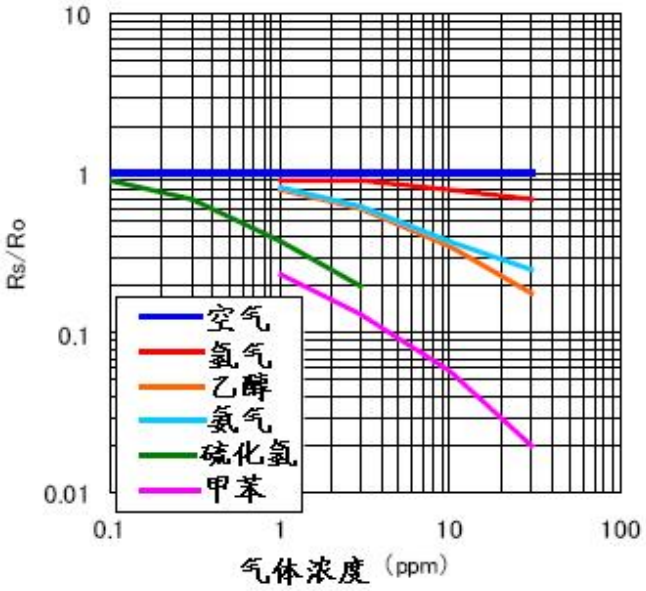
这里的纵轴也用传感器电阻比 Rs/Rs(Air)来表示，

这里的 Rs、Rs(Air)定义如下：

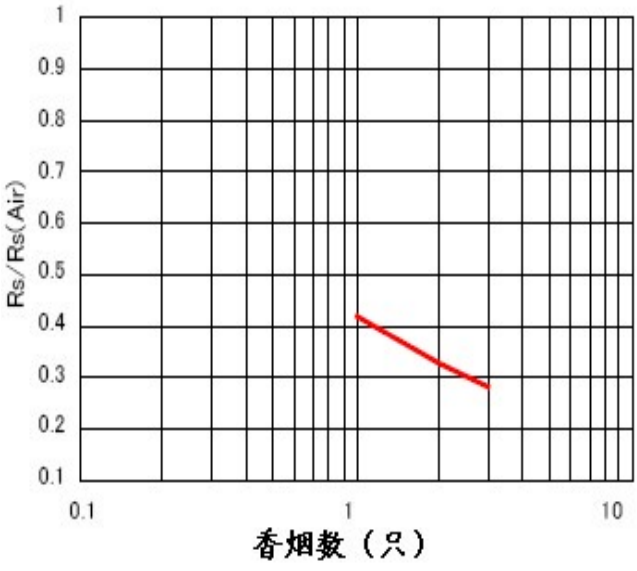
Rs　　＝香烟的烟雾存在时的传感器电阻值

Rs(Air)　＝清洁大气中的传感器电阻值

灵敏度特性：



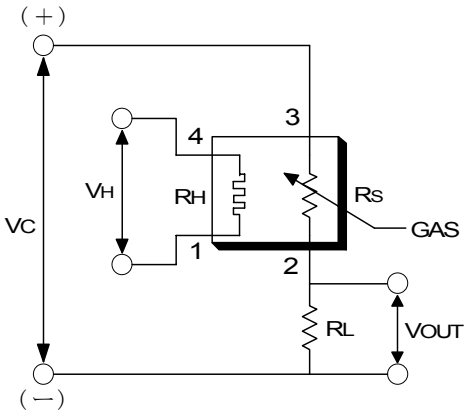
香烟灵敏度特性：



基本测试回路：

此传感器需要施加 2 个电压：加热器电压（V_H）和回路电压（V_C）。这个V_H用于维持敏感素子处于与对象气体相适应的特定温度而施加在集成的加热器上。V_C则是用于测定与传感器串联的负载电阻(R_L)上的两端电压(V_{OUT})。

这种传感器具有极性，所以V_C需用直流电源。
只要能满足传感器的电性要求，V_C和V_H可以共用同一个电源电路。为了将判定值水平最佳化，并使敏感素子的功耗（P_S）低于 15mW的限度值，需要选择R_L的值。



规格：

型　　号			TGS2602	
素子类型			26 系列	
标准封装			金属	
对象气体			氢气、酒精等	
检测范围			1 ～10 ppm	
标准回路	加热器电压	VH	5.0±0.2V DC/AC	
	回路电压	VC	5.0±0.2V DC	Ps≦15mW
	负载电阻	RL	可变	Ps≦15mW
标准试验	加热器电阻	RH	59 Ω（室温）	
	加热器电流	IH	56mA	
	加热器功耗	PH	280mW	VH＝5.0V DC/AC
	传感器电阻	Rs	10～100 KΩ（空气中）	
	灵敏度 (Rs 的变化率)		0.15～0.5	Rs(乙醇：10) Rs(Air)
标准试验	试验气体条件		20±2℃，65±5%RH	
	回路条件		VC＝5.0±0.2V DC VH＝5.0±0.2V DC/AC	
	试验前预热时间		96 小时以上	

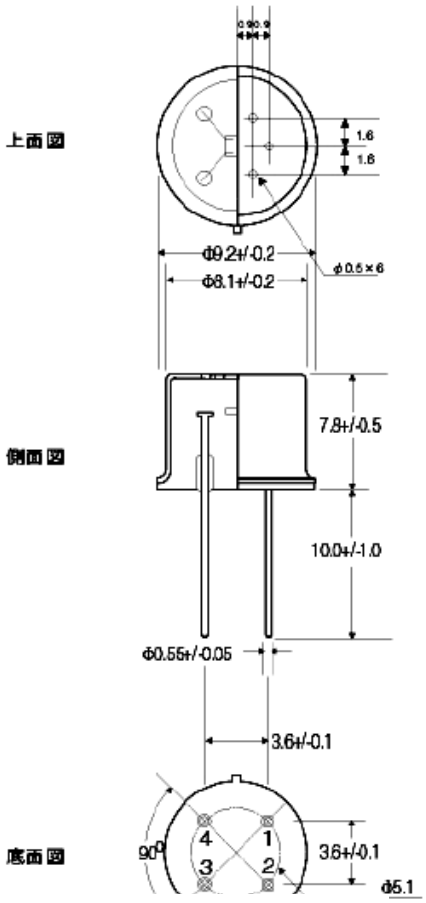
功耗（Ps）值可用下式计算：

$$Ps=\frac{(Vc-VOUT)^2}{Rs}$$

传感器电阻（Rs），可根据 VOUT 测定值，用下式计算：

$$Rs=\frac{Vc\times RL}{VRL}-RL$$

结构及尺寸：



1：加热器电极
2：传感器电极（－）
3：传感器电极（＋）
4：加热器电极

为提高性能，本规格书将不事先预告而变更。

天津费加罗电子有限公司
天津经济技术开发区微山路 19 号
TEL：022-6620-3977　　FAX：2532-5913
WEB：　www.tjfigaro.com

DMATEK

ARM周邊配件系列

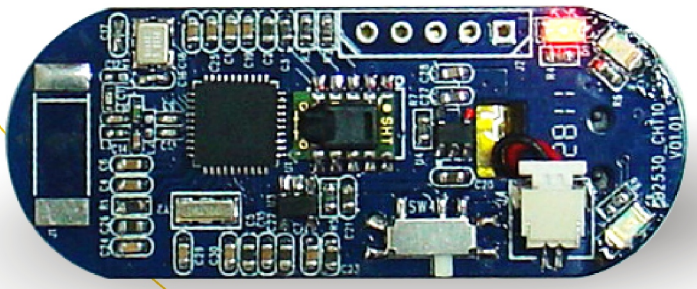
ZigBee 溫濕度感測無線通訊模組

用户手册

作業系統：WinCE 6.0 / Android 2.1

適用平台：DMA-210U / DMA-210XP
DMA-6410XP / DMA-6410L / DMA-6410
HMI430/500/700-6410S

產品編號：ZB2530-SHT10



目錄

目錄

第〇章　導 讀

0-1　手冊內容簡介

本手冊是爲了讓讀者有更好的瞭解、熟悉及操作 ZB2530-SHT-GS 感測器無線通訊模組而編寫的，同時讀者還可以 ZigBee Debugger 與 CC2530-ZigBee 之間燒錄的使用。本手冊第一章主要介紹了 ZigBee 的功能描述和基本使用說明；第二章則介紹了 ZB2530-SHT-GS 無線通訊模組的特色及相關配件和基本使用說明；第三章是 IAR 開發環境的架構與開發流程；第四章則介紹了感測器圖控軟體；附錄一是 ZB2530-01 模組的介紹；附錄二是 ZigBee Debugger 模擬器的介紹；附錄三是 SHT10 溫濕度感測器的資料；附錄四是 ZB2530-SHT-GS 的電路圖。

0-2 光碟內容介紹

本產品光碟內容包括：

目錄名稱	主要檔描述
IAR\CC2530_lib\ basicrf	CC2530 單片機的無線函數庫
IAR\CC2530_lib\ board	ZIGBEE 模組上的資源函數庫
IAR\CC2530_lib\ common	CC2530 單片機的通用函數庫
IAR\CC2530_lib\ module	本公司 8 個感測器模組函數庫
IAR\CC2530_lib\ utils	CC2530 單片機的工具函數庫
IAR\software	IAR 安裝軟體以及模擬器的驅動軟體
IAR\source_exp\ SHT10+BMA	溫濕度感測器+Gsensor 原始程式碼
IAR\hex\ SHT10+BMA.hex	溫濕度感測器+Gsensor 燒寫檔案
IAR\hex\ rf_send_recv.hex	感測器無線接受端（主）燒寫檔

第一章　ZigBee 概述

■ 什麼是 ZigBee

ZigBee是IEEE 802.15.4 協議的代名詞。根據這個協定規定的技術是一種短距離、低功耗的無線通信技術。ZigBee 的命名，源自於蜜蜂的八字舞，由於蜜蜂(bee)是靠飛翔和"嗡嗡" (zig) 地抖動翅膀的"舞蹈"來與同伴傳遞花粉所在方位資訊，也就是說蜜蜂依靠這樣的方式構成了群體中的通信網路。其特點是短距離、低複雜度、低功耗、低資料速率、低成本。主要應用的方向在於家庭裝置自動化、環境安全與控制、個人醫療照護、自動控制和遠端控制領域、嵌入式各種設備。

簡而言之，ZigBee 就是一種便宜，低功耗的近距離無線組網通訊技術。

■ ZigBee 的起源

ZigBee 在中國被譯爲"紫蜂"，它與藍芽相類似.是一種新興的短距離無線技術，用於感測控制應用 (sensor and control)。此想法在 IEEE 802.15 工作組中提出，於是成立了 TG4 工作組，並制定規範 IEEE 802.15.4。

2002年，ZigBee Alliance 成立。

2004年，ZigBee V1.0 誕生，它是 ZigBee 第一個規範，但由於推出倉促，存在一些錯誤。

2006年，推出 ZigBee 2006，比較完善。

2007 年底，ZigBee PRO 推出。

ZigBee 的底層技術基於IEEE 802.15.4.

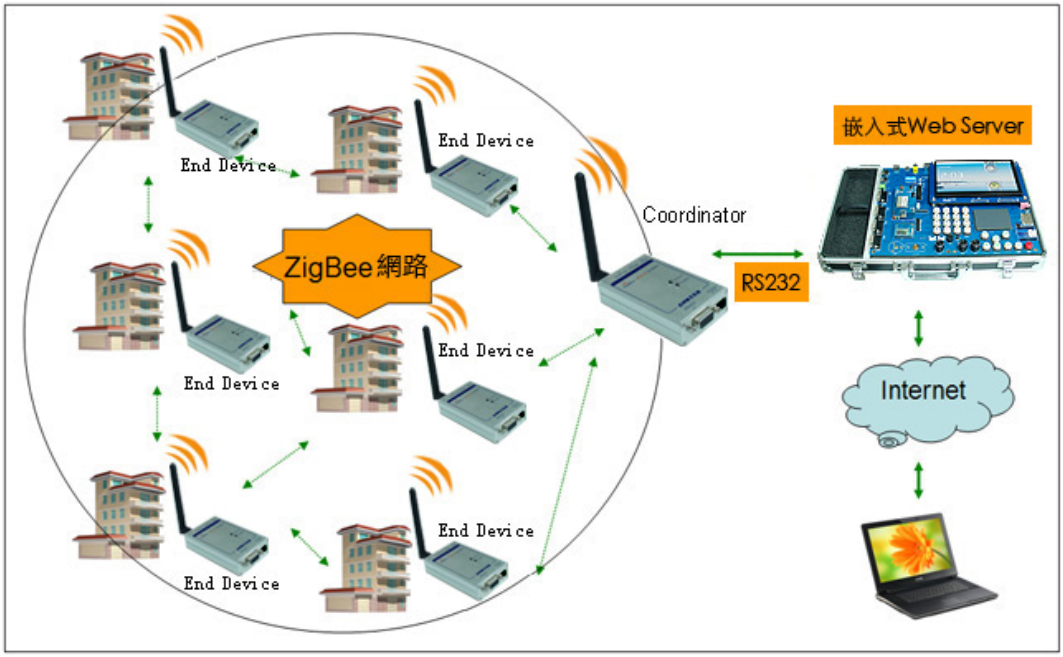
物理層和 MAC 層直接引用了IEEE 802.15.4

在藍芽技術的使用過程中，人們發現藍芽技術儘管有許多優點，但仍存在許多缺陷。對工業，家庭自動化控制和工業遙測遙控領域而言，藍芽技術顯得太複雜、功耗大、距離近、組網規模太小等，而工業自動化，對無線資料通信的需求越來越強烈，而且，對於工業現場，這種無線資料傳輸必須是高可靠性的，並能抵抗工業現場的各種電磁干擾。因此，經過人們長期努力，ZigBee 協議在 2003 年正式問世。 另外，ZigBee 使用了在它之前所研究過的面向家庭網路的通信協定 Home RF Lite。

長期以來，低價、低傳輸率、短距離、低功率的無線通訊市場一直存在著。自從Bluetooth 出現以後，曾讓工業控制、家用自動控制、玩具製造商等業者雀躍不已，但是 Bluetooth 的售價一直居高不下，嚴重影響了這些廠商的使用意願。如今，這些業者都參加了IEEE802.15.4 小組，負責制定 ZigBee 的物理層和媒體介入控制層。IEEE802.15.4 規範是一種經濟、高效、低資料速率（<250kbps）、工作在 2.4 GHz 和 868/928MHz 的無線技術，用於個人區域網和對等網路。它是 ZigBee 應用屋和網路層協定的基礎。ZigBee是一種新興的近距離、低複雜度、低功耗、低資料速率、低成本的無線網路技術，它是一種介於無線標記技術和藍芽之間的技術提案。主要用於近距離無線連接。它依據 802.15.4 標準，在數千個微小的感測器之

間相互協調實現通信。這些感測器只需要很少的能量，以接力的方式透過無線電波將資料從一個感測器傳到另一個感測器，所以它們的通信效率非常高。

■ ZigBee 無線資料傳輸網路描述



圖：ZigBee 無線抄表系統

簡單的說，ZigBee 是一種高可靠的無線數傳網路，類似於 CDMA 和 GSM 網路。ZigBee 數傳模組類似於移動網路基站。通訊距離從標準的 75m 到幾百米、幾公里，並且支持無限擴展。

ZigBee 是一個由可多到 65000 個無線數傳模組組成的一個無線數傳網路平台，在整個網路範圍內，每一個 ZigBee 網路數傳模組之間可以相互通信，每個網路節點間的距離可以從標準的 75m 無限擴展。

與移動通信的 CDMA 或 GSM 不同的是，ZigBee 網路主要是爲工業現場自動化控制資料傳輸而建立。因此，它必須具有簡單，使用方便，可靠高，價格低的特點。而移動通信網主要是爲語音通信而建立，每個基地台價值一般都在百萬元以上，而每個 ZigBee 設備卻不到 4000 元。每個 ZigBee 網路節點不僅本身可以作爲監控

物件，例如其所連接的感測器直接進行資料獲取和監控，還可以自動中轉別的網路節點傳過來的資料資料。除此之外，每一個 ZigBee 網路節點(FFD)還可在自己信號覆蓋的範圍內，和多個不承擔網路資訊中轉任務的孤立的子節點(RFD)無線連接。

■ ZigBee 採用的自組織網通信方式

➢ ZigBee 技術所採用的自組織網是怎麼回事？

舉一個簡單的例子就可以說明這個問題，當一隊傘兵空降後，每人持有一個 ZigBee 網路模組終端，降落到地面後，只要他們彼此間在網路模組的通信範圍內，透過彼此自動尋找，很快就可以形成一個互聯互通的 ZigBee 網路。而且，由於人員的移動，彼此間的聯絡還會發生變化。因而，模組還可以透過重新尋找通信物件，確定彼此間的聯絡，對原有網路進行更新。這就是自組織網。

➢ ZigBee 技術爲什麼要使用自組織網來通信？

網狀網通信實際上就是多通道通信，在實際工業現場，由於各種原因，往往並不能保證每一個無線通道都能夠始終暢通，就像城市的街道一樣，可能因爲車禍，道路維修等，使得某條道路的交通出現暫時中斷，此時由於我們有多個通道，車輛（相當於我們的控制資料）仍然可以透過其他道路到達目的地。而這一點對工業現場控制而言則非常重要。

➢ 爲什麼自組織網要採用動態路由的方式？

所謂動態路由是指網路中資料傳輸的路徑並不是預先設定的，而是傳輸資料前，透過對網路當時可利用的所有路徑進行搜索，分析它們的位置關係以及遠近，然後選擇其中的一條路徑進行資料傳輸。在我們的網路管理軟體中，路徑的選擇使用的是"梯度法"，即先選擇路徑最近的一條通道進行傳輸，如傳不通，再使用另外一條稍遠一點的通路進行傳輸，以此類推，直到資料送達目的地爲止。在實際工業現場，預先確定的傳輸路徑隨時都可能發生變化，或者因各種原因路徑被中斷了，或者過於繁忙不能進行及時傳送。動態路由結合網狀拓撲結構，就可以很好解決這個問題，從而保證資料的可靠傳輸。

■ ZigBee 自身的技術優勢

① 低功耗。在低耗電待機模式下，2節 5號乾電池可支持 1 個節點工作 6～24 個月，甚至更長。這是 ZigBee 的突出優勢。相比較，藍芽能工作數周、WiFi 可工作數小時。

現在，TI 公司和德國的 MicropeIt 公司共同推出新能源的 ZigBee 節點。該節點採用 MicropeIt 公司的熱電發電機給 TI 公司的 ZigBee 提供電源。

② 低成本。透過大幅簡化協議(不到藍芽的1/10)，降低了對通信控制器的要求，按預測分析，以8051 的 8 位微控制器測算，全功能的主節點需要 32KB 程式碼，子功能節點少至 4KB 程式碼，而且 ZigBee 免協議專利費。每塊晶片的價格大約爲 2美元。

③ 低速率。ZigBee 工作在20～250 kbps的較低速率，分別提供250 kbps (2.4GHz)、40kbps (915 MHz)和 20kbps(868 MHz) 的原始資料吞吐率，滿足低速率傳輸資料的應用需求。

④ 近距離。傳輸範圍一般介於10～100m之間，在增加RF發射功率後，亦可增加到1～3 km。這指的是相鄰節點間的距離。如果透過路由和節點間通信的接力，傳輸距離將可以更遠。

⑤ 短時延。ZigBee的回應速度較快，一般從睡眠轉入工作狀態只需15ms，節點連接進入網路只需30 ms，進一步節省了電能。相比較，藍芽需要3～10s、WiFi 需要3 s。

⑥ 高容量。ZigBee可採用星狀、片狀和網狀網路結構，由一個主節點管理若干子節點，最多一個主節點可管理 254個子節點；同時主節點還可由上一層網路節點管理，最多可組成65000個節點的大網。

⑦ 高安全。ZigBee 提供了三級安全模式,包括無安全設定、使用接入控制清單(ACL)防止非法獲取資料以及採用高級加密標準(AES 128) 的對稱密碼，以靈活確定其安全屬性。

⑧ 免執照頻段。採用直接序列擴頻在工業科學醫療(ISM)頻段，2.4 GHz (全球)、915MHz(美國)和 868 MHz (歐洲)。

■ ZigBee 的頻率

1) 868MHZ 傳輸速率爲 20KB/S 適用於歐洲

2) 915MHZ 傳輸速率爲 40KB/S 適用於美國

3) 2.4GHZ 傳輸速率爲 250KB/S 全球通用

由於此三個頻帶物理層並不相同，其各自通道帶寬也不同，分別爲 0.6MHZ，2MHz 和 5MHz；分別有 1 個 10 個和 16 個通道。

不同頻帶的擴頻和調製方式有區別，雖然都使用了直接擴頻(DSSS)的方式，但從比特到碼片的變換方式有較大的差別。

調製方式都用了調相技術，但 868MHZ 和 915MHZ 頻段採用的是 BPSK，而 2.4GHZ 頻段採用的是 OQPSK。

在發射功率爲 0dBm 的情況下，BLUETOOTH 通常能用 10M 的作用範圍。

而基於IEEE 802.15.4 的ZigBee 在室內通常能達到30-50米作用距離，在室外如果障礙物少，甚至可以達到 100 米作用距離。

所以 ZigBee 可歸爲低速率的短距離無線通信技術。

■ ZigBee 性能分析

1、**資料速率比較低** 在2.4GHZ 的頻段只有250Kb/S，而且只是鏈路上的速率，除掉通道競爭應答和重傳等消耗，真正能被應用所利用的速率可能不足 100Kb/S，並且餘下的速率可能要被鄰近多個節點和同一個節點的多個應用所瓜分。因此不適合做視頻之類事情。

適合的應用領域 — 傳感和控制

2、**可靠性** 在可靠性方面，ZigBee 有很多方面進行保證。物理層採用了擴頻技術，能夠在一定程度上抵抗干擾。

MAC應用層(APS 部分)有應答重傳功能。

MAC層的CSMA機制使節點發送前先監聽信道，可以起到避開干擾的作用。

當 ZigBee 網路受到外界干擾，無法正常工作時，整個網路可以動態的切換到另一個工作通道上。

3、**時延** 由於 ZigBee 採用隨機接入 MAC 層，且不支援時分複用的通道接入方式，因此不能很好的支持一些即時的業務。

4、**能耗特性**　能耗特性是 ZigBee 的一個技術優勢。

通常ZigBee節點所承載的應用資料速率都比較低，在不需要通信時，節點可以進入很低功耗的休眠狀態，此時能耗可能只有正常工作狀態下的千分之一。由於一般情況下，休眠時間占總執行時間的大部分，有時正常工作的時間還不到百分之一，因此達到很高的節能效果。

5、**組網和路由性** — 網路層特性

ZigBee 大規模的組網能力— 每個網路 65000 個節點

Bluetooth — 每個網路 8 個節點

因爲ZigBee 底層採用了直擴技術，如果採用非信標模式，網路可以擴展得很大，因爲不需同步而且節點加入網路和重新加入網路的過程很快，一般可以做到 1 秒以內，甚至更快。

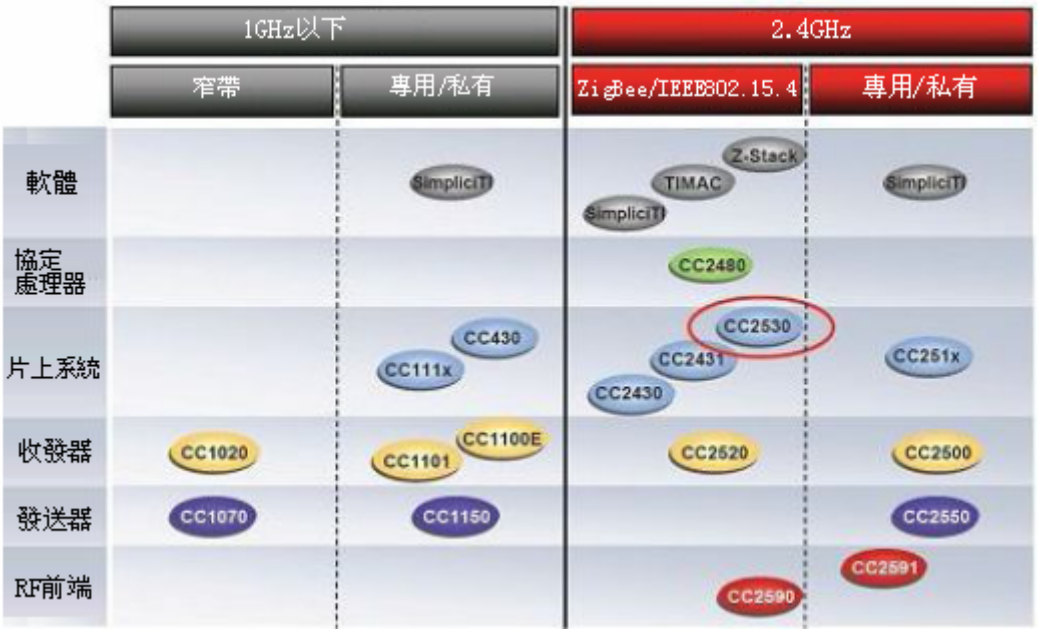
Bluetooth 通常需要 3 秒。

在路由方面，ZigBee支持可靠性很高的網狀網的路由，所以可以佈置範圍很廣的網路，並支援多播和廣播特性，能夠給豐富的應用帶來有力的支持。

■ ZigBee 的應用前景

ZigBee並不是用來與藍芽或者其他已經存在的標準競爭，它的目標定位于現存的系統還不能滿足其需求的特定的市場，它有著廣闊的應用前景。ZigBee 聯盟預言在未來的四到五年，每個家庭將擁有 50 個 ZigBee 器件，最後將達到每個家庭150個。據估計，到2007年，ZigBee 市場價值將達到數億美元。其應用領域主要包括：

- ◆ 家庭和樓宇網路：空調系統的溫度控制、照明的自動控制、窗簾的自動控制、煤氣計量控制、家用電器的遠程控制等；
- ◆ 工業控制：各種監控器、感測器的自動化控制；
- ◆ 商業：智慧型標籤等；
- ◆ 公共場所：煙霧探測器等；
- ◆ 農業控制：收集各種土壤資訊和氣候資訊；
- ◆ 醫療：老人與行動不便者的緊急呼叫器和醫療感測器等。



圖：TI RF 產品 Road Map

■ ZigBee 聯盟

ZigBee 聯盟是一個高速成長的非盈利業界組織，成員包括國際著名半導體生產商、技術提供者、技術集成商以及最終使用者。聯盟制定了基於 IEEE802.15.4，具有高可靠、高性價比、低功耗的網路應用規格。

ZigBee 聯盟的主要目標是以透過加入無線網路功能，爲消費者提供更富有彈性、更容易使用的電子產品。ZigBee 技術能融入各類電子產品，應用範圍橫跨全球的民用、商用、公共事業以及工業等市場。使得聯盟會員可以利用ZigBee 這個標準化無線網路平台，設計出簡單、可靠、便宜又節省電力的各種產品來。

ZigBee 聯盟所鎖定的焦點爲制定網路、安全和應用軟體層；提供不同產品的協調性及互通性測試規格；在世界各地推廣ZigBee 品牌並爭取市場的關注；管理技術的發展。

ZigBee 聯盟對 ZigBee 標準的制定：IEEE802.15.4 的物理層、MAC層及資料連結層，標準已在 2003 年 5 月發佈。ZigBee 網路層、加密層及應用描述層的制定也取得了較大的進展。V1.0 版本已經發佈。其他應用領域及其相關的設備描述也會陸續發佈。由於 ZigBee 不僅只是 802.15.4 的代名詞，而且 IEEE 僅處理低

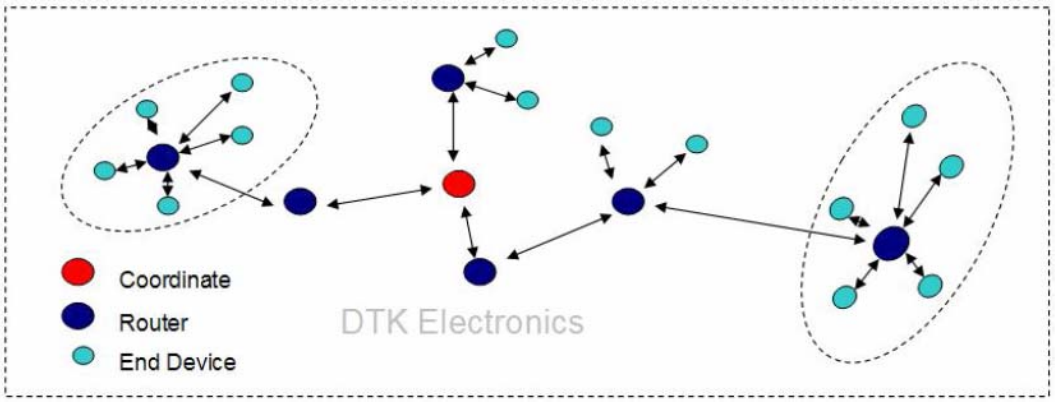
級MAC 層和物理層協定，因此 ZigBee 聯盟對其網路層協定和 API 進行了標準化。完全協定用於一次可直接連接到一個設備的基本節點的 4K 位元組或者作爲 Hub 或路由器的協調器的 32K 位元組。每個協調器可連接多達 255 個節點，而幾個協調器則可形成一個網路，對路由傳輸的數目則沒有限制。

ZigBee 聯盟還開發了安全層，以保證這種便攜設備不會意外洩漏其標識，而且這種利用網路的遠距離傳輸不會被其他節點獲得。

■ ZigBee 的網路拓撲

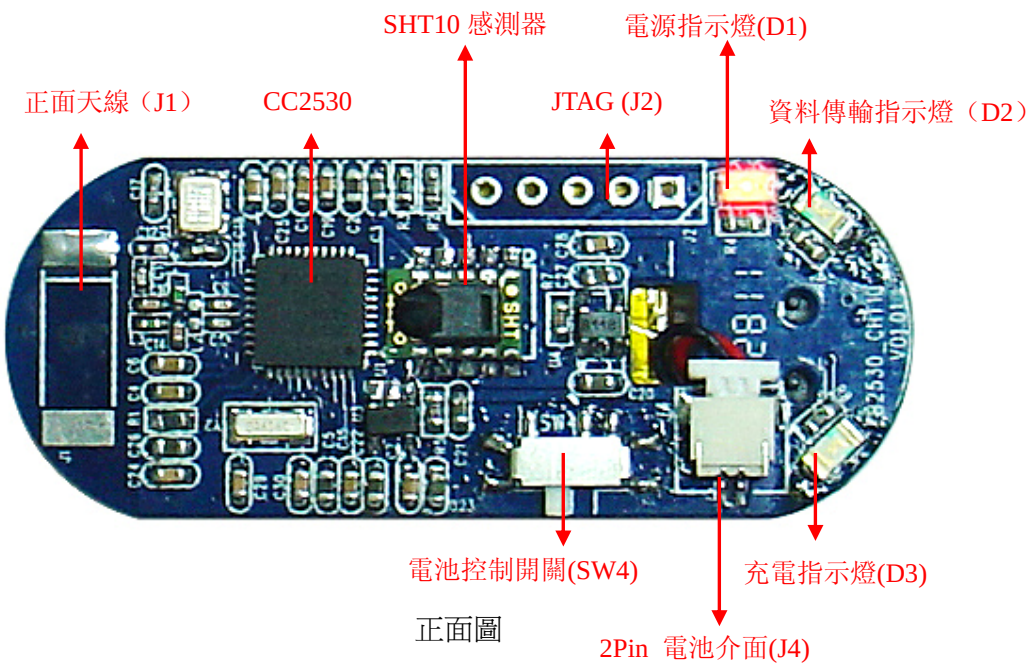
在 ZigBee 網路中，定義了三種網路角色，分別是 Coordinator（網路協調器節點），Router（網路路由器節點），End Device（網路終端節點）。

- Coordinator（網路協調器節點）：負責網路的建立（WPAN Formation）及網路位址（Short Address）的分配。
- Router（網路路由器節點）：負責找尋、建立及修復資料包路由路徑（Routing Path），並負責轉送資料包，同時也可配置網路位址（Short Address）給子節點（Child）。
- End Device（網路終端節點）：只能選擇加入已經形成的網路，可傳送資料。

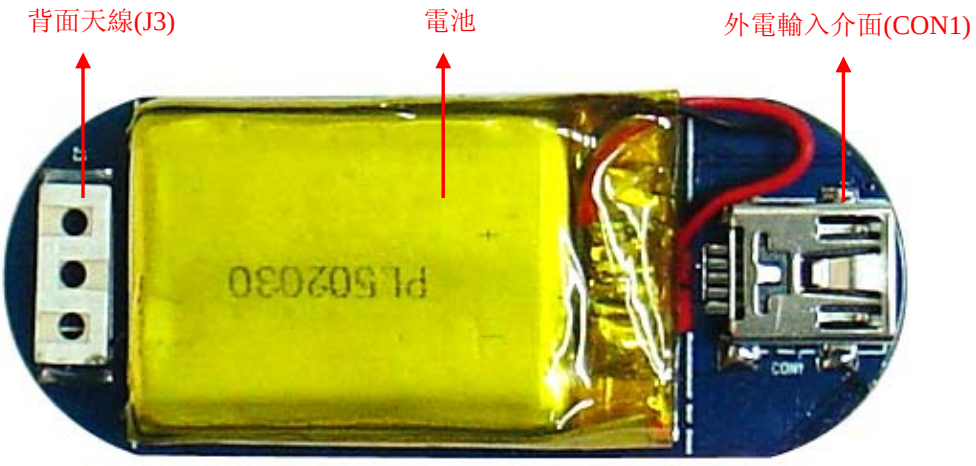


第二章　硬體介紹

2-1 產品外觀圖及介面說明



正面圖



背面圖

1、電源指示燈—紅燈（D1）

2、JTAG 燒寫介面（J2）

3、SHT10 感測器

4、CC2530 處理器

5、正面天線介面（J1）

6、電池控制開關（SW4）

7、電池介面（J4）

8、充電指示—綠燈（D3）

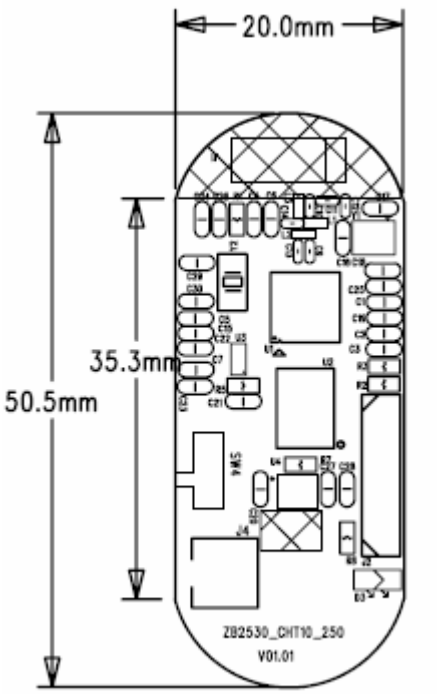
9、資料傳輸指示---綠燈（D2）

10、外電輸入介面（CON1）

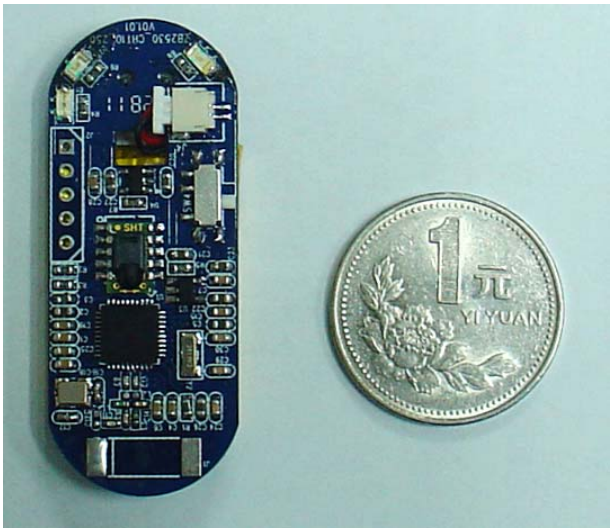
11、電池

12、背面天線（J3）

2-2　結構尺寸



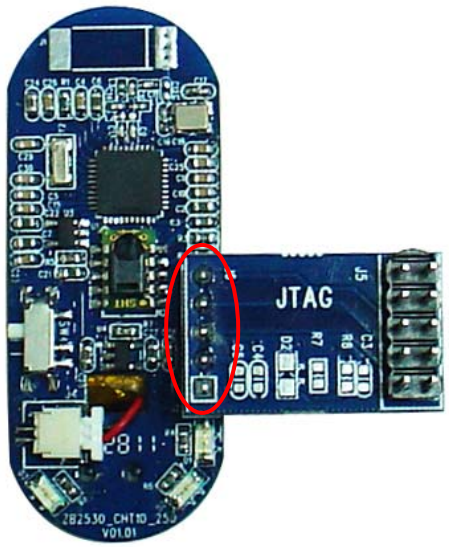
PCB 結構尺寸圖



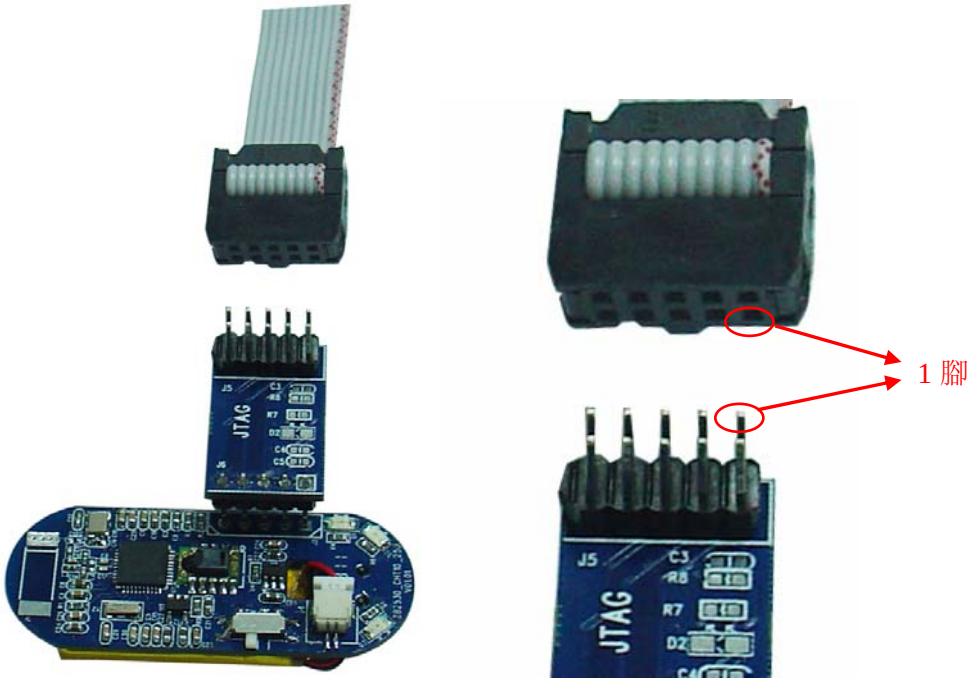
與一元硬幣作比較

2-3 操作說明

1. 燒寫 ZB2530-SHT10 時，先將 JTAG 轉接板燒寫頭接入 J2 介面，如下圖連接所示：

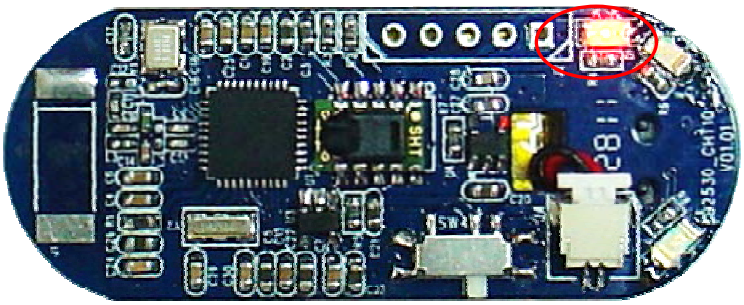


2. 接著將燒寫器數據線一端與 JTAG 轉接板連接，如下圖所示，注意 1 腳的位置：



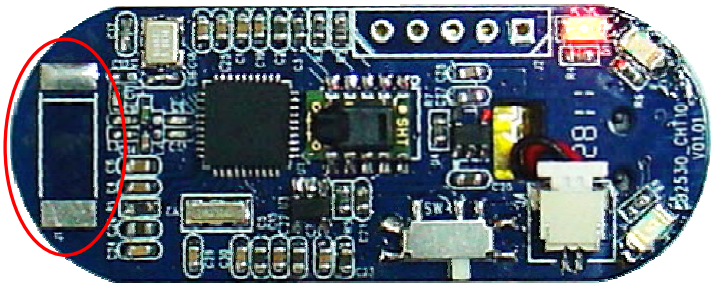
接好後如下圖所示：

3. 打開電池控制開關，電源正常時，電源指示燈 D1 亮紅光。



其他功能說明：

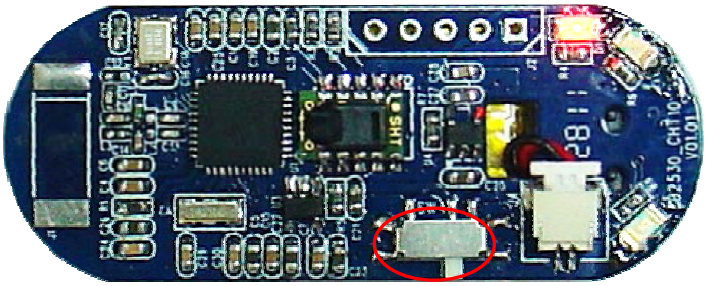
4. 如果背面天線信號不良時，可改爲正面天線。天線信號品質可由產品放置的位置及與地面的距離有關！



正面天線

背面天線

5. 電池開關 SW4 是控制電池的充放電。

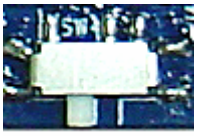


- 在 SW4 打開狀態下（指撥按鈕在右邊）：

a) 有外部電源插入時，這時可給電池充電同時給系統供電。

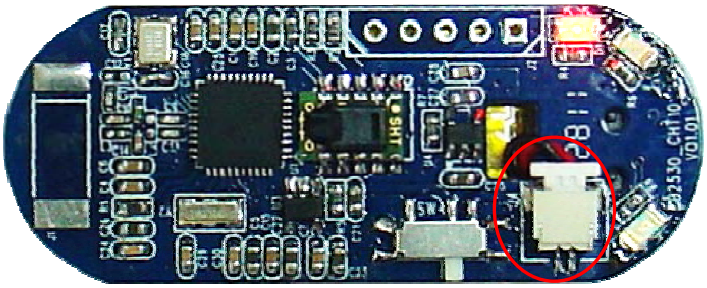
b) 如沒有外部電源，這時由電池給系統供電。

- 在 SW4 關閉狀態下（指撥按鈕在左邊）：

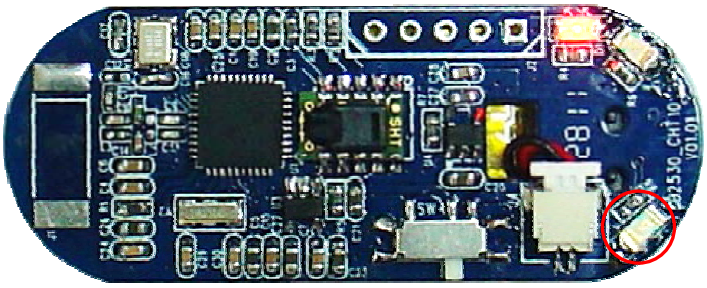


a) 接上外部電源，此時外部電源只給系統供電，不能充電。

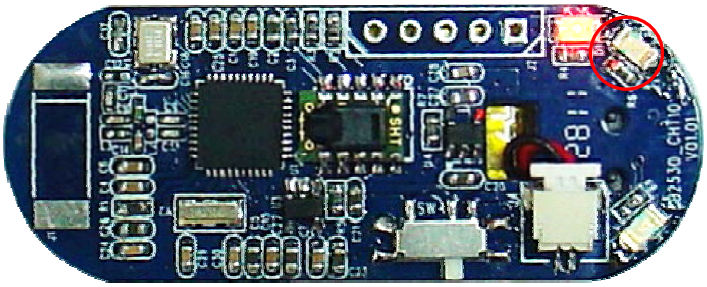
6. 2PIN 的電池介面 J4，插反時是不能插入！



7. 充電指示燈 D3。
   - 當電池開關 SW4 打開狀態下：
     a) 電池在充電時，D3 燈亮綠光。
     b) 當電池充滿時，D3 綠燈不亮。
   - 當電池開關 SW4 關閉狀態下，插入外部電源，D3 燈亮綠光。



8. D2 燈爲資料傳輸指示燈，在發送資料時，D2 會一直閃動綠光；不傳輸資料時，D2 燈不會亮。



9. 外部電源輸入埠 CON1，當外部電源插入 5V/1A 的電源時，可爲系統供電及電池充電。

2-4 配件說明

- 基本配件

1. ZB2530-SHT10 PCB 	2. 資料光碟 
3. Mini 5Pin USB 電源線 	4. 電池 

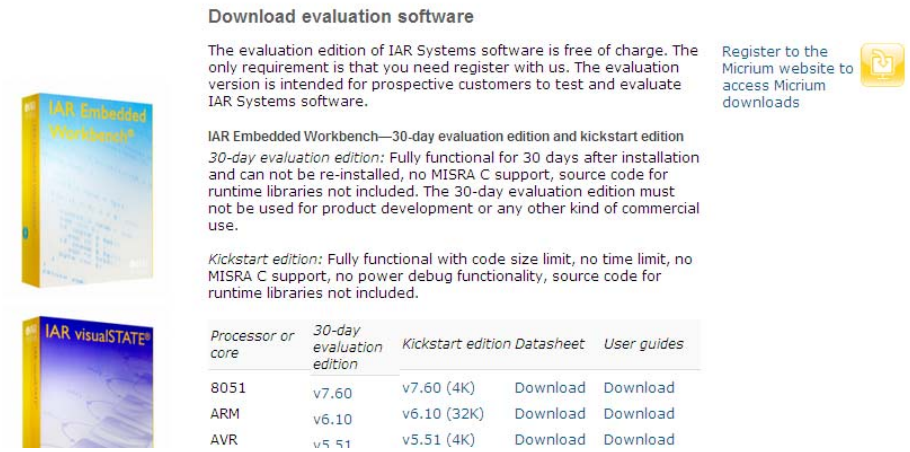
- 選購配件

1. JTAG 轉接頭 	2. ZigBee Debugger 模擬器 

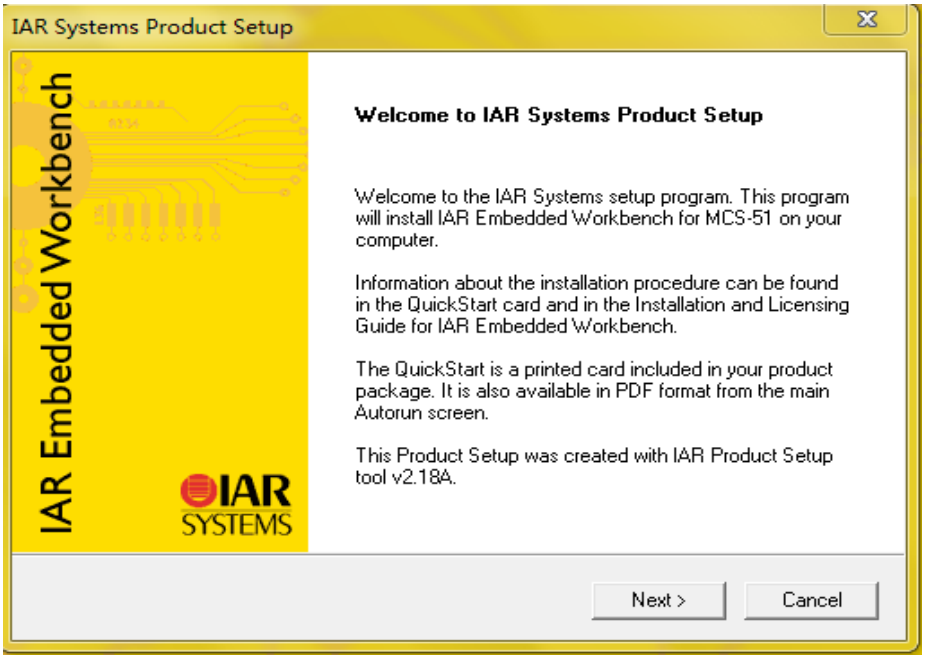
第三章 SHT10 溫濕度感測器 基於 IAR 環境架設和開發流程

3-1 IAR 軟體的安裝

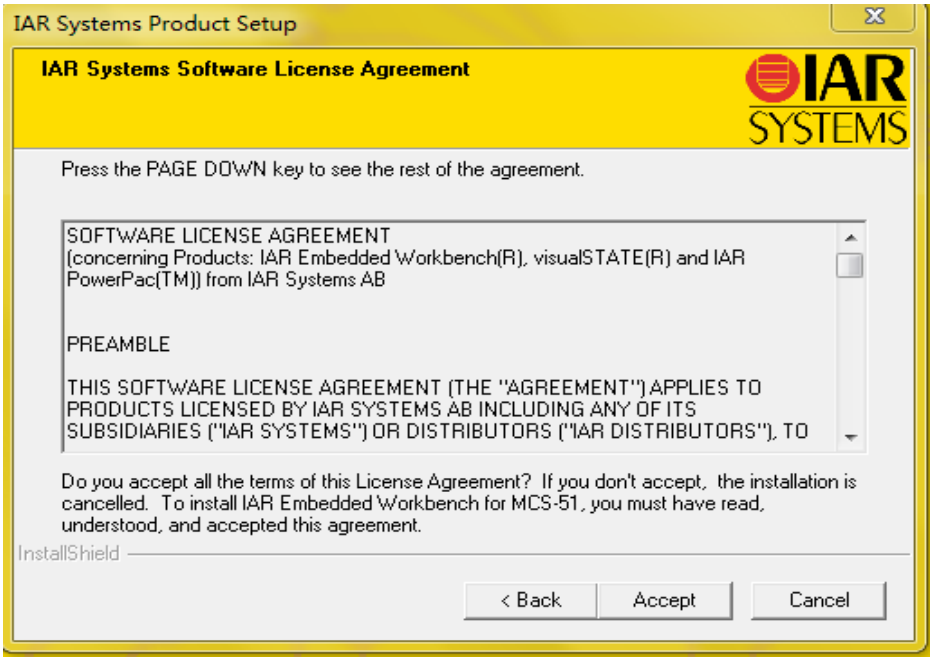
可至IAR官網，購買正版軟體或到至首頁下的Downloads下載30天試用版，記住要下載對應 C8051 的 IAR 軟體），如下圖所示：



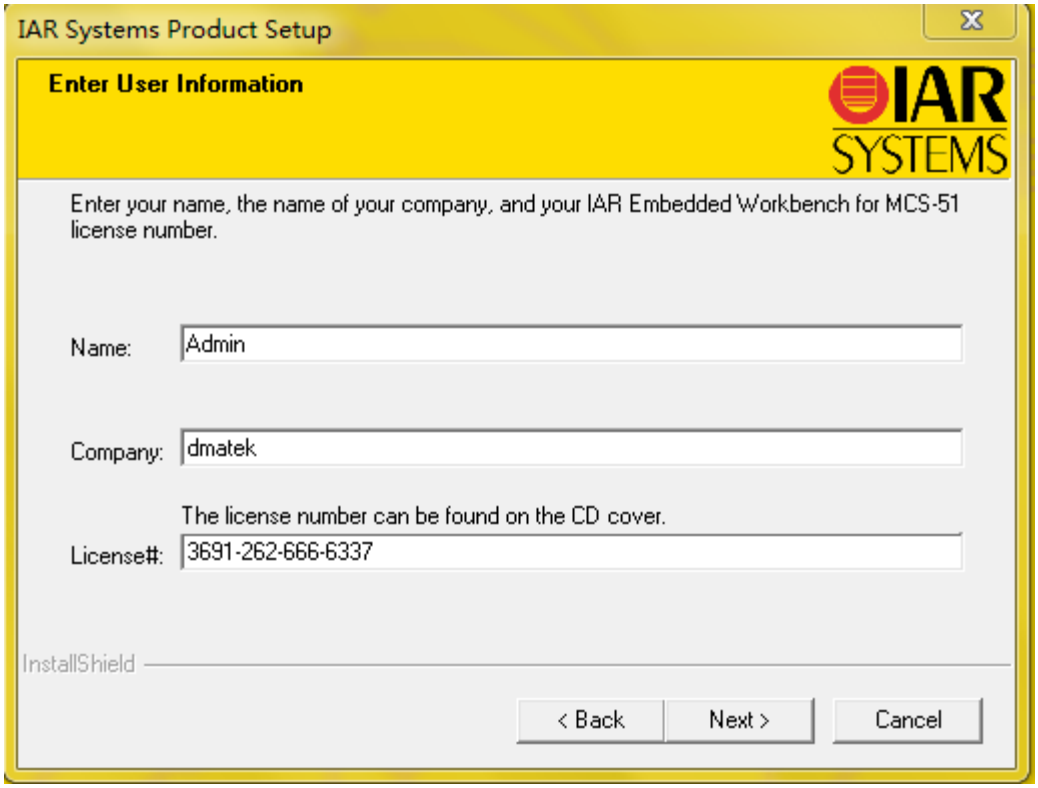
下面是以IAR 8051 7.51A 安裝爲例：



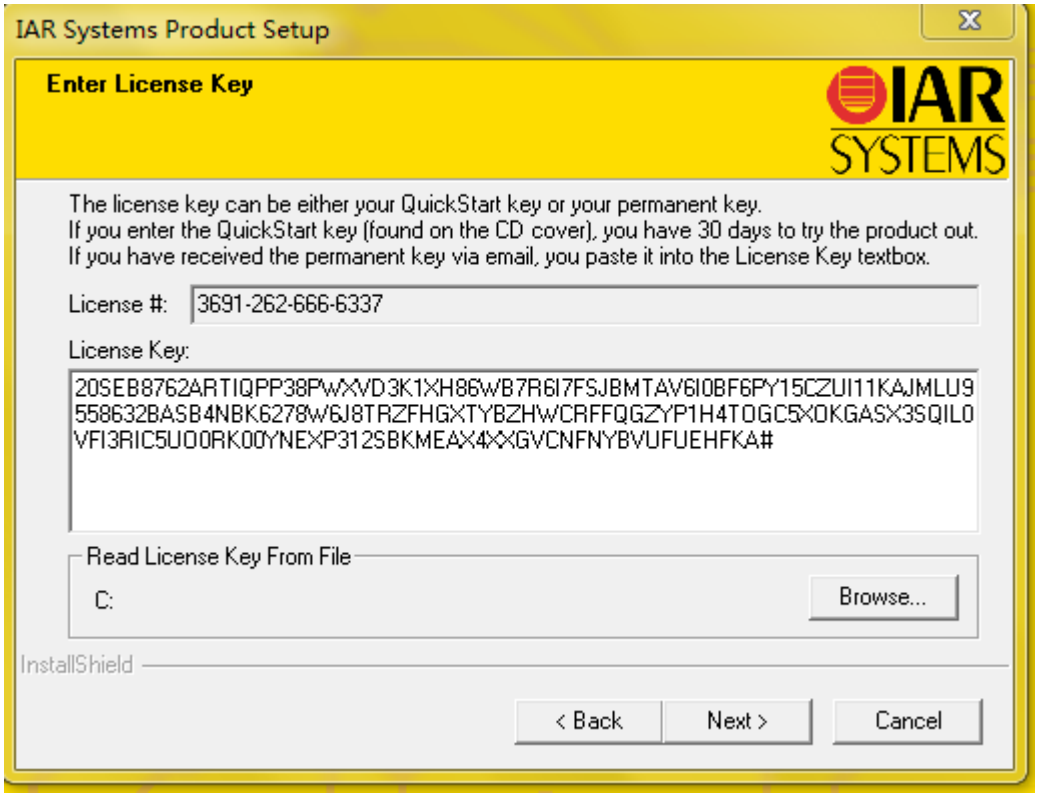
點選 Next，進入如下圖所示：



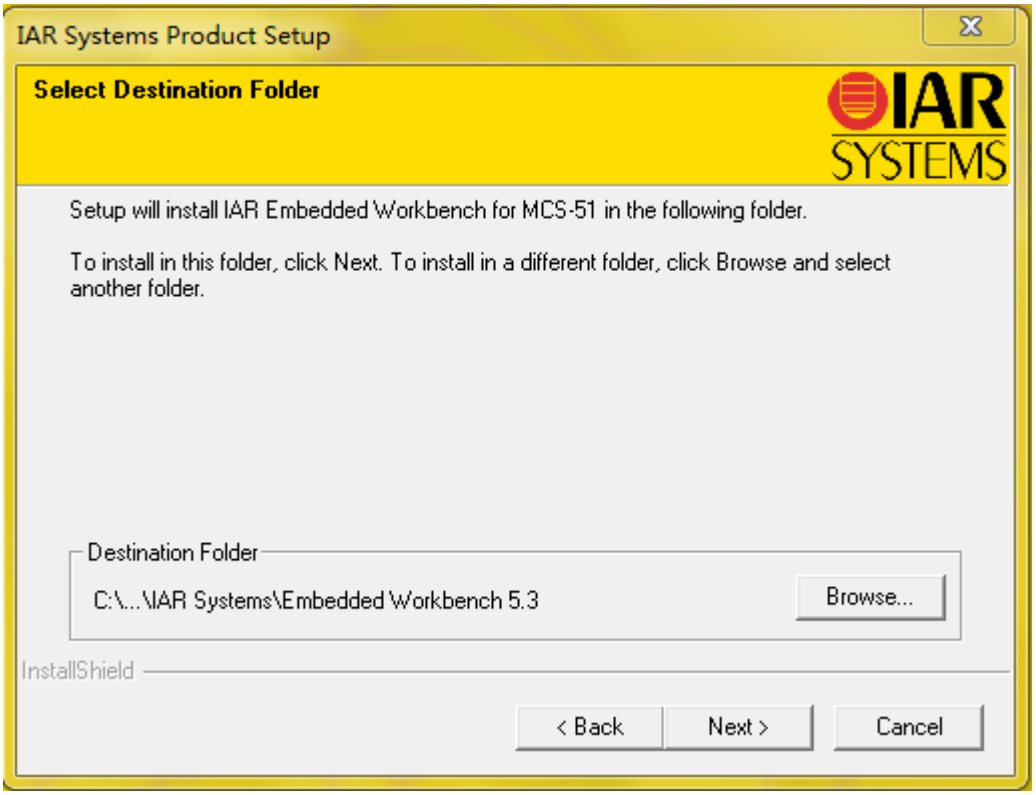
點選 Accept，進入如下圖所示：



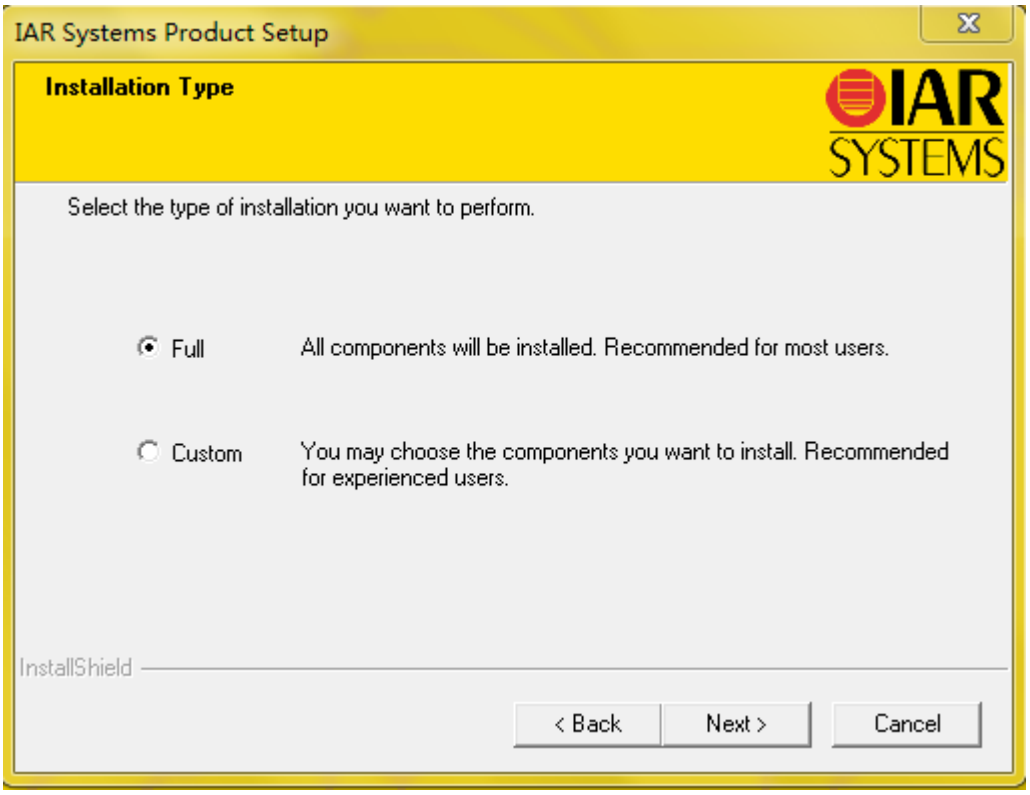
輸入相關資訊後點選 Next



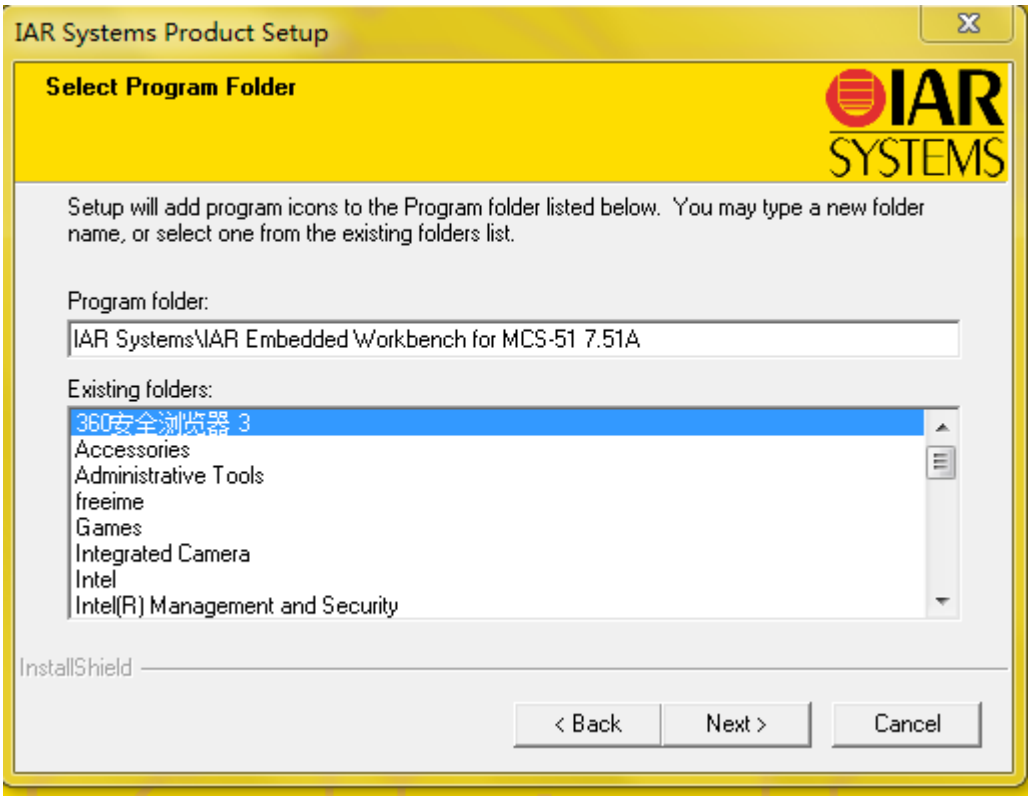
點選 Next，如下圖所示：



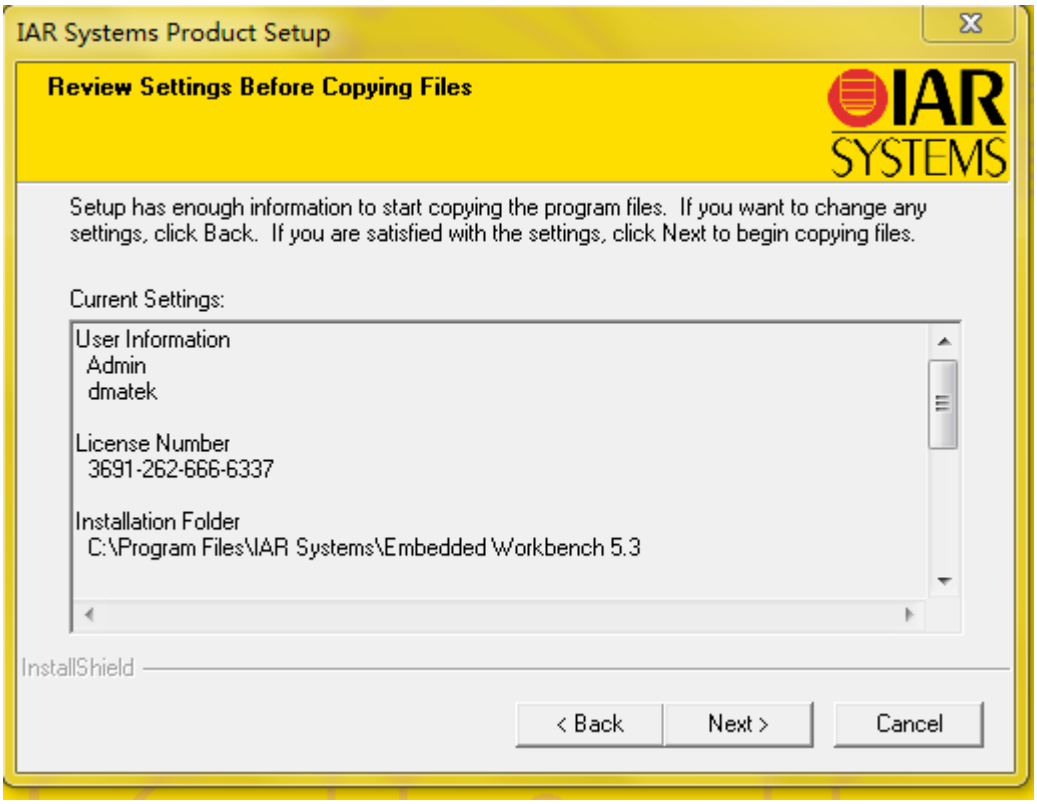
點選 Next，如下圖所示：



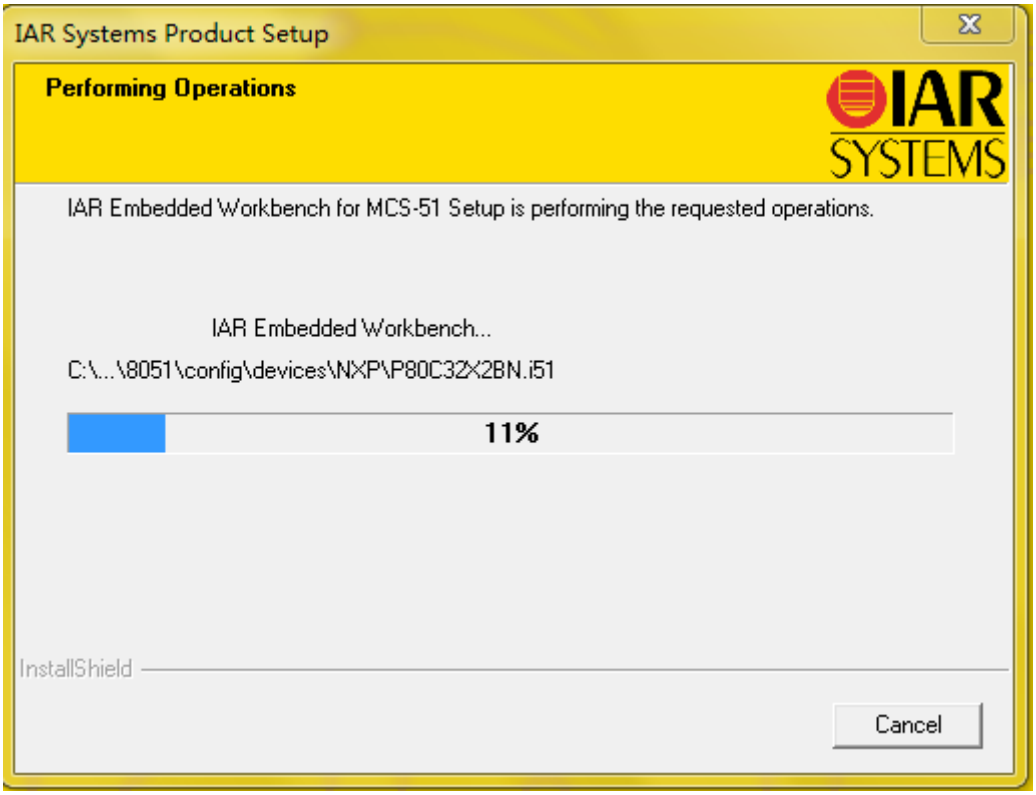
點選 Next，如下圖所示：



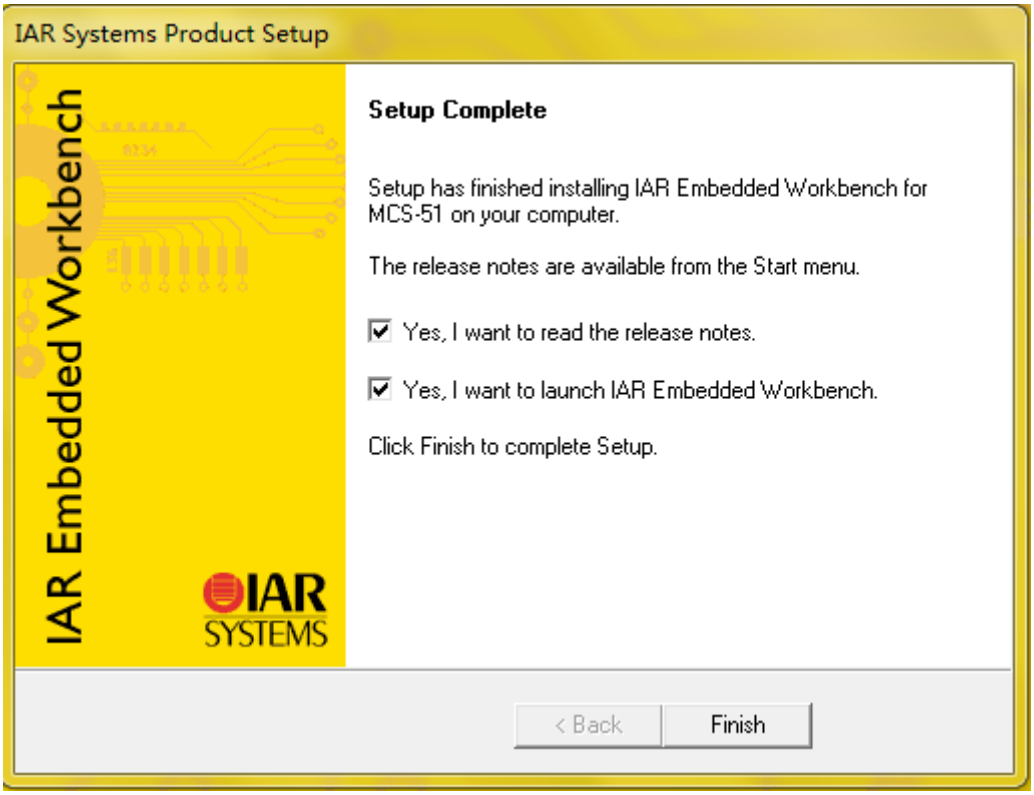
點選 Next，如下圖所示：



點選 Next，如下圖所示：


IAR Systems Product Setup
Performing Operations
IAR SYSTEMS
IAR Embedded Workbench for MCS-51 Setup is performing the requested operations.
IAR Embedded Workbench...
C:\...\8051\config\devices\NXP\P80C32X2BN.i51
11%
InstallShield
Cancel

安裝完成後，如下圖所示：

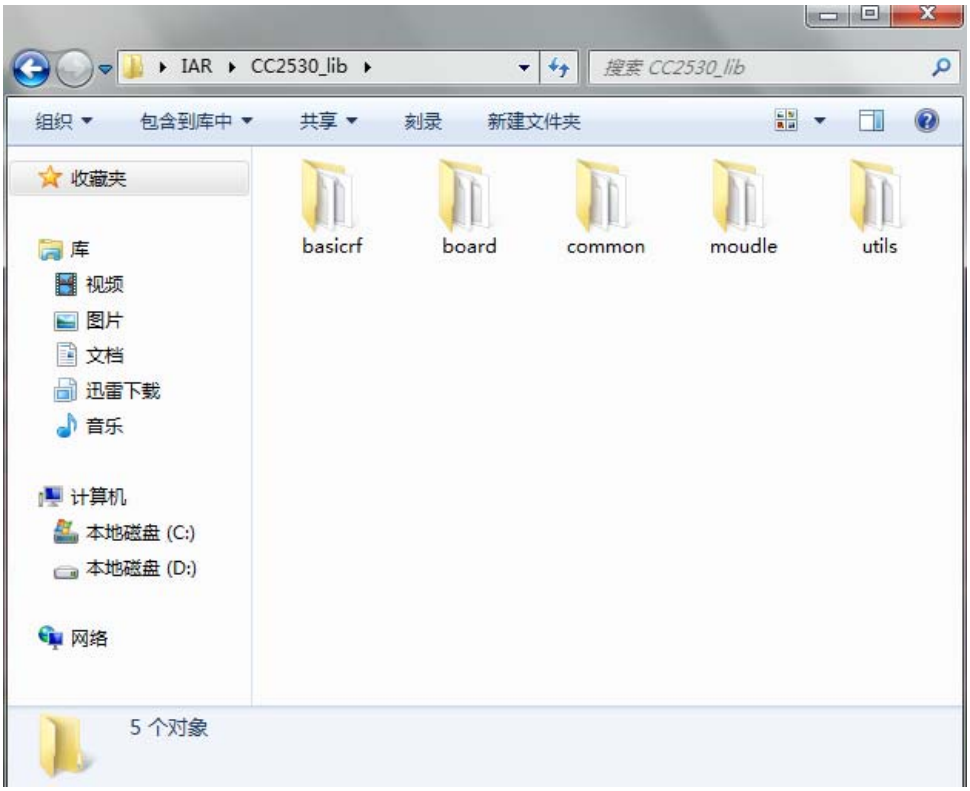


點選 Finish，就完成了 IAR 軟體的安裝。下面將以光敏電阻感測器爲實例介紹如何使用 IAR 軟體來編譯原始程式碼生成 HEX 檔，並透過下載器將 HEX 檔下載到 CC2530 晶片的 RAM 中，這樣就可以執行相應的動作了。

3-2 在 IAR 環境下開發溫濕度感測器 SHT10 的應用程式

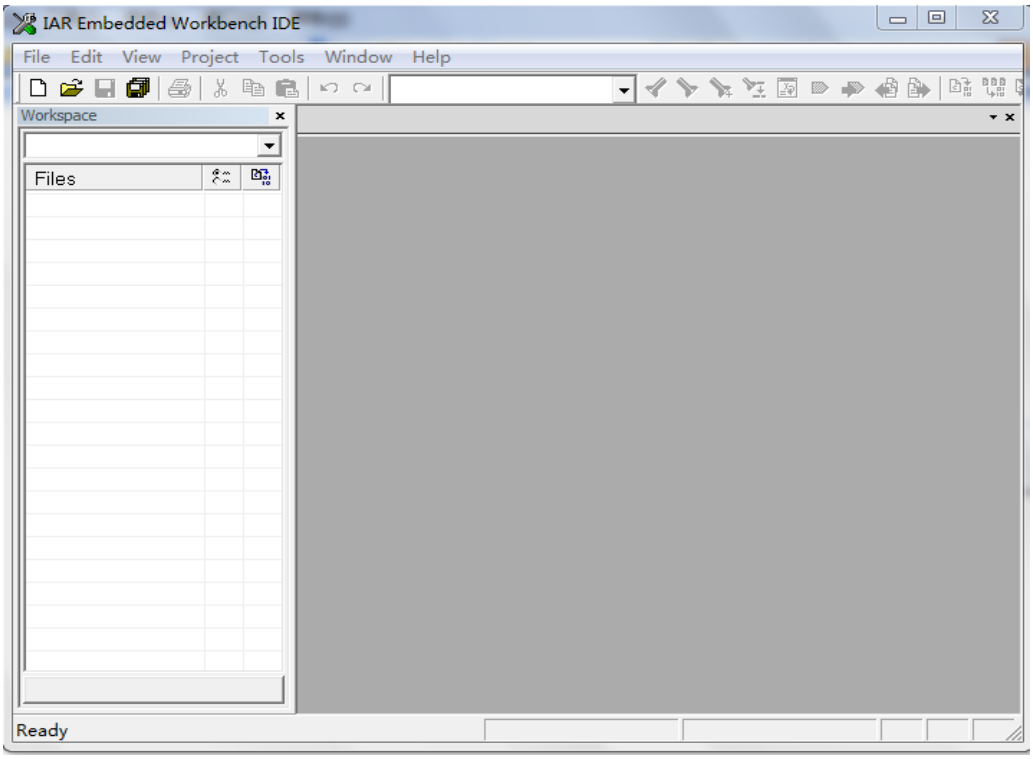
1.固件庫的使用

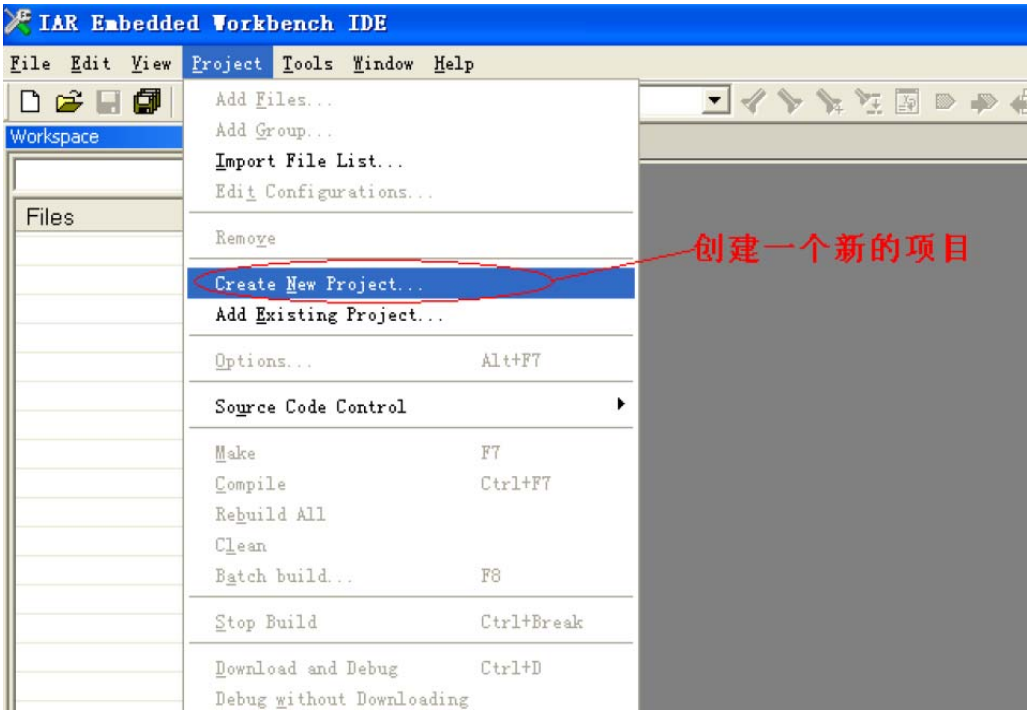
在利用 IAR 環境開發應用之前，先將關於 CC2530 單晶片的 ST 固件庫（也可以進入 ST 官網自己下載）複製到自己創建的工程目錄下，我們提供的固件庫在光碟根目錄下 IAR/ CC2530_lib 的檔案夾中，如下圖所示：

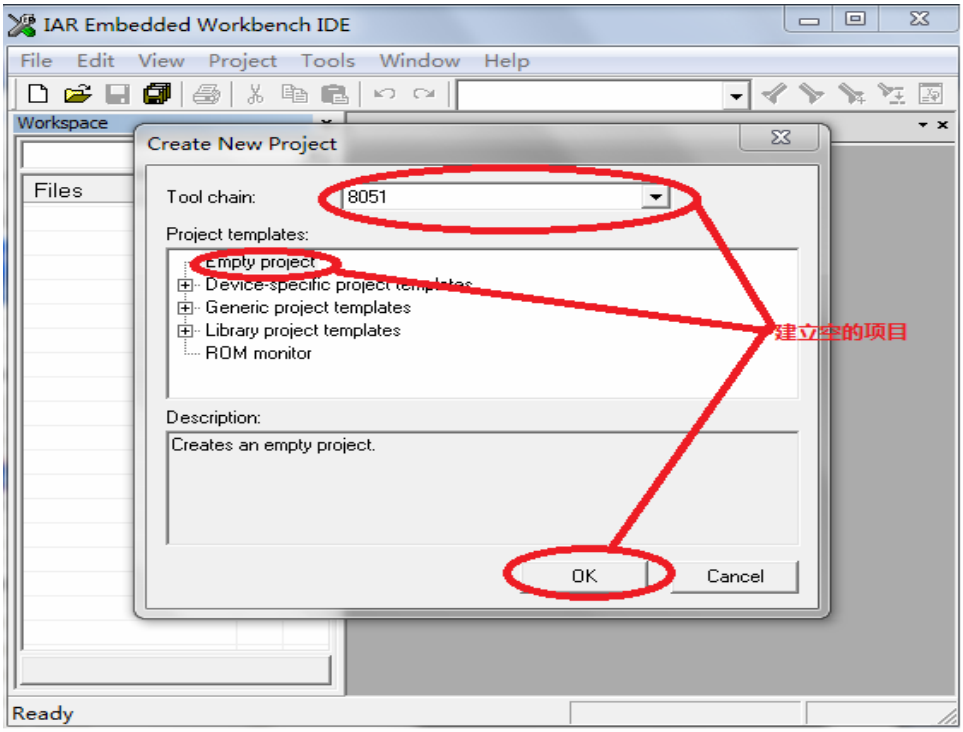


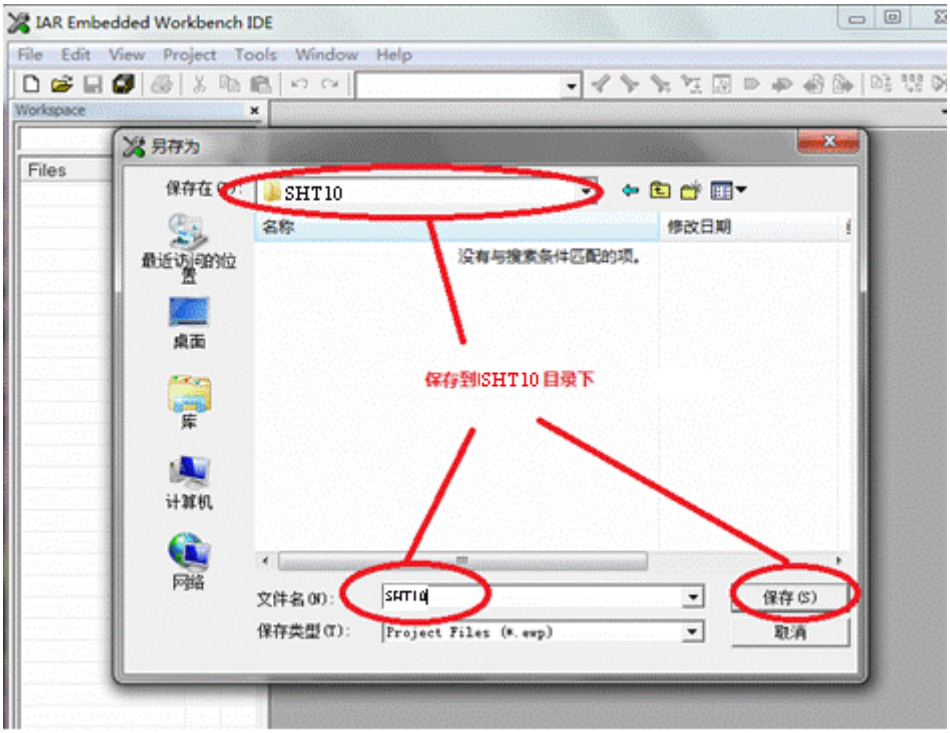
2.利用 IAR 環境建立 SHT10 溫濕度感測器專案工程

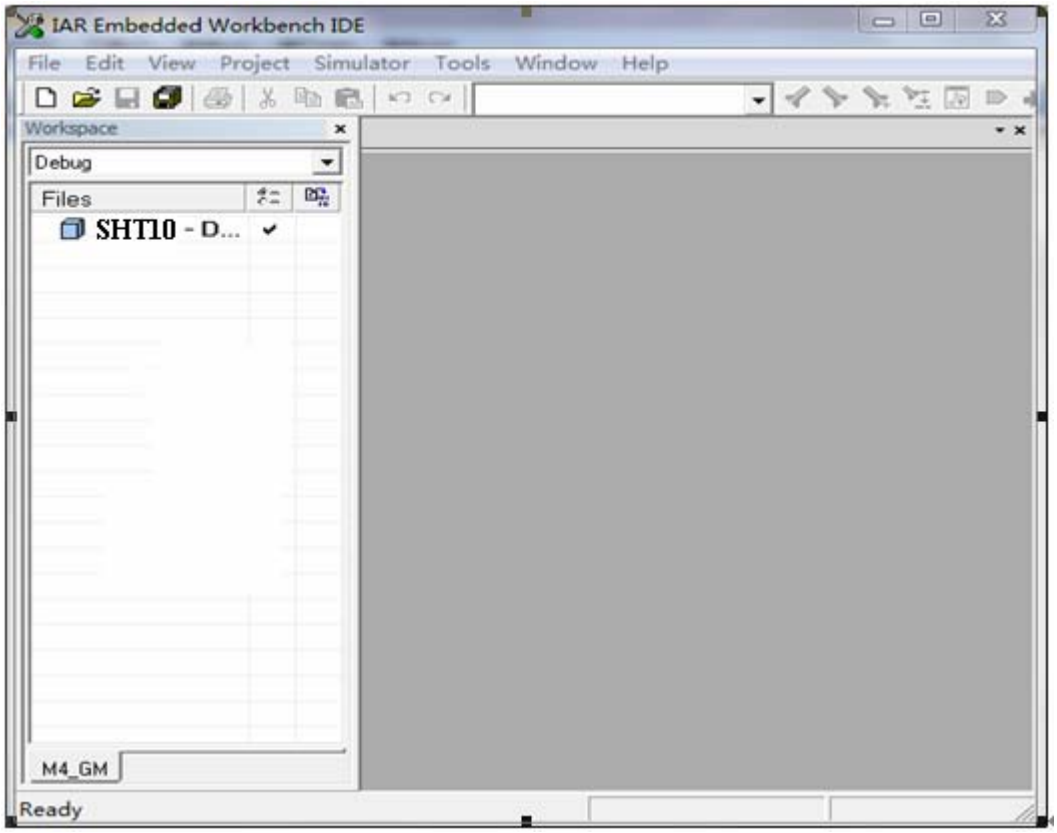
打開安裝好的 IAR IDE，如下圖所示：


IAR Embedded Workbench IDE
File Edit View Project Tools Window Help
Workspace
Files
Ready

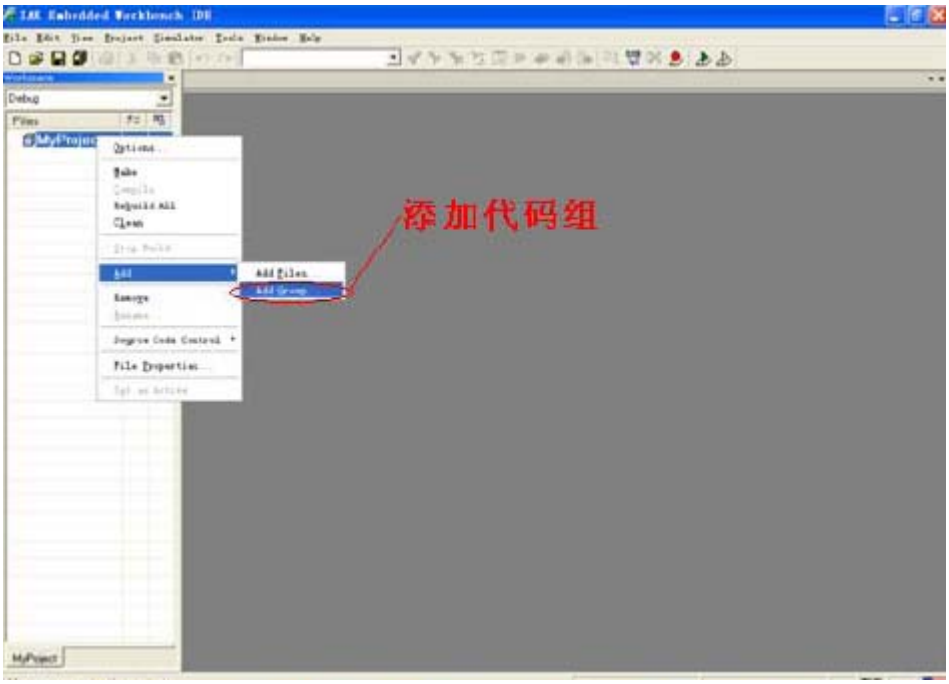

IAR Embedded Workbench IDE
File Edit View Project Tools Window Help
Workspace
Files
Add Files...
Add Group...
Import File List...
Edit Configurations...
Remove
Create New Project...
Add Existing Project...
Options... Alt+F7
Source Code Control
Make F7
Compile Ctrl+F7
Rebuild All
Clean
Batch build... F8
Stop Build Ctrl+Break
Download and Debug Ctrl+D
Debug without Downloading
创建一个新的项目


IAR Embedded Workbench IDE
File Edit View Project Tools Window Help
Workspace
Files
Create New Project
Tool chain:
8051
Project templates:
Empty project
Device-specific project templates
Generic project templates
Library project templates
ROM monitor
建立空的項目
Description:
Creates an empty project.
OK
Cancel
Ready

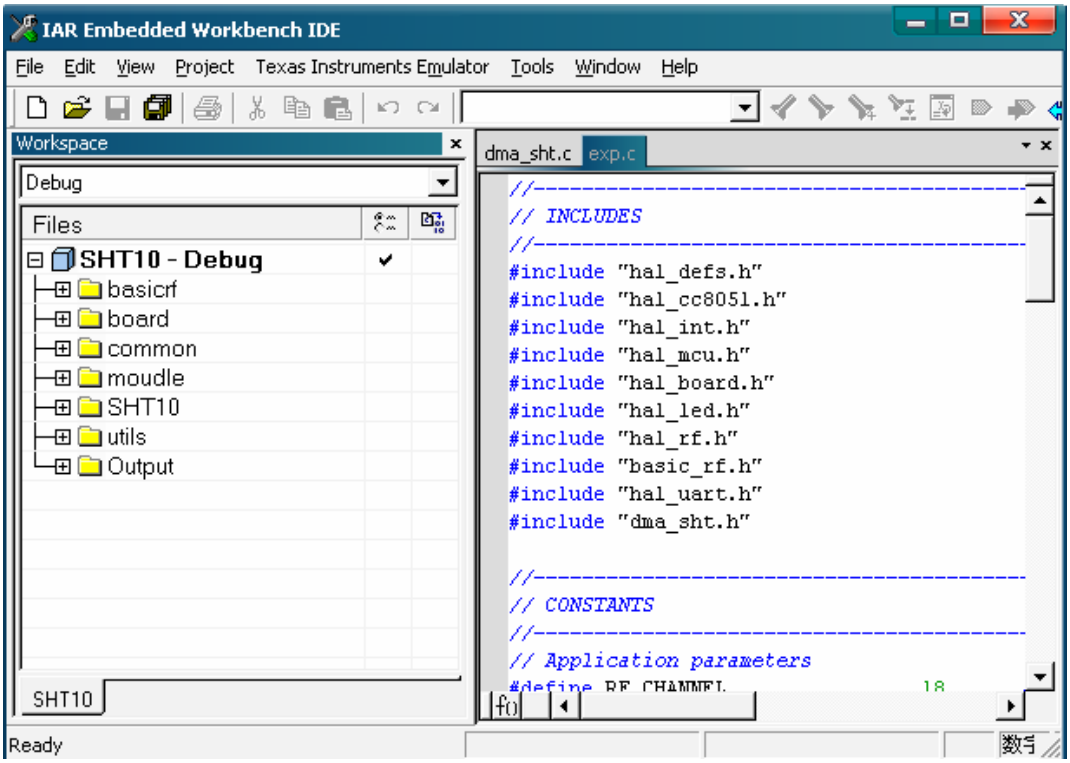

IAR Embedded Workbench IDE
File Edit View Project Tools Window Help
Workspace
Files
另存为
保存在(I):
SHT10
名称
修改日期
没有与搜索条件匹配的项。
最近访问的位置
桌面
库
计算机
网络
保存到SHT10目录下
文件名(N):
SHT10
保存(S)
保存类型(T):
Project Files (*.ewp)
取消



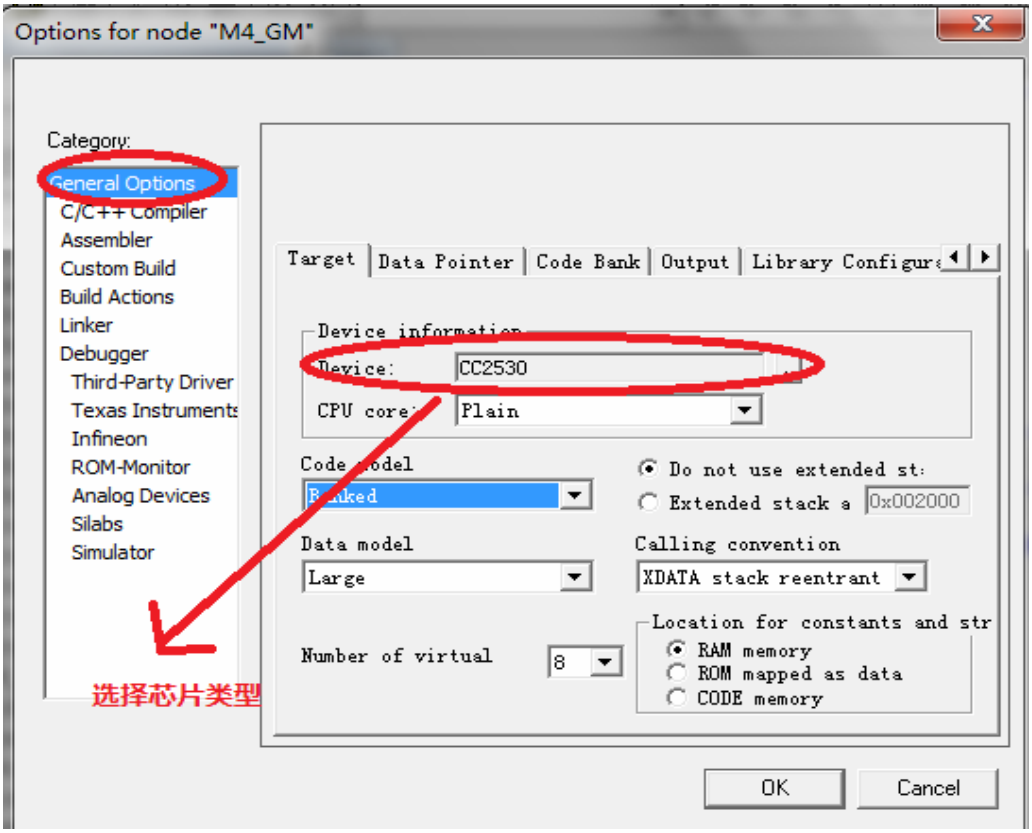
用戶可以向項目中添加*.c 文件；也可以添加程式碼組；對程式碼進行分組可以很好的進行原始程式碼的管理，也有助於生成較好的目標程式碼，如下圖所示：

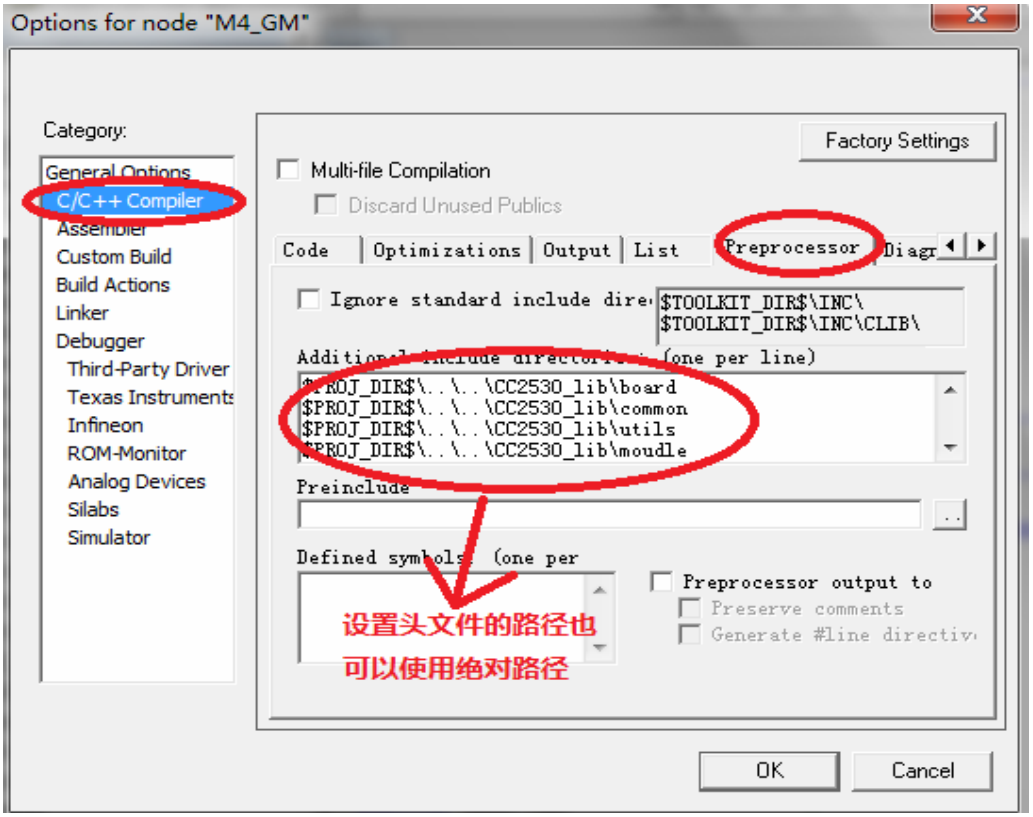


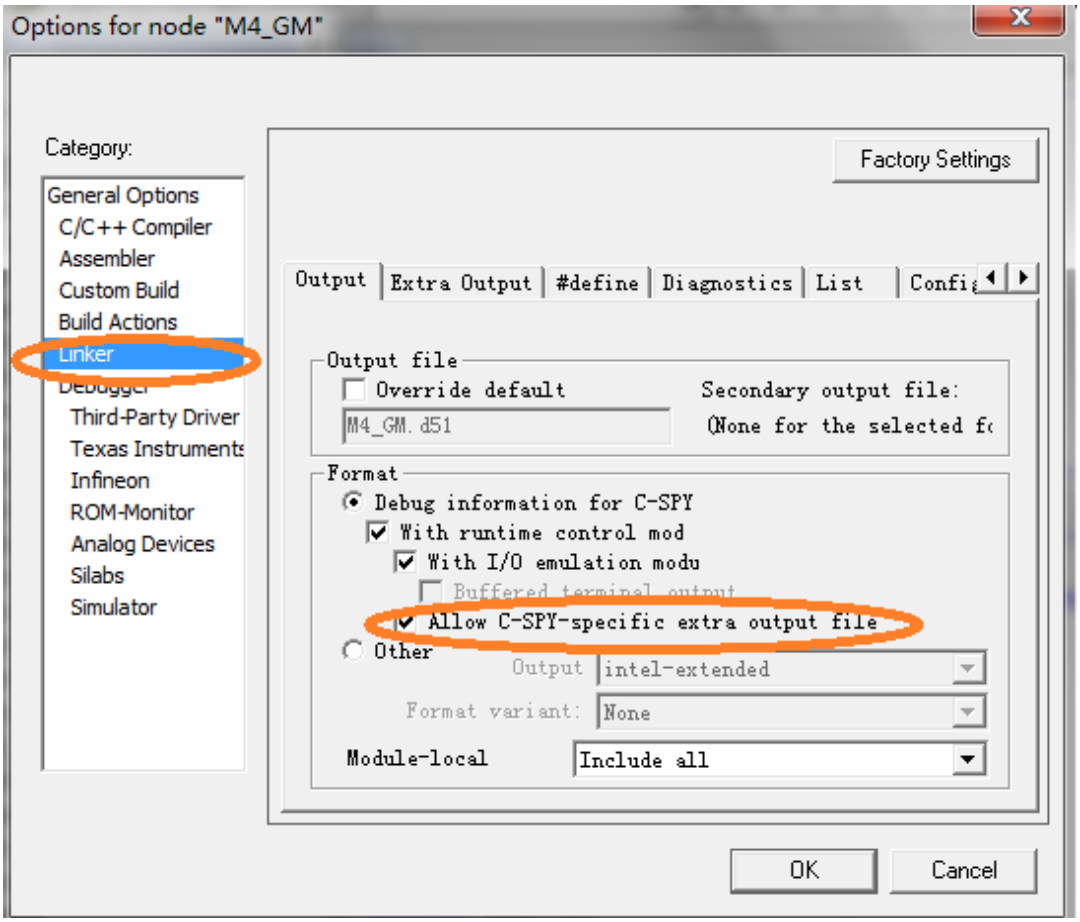
接著在對應程式碼組中添加我們已經準備好的庫檔和自己編寫的程式碼，SHT10 添加的是 exp.c 文件，moudle 下添加的是 dma_sht.c 文件，如下圖所示：

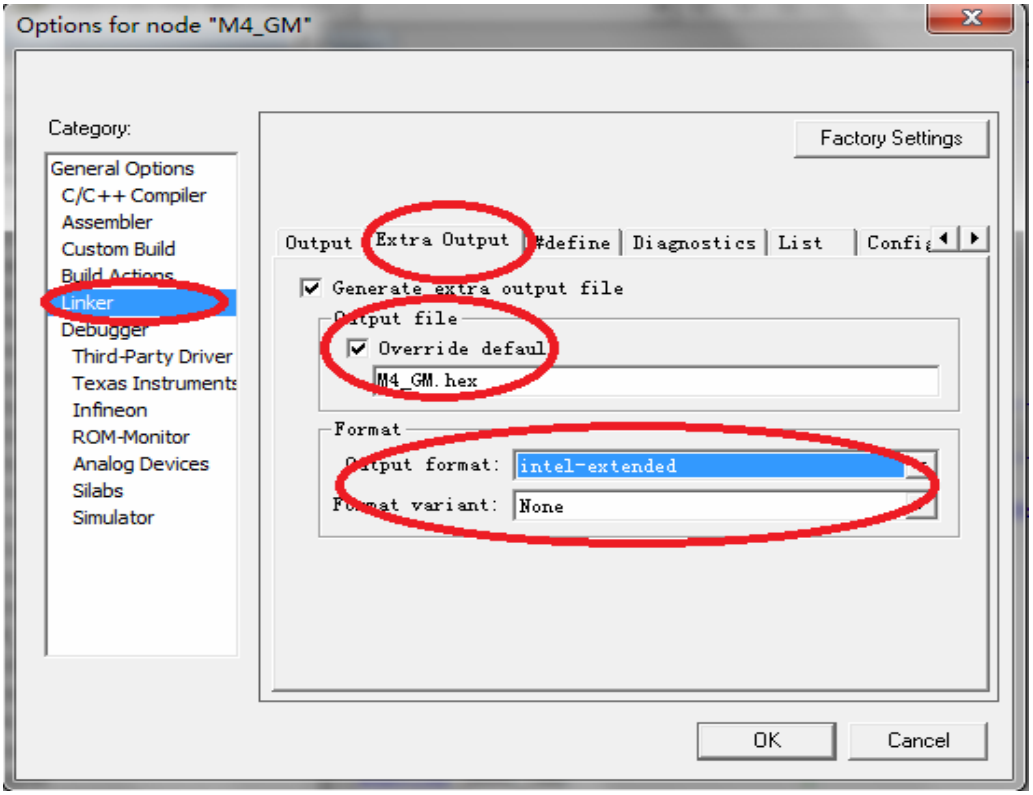


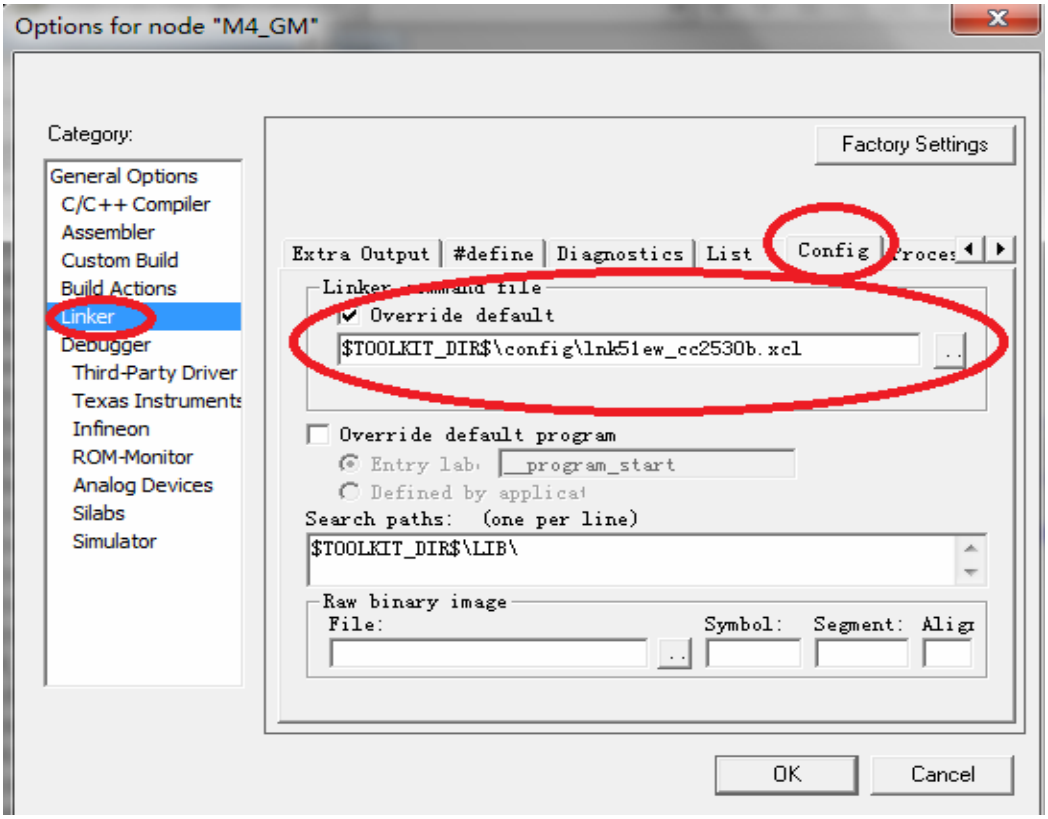
在上圖紅色框地方滑鼠右鍵，選擇 Option，如下圖所示：

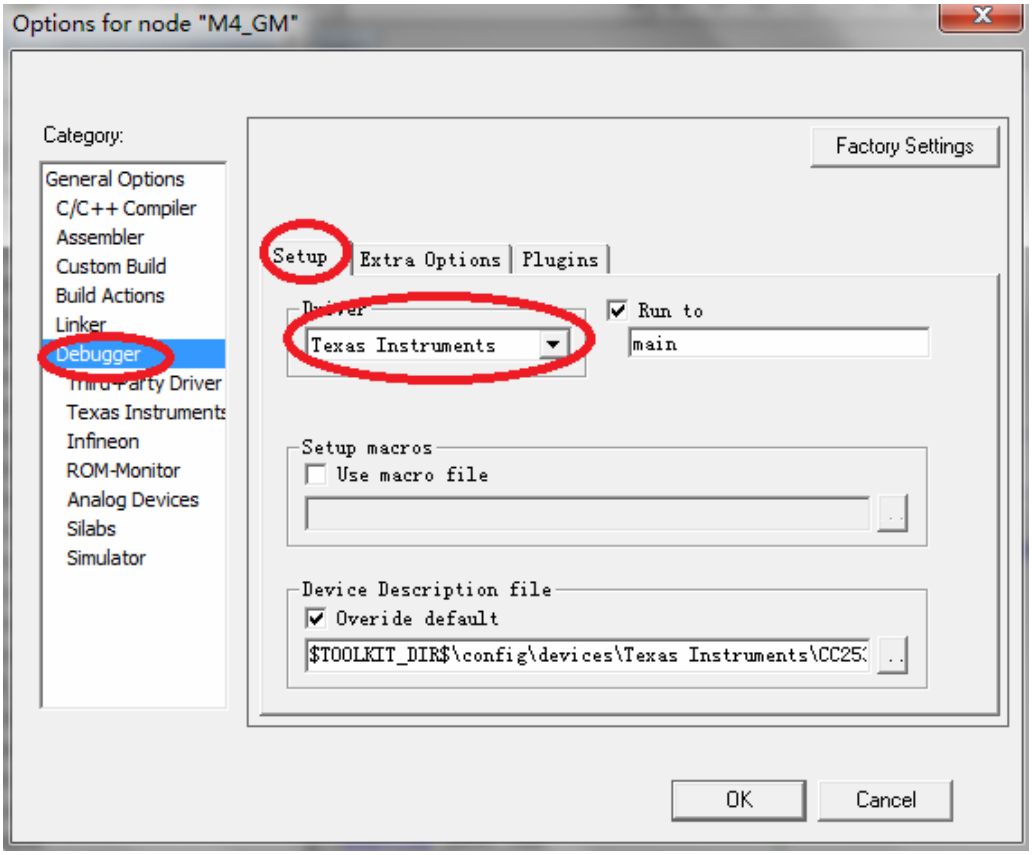

Options for node "M4_GM"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Debugger
Third-Party Driver
Texas Instruments
Infineon
ROM-Monitor
Analog Devices
Silabs
Simulator
Target
Data Pointer
Code Bank
Output
Library Configur
Device information
Device:
CC2530
CPU core:
Plain
Code model
Banked
Do not use extended st:
Extended stack a
0x002000
Data model
Large
Calling convention
XDATA stack reentrant
Number of virtual
8
Location for constants and str
RAM memory
ROM mapped as data
CODE memory
OK
Cancel
选择芯片类型


Options for node "M4_GM"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Debugger
Third-Party Driver
Texas Instruments
Infineon
ROM-Monitor
Analog Devices
Silabs
Simulator
Factory Settings
Multi-file Compilation
Discard Unused Publics
Code
Optimizations
Output
List
Preprocessor
Ignore standard include dire
$TOOLKIT_DIR$\INC\
$TOOLKIT_DIR$\INC\CLIB\
(one per line)
$PROJ_DIR$\..\..\CC2530_lib\board
$PROJ_DIR$\..\..\CC2530_lib\common
$PROJ_DIR$\..\..\CC2530_lib\utils
$PROJ_DIR$\..\..\CC2530_lib\moudle
Preinclude
Defined symbols (one per
Preprocessor output to
Preserve comments
Generate #line directiv
设置头文件的路径也
可以使用绝对路径
OK
Cancel

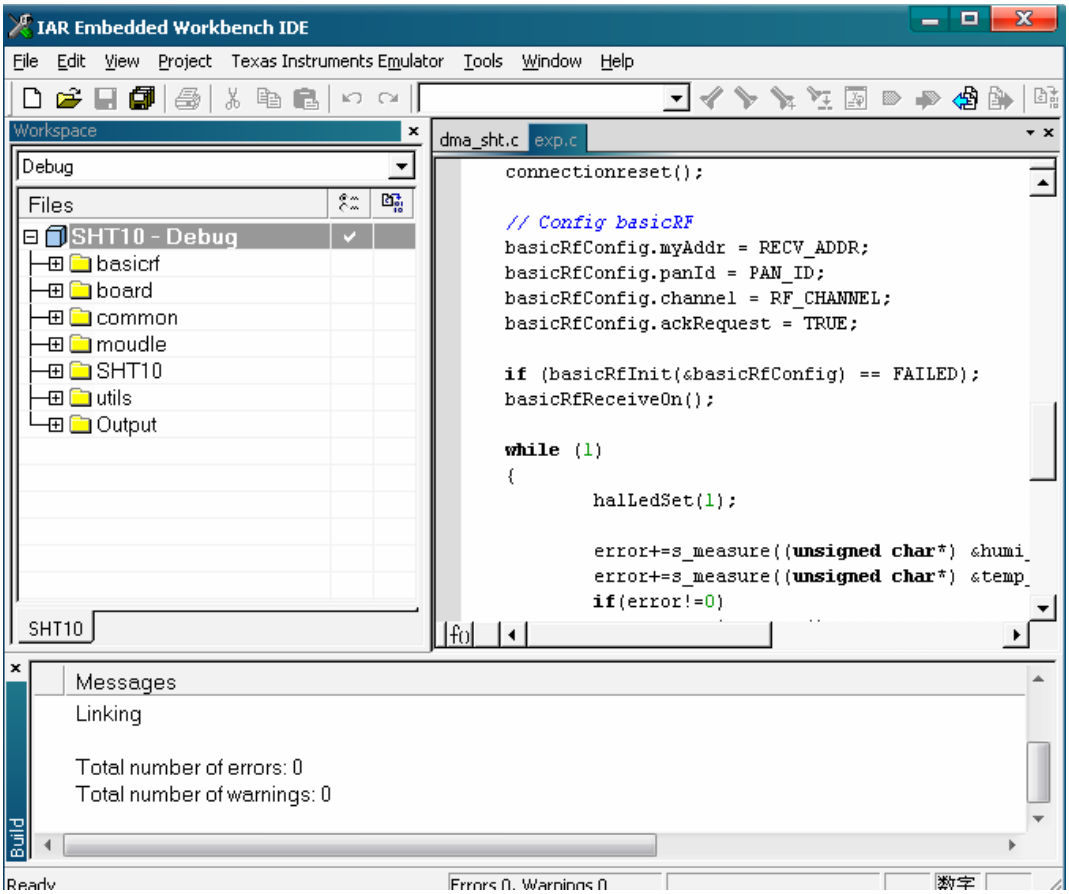

Options for node "M4_GM"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Third-Party Driver
Texas Instruments
Infineon
ROM-Monitor
Analog Devices
Silabs
Simulator
Factory Settings
Output
Extra Output
#define
Diagnostics
List
Output file
Override default
M4_GM.d51
Secondary output file:
(None for the selected f
Format
Debug information for C-SPY
With runtime control mod
With I/O emulation modu
Allow C-SPY-specific extra output file
Other
Output
intel-extended
Format variant:
None
Module-local
Include all
OK
Cancel


Options for node "M4_GM"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Debugger
Third-Party Driver
Texas Instruments
Infineon
ROM-Monitor
Analog Devices
Silabs
Simulator
Factory Settings
Output
Extra Output
#define
Diagnostics
List
Generate extra output file
Override defaul
M4_GM.hex
Format
intel-extended
None
OK
Cancel


Options for node "M4_GM"
Category:
General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Debugger
Third-Party Driver
Texas Instruments
Infineon
ROM-Monitor
Analog Devices
Silabs
Simulator
Factory Settings
Extra Output
#define
Diagnostics
List
Config
Override default
$TOOLKIT_DIR$\config\lnk51ew_cc2530b.xcl
Override default program
__program_start
Search paths: (one per line)
$TOOLKIT_DIR$\LIB\
Raw binary image
File:
Symbol:
Segment:
OK
Cancel



配置完成後，點選 OK，然後編譯，成功如下圖所示：



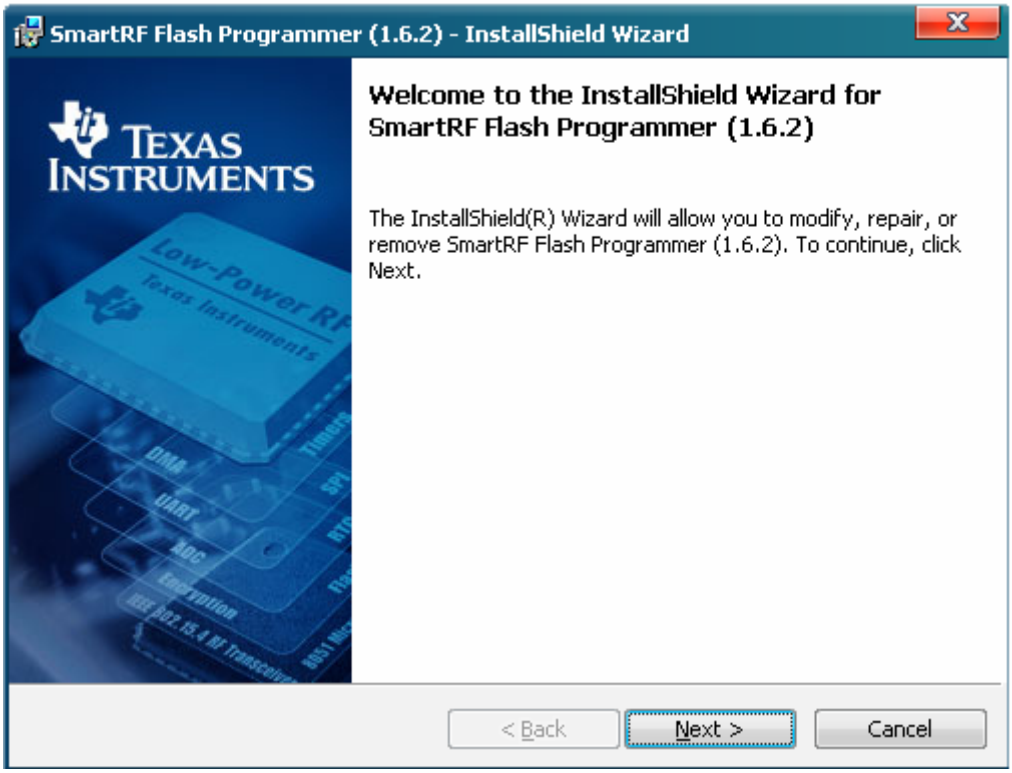
最後生成的 SHT10.hex，在 SHT10\Debug\Exe 下，如下圖所示：

最後將 hex 檔透過模擬器燒錄進晶片內。

3-3 HEX 檔案燒錄過程

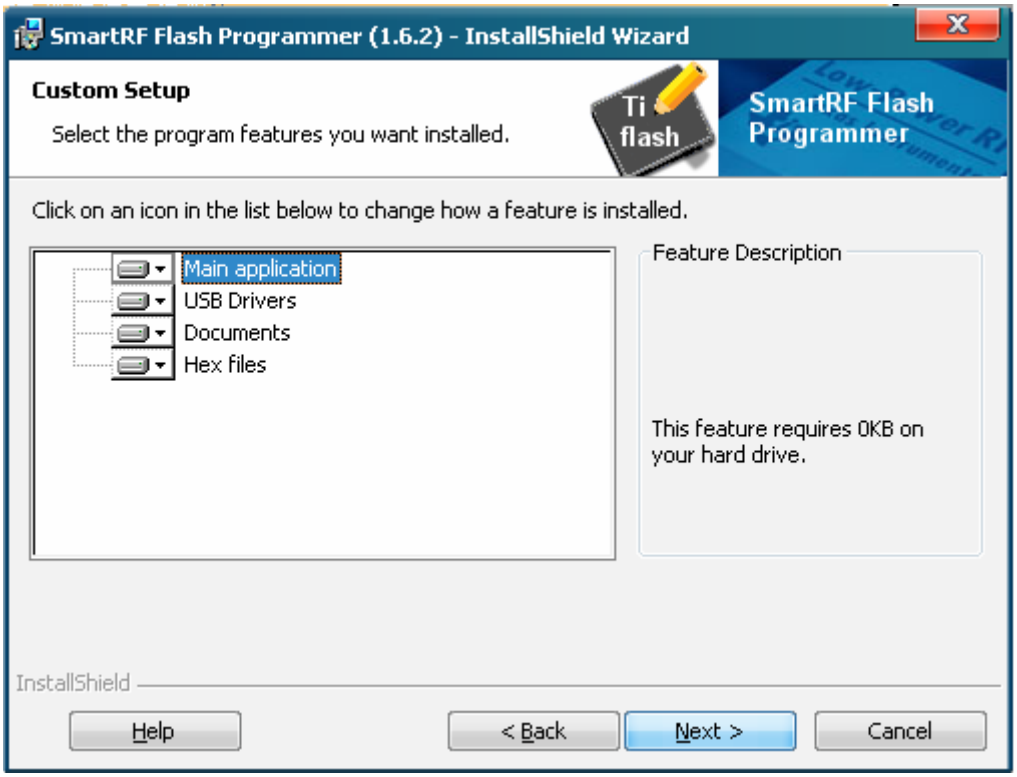
在前面一章已經介紹了燒錄過程的硬體連接步驟，這裏主要介紹燒錄時所需軟體的使用。

首先，安裝燒錄軟體 Setup_SmartRFProgr_1.6.2.exe：

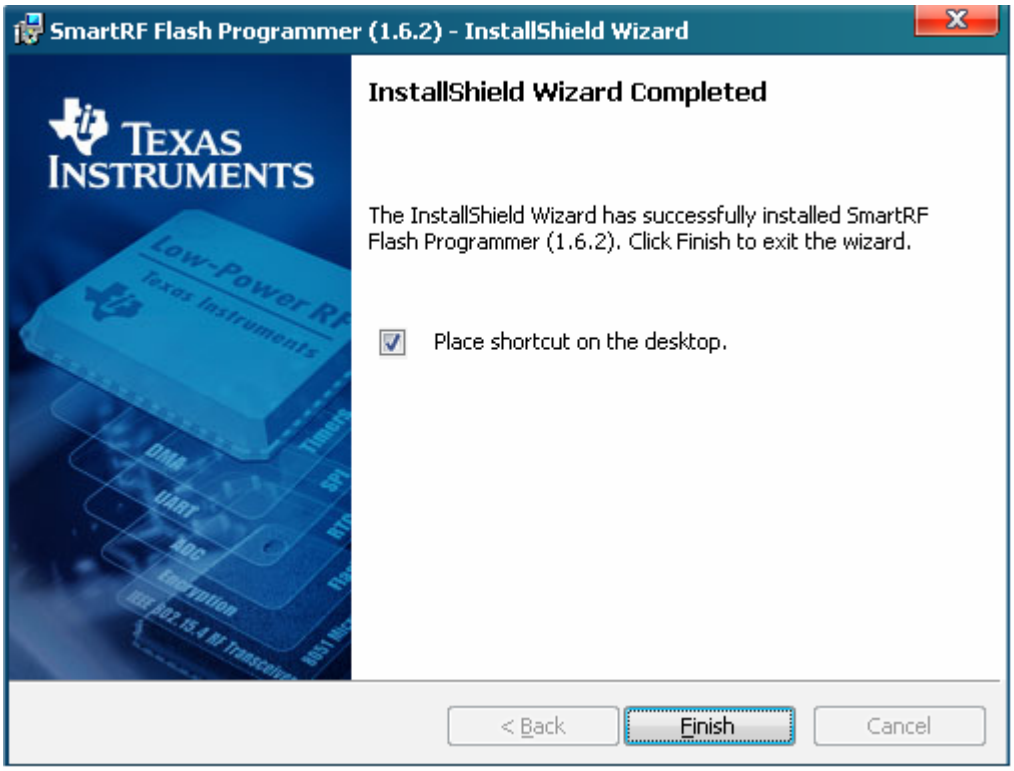


點擊 Next，出現如下圖所示：

點擊 Next，出現如下圖所示：

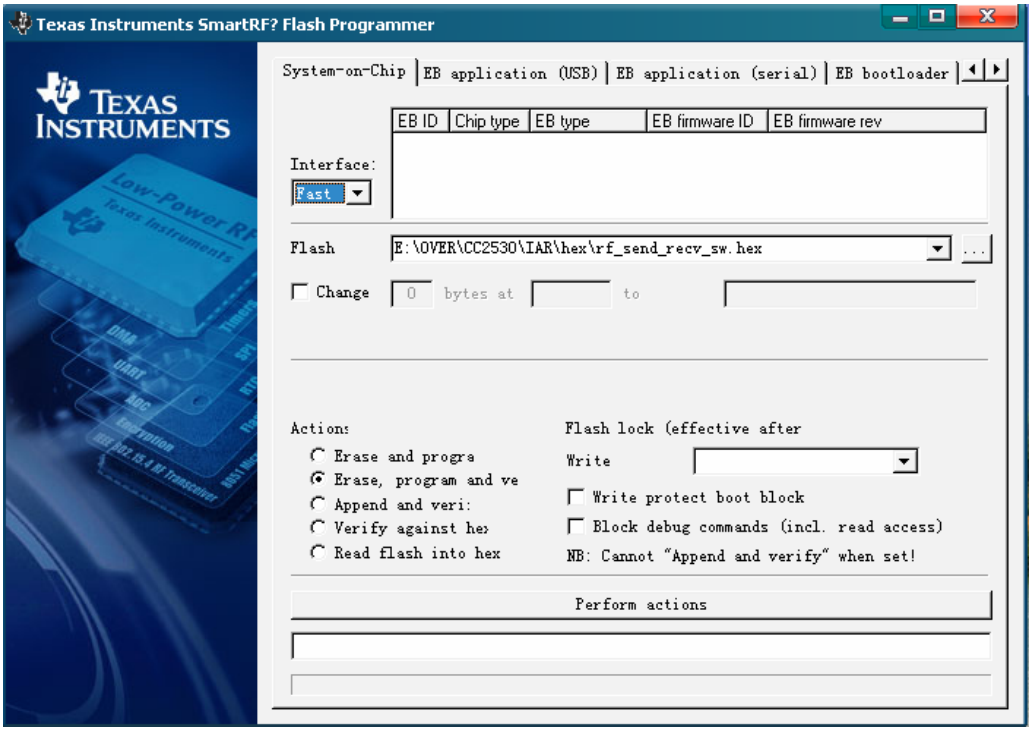


點擊 Next，出現如下圖所示：

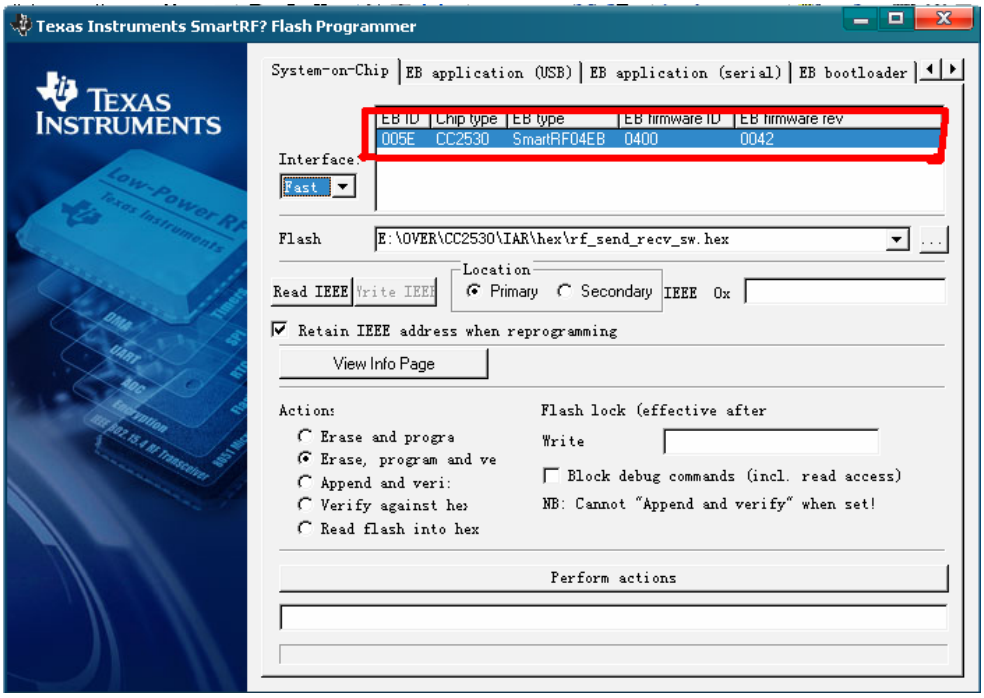


勾選“Place shortcut on the desktop”，點擊 Finish 完成安裝後，將會在電腦桌面上出現如下圖示：

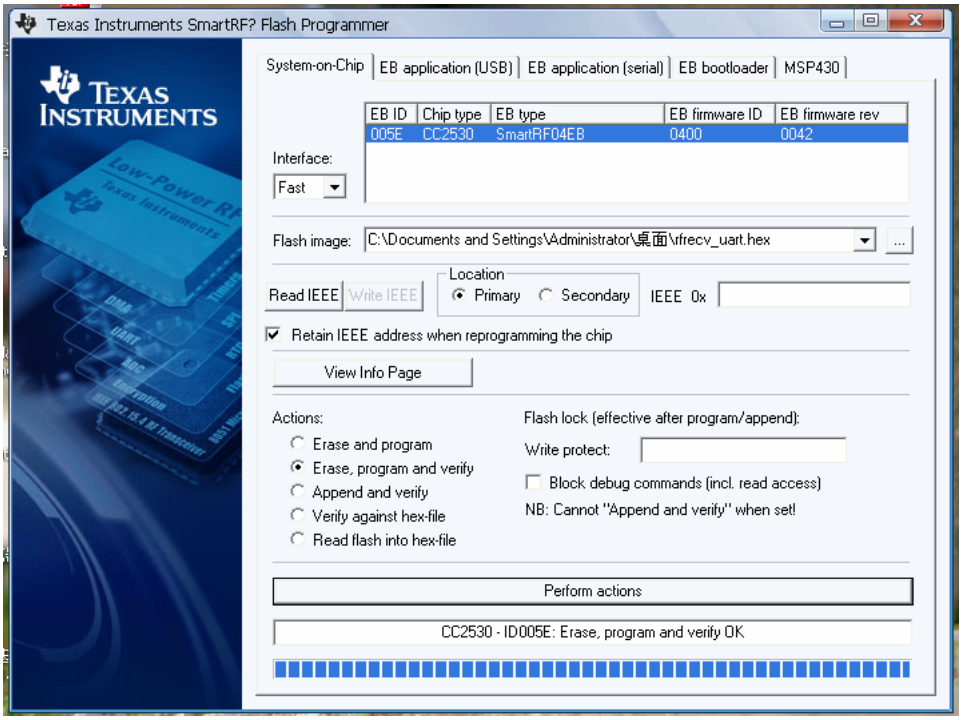
然後，點兩下此圖示打開軟體，如下圖所示：



接著，按下模擬器上的重定按鍵，將會檢測到晶片型號，這樣說明已經識別到所要燒錄的晶片，如下圖所示：



最後點擊 Perform action ，出現如下圖所示，說明燒錄成功。



3-4 SHT10 溫濕度感測器原始程式碼分析

透過瞭解 SHT10 溫濕度感測器模組的原理和模組的原理圖（電路圖見附錄）後，我們知道模組採集到的資料是透過感測器內部的非標準的 IIC 協議來傳輸的，具體程式如下：

SHT10 非標準的 IIC 協議程式碼如下所示：

```
//--------- write a byte and checks the ack signal----------------
char write_byte(uchar value)
{
  uchar i,error=0;
  for (i=0x80;i>0;i/=2)                    /*連續向右移動 8 位*/
  {
    if (i & value)
      HAL_SHT_SDA_SET();                    /*把相應的位元送資料線*/
    else
      HAL_SHT_SDA_CLR();

    HAL_SHT_SCK_SET();                      /*時序脈衝，應嚴格按著此標準*/
    halMcuWaitUs(1);                        /*大約 1 us*/
    HAL_SHT_SCK_CLR();
  }
  HAL_SHT_SDA_SET();                        /*釋放資料線*/
  HAL_SHT_SDA_DIR_IN();
  HAL_SHT_SCK_SET();                        /*第 9 位作爲回應位*/
  error=HAL_SHT_SDA_VAL();                  /*檢測回應情況，如果有回應資料線就
                                             會被 SHT10 拉低*/
  HAL_SHT_SCK_CLR();

  return error;                             /*返回 1 表示沒回應*/
```

```
}

//--------- read a byte and checks the ack signal----------------
char read_byte(uchar ack)
{
   uchar i,val=0;

   HAL_SHT_SDA_SET();                        /*釋放資料線*/
   HAL_SHT_SDA_DIR_IN();
   for (i=0x80;i>0;i/=2)                     /*連續向右移動 8 位*/
   {
      HAL_SHT_SCK_SET();                     /*clk for SENSI-BUS*/
      if (HAL_SHT_SDA_VAL())
         val=(val | i);                      /*read bit */
      HAL_SHT_SCK_CLR();
   }
   MCU_IO_OUTPUT(HAL_BOARD_IO_SHT_SDA_PORT,
   HAL_BOARD_IO_SHT_SDA_PIN,!ack);     /*當 ack＝1 時拉低數據線*/
   HAL_SHT_SCK_SET();                        /*clk #9 for ack*/
    halMcuWaitUs(1);                         /*pulswith approx 1 us */
   HAL_SHT_SCK_CLR();
   HAL_SHT_SDA_SET();                        /*釋放資料線*/

   return val;
}


void transstart(void)
{
```

```
  HAL_SHT_SCK_SET();
   halMcuWaitUs(1);
  HAL_SHT_SDA_CLR();
   halMcuWaitUs(1);
  HAL_SHT_SCK_CLR();
     halMcuWaitUs(2);
  HAL_SHT_SCK_SET();
   halMcuWaitUs(1);

  HAL_SHT_SDA_SET();
   halMcuWaitUs(1);
  HAL_SHT_SCK_CLR();
     halMcuWaitUs(2);
  HAL_SHT_SDA_CLR();
}

void connectionreset(void)
{
  unsigned char i;

  HAL_SHT_SDA_SET();
  HAL_SHT_SCK_CLR();                    //Initial state

  for(i=0;i<9;i++)                  //9 SCK cycles
  {
     HAL_SHT_SCK_SET();
     HAL_SHT_SCK_CLR();
  }
```

```
  transstart();                          //transmission start
}
```

獲取 SHT10 內部溫濕度資料後的資料處理程式碼，如下所示：

```
void calc_sth11(float *p_humidity ,float *p_temperature)
{
  const float C1=-4.0;                  //for 12 Bit
  const float C2=+0.0405;               //for 12 Bit
  const float C3=-0.0000028;            //for 12 Bit
  const float T1=+0.01;                 //for 14 Bit @ 5V
  const float T2=+0.00008;              //for 14 Bit @ 5V

  float rh=*p_humidity;                 //rh:          Humidity [Ticks] 12 Bit
  float t=*p_temperature;               //*t:          Temperature [Ticks] 14 Bit
  float rh_lin;                         //rh_lin:    Humidity linear
  float rh_true;                        //rh_true: Temperature compensated
                                              humidity
  float t_C;                            //t_C     :    Temperature

  t_C=t*0.01 - 40;                      //calc. temperature from ticks to
  rh_lin=C3*rh*rh + C2*rh + C1;         //calc. humidity from ticks to [%RH]
  rh_true=(t_C-25)*(T1+T2*rh)+rh_lin;   //calc. temperature compensated
                                                humidity [%RH]
  if(rh_true>100)rh_true=100;           //cut if the value is outside of
  if(rh_true<0.1)rh_true=0.1;           //the physical possible range

  *p_temperature=t_C;                   //return temperature
  *p_humidity=rh_true;                  //return humidity[%RH]
}
```

將溫濕度資料分割成單個字元，放入 DateString1 陣列中，最後發送給串列埠，即可。

```
void DateToStr(WENSHIDU *Time,float datax,float datax1)
{
        uint i;
          //    Time->DateString1[0]='9';
       i=(int)datax;
        Time->shiwei =i/10+'0';
        Time->gewei =i%10+'0';
       Time->DateString1[0] = Time->shiwei;
        Time->DateString1[1] = Time->gewei;
        i=(int)((datax-i)*10);
        Time->DateString1[2] ='.';
        Time->DateString1[3] = i+'0';

             Time->DateString1[4] = ' ';
          //    Time->DateString1[5] = ' ';
          //    Time->DateString1[6] = '\0';

       i=(int)datax1;
        Time->shiwei=i/10+'0';
        Time->gewei =i%10+'0';
       Time->DateString1[5] = Time->shiwei;
        Time->DateString1[6] = Time->gewei;
        i=(int)((datax1-i)*10);
        Time->DateString1[7] ='.';
        Time->DateString1[8] = i+'0';
        Time->DateString1[9] = '%';
```

```
            Time->DateString1[10] = '\r';
            Time->DateString1[11] = '\n';
            //Time->DateString1[14] = '\0';
}
```

注意：這裏做的資料轉換只是在 PC 終端下顯示對應的溫濕度資料。若要在 ANDROID 介面上顯示的話，將以上程式碼替換成如下所示：

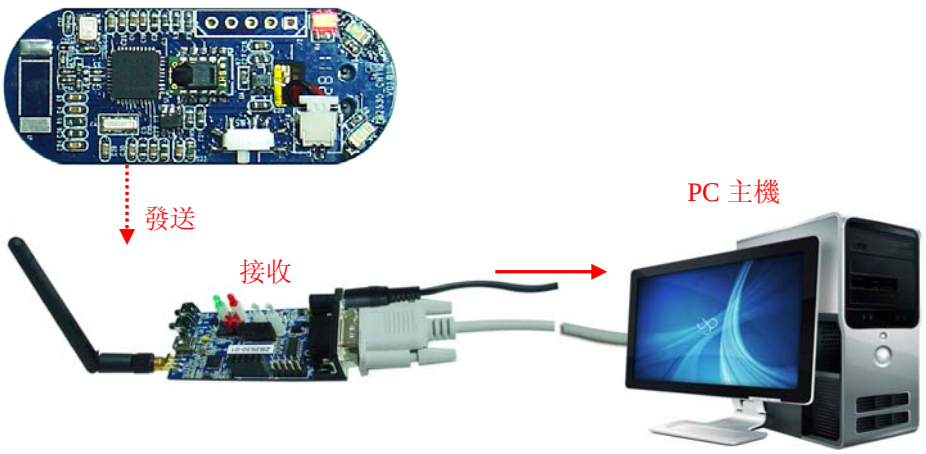
```
void DateToStr(WENSHIDU *Time,float datax,float datax1)
{
      uint i;
            Time->DateString1[0]='9';
      i=(int)datax;
       Time->shiwei =i/10;
       Time->gewei =i%10;
      Time->DateString1[1] = Time->shiwei;
       Time->DateString1[2] = Time->gewei;
       i=(int)((datax-i)*10);
       //Time->DateString1[2] ='.';
       Time->DateString1[3] = i;

          // Time->DateString1[4] = ' ';
          // Time->DateString1[5] = ' ';
          // Time->DateString1[6] = '\0';

      i=(int)datax1;
       Time->shiwei=i/10;
       Time->gewei =i%10;
      Time->DateString1[4] = Time->shiwei;
       Time->DateString1[5] = Time->gewei;
```

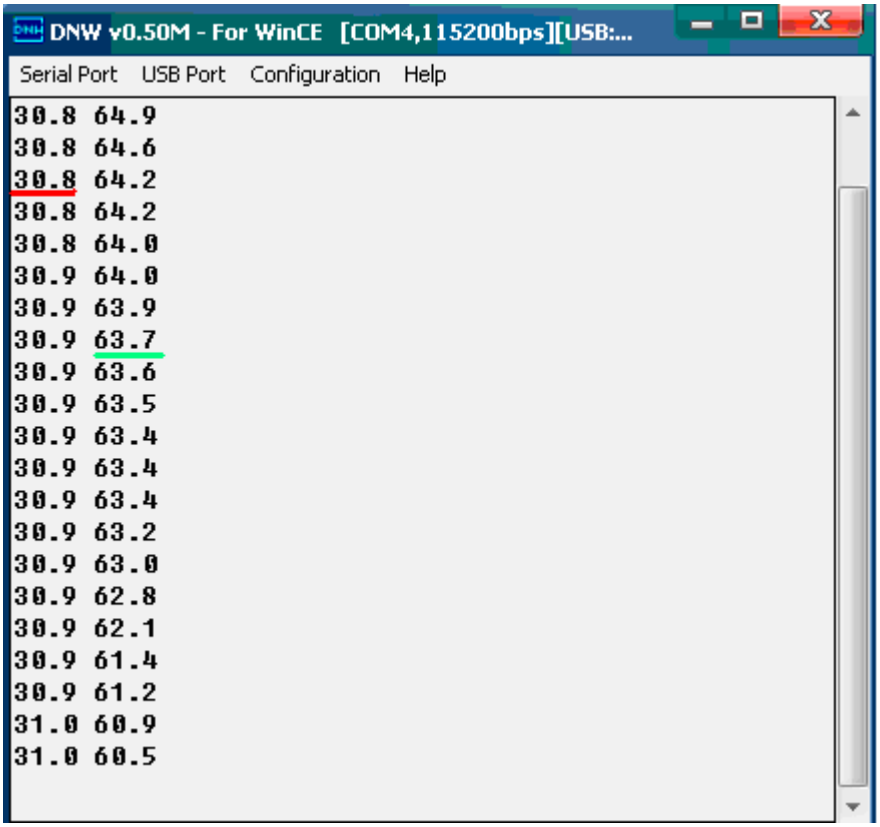
```
        i=(int)((datax1-i)*10);
        //Time->DateString1[8] ='.';
        Time->DateString1[6] = i;
        //Time->DateString1[10] = '%';

            //Time->DateString1[11] = '\r';
          // Time->DateString1[12] = '\n';
          // Time->DateString1[13] = '\0';
}
```

3-5 ZB2530-SHT10 模組在 PC 及 Android 下的結果以及現象

1) 基於 PC 下的現象

我們首先將 SHT10.hex 透過模擬器燒錄進 ZB2530_SHT10（燒錄過程見 3-3 節），然後將 rf_send_recv.hex 燒錄進 ZB2530-01（燒錄過程見附錄一），最後透過串列線將 ZB2530-01 的串列埠與 PC 的串列埠相連，打開串列埠除錯工具（串列傳輸速率 115200），結果如下圖所示：





注意：紅色下劃線的是溫度資料，綠色下劃線的是濕度資料。

2) 基於 Android 下的現象

我們首先將 SHT10.hex 透過模擬器燒錄進 ZB2530_SHT10，然後將 rf_send_recv.hex 燒錄進 ZB2530-01，最後透過串列線將 ZB2530-01 的串列埠與 Android 平台的串列埠相連（如：DMA-6410XP 平台下的串列埠 2，如下圖）。

將提供的感測器測試 APK，即：Sensors_uart2.apk 安裝到實驗平台下，結果如下圖所示：

第四章 DMA-Android 感測器圖控軟體介紹

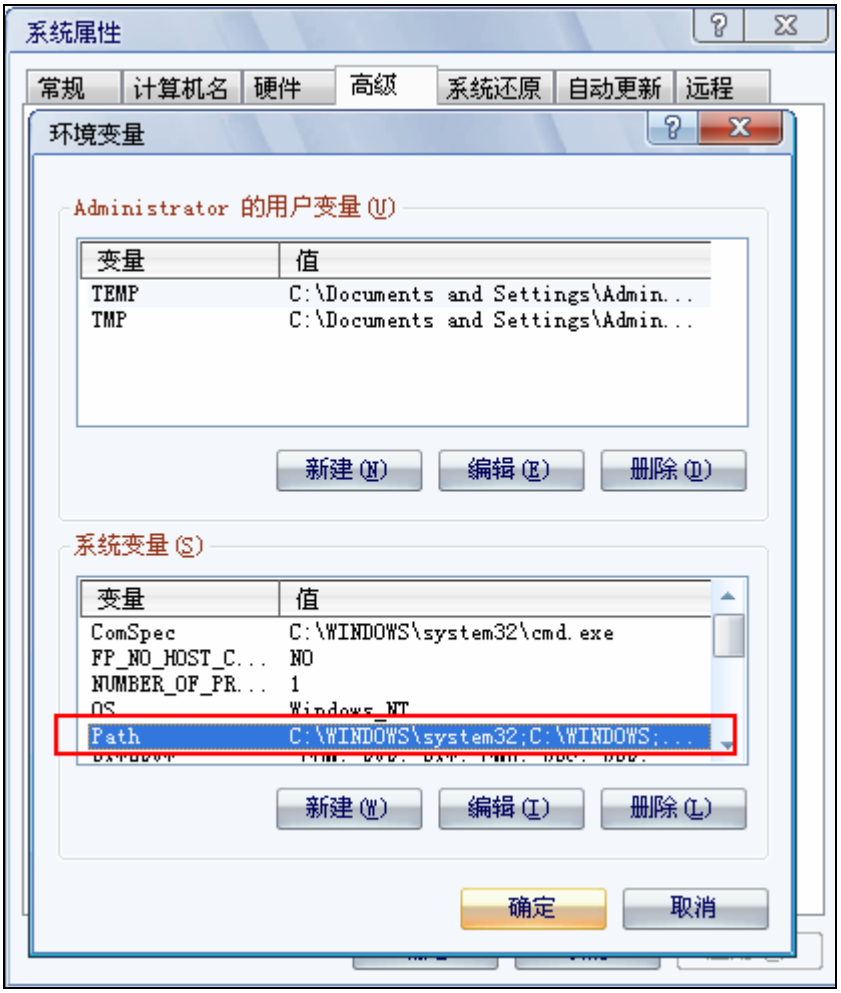
4-1 Windows 環境搭建

4-1.1 安裝 JDK

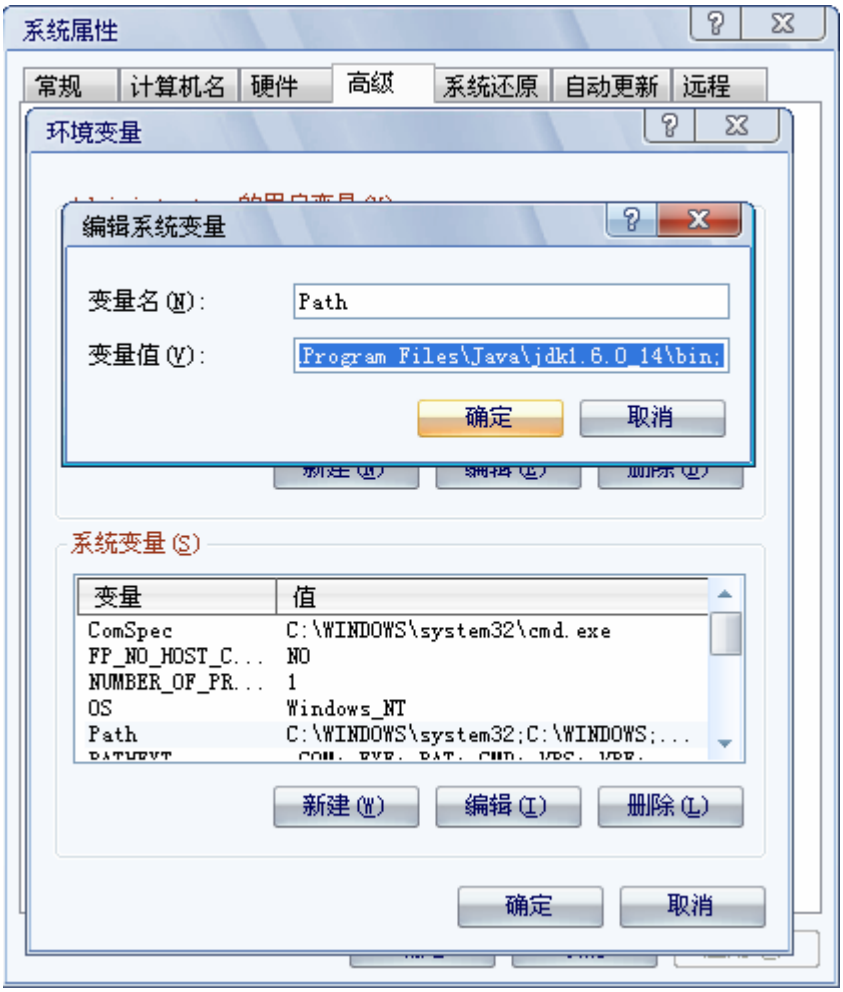
安裝 Eclipse 的開發環境需要 JRE 的支援，如果沒有 JRE，則啓動Eclipse時會報告錯誤。

在 Windows上安裝JRE/JDK 非常簡單，首先在 Sun 官方網站上下載 JDK，網址爲：http://java.sun.com/javase/downloads，下載完後，點兩下JDK安裝檔，打開安裝精靈，然後按照預設的設置進行安裝。預設的情況下JDK安裝路徑爲：C:\Program Files\Java。

安裝完成後還需配置 JDK 的環境變數，將 JDK 的 bin 檔的路徑C:\Program Files\Java\jdk1.6.0_14\bin 添加到 Path 中，右鍵打開“我的電腦”，依次選擇“屬性”->“高級”->“環境變數”選項，選擇“系統變數”中的“PATH“選項，如下圖所示：

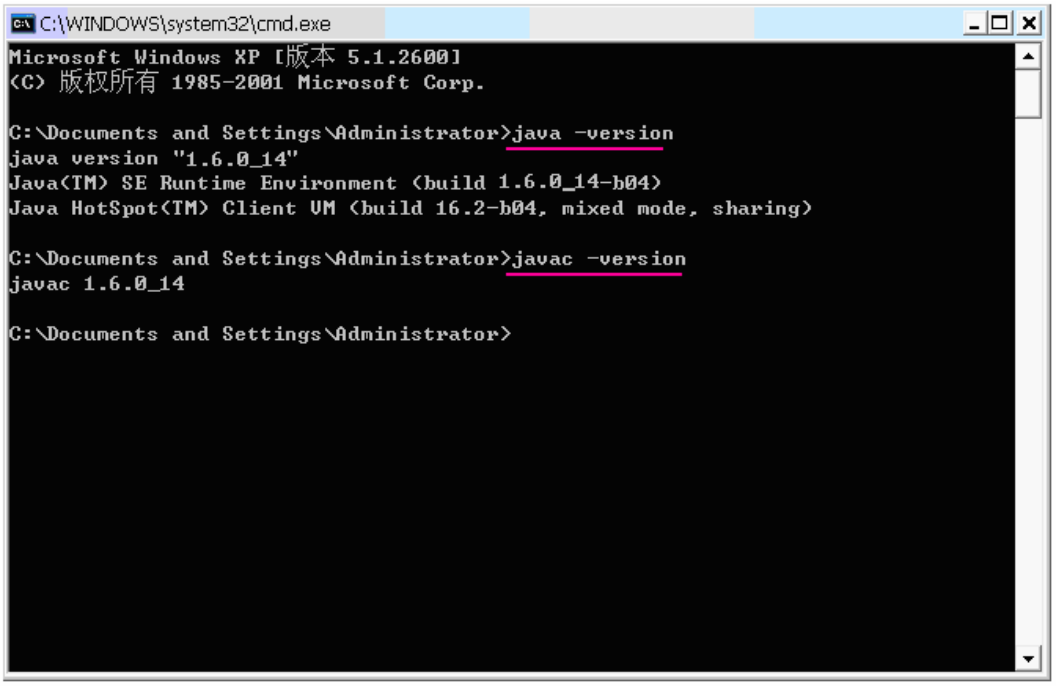


點選編輯按鈕，將 JDK 所在路徑添加進入即可，句尾以分號結尾，如下圖所示：



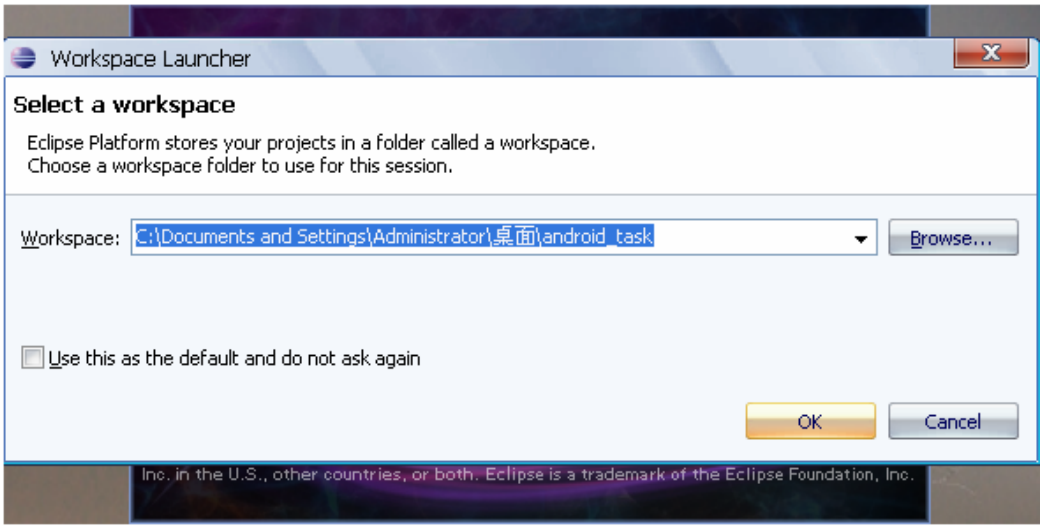
配置完成後，依次點選確定。

配置完成後，打開“開始”，選擇“執行”，在出現的對話方塊中輸入 cmd 命令，打開 cmd 視窗，在視窗輸入命令：java -version，如果出現下圖所示資訊說明 JDK 安裝成功。


C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.
C:\Documents and Settings\Administrator>java -version
java version "1.6.0_14"
Java(TM) SE Runtime Environment (build 1.6.0_14-b04)
Java HotSpot(TM) Client VM (build 16.2-b04, mixed mode, sharing)
C:\Documents and Settings\Administrator>javac -version
javac 1.6.0_14
C:\Documents and Settings\Administrator>

4-1.2 安裝 Eclipse

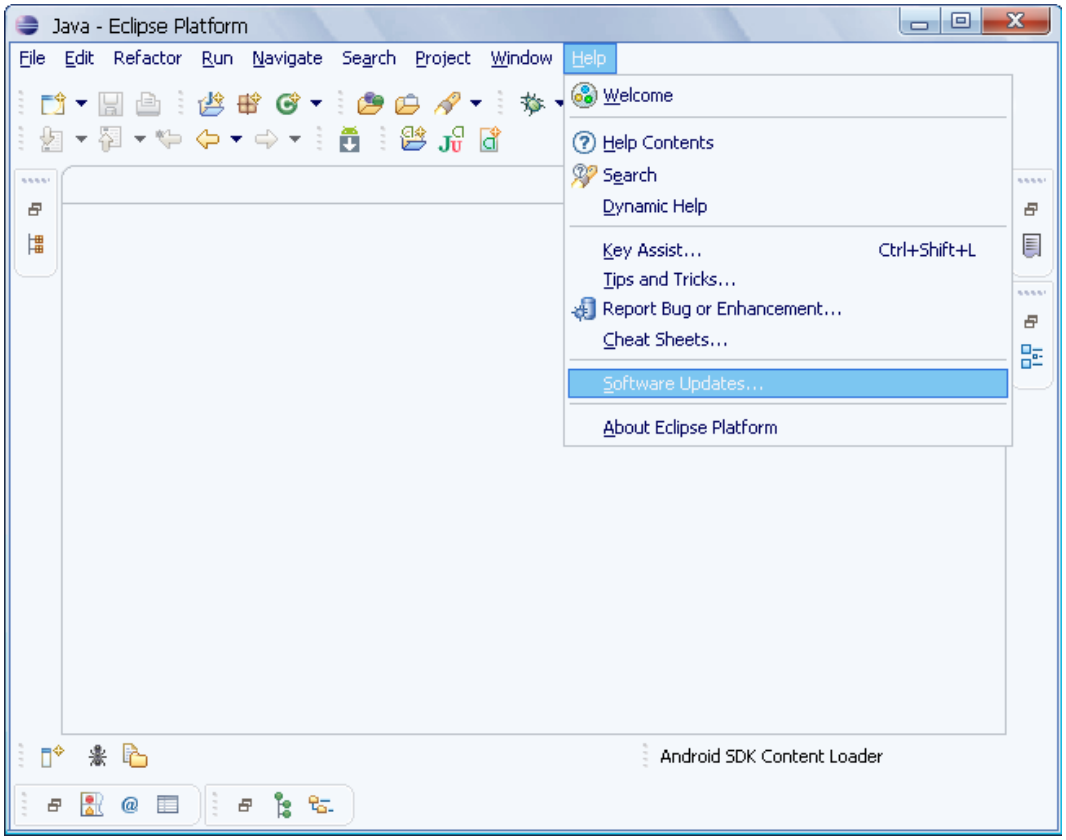
安裝好 JDK 後，就可以安裝 Eclipse 了，打開 Eclipse 下載頁面，網址爲：http://www.eclipse.org/downloads/，下載完成後，解壓下載的壓縮包檔就可以使用。進入解壓目錄，可以看到一個名爲 eclipse.exe 的可執行檔，點兩下此檔直接執行 Eclipse，如果用戶是第一次啓動 Eclipse，將會看到選擇工作空間的提示，如下圖所示：



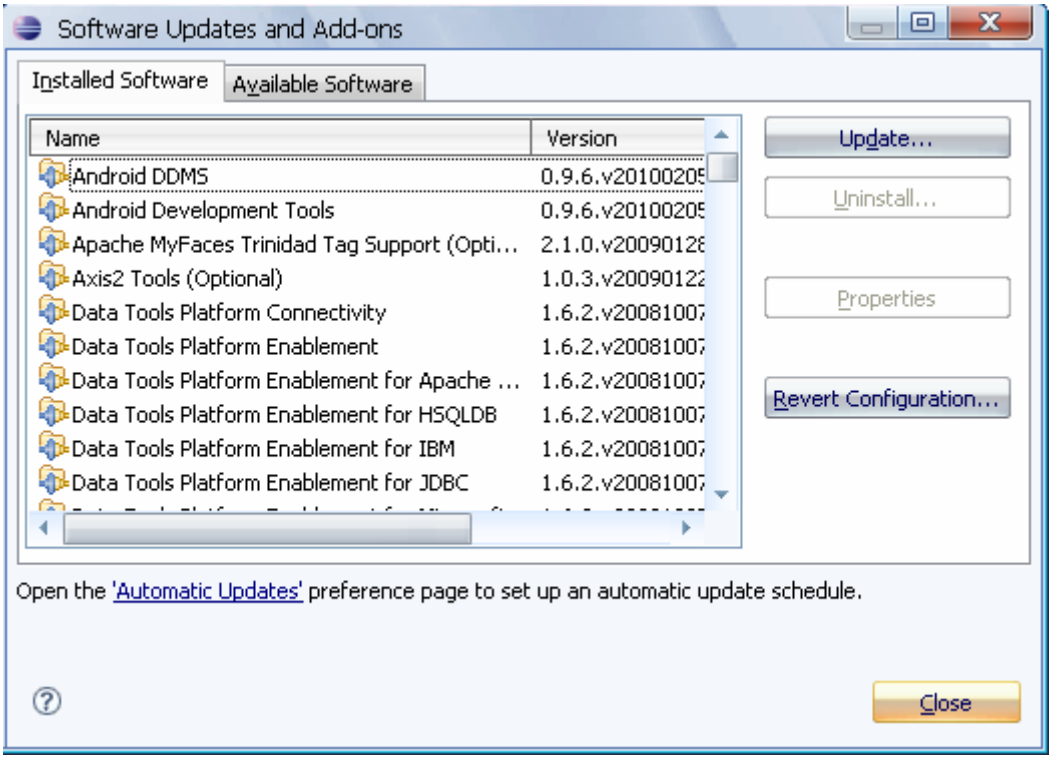
選擇工作空間路徑，然後點選“OK“按鈕即可。

4-1.3 安裝 ADT

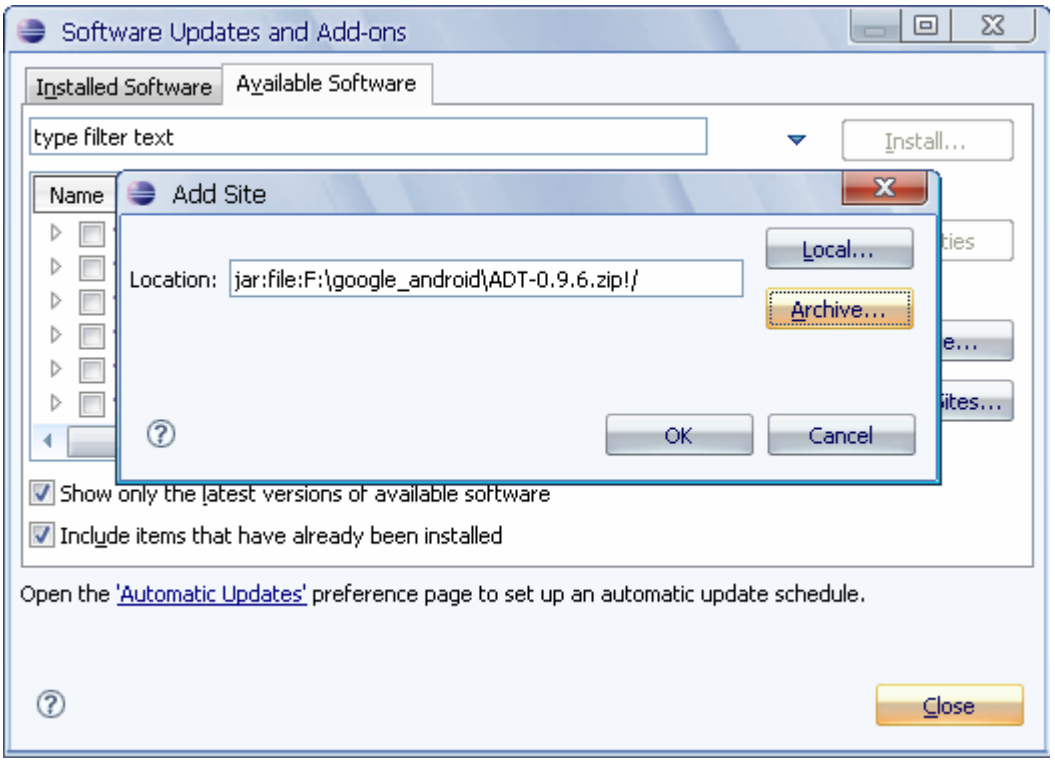
打開 ADT 下載頁面，下載 ADT-0.9.6，網址爲：http://androidappdocs.appspot.com/sdk/eclipse-adt.html。下載完成後，啓動 Eclipse，點選選擇「Help > Software Updates」。



進入如下圖所示對話方塊：

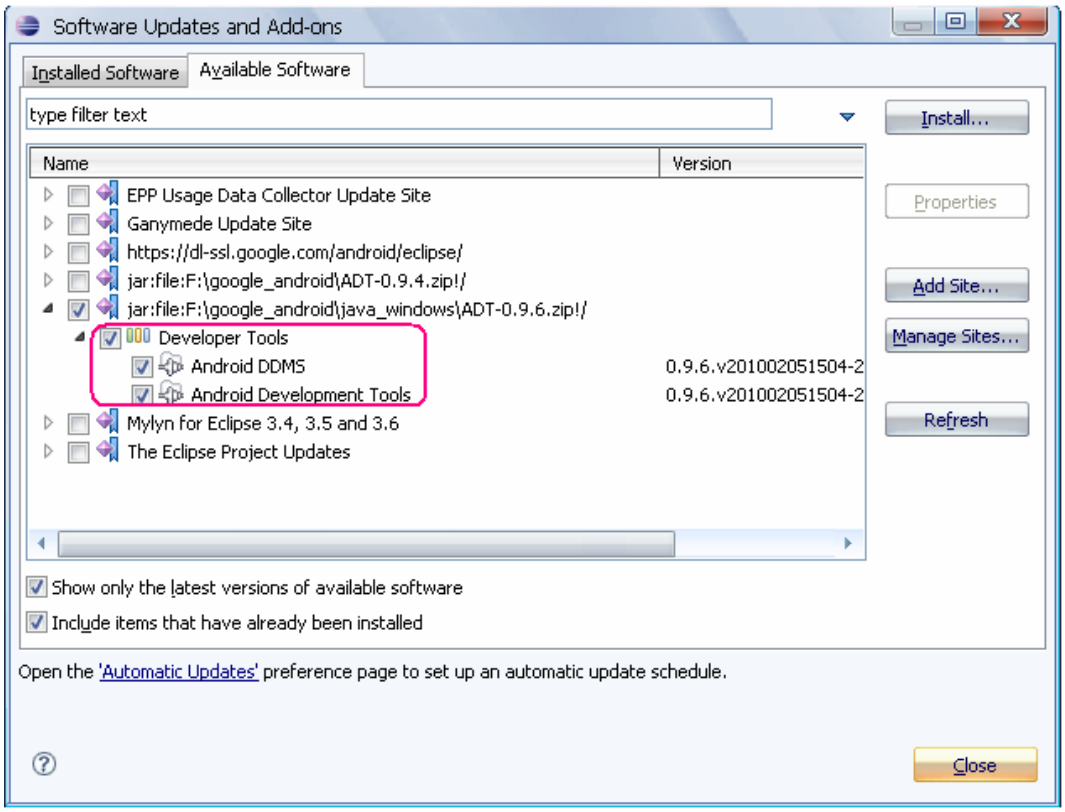


選擇 Available Software 頁面。然後點選 Add Site 按鈕，然後在 Add Site 對話方塊中輸入 ADT 存放的路徑，如下圖所示：



點選"OK"確定。

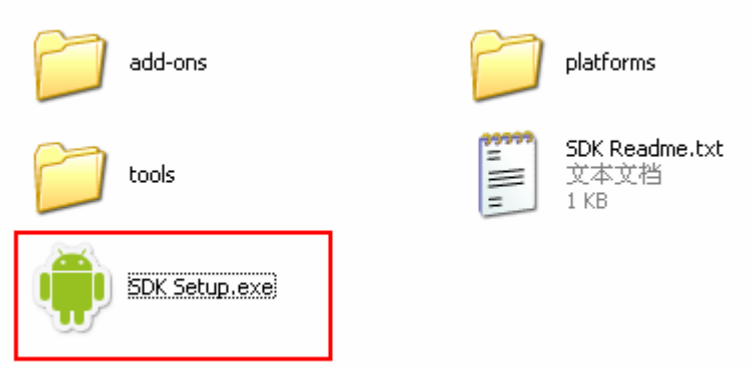
然後勾選住 Developer Tools 選項，點選 Install 安裝。如下圖所示：



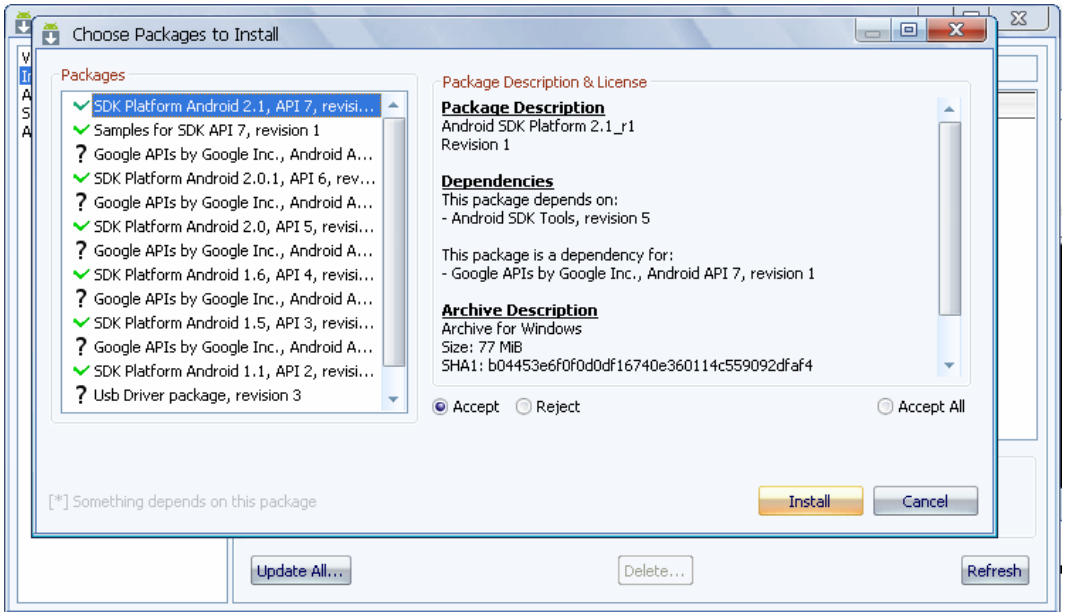
按照預設的設置進行安裝，安裝完成後重新啓動 Eclipse 即可。

4-1.4 安裝 SDK

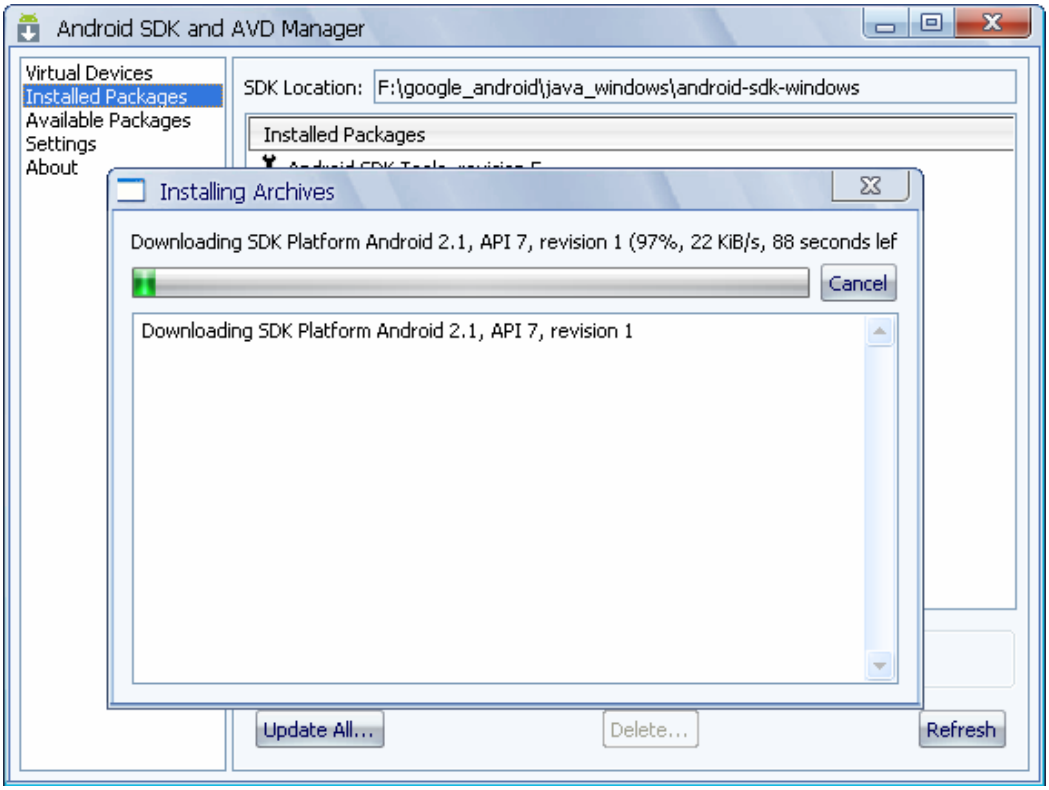
首先得下載 SDK2.1，網址爲 http://androidappdocs.appspot.com/sdk/，下載 Windows 平台下的 SDK 包 android-sdk_r05-windows.zip，解壓後的路徑爲：F:\google_android\java_windows\android-sdk-windows\tools。由於 Android SDK2.1 不再捆綁 platform 和 add-on，因此這兩部分需要手工下載。點選 SDK 裏附帶的"SDK Setup.exe"，如下圖所示：



進入如下圖介面：



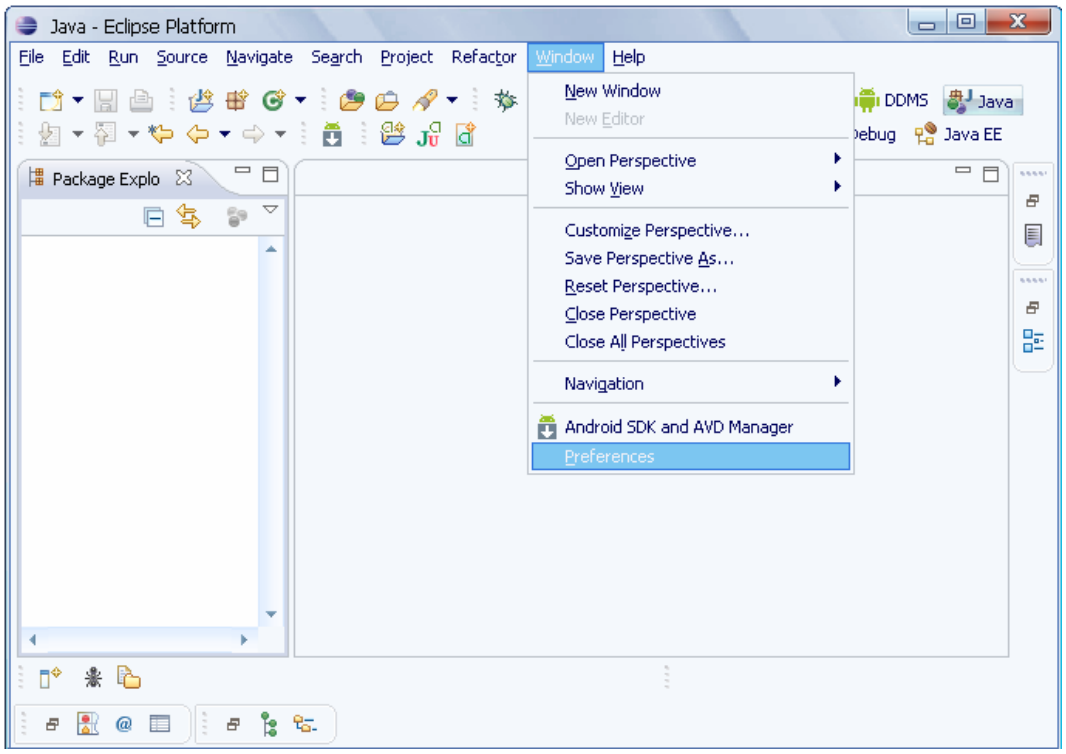
選擇 Accept All，然後點選 Install 即可開始安裝並下載。



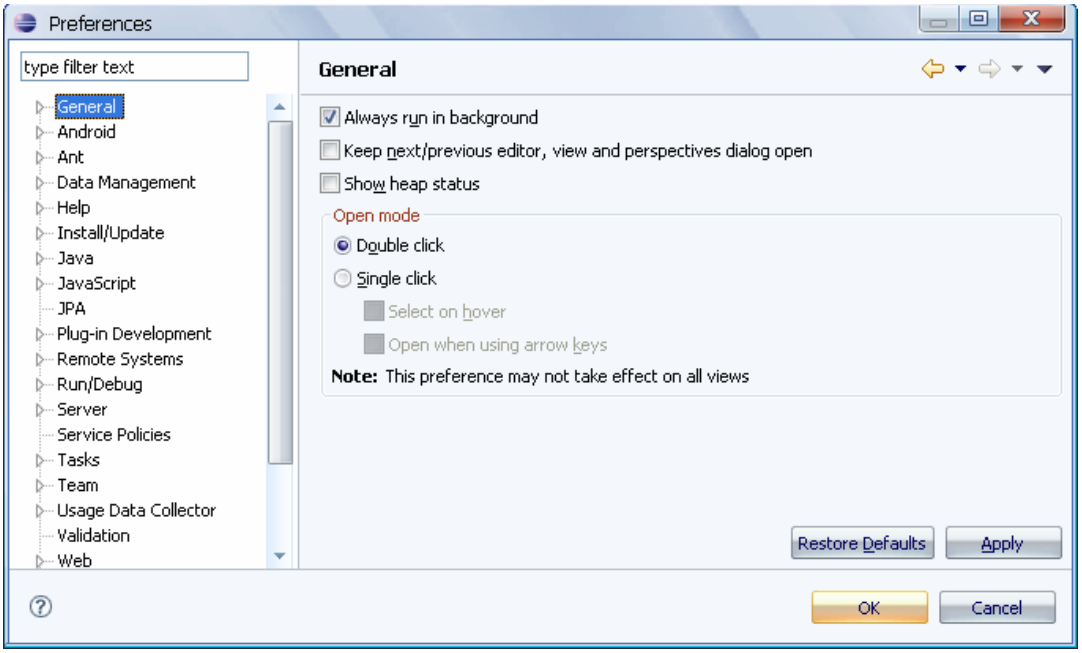
安裝完成後，退出此頁面，然後對 SDK 進行配置。

需要將 Android SDK 安裝目錄中的 tools 檔案夾路徑(F:\google_android\java_windows\android-sdk-windows\tools)添加到環境變數 PATH 中，具體配置過程在此不再講述，請讀者參看 JDK 環境配置過程。

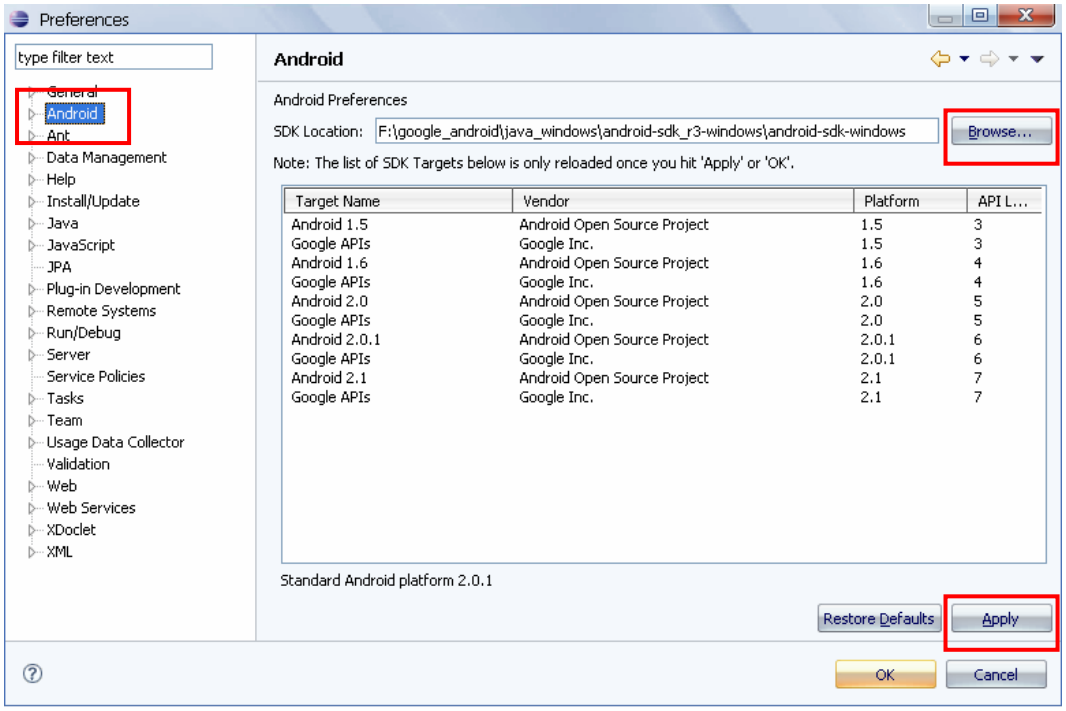
接著重新啓動 Eclipse，並在 Eclipse 的 Preferences 中添加 Android SDK 的路徑。如下圖所示，由 Windows 選單開啓 Preferences 設定。



然後會顯示如下圖所示的對話窗口



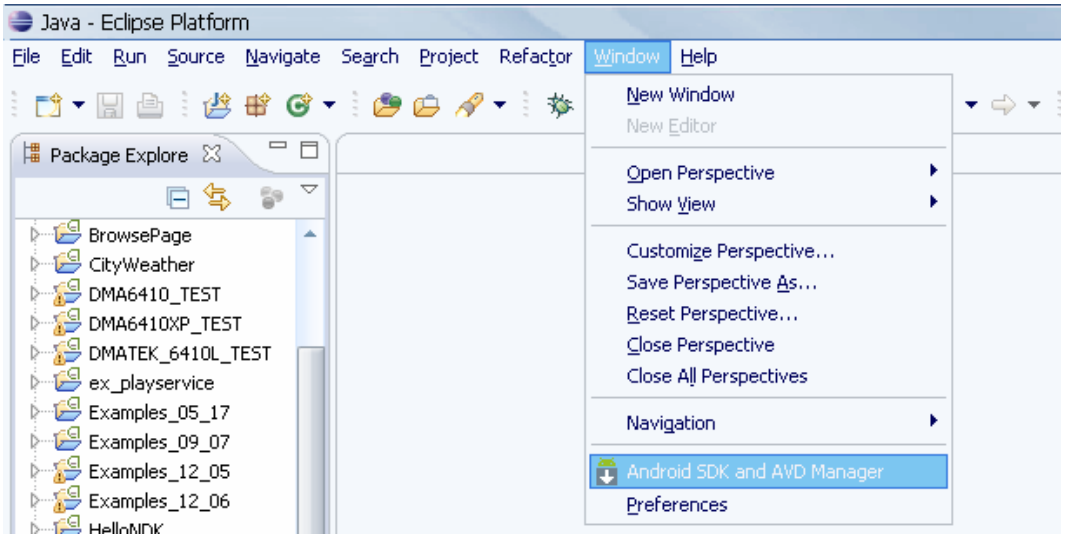
點選 Android，按下 Browse 按鈕並選擇 Android SDK 的路徑。如下圖所示：



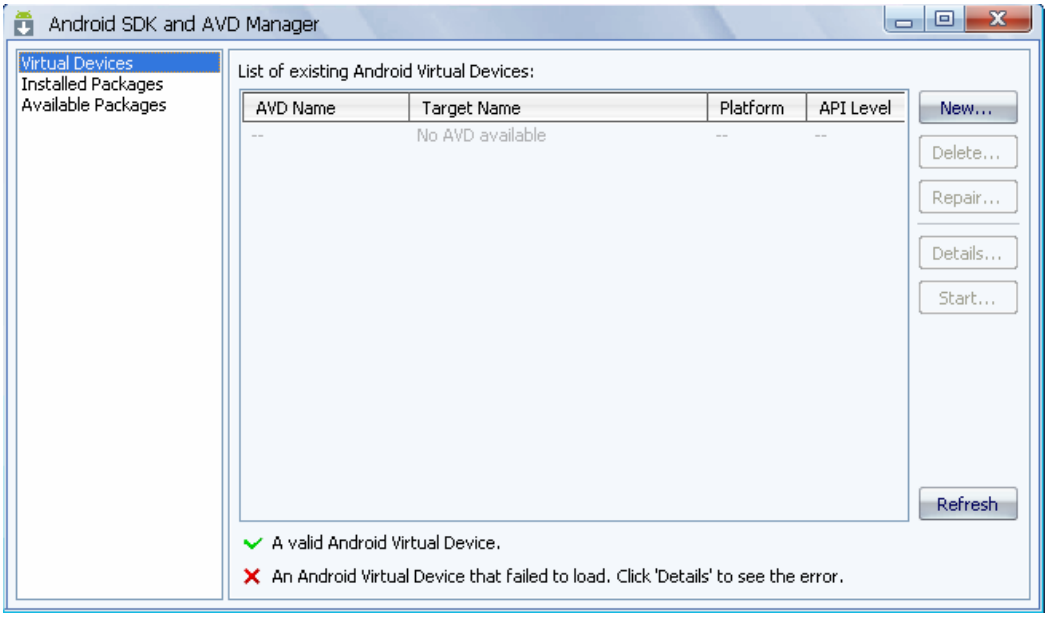
點選“Apply“按鈕進行載入，載入完成後點選“OK“退出。

4-1.5 創建 Android 虛擬設備(AVD)

如下圖所示，依次點選“Window”->“Android SDK and AVD Manager”。

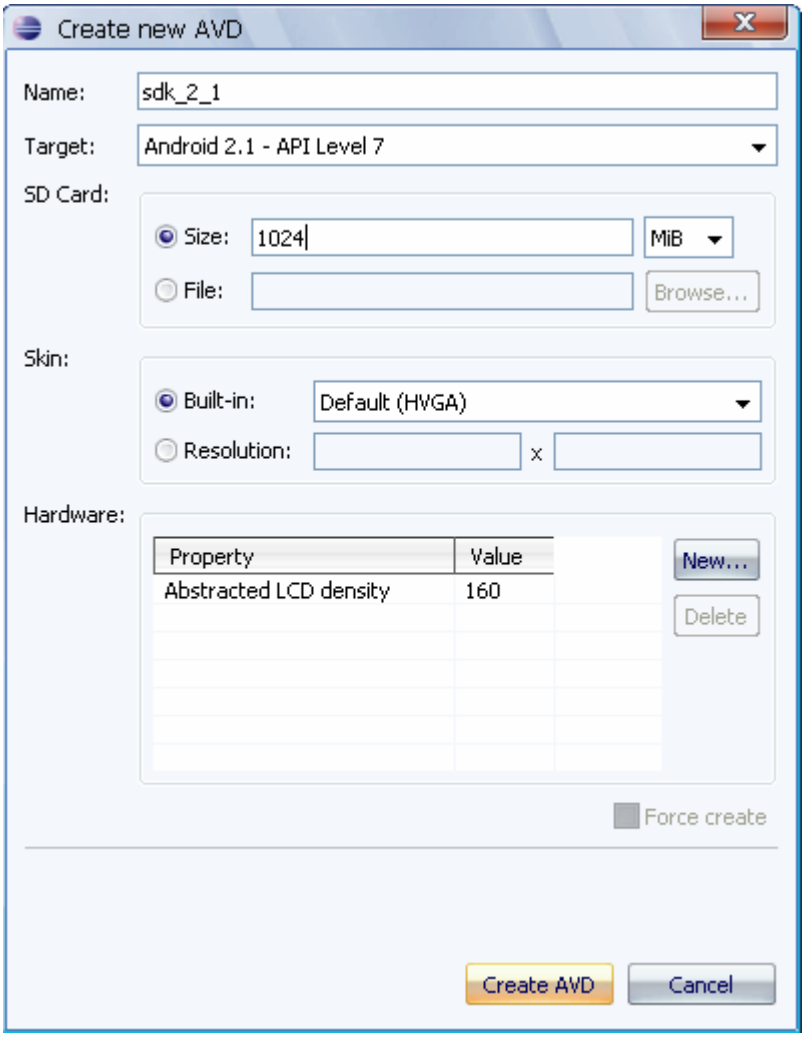


將進入下圖所示介面：



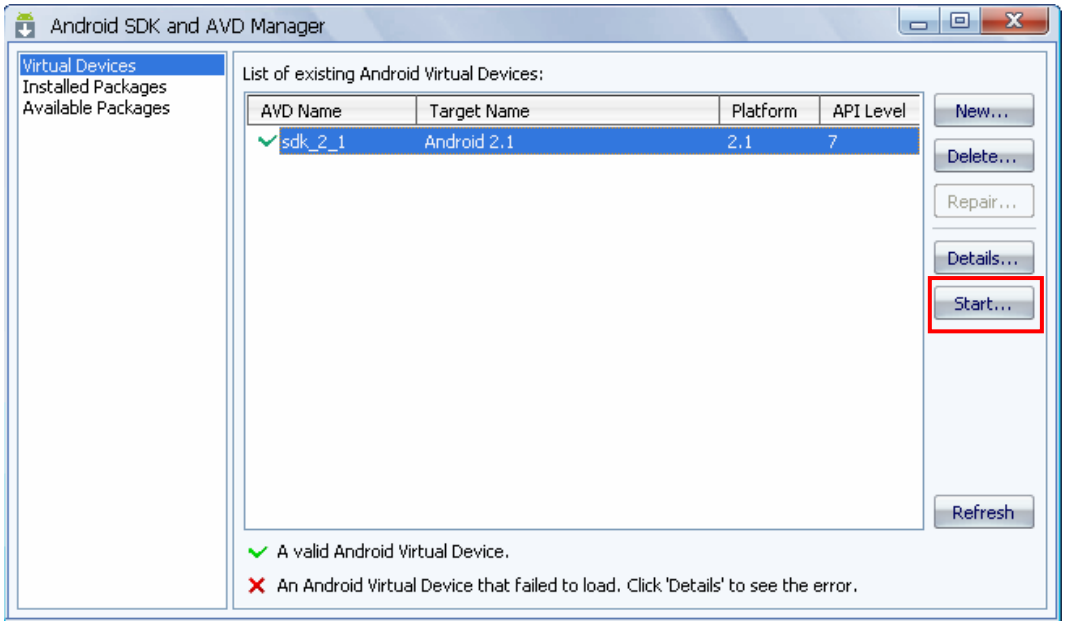
此時無任何 AVD 設備，AVD 虛擬設備是類比了一套虛擬設備來執行 Android 平台，這個平台有自己的核心和資料分區。現在需要創建一個 AVD 設備。點選“New”按鈕。

出現下圖所示對話窗口：



Name 選項爲創建的模擬器的名稱，Target 爲創建模擬器的版本，Size 爲 SD 卡的容量。

在此對話方塊中填入 Name，Target 等選項，創建完畢後點選“Create AVD”按鈕即可。

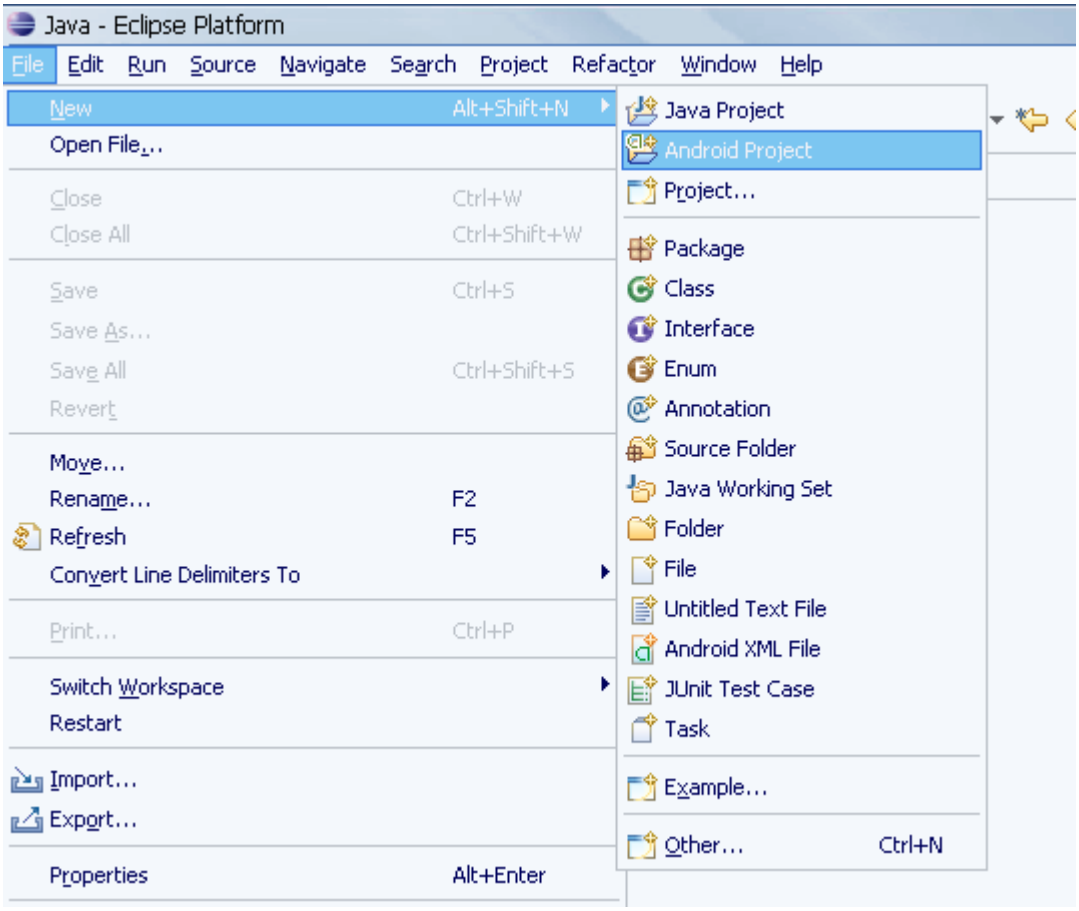


此時虛擬設備創建成功，選中創建的虛擬設備，然後點選“Start“按鈕，啓動模擬器。

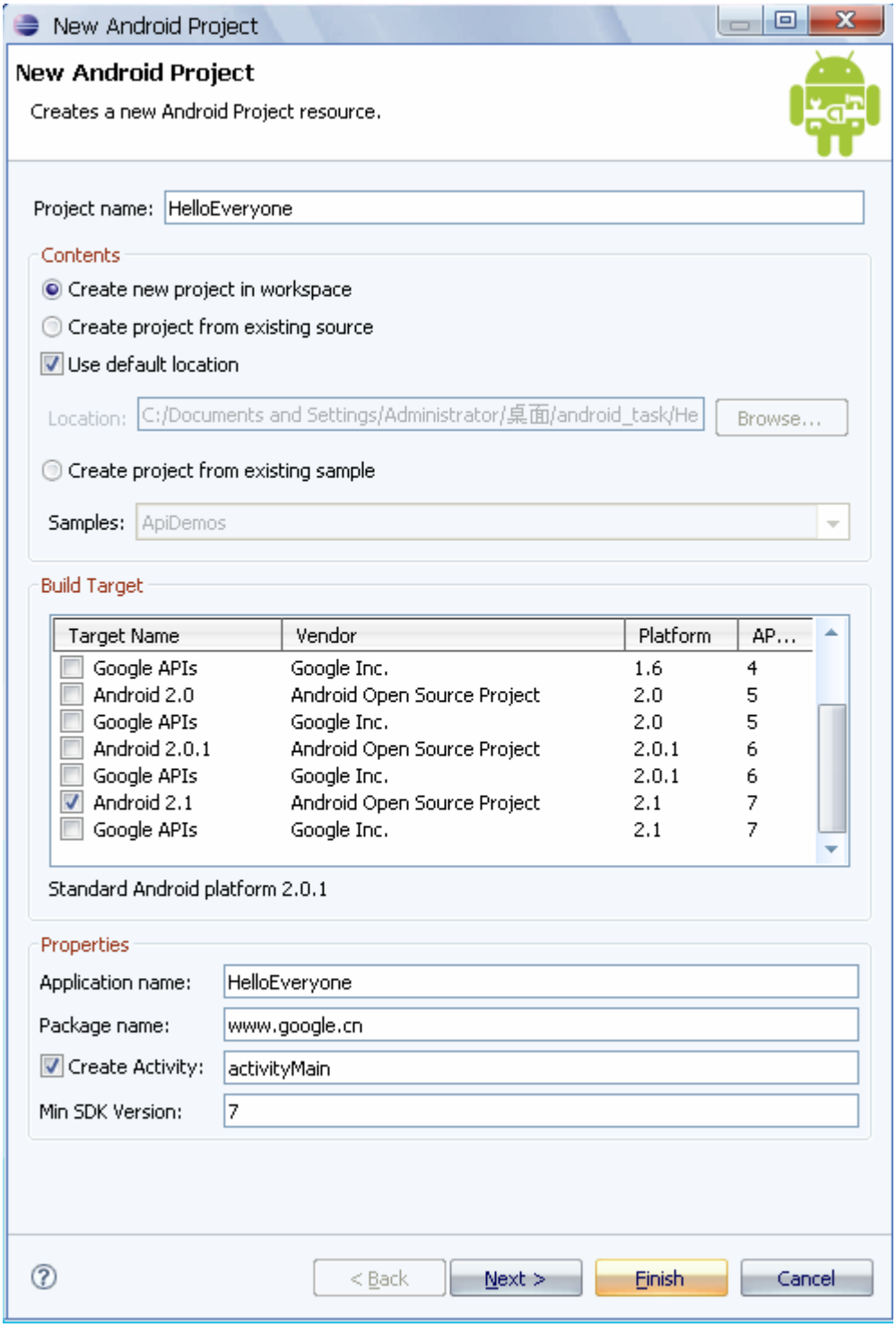
進入上圖所示介面表示模擬器創建成功。

4-1.6 創建第一個 Android 項目 HelloEveryone

打開 Eclipse，依次選擇“File”->“New”->“Android Project”，如下圖所示：

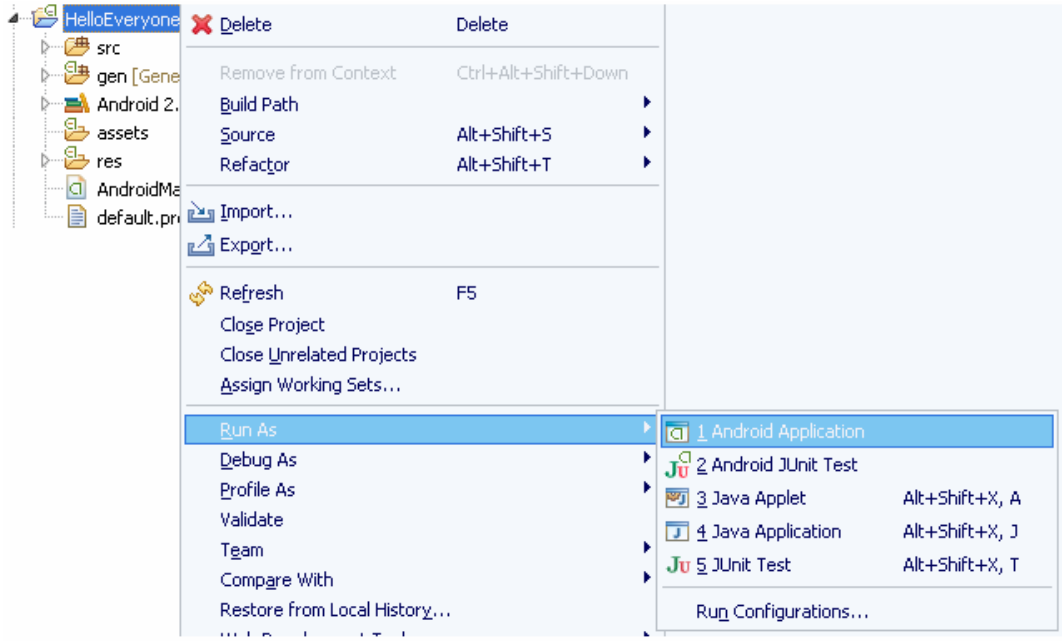


點選 Android Project 後進入下圖所示介面，需要填寫相關工程資訊。下圖是已經填好的工程資訊：

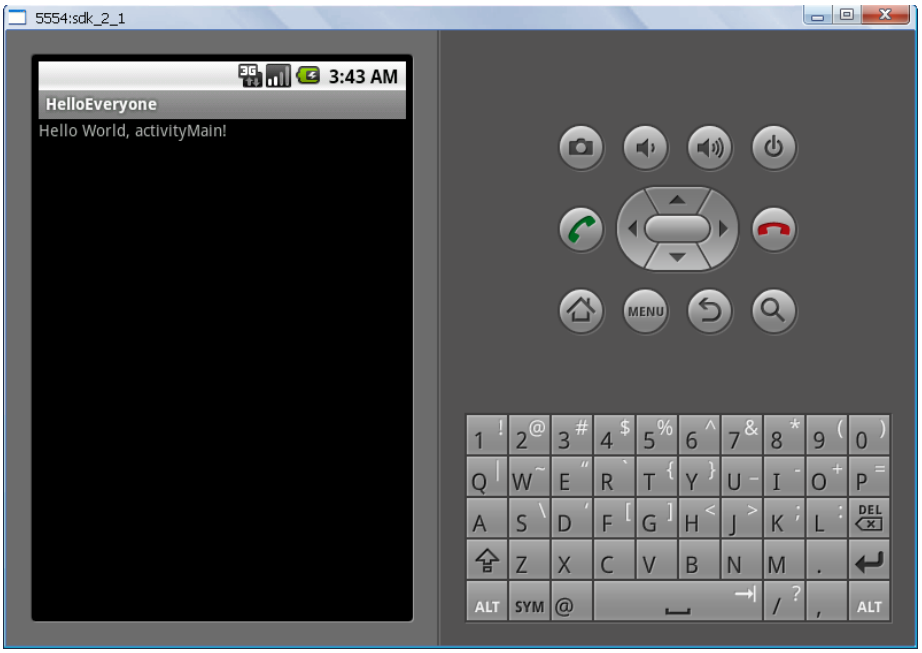


填寫完畢後點選 Finish 按鈕即可。

右鍵打開“HelloEveryone“，依次選擇“Run As”->“Android Application”，啓動模擬器並執行此程式。



模擬器顯示如下表明 Android 開發環境配置成功！



4-2 APK 安裝器

APK 安裝器在 Android 系統中起到一個橋樑的作用，透過它可以把外部應用程式植入到 Android 系統內部，同時也可以卸載部分應用程式，大大方便了對應用程式的除錯，所以 APK 安裝器在 Android 系統中具有舉足輕重的作用。下面就介紹一下如何使用 APK 安裝器來安裝應用程式。

點選桌面圖示 

進入如下圖所示介面：

點選“安裝 Installer”將進入如下圖所示的 APK 安裝介面：

選中所需要安裝的 apk 檔進行安裝即可。點選“管理 Manager”鍵進入 apk 卸載介面，選中要卸載的軟體，點選就可以開始卸載了。點選“退出 Exit”按鈕即可退出此應用程式。

4-3 介面介紹

4-3.1 在7吋螢幕上的介面介紹

安裝好我們編寫好的APK後，我們現在可以進行各個模組的測試。

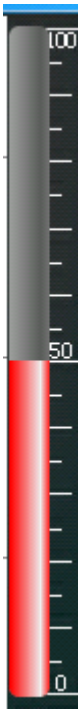
點選如下圖紅框處的APK 圖示：

進入如下圖的主介面：

感測器功能模組：這裏有我們編寫的 10 個感測器模組，當測試每一個模組時，可以選擇進行測試。如下圖所示：

曲線繪製：這是我們繪製曲線區域，在這裏我們繪製出，當您選擇某個模組時對應這個模組電壓值的即時曲線。例如，溫度值、電壓值。如下所示：

光柱圖繪製：這是我們自己編寫的光柱圖，當您選擇某個模組對應的即時值，例如，溫度值、電壓值等，這個值與曲線值對應的。如下圖所示



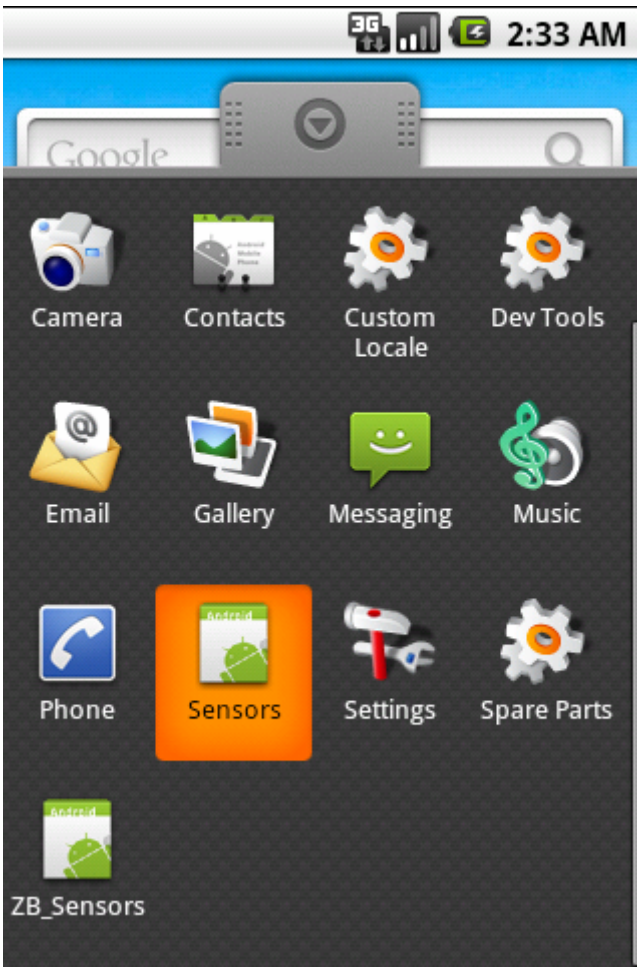
資料顯示：這是資料顯示區域，當您選擇某個模組時，模組名稱會變成您選擇模組的名稱，下面會出現相關的資料標籤和相應的值，在後面我們會一一看到。如下圖所示：

系統功能按鈕：當您選擇了某個模組時，點選開始，就可以進行資料的採集了，當您要離開這個介面是點選退出按鈕，就可以離開當前介面。如下圖所示：

到此，我們就把這個介面介紹完成了，下面將介紹各個模組如何進行測試，和資料怎樣的顯示。

4-3.2 在4.3吋螢幕上的介面介紹

在我們安裝好 APK 後，點擊下面紅框處的應用程式，如下圖所示

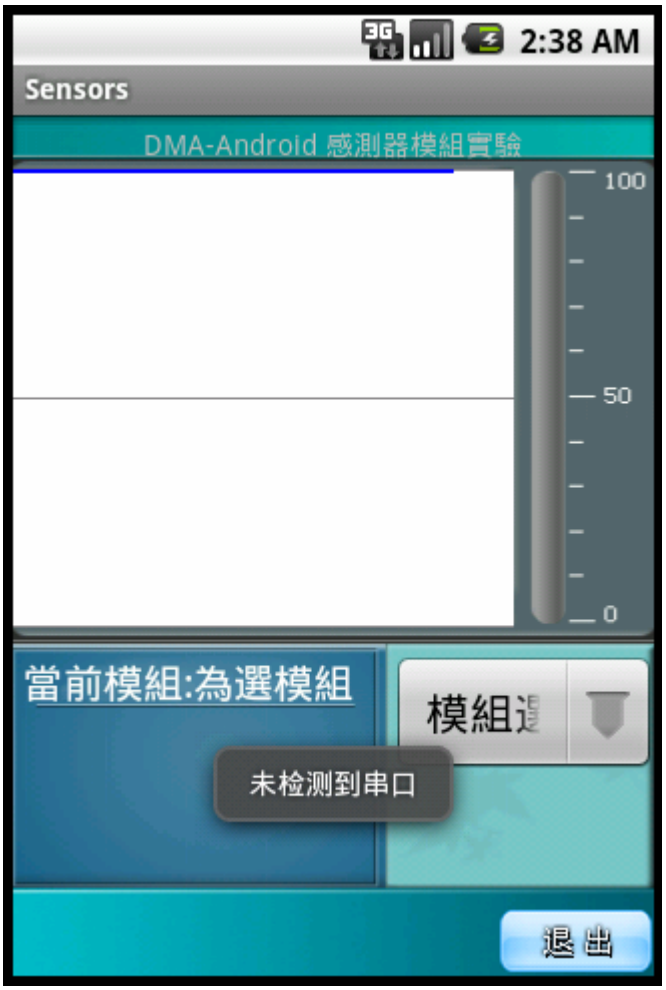


安裝好的 APK

點擊打開，出現下圖

進入介面

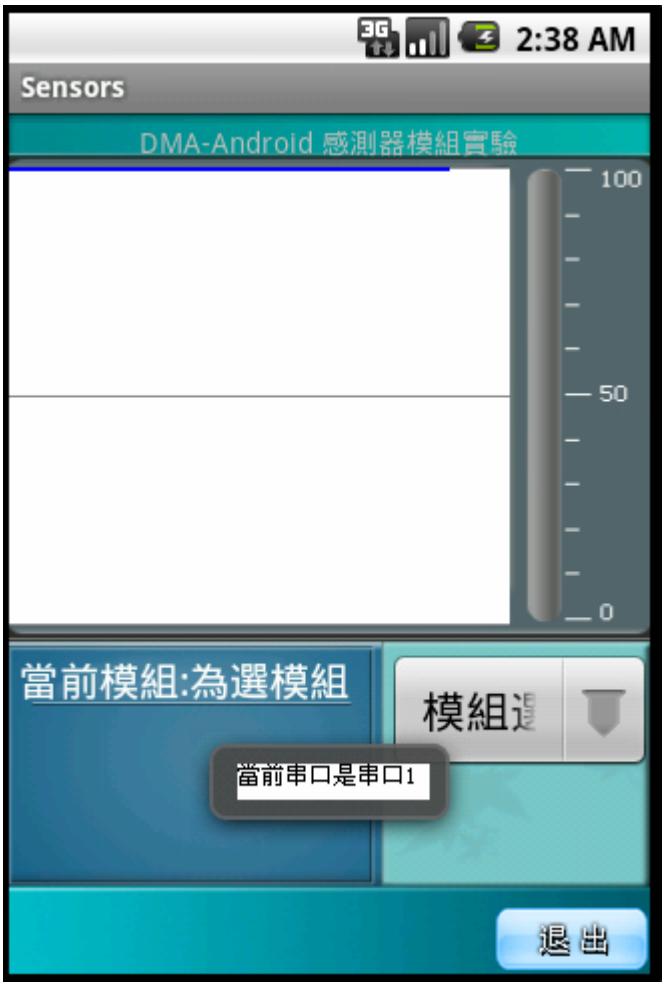
這個介面是剛剛開始時的介面，開始檢測你插入的串列埠，我們會自動檢測你插入的串列埠，並檢測出來，給與提示。



未發現串列埠時的提示資訊

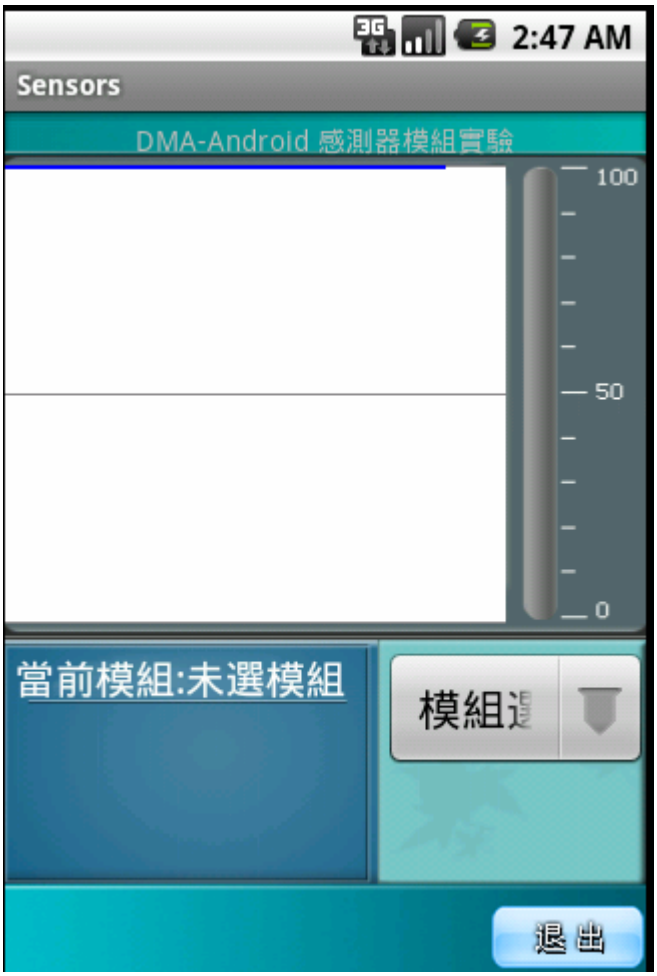
我們這裡檢測了 7 個串列埠，分別是普通的串列埠 1，2，3 和 usb 轉串列埠 0，1，2，3

當檢測但串列埠時的提示資訊如下圖



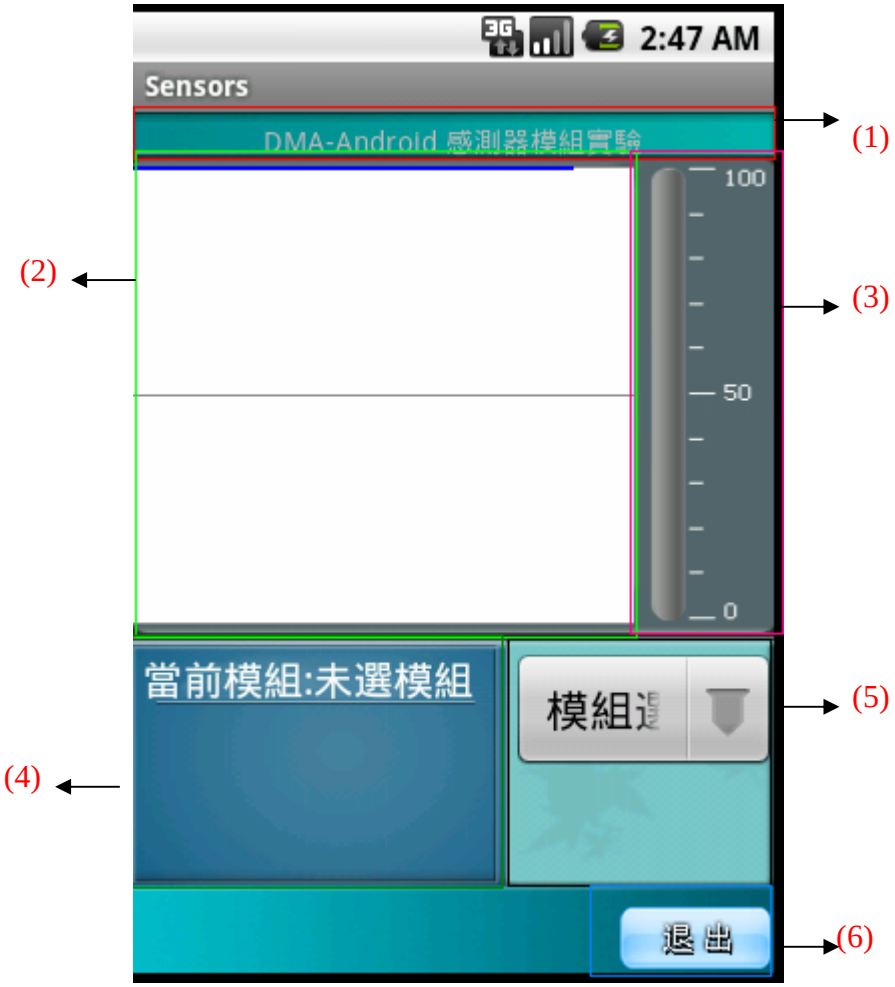
檢測到串列埠時的介面

最後進入到下面的介面



總介面

我們把這個介面分爲 6 個部分



1. 第一塊是標題顯示
2. 第二塊是曲線繪製顯示
3. 第三快是光柱圖顯示
4. 第四塊是獲取的資訊顯示
5. 第五塊是選擇模組
6. 第六塊是突出系統

在下面的測試實驗中，我們會介紹是如何顯示的。

4-4 感測器介面測試

4-4.1 7 吋螢幕的介面測試

接下來溫濕度感測器的測試及顯示，首先我們進入到下面的介面：

點選溫濕度模組按鈕，出現如下圖所示介面：

然後點選開始按鈕，介面變成了如下所示：

由於這裡有兩個模組，所以就沒有在顯示其他曲線及光柱了。溫度值是溫濕度模組中溫度模組採集的數據，濕度值是溫濕度模組中濕度模組採集的數據。

4-4.2 4.3 吋螢幕的介面測試

當我們接好串列埠時，並啓動程式，最終進入到我們的介面，如下所示

主介面

這時候如果你插入了串列埠模組，並提示已經打開了串列埠，你只要在上圖紅框中選擇你現在在測試的模組，觀察曲線和光柱及資訊顯示出的資訊接好了，當你換了一個模組時，也只需要選擇你相應的模組即可。觀察和 7 吋螢幕的類似，再從不在累贅。

4-5 ZB2530-01 LED DEMO 實驗

硬體連線

將我們的開發平台上的串列埠與 ZB2530_01 連接，如下圖所示，並插上電源；如下圖所示。

實驗

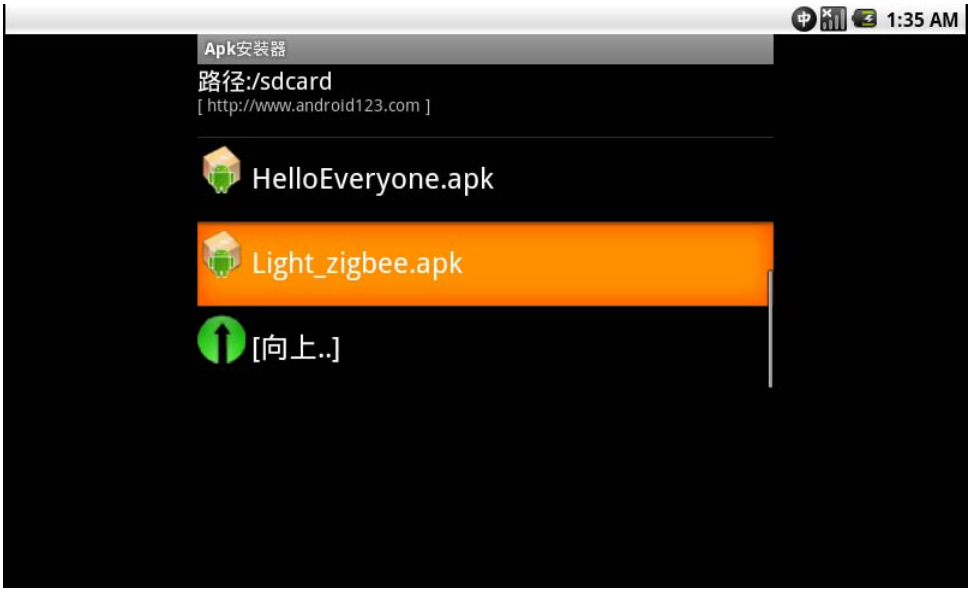
一、 使用 APK 安裝器安裝我們編寫好的程式

APK 安裝器在 Android 系統中起到一個橋樑的作用，透過它可以把外部應用程式植入到 Android 系統內部，同時也可以卸載部分應用程式，大大方便了對應用程式的除錯，所以 APK 安裝器在 Android 系統中具有舉足輕重的作用。下面就介紹一下如何使用 APK 安裝器來安裝應用程式。

點選桌面圖示 

進入如下圖所示介面：

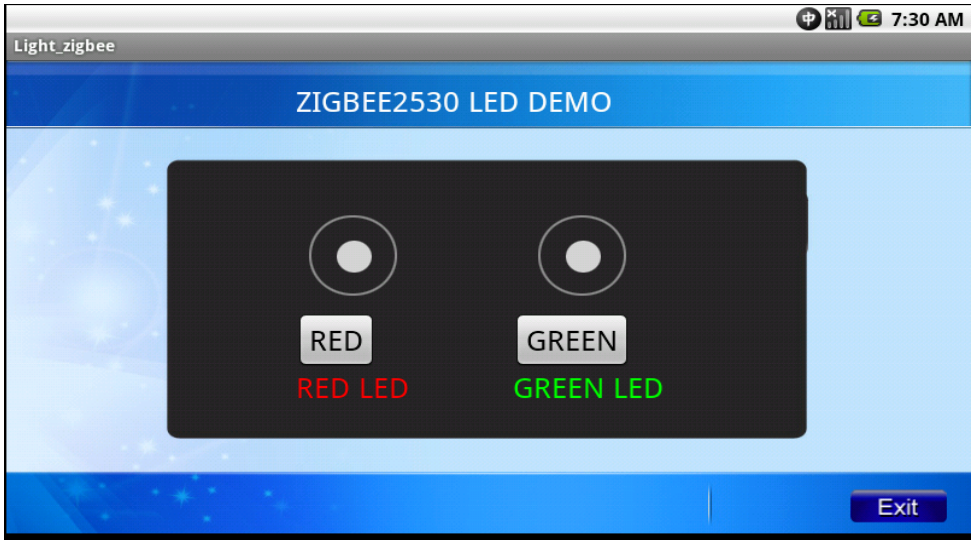
點選“安裝 Installer”將進入如下圖所示的 APK 安裝介面：



選中所需要安裝的apk檔進行安裝即可。點選“管理 Manager”按鍵進入apk卸載介面，選中要卸載的軟體，點選就可以開始卸載了。點選“退出 Exit”按鈕即可退出此應用程式。

二、進行實驗

打開安裝好的APK，如下圖：



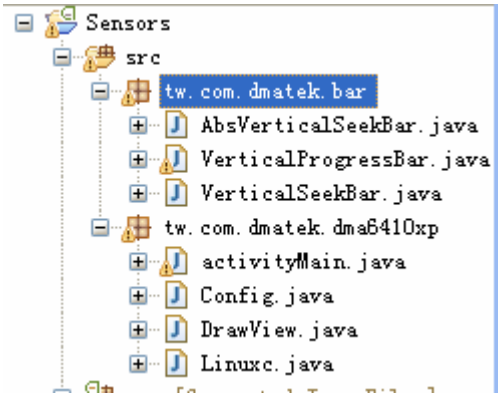
1. 我們先來測試左邊的紅燈，您點擊左邊的紅燈按鈕，這時您可以看到左邊的紅燈在不停的閃爍，同時您觀測 ZigBee 另一端的紅燈，也在不停的閃爍。再點擊時，介面的紅燈不在閃爍，ZigBee 另一端的也停止閃爍。

2. 現在測試右邊的綠燈，點擊綠燈按鈕，綠燈亮，同時另一端 ZigBee 上的綠燈也亮，再點擊綠燈，綠燈滅，同時另一端的綠燈也滅。

如此，這個實驗展示完成了。

4-6 程式碼編譯說明

感測器的 Java 程式碼結構如下圖所示：



其中有兩個包，一個是對 progressBar 和 seekBar 進行的重寫的包，由於 android 下的 progressBar 和 seekBar 是水準的，並且設置都很不方便，爲此，在這裏對這兩個類進行了重構。這樣我們就可以設計出我們自己想要的 progressBar 和 seekBar 了；另一個是感測器模組的主介面與曲線類以及調用 SO 本地方法的 Linuxc.Java 的包。

下面我會對每一個檔進行分析：

先從第一個包開始，裏面有三個 Java 檔，AbsVerticalSeekBar.java, VerticalProgressBar.java, VerticalSeekBar.java,這三個類之間有一定的繼承關係。VerticalProgressBar 是父類，AbsVerticalSeekBar 繼承之，VerticalSeekBar 又繼承 AbsVerticalSeekBar。我們先從 VerticalProgressBar 這個類來看看。

打開 VerticalProgressBar.Java，程式碼如下：

```
package tw.com.dmatek.bar;
import tw.com.dmatek.dma6410xp.R;
import android.content.Context;
import android.content.res.TypedArray;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Rect;
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.ClipDrawable;
import android.graphics.drawable.Drawable;
import android.graphics.drawable.LayerDrawable;
import android.graphics.drawable.ShapeDrawable;
import android.graphics.drawable.StateListDrawable;
import android.graphics.drawable.shapes.RoundRectShape;
import android.graphics.drawable.shapes.Shape;
import android.util.AttributeSet;
import android.view.Gravity;
import android.view.View;
import android.view.ViewDebug;
```

```
import android.view.ViewParent;
import android.widget.RemoteViews.RemoteView;
import android.os.Parcel;
import android.os.Parcelable;
/********
```

以上是所使用到的包，東西很多酒不一一說明，主要就說說

`import android.view.View;`

View是很多元件的父類，這個類主要用於顯示我們可以看到的東西，空間，下面我把android下定義如下This class represents the basic building block for user interface components。這是用戶介面元件的基礎，換句話就是使用到元件的話都一般繼續這個類，當然也有少許直接寫的**********/

```
@RemoteView
public class VerticalProgressBar extends View {
    private static final int MAX_LEVEL = 10000;

    int mMinWidth;
    int mMaxWidth;
    int mMinHeight;
    int mMaxHeight;

    private int mProgress;
    private int mSecondaryProgress;
    private int mMax;

    private Drawable mProgressDrawable;
    private Drawable mCurrentDrawable;
    Bitmap mSampleTile;
    private boolean mNoInvalidate;
```

```
    private RefreshProgressRunnable mRefreshProgressRunnable;
    private long mUiThreadId;

    private boolean mInDrawing;
    protected int mScrollX;
    protected int mScrollY;
    protected int mPaddingLeft;
    protected int mPaddingRight;
    protected int mPaddingTop;
    protected int mPaddingBottom;
    protected ViewParent mParent;

  /************以上是定義的一些變數*********/

    public VerticalProgressBar(Context context)
     {
    this(context, null);
    }

    public VerticalProgressBar(Context context, AttributeSet
attrs)
    {
        this(context, attrs, android.R.attr.progressBarStyle);
    }

    public VerticalProgressBar(Context context, AttributeSet
attrs, int defStyle)
     {
        super(context, attrs, defStyle);
```

```
        mUiThreadId = Thread.currentThread().getId();
        initProgressBar();
/*******以上是這個類的重構部分,根據不同的參數就行重構******/
        TypedArray a =
            context.obtainStyledAttributes(attrs,
R.styleable.ProgressBar, defStyle, 0);
        mNoInvalidate = true;

        Drawable drawable =
a.getDrawable(R.styleable.ProgressBar_android_progressDrawab
le);
        if (drawable != null)
{
            drawable = tileify(drawable, false);
            setProgressDrawable(drawable);
        }

        mMinWidth =
a.getDimensionPixelSize(R.styleable.ProgressBar_android_minW
idth, mMinWidth);
        mMaxWidth =
a.getDimensionPixelSize(R.styleable.ProgressBar_android_maxW
idth, mMaxWidth);
        mMinHeight =
a.getDimensionPixelSize(R.styleable.ProgressBar_android_minH
eight, mMinHeight);
        mMaxHeight =
a.getDimensionPixelSize(R.styleable.ProgressBar_android_maxH
eight, mMaxHeight);
```

```
setMax(a.getInt(R.styleable.ProgressBar_android_max, mMax));
setProgress(a.getInt(R.styleable.ProgressBar_android_progres
s, mProgress));
       setSecondaryProgress(
a.getInt(R.styleable.ProgressBar_android_secondaryProgress,
mSecondaryProgress));
       mNoInvalidate = false;
       a.recycle();
   }

   private Drawable tileify(Drawable drawable, boolean clip) {

       if (drawable instanceof LayerDrawable) {
        LayerDrawable background = (LayerDrawable) drawable;
           final int N = background.getNumberOfLayers();
           Drawable[] outDrawables = new Drawable[N];

           for (int i = 0; i < N; i++) {
               int id = background.getId(i);
               outDrawables[i] =
tileify(background.getDrawable(i),
               (id == android.R.id.progress || id ==
android.R.id.secondaryProgress));
           }

           LayerDrawable newBg = new
LayerDrawable(outDrawables);
```

```
            for (int i = 0; i < N; i++) {
                newBg.setId(i, background.getId(i));
            }
            return newBg;
        } else if (drawable instanceof StateListDrawable) {
            StateListDrawable in = (StateListDrawable) drawable;
            StateListDrawable out = new StateListDrawable();
            return out;

        } else if (drawable instanceof BitmapDrawable) {
            final Bitmap tileBitmap = ((BitmapDrawable)
drawable).getBitmap();
            if (mSampleTile == null) {
                mSampleTile = tileBitmap;
            }

            final ShapeDrawable shapeDrawable = new
ShapeDrawable(getDrawableShape());
            return (clip) ? new ClipDrawable(shapeDrawable,
Gravity.LEFT,
            ClipDrawable.HORIZONTAL) : shapeDrawable;
        }
        return drawable;
    }

    Shape getDrawableShape() {
        final float[] roundedCorners = new float[] { 5, 5, 5, 5,
5, 5, 5, 5 };
        return new RoundRectShape(roundedCorners, null, null);
```

```
}

private void initProgressBar() {
    mMax = 100;
    mProgress = 0;
    mSecondaryProgress = 0;
    mMinWidth = 24;
    mMaxWidth = 48;
    mMinHeight = 24;
    mMaxHeight = 48;
}

public Drawable getProgressDrawable() {
    return mProgressDrawable;
}

public void setProgressDrawable(Drawable d) {
    if (d != null) {
        d.setCallback(this);
        int drawableHeight = d.getMinimumHeight();
        if (mMaxHeight < drawableHeight) {
            mMaxHeight = drawableHeight;
            requestLayout();
        }
    }
    mProgressDrawable = d;
    mCurrentDrawable = d;
    postInvalidate();
}
```

```
    Drawable getCurrentDrawable() {
        return mCurrentDrawable;
    }

    @Override
    protected boolean verifyDrawable(Drawable who) {
        return who == mProgressDrawable ||
super.verifyDrawable(who);
    }

    @Override
    public void postInvalidate() {
        if (!mNoInvalidate) {
            super.postInvalidate();
        }
    }

    private class RefreshProgressRunnable implements Runnable {

        private int mId;
        private int mProgress;
        private boolean mFromUser;

        RefreshProgressRunnable(int id, int progress, boolean
fromUser) {
            mId = id;
            mProgress = progress;
            mFromUser = fromUser;
```

```
        }

        public void run() {
            doRefreshProgress(mId, mProgress, mFromUser);
            mRefreshProgressRunnable = this;
        }

        public void setup(int id, int progress, boolean fromUser)
{
            mId = id;
            mProgress = progress;
            mFromUser = fromUser;
        }
    }

    private synchronized void doRefreshProgress(int id, int
progress, boolean fromUser) {
  float scale = mMax > 0 ? (float) progress / (float) mMax : 0;
        final Drawable d = mCurrentDrawable;
        if (d != null) {
            Drawable progressDrawable = null;

            if (d instanceof LayerDrawable) {
                progressDrawable = ((LayerDrawable)
d).findDrawableByLayerId(id);
            }
            final int level = (int) (scale * MAX_LEVEL);
            (progressDrawable != null ? progressDrawable :
d).setLevel(level);
```

```
        } else {
            invalidate();
        }

        if (id == android.R.id.progress) {
            onProgressRefresh(scale, fromUser);
        }
    }

    void onProgressRefresh(float scale, boolean fromUser) {
    }

    private synchronized void refreshProgress(int id, int
progress, boolean fromUser) {
        if (mUiThreadId == Thread.currentThread().getId()) {
            doRefreshProgress(id, progress, fromUser);
        } else {
            RefreshProgressRunnable r;
            if (mRefreshProgressRunnable != null) {
                r = mRefreshProgressRunnable;
                mRefreshProgressRunnable = null;
                r.setup(id, progress, fromUser);
            } else {
                r = new RefreshProgressRunnable(id, progress,
fromUser);
            }
            post(r);
        }
    }
```

```
    public synchronized void setProgress(int progress) {
        setProgress(progress, false);
    }

    synchronized void setProgress(int progress, boolean
fromUser) {
        if (progress < 0) {
            progress = 0;
        }

        if (progress > mMax) {
            progress = mMax;
        }

        if (progress != mProgress) {
            mProgress = progress;
            refreshProgress(android.R.id.progress, mProgress,
fromUser);
        }
    }

    public synchronized void setSecondaryProgress(int
secondaryProgress) {
        if (secondaryProgress < 0) {
            secondaryProgress = 0;
        }

        if (secondaryProgress > mMax) {
```

```
            secondaryProgress = mMax;
        }

        if (secondaryProgress != mSecondaryProgress) {
            mSecondaryProgress = secondaryProgress;
            refreshProgress(android.R.id.secondaryProgress,
mSecondaryProgress, false);
        }
    }

    @ViewDebug.ExportedProperty
    public synchronized int getProgress() {
        return mProgress;
    }

    @ViewDebug.ExportedProperty
    public synchronized int getSecondaryProgress() {
        return mSecondaryProgress;
    }

    @ViewDebug.ExportedProperty
    public synchronized int getMax() {
        return mMax;
    }

    public synchronized void setMax(int max) {
        if (max < 0) {
            max = 0;
        }
```

```
        if (max != mMax) {
            mMax = max;
            postInvalidate();

            if (mProgress > max) {
                mProgress = max;
                refreshProgress(android.R.id.progress,
mProgress, false);
            }
        }
    }

    public synchronized final void incrementProgressBy(int diff)
{
        setProgress(mProgress + diff);
    }

    public synchronized final void
incrementSecondaryProgressBy(int diff) {
        setSecondaryProgress(mSecondaryProgress + diff);
    }

    @Override
    public void setVisibility(int v) {
        if (getVisibility() != v) {
            super.setVisibility(v);
        }
    }
```

```
    @Override
    public void invalidateDrawable(Drawable dr) {
        if (!mInDrawing) {
            if (verifyDrawable(dr)) {
                final Rect dirty = dr.getBounds();
                final int scrollX = mScrollX + mPaddingLeft;
                final int scrollY = mScrollY + mPaddingTop;

        invalidate(dirty.left + scrollX, dirty.top + scrollY,
        dirty.right + scrollX, dirty.bottom + scrollY);
            } else {
                super.invalidateDrawable(dr);
            }
        }
    }

    @Override
    protected void onSizeChanged(int w, int h, int oldw, int oldh)
{
        int right = w - mPaddingRight - mPaddingLeft;
        int bottom = h - mPaddingBottom - mPaddingTop;

        if (mProgressDrawable != null) {
            mProgressDrawable.setBounds(0, 0, right, bottom);
        }
    }

    @Override
    protected synchronized void onDraw(Canvas canvas) {
```

```
        super.onDraw(canvas);

        Drawable d = mCurrentDrawable;
        if (d != null) {
            canvas.save();
            canvas.translate(mPaddingLeft, mPaddingTop);
            d.draw(canvas);
            canvas.restore();
        }
    }

    @Override
    protected synchronized void onMeasure(int widthMeasureSpec,
int heightMeasureSpec) {
        Drawable d = mCurrentDrawable;
        int dw = 0;
        int dh = 0;
        if (d != null) {
            dw = Math.max(mMinWidth, Math.min(mMaxWidth,
d.getIntrinsicWidth()));
  dh = Math.max(mMinHeight, Math.min(mMaxHeight,
d.getIntrinsicHeight()));
        }
        dw += mPaddingLeft + mPaddingRight;
        dh += mPaddingTop + mPaddingBottom;

        setMeasuredDimension(resolveSize(dw,
widthMeasureSpec),
        resolveSize(dh, heightMeasureSpec));
```

```
    }

    @Override
    protected void drawableStateChanged() {
        super.drawableStateChanged();

        int[] state = getDrawableState();

        if (mProgressDrawable != null &&
mProgressDrawable.isStateful()) {
            mProgressDrawable.setState(state);
        }
    }

    static class SavedState extends BaseSavedState {
        int progress;
        int secondaryProgress;

        SavedState(Parcelable superState) {
            super(superState);
        }

        private SavedState(Parcel in) {
            super(in);
            progress = in.readInt();
            secondaryProgress = in.readInt();
        }

        @Override
```

```
        public void writeToParcel(Parcel out, int flags) {
            super.writeToParcel(out, flags);
            out.writeInt(progress);
            out.writeInt(secondaryProgress);
        }

        public static final Parcelable.Creator<SavedState>
CREATOR
                = new Parcelable.Creator<SavedState>() {
            public SavedState createFromParcel(Parcel in) {
                return new SavedState(in);
            }

            public SavedState[] newArray(int size) {
                return new SavedState[size];
            }
        };
    }

    @Override
    public Parcelable onSaveInstanceState() {
        Parcelable superState = super.onSaveInstanceState();
        SavedState ss = new SavedState(superState);

        ss.progress = mProgress;
        ss.secondaryProgress = mSecondaryProgress;
        return ss;
    }
```

```
    @Override
    public void onRestoreInstanceState(Parcelable state) {
        SavedState ss = (SavedState) state;
        super.onRestoreInstanceState(ss.getSuperState());

        setProgress(ss.progress);
        setSecondaryProgress(ss.secondaryProgress);
    }
}
/**********以上部分就是存儲初始化的第一進度和第二進度***********/
```

程式碼比較長不能一個個介紹到，有興趣的可以對 Android 下的 processBar 進行瞭解，再在只是對我們使用到的進行了重構，增加自己需要的功能。

現在我們來看看第二個 java 檔：AbsVerticalSeekBar

```
package tw.com.dmatek.bar;

import tw.com.dmatek.dma6410xp.R;
import android.content.Context;
import android.content.res.TypedArray;
import android.graphics.Canvas;
import android.graphics.Rect;
import android.graphics.drawable.Drawable;
import android.util.AttributeSet;
import android.view.KeyEvent;
import android.view.MotionEvent;
/**
這個類直接繼承了VerticalProgressBar，也就是說它已具體了
VerticalProgressBar的屬性 **/
public class AbsVerticalSeekBar extends VerticalProgressBar{
```

```
    private Drawable mThumb;
    private int mThumbOffset;
    float mTouchProgressOffset;
    boolean mIsUserSeekable = true;
    private int mKeyProgressIncrement = 1;
    private static final int NO_ALPHA = 0xFF;
    private float mDisabledAlpha;
    public AbsVerticalSeekBar(Context context) {
        super(context);
    }

    public AbsVerticalSeekBar(Context context, AttributeSet
attrs) {
        super(context, attrs);
    }

    public AbsVerticalSeekBar(Context context, AttributeSet
attrs, int defStyle) {  super(context, attrs, defStyle);

        TypedArray a = context.obtainStyledAttributes(attrs,
                R.styleable.SeekBar, defStyle, 0);
        Drawable thumb =
a.getDrawable(R.styleable.SeekBar_android_thumb);
        setThumb(thumb);
        int thumbOffset =
a.getDimensionPixelOffset(R.styleable.SeekBar_android_thumbO
ffset, getThumbOffset());
        setThumbOffset(thumbOffset);
        a.recycle();
```

```
        a = context.obtainStyledAttributes(attrs,
                R.styleable.Theme, 0, 0);
        mDisabledAlpha =
a.getFloat(R.styleable.Theme_android_disabledAlpha, 0.5f);
        a.recycle();
    }


    public void setThumb(Drawable thumb) {
        if (thumb != null) {
            thumb.setCallback(this);
            mThumbOffset = (int)thumb.getIntrinsicHeight() / 2;
        }
        mThumb = thumb;
        invalidate();
    }

    public int getThumbOffset() {
        return mThumbOffset;
    }

    public void setThumbOffset(int thumbOffset) {
        mThumbOffset = thumbOffset;
        invalidate();
    }

    public void setKeyProgressIncrement(int increment) {
        mKeyProgressIncrement = increment < 0 ? -increment :
```

```
increment;
    }

    public int getKeyProgressIncrement() {
        return mKeyProgressIncrement;
    }

    @Override
    public synchronized void setMax(int max) {
        super.setMax(max);

        if ((mKeyProgressIncrement == 0) || (getMax() /
mKeyProgressIncrement > 20)) {
            setKeyProgressIncrement(Math.max(1,
Math.round((float) getMax() / 20)));
        }
    }

    @Override
    protected boolean verifyDrawable(Drawable who) {
        return who == mThumb || super.verifyDrawable(who);
    }

    @Override
    protected void drawableStateChanged() {
        super.drawableStateChanged();

        Drawable progressDrawable = getProgressDrawable();
        if (progressDrawable != null) {
```

```
        progressDrawable.setAlpha(isEnabled() ? NO_ALPHA : (int)
(NO_ALPHA * mDisabledAlpha));
        }

        if (mThumb != null && mThumb.isStateful()) {
            int[] state = getDrawableState();
            mThumb.setState(state);
        }
    }

    @Override
    void onProgressRefresh(float scale, boolean fromUser) {
        Drawable thumb = mThumb;
        if (thumb != null) {
            setThumbPos(getHeight(), thumb, scale,
Integer.MIN_VALUE);
            invalidate();
        }
    }

    @Override
    protected void onSizeChanged(int w, int h, int oldw, int oldh)
{
        Drawable d = getCurrentDrawable();
        Drawable thumb = mThumb;
        int thumbWidth = thumb == null ? 0 :
thumb.getIntrinsicWidth();
        int trackWidth = Math.min(mMaxWidth, w - mPaddingRight
- mPaddingLeft);
```

```
       int max = getMax();
       float scale = max > 0 ? (float) getProgress() / (float)
max : 0;
     if (thumbWidth > trackWidth) {
      int gapForCenteringTrack = (thumbWidth - trackWidth) / 2;
          if (thumb != null) {
              setThumbPos(h, thumb, scale,
gapForCenteringTrack * -1);
          }
          if (d != null) {
              d.setBounds(gapForCenteringTrack, 0,
                     w - mPaddingRight - mPaddingLeft -
gapForCenteringTrack,
                     h - mPaddingBottom - mPaddingTop);
          }
       } else {
          if (d != null) {
              d.setBounds(0, 0, w - mPaddingRight - mPaddingLeft,
h - mPaddingBottom - mPaddingTop);
          }
          int gap = (trackWidth - thumbWidth) / 2;
          if (thumb != null) {
              setThumbPos(h, thumb, scale, gap);
          }
       }
   }

   private void setThumbPos(int h, Drawable thumb, float scale,
```

```
int gap) {
       int available = h - mPaddingTop - mPaddingBottom;
       int thumbWidth = thumb.getIntrinsicWidth();
       int thumbHeight = thumb.getIntrinsicHeight();
       available -= thumbHeight;
       available += mThumbOffset * 2;
       int thumbPos = (int) ((1-scale) * available);
       int leftBound, rightBound;
       if (gap == Integer.MIN_VALUE) {
        Rect oldBounds = thumb.getBounds();
           leftBound = oldBounds.left;
           rightBound = oldBounds.right;
       } else {
           leftBound = gap;
           rightBound = gap + thumbWidth;
       }
       thumb.setBounds(leftBound, thumbPos, rightBound,
thumbPos + thumbHeight);
   }

   @Override
   protected synchronized void onDraw(Canvas canvas) {
       super.onDraw(canvas);
       if (mThumb != null) {
           canvas.save();

           canvas.translate(mPaddingLeft, mPaddingTop -
mThumbOffset);
           mThumb.draw(canvas);
```

```
            canvas.restore();
        }
    }

    @Override
    protected synchronized void onMeasure(int widthMeasureSpec,
int heightMeasureSpec) {
        Drawable d = getCurrentDrawable();

        int thumbWidth = mThumb == null ? 0 :
mThumb.getIntrinsicWidth();
        int dw = 0;
        int dh = 0;
        if (d != null) {
            dw = Math.max(mMinWidth, Math.min(mMaxWidth,
d.getIntrinsicWidth()));
            dw = Math.max(thumbWidth, dh);
            dh = Math.max(mMinHeight, Math.min(mMaxHeight,
d.getIntrinsicHeight()));
        }
        dw += mPaddingLeft + mPaddingRight;
        dh += mPaddingTop + mPaddingBottom;

        setMeasuredDimension(resolveSize(dw,
widthMeasureSpec),
        resolveSize(dh, heightMeasureSpec));
    }

    @Override
```

```
public boolean onTouchEvent(MotionEvent event) {
    if (!mIsUserSeekable || !isEnabled()) {
        return false;
    }
    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            setPressed(true);
            onStartTrackingTouch();
            trackTouchEvent(event);
            break;

        case MotionEvent.ACTION_MOVE:
            trackTouchEvent(event);
            attemptClaimDrag();
            break;

        case MotionEvent.ACTION_UP:
            trackTouchEvent(event);
            onStopTrackingTouch();
            setPressed(false);
            invalidate();
            break;

        case MotionEvent.ACTION_CANCEL:
            onStopTrackingTouch();
            setPressed(false);
            invalidate();
            break;
    }
```

```
        return true;
    }

    private void trackTouchEvent(MotionEvent event) {
        final int height = getHeight();
        final int available = height - mPaddingTop -
mPaddingBottom;
        int y = height - (int)event.getY();
        float scale;
        float progress = 0;
        if (y < mPaddingBottom) {
            scale = 0.0f;
        } else if (y > height - mPaddingTop) {
            scale = 1.0f;
        } else {
            scale = (float)(y - mPaddingBottom) /
(float)available;
            progress = mTouchProgressOffset;
        }

        final int max = getMax();
        progress += scale * max;

        setProgress((int) progress, true);
    }

    private void attemptClaimDrag() {
        if (mParent != null) {
            mParent.requestDisallowInterceptTouchEvent(true);
```

```
        }
    }

    void onStartTrackingTouch() {
    }
    void onStopTrackingTouch() {
    }
    void onKeyChange() {
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        int progress = getProgress();

        switch (keyCode) {
            case KeyEvent.KEYCODE_DPAD_DOWN:
                if (progress <= 0) break;
                setProgress(progress - mKeyProgressIncrement,
true);
                onKeyChange();
                return true;

            case KeyEvent.KEYCODE_DPAD_UP:
                if (progress >= getMax()) break;
                setProgress(progress + mKeyProgressIncrement,
true);
                onKeyChange();
                return true;
        }
```

```
        return super.onKeyDown(keyCode, event);
    }

}
```

現在我們來看看第三個 java 檔：AbsVerticalSeekBar

```
package tw.com.dmatek.bar;
import android.content.Context;
import android.util.AttributeSet;

/**
 *
 *
 */
public class VerticalSeekBar extends AbsVerticalSeekBar {
public interface OnSeekBarChangeListener {
void onProgressChanged(VerticalSeekBar seekBar, int
progress,boolean fromUser);
void onStartTrackingTouch(VerticalSeekBar seekBar);
void onStopTrackingTouch(VerticalSeekBar seekBar);
    }
private OnSeekBarChangeListener mOnSeekBarChangeListener;
public VerticalSeekBar(Context context) {
this(context, null);
}
public VerticalSeekBar(Context context, AttributeSet attrs) {
this(context, attrs, android.R.attr.seekBarStyle);
}
```

```
public VerticalSeekBar(Context context, AttributeSet attrs, int
defStyle) {
super(context, attrs, defStyle);
}

@Override
void onProgressRefresh(float scale, boolean fromUser) {
super.onProgressRefresh(scale, fromUser);
if (mOnSeekBarChangeListener != null) {
mOnSeekBarChangeListener.onProgressChanged(this,
getProgress(),
fromUser);
}
}

public void
setOnSeekBarChangeListener(OnSeekBarChangeListener l) {
mOnSeekBarChangeListener = l;
}

@Override
void onStartTrackingTouch() {
if (mOnSeekBarChangeListener != null) {
mOnSeekBarChangeListener.onStartTrackingTouch(this);
}
}

@Override
```

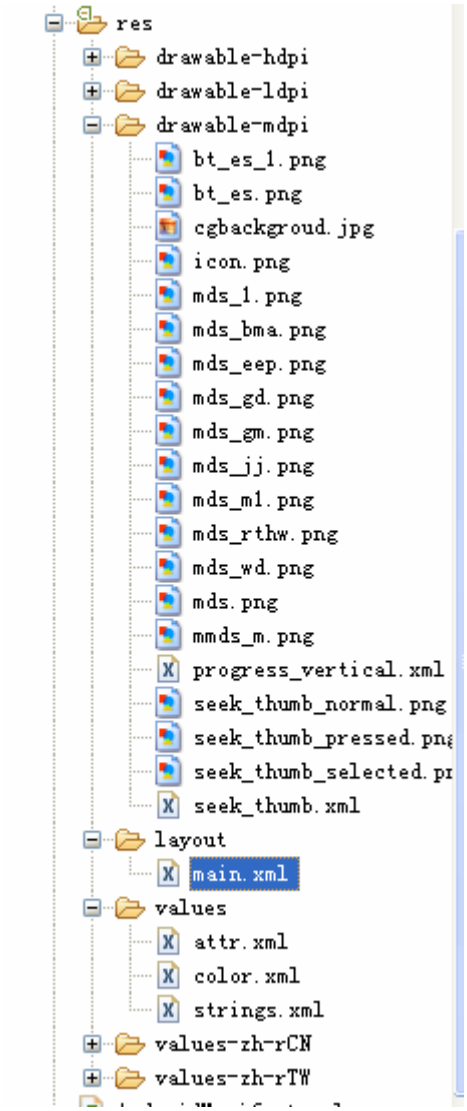
```
void onStopTrackingTouch() {
if (mOnSeekBarChangeListener != null) {
mOnSeekBarChangeListener.onStopTrackingTouch(this);
}
}

}
/****這個類又繼承了AbsVerticalSeekBar，它的功能就更加完善了，可以進行定位****/
```

到此我們就把第一個包的 java 檔介紹完了，其主要功能就是定義出我們自己想要的進度條。

現在我們來介紹我們主要使用到的另一個包。

我們先來看看最主要的類及開始執行起來的介面的 java 程式：activityMain.java, 這個程式碼我會詳細的介紹給您如何設計的，由於它繼承了 Activity 這個機動程式類，所以我們有必要先介紹下它的 XMl 的構造。

先介紹 xml 會使用到的源結構如下所示：



由於我們使用的大解析度的圖片，所以把圖片都放在了 drawable-mdpi 下面，現在讓我來說說我們這個機動程式使用到的 xml 檔 main.xml，其程式碼如下

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
```

```
android:background="@drawable/cgbackgroud"
>

<tw.com.dmatek.dma6410xp.DrawView
 android:id="@+id/drawview"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 />

 <com.clb.barapp.VerticalProgressBar
        android:id="@+id/ProgressBar01"

 android:progressDrawable="@drawable/progress_vertical"
        android:layout_height="335dip"
        android:layout_width="20dip"
        android:layout_marginLeft="581px"
        android:layout_marginTop="75px"
         />

         <TextView
  android:id="@+id/textview0"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textColor="#FFffffff"
android:textSize="25dip"
android:text="@string/app_table"
android:layout_marginLeft="200dip"
android:layout_marginTop="20dip"/>
    <TextView
```

```
 android:id="@+id/textview1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textColor="#FF00ffff"
android:textSize="25dip"
android:text="@string/mk_name"
android:layout_marginLeft="670dip"
android:layout_marginTop="100dip"/>

    <TextView
 android:id="@+id/textview2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textColor="#FFff00ff"
android:textSize="25dip"
android:layout_marginLeft="640dip"
android:layout_marginTop="160dip"/>
<TextView
 android:id="@+id/textview3"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textColor="#FFff00ff"
android:textSize="25dip"
android:drawablePadding="20px"
android:layout_marginLeft="640dip"
android:layout_marginTop="240dip"/>
 <TextView
 android:id="@+id/textview4"
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:textColor="#FFff00ff"
android:textSize="25dip"
android:layout_marginTop="320dip"
android:layout_marginLeft="640dip"
/>

 <TextView
  android:id="@+id/textview5"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textColor="#FF000000"
android:textSize="25dip"
android:layout_marginLeft="730dip"
android:layout_marginTop="160dip"/>
<TextView
 android:id="@+id/textview6"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textColor="#FF000000"
android:textSize="25dip"
android:drawablePadding="20px"
android:layout_marginLeft="730dip"
android:layout_marginTop="240dip"/>
 <TextView
 android:id="@+id/textview7"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textColor="#FF000000"
```

```
android:textSize="25dip"
android:layout_marginTop="320dip"
android:layout_marginLeft="730dip"
/>

<Button
android:id="@+id/bt_b_exit"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
android:layout_marginBottom="10dip"
 android:background="@drawable/bt_es"
android:layout_marginLeft="700dip"
android:text="@string/bt_exit"
android:textColor="@drawable/white"
/>

<Button
android:id="@+id/bt_b_start"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
android:layout_marginBottom="10dip"
android:background="@drawable/bt_es"
android:layout_marginLeft="600dip"
android:text="@string/bt_start"
android:textColor="@drawable/white"
/>
```

```
<Button
android:id="@+id/S_mk"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
android:layout_marginBottom="370dip"
android:layout_marginLeft="13dip"
android:background="@drawable/mmds_m"
android:text="@string/mk_select"
android:textColor="@drawable/blue"
android:textSize="22px"
/>

 <Button
android:id="@+id/S_gd"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
android:layout_marginBottom="320dip"
android:background="@drawable/mds"
android:layout_marginLeft="10dip"
android:text="@string/mk_gd"
android:textColor="@drawable/white"
/>
  <Button
android:id="@+id/S_gm"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
```

```
android:layout_marginBottom="280dip"
android:background="@drawable/mds"
android:layout_marginLeft="10dip"
android:text="@string/mk_gm"
android:textColor="@drawable/white"
/>
  <Button
android:id="@+id/S_wd"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
android:layout_marginBottom="240dip"
android:background="@drawable/mds"
android:layout_marginLeft="10dip"
android:text="@string/mk_wd"
android:textColor="@drawable/white"
/>

<Button
android:id="@+id/S_bma150"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
android:layout_marginBottom="200dip"
android:background="@drawable/mds"
android:layout_marginLeft="10dip"
android:text="@string/mk_bma"
android:textColor="@drawable/white"
/>
```

```
  <Button
android:id="@+id/S_ep"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
android:layout_marginBottom="160dip"
android:background="@drawable/mds"
android:layout_marginLeft="10dip"
android:text="@string/mk_eep"
android:textColor="@drawable/white"
/>
<Button
android:id="@+id/S_rt"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
android:layout_marginBottom="120dip"
 android:background="@drawable/mds"
android:layout_marginLeft="10dip"
android:text="@string/mk_rt"
android:textColor="@drawable/white"
/>

<Button
android:id="@+id/S_jj"
android:layout_alignParentBottom="true"
android:layout_width="wrap_content"
android:layout_height="40px"
```

```
    android:layout_marginBottom="80dip"
    android:layout_marginLeft="10dip"
    android:background="@drawable/mds"
    android:text="@string/mk_jj"
    android:textColor="@drawable/white"
    />
</RelativeLayout>
```

這是個佈局檔，是對我們的 activity 進行的設置，如果您設定完了，效果就是我們進入的頁面一樣了。

現在讓我們來看看 activityMain.java 文件吧。

其程式碼如下：

```
package tw.com.dmatek.dma6410xp;
import java.text.DecimalFormat;
import tw.com.dmatek.bar.VerticalProgressBar;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.TextView;
/******使用到的包，在此就不多介紹**********************/
```

```
public class activityMain extends Activity        //繼續了
Activity
{


    private Button
mBt_exit,mBt_start,mBt_mk,mBt_gd,mBt_gm,mBt_wd,mBt_BMA150,mB
t_EEPROM,mBt_rt,mBt_jj;
    private TextView
mTV_mk,mTV_7,mTV_2,mTV_3,mTV_4,mTV_5,mTV_6;
    private static int
fg=1,fg_gd=1,fg_gm=1,fg_wd=1,fg_bma150=1,fg_eeprom=1,fg_rt=1
,fg_jj=1;
    private static int fg_mk;
    private boolean pth_read1;
    private static float
v_gd,v_gm,v_wd,v_wd_x,v_bma_x,v_bma_x_sm,v_bma_y,v_bma_y_sm,
v_bma_z,v_bma_z_sm,v_eeprom,v_rt,v_jj;
    private static float
dv_gd,dv_gm,dv_wd,dv_bma_x,dv_bma_x_sm,dv_bma_y,dv_bma_y_sm,
dv_bma_z,dv_bma_z_sm,dv_eeprom,dv_rt,dv_jj;
    private String
sv_gd,sv_gm,sv_wd,sv_bma_x,sv_bma_y,sv_bma_z,sv_eeprom,sv_rt
,sv_jj;
    private int fd;
     VerticalProgressBar vProgressBar;
     DecimalFormat df;
/***************以上是進行的變數定義*************************/
     Handler myHandler = new Handler()
     {
```

```
public void handleMessage(Message msg)
{

    if(msg.what==1)
    {
          if(fg_mk==1)
          {
          mTV_2.setVisibility(TextView.VISIBLE);
          mTV_5.setVisibility(TextView.VISIBLE);
          mTV_3.setVisibility(TextView.VISIBLE);
          mTV_6.setVisibility(TextView.VISIBLE);

          mTV_2.setText(R.string.ad_v);
          mTV_5.setText(""+(int)v_gd);
          int ss=(int)((0-v_gd)*350/32000+390);
          if(Config.num1<=45)
          {
               if(Config.num1<45)
                   Config.num1++;
               for(int sss=Config.num1;sss>0;sss--)
                   Config.Y[sss]=Config.Y[sss-1];
               Config.Y[0]=ss;
          }
          dv_gd=v_gd/32000*10/3;
          vProgressBar.setProgress((int)v_gd/320+5);
          mTV_3.setText(R.string.dy_v);
          sv_gd=df.format(dv_gd).toString();
          mTV_6.setText(" "+sv_gd);
```

```
                mTV_4.setVisibility(TextView.VISIBLE);
                mTV_7.setVisibility(TextView.VISIBLE);
                mTV_4.setText(R.string.gd_v);
            mTV_7.setText("  ");
                }
                if(fg_mk==2)
            {
                mTV_2.setVisibility(TextView.VISIBLE);
            mTV_5.setVisibility(TextView.VISIBLE);
            mTV_3.setVisibility(TextView.VISIBLE);
                mTV_6.setVisibility(TextView.VISIBLE);
                mTV_2.setText(R.string.ad_v);
            mTV_5.setText(""+(int)v_gm);

                int ss=(int)((3000-v_gm)*350/30000+390);
                if(Config.num1<=45)
                {
                     if(Config.num1<45)
                         Config.num1++;
                     for(int sss=Config.num1;sss>0;sss--)
                         Config.Y[sss]=Config.Y[sss-1];
                     Config.Y[0]=ss;
                }
                dv_gm=(v_gm-3000)/30000*10/3;

vProgressBar.setProgress((int)((v_gm-3000)/300+5));
            mTV_3.setText(R.string.dy_v);
                sv_gm=df.format(dv_gm).toString();
            mTV_6.setText(" "+sv_gm);
```

```
            mTV_4.setVisibility(TextView.VISIBLE);
            mTV_7.setVisibility(TextView.VISIBLE);
            mTV_4.setText(R.string.gm_v);
            mTV_7.setText("  ");

            }
                if(fg_mk==3)
                {

                mTV_2.setVisibility(TextView.VISIBLE);
            mTV_5.setVisibility(TextView.VISIBLE);
            mTV_3.setVisibility(TextView.VISIBLE);
            mTV_6.setVisibility(TextView.VISIBLE);
            mTV_4.setVisibility(TextView.VISIBLE);
            mTV_7.setVisibility(TextView.VISIBLE);
            mTV_4.setText(R.string.wd_v);
            mTV_7.setText(""+(int)v_wd);
                vProgressBar.setProgress((int)(v_wd));
                int ss=(int)((0-v_wd)+340);
                if(Config.num1<=45)
                {
                     if(Config.num1<45)
                         Config.num1++;
                     for(int sss=Config.num1;sss>0;sss--)
                         Config.Y[sss]=Config.Y[sss-1];
                     Config.Y[0]=ss;
                }

                mTV_3.setText(R.string.dy_v);
```

```
                dv_wd=(float) (v_wd*0.001+2.69);
                sv_wd=df.format(dv_wd).toString();
                mTV_6.setText("  "+sv_wd);

                mTV_2.setText(R.string.ad_v);
            v_wd=((v_wd*10+v_wd_x)+26810)*16384/13810;
            mTV_5.setText(""+(int)v_wd);



                }
                if(fg_mk==4)
                {
            mTV_2.setVisibility(TextView.VISIBLE);
            mTV_5.setVisibility(TextView.VISIBLE);
            mTV_2.setText("    X:");
            if(v_bma_x_sm==43)
            {
            mTV_5.setText("+"+(int)(v_bma_x));
            }
            if(v_bma_x_sm==45)
            {
            mTV_5.setText("-"+(int)(v_bma_x));
            }

            mTV_3.setVisibility(TextView.VISIBLE);
            mTV_6.setVisibility(TextView.VISIBLE);
            mTV_3.setText("y:");

            if(v_bma_y_sm==43)
```

```
{
mTV_6.setText("+"+(int)v_bma_y);
}
if(v_bma_y_sm==45)
{
mTV_6.setText("-"+(int)v_bma_y);
}

mTV_4.setVisibility(TextView.VISIBLE);
mTV_7.setVisibility(TextView.VISIBLE);
mTV_4.setText("    z:");
if(v_bma_z_sm==43)
{
mTV_7.setText("+"+(int)v_bma_z);
}
if(v_bma_z_sm==45)
{
    mTV_7.setText("-"+(int)v_bma_z);
    }

    }
    if(fg_mk==5)
    {
    mTV_3.setVisibility(TextView.VISIBLE);
    mTV_6.setVisibility(TextView.VISIBLE);
    mTV_3.setText(R.string.eep_v_out);
    mTV_6.setText(" "+(int)v_eeprom);
    }
    if(fg_mk==6)
```

```
{
mTV_2.setVisibility(TextView.VISIBLE);
mTV_3.setVisibility(TextView.VISIBLE);

if(v_rt==0)
{
    mTV_2.setText("ON:");
    mTV_3.setText(R.string.rt_v_on);

}
else
{
    mTV_2.setText("OFF:");
    mTV_3.setText(R.string.rt_v_off);
}


}
if(fg_mk==7)
{
mTV_2.setVisibility(TextView.VISIBLE);
mTV_5.setVisibility(TextView.VISIBLE);
mTV_3.setVisibility(TextView.VISIBLE);
mTV_6.setVisibility(TextView.VISIBLE);
mTV_2.setText(R.string.ad_v);
mTV_5.setText(" "+(int)v_jj);

int ss=(int)((1416-v_jj)*350/24000+390);
if(Config.num1<=45)
```

```
                {
                    if(Config.num1<45)
                        Config.num1++;
                    for(int sss=Config.num1;sss>0;sss--)
                        Config.Y[sss]=Config.Y[sss-1];
                    Config.Y[0]=ss;
                }

                dv_jj=(v_jj-1416)/24000*10/3;

vProgressBar.setProgress((int)((v_jj-1416)/240+5));
             mTV_3.setText(R.string.dy_v);
             sv_jj=df.format(dv_jj).toString();
             mTV_6.setText(" "+sv_jj);
             mTV_4.setVisibility(TextView.VISIBLE);
             mTV_7.setVisibility(TextView.VISIBLE);
             mTV_4.setText(R.string.jj_v);
             mTV_7.setText("  ");
                 }
            }
            super.handleMessage(msg);
        }
     };

/****************以上是對消息進行處理**********************/
class myThread1 implements Runnable
{
    public void run()
    {
```

```
while (!Thread.currentThread().isInterrupted())
{
  Message message = new Message();
  message.what = 1;
  activityMain.this.myHandler.sendMessage(message);
  try {
    if(pth_read1)
    {
      Thread.sleep(100);
      Log.v("thread:","rrr");
      //sv_wd=Linuxc.receiveMsgSensors(3);
      v_gd=Linuxc.receiveSensors(1);
      v_gm=Linuxc.receiveSensors(2);
      v_wd=Linuxc.receiveSensors(3);
      v_wd_x=Linuxc.receiveSensors(33);

      v_bma_x=Linuxc.receiveSensors(4);
      v_bma_x_sm=Linuxc.receiveSensors(8);
      v_bma_y=Linuxc.receiveSensors(44);
      v_bma_y_sm=Linuxc.receiveSensors(88);
      v_bma_z=Linuxc.receiveSensors(444);
      v_bma_z_sm=Linuxc.receiveSensors(888);

      v_eeprom=Linuxc.receiveSensors(5);
      Log.v("thread:","run"+v_eeprom);

      v_rt=Linuxc.receiveSensors(6);
      v_jj=Linuxc.receiveSensors(7);
```

```
                Log.v("thread:","run"+v_eeprom);

              }
              else break;
            } catch (InterruptedException e)
            {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
          }
    }
}
/********************以上是對線程繼承的介面的實現
*****************/

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState); //調用父類的構造方法
        requestWindowFeature(Window.FEATURE_NO_TITLE);

    getWindow().setFlags(WindowManager.LayoutParams.FLAG_FUL
LSCREEN ,

WindowManager.LayoutParams.FLAG_FULLSCREEN);
        /***********************實現全螢幕顯示
*********************/
        setContentView(R.layout.main);  //調用佈局文件
```

```
        vProgressBar =
(VerticalProgressBar)findViewById(R.id.ProgressBar01);
        vProgressBar.setMax(100);//設置進度條的最大值
        vProgressBar.setProgress(0);//設置進度條的初始值
        df = new DecimalFormat("0.00"); //設置顯示格式
       // vProgressBar.setSecondaryProgress(90);


`
        mTV_mk=(TextView)findViewById(R.id.textview1);
        mTV_2=(TextView)findViewById(R.id.textview2);
        mTV_3=(TextView)findViewById(R.id.textview3);
        mTV_4=(TextView)findViewById(R.id.textview4);
        mTV_5=(TextView)findViewById(R.id.textview5);
        mTV_6=(TextView)findViewById(R.id.textview6);
        mTV_7=(TextView)findViewById(R.id.textview7);

        /*******************獲取 TextVIew 的資源 ID 號
****************/

        mBt_exit=(Button)findViewById(R.id.bt_b_exit);
        mBt_exit.setOnClickListener(exit_button_listener);
        mBt_start=(Button)findViewById(R.id.bt_b_start);
        mBt_start.setOnClickListener(start_button_listener);

        mBt_mk=(Button)findViewById(R.id.S_mk);
        mBt_gd=(Button)findViewById(R.id.S_gd);
        mBt_gm=(Button)findViewById(R.id.S_gm);
        mBt_wd=(Button)findViewById(R.id.S_wd);
        mBt_BMA150=(Button)findViewById(R.id.S_bma150);
```

```
        mBt_EEPROM=(Button)findViewById(R.id.S_ep);
        mBt_rt=(Button)findViewById(R.id.S_rt);
        mBt_jj=(Button)findViewById(R.id.S_jj);
      /***********設置 Button 的資源 ID 號*****************/
        mBt_gd.setVisibility(Button.INVISIBLE);
        mBt_gm.setVisibility(Button.INVISIBLE);
        mBt_wd.setVisibility(Button.INVISIBLE);
        mBt_BMA150.setVisibility(Button.INVISIBLE);
        mBt_EEPROM.setVisibility(Button.INVISIBLE);
        mBt_rt.setVisibility(Button.INVISIBLE);
        mBt_jj.setVisibility(Button.INVISIBLE);

    /***************設置 button 的可見度******************/

        mTV_2.setVisibility(TextView.INVISIBLE);
        mTV_3.setVisibility(TextView.INVISIBLE);
        mTV_4.setVisibility(TextView.INVISIBLE);
        mTV_5.setVisibility(TextView.INVISIBLE);
        mTV_6.setVisibility(TextView.INVISIBLE);
        mTV_7.setVisibility(TextView.INVISIBLE);
      /*************************設置 TextView 的可見度
***************/
        mBt_mk.setOnClickListener(mk_button_listener);
        mBt_gd.setOnClickListener(gd_button_listener);
        mBt_gm.setOnClickListener(gm_button_listener);
        mBt_wd.setOnClickListener(wd_button_listener);

mBt_BMA150.setOnClickListener(BMA150_button_listener);
```

```
mBt_EEPROM.setOnClickListener(EEPROM_button_listener);
       mBt_rt.setOnClickListener(rt_button_listener);
       mBt_jj.setOnClickListener(jj_button_listener);
/****************設置 button 的監聽事件
*****************************/
       DrawView.flags=0;//定義曲線繪製

   }
   private Button.OnClickListener exit_button_listener=new
Button.OnClickListener()
   {

       public void onClick(View v)
       {
           // TODO Auto-generated method stub
           pth_read1=false;
           fd=Linuxc.closeSensors(0);
           finish();

       }
   };

   private Button.OnClickListener start_button_listener=new
Button.OnClickListener()
   {

       public void onClick(View v)
       {
```

```
            // TODO Auto-generated method stub

             //Linuxc.setSensors(7);

    mBt_start.setBackgroundResource(R.drawable.bt_es_1);
            fd=Linuxc.openSensors(2);
           if(fd>0)
           {
              pth_read1=true;
              Linuxc.setSensors(9);
            Log.v("open is success", fd+"");
           }
           else Log.v("open is false",""+fd);
           DrawView.flags=1;
           pth_read1=true;
            new Thread(new myThread1()).start();
        }
    };
    private Button.OnClickListener mk_button_listener=new
Button.OnClickListener()
    {
        public void onClick(View v)
        {
            // TODO Auto-generated method stub

            if(fg==1)
            {
               mBt_gd.setVisibility(Button.VISIBLE);
              mBt_gm.setVisibility(Button.VISIBLE);
```

```
              mBt_wd.setVisibility(Button.VISIBLE);
              mBt_BMA150.setVisibility(Button.VISIBLE);
              mBt_EEPROM.setVisibility(Button.VISIBLE);
              mBt_rt.setVisibility(Button.VISIBLE);
              mBt_jj.setVisibility(Button.VISIBLE);

mBt_mk.setBackgroundResource(R.drawable.mds_m1);
              mBt_mk.setText("");
              fg=0;

            }
            else
            {
                mBt_gd.setVisibility(Button.INVISIBLE);
                mBt_gm.setVisibility(Button.INVISIBLE);
                mBt_wd.setVisibility(Button.INVISIBLE);
                mBt_BMA150.setVisibility(Button.INVISIBLE);
                mBt_EEPROM.setVisibility(Button.INVISIBLE);
                mBt_rt.setVisibility(Button.INVISIBLE);
                mBt_jj.setVisibility(Button.INVISIBLE);
                mTV_mk.setText(R.string.mk_name);
                mTV_2.setVisibility(TextView.INVISIBLE);
                 mTV_3.setVisibility(TextView.INVISIBLE);
                 mTV_4.setVisibility(TextView.INVISIBLE);
                 mTV_5.setVisibility(TextView.INVISIBLE);
                 mTV_6.setVisibility(TextView.INVISIBLE);
                 mTV_7.setVisibility(TextView.INVISIBLE);
                 fg=1;
               // mBt_mk.setTextColor(R.drawable.blue);
```

```
            }
        }
    };
    private Button.OnClickListener gd_button_listener=new
Button.OnClickListener()
    {

        public void onClick(View v)
        {

            // TODO Auto-generated method stub

            if(fg_gd==1)
            {

              mBt_gm.setVisibility(Button.INVISIBLE);
              mBt_wd.setVisibility(Button.INVISIBLE);
              mBt_BMA150.setVisibility(Button.INVISIBLE);
              mBt_EEPROM.setVisibility(Button.INVISIBLE);
              mBt_rt.setVisibility(Button.INVISIBLE);
              mBt_jj.setVisibility(Button.INVISIBLE);
              mTV_mk.setText(R.string.mk_gd);
              fg_mk=1;
              fg_gd=0;
              DrawView.flags=1;
              mBt_gd.setBackgroundResource(R.drawable.mds_1);
            }
            else
```

```
            {
                 mBt_gd.setVisibility(Button.VISIBLE);
                mBt_gm.setVisibility(Button.VISIBLE);
                mBt_wd.setVisibility(Button.VISIBLE);
                mBt_BMA150.setVisibility(Button.VISIBLE);
                mBt_EEPROM.setVisibility(Button.VISIBLE);
                mBt_rt.setVisibility(Button.VISIBLE);
                mBt_jj.setVisibility(Button.VISIBLE);
                mTV_mk.setText(R.string.mk_name);
                mTV_2.setVisibility(TextView.INVISIBLE);
                 mTV_3.setVisibility(TextView.INVISIBLE);
                 mTV_4.setVisibility(TextView.INVISIBLE);
                 mTV_5.setVisibility(TextView.INVISIBLE);
                 mTV_6.setVisibility(TextView.INVISIBLE);
                 mTV_7.setVisibility(TextView.INVISIBLE);
                 fg_mk=0;
                fg_gd=1;
                DrawView.flags=0;
                mBt_gd.setBackgroundResource(R.drawable.mds);

            }
        }
    };
    private Button.OnClickListener gm_button_listener=new
Button.OnClickListener()
    {

        public void onClick(View v)
        {
```

```
            if(fg_gm==1)
            {

              mBt_gd.setVisibility(Button.INVISIBLE);
              mBt_wd.setVisibility(Button.INVISIBLE);
              mBt_BMA150.setVisibility(Button.INVISIBLE);
              mBt_EEPROM.setVisibility(Button.INVISIBLE);
              mBt_rt.setVisibility(Button.INVISIBLE);
              mBt_jj.setVisibility(Button.INVISIBLE);
              mTV_mk.setText(R.string.mk_gm);
              fg_mk=2;
              fg_gm=0;
              DrawView.flags=1;
              mBt_gm.setBackgroundResource(R.drawable.mds_1);
            }
            else
            {
                mBt_gd.setVisibility(Button.VISIBLE);
                mBt_gm.setVisibility(Button.VISIBLE);
                mBt_wd.setVisibility(Button.VISIBLE);
                mBt_BMA150.setVisibility(Button.VISIBLE);
                mBt_EEPROM.setVisibility(Button.VISIBLE);
                mBt_rt.setVisibility(Button.VISIBLE);
                mBt_jj.setVisibility(Button.VISIBLE);
                mTV_mk.setText(R.string.mk_name);
                mTV_2.setVisibility(TextView.INVISIBLE);
                mTV_3.setVisibility(TextView.INVISIBLE);
                mTV_4.setVisibility(TextView.INVISIBLE);
                mTV_5.setVisibility(TextView.INVISIBLE);
```

```
                mTV_6.setVisibility(TextView.INVISIBLE);
                mTV_7.setVisibility(TextView.INVISIBLE);
                fg_mk=0;
                fg_gm=1;
                DrawView.flags=0;
                mBt_gm.setBackgroundResource(R.drawable.mds);

            }
        }
    };
    private Button.OnClickListener wd_button_listener=new
Button.OnClickListener()
    {

        public void onClick(View v)
        {
            // TODO Auto-generated method stub
            if(fg_wd==1)
            {

              mBt_gm.setVisibility(Button.INVISIBLE);
              mBt_gd.setVisibility(Button.INVISIBLE);
              mBt_BMA150.setVisibility(Button.INVISIBLE);
              mBt_EEPROM.setVisibility(Button.INVISIBLE);
              mBt_rt.setVisibility(Button.INVISIBLE);
              mBt_jj.setVisibility(Button.INVISIBLE);
              mTV_mk.setText(R.string.mk_wd);
              fg_mk=3;
              fg_wd=0;
```

```
            DrawView.flags=1;
            mBt_wd.setBackgroundResource(R.drawable.mds_1);
          }
          else
          {
              mBt_gd.setVisibility(Button.VISIBLE);
              mBt_gm.setVisibility(Button.VISIBLE);
              mBt_wd.setVisibility(Button.VISIBLE);
              mBt_BMA150.setVisibility(Button.VISIBLE);
              mBt_EEPROM.setVisibility(Button.VISIBLE);
              mBt_rt.setVisibility(Button.VISIBLE);
              mBt_jj.setVisibility(Button.VISIBLE);
              mTV_mk.setText(R.string.mk_name);
              mTV_2.setVisibility(TextView.INVISIBLE);
              mTV_3.setVisibility(TextView.INVISIBLE);
              mTV_4.setVisibility(TextView.INVISIBLE);
              mTV_5.setVisibility(TextView.INVISIBLE);
              mTV_6.setVisibility(TextView.INVISIBLE);
              mTV_7.setVisibility(TextView.INVISIBLE);
              fg_mk=0;
             fg_wd=1;
             DrawView.flags=0;
             mBt_wd.setBackgroundResource(R.drawable.mds);

          }
       }
   };
   private Button.OnClickListener BMA150_button_listener=new
Button.OnClickListener()
```

```
    {

        public void onClick(View v)
        {
            // TODO Auto-generated method stub
            if(fg_bma150==1)
            {

              mBt_gm.setVisibility(Button.INVISIBLE);
              mBt_wd.setVisibility(Button.INVISIBLE);
              mBt_gd.setVisibility(Button.INVISIBLE);
              mBt_EEPROM.setVisibility(Button.INVISIBLE);
              mBt_rt.setVisibility(Button.INVISIBLE);
              mBt_jj.setVisibility(Button.INVISIBLE);
              mTV_mk.setText(R.string.mk_bma);
              fg_mk=4;
              fg_bma150=0;
              DrawView.flags=1;

mBt_BMA150.setBackgroundResource(R.drawable.mds_1);
            }
            else
            {
                 mBt_gd.setVisibility(Button.VISIBLE);
                mBt_gm.setVisibility(Button.VISIBLE);
                mBt_wd.setVisibility(Button.VISIBLE);
                mBt_BMA150.setVisibility(Button.VISIBLE);
                mBt_EEPROM.setVisibility(Button.VISIBLE);
                mBt_rt.setVisibility(Button.VISIBLE);
```

```
                mBt_jj.setVisibility(Button.VISIBLE);
                mTV_mk.setText(R.string.mk_name);
                mTV_2.setVisibility(TextView.INVISIBLE);
                 mTV_3.setVisibility(TextView.INVISIBLE);
                 mTV_4.setVisibility(TextView.INVISIBLE);
                 mTV_5.setVisibility(TextView.INVISIBLE);
                 mTV_6.setVisibility(TextView.INVISIBLE);
                 mTV_7.setVisibility(TextView.INVISIBLE);
                 fg_mk=0;
                fg_bma150=1;
                DrawView.flags=0;

    mBt_BMA150.setBackgroundResource(R.drawable.mds);
            }
        }
    };
    private Button.OnClickListener EEPROM_button_listener=new
Button.OnClickListener()
    {

        public void onClick(View v)
        {
            // TODO Auto-generated method stub
            if(fg_eeprom==1)
            {

              mBt_gm.setVisibility(Button.INVISIBLE);
              mBt_wd.setVisibility(Button.INVISIBLE);
              mBt_BMA150.setVisibility(Button.INVISIBLE);
```

```
                mBt_gd.setVisibility(Button.INVISIBLE);
                mBt_rt.setVisibility(Button.INVISIBLE);
                mBt_jj.setVisibility(Button.INVISIBLE);
                mTV_mk.setText(R.string.mk_eep);
                fg_mk=5;
                fg_eeprom=0;
                DrawView.flags=1;

mBt_EEPROM.setBackgroundResource(R.drawable.mds_1);
            }
            else
            {
                 mBt_gd.setVisibility(Button.VISIBLE);
                mBt_gm.setVisibility(Button.VISIBLE);
                mBt_wd.setVisibility(Button.VISIBLE);
                mBt_BMA150.setVisibility(Button.VISIBLE);
                mBt_EEPROM.setVisibility(Button.VISIBLE);
                mBt_rt.setVisibility(Button.VISIBLE);
                mBt_jj.setVisibility(Button.VISIBLE);
                mTV_mk.setText(R.string.mk_name);
                mTV_2.setVisibility(TextView.INVISIBLE);
                 mTV_3.setVisibility(TextView.INVISIBLE);
                 mTV_4.setVisibility(TextView.INVISIBLE);
                 mTV_5.setVisibility(TextView.INVISIBLE);
                 mTV_6.setVisibility(TextView.INVISIBLE);
                 mTV_7.setVisibility(TextView.INVISIBLE);
                 fg_mk=0;
                fg_eeprom=1;
                DrawView.flags=0;
```

```
    mBt_EEPROM.setBackgroundResource(R.drawable.mds);

            }
        }
    };
    private Button.OnClickListener rt_button_listener=new
Button.OnClickListener()
    {

        public void onClick(View v)
        {
            // TODO Auto-generated method stub
            if(fg_rt==1)
            {
              mBt_gm.setVisibility(Button.INVISIBLE);
              mBt_wd.setVisibility(Button.INVISIBLE);
              mBt_BMA150.setVisibility(Button.INVISIBLE);
              mBt_EEPROM.setVisibility(Button.INVISIBLE);
              mBt_gd.setVisibility(Button.INVISIBLE);
              mBt_jj.setVisibility(Button.INVISIBLE);
              mTV_mk.setText(R.string.mk_rt);

              fg_mk=6;
              fg_rt=0;
              DrawView.flags=1;
              mBt_rt.setBackgroundResource(R.drawable.mds_1);
            }
            else
```

```
            {
                 mBt_gd.setVisibility(Button.VISIBLE);
                mBt_gm.setVisibility(Button.VISIBLE);
                mBt_wd.setVisibility(Button.VISIBLE);
                mBt_BMA150.setVisibility(Button.VISIBLE);
                mBt_EEPROM.setVisibility(Button.VISIBLE);
                mBt_rt.setVisibility(Button.VISIBLE);
                mBt_jj.setVisibility(Button.VISIBLE);
                mTV_mk.setText(R.string.mk_name);
                mTV_2.setVisibility(TextView.INVISIBLE);
                 mTV_3.setVisibility(TextView.INVISIBLE);
                 mTV_4.setVisibility(TextView.INVISIBLE);
                 mTV_5.setVisibility(TextView.INVISIBLE);
                 mTV_6.setVisibility(TextView.INVISIBLE);
                 mTV_7.setVisibility(TextView.INVISIBLE);
                 fg_mk=0;
                fg_rt=1;
                DrawView.flags=0;
                mBt_rt.setBackgroundResource(R.drawable.mds);

            }
        }
    };
    private Button.OnClickListener jj_button_listener=new
Button.OnClickListener()
    {

        public void onClick(View v)
        {
```

```
            // TODO Auto-generated method stub
            if(fg_jj==1)
            {

              mBt_gm.setVisibility(Button.INVISIBLE);
              mBt_wd.setVisibility(Button.INVISIBLE);
              mBt_BMA150.setVisibility(Button.INVISIBLE);
              mBt_EEPROM.setVisibility(Button.INVISIBLE);
              mBt_rt.setVisibility(Button.INVISIBLE);
              mBt_gd.setVisibility(Button.INVISIBLE);
              mTV_mk.setText(R.string.mk_jj);
              fg_mk=7;
              fg_jj=0;
              DrawView.flags=1;
              mBt_jj.setBackgroundResource(R.drawable.mds_1);

            }
            else
            {
                 mBt_gd.setVisibility(Button.VISIBLE);
                mBt_gm.setVisibility(Button.VISIBLE);
                mBt_wd.setVisibility(Button.VISIBLE);
                mBt_BMA150.setVisibility(Button.VISIBLE);
                mBt_EEPROM.setVisibility(Button.VISIBLE);
                mBt_rt.setVisibility(Button.VISIBLE);
                mBt_jj.setVisibility(Button.VISIBLE);
                mTV_mk.setText(R.string.mk_name);
                mTV_2.setVisibility(TextView.INVISIBLE);
                 mTV_3.setVisibility(TextView.INVISIBLE);
```

```
                mTV_4.setVisibility(TextView.INVISIBLE);
                mTV_5.setVisibility(TextView.INVISIBLE);
                mTV_6.setVisibility(TextView.INVISIBLE);
                mTV_7.setVisibility(TextView.INVISIBLE);
                fg_mk=0;
               fg_jj=1;
               DrawView.flags=0;
               mBt_jj.setBackgroundResource(R.drawable.mds);
           }
       }
   };
   /**********************實現 button 事件介面
**********************/
}
```

到此主介面設計就介紹完成了。

下面介紹曲線繪製使用到的 Config.java 和 drawView.java 文件。

Config.java 程式碼如下：

```
package tw.com.dmatek.dma6410xp;
public class Config {
    public static int Y[]={
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0
        };
    public static int X[]={
        130,140,150,160,170,180,190,200,
```

```
        210,220,230,240,250,260,270,280,290,300,
        310,320,330,340,350,360,370,380,390,400,
        410,420,430,440,450,460,470,480,490,500,
        510,520,530,540,550,560,570,580
        };
    public static int num1=45;
}
/*************這個是曲線繪製的初始化，包括 X 軸和 Y 軸*****************/
```

DrawView.java 程式碼如下：

```
package tw.com.dmatek.dma6410xp;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.util.AttributeSet;
import android.view.View;

public class DrawView extends View{
    private Rect  rect=new Rect();
    public DrawView(Context context) {
        super(context);
        // TODO Auto-generated constructor stub
    }
    public DrawView(Context context, AttributeSet attrs) {
        super(context, attrs);
        // TODO Auto-generated constructor stub
    }
```

```
    public DrawView(Context context, AttributeSet attrs, int
defStyle) {
        super(context, attrs, defStyle);
        // TODO Auto-generated constructor stub
    }

    protected void onDraw(Canvas canvas)
    {
        final Paint pa=new Paint();
        pa.setColor(Color.WHITE);
        rect.set(120, 70, 580, 420);
        canvas.drawRect(rect, pa);

        pa.setColor(Color.GRAY);
        canvas.drawLine(120, 140, 580, 140, pa);
        canvas.drawLine(120, 240, 580, 240, pa);
        canvas.drawLine(120, 340, 580, 340, pa);
        pa.setColor(Color.BLUE);
        pa.setStrokeWidth(4);

        if(flags==1)
        {
        for(int i=0;i<Config.num1-1;i++)
        canvas.drawLine(Config.X[i],Config.Y[i],
Config.X[i+1], Config.Y[i+1], pa);
        }
        invalidate();
    }
```

```
    public static int flags=0;
}
/***************以上是曲線繪製***********************/
```

附錄一　ZB2530-01 CC2530-ZigBee 無線通訊模組

1-1　基本介紹

ZB2530-01 CC2530 ZigBee 無線通訊模組（以下簡稱ZB2530-01 模組）的硬體爲參考TI 公司的CC2530來設計的。且下列的工具，能夠在這個板上使用，這些工具的用戶手冊可以從TI 網站下載：

1、SmartRF® Studio
(http://focus.ti.com/docs/toolsw/folders/print/smartrftm-studio.html)

2、SmartRF Flash Programmer
(http://focus.ti.com/docs/toolsw/folders/print/flash-programmer.html)

3、Texas Instruments Packet Sniffer
(http://focus.ti.com/docs/toolsw/folders/print/packet-sniffer.html?DCMP=MSP430&HQS=Other+OT+smartrfsniffer)

4、IAR-EW8051-7.51A and C-Spy debugger(www.IAR.se)

ZB2530-01 CC2530 ZigBee 模組包含若干功能和應用程式，它允許快速測試RF 介面和CC2530 的外部設備。它能作爲以下例子使用：

1、ZigBee Debugger模擬器 (SmartRF04 模擬器)。

2、模組功能：ZB2530-01 CC2530 ZigBee 包含一個debugger 介面用來線上模擬及編程。

1-2 產品內容

ZB2530-01 CC2530-ZigBee 產品包含下列東西：

- **基本配件：**

1、1 片ZB2530-01模組

2、1 條USB電源連接線

3、1 條USB資料傳輸線

4、用戶手冊（本手冊）

5、資料光碟

6、資料傳輸線（配合相應的平台，選擇其一）：

a) DB9公 To DB9母資料傳輸線：用於DMA-6410XP平台

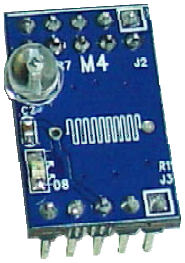
b) DB9公 To 3Pin 資料傳輸線：用於DMA-6410L平台

c) DB9公 To Mini 5Pin 資料傳輸線：用於HMI人機介面平台

7、1個ZigBee Debugger模擬器(選購品)

8、5V 2A電源 (選購品)

- **可選配件：**

1、光敏電阻感測器 	2、光電晶體感測器 
3、酒精感測器 	4、人體感應器 
5、直流馬達模組 	6、EEPROM模組 
7、類比溫度感測器 	8、Gsensor三軸位移加速感測器 

1-3 功能特色

ZB2530-01 無線通訊模組是採用 TI 最新一代CC2530 ZigBee標準晶片，適用於2.4GHz、IEEE 802.15.4、ZigBee 和 RF4CE 應用。CC2530 晶片包括了極好性能的一流 RF 收發器，工業標準增強性8051 MCU，系統中可編程的快閃記憶體，8KB RAM 以及許多其他功能強大的特性，可廣泛應用在2.4-GHz IEEE 802.15.4 系統、RF4CE控制系統、ZigBee系統，其應用領域可爲：家庭/醫院/建築物自動化，工業控制測量和監視，低功耗無線感測器網路等各方面應用。

1-4 規格特性

- 主晶片：CC2530F256，256K Flash，TI 公司最新一代 ZigBee SOC 晶片
- 輸入電壓：DC 5-12V
- 溫度範圍：-30℃ --85℃
- 串列速率：38400bps（預設），可設置 9600bps，19200bps， 38400bps，115200bps
- 無線頻率：2.4GHz
- 無線協議：ZigBee2007 /PRO
- 傳輸距離：可視距離 10 米
- 發射電流：34mA（ 最大）
- 接收電流：25mA（最大）
- 低功耗模式：可設定進入低功耗模式（與應用模式相關），2 節 AA 電池可使用半年
- 接收靈敏度：-96DBm
- 搭配平台：CE6.0 / Android2.1 / Linux2.6.X / Windows XP / Windows 7
- 尺寸大小：長 74mm × 寬 42mm

1-5 應用領域

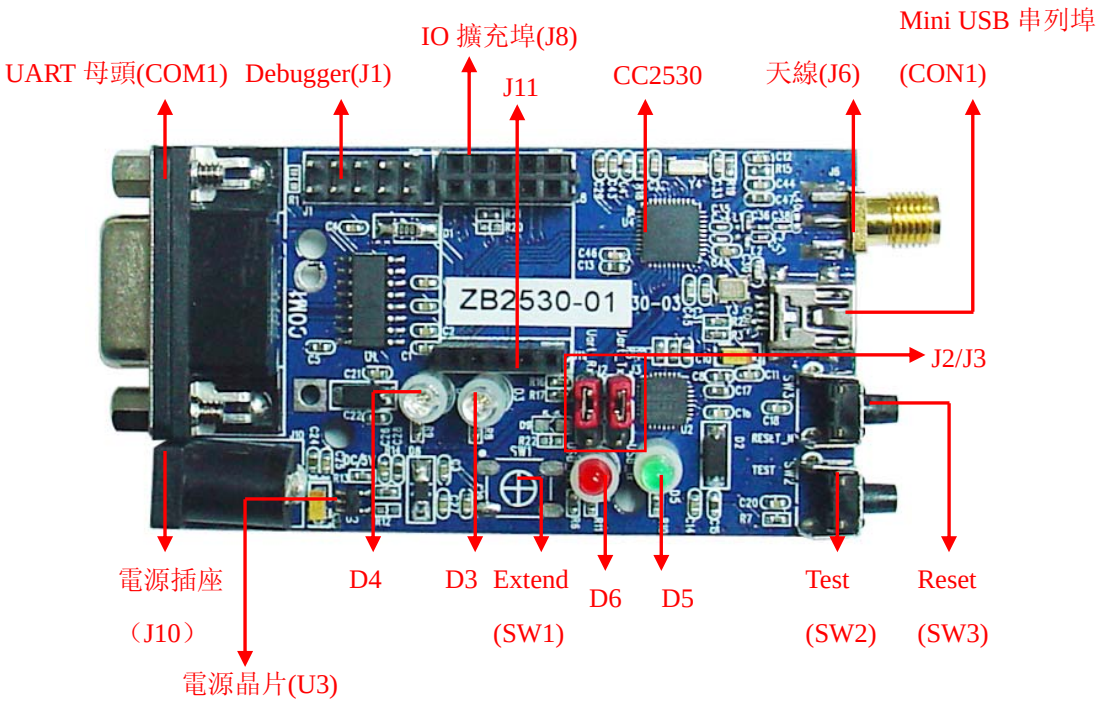
- 2.4GHz 資料傳輸應用
- RF4CE 無線遙控器
- ZigBee 無線感測網路
- 工業測量與監控
- 消費電子領域應用

1-6 準備工作

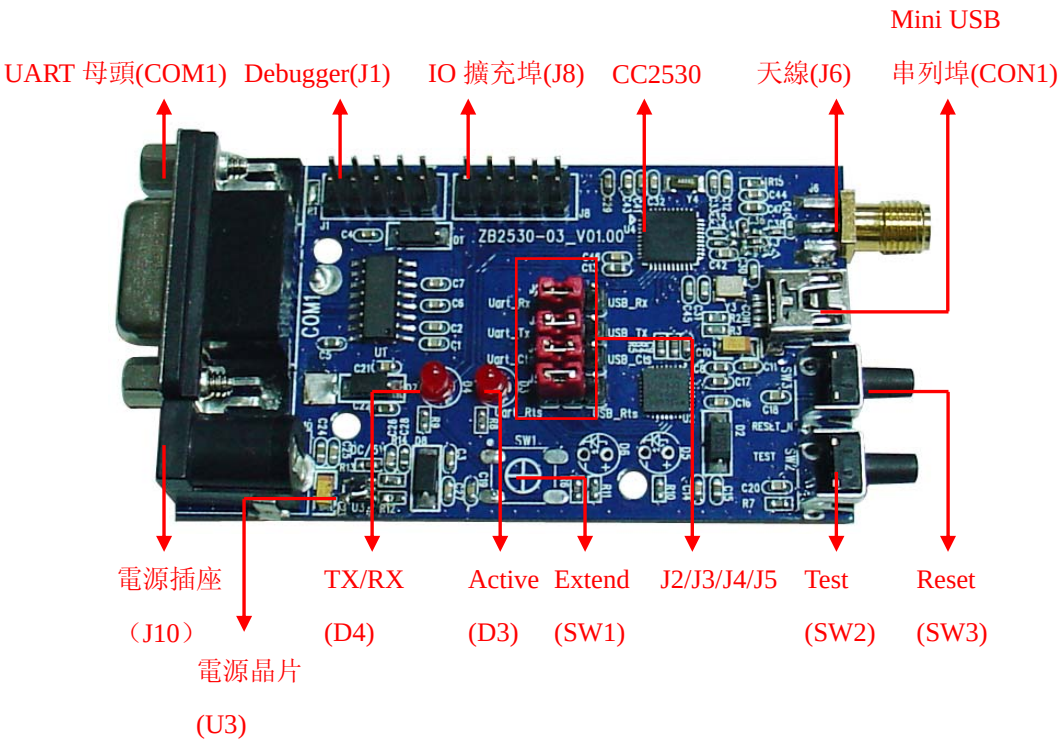
ZB2530-01在連接電腦之前要先安裝SmartRF® Studio，因爲它需要在電腦上安裝硬體驅動。SmartRF® Studio可以從TI的網站下載：

http://focus.ti.com/docs/toolsw/folders/print/smartrftm-studio.html

1-7 ZB2530-01硬體描述



ZB2530-01無線通訊模組(帶感測器)正面圖



ZB2530-01無線通訊模組(不帶感測器)正面圖

CC2530-01 ZigBee 無線通訊模組背面圖

1) ZB2530-01 ZigBee 供電電源

ZB2530-01 ZigBee供電方法有三種：

1. 外電插入：這個插座在板的上面，如上圖所示的J10，輸入電壓5V。經過板上的U3 轉3.3V後供電主板使用，這時的串列埠轉USB電路CC2530將同時工作。
2. 第二，Mini USB 供電：當在CON1插入 Mini USB線時經過CC2530將串列埠轉換成USB並提供一個3.3V/100mA電源給系統供電，這時RS232電路將同時工作。
3. 第三，電池供電：當裝上電池座J9時，兩節7 號AA電池串聯成3V給系統供電，但這時的串列埠轉USB電路CC2530將不工作。

注意：三種電源可同時存在，互不干擾。

2) 用戶介面

ZB2530-01 ZigBee 包含三個按鍵作爲用戶的輸入設備，並且有4 個LED作爲用戶輸出設備及一個天線收發信號介面。

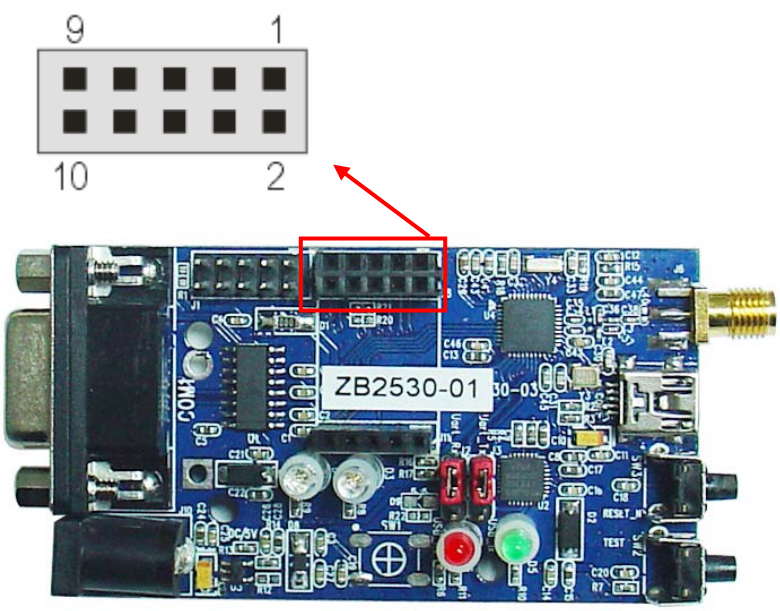
兩按鍵分別爲： SW1--爲外部擴展中斷輸入功能
SW2---爲TEST功能
SW3---爲RESET功能

兩LED燈分別爲：D3---爲ACTIVE指示
D4---爲TX/RX指示
D5，D6---爲外部擴展輸出指示功能

天線介面爲： J6---爲天線信號介面

3) I/O 連接器

I/O 連接器J8拉出CC2530 信號。透過I/O連接器可以方便訪問CC2530I/O腳。外部電路可以連接到I/O 連接器。J8連接器的接腳圖：

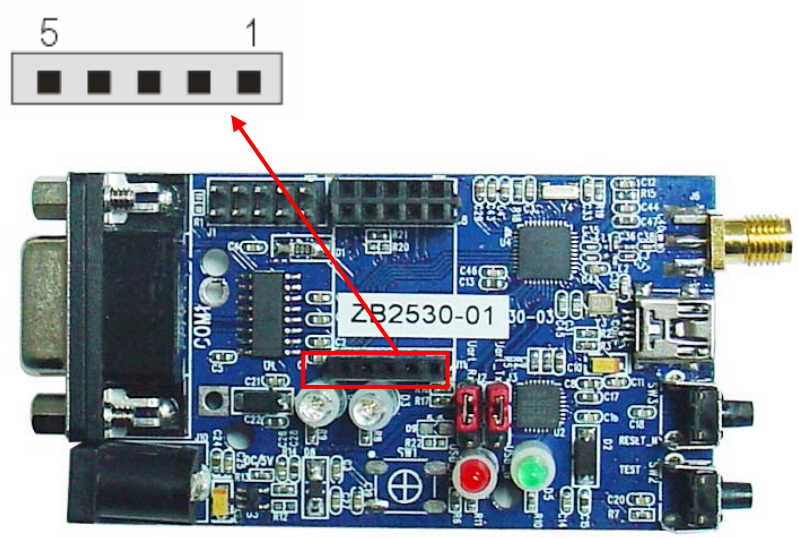


接腳定義：

接腳	功能
1	P1.7
2	P2.0
3	SDI
4	SCL
5	SDO
6	SSS
7	I2C_SCL
8	INT
9	I2C_SDA
10	ADC0

I/O 連接器J8接腳

J11 連接器的接腳圖：



接腳定義：

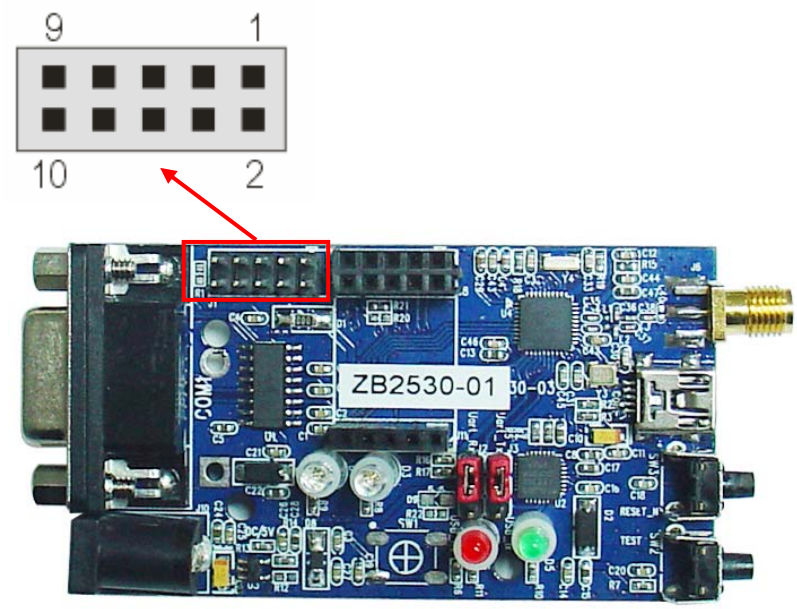
接腳	功能
1	VDD_5V
2	GND
3	VDD_3.3V
4	GND
5	GND

I/O 連接器J11接腳

4) Debugger 連接器

爲了方便用戶下載及除錯 ZB2530-01 ZigBee 模組，可透過模組上的 J1 連接器來下載及除錯。

連接器J1的接腳圖：

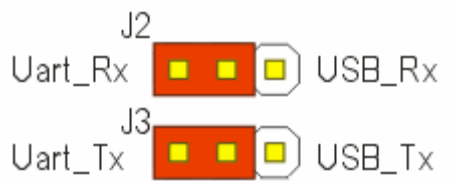


接腳定義：

接腳	功能
1	ND
2	VCC
3	DC
4.	DD
5	NC
6	NC
7	RESET_N
8	NC
9	VCC（NC）
10	NC

5) UART介面

第一，ZB2530-01 ZigBee 的UART 介面可以透過DP9母頭COM1訪問，其信號電平爲3.3V。板內帶RS232 驅動電路將TTL 電平轉成RS232 電平，將跳線帽J2、J3 插上（如下圖所示），然後連接至PC 的。



CC2530-01(帶感測器)跳線

CC2530-01(不帶感測器)跳線

DB9定義爲：

2 = TX

3 = RX

5 = GND

與其他設備連接時應該是TX 接RX，RX 接TX，GND 接GND

電腦的COM1 爲標準的針式DB9，可透過標準線連接（或者直接連接）。

串列埠的配置應該是：

數據位元 = 8

停止位 = 1

效驗位 = None

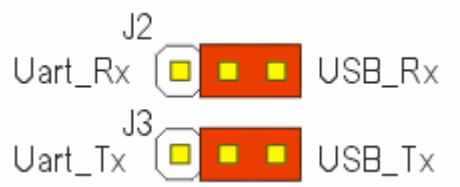
流控制 = None

具體定義可參考原理圖

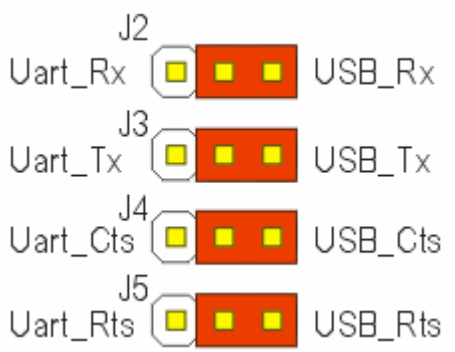
第二，ZB2530-01 ZigBee 的UART 介面可以透過Mini USB頭CON1訪問，其信號電平爲3.3V。板內透過CC2530串列埠轉USB 驅動電路，將跳線帽J2、J3插上

（如下圖所示），然後連接至PC 的USB埠。

注意：两种连接方式不可能同时存在，只能任选一种连接！



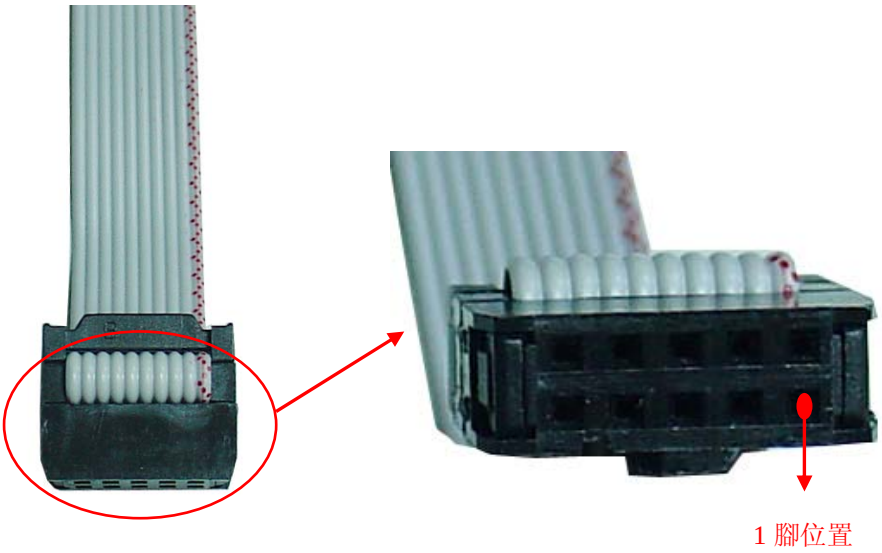
CC2530-01(帶感測器)跳線



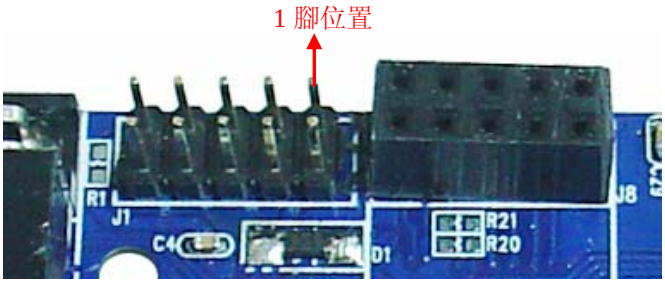
CC2530-01(不帶感測器)跳線

1-8　配合ZigBee Debugger 模擬

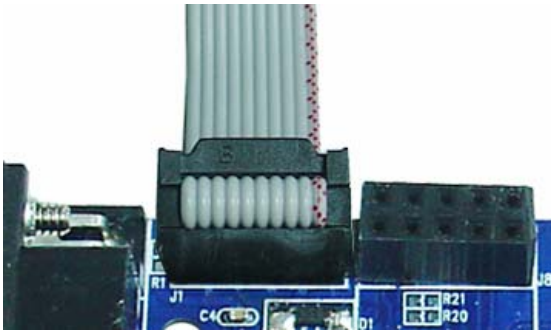
在一般情況下，模擬器連接到電腦安裝驅動後，打開SmartRF Flash Programmer 軟體，會自動辨識出模擬器，當模擬器連接到目標板後，按一下模擬器的RESET 按鈕，就會自動辨識出模擬器連接的晶片。使用這個軟體可以直接給晶片下載程式或者指定晶片的物理位址透過ZigBee Debugger模擬器對ZB2530-01 ZigBee 進行編程。如圖9所示目標板與模擬器的連接及Flash rpgramming 介面。在與ZB2530-01 ZigBee連接時，請注意接腳方向，排線的1腳要對應板上插針的1腳，如下圖所示：



排線1腳位置圖



Debugger插針1腳位置圖



CC2530-01插上排線後的圖

模擬器與目標板的連接完成後圖

a. 先按照上圖連接好連線。

b. 將 5V/2A 電源連接到 ZB2530-01 模組的電源插座（J10）處。

c. 透過 Debugger 模擬器將寫好的感測器模組的測試檔（hex檔）燒錄進CC2530晶片內，如圖所示，燒錄完成後將 Debugger 模擬器從ZB2530-01模組的 Debugger(J1)處取下。

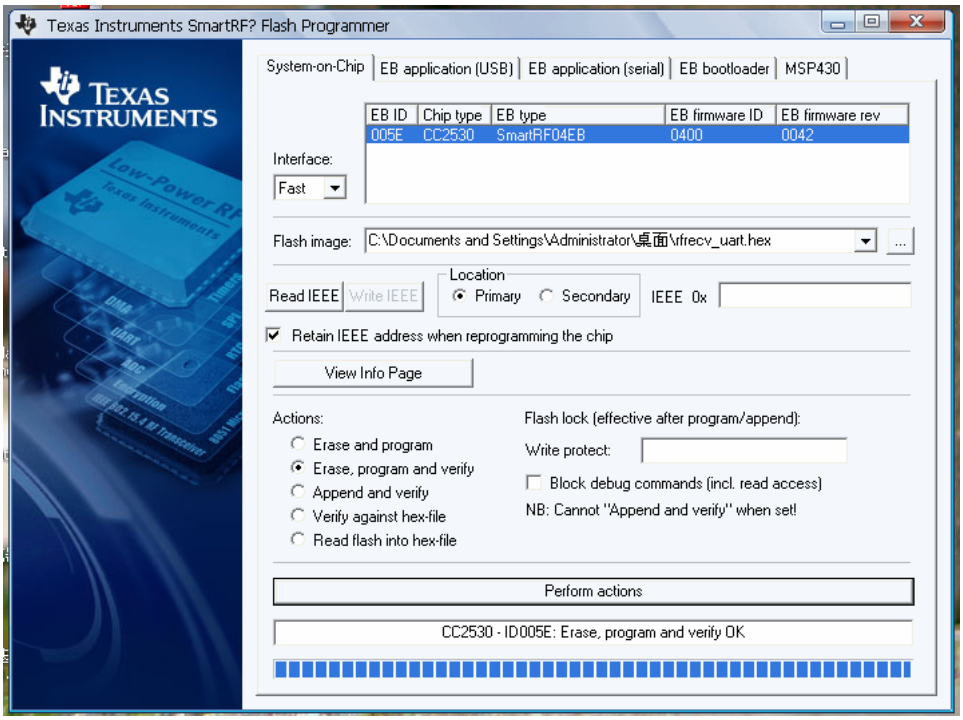


圖 Flash rpgramming 介面

接收端可以重複發送端的步驟，只需取消步驟a，並將步驟 c 的燒寫檔換成rf_send_recv.hex，最後將ZB2530-01 模組的 COM1 埠透過 RS232 有線連接到教學平台（DMA-6410XP）上，如圖。

圖　與 6410XP 連接

1-9　ZB2530-01　LED　DEMO實驗

硬體連線

將我們的教學平台上的串列埠 2 與 ZB2530_01 連接，如下圖所示，並插上電源；如下圖所示。

實驗

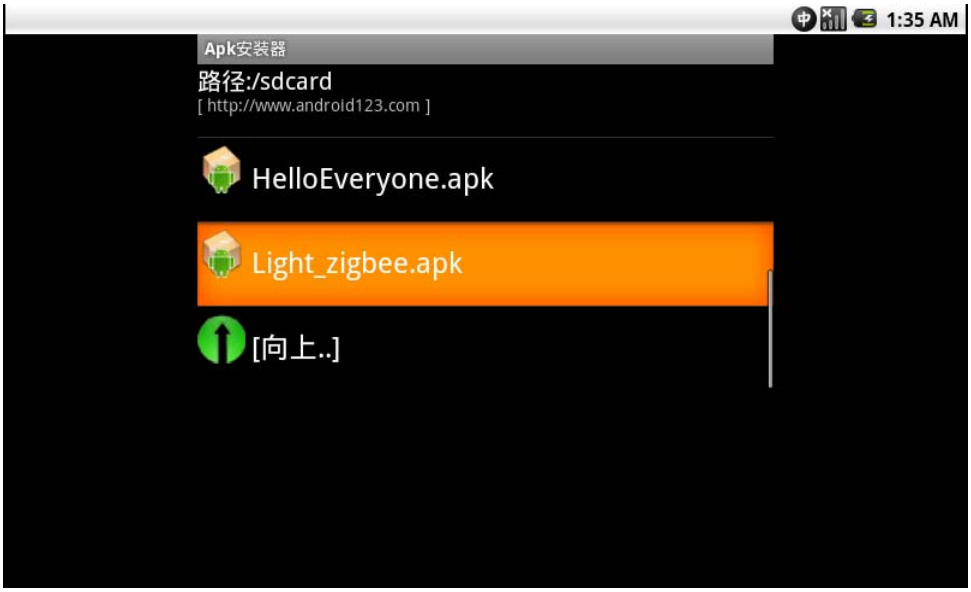
一、 使用 APK 安裝器安裝我們編寫好的程式

APK安裝器在Android系統中起到一個橋樑的作用，透過它可以把外部應用程式植入到Android系統內部，同時也可以卸載部分應用程式，大大方便了對應用程式的除錯，所以APK安裝器在Android系統中具有舉足輕重的作用。下面就介紹一下如何使用APK安裝器來安裝應用程式。

點選桌面圖示 

進入如下圖所示介面：

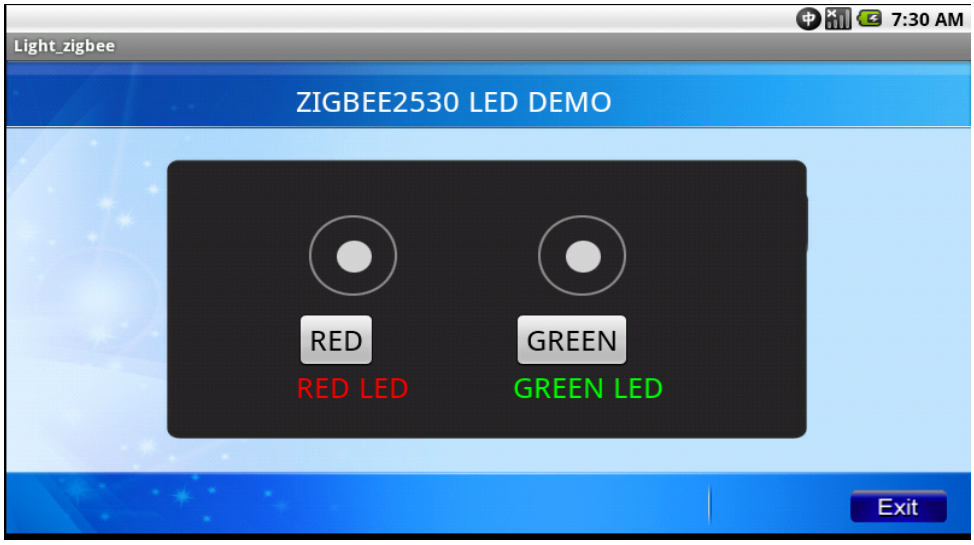
點選“安裝 Installer”將進入如下圖所示的 APK 安裝介面：



選中所需要安裝的apk檔進行安裝即可。點選“管理 Manager”按鍵進入apk卸載介面，選中要卸載的軟體，點選就可以開始卸載了。點選“退出 Exit”按鈕即可退出此應用程式。

二、進行實驗

打開安裝好的APK，如下圖：



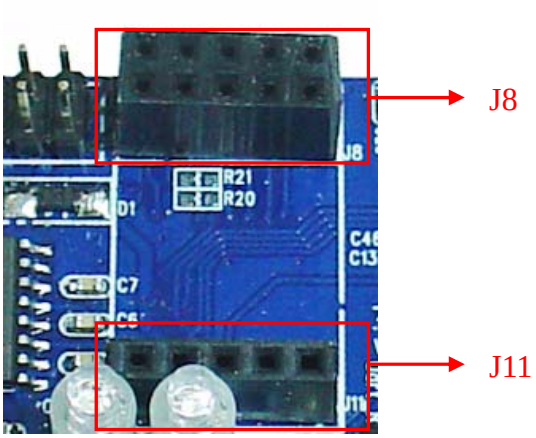
1. 我們先來測試左邊的紅燈，您點擊左邊的紅燈按鈕，這時您可以看到左邊的紅燈在不停的閃爍，同時您觀測 ZigBee 另一端的紅燈，也在不停的閃爍。再點擊時，介面的紅燈不在閃爍，ZigBee 另一端的也停止閃爍。

2. 現在測試右邊的綠燈，點擊綠燈按鈕，綠燈亮，同時另一端 ZigBee 上的綠燈也亮，再點擊綠燈，綠燈滅，同時另一端的綠燈也滅。

如此，這個實驗演示完成了。

1-10　感測器與ZB2530-01的連結

首先將感測器模組模組插入ZB2530-01 模組的 IO 擴充埠（J8）和J（11）處，如圖（11）所示。



已插上模組（酒精感測器模組）

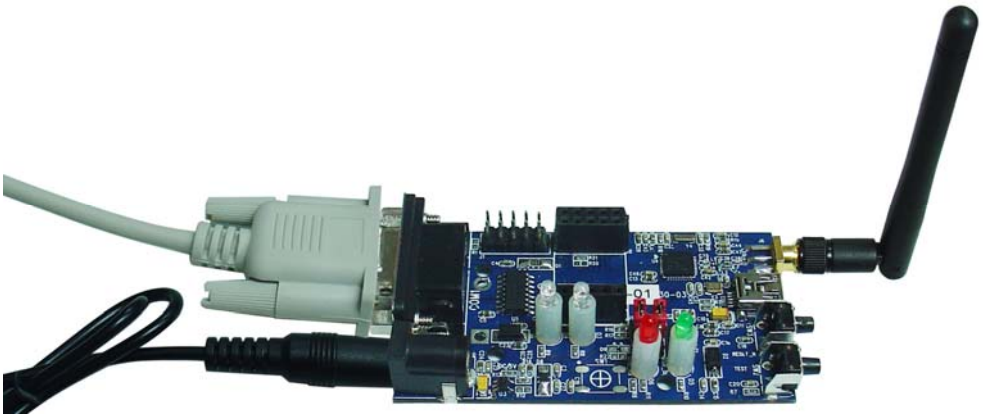
圖（11）

1-11 各平台連接方式

1. ZB2530-01接收端(主)與PC端的連接

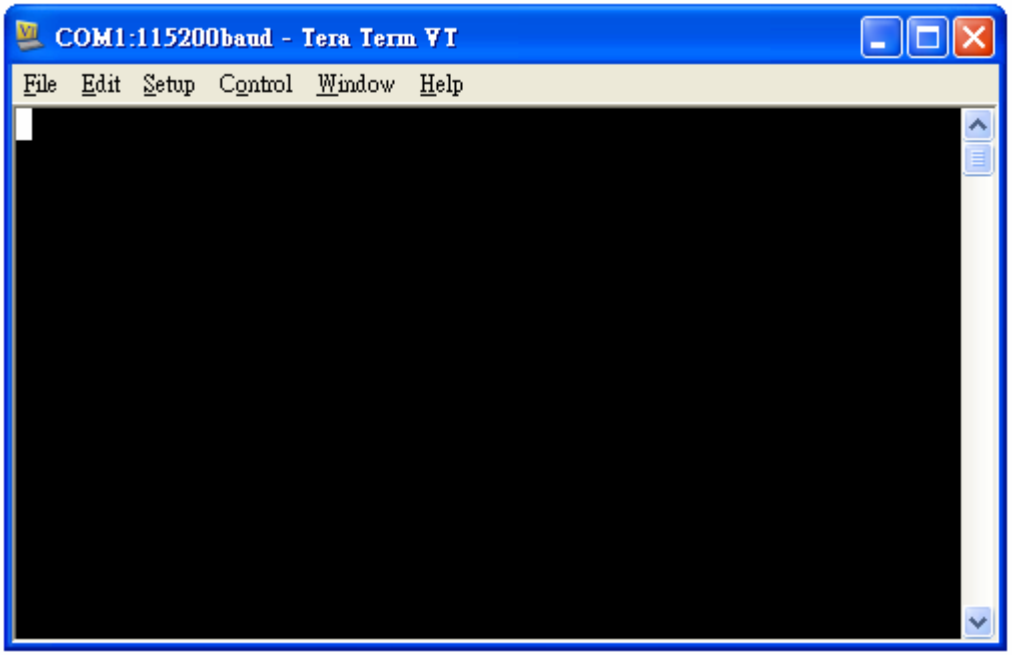
先將DB9公 TO DB9母線連接於PC及ZB2530-01接收端(主)裝置

並將ZB2530-01接收端裝（主）置接上電源

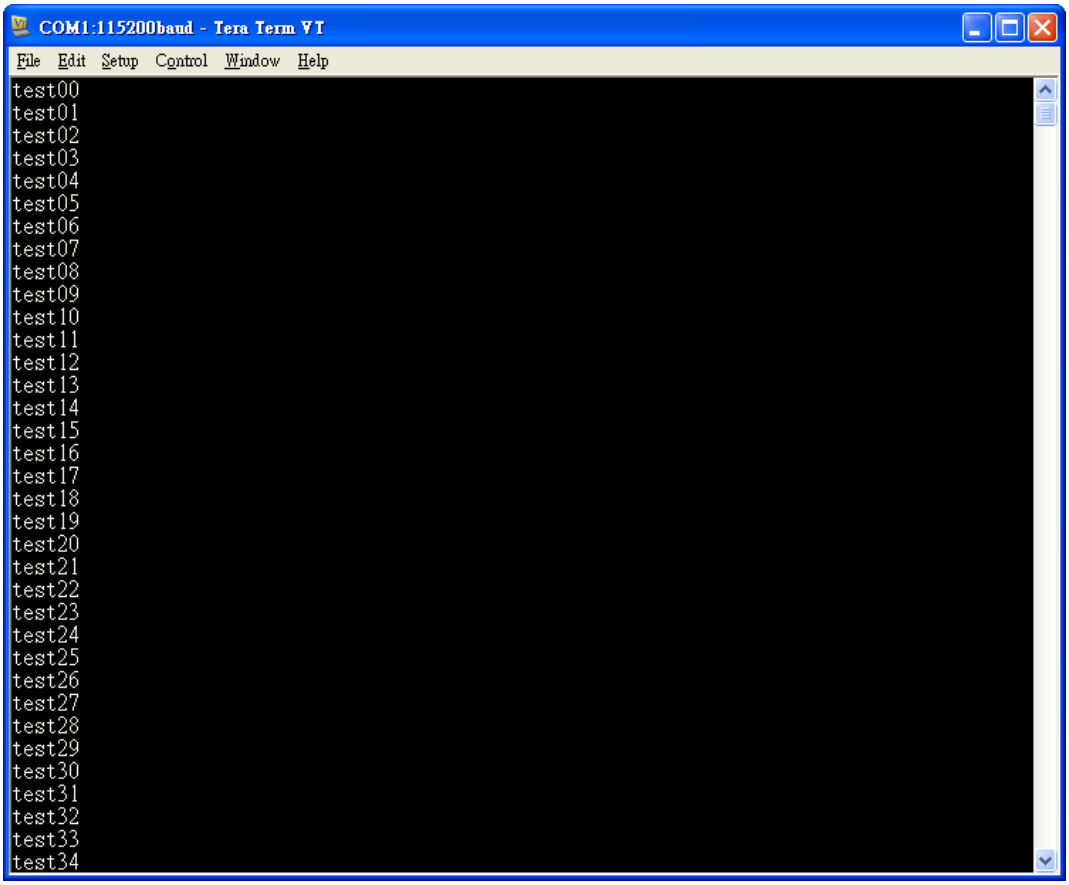


在PC端開啓終端機軟體並開啓PC端相對應COM Port

鮑率(Baud Rate) 設定爲 115200



此時在將 ZB2530-01 發射端(從)接上電源



2. ZB2530-01模組接收端（主）與 DMA-6410L 開發平台連接：

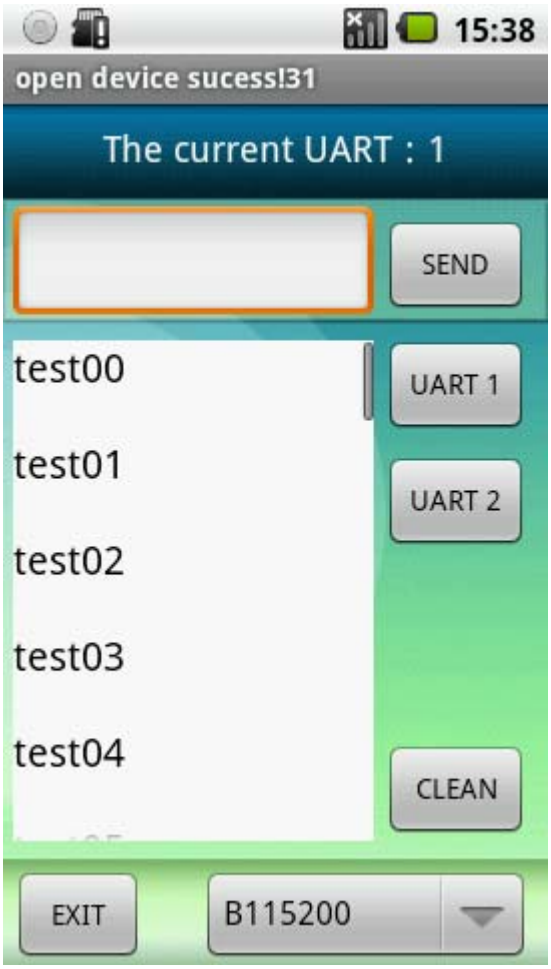
ZB2530-01 接收端（主）與 DMA-6410L 開發平台連接圖

接著執行 DMATEK_TEST 程式

接著選取程式內的 UART 程式

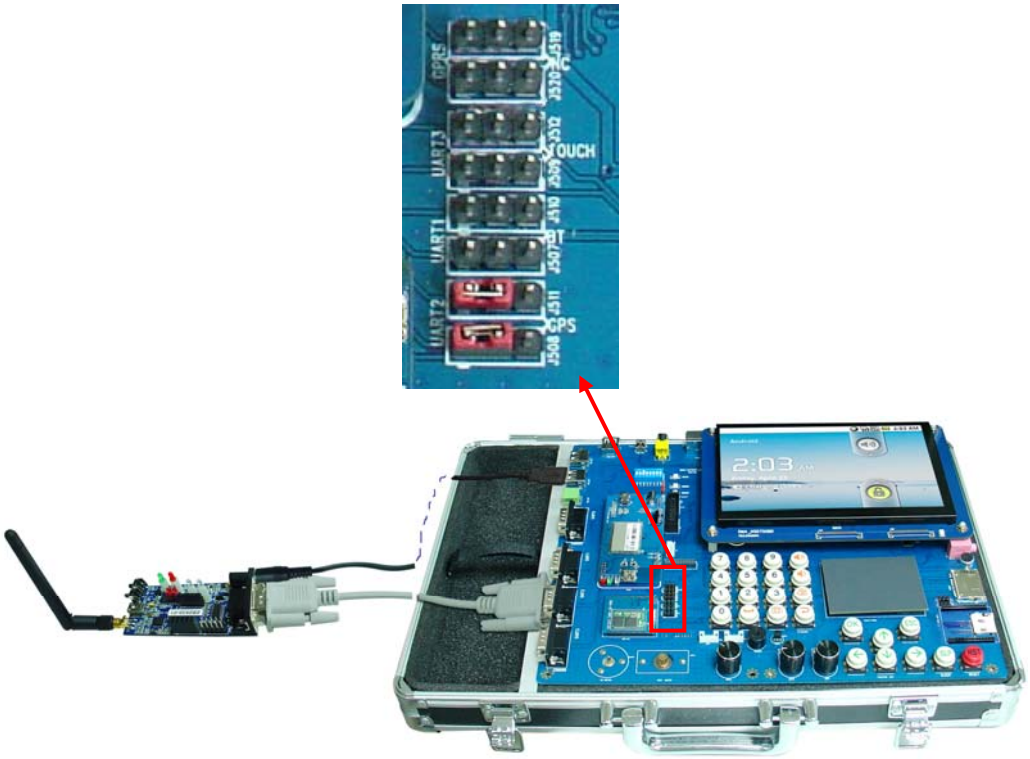
如果 ZB2530-01 模組接收端(主)接於 6410L 開發平台 UART 1 則點選 UART 1，UART 2 則點選 UART 2，鮑率(Baud Rate) 設定爲 115200

設定完成後將 ZB2530-01 模組發射端（從）接上電源或電池，6410L 將會自動收到 ZB2530-01 模組發射端（從）發出的訊息

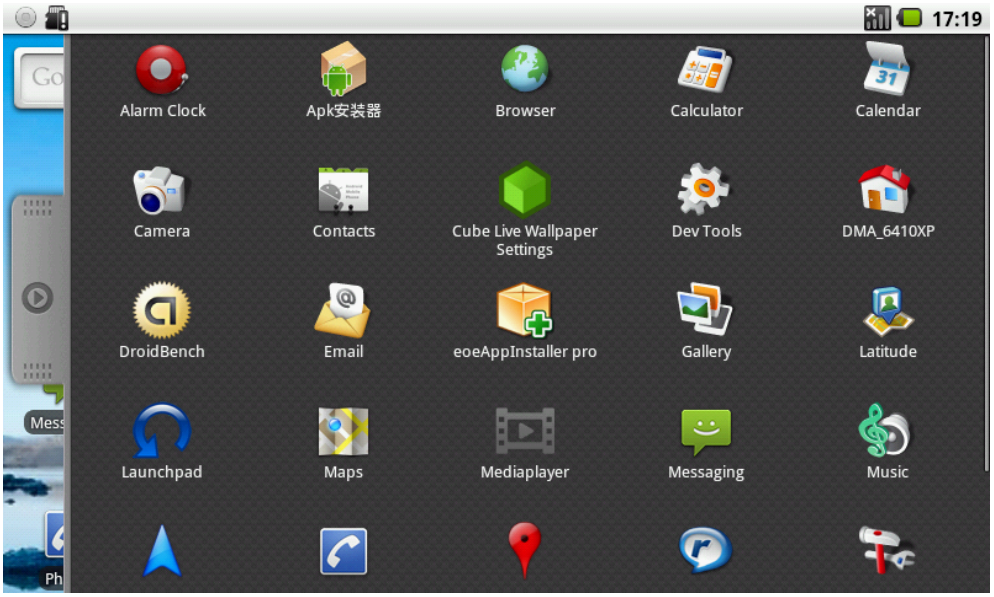


3. ZB2530-01 模組接收端（主）與 DMA-6410XP 教學平台連接

先將 ZB2530-01 模組接收端（主）與 DMA-6410XP 教學平台連接至 UART2，首先請注意 DMA-6410XP 串列埠 2 跳線地設定如圖

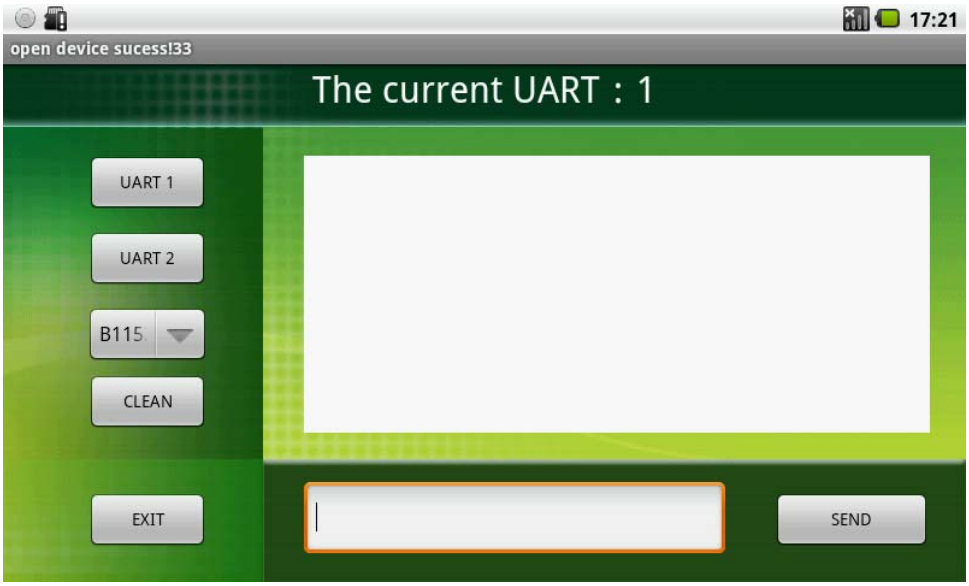


執行 6410XP 程式 DMA_6410XP 測試程式

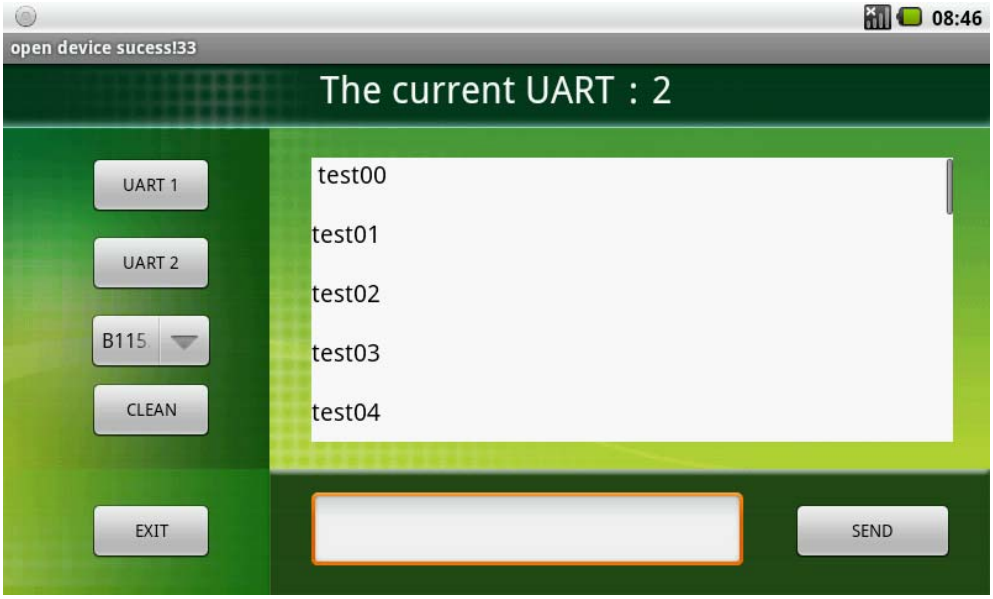


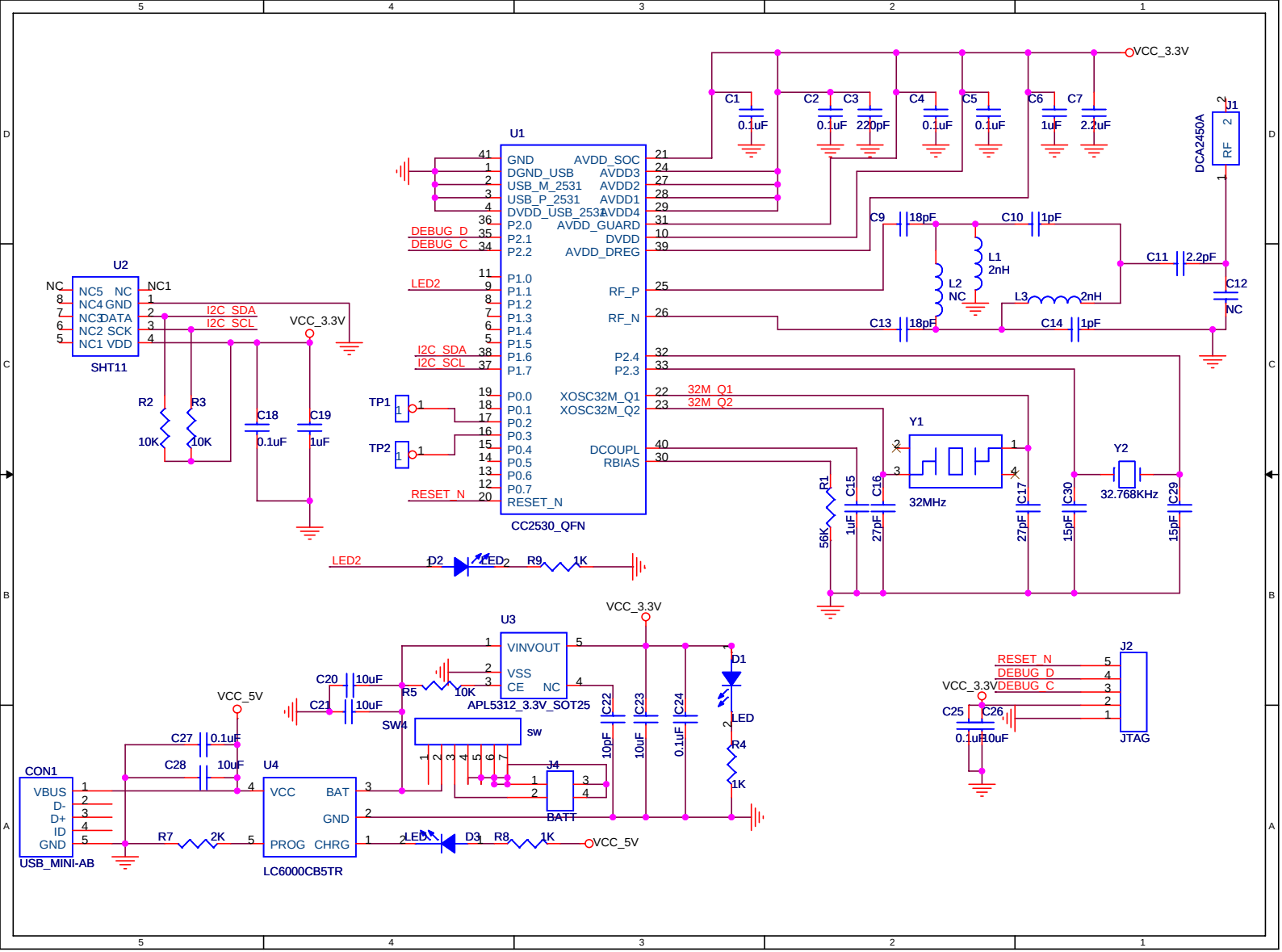
選取裡面 UART 測試程式

選取 UART 2 請不要選取 UART1，因爲會跟 6410XP 教學平台上的 GPRS 模組相衝突，鮑率(Baud Rate) 設定爲 115200



接著將 ZB2530-01 模組發射端(從)接上電源，就可從畫面上看到從 ZB2530-01 模組發射端(從)傳送給 ZB2530-01 模組接收端(主)傳至 6410XP 教學平台 UART 2 的訊息。




VCC_3.3V
U1
CC2530_QFN
GND
DGND_USB
USB_M_2531
USB_P_2531
DVDD_USB_2531
P2.0
P2.1
P2.2
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
RESET_N
AVDD_SOC
AVDD3
AVDD2
AVDD1
AVDD4
AVDD_GUARD
DVDD
AVDD_DREG
RF_P
RF_N
P2.4
P2.3
XOSC32M_Q1
XOSC32M_Q2
DCOUPL
RBIAS
DEBUG_D
DEBUG_C
LED2
I2C_SDA
I2C_SCL
32M_Q1
32M_Q2
C1 0.1uF
C2 0.1uF
C3 220pF
C4 0.1uF
C5 0.1uF
C6 1uF
C7 2.2uF
C9 18pF
C10 1pF
C11 2.2pF
C12 NC
C13 18pF
C14 1pF
L1 2nH
L2 NC
L3 2nH
J1
DCA2450A
RF
U2
SHT11
NC5 NC
NC4 GND
NC3 DATA
NC2 SCK
NC1 VDD
R2 10K
R3 10K
C18 0.1uF
C19 1uF
TP1
TP2
Y1
32MHz
Y2
32.768KHz
R1 56K
C15 1uF
C16 27pF
C17 27pF
C30 15pF
C29 15pF
D2 LED
R9 1K
U3
APL5312_3.3V_SOT25
VINVOUT
VSS
CE
NC
C20 10uF
C21 10uF
R5 10K
SW4
SW
C22 10pF
C23 10uF
C24 0.1uF
D1 LED
R4 1K
J4
BATT
J2
JTAG
RESET_N
DEBUG_D
DEBUG_C
C25 0.1uF
C26 10uF
VCC_5V
CON1
USB_MINI-AB
VBUS
D-
D+
ID
GND
C27 0.1uF
C28 10uF
U4
LC6000CB5TR
VCC
BAT
GND
PROG
CHRG
R7 2K
LED
D3
R8 1K

聚合物锂电池产品规格、承认书

Lithium-ion Polymer Battery Specifications 、 Approval Sheet

客户名称	
Company Name	
产品型号 Production Model	PL502030 3.7V
电池图纸 Battery Drawing	
制 订 Draft	JACK Tang
批 准 Approved By	William HE

客户确认	签名
Customer Approval	盖章
承认日期	

* 样品测试合格后，请将此页与最后一页图纸盖章签字回传*

Product Specifications Approval Sheet

Revision History

修 订 记 录

Date	Description	Item
日期	描　　述	更改内容
2005-8-9	First Issue (第一版)	

Product Specifications Approval Sheet

Contents
内容

Product Specifications Approval Sheet

1. Scope: 适用范围

This document describes the Product Specification of Chargeable Lithium-ion Polymer Battery produced by production company。

本规格说明书适用于本公司生产的可充聚合物锂离子电芯。

2. System: Rechargeable Lithium-ion Polymer Battery

化学系统: 可充电聚合物锂离子电池

3. Model： PL502030

电池型号： PL502030

4.Specification:

产品规格参数：

No.	Item	Spec	Note
1	Model 型号	502030	single cell 单电芯
2	Charge Voltage 充电电压	4.2V±0.05V	single cell 单电芯
3	Nominal Voltage 标称电压	3.7V	single cell 单电芯
4	Nominal Capacity 额定容量	250mAh at 0.2C Discharge(放电)	Nominal Capacity refer to the capacity of 0.2C discharge with 3.0V cut-off voltage, after charging with standard method. 额定容量指的是用标准方法充电后，用0.2C电流放电至2.75V 的容量。
	Min Capacity 最小容量	240mAh	single cell 单电芯
5	Max.Charging Current 最大充电电流	1C	240mA
6	Charging Method: 充电方法:	CC CV 恒流恒压	Constant Current Constant Voltage
	Standard Charge(0.2C) 标准充电(0.2C)	0.2C (constant current) charge to 4.20V, then 4.2V (constant voltage) for 3.5hr or (0.02C) cut off 以0.2C的恒定电流充到4.2V，然后以4.2V的恒定电压充电,共充3.5小时或截止电流为(0.02C)	
	Quick Charge(1.0C) 快速充电(1.0C)	1C (constant current) charge to 4.20V, then 4.2V (constant voltage) for 3.0hr or 0.02C cut off 以1C的恒定电流充到4.2V，然后以4.2V的恒定电压充电,共充3.0小时或截止电流为(0.02C)	
7	Standard Charge 标准放电	0.2C	Constant discharge at 0.2C to cut-off voltage of 3.0V. 0.2C 电流恒流放电至3.0V

	Max.Discharge Current 最大放电电流	1C	240mAh No Use （不建议使用）
8	Discharge Cut-off Voltage 放电的终止电压	2.75V	single cell 单电芯
9	Cell Impedance 电芯内阻	≤160mΩ	single cell 单电芯
10	Cell Weight 电芯重量	5.5±2g	single cell 单电芯
11	Open Cell Voltage 电芯电压	3.75~3.95V(Before shipping) 出货状态：3.75~3.95V	single cell 单电芯
12	Delivery Condition 出货状态	about 50% charged 大约50%容量	single cell 单电芯
13	Cycle Life 循环寿命	After 300 cycles, The capacity ≥ 80% After 500 cycles, The capacity ≥ 60%	
14	Operating Temperature 工作温度	Discharge:（放电） -20℃ - +60℃	
		Charge:（充电） 0 ℃ - +45℃	
15	* Storage in 50% charged state (cell) *		
	Storage Temperature 储存温度	-20°C - +60°C (for less than one month, 一个月内)	
		-20°C - +45°C (for less than three months, 三个月内)	
		-20°C - +25°C (for less than six months, 六个月内)	
16	Battery Dimension 电池尺寸	Thickness厚度:5.00mm(Max)	Measured weight of 300gf at 25℃±5℃. Not including Tabs 测量时测量仪器作用于电池上的压力为300gf，温度25℃±5℃，不包括极耳。
		Width宽度: 21.00mm(Max)	
		Length长度: 32.00mm(Max)	

5.Battery Cell Performance Criteria 电芯性能及测试

5.1 Visual Inspection 电池外观

There shall be no such defects as remarkable scratches, cracks,leakage or deformations.

电池外观应无明显损伤、变形、漏液或破裂。

5.2 Standard testing environment 标准测试条件及环境

Test new cells within one month after shipment from our factory and the cells shall not be cycled over five times before the tests. All the tests in this specification shall be conducted in anambient temperature of 25oC ±5°C under a humidity of 65 ± 20%RH, unless otherwise specified.

测试电池必须是本公司出厂时间不超过一个月，且电池未进行过五次以上充放电循环除非另有规定，本规格书中各项试验应在标准大气条件下进行：温度：25℃±5℃；相对湿度：65 ± 20%RH。

5.3 Measuring Instrument or Apparatus 测试设备要求

5.3.1 The measurement instrument has been certified by a qualified source.

所有测试（量）设备、仪器需经检定机构检验合格。

5.3.2 The principle internal resistance is 1KHz LCR; the accuracy is 0.2%.

内阻测试仪测量原理应为交流阻抗法（1KHz LCR）,精确度0.2%。

5.3.3 The accuracy of the measuring instrument is less than 0.01mm

测量尺寸的仪器精确度不大于0.01mm。

5.3.4 The accuracy of multimeter is at least 0.5%.While measuring the voltage,the internal resistance can not be less than 10KΩ.

万用表测量电压及电流的准确度应不低于0.5级，测电压时内阻不应小于10k Ω/v 。

5.3.5 The accuracy of the thermometer is at least ±0.5°C.

温度测量的仪表准确度应不低于±0.5℃.

5.3.6 The current measurement shall be implemented by instrument with equal to more precision scale of ±0.1% and the constant voltage precision should be implemented with ±0.5%, and the timing precision should be not below ±0.1%.

电池测试系统的电流精度应在±0.1%以上,恒压精度±0.5%,计时精度不低于±0.1%.

5.3.7 The internal resistance can vary based upon temperature and the charging mode. It is relevant to the PTC and the length and resistance of the wiring.

内阻不是恒定值，会随着温度及充电状态的饱和度变化而变化。与是否装有PTC保护元件及引线长度、容量有关。

6. Electrical Characteristics 电气特性

6.1 High Rate Discharge Capacity (1C) 高倍率放电容量(1C)

85% (minimum) of Rated Capacity 85% (最小值) 的额定容量

The capacity shall be measured at a discharge current of 1.0C and a cut-off voltage of 2.75V after the standard charge。

电池按 标准充电 方法规定充电结束后，以 1.0C 电流放电到终止电压2.75V时的测量值。

6.2 Low Temperature Discharge Capacity (0°C) 80% (minimum) of Rated Capacity

低温放电容量(0°C) 80% (最小值) 的额定容量

The capacity shall be measured at a discharge current of 0.2C in an ambient temperature of 0°C ±2°C and a cut-off voltage of 2.75V after the standard charge

电池按 标准充电 方式充电结束后，在环境温度为0°C ±2°C 条件下，以 0.2C 电流放电到终止电压2.75V时的测量值。

6.3 Low Temperature Discharge Capacity (-10°C) 70% (minimum) of Rated Capacity

低温放电容量(-10°C) 70% (最小值) 的额定容量

The capacity shall be measured at a discharge current of (0.2C) in an ambient temperature of -10°C ±2°Cand a cut-off voltage of 2.75V after the standard charge

电池按 标准充电 方式充电结束后，在环境温度为-10°C ±2°C条件下，以 0.2C 电流放电到终电压2.75V时的测量值。

6.4 High Temperature Discharge Capacity (60°C) 100% (minimum) of Rated Capacity

高温放电容量(60°C) 100% (最小值) 的额定容量

The capacity shall be measured at a discharge current of (0.2C) in an ambient temperature of 60°C ±2°Cand a cut-off voltage of 2.75V after the standard charge

电池按 标准充电 方式充电结束后，在环境温度为60°C ±2°C条件下，以 0.2C 电流放电到终电压2.75V时的测量值。

6.5 Storage Characteristics 储存特性(25°C)

Capacity Retention 85% (minimum) of Rated Capacity

容量保持量 85% (最小值)的额定容量

Capacity Recovery:90% (minimum) of Rated Capacity (最小值) 的额定容量

容量恢复量 90% (最小值) 的额定容量

The capacity retention shall be measured at a discharge current of 0.2C and a cut-off voltage of 2.75V after standard charge and being stored for 30 days at 25oC ±5°C. the capacity recovery shall be measured at a discharge current of 0.2C and cut-off voltage of 2.75V after standard charge

容量保持量应在电池 标准充电 方式充电结束后，在环境温度为25°C ±5°C 条件下，将电池开路搁置30天，再以0.2C 电流进行放电到终止电压2.75V 时测量；而容量恢复量，在测量容量保持量后按 标准充电 方式充电结束后，以 0.2C 电流进行放电到终止电压2.75V 时测量。

6.6 Storage Characteristics 储存特性(45°C)

Capacity Retention 60% (minimum) of Rated Capacity

容量保持量 60% (最小值)的额定容量

Capacity Recovery:70% (minimum) of Rated Capacity (最小值) 的额定容量

容量恢复量 70% (最小值) 的额定容量

The capacity retention shall be measured at a discharge current of 0.2C and a cut-off voltage of 2.75V after standard charge and being stored for 30 days at 45℃ ±5℃. The capacity recovery shall be measured at a discharge current of 0.2C and cut-off voltage of 2.75V after standard charge

容量保持量应在电池 标准充电 方式充电结束后，在环境温度为45℃ ±5℃ 条件下，将电池开路搁置30天，再以0.2C 电流进行放电到终止电压2.75V 时测量；而容量恢复量，在测量容量保持量后按 标准充电 方式充电结束后，以 0.2C 电流进行放电到终止电压2.75V 时测量。

7.Mechanical Performance 机械性能

7.1 Vibration Test Test 95% (min) of Rated Capacity, No Leakage

振动测试 95%（最小）的额定容量，不漏液

After standard charge Standard charge,the battery is vibrated with an amplitude of 0.8mm (1.6mm total maximum excursion) for 60 minutes in three (X, Y, and Z axis) mutually perpendicular directions. The vibration is performed between 10Hz and 55Hz at a rate of 1Hz per minute. After the completion of the vibration, the capacity shall be measured at a discharge current of 0.2C and cut-off voltage of 2.75V

电池按 标准充电 方式充电结束后，电池以0.8mm 单振幅在沿X、Y、Z三个不同的方向向每个方向垂直从10Hz－55Hz循环扫频振动60min，扫频速率为1Hz /min，振动结束后，电池容量应在以 0.2C 电流放电到终止电压2.75V时测量。

8 .Safety Performance 安全性能

8.1 Short Circuit Test No Fire, No Explosion

短路测试应 不起火，不爆炸.

The charged battery is short-circuited for 1hour at 50mΩ.

用标准充电 方式充电后的电芯，用50mΩ电阻器将正负极短接1h。

8.2 Thermal Exposure Test No Fire, No Explosion

热冲击测试 不起火，不爆炸

The cell is placed in a thermal chamber.Temperature is raised to 130±2°C at the rate

of (5±2°C)/min and held for 15 minutes, then cooled to room temperature at the rate of 5±2°C/min.

电池放于热箱中，温度以（5±2℃）/min 的速率升至130±2℃并恒温15min,再以5±2℃/min降至室温

8.3 Over charge Test [With a PCM] No Fire, No Explosion

过充电测试（带保护板） 不起火，不爆炸

After standard charge, the battery shall be charged at 1C /12V for 2.5 hrs.

电池按标准充电方式充电结束后，电池以1C /12V 充电2.5小时。

9. Electricity maintenance 荷电保持

Standard charged battery sits for 30 days at ambienttemperature of 25°C±5°C, then discharged at a 0.2C current to the termination voltage.Discharge more than 85 percent of the original capacity

将标准充电后的电芯，在环境温度为25±5℃ 的条件下，将电芯开路搁置30 天,再以0.2C电流放电至终止电压。可放出初始容量的85%以上.

10.Storage and others 储存及其它事项

10.1 If the cell is to be stored for 3 months or longer it should be held in a dry and cool environment.Voltage during storage needs to me maintained between 3.6~3.9V and the storage conditions

长期贮存的电池(超过3 个月)须置于干燥凉爽处，贮存电压为3.6~3.9V 且按电池储存环境要求

10.2 Do not store batteries in a manner that allow s terminals to short circuit

电池存放应注意避免其短路。

10.3 Prohibit reversing cell polarity within a battery assembly

严禁把电池的正负极接反。

10.4 Do not open or manipulate the folded cell edge.

不要打开或任意地折叠电池的折边。

10.5 Keep away batteries from children.

请将电池放在儿童无法拿到的地方。

11 Date of Minimum Durability 保质期

Date of Minimum Durability:six months after shipment according to Storage temperature & environment

电芯出厂后在标准存储条件下保质期为六个月。

12.Handing Guide 安全使用指导

12.1 Battery Charging 电池充电

Charge the battery using the “CCCV” or constant current /constant voltage method Do Not charge the battery with a current or voltage higherthan the specified maximum value in this specifications. The absolute maximum charging voltage is 4.25V per cell.

使用“CCCV”或恒流恒压的方式充电。不要使用高于本规格书规定的最大电流或电压充电。每个电池的最大的充电电压为4.25V 。

12.2 Prohibit reverse charging of the battery. The battery mustbe connected correctly.

严禁反充电池（正负极接反)。电池应按恰当的方法连结。

12.3 Battery Discharging 电池放电

Discharge battery at the max current specified in this specification. If you plan to discharge battery ata higher current than the max current,please consult production company.

应以本规格书说明的最大电流放电，如欲以高于最大电流的电流放电,请先咨询生产公司。

12.4 Avoid discharge the battery below 2.75V for each cell.

每个电池都应避免在低于2.75V 的情况下放电。

12.5 Do not over-discharge the battery. Over-discharging can damage the performance of the battery.It should be noted that the cell would be at an over-discharged state by its self-discharge characteristics in case the cell is not used for long time.In order to prevent over-discharging,the cell shall be charged periodically to maintain between 3.7V ~4.1V.

电池长时间不使用其过放状态会通过其自放电特性显示出来，为了防止过放，电池应周期性地充电，使电池电压保持在3.7V~4.1V范围。

12.6 Operation Temperature 操作温度

The battery shall be operated (stored, charged and discharged) in the temperature specified in this specifications.

电池的贮存，充电，放电温度应遵照本规格书的规定。

12.7 Battery Short Circuit 电池短路

Do not short-circuit a battery. A short circuit can result in over-heating of the terminals and provide an ignition source.More than a momentary short circuit will generally reduce the cell or battery service life and can lead to ignition of surrounding materials or materials within the cell or battery if the seal integrity is damaged. Extended short

-circuiting creates high temperature in the cell and at the terminals. Physical contact to high temperatures can cause skin burns. In addition, extended short-circuit may cause the cell or battery to flame.

不能够让电池短路，电池短路会使其发热，严重的会导致起火。如果电池的结构被破坏,多次瞬时的短路会减少电池的服务寿命，严重的会导致电池起火。电池短路会引起电芯和线接头处温度升高,避免与之接触，以免烧伤皮肤。

13. Remarks(备注)

13.1 My company shall make no liability for problems that occur when the above specifica - tions are not followed.

如不按以上规定操作导致发生意外，与本司无关。

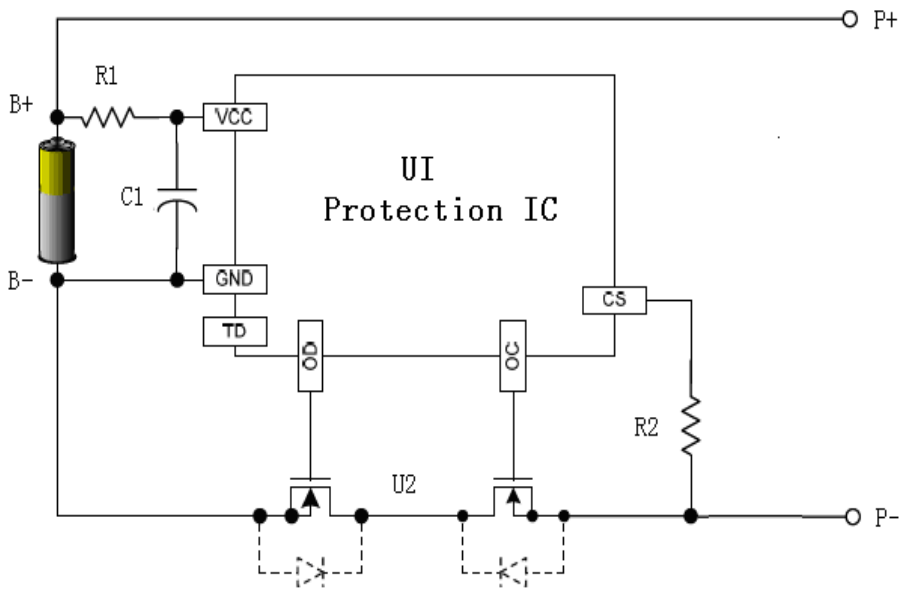
13.2 Any issues not covered in this specification should be discussed between the customer and Company .

任何本说明书中未提及的事项，须经双方协商确定。

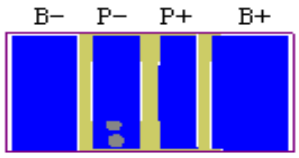
14 Protection circuit board （PCB） Specifications

保护板规格书

14.1 PCB Application drawing （保护板应用原理图）



14.2 PCB Layout (PCB图)



14.3 PCB Bom List (保护板BOM清单)

位号	器件名称	器件型号	封装	数量	备注
U1	精工锂电保护	S8261	SOT-23-6	1	
U2	MOS管	8205	SOT-23-6	1	
R1	贴片电阻	470Ω/0603/±5%	0603	1	
R2	贴片电阻	2K/0402/±5%	0603	1	
C1	贴片电容	0.1uF/0603/±20%/16V	0603	1	
		印制电路板		1	

14.4 PCM Characteristics (保护板电气特性)

项目		符号	检验方法及设备	检验标准			单位
				最小值	典型值	最大值	
过充保护	过充电检测电压	V_{DET1}	锂电保护板测试仪	4.255	4.28	4.305	V
	过充电检测延迟时间	tV_{DET1}	锂电保护板测试仪	0.96	1.2	1.4	S
	过充电解除电压	V_{REL1}	锂电保护板测试仪	4.055	4.08	4.105	V
过放保护	过放电检测电压	V_{DET2}	锂电保护板测试仪	2.95	3.00	3.05	V
	过放电检测延迟时间	tV_{DET2}	锂电保护板测试仪	115	144	173	ms
	过放电解除电压	V_{REL2}	锂电保护板测试仪	2.95	3.00	3.05	V
过流保护	过电流检测电压	V_{DET3}	锂电保护板测试仪	0.065	0.08	0.095	V
	过电流保护电流	I_{DP}	锂电保护板测试仪	1.00	2.00	3.00	A
	检测延迟时间	tV_{DET3}	锂电保护板测试仪	7.20	9.00	11.00	ms
短路保护	检测延迟时间	T_{SHORT}	锂电保护板测试仪	220	320	380	μS
	保护解除条件		万用表	断开外部短路负载或充电恢复			
内阻	主回路通态电阻	R_{DS}	锂电保护板测试仪	40.00	50.00	60.00	mΩ
消耗电流	通常工作时消耗电流	I_{DD}	锂电保护板测试仪	1.00	3.50	7.00	μA
静态电流	休眠时消耗电流	I_{PWN}	锂电保护板测试仪	* -* - *	* -* - *	0.10	μA
* -* - *							

注:以上测试环境均为25℃所测出的值,非常温下可能有所不同,该电路的工作温度范围为-40--85℃,具体测试条件及测试电路请参照保护IC之规格书。

Product Specifications Approval Sheet

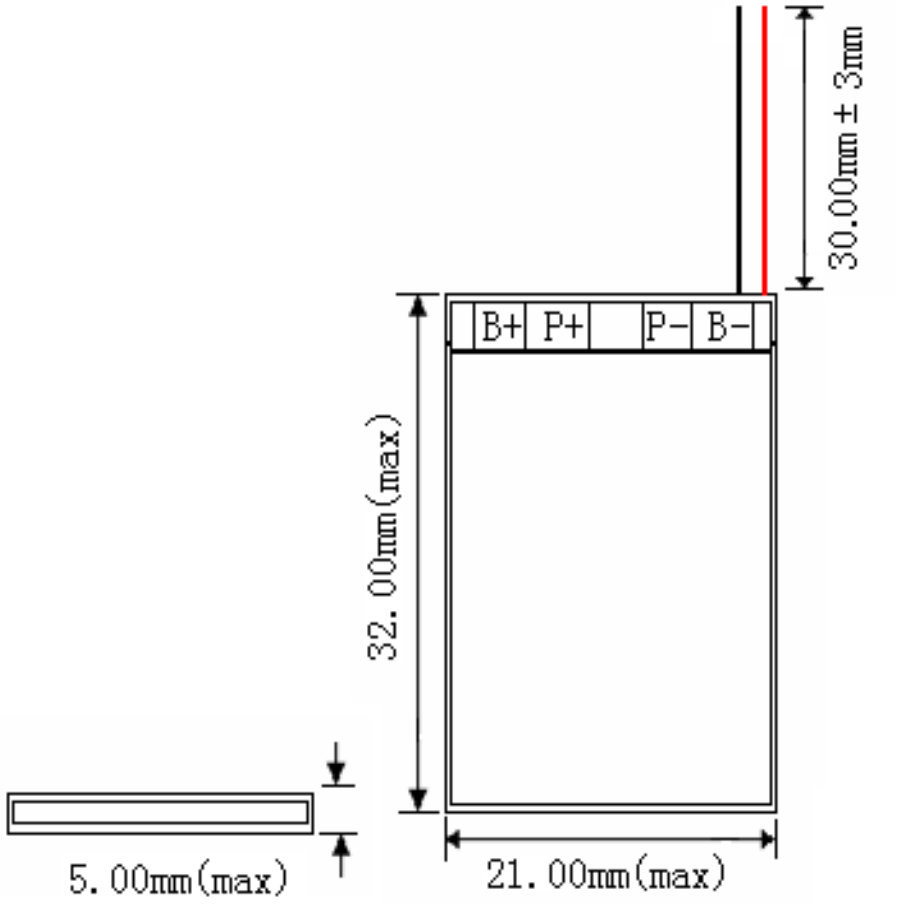
15 Battery + PCB Drawing 电池成品图

以下物料均符合RoHS要求

	元件名称	型 号	数 量	图 号	
	电 芯 Cell	502030	1	---	
	保护板 PCM	精工G3J过放3.0V截止保护板	1	---	
	红 线 Red Wire	(P+)	1	---	
	黑 线 Black Wire	(P-)	1	---	
	绝缘胶布 Insulating Tape	橙色/Orange	1	---	

1. 电池容量：

240~250mAh



				图纸编号		
客 户				标记	处数	更改前内容
版 本	A0	客户确认	盖章			
日 期	2005-8-9		签名			

Android / ZigBee 無線感測器平台
用戶手冊

編著者：長高科技股份有限公司

發行人：葉輔熛

發行所：長高科技股份有限公司

電　話：（04）23178130

劃　撥：第 22531058 帳號

地　址：407 台中市西屯區市政路 386 號 11 樓之 2

電　話：（04）22516589

傳　真：（04）22518358