OctopusX 平台教學

2010/3/15

報告者：汪昱志

Outline

- Zigbee開發 & 燒錄方式
- 如何建立一個Network
- 程式架構&範例介紹
- UART使用介紹
- 如何傳送資料經多點傳輸

Zigbee開發 & 燒錄方式

- Zigbee簡介
- 環境介紹
- 軟體設定
- 燒錄流程

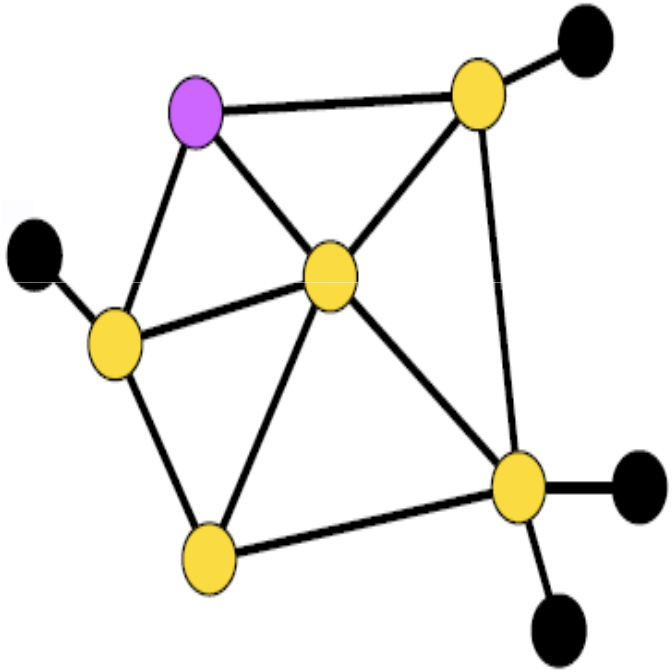
Zigbee簡介

- 全名為：
  - Wireless MAC and PHY Specification for Low-Rate Wireless Personal Area Networks (LR-WPANs)

- 在硬體架構來看，分為兩種角色
  - Full Function Node (FFD)
    - 提供完整IEEE 802.15.4規範的功能
    - 需要較高的運算效能以及記憶體
    - 通常採用固定的電源
  - Reduced Function Node (RFD)
    - 提供精簡的IEEE 802.15.4規範的功能
    - 使用較低的運算效能以及記憶體
    - 通常使用電池

Zigbee簡介

- 由網路架構來看，分成三種角色



- PAN coordinator (FFD)

管理整個 ZigBee 網路的控制中心

PAN ID , Security ,通道....

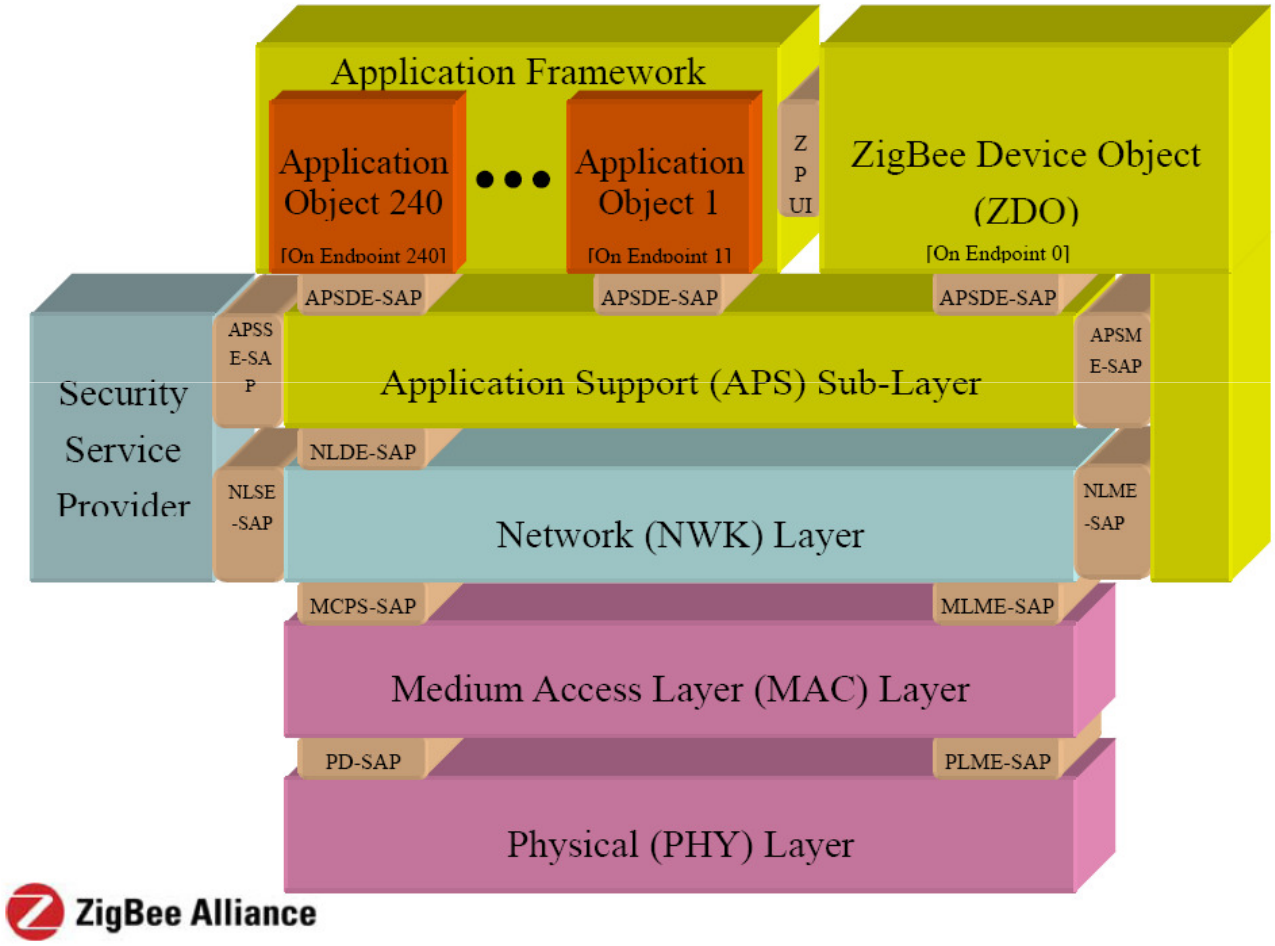
- Network Router (FFD)

負責延展整個網路的路由器

- End Device (RFD)

網路末端裝置(Sensor)

Zigbee簡介


Application Framework
Application Object 240
[On Endpoint 240]
Application Object 1
[On Endpoint 1]
ZPUI
ZigBee Device Object (ZDO)
[On Endpoint 0]
APSDE-SAP
APSDE-SAP
APSDE-SAP
APSSE-SAP
Application Support (APS) Sub-Layer
APSME-SAP
Security Service Provider
NLDE-SAP
NLSE-SAP
Network (NWK) Layer
NLME-SAP
MCPS-SAP
MLME-SAP
Medium Access Layer (MAC) Layer
PD-SAP
PLME-SAP
Physical (PHY) Layer
ZigBee Alliance

Zigbee開發 & 燒錄方式

- Zigbee簡介
- 環境介紹
- 軟體設定
- 燒錄流程

環境介紹

- 軟體：
  - IAR Embedded Workbench (EW8051 -730B)
  - SmartRF Flash Programmer
  - Z-stack 1.4.3-1.2.1(TI)
  - OctopusX 腳位設定檔
- 硬體：
  - OctopusX
  - OctopusX-Programming broad

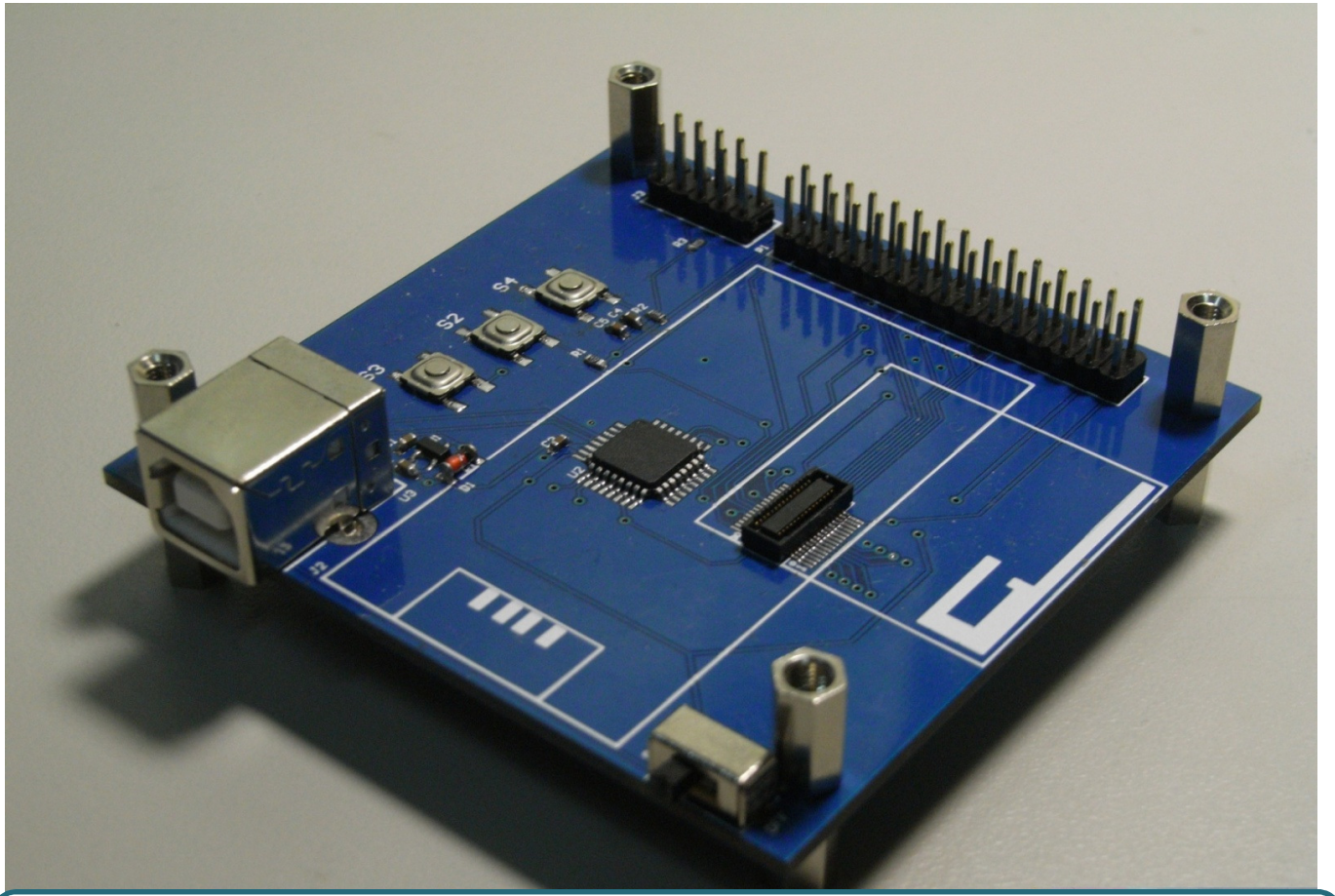
環境介紹 - 軟體介紹

- IAR Embedded WorkBench
  - IAR Embedded Workbench(EW8051)集成開發環境支援工程管理、編譯、彙編、鏈結、下載和除錯等各種基於8051 內核的處理器

- Z-stack
  - Z-Stack 是德州儀器公司（TI）推出的ZigBee 協定堆疊的免費下載版本。

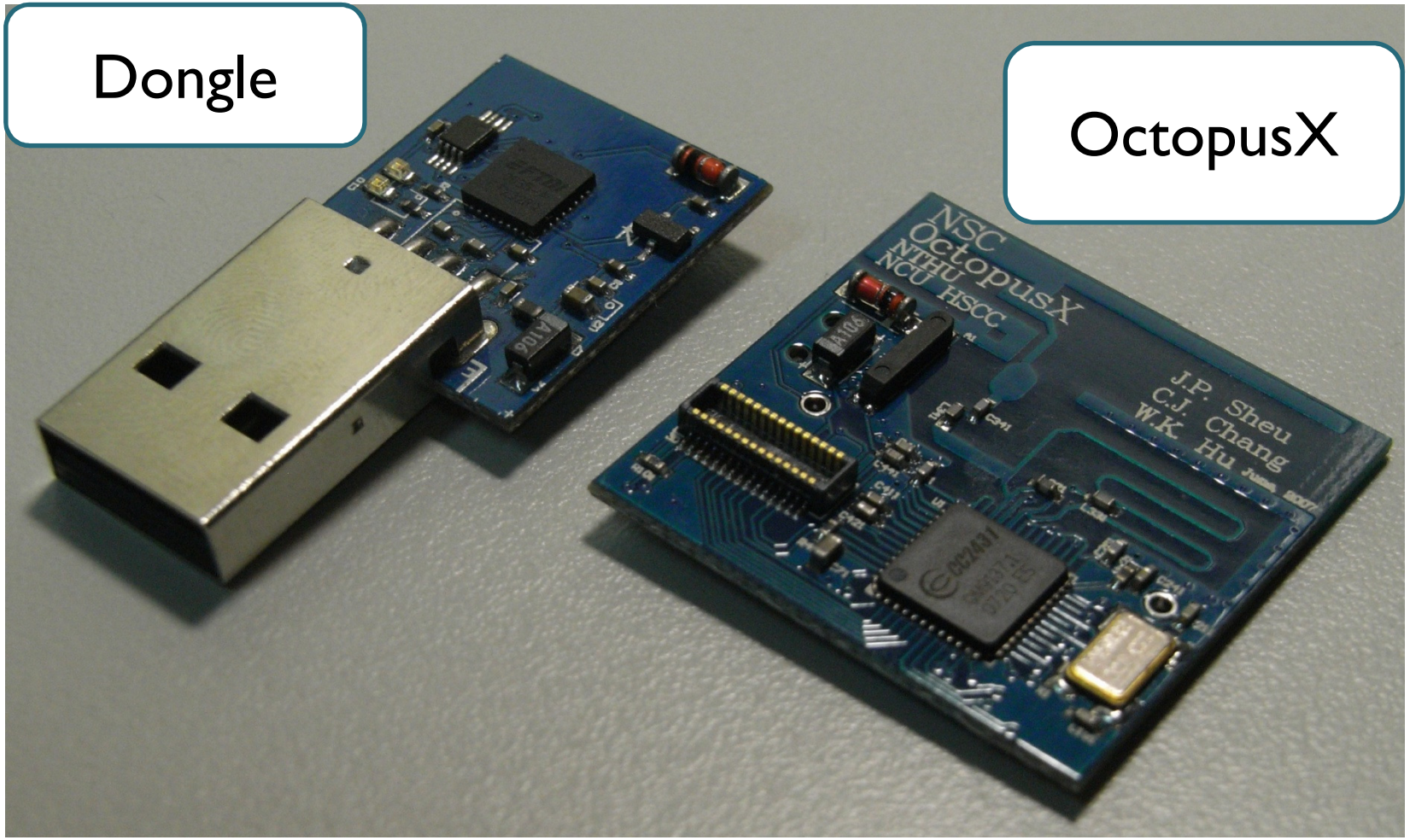
- SmartRF Flash Programmer工具軟體
  - 可被用來編譯TI 公司的晶片上系統微控制器的Flash 記憶體，它還可以支援IEEE 位址的讀/寫。

環境介紹 – 硬體介紹



OctopusX – Debugger board

環境介紹 – 硬體介紹


Dongle
OctopusX
NSC
OctopusX
NTHU
NCU HSCC
J.P. Sheu
C.J. Chang
W.K. Hu

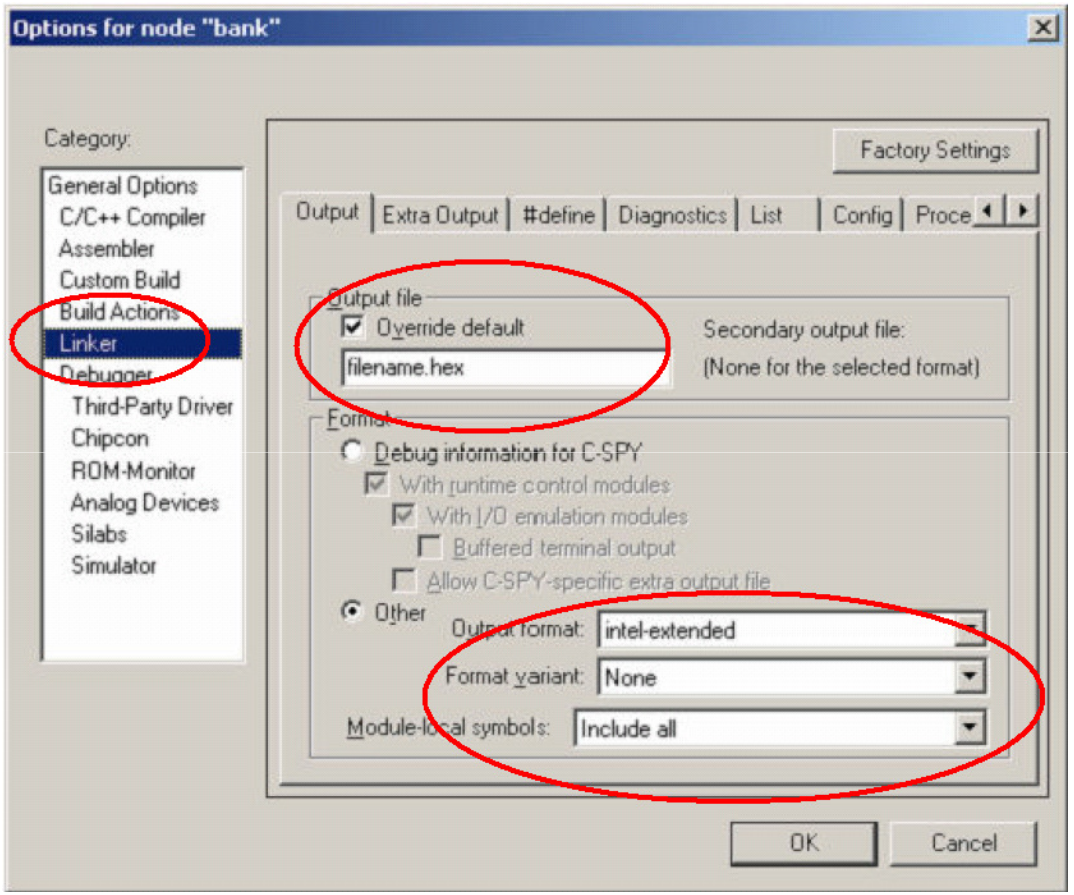
Zigbee開發 & 燒錄方式

- Zigbee簡介
- 環境介紹
- 軟體設定
- 燒錄流程

軟體設定

- IAR Embedded Workbench設定：
  - 請在專案上右鍵點選options
  - 選擇Linker Category
  - 更改output file，勾選Override default並將副檔名改為xxx.hex
  - Format 請選擇Other（如下頁圖所示）
  - 重新編譯前，修改tools資料夾，將其中的f8w2430.xcl 第90行的地方註解取消

軟體設定




```
//
//
//     HEX FILE GENERATION
//
// Include the following line when generating hex file:
-M(CODE)_BANK1A-1FFFF,28000-2FFFF,38000-3DFFF,3F000-3fff7=(_CODE_END+1)-0xFFFF,0x10000-0x17FFF,
//
```

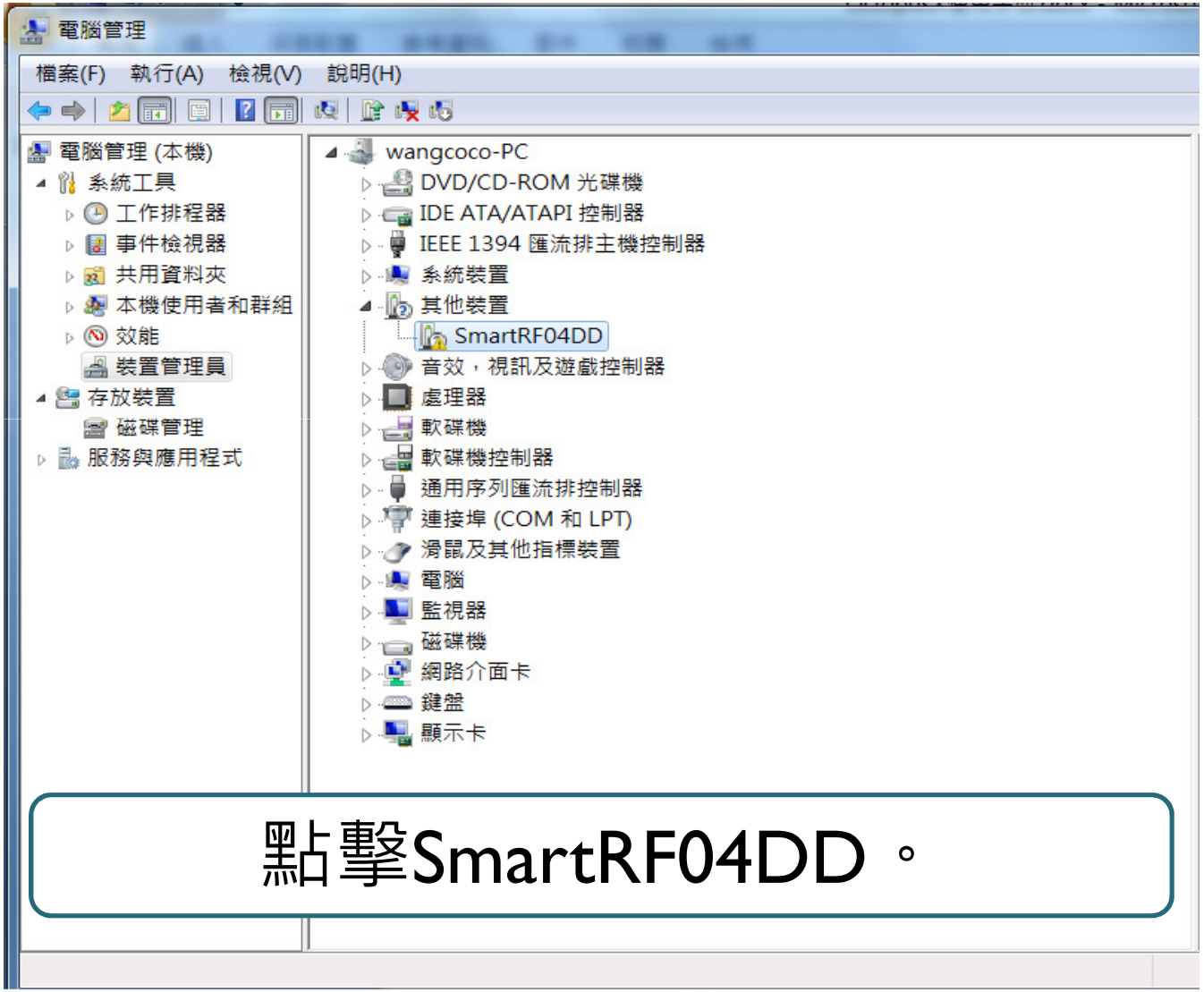
軟體設定

- Z-Stack v1.4.3 的安裝
  - 在http://www.ti.com 免費下載其最新版本
- Z-Stack v1.4.3 安裝完成後(預設安裝到C 槽)，其中會包含四個目錄：
  - Components：資料夾包含Z-Stack 的各種元件。
  - Projects：資料夾中包含了幾個IAR 工程，它們是Z-Stack 應用實
  - Documents：包含了Z-Stack 的各種說明文檔。
  - Tools：包含了ZOAD 和Z-Tool 兩個工具。
- OctopusX腳位設定檔
  - 請直接複蓋C:\Texas Instruments\ZStack-1.4.3-1.2.1\Components\hal

軟體設定

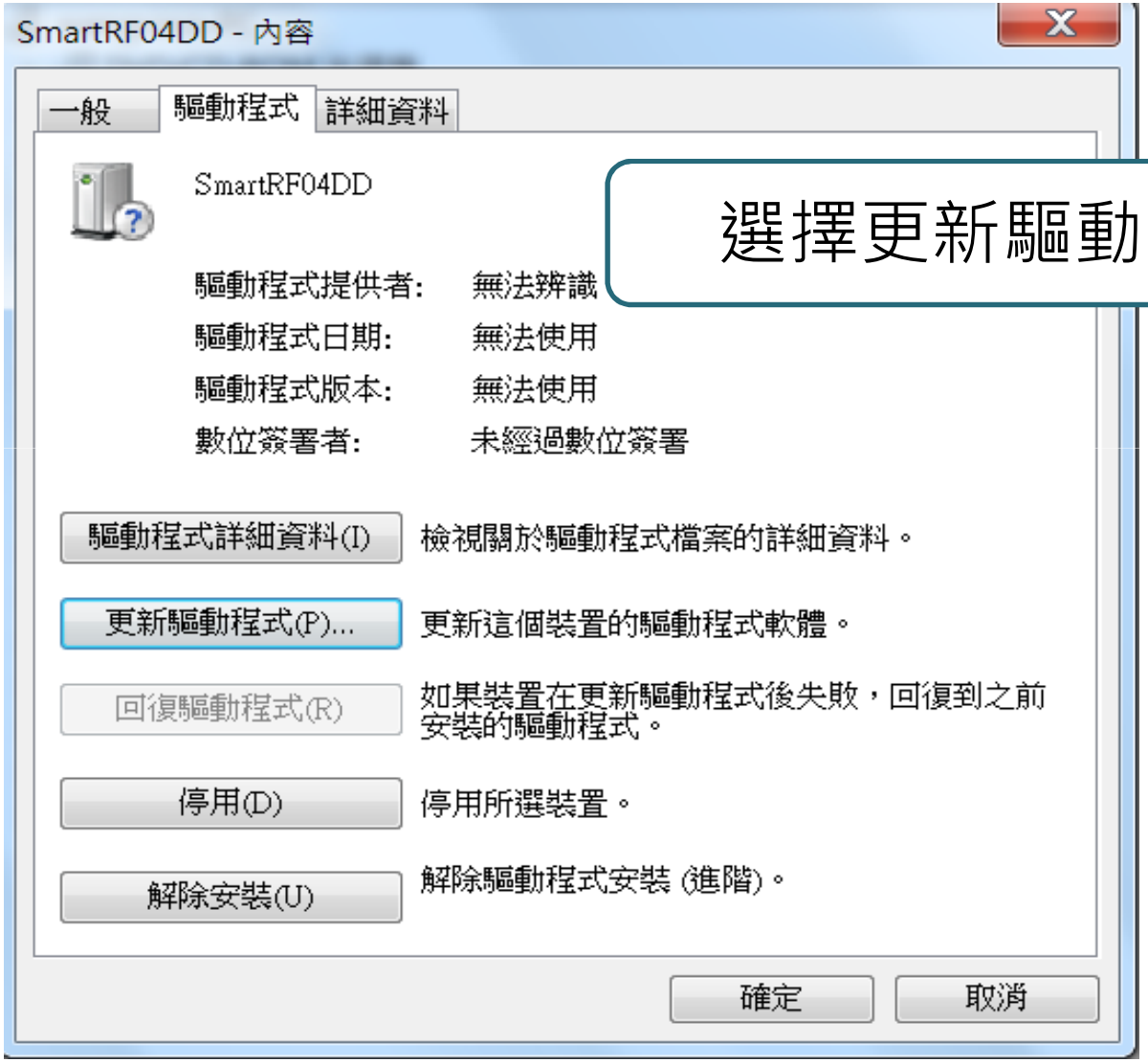
- PC端的驅動程式安裝
  - 在使用SmartRF Flash Programmer工具軟體前，首先需要先安裝PC端驅動程式，安裝過程請依據新增硬體精靈步驟依序執行即可完成。
  - 首先確保OctopusX-Debugger 沒有連接任何目標板，然後用我們配套提供的USB 電纜連接OctopusX-Debugger 與主機，並將開關切換至On，與此同時使用者PC 端將提示發現新硬體：
  - 接下來按以下各圖所示，安裝CC243x-Debugger 的PC 端驅動程式：

軟體設定



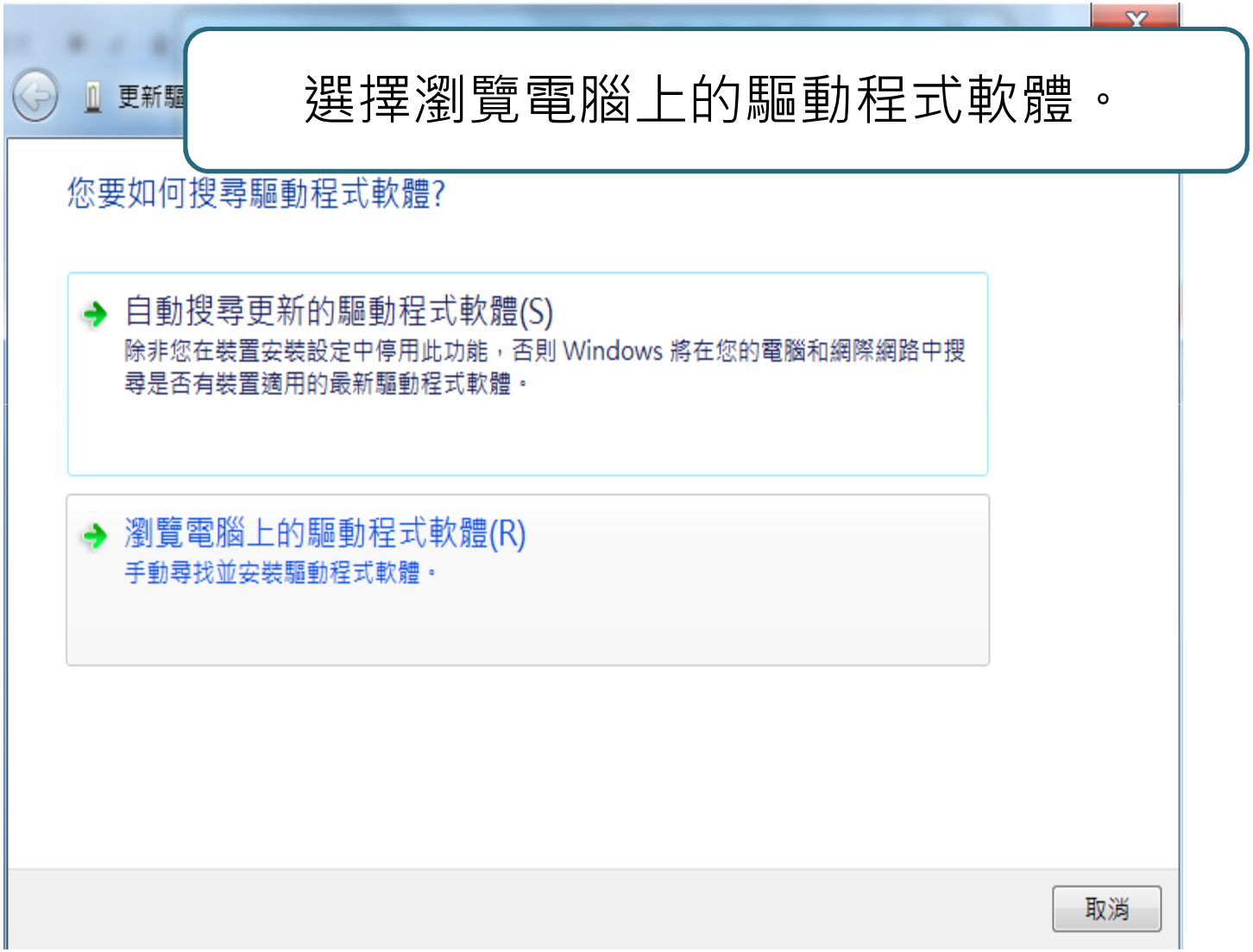
點擊SmartRF04DD。

軟體設定



選擇更新驅動程式。

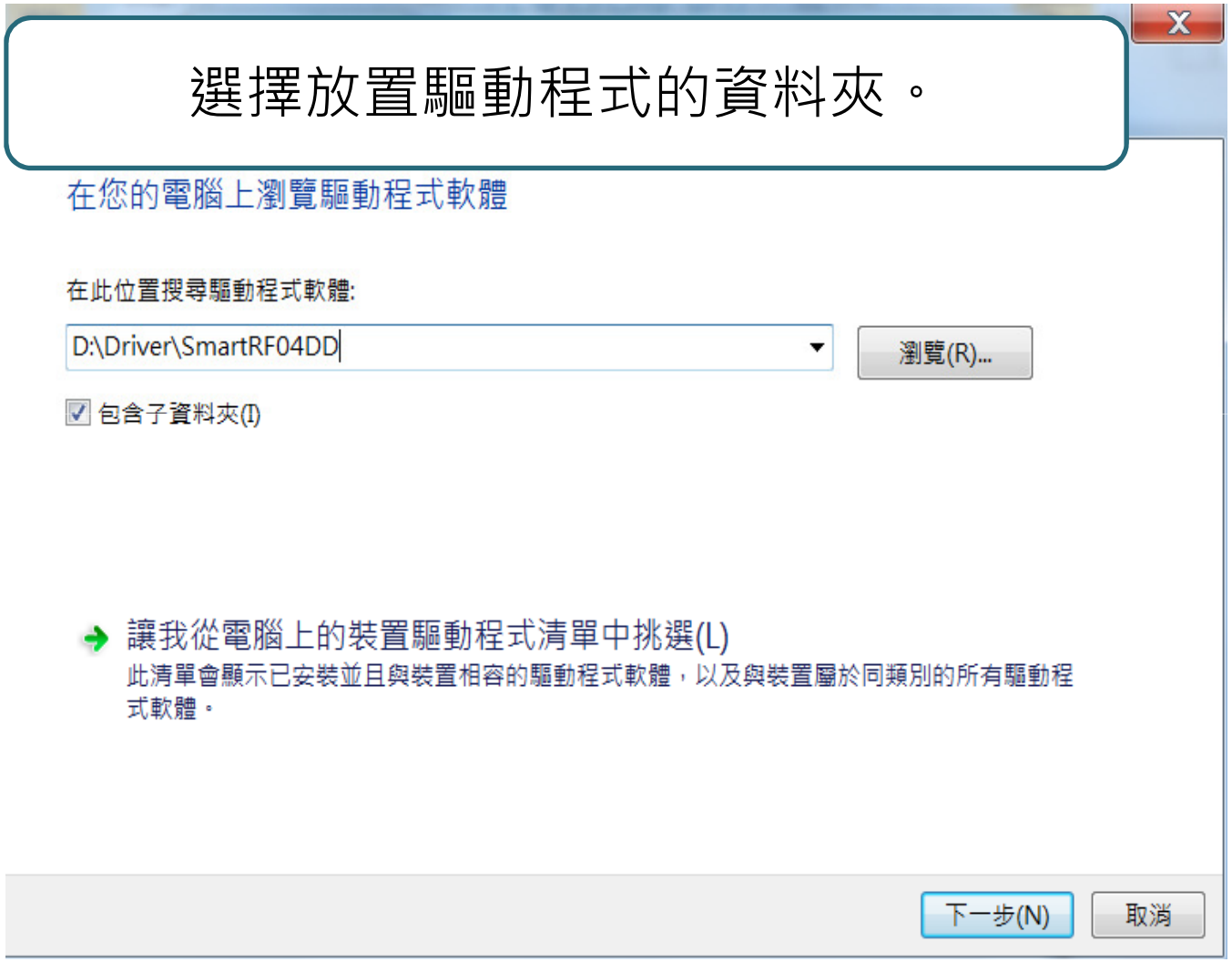
軟體設定



選擇瀏覽電腦上的驅動程式軟體。

軟體設定

選擇放置驅動程式的資料夾。



軟體設定

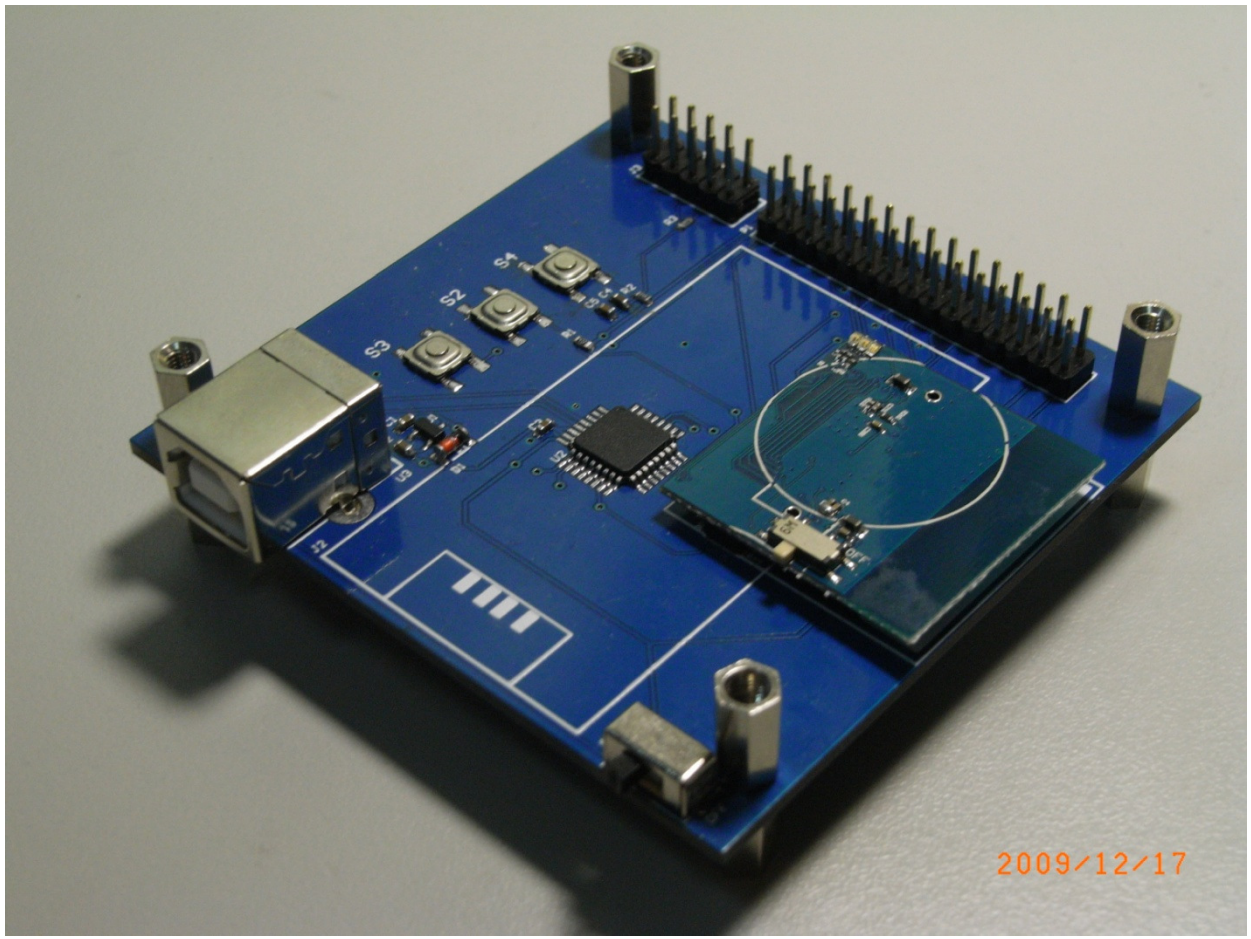

更新驅動程式軟體 - Chipcon Debug Dongle
Windows 已順利更新您的驅動程式軟體
Windows 已完成安裝這個裝置的驅動程式軟體:
Chipcon Debug Dongle
關閉(C)

Zigbee開發 & 燒錄方式

- Zigbee簡介
- 環境介紹
- 軟體設定
- 燒錄流程

OctopusX燒錄流程

- 燒錄方式 – 硬體安裝方式
- 1.將OctopusX裝載上Debugger broad

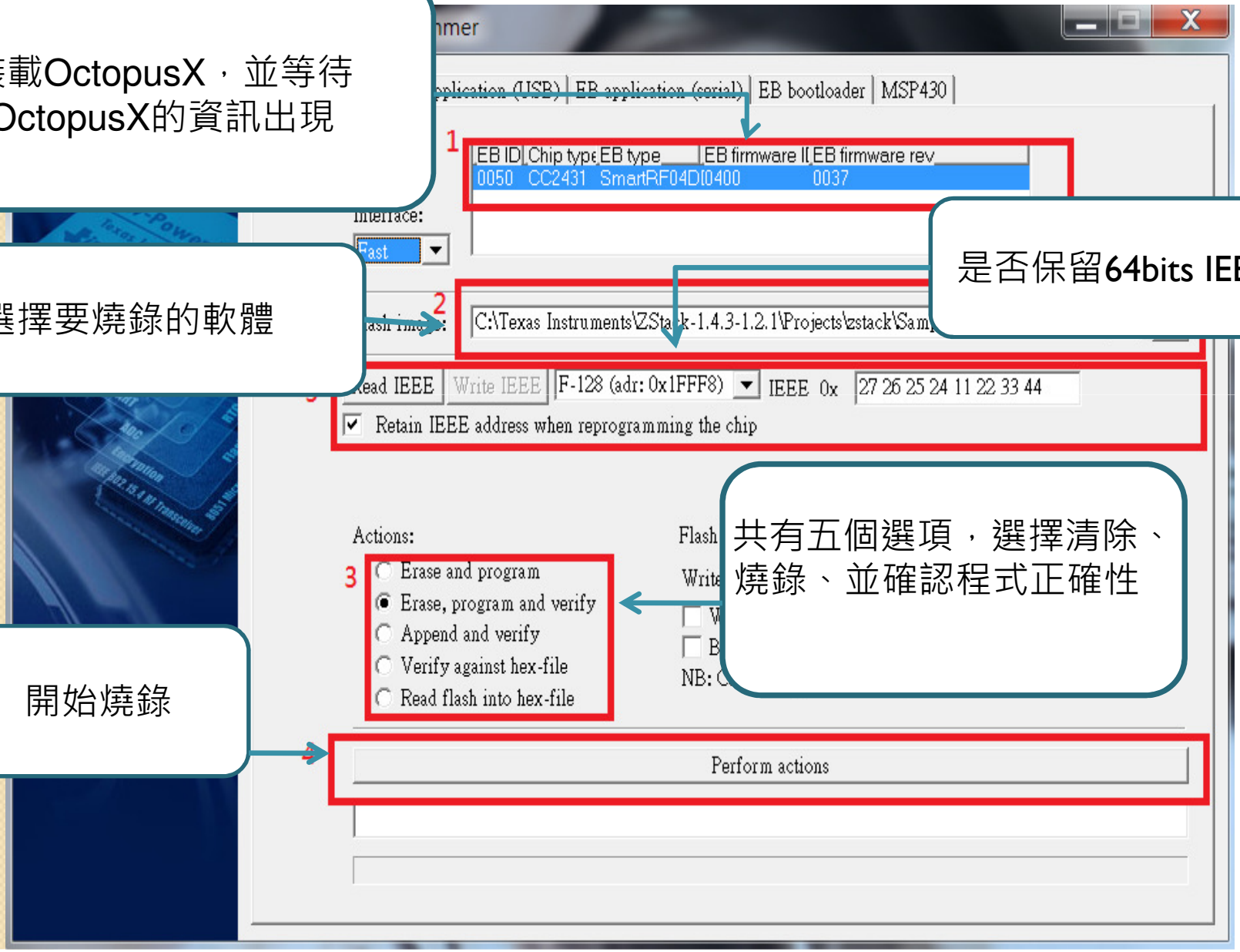


OctopusX燒錄流程

裝載OctopusX，並等待OctopusX的資訊出現

選擇要燒錄的軟體

開始燒錄



是否保留64bits IEEE位址

共有五個選項，選擇清除、燒錄、並確認程式正確性

Outline

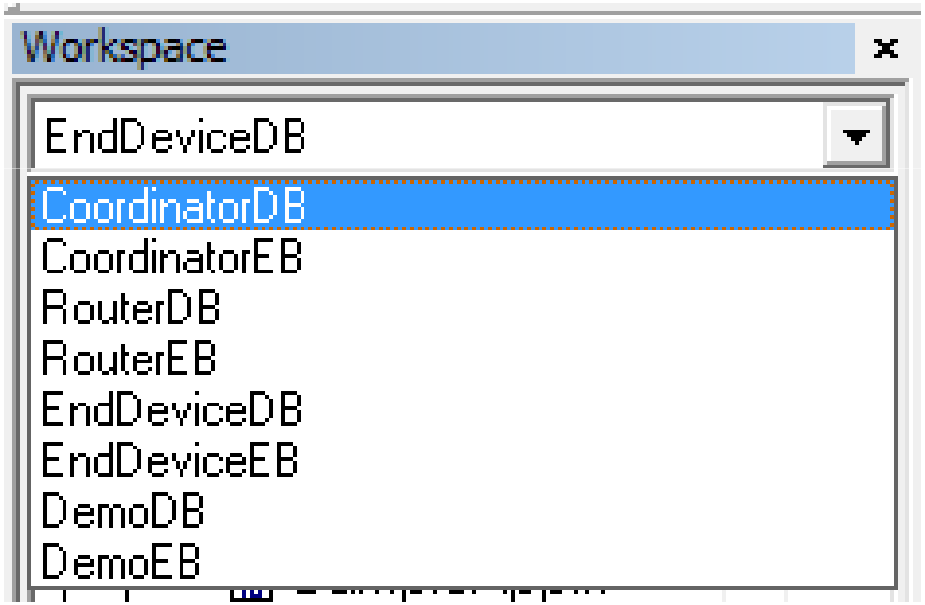
- Zigbee開發 & 燒錄方式
- 如何建立一個Network
- 程式架構&範例介紹
- UART使用介紹
- 如何傳送資料經多點傳輸

如何建立一個Network

- Coordinator
  - 啟動一個ZigBee 網路(網路中的第一個設備)
  - 選擇一個頻道和一個網路識別字（PAN ID）
  - 允許其他設備加入網路
- Multi-hop routing
  - Router
  - 允許其他設備加入網路
  - Multi-hop routing
- End device

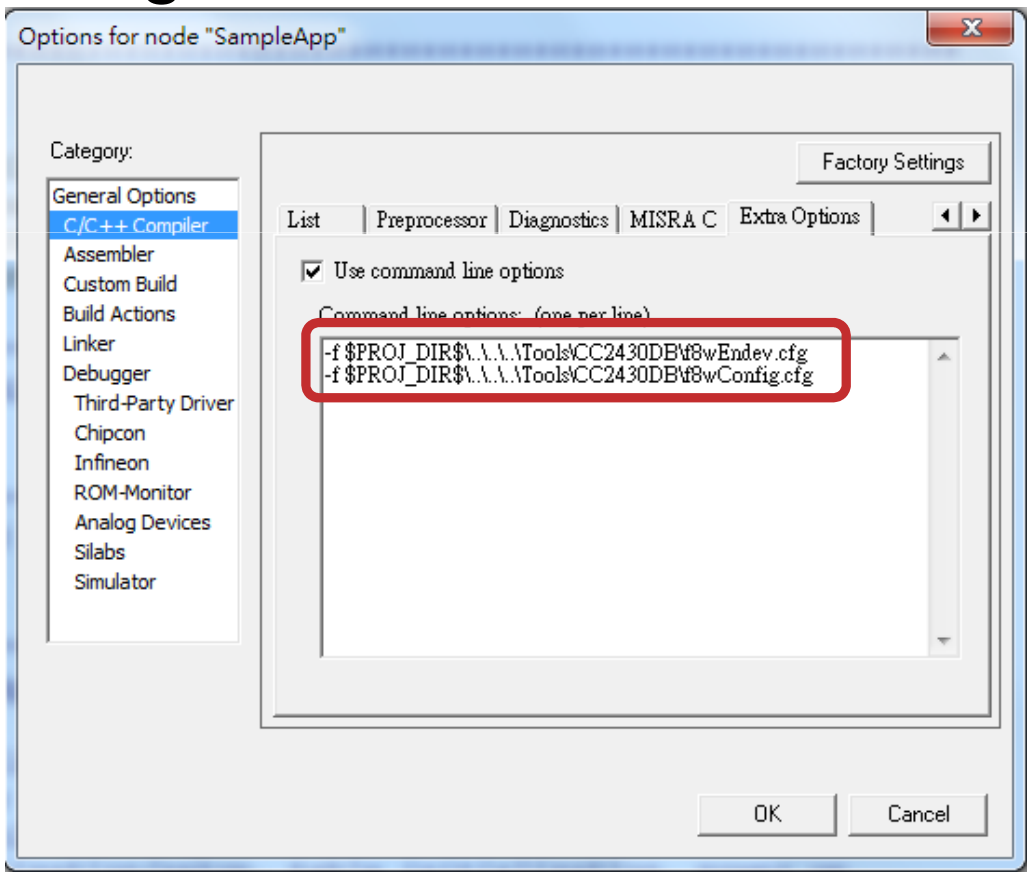
如何建立一個Network

- 如何設定感測器為特定類型?
  - 專案中選擇



如何建立一個Network

- 在專案→C/C++ Compiler下選擇對應的config檔



如何建立一個Network

- 將各個感測器燒錄不同角色的程式，即可自行形成網路。
- 一個網路只能有一個Coordinator
- 可自行調整一個網路的最大深度，一個Router能擁有的Child數
  - nwk_globals.h　　Line 80
  - nwk_globals.c　　Line 121

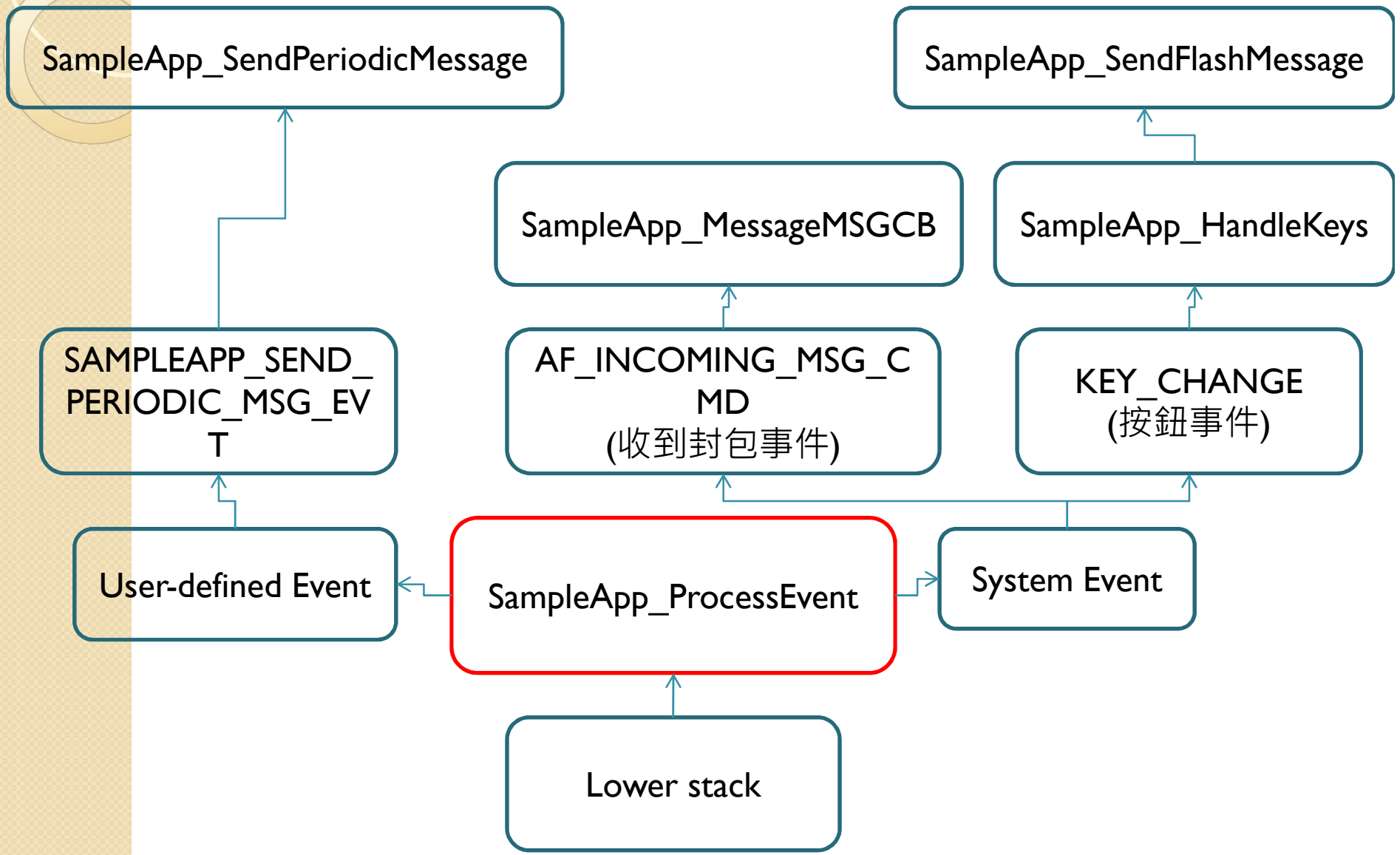
Outline

- Zigbee開發 & 燒錄方式
- 如何建立一個Network
- 程式架構&範例介紹
- UART使用介紹
- 如何傳送資料經多點傳輸

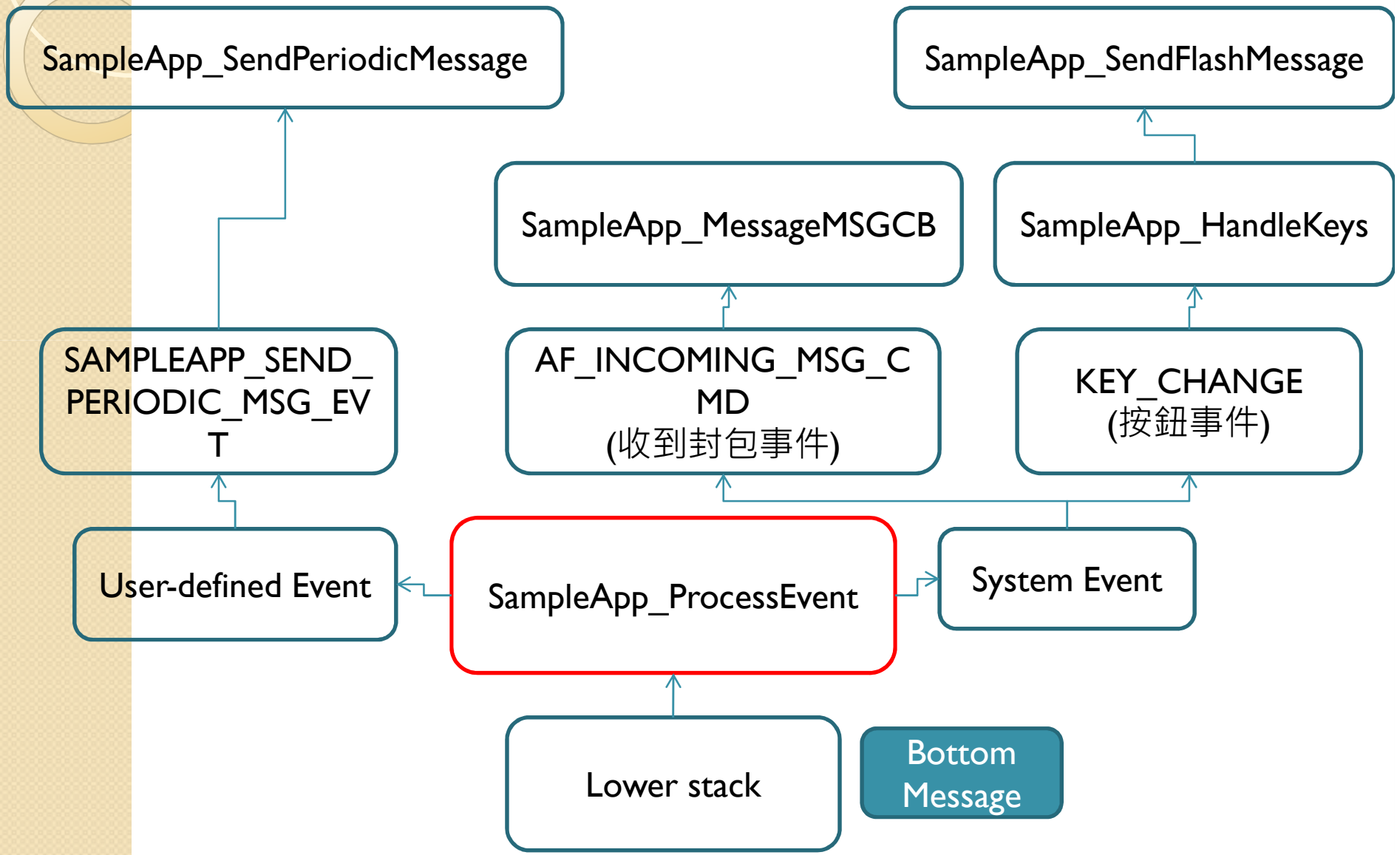
基礎程式範例介紹—SampleApp

- SampleApp 實驗
  - 實驗目的：
    一個ZigBee 網路中的某個設備發送“閃爍LED”命令給該網路中群組1 的所有成員。群組1 的所有成員在收到命令後，將會閃爍LED。
  - 實驗設備：
    OctopusX ：3個
    OctopusX Programming Board ：1個

基礎程式範例介紹—程式架構



基礎程式範例介紹—程式架構



SampleApp — 重要參數介紹

```
// This list should be filled with Application specific Cluster IDs.
const cId_t SampleApp_ClusterList[SAMPLE
{
  SAMPLEAPP_PERIODIC_CLUSTERID,
  SAMPLEAPP_FLASH_CLUSTERID
};

const SimpleDescriptionFormat_t SampleApp_SimpleDesc =
{
  SAMPLEAPP_ENDPOINT,              //  i
  SAMPLEAPP_PROFID,                //  u
  SAMPLEAPP_DEVICEID,              //  u
  SAMPLEAPP_DEVICE_VERSION,        //  i
  SAMPLEAPP_FLAGS,                 //  int   AppFlags:4;
  SAMPLEAPP_MAX_CLUSTERS,          //  uint8  AppNumInClusters;
  (cId_t *)SampleApp_ClusterList,  //  u
  SAMPLEAPP_MAX_CLUSTERS,          //  u
  (cId_t *)SampleApp_ClusterList   //  u
};

// This is the Endpoint/Interface description.  It is defined here, but
// filled-in in SampleApp_Init().  Another way to go would be to fill
// in the structure here and make it a "const" (in code space).  The
// way it's defined in this sample app
endPointDesc_t SampleApp_epDesc;
```

傳送封包的ID，此應用傳送兩種封包：
1.SAMPLEAPP_PERIODIC_CLUSTERID
2.SAMPLEAPP_FLASH_CLUSTERID

應用程式的辨識ID(對於下層來說)

接收&傳送的封包ID表，此應用只傳送上述的封包ID

完整描述此應用的資料結構

SampleApp — SampleApp_Init介紹

```
afAddrType_t SampleApp_Periodic_DstAddr;
afAddrType_t SampleApp_Flash_DstAddr;
void SampleApp_Init( uint8 task_id )
{
  SampleApp_TaskID = task_id;

  SampleApp_Periodic_DstAddr.addrMode = (afAddrMode_t)AddrBroadcast;
  SampleApp_Periodic_DstAddr.endPoint = SAMPLEAPP_ENDPOINT;
  SampleApp_Periodic_DstAddr.addr.shortAddr = 0xFFFF;

  SampleApp_Flash_DstAddr.addrMode = (afAddrMode_t)afAddrGroup;
  SampleApp_Flash_DstAddr.endPoint = SAMPLEAPP_ENDPOINT;
  SampleApp_Flash_DstAddr.addr.shortAddr = SAMPLEAPP_FLASH_GROUP;

  SampleApp_epDesc.endPoint = SAMPLEAPP_ENDPOINT;
  SampleApp_epDesc.task_id = &SampleApp_TaskID;
  SampleApp_epDesc.simpleDesc = (SimpleDescriptionFormat_t *)&SampleApp_SimpleDesc;
  SampleApp_epDesc.latencyReq = noLatencyReqs;

  afRegister( &SampleApp_epDesc );

  RegisterForKeys( SampleApp_TaskID );

  SampleApp_Group.ID = 0x0001;
  osal_memcpy( SampleApp_Group.name, "Group 1", 7  );
  aps_AddGroup( SAMPLEAPP_ENDPOINT, &SampleApp_Group );
}
```

目的地址的資料結構

1.傳送模式-廣播

2.對象應用程式ID

3.廣播內定地址

4.傳送模式-組內傳送

5.組名稱(此應用內定)

6.向下層註冊此應用程式

7.向下層註冊按鈕事件

8.註冊群組

SampleApp_ProcessEvent —SYS_MSG

```
if ( events & SYS_EVENT_MSG )
{
  MSGpkt = (afIncomingMSGPacket_t *)osal_msg_receive( SampleApp_TaskID );
  while ( MSGpkt )
  {
    switch ( MSGpkt->hdr.event )
    {
      case KEY_CHANGE:
        SampleApp_HandleKeys( ((keyChange_t *)MSGpkt)->state, ((keyChange_t *)MSGpkt)->keys );
        break;
      case AF_INCOMING_MSG_CMD:
        SampleApp_MessageMSGCB( MSGpkt );
        break;
      case ZDO_STATE_CHANGE:
        SampleApp_NwkState = (devStates_t)(MSGpkt->hdr.status);
        if ( (SampleApp_NwkState == DEV_ZB_COORD)
            || (SampleApp_NwkState == DEV_ROUTER)
            || (SampleApp_NwkState == DEV_END_DEVICE) )
        {
          osal_start_timerEx( SampleApp_TaskID,
                              SAMPLEAPP_SEND_PERIODIC_MSG_EVT,
                              SAMPLEAPP_SEND_PERIODIC_MSG_TIMEOUT );
        }
        else
        {
        }
        break;

      default:
        break;
    }
    osal_msg_deallocate( (uint8 *)MSGpkt );
    MSGpkt = (afIncomingMSGPacket_t *)osal_msg_receive( SampleApp_TaskID );
  }
```

- 加入網路後，狀態改變

SampleApp_ProcessEvent — User-defined event

```
// Send Message Timeout
#define SAMPLEAPP_SEND_PERIODIC_MSG_TIMEOUT    5000     // Every 5 seconds
// Application Events (OSAL) - These are bit weighted definitions.
#define SAMPLEAPP_SEND_PERIODIC_MSG_EVT        0x0001
// Group ID for Flash Command
#define SAMPLEAPP_FLASH_GROUP                  0x0001
// Flash Command Duration - in milliseconds
#define SAMPLEAPP_FLASH_DURATION               1000
```

Self - defined

In SampleApp.h

```
// Send a message out - This event is generated by a timer
//  (setup in SampleApp_Init()).
if ( events & SAMPLEAPP_SEND_PERIODIC_MSG_EVT )
{
  // Send the periodic message
  SampleApp_SendPeriodicMessage();

  // Setup to send message again in normal period (+ a little jitter)
  osal_start_timerEx( SampleApp_TaskID, SAMPLEAPP_SEND_PERIODIC_MSG_EVT,
      (SAMPLEAPP_SEND_PERIODIC_MSG_TIMEOUT + (osal_rand() & 0x00FF)) );

  // return unprocessed events
  return (events ^ SAMPLEAPP_SEND_PERIODIC_MSG_EVT);
}
```

1. 通知應用的ID
2. 哪一個自定事件
3. 間隔多久時間

SampleApp_HandleKeys

```
void SampleApp_HandleKeys( uint8 shift, uint8 keys )
{
  if ( keys & HAL_KEY_SW_1 )
  {
    SampleApp_SendFlashMessage( SAMPLEAPP_FLASH_DURATION );
  }

  if ( keys & HAL_KEY_SW_2 )
  {
    aps_Group_t *grp;
    grp = aps_FindGroup( SAMPLEAPP_ENDPOINT, SAMPLEAPP_FLASH_GROUP );
    if ( grp )
    {
      // Remove from the group
      aps_RemoveGroup( SAMPLEAPP_ENDPOINT, SAMPLEAPP_FLASH_GROUP );
    }
    else
    {
      // Add to the flash group
      aps_AddGroup( SAMPLEAPP_ENDPOINT, &SampleApp_Group );
    }
  }
}
```

更改為HAL_KEY_SW_6

SampleApp_MessageMSGCB

```
void SampleApp_MessageMSGCB( afIncomingMSGPacket_t *pkt )
{
  uint16 flashTime;

  switch ( pkt->clusterId )
  {
    case SAMPLEAPP_PERIODIC_CLUSTERID:
      break;

    case SAMPLEAPP_FLASH_CLUSTERID:
      flashTime = BUILD_UINT16(pkt->cmd.Data[1], pkt->cmd.Data[2] );
      HalLedBlink( HAL_LED_4, 4, 50, (flashTime / 4) );
      break;
  }
}
```

- 收到ID為SAMPLEAPP_PERIODIC_CLUSTERID封包處理方式

1. 燈ID
2. 亮的次數
3. 亮的時間區段內，亮多少%時間
4. 亮的時間長度

SampleApp_SendFlashMessage

```
void SampleApp_SendFlashMessage( uint16 flashTime )
{
  uint8 buffer[3];
  buffer[0] = (uint8)(SampleAppFlashCounter++);
  buffer[1] = LO_UINT16( flashTime );
  buffer[2] = HI_UINT16( flashTime );

  if ( AF_DataRequest( &SampleApp_Flash_DstAddr, &SampleApp_epDesc,
                       SAMPLEAPP_FLASH_CLUSTERID,
                       3,
                       buffer,
                       &SampleApp_TransID,
                       AF_DISCV_ROUTE,
                       AF_DEFAULT_RADIUS ) == afStatus_SUCCESS )
  {
  }
  else
  {
    // Error occurred in request to send.
  }
}
```

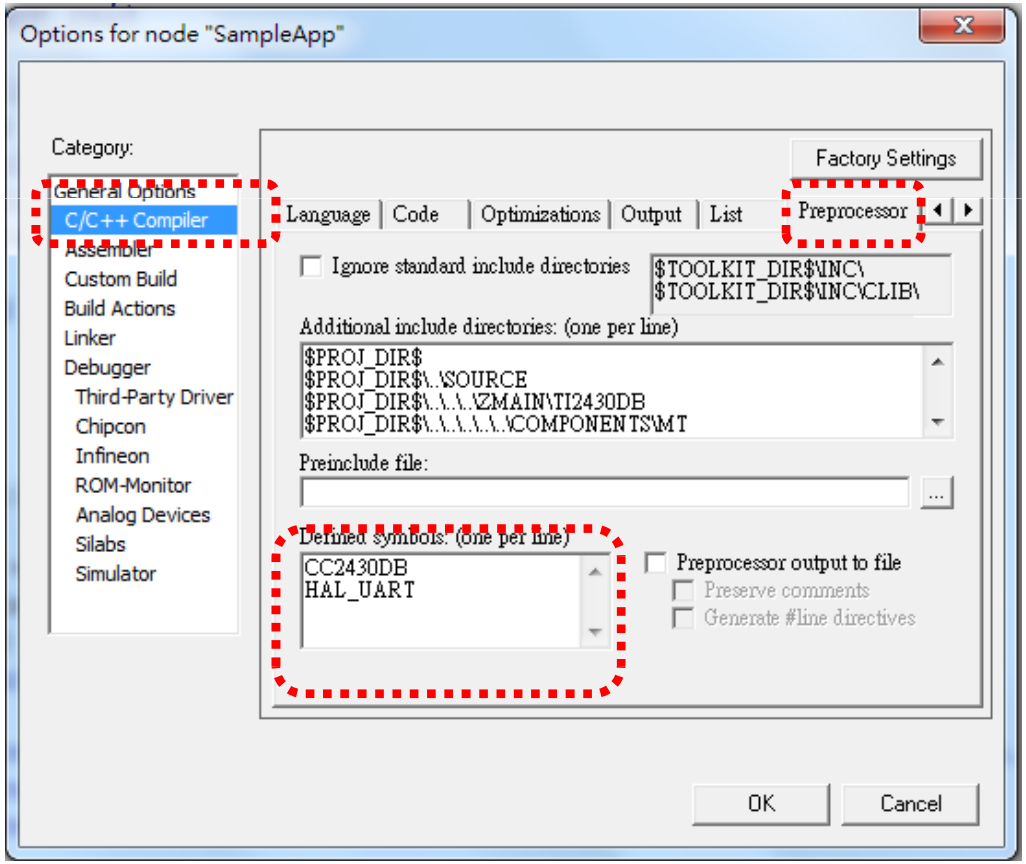
- DATA為閃燈秒數
- 傳送DATA的FUNCTION

Outline

- Zigbee開發 & 燒錄方式
- 如何建立一個Network
- 程式架構&範例介紹
- UART使用介紹
- 如何傳送資料經多點傳輸
- ADC使用介紹

UART使用介紹

- 專案→C/C++ Compiler→Preprocessor 加入HAL_UART參數



UART使用介紹

- 在需要使用UART的程式下，加入

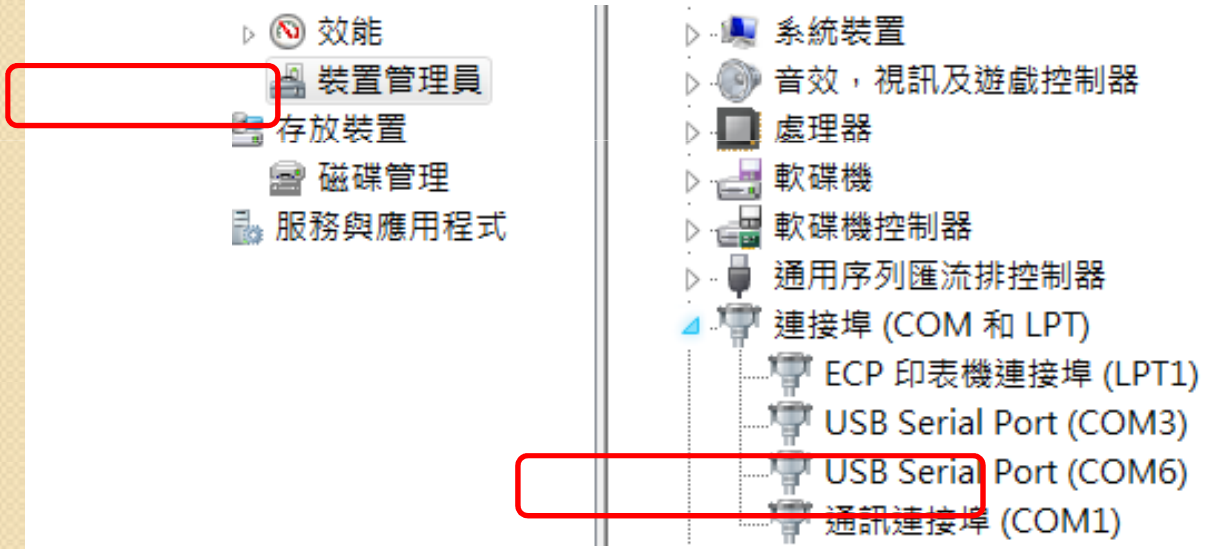
```
#include "uart.c"
```

- 在程式的Init function裡加入
  - `void SampleApp_Init( uint8 task_id )`
  - open(projectname_TaskID);
- 在需要由uart傳回資料的地方使用

```
HalUARTWrite(
SERIAL_APP_PORT,        //內定port
data,                   //資料pointer(uint8*)
length                  //資料長度
);
```

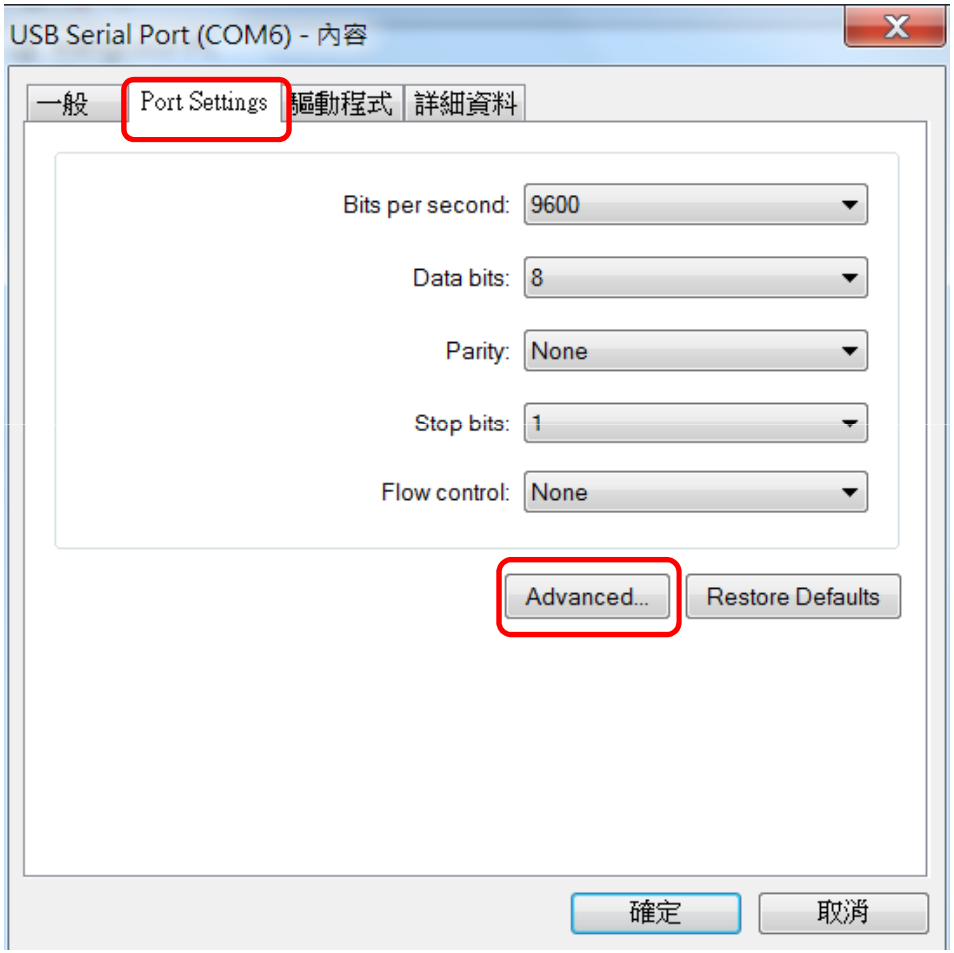
UART使用介紹

- 接收uart資料
  - 設定comport



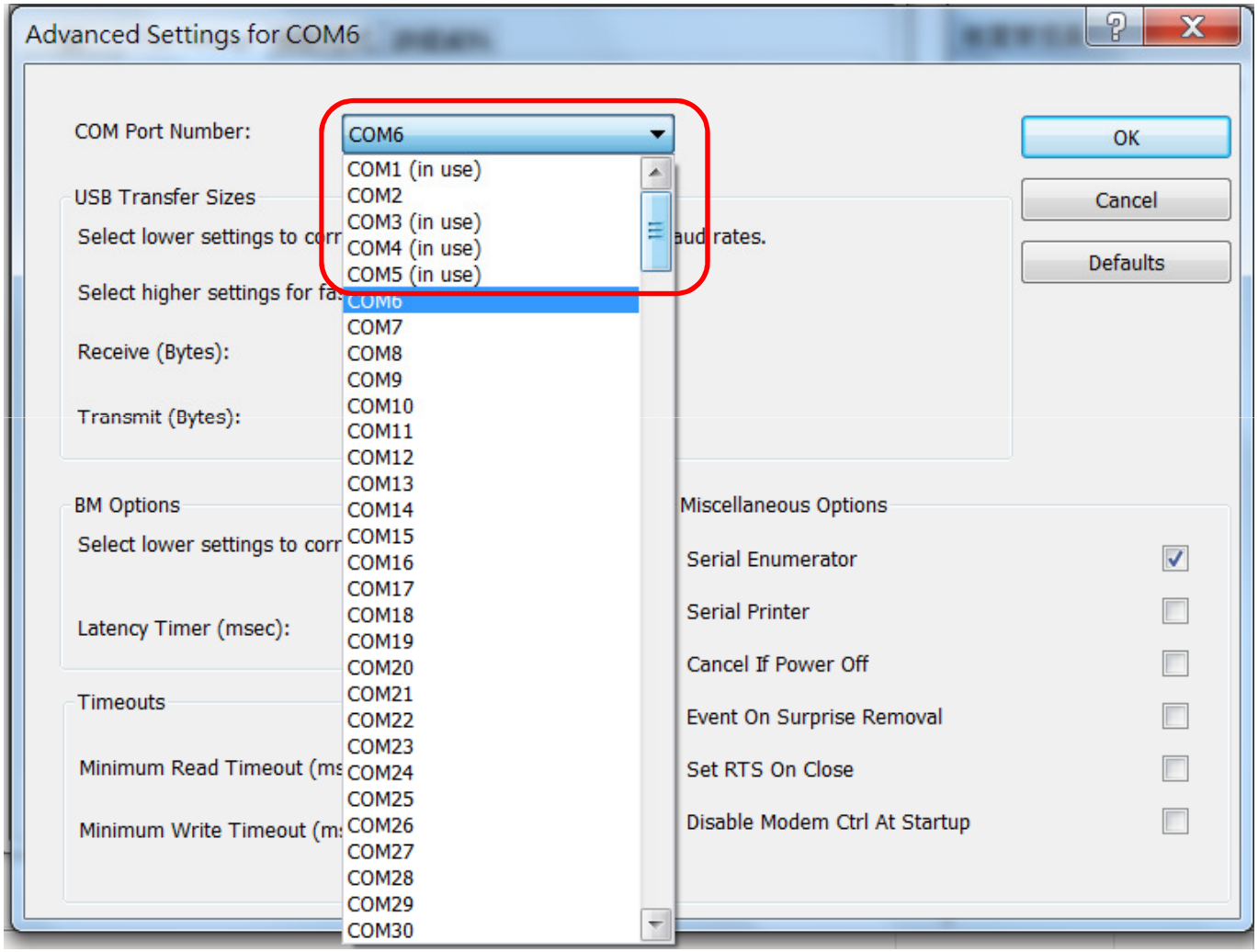
  - 裝置管理員→USB Serial Port內容

UART使用介紹



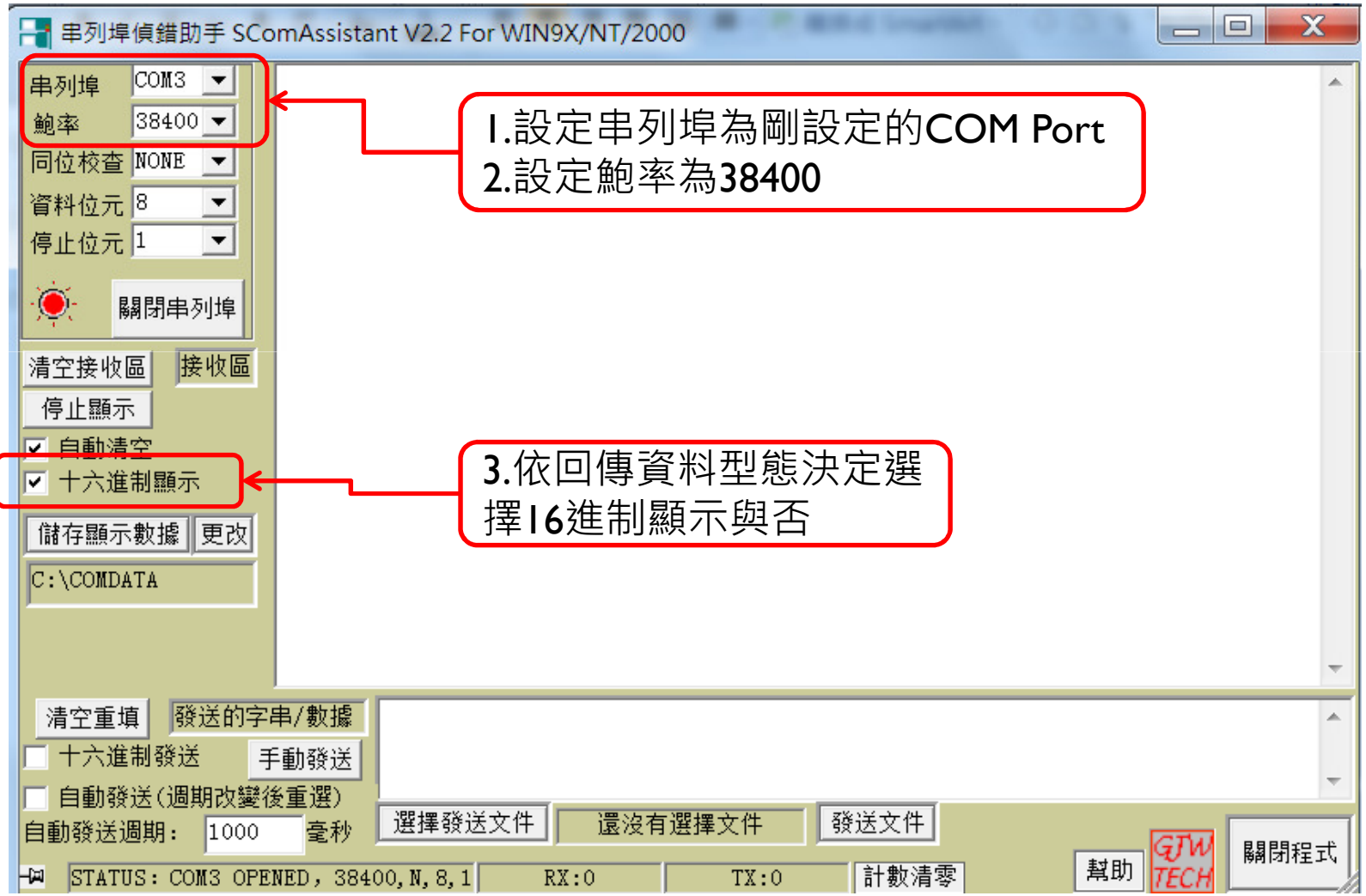
選取Port Settings中的Advanced選項

UART使用介紹

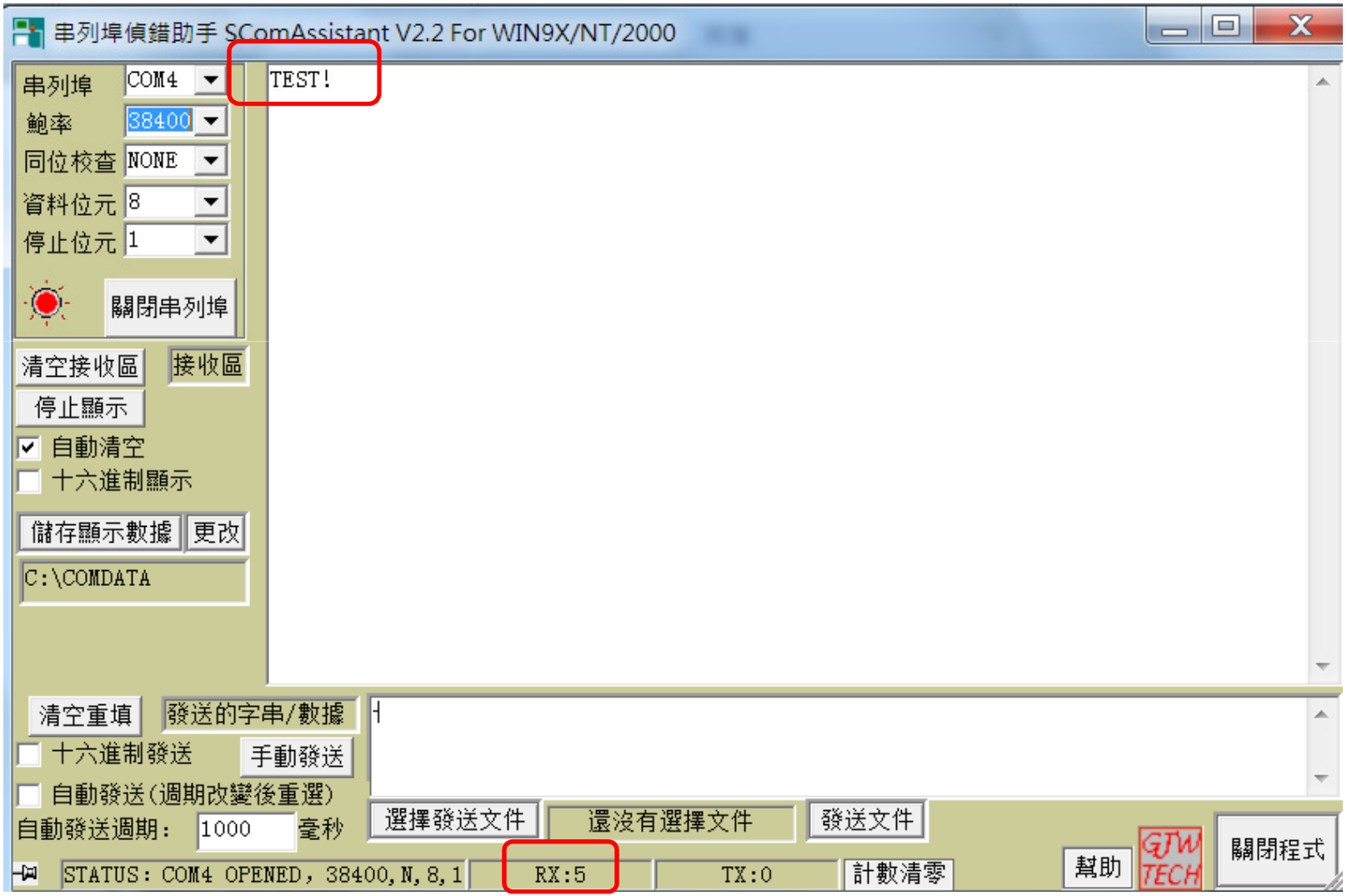


將COM Port改為1~4其中一個

UART使用介紹


串列埠偵錯助手 SComAssistant V2.2 For WIN9X/NT/2000
串列埠 COM3
鮑率 38400
同位校查 NONE
資料位元 8
停止位元 1
關閉串列埠
清空接收區
接收區
停止顯示
自動清空
十六進制顯示
儲存顯示數據
更改
C:\COMDATA
清空重填
發送的字串/數據
十六進制發送
手動發送
自動發送(週期改變後重選)
自動發送週期: 1000 毫秒
選擇發送文件
還沒有選擇文件
發送文件
STATUS: COM3 OPENED, 38400,N,8,1
RX:0
TX:0
計數清零
幫助
關閉程式
1.設定串列埠為剛設定的COM Port
2.設定鮑率為38400
3.依回傳資料型態決定選擇16進制顯示與否

UART使用介紹


串列埠偵錯助手 SComAssistant V2.2 For WIN9X/NT/2000
串列埠
COM4
鮑率
38400
同位校查
NONE
資料位元
8
停止位元
1
關閉串列埠
清空接收區
接收區
停止顯示
自動清空
十六進制顯示
儲存顯示數據
更改
C:\COMDATA
TEST!
清空重填
發送的字串/數據
十六進制發送
手動發送
自動發送(週期改變後重選)
自動發送週期：
1000
毫秒
選擇發送文件
還沒有選擇文件
發送文件
STATUS: COM4 OPENED, 38400,N,8,1
RX:5
TX:0
計數清零
幫助
GJW TECH
關閉程式

Outline

- Zigbee開發 & 燒錄方式
- 如何建立一個Network
- 程式架構&範例介紹
- UART使用介紹
- 如何傳送資料經多點傳輸

如何傳送資料經多點傳輸

- 知道目的地的地址
  - 對方和自已在同一個群組
    - 組內廣播
  - 目的地可借由定義特定某包形式，並使用廣播來讓可能的來源地得知自已的位址
- 選擇相對應的地址格式，並填入目的地位址

```
afAddrType_t SampleApp_Flash_DstAddr;
SampleApp_Flash_DstAddr.addrMode = (afAddrMode_t)Addr16Bit;
SampleApp_Flash_DstAddr.endPoint = SAMPLEAPP_ENDPOINT;
SampleApp_Flash_DstAddr.addr.shortAddr = addr;
```