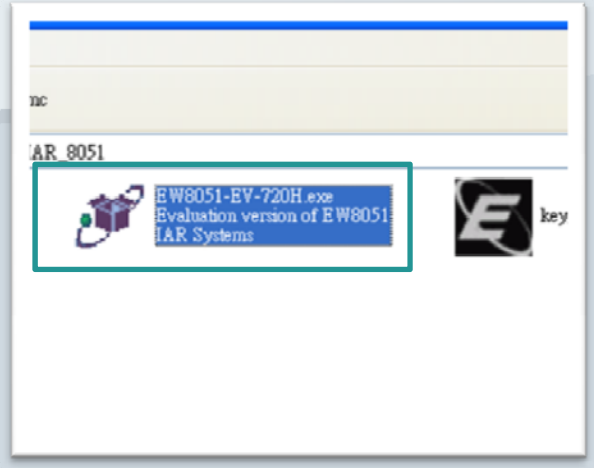
無線感測網路及應用系統技術

單元2: 無線感測網路與程式開發

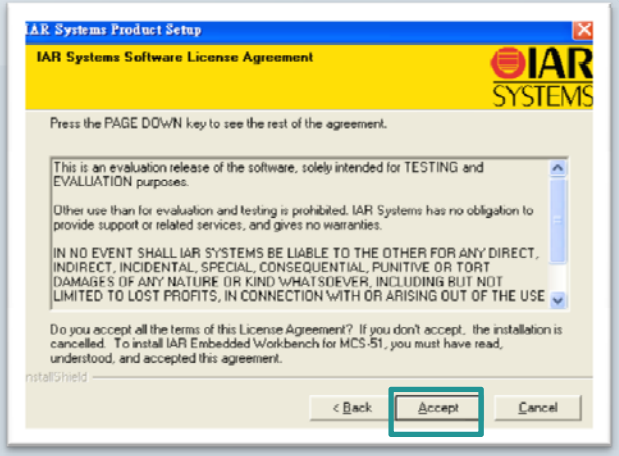
單元2: 無線感測網路與程式開發

- 安裝IAR軟體
- 安裝Z-stack軟體
- 創建使用者Z-Stack 應用專案
- 基本開發程式架構
- 重要函式介紹

安裝IAR軟體(1/4)

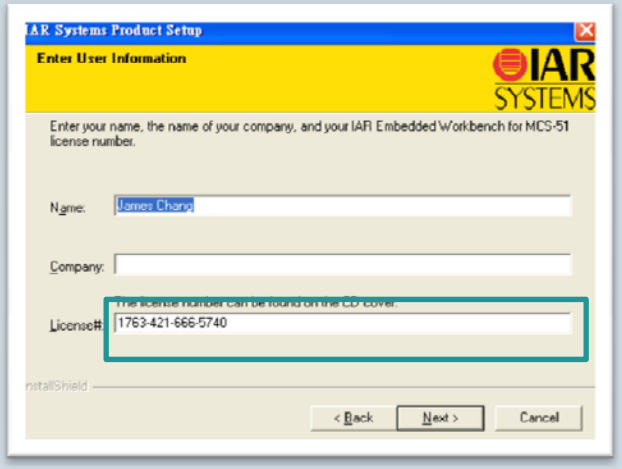


點選 EW8051-EV-720H.exe軟體



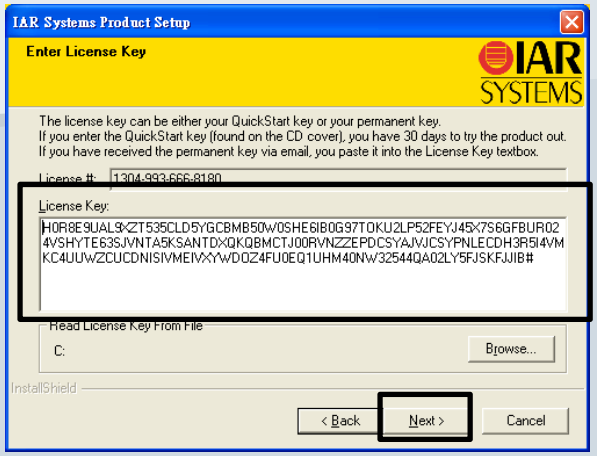
點選 Accept

點選Next

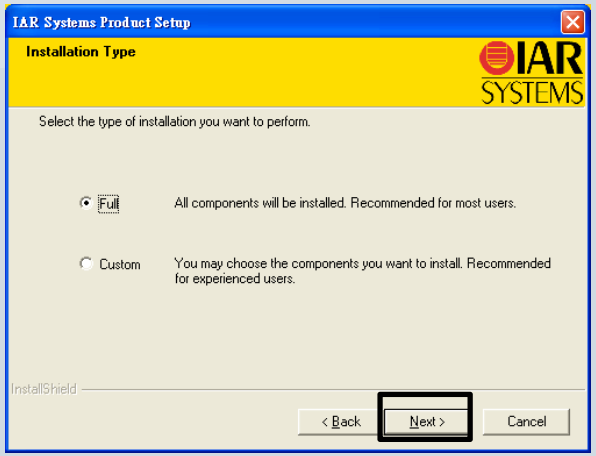


寫入正版 License number

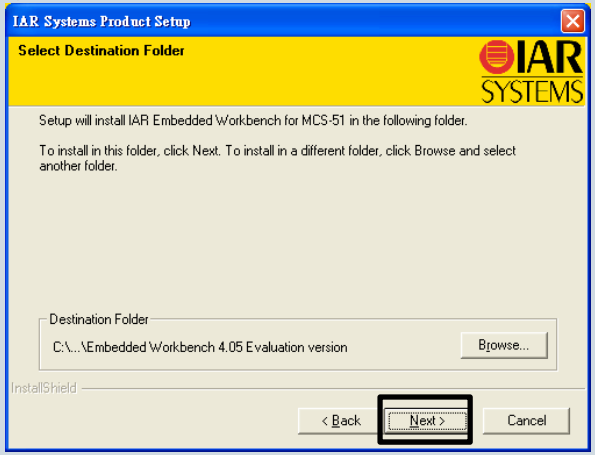
安裝IAR軟體(3/4)



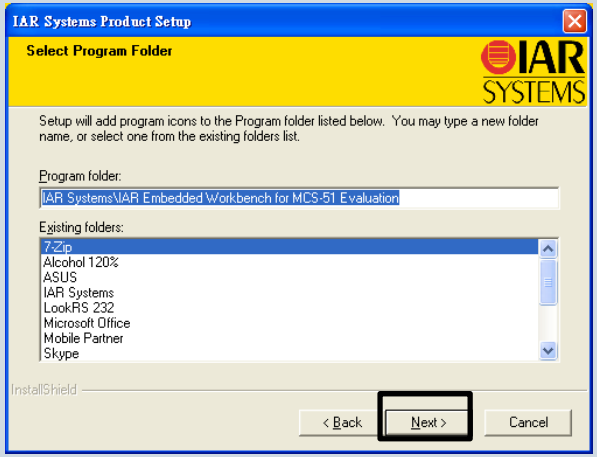
寫入正版 License key，
填入後按下Next鍵



點選Next鍵

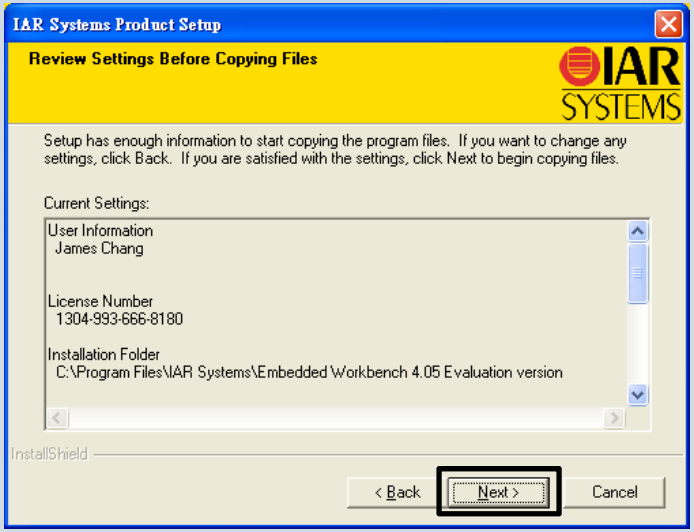


點選Next鍵



點選Next鍵

安裝IAR軟體(4/4)

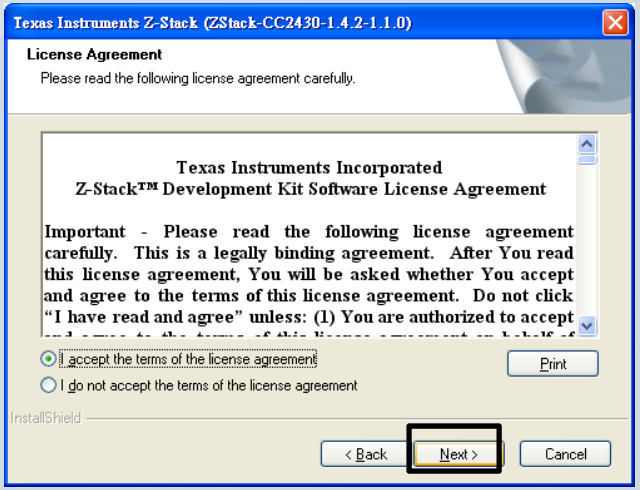


點選Next鍵

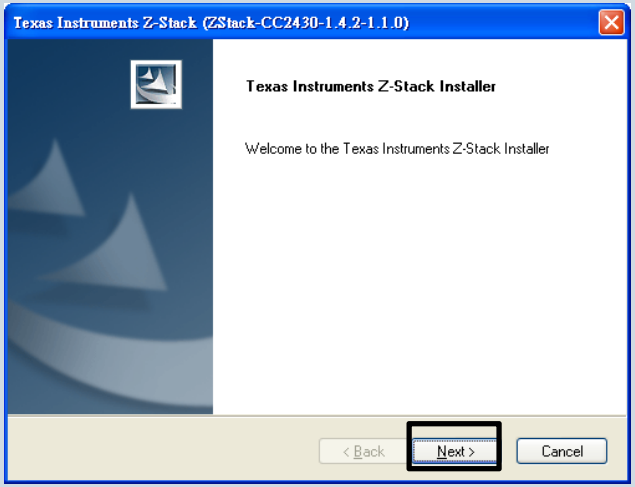
點選Finish鍵後為完成安裝

安裝Z-stack軟體(1/2)

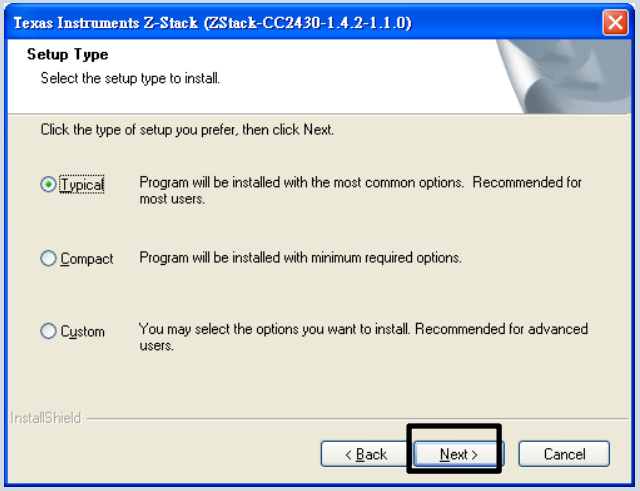
點選Z-Stack-CC2430-1.4.2-1.1.0.exe



選擇接受後點選Next

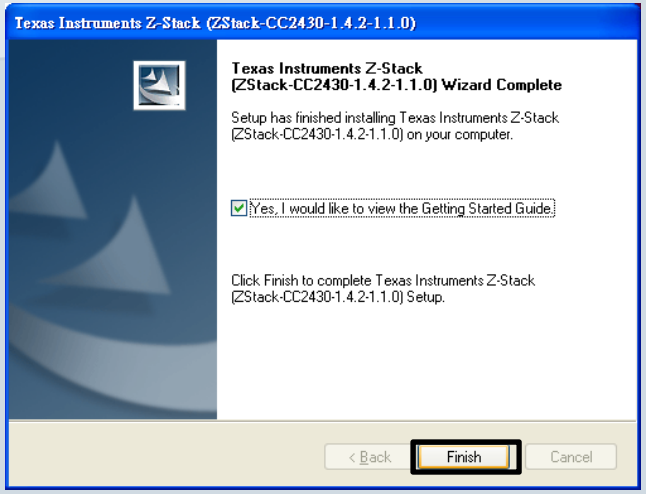


點選Next

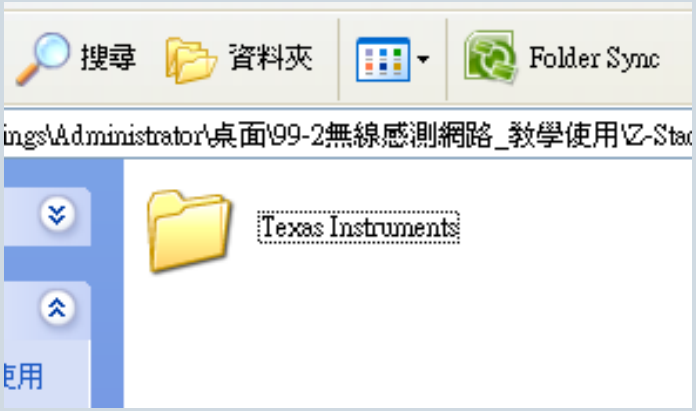


點選Next

安裝Z-stack軟體(2/2)



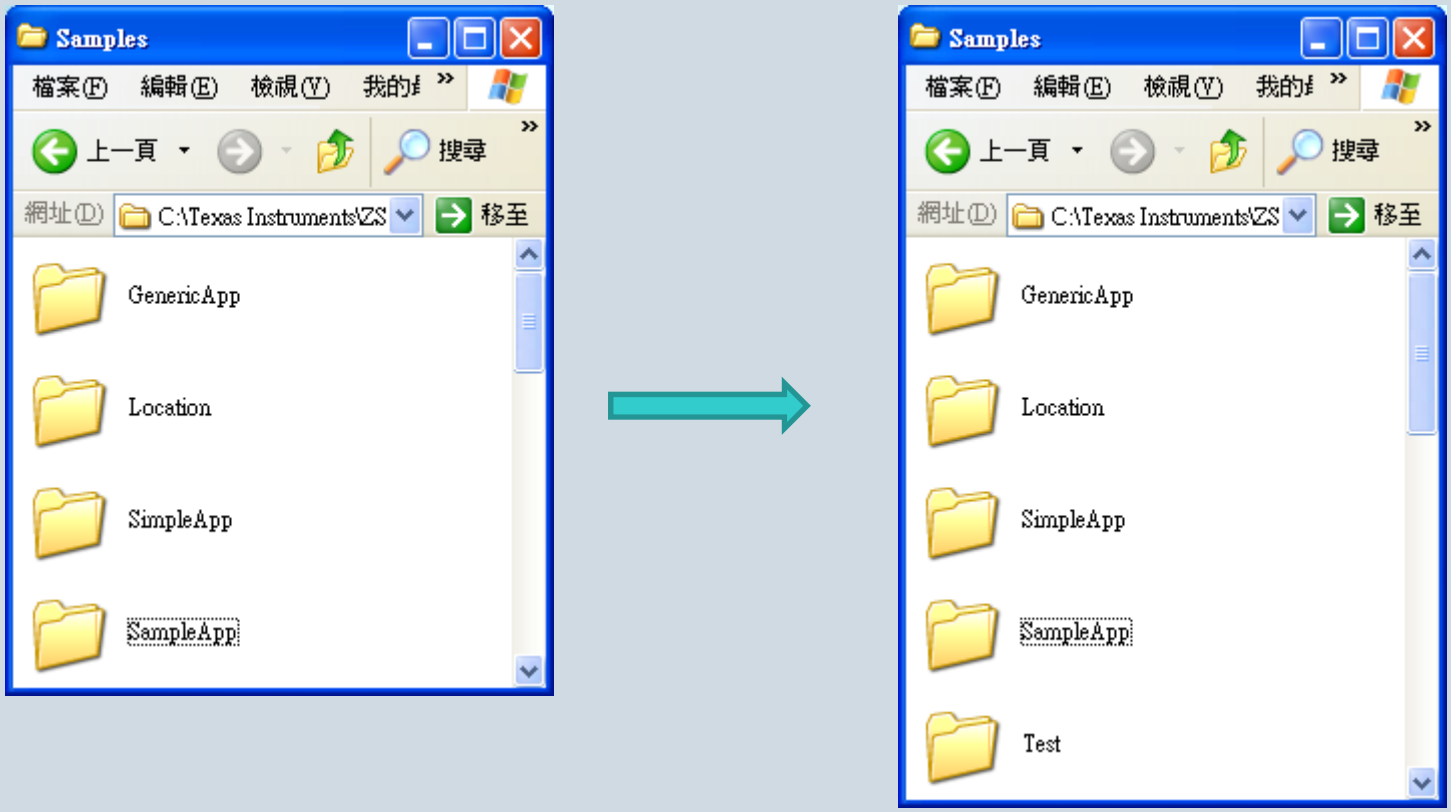
點選Finish



將Texas Instruments資料夾複製後取代C槽根目錄下的TI 資料夾

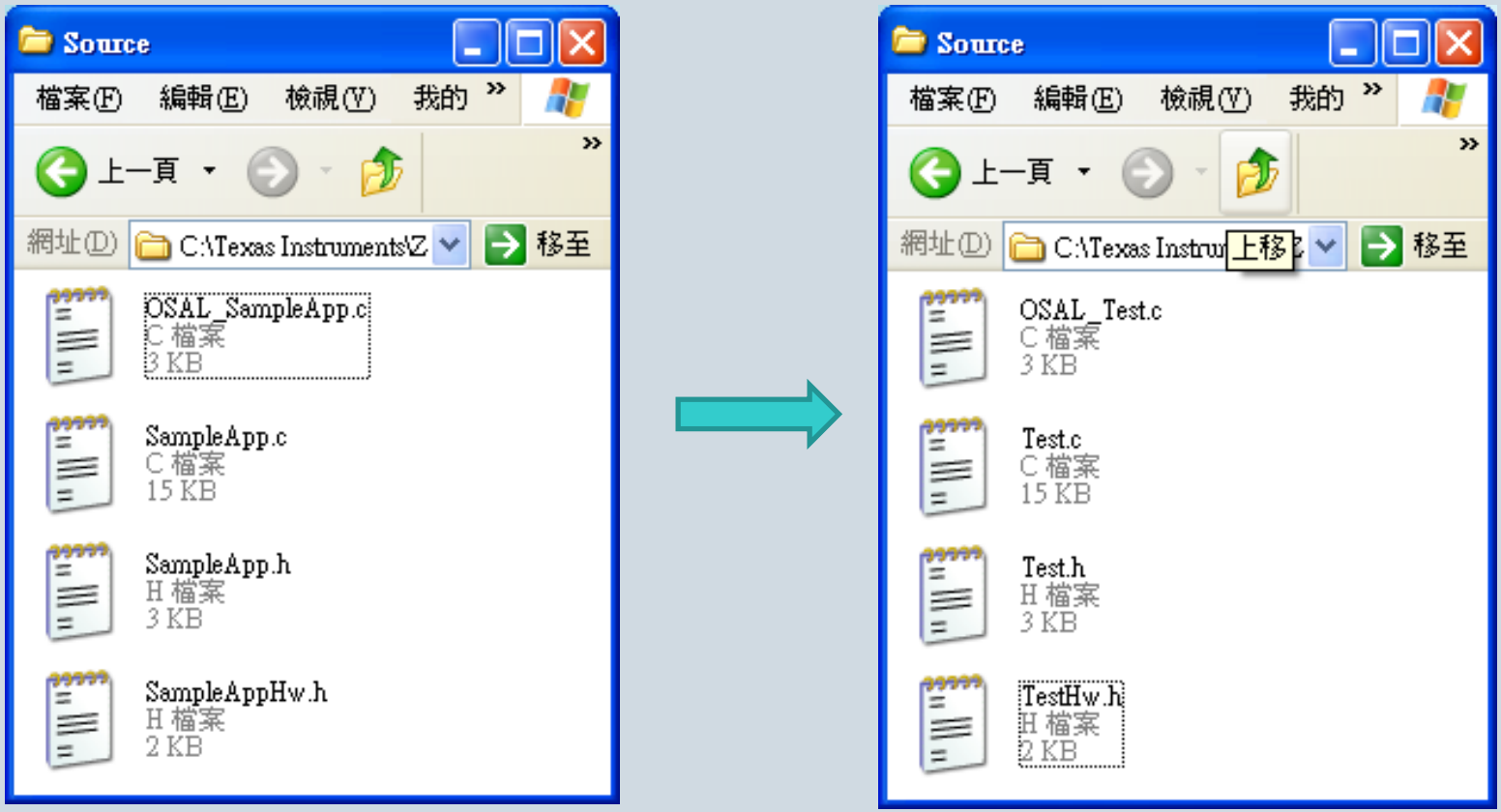
創建使用者Z-Stack 應用專案(1/11)

1. 選擇一個你想要使用的範例類型(假設應用範例類型為SampleApp)
2. 複製後更改名稱資料夾名稱(假設專案夾名稱為Test)



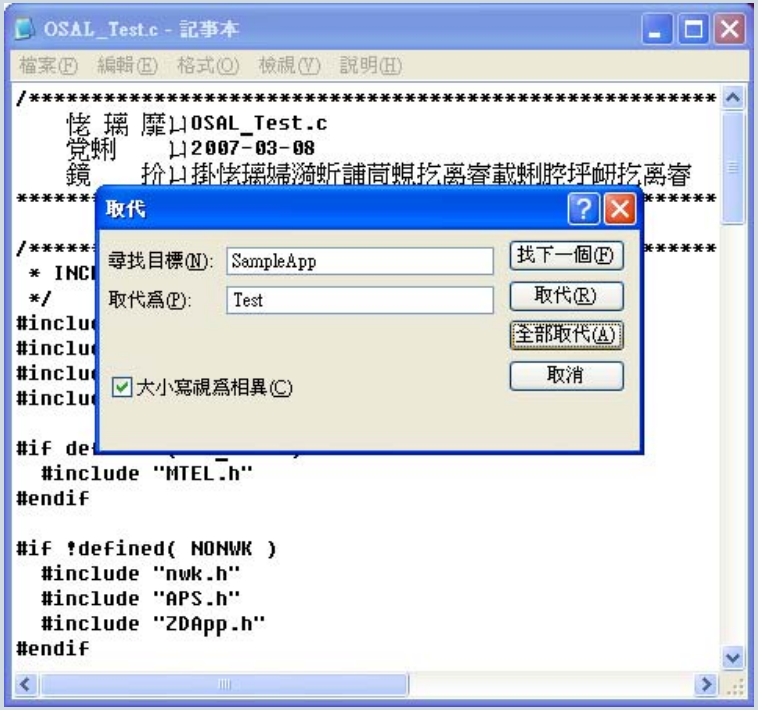
創建使用者Z-Stack 應用專案(2/11)

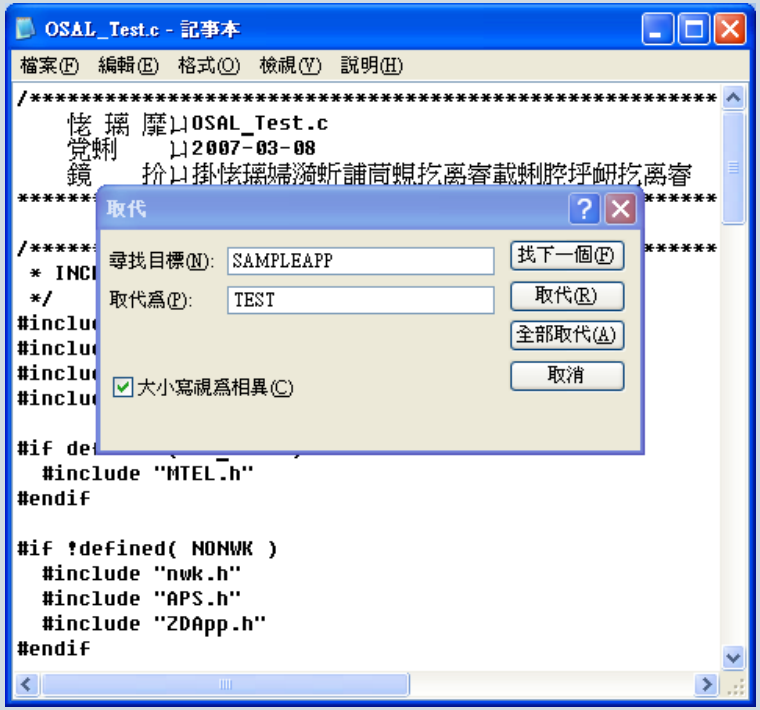
1. 打開Test 文件夾中的Source文件夾
2. 更改檔案名稱，用Test替換SampleApp



創建使用者Z-Stack 應用專案(3/11)

- 用記事本打開Source夾的OSAL_Test.c檔案，將檔案的SampleApp字樣全部替換為Test，SAMPLEAPP字樣全部替換為TEST



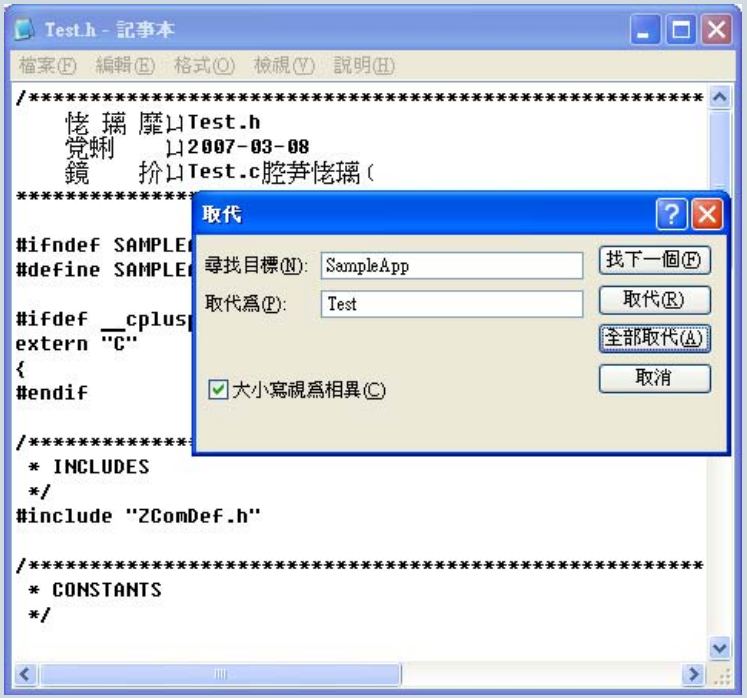


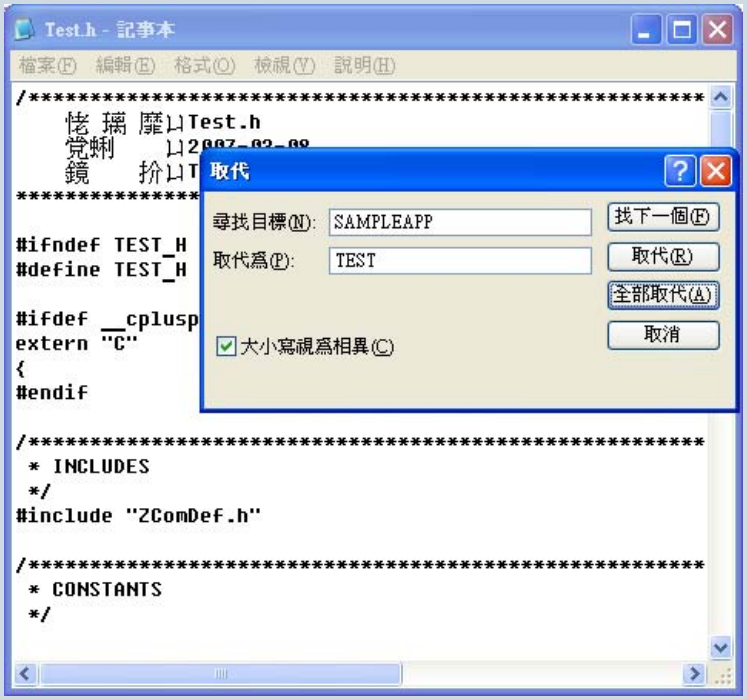
創建使用者Z-Stack 應用專案(4/11)

- 用記事本打開Source夾的Test.c檔案，將檔案的SampleApp字樣全部替換為Test，SAMPLEAPP字樣全部替換為TEST

創建使用者Z-Stack 應用專案(5/11)

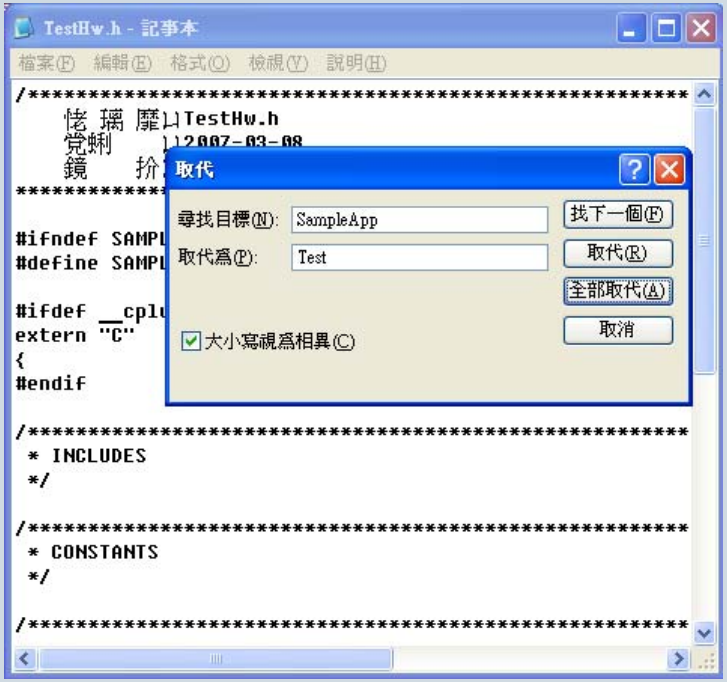
- 用記事本打開Source夾的Test.h檔案，將檔案的SampleApp字樣全部替換為Test，SAMPLEAPP字樣全部替換為TEST

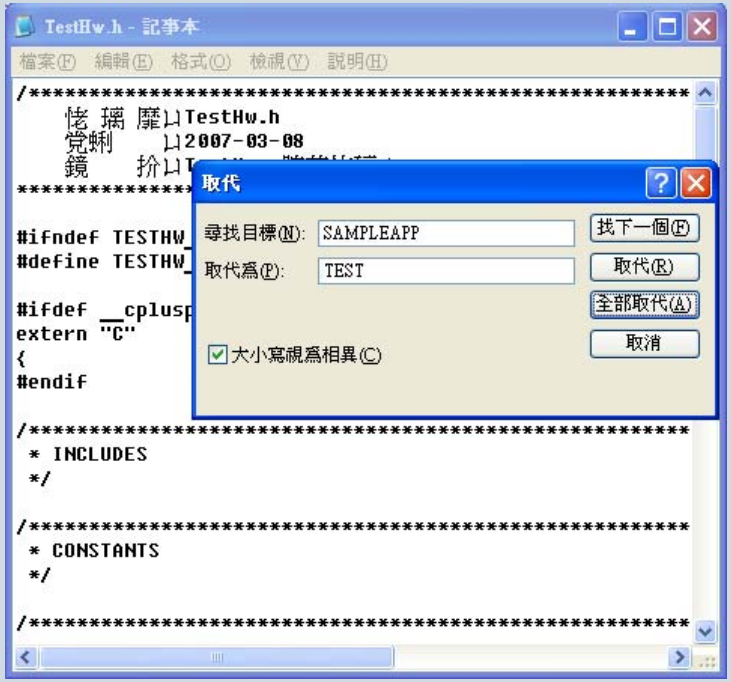




創建使用者Z-Stack 應用專案(6/11)

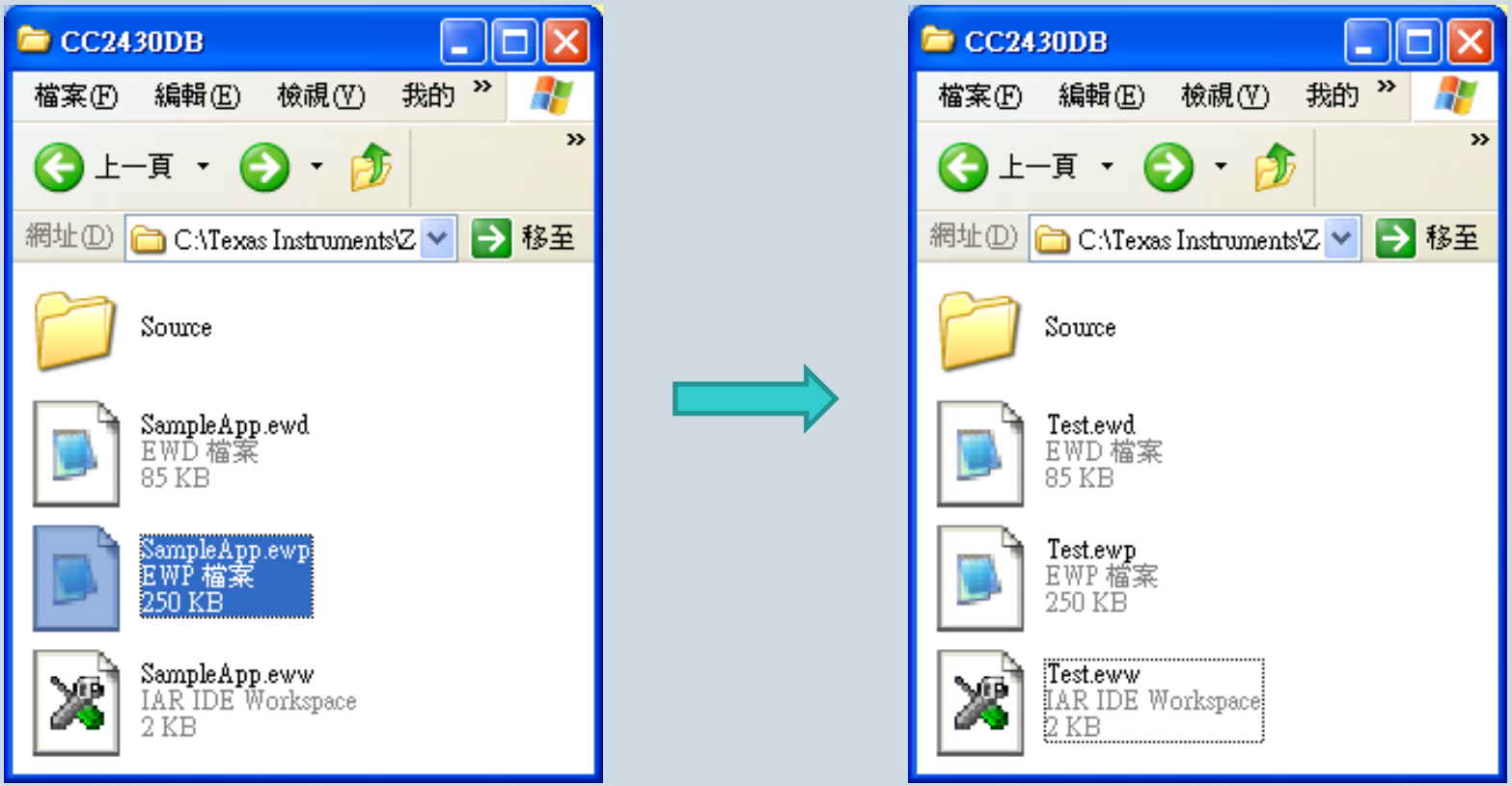
- 用記事本打開Source夾的TestHw.h檔案，將檔案的SampleApp字樣全部替換為Test，SAMPLEAPP字樣全部替換為TEST





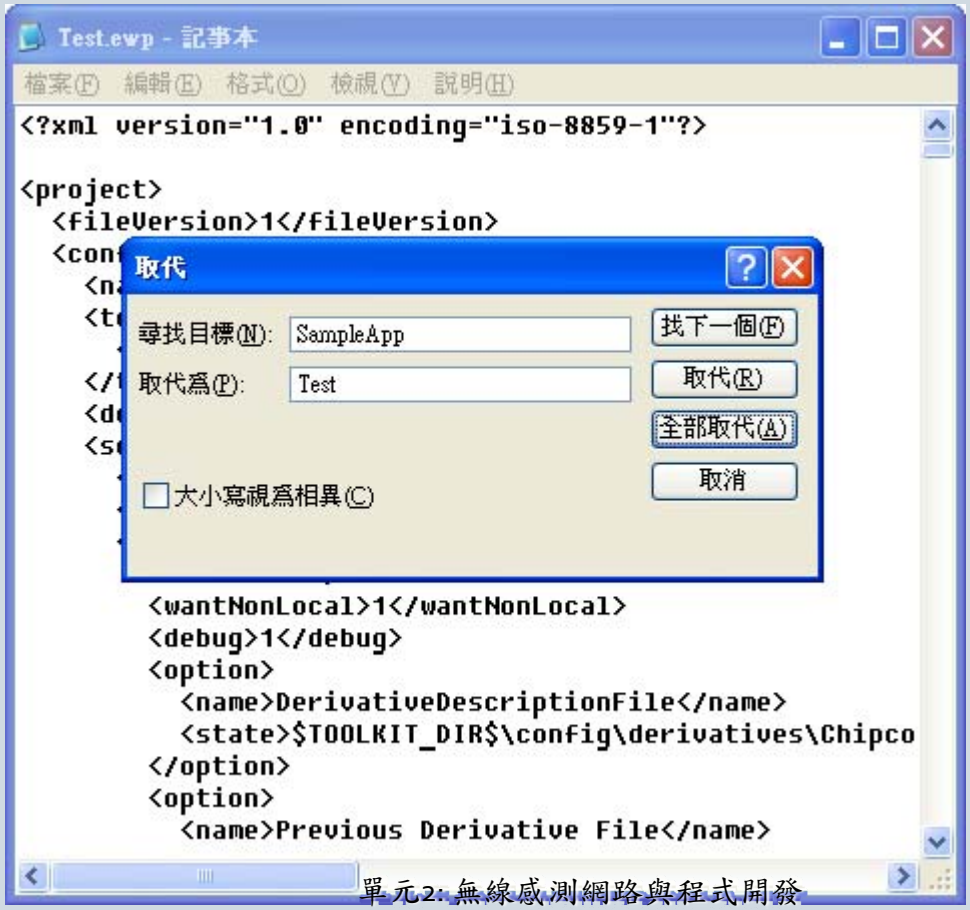
創建使用者Z-Stack 應用專案(7/11)

1. 打開Test 文件夾中的CC2430DB文件夾
2. 更改檔案名稱，用Test替換SampleApp



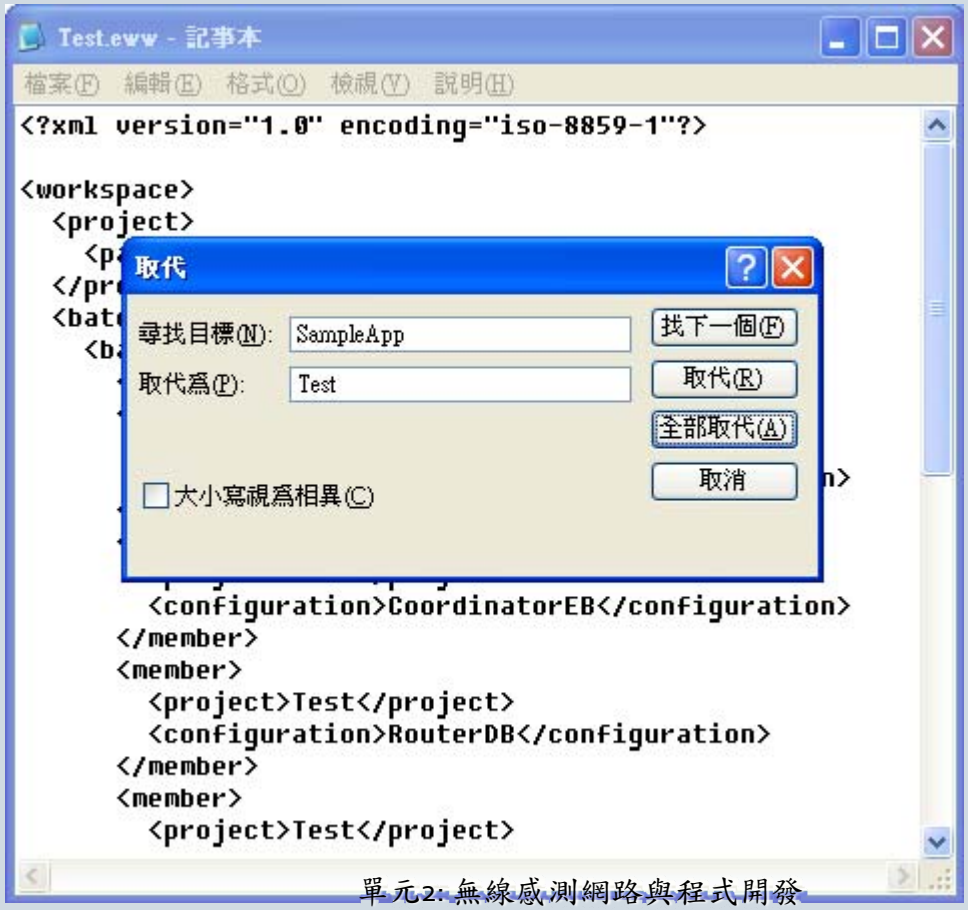
創建使用者Z-Stack 應用專案(8/11)

- 用記事本打開CC2430DB文件夾的Test.ewp檔案，將檔案的SampleApp字樣全部替換為Test



創建使用者Z-Stack 應用專案(9/11)

- 用記事本打開CC2430DB文件夾的Test.eww檔案，將檔案的SampleApp字樣全部替換為Test

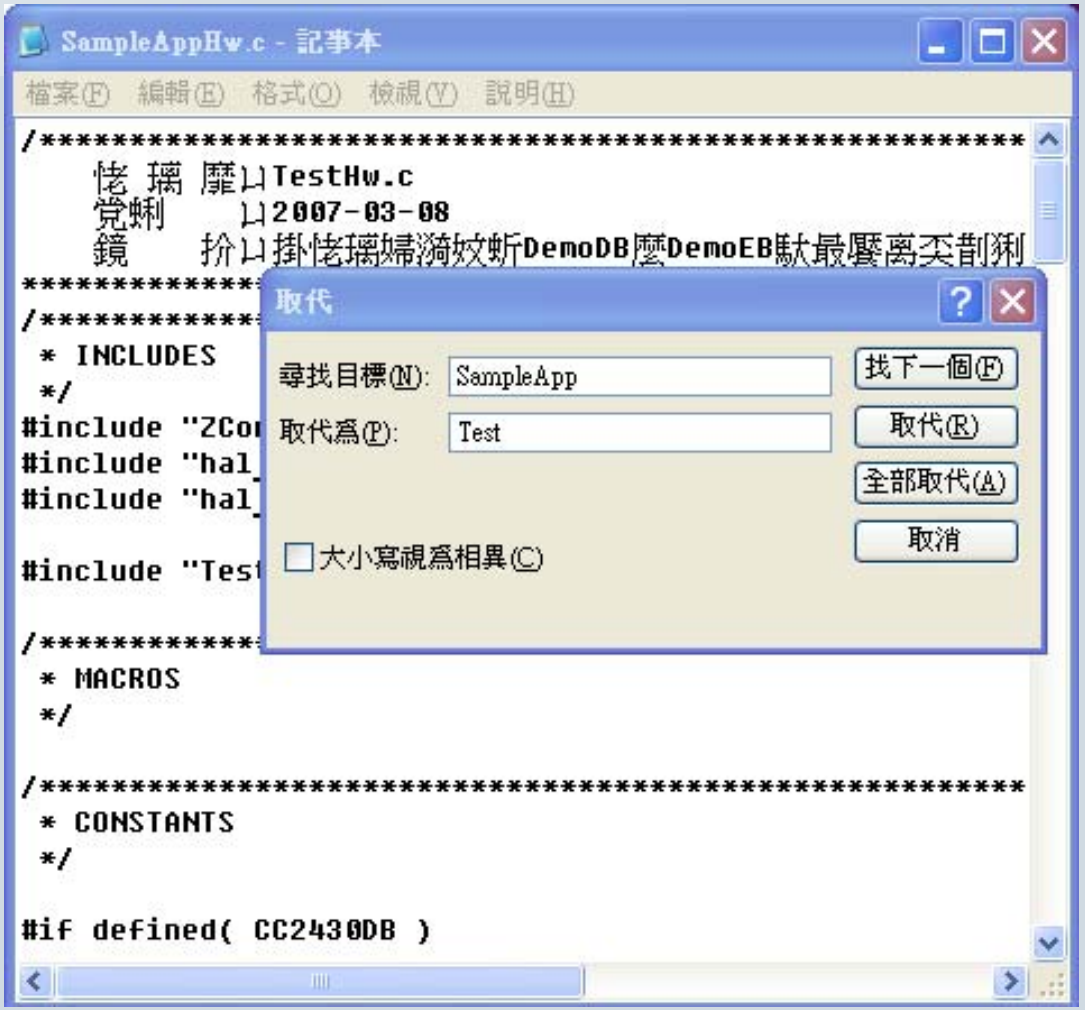


創建使用者Z-Stack 應用專案 (10/11)

1. 打開CC2430DB文件夾中的Source文件夾
2. 更改檔案名稱，用Test替換SampleApp

創建使用者Z-Stack 應用專案 (11/11)

- 用記事本打開Source文件夾的TestHw.c檔案，將檔案的SampleApp字樣全部替換為Test



基本開發程式架構

- 基本事件介紹(SYS_EVENT_MSG)：

  在ZComDef.h中所定義的系統事件訊息，這些事件會依據以下所收到的不同事件而有不同的處理。

1. AF_DATA_CONFIRM_CMD
2. AF_INCOMING_MSG_CMD
3. KEY_CHANGE
4. ZDO_NEW_DSTADDR
5. ZDO_STATE_CHANGE

基本開發程式架構

- AF_DATA_CONFIRM_CMD

  這個事件訊息是用來回報程式所發出的要求(request)是否成功完成。如果成功完成，程式會回報Zsuccess，用來確認data request成功傳送。

- AF_INCOMING_MSG_CMD

  專門用於處理接收資料、訊息的事件型態。

- KEY_CHANGE

  專門用於處理device上按下按鈕的事件。

- ZDO_NEW_DSTADDR

  專門用於回覆Match Descriptor Request的事件型態。

- ZDO_STATE_CHANGE

  專門用於處理網路狀態改變的事件型態。

  EX: 當Router \ End device加入網路後，則網路狀態改變。

基本開發程式架構

SampleApp範例解說

- ProjectName_Init()

  程式一開始的初始化函式，在程式一開始執行時，註冊各種事件。

- ProjectName_ProcessEvent()

  程式處理不同事件的函式，當程式遇到不同事件的發生時，這個函式會被呼叫來處理這些task，這些事件包含timer、messages以及其他各自定義的事件等等。

- ProjectName_HandleKeys()

  當偵測到硬體上得按鈕狀態發生改變時，負責處理這類事件的函式。

- ProjectName_MessageMSGCB

基本開發程式架構SampleApp_Init()介紹

目的地址的資料結構

```
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 );
}
```

傳送模式-廣播

短地址

傳送模式-群組

群組名稱

節點加入的群組

基本開發程式架構SampleApp_Init()介紹

Test.c

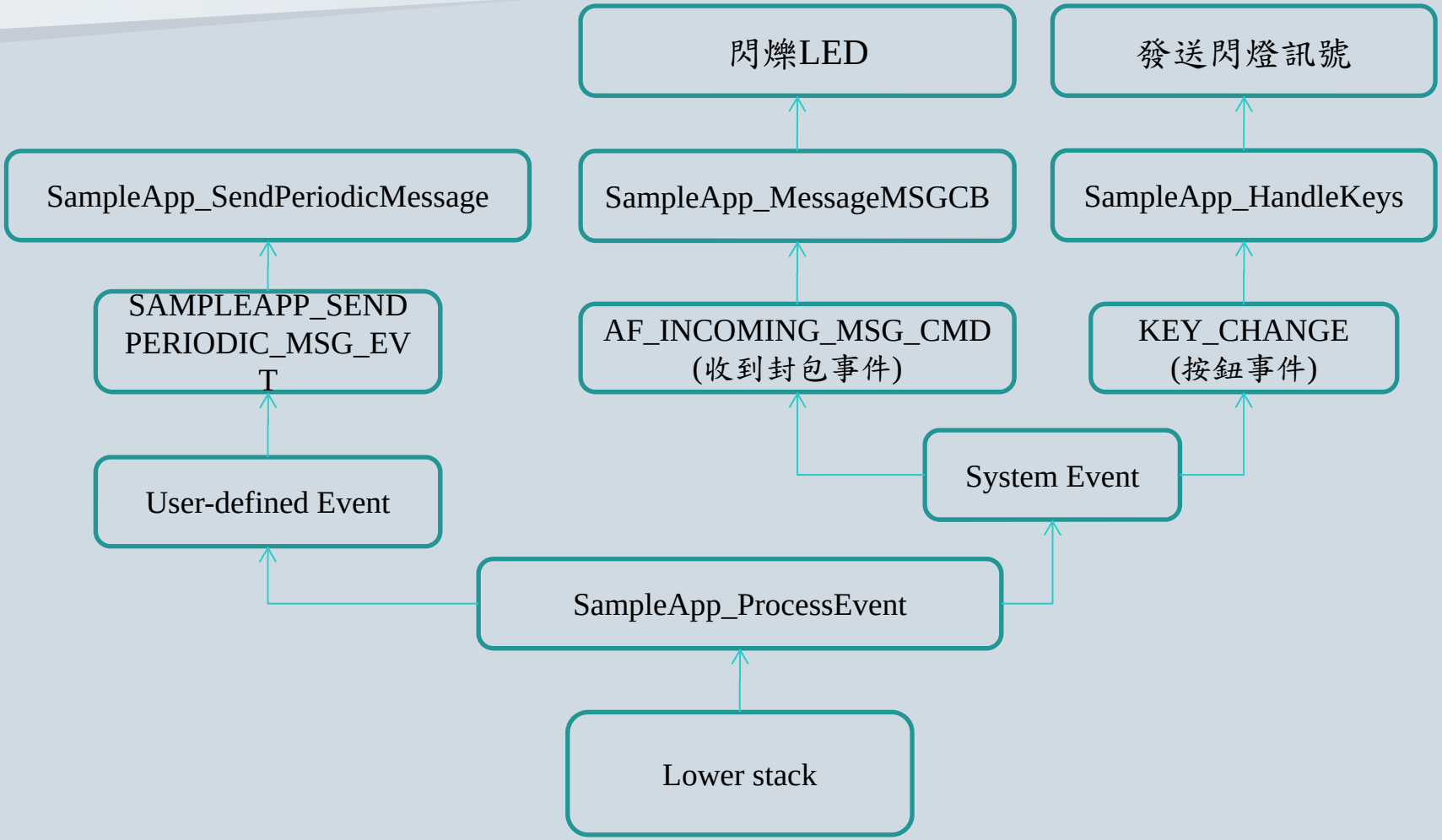
```
uint16 Test_ProcessEvent( uint8 task_id, uint16 events )
{
  afIncomingMSGPacket_t *MSGpkt;

  if ( events & SYS_EVENT_MSG )
  {
    MSGpkt = (afIncomingMSGPacket_t *)osal_msg_receive( Test_TaskID );
    while ( MSGpkt )
    {
      switch ( MSGpkt->hdr.event )
      {
        // Received when a key is pressed
        case KEY_CHANGE:
          Test_HandleKeys( ((keyChange_t *)MSGpkt)->state, ((keyChange_t *)MSGpkt)->keys );
          break;

        // Received when a messages is received (OTA) for this endpoint
        case AF_INCOMING_MSG_CMD:
          Test_MessageMSGCB( MSGpkt );
          break;

        // Received whenever the device changes state in the network
        case ZDO_STATE_CHANGE:
          Test_NwkState = (devStates_t)(MSGpkt->hdr.status);
          if ( (Test_NwkState == DEV_ZB_COORD)
              || (Test_NwkState == DEV_ROUTER)
              || (Test_NwkState == DEV_END_DEVICE) )
          {
            // Start sending the periodic message in a regular interval.
            osal_start_timerEx( Test_TaskID,
                                TEST_SEND_PERIODIC_MSG_EVT,
                                TEST_SEND_PERIODIC_MSG_TIMEOUT );
          }
          else
          {
            // Device is no longer in the network
          }
          break;

        default:
          break;
      }
```

按下按鈕

接收到訊號

加入網路後，狀態改變

基本開發程式架構 SAMPLEAPP範例程式架構



重要函式介紹 任務同步應用

- byte osal_set_event(byte task_id, uint16 event_flage)
  - 該函數被用來執行事件
  - task_id 任務名稱
  - event_id 事件名稱
  - 回傳值ZSUCCESS表示成功，INVALID_TASK 表示任務無效

重要函式介紹 計時器管理應用

- byte osal_start_timerEx(byte task_id, uint16 event_id , uint16 timeout_value)
  - 該函數被調用來啟動計時器，當計時器到期時，該設定事件會被執行
  - task_id 任務名稱
  - event_id 事件名稱
  - timeout_value 等待時間以毫秒為單位
  - 回傳值ZSUCCESS表示操作成功，NO_TIMER_AVAILABLE表示任務無效

- byte osal_stop_timerEx(byte task_id, uint16 event_id)
  - 該函數被調用來停止計時器的任務
  - task_id 任務名稱
  - event_id 事件名稱
  - 回傳值ZSUCCESS表示操作成功，NO_EVENT_ID表示任務無效
  - 一但執行任務已執行，就無法使用該函式庭指任務

重要函式介紹 NV記憶體應用

- byte osal_nv_item_init(uint16 id, uint16 len, void *buf)
  - 該函數檢查NV中的一個專案是否存在。若不存在，該函數將會建立專案並將專案初始化
  - 必須加入#include "osal_nv.h“
  - id 使用者自行定義id
  - len 專案資料長度
  - *buf指向專案初始化資料的指標，若沒有資料初始化，可設為 NULL
  - 在使用osal_nv_read()或是osal_nv_write()之前，本函數必須要被專案調用
  - 回傳值為ZSUCCESS表是操作成功，NV_ITEM_UNINIT表是成功但是專案不存在，NV_OPER_FAILED表示操作失敗
  - id 可以使用的範圍為 0x0201~0x0FFF

VALUE	USER
0x0000	Reserved
0x0001 – 0x0020	OSAL
0x0021 – 0x0040	NWK
0x0041 – 0x0060	APS
0x0061 – 0x0080	Security
0x0081 – 0x00A0	ZDO
0x00A1 – 0x0200	Reserved
0x0201 – 0x0FFF	Application
0x1000 – FFFF	Reserved

重要函式介紹 NV記憶體應用

- byte osal_nv_read(uint16 id, uint16 offset, uint16 len, void *buf)
  - 從NV讀取資料，讀取的資料將被複製到*buf
  - 必須加入#include "osal_nv.h“
  - id 使用者自行定義id
  - offset 讀取指標偏移量
  - len 讀取資料長度
  - *buf 資料將被讀取到該暫存器
  - 回傳值為ZSUCCESS表是操作成功，NV_ITEM_UNINIT表是專案未被初始化，NV_OPER_FAILED表示操作失敗
- byte osal_nv_write(uint16 id, uint16 len, void *buf)
  - id 使用者自行定義id
  - len 寫入資料長度
  - *buf 寫入資料指標
  - 回傳值為ZSUCCESS表是操作成功，NV_ITEM_UNINIT表是專案未被初始化，NV_OPER_FAILED表示操作失敗

重要函式介紹 ADC應用

- uint16 HalAdcRead (uint8 channel,uint8 resolution);
  - 將類比訊號轉換為數位訊號
  - 必須加入#include “hal_adc.h“
  - Test_Init()裡加入HalAdcInit ();
  - 在需要由回傳ADC數值的地方使用

參數	描述
HAL_ADC_CHANNEL_0	Port 0 Pin0
HAL_ADC_CHANNEL_1	Port 0 Pin1
HAL_ADC_CHANNEL_2	Port 0 Pin2
HAL_ADC_CHANNEL_3	Port 0 Pin3
HAL_ADC_CHANNEL_4	Port 0 Pin4
HAL_ADC_CHANNEL_5	Port 0 Pin5
HAL_ADC_CHANNEL_6	Port 0 Pin6
HAL_ADC_CHANNEL_7	Port 0 Pin7

參數	描述
HAL_ADC_RESOLUTION_8	8位解析度
HAL_ADC_RESOLUTION_10	10位解析度
HAL_ADC_RESOLUTION_12	12位解析度
HAL_ADC_RESOLUTION_14	14位解析度

重要函式介紹 led應用

- HalLedBlink(uint8 leds, uint8 numBlinks, uint8 percent, uint8 period);
  - 該函數將基於設定參數指定LED閃爍
  - leds 指定LED的位遮罩
  - numBlinks LED將要閃爍的次數
  - percent 一個點亮/熄滅週期中點亮狀態所占的百分比
  - period 一個點亮/熄滅週期（以毫秒為單位）

LED	描述
HAL_LED_1	LED1
HAL_LED_2	LED2
HAL_LED_3	LED3
HAL_LED_4	LED4
HAL_LED_ALL	所有LED

重要函式介紹 led應用

- HalLedSet(uint8 leds, uint8 mode);
  - 該函數將設定指定的LED熄滅、點亮、閃爍及轉換LED狀態
  - leds 指定LED的位遮罩
  - mode Led模式

LED	描述
HAL_LED_1	LED1
HAL_LED_2	LED2
HAL_LED_3	LED3
HAL_LED_4	LED4
HAL_LED_ALL	所有LED

Mode	描述
HAL_LED_MODE_OFF	熄滅LED
HAL_LED_MODE_ON	點亮LED
HAL_LED_MODE_BLINK	閃爍LED一次
HAL_LED_MODE_FLASH	持續閃爍LED
HAL_LED_MODE_TOGGLE	轉換LED狀態

重要函式介紹 UART應用

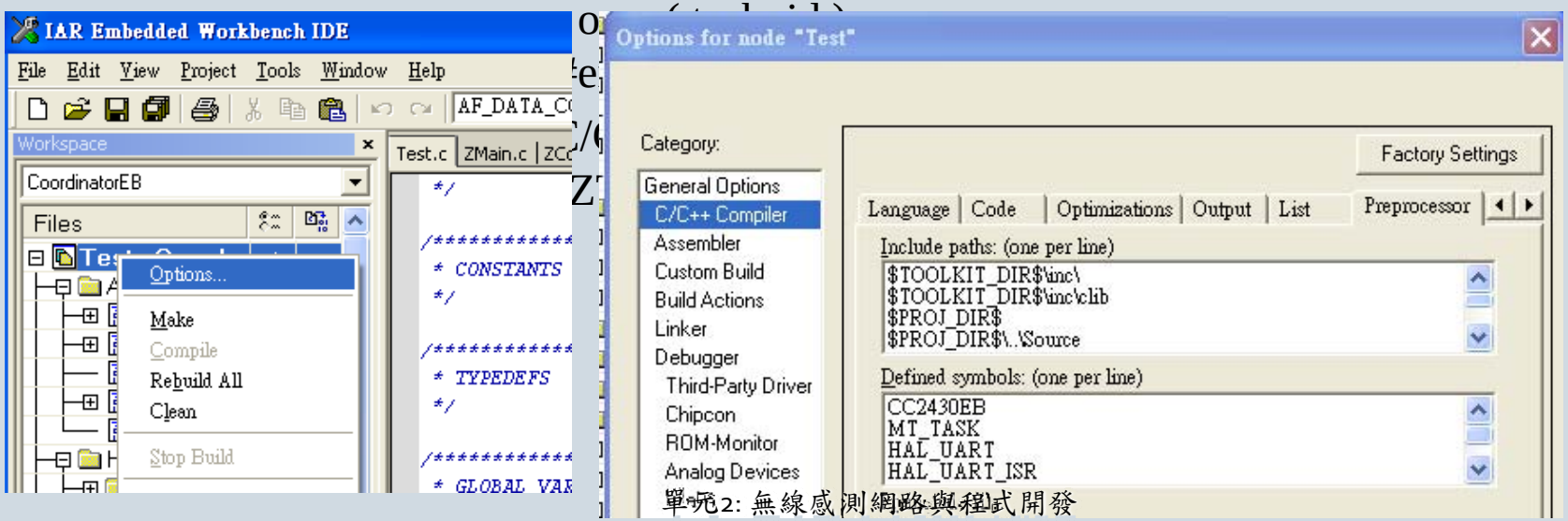
- Z-Stack UART/uart.c放入C:\Texas Instruments\ZStack-1.4.2-1.1.0\Projects\zstack\Samples\Test\Source中

- Test.c裡加入

```
#if defined( HAL_UART )
  #include "uart.c"
#endif
```

- Test_Init裡加入

```
#if defined( HAL_UART )
```



重要函式介紹 UART應用

- 打開TestHw.c將下列程式註解

#error The UART will not work with this configuration. The RX & TX pins are used.

- 在需要由uart傳回資料的地方使用

  HalUARTWrite(SERIAL_APP_PORT , data , length);

  data為資料pointer(uint8*)

  length為資料長度

- 在UART接收事件下，需要加上FREE_OTABUF()函式

```
if ( events & SEND_UARTMSG_EVT )
{
  HalUARTWrite( SERIAL_APP_PORT , otaBuf  , otaLen );
  FREE_OTABUF();
  return (events ^ SEND_UARTMSG_EVT);
}
```

重要函式介紹 UART應用

■ 更改鮑率方法

- 開啟hal_uart.c
- 到HalUARTOpen函式內將下列程式註解

  U0BAUD = (config->baudRate == HAL_UART_BR_38400) ? 59 : 216;

  U0GCR = (config->baudRate == HAL_UART_BR_38400) ? 10 : 11;
- 使用下表，根據欲使用之鮑率更改特殊暫存器U0BAUD和U0GCR內容

Baud rate (bps)	UxBAUD.BAUD_M	UxGCR.BAUD_E	Error (%)
2400	59	6	0.14
4800	59	7	0.14
9600	59	8	0.14
14400	216	8	0.03
19200	59	9	0.14
28800	216	9	0.03
38400	59	10	0.14
57600	216	10	0.03
76800	59	11	0.14
115200	216	11	0.03
230400	216	12	0.03

單元2: 無線感測網路與程式開發

- 實驗一: Ad hoc無線感測網路實驗
- 實驗二: Multiple hops無線感測網路實驗
- 實驗三: 環境品質無線感測網路實驗

- 為了使學生瞭解無線測網路技術特性，本實驗提供Z-StackAd hoc無線測網路讓學生了解使用C/C++語言控制並開發多點傳輸的Multiple hops無線感測網路
- 進一步結合CO, CO2及TVOC模組開發環境品質無線感測網路並在伺服器端或手持行動裝置顯示感測數據

實驗一: Ad hoc無線感測網路實驗

- 實驗目的
  - 利用Z-StackWSN開發系統，以C/C++語言控制、進行單一點對點的無線網路傳輸實驗
- 實驗內容
  - 在程序中建立Timer事件，以定時傳送資料
  - 伺服器端或手持行動裝置設置GUI以顯示接收到的資料

實驗二: Multiple hops無線感測網路實驗

- 實驗目的
  - 將資料經由多個節點傳達到目的地
- 實驗內容
  - 在實驗一的網路中，加入多個路由器(Router)傳送資料

實驗三: 環境品質無線感測網路實驗

- 實驗目的
  - 結合模組開發環境品質無線感測網路並在伺服器端或手持行動裝置顯示感測數據
- 實驗內容
  - 將實驗二的傳送資料改為讀取CO, CO2及TVOC模組所偵測得的感測數據，顯示於GUI
  - 學習I2C、SPI介面溝通(從模組讀取數據)

The End.

Thanks for your attention.