

HT32 MCU UART 應用範例

文件編號：AN0609TC

簡介

通用非同步收發器(Universal Asynchronous Receiver / Transmitter -- UART)為一種廣泛的串列傳輸介面，提供了一個靈活的非同步全雙工數據傳輸。

本文所提供的“Module_UART” Application Code 使用具有軟體環形緩衝區(Ring Buffer)的 TX/RX 中斷，通過 API 來進行簡單的 UART 發送/接收功能，相關功能如下列所示，以此簡化整個資料傳輸的瑣碎流程，讓使用者快速的了解與進行 UART 通訊應用。

- 發送/接收功能支援：byte read、byte write、buffer read、buffer write 等等
- 狀態功能支援：get buffer length、Tx status 等等

本文首先會介紹 UART 通訊協定，利於使用者能從原理到應用都更好的理解 UART 通訊。接著是 Application Code 所需的資源下載與準備，內容包含 Firmware Library 與應用範例程式下載和檔案及目錄配置流程，以及本文使用到的終端機軟體工具介紹。在功能說明章節，主要介紹了應用範例目錄結構、參數設定說明、API 說明以及透過“Module_UART” Application Code 流程簡述 API 使用方法，也記錄了 API 所需的 Flash / RAM 資源佔用。最後使用指南章節，則是按照步驟引導使用者進行環境準備、編譯與測試步驟，確認有正常運作，接著透過移植說明來說明如何整合 API 到用戶的專案，最後針對使用上的設定變更及常見問題來提供使用者做為參考。

縮寫說明

- UART : Universal Asynchronous Receiver / Transmitter
- API : Application Programming Interface
- LSB : Least Significant Bit
- MSB : Most Significant Bit
- PC : Personal Computer
- SK : Starter Kit，HT32 的開發板
- IDE : Integrated Development Environment

UART 通訊協定

UART 是一種串列通訊設備，它在發送端執行並列到串列數據轉換，並進行串列通訊給接收端，接收端接收到數據後，則會執行串列到並列數據轉換。圖 1 為串列通訊示意圖，其中可以看到由於串列通訊使用的方式是按位順序傳輸，因此對於發送器與接收器之間的雙向通訊，只需要兩條線(TX/RX)即可成功互相傳輸串列數據，TX 為 UART 發送端發送串列數據的腳位，會與接收端的 RX 做連接，因此兩個設備以 UART 進行雙向通訊需將 TX 與 RX 交叉連接，如圖 2 所示。

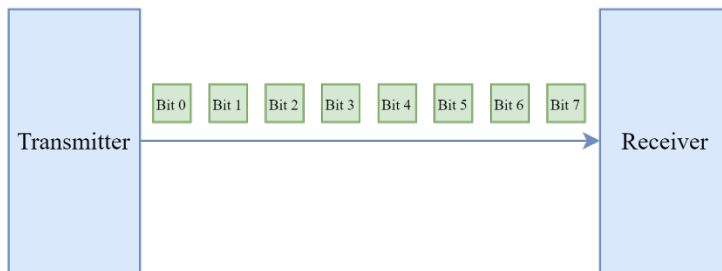


圖 1. 串列通訊示意圖

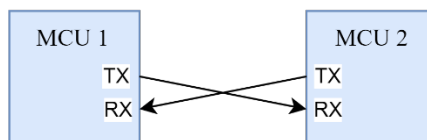


圖 2. UART 電路示意圖

在 UART 串列通訊中，數據是以非同步方式傳輸，也就是發送端與接收端之間沒有時鐘或其它同步訊號，而是使用通訊雙方事先定好的鮑率作為串列數據傳送/接收速度，並結合開始位和停止位等特殊位，加在數據封包的開頭與結尾作為數據資料，以此來組成一個完整的 UART 數據封包。圖 3 為 UART 數據封包結構圖，而圖 4 則是沒有校驗碼的 UART 8-bit 數據封包格式示意圖。

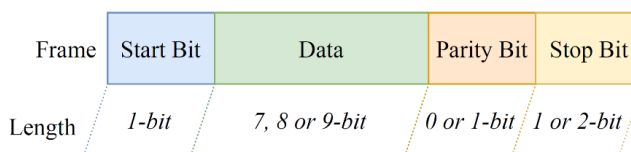


圖 3. UART 數據封包結構圖



圖 4. UART 8-bit 數據封包格式示意圖

底下依序介紹 UART 數據封包的每一部分。

- Start Bit：用來標記數據封包的開頭。當尚未開始傳輸數據時，UART TX 通常保持高電位狀態，若開始數據傳輸，UART 發送端會將 TX 從高電位拉到低電位(從 1 到 0)，並保持一個時脈週期。當 UART 接收端在 RX 檢測到這種由高至低的變化，就會開始讀取實際數據。
- Data：傳輸的實際數據。數據長度範圍可以為 7, 8 or 9-bit，數據通常以 LSB 優先方式來傳輸。

- Parity Bit：透過傳輸期間數據中值為 1 的個數來判斷是否有數據發生改變。如果為偶校驗，則數據中 1 的個數應總計為偶數；反之若為奇校驗，則數據中 1 的個數應總計為奇數。
- Stop Bit：用來標記數據封包的結尾。UART 發送端將 TX 從低電位狀態拉到高電位狀態（從 0 到 1），並保持 1 or 2-bit 時間。

接著如前所述，由於 UART 電路上沒有時鐘訊號，因此發送器與接收器之間就必須定義一個串列數據傳送/接收速度，也就是鮑率(baud rate)，透過該速度來進行傳送/接收，以實現無差錯傳輸。鮑率是以每秒傳幾個 bit 的方式作為定義，單位：bps (bit per sec)，標準常用的鮑率有 4800bps、9600bps、19200bps、115200bps 等，轉換成一個 bit 的傳輸所需時間如表 1 所示。

鮑率	1-bit 的傳輸所需時間
4800bps	208.33μs
9600bps	104.16μs
19200bps	52.08μs
115200bps	8.68μs

表 1. 鮑率對應 1-bit 的傳輸所需時間表

資源下載與準備

本章節介紹使用到的範例程式與軟體工具，並說明如何對目錄及檔案路徑進行配置。

Firmware Library

使用範例前，請先確認已下載最新的 Holtek HT32 Firmware Library。下載位置如下方連結，可以選擇 HT32 不同系列的 Firmware Library，例如：HT32F5xxxx 系列(HT32_M0p_Vyyyyymmdd.zip)或 HT32F1xxxx 系列(HT32_M3_Vyyyyymmdd.zip)等，下載後請將其解壓縮。

壓縮檔內包含多個資料夾，可分類為 Document、Firmware Library、Tools 等項目，放置路徑以圖 5 作為舉例。在 Firmware_Library 資料夾內會看到 HT32 Firmware Library 壓縮檔，檔名為 HT32_STD_5xxxx_FWLib_V1.0.24_5734.zip。

下載路徑：<https://mcu.holtek.com.tw/ht32/resource/>

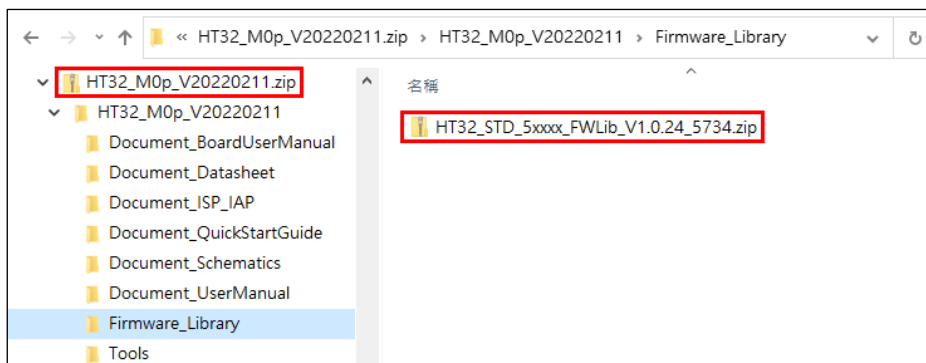


圖 5. HT32_M0p_Vyyyyymmdd.zip 內容

應用範例程式

範例程式請透過下方連結下載，此範例程式被打包成一個 ZIP 壓縮檔，檔名為 HT32_APPFW_XXXXX_APPCODENAME_Vm.n.r.s.zip。檔案名稱格式說明，請看下圖 6。

下載路徑：https://mcu.holtek.com.tw/ht32/app.fw/Module_UART/

HT32_APPFW_XXXXX_APPCODENAME_Vm.n.r.s.zip

Holtek 32-bit MCU	SVN Repository Revision (s)
Application Code	Firmware Release Version (m.n.r)
Code Type (5 digits)	Application Code Name

Non-detailed list.

XXXXX All Series
1XXXX M3 Series
5XXXX M0+ Series
...

圖 6. 範例程式檔名介紹

檔案及目錄配置

由於 Application Code 範例程式不包含 HT32 Firmware Library 相關檔案，因此在開始編譯之前，需要將 Application Code 及 Firmware Library 解壓縮的檔案放置到正確的路徑。

Application Code 壓縮檔內通常包含 1 個至數個資料夾(例如：application、library)，放置路徑以圖 7 作為舉例，只需將 application 等資料夾放到 HT32 Firmware Library 根目錄下，就可以完成檔案路徑的配置，以圖 8 作為舉例。

您也可以將 Application Code 及 HT32 Firmware Library 兩個壓縮檔，同時解壓縮到相同路徑，此操作具有同樣的效果。

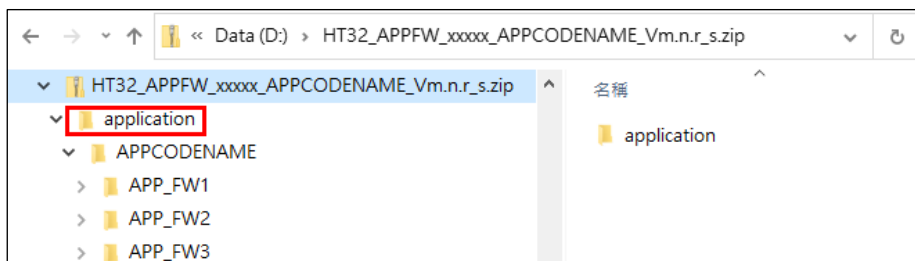


圖 7. HT32_APPFW_XXXXX_APPCODENAME_Vm.n.r.s.zip 內容

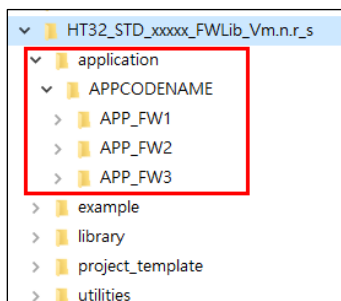


圖 8. 解壓縮路徑

終端機軟體

Application Code 範例程式可以透過 COM Port 傳遞訊息，作為功能選擇或狀態顯示。而主機端需要事先安裝終端機軟體，用戶可以選擇適合的連線軟體，或者使用免費授權軟體，例如 Tera Term。

範例程式中 UART 介面配置，字長 8-bit，無奇偶校驗，1 個停止位元，串列傳輸速率設為 115200bps。

功能說明

本章節介紹 Application Code 的功能說明，包含目錄結構、API 架構、設定說明等資訊。

目錄結構

Application Code 內包含資料夾 application，下一層為“Module_UART” Application Code 資料夾，其中含有“UART_Module_Example”與“UART_Bridge”兩種範例程式，相關檔案及說明如下表。

資料夾/檔案名稱	說明
\Application\Module_UART\UART_Module_Example ¹	
_CreateProject.bat	創建專案文件的批處理腳本
_ProjectSource.ini	添加源代碼文件於專案之下的初始化文件
ht32_board_config.h	用於 IC 週邊 I/O 分配相關設定文件
ht32fxxxxx_01_it.c	包含中斷服務程式文件
main.c	主程式的源代碼文件
\Application\Module_UART\UART_Bridge ²	
_CreateProject.bat	創建專案文件的批處理腳本
_ProjectSource.ini	添加源代碼文件於專案之下的初始化文件
ht32_board_config.h	用於 IC 週邊 I/O 分配相關設定文件
ht32fxxxxx_01_it.c	包含中斷服務程式文件
main.c	主程式的源代碼文件
uart_bridge.h	UART bridge 標頭文件和源代碼文件
uart_bridge.c	
\Utilities\middleware	
uart_module.h ³	API 標頭文件和源代碼文件
uart_module.c ³	
\Utilities\common	
ring_buffer.h	軟體環形緩衝區標頭文件和源代碼文件
ring_buffer.c	

表 2. 範例程式目錄結構

Note: 1. “UART_Module_Example” Application Code 是以 Loopback 的方式來進行 API 讀與寫操作範例，詳細說明請參考“API 使用實例”章節。

2. “UART_Bridge” Application Code 則是在演示透過設定開啟兩個 UART 通道(UART CH0、UART CH1)，然後將兩個 UART 通訊裝置透過 COMMAND 結構來進行自定義的通訊協定，詳細說明請參考“API 使用實例”章節。

3. Application Code 需用到 uart_module.c/h 檔案，檔案有 Firmware Library 版本要求，版本要求會依照更新而不定時變動，當前檔案 Firmware Library 版本要求，請參考 main.c 中的相依性檢查內容進行確認(搜尋關鍵字“Dependency check”)。

註：若 Firmware Library 版本不符合要求，請從“Firmware Library”章節中下載最新版本的 Firmware Library。

API 架構

各 API 必有一個重要參數 CH，此參數決定對哪一組 UART 進行控制，我們將其簡稱為 UART CH (UART Channel)，目前最多支援 4 組 UART CH，因此定義了 4 個常數符號整理如下，作為參數 CH 給 API 作為控制的依據。

- UARTM_CH0：輸入參數，代表對 UART CH0 進行控制或設定
- UARTM_CH1：輸入參數，代表對 UART CH1 進行控制或設定
- UARTM_CH2：輸入參數，代表對 UART CH2 進行控制或設定
- UARTM_CH3：輸入參數，代表對 UART CH3 進行控制或設定

若只用到 1 組 UART CH 並不會造成記憶體浪費，原因是支援 UART CH 的數量可設定，設定支援數量後會透過預處理器(Preprocessor)移除未使用的 UART CH 程式，減少記憶體的浪費。

API 架構如圖 9 所示。

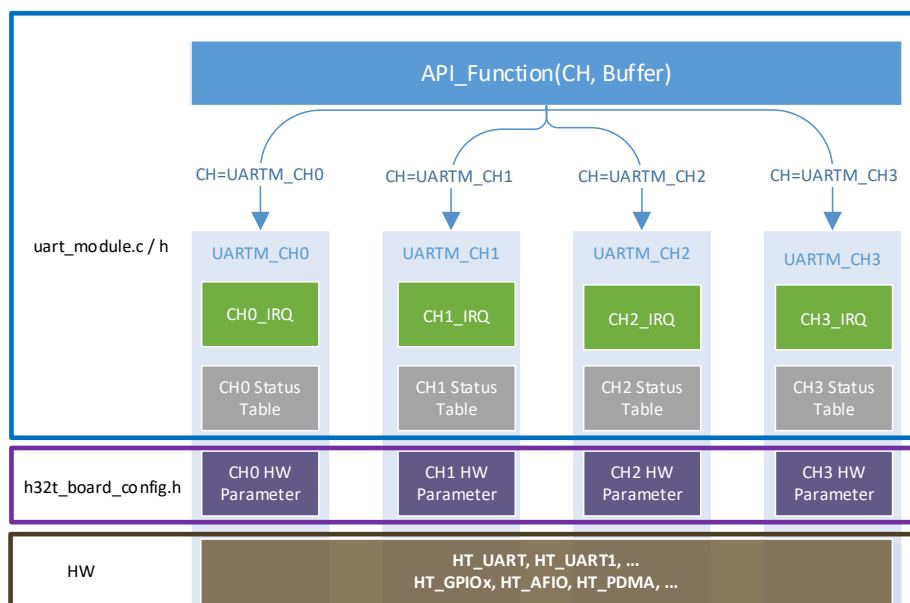


圖 9. API 架構圖

各 API 都是由 4 組 UART CH 的相關設定或控制所組成，使用者只需透過 CH 參數即可。若是需要設定相關的 API，使用者只需多提供 UART 基本配置結構體(USART_InitTypeDef)表格參數，API 會根據表格設定之參數內容進行 UART 基本配置。UART 基本配置結構體表格請參考“API 說明”章節。

uart_module.c/h 內僅有各組 UART 的中斷(CHx_IRQ)與狀態表格(CHx Status)，而 UART 通訊所需的一切設定都是參考 ht32_board_config.h，其中 ht32_board_config.h 硬體相關參數如下所示，ht32_board_config.h 設定說明請參考“設定說明”小節。

ht32_board_config.h：硬體相關參數，包括：I/O 設定、實體 UART Port 設定。定義的符號整理如下表。

符號	說明
HTCFG_UARTM_CH0	實體 UART Port 名稱，Example：UART0, UART1 ...
HTCFG_UARTM0_TX_GPIO_PORT	定義 CH0 的 TX 的 Port 名稱，Example：A, B, C ...
HTCFG_UARTM0_TX_GPIO_PIN	定義 CH0 的 TX 的 Pin 編號，Example：0~15
HTCFG_UARTM0_RX_GPIO_PORT	定義 CH0 的 RX 的 Port 名稱，Example：A, B, C ...
HTCFG_UARTM0_RX_GPIO_PIN	定義 CH0 的 RX 的 Pin 編號，Example：0~15
HTCFG_UARTM0_TX_BUFFER_SIZE	定義 CH0 的 TX 的 Buffer 大小，Example：128
HTCFG_UARTM0_RX_BUFFER_SIZE	定義 CH0 的 RX 的 Buffer 大小，Example：128

表 3. ht32_board_config.h 內定義符號表

要修改 UART 通道的 AFIO 設定請參考各型號的 Datasheet，而目前 ht32_board_config.h 文件中設定 UART CH0，因此只有 UART CH0 的 I/O 定義生效，如果要增加 UART CH1~3 需完成各自的 I/O 定義，具體做法可參考 UART CH0 的寫法，或是參考“設定變更及常見問題”章節。

API 架構特色可整理如下面 3 點。

1. 最多可支援 4 個 UART CH，代入參數為 UARTM_CH0、UARTM_CH1、UARTM_CH2、UARTM_CH3。
2. UART CH 數量可設定，並且未使用到的 UART CH 不會造成記憶體浪費。
3. UART 全部的設定與 I/O 全部定義完全與 API 分離，可增加設定值的管理方便性，減少設定錯誤或遺漏設定的風險。

設定說明

這邊為設定文件的設定參數說明，分別有 ht32_board_config.h 與 uart_module.h。

1. ht32_board_config.h，此文件用於引腳分配和各個開發板的相關設定，在這文件中設定所使用的 Starter Kit (SK) 的 UART IP 通道(UART0、UART1、USART0 ...)、相應 TX / RX 腳位與 TX / RX Buffer Size 定義，圖 10 為文件中 SK 為 HT32F52352 時的設定內容，依據開發上的功能整合使用，使用者可以參考使用型號之 Datasheet 中的“Pin Assignment”章節來進行腳位規劃設定，詳細設定變更介紹可以於“設定變更及常見問題”章節看到。

```

111 #if defined(USE_HT32F52352_SK)
112     #define HTCFG_UARTM_CH0 ..... USART1
113     #define HTCFG_UARTM0_TX_GPIO_PORT ..... A
114     #define HTCFG_UARTM0_TX_GPIO_PIN ..... 4
115     #define HTCFG_UARTM0_RX_GPIO_PORT ..... A
116     #define HTCFG_UARTM0_RX_GPIO_PIN ..... 5
117     #define HTCFG_UARTM0_TX_BUFFER_SIZE ..... 128
118     #define HTCFG_UARTM0_RX_BUFFER_SIZE ..... 128
119
120     // #define HTCFG_UARTM_CH1 ..... USART0
121     // #define HTCFG_UARTM1_TX_GPIO_PORT ..... A
122     // #define HTCFG_UARTM1_TX_GPIO_PIN ..... 2
123     // #define HTCFG_UARTM1_RX_GPIO_PORT ..... A
124     // #define HTCFG_UARTM1_RX_GPIO_PIN ..... 3
125     // #define HTCFG_UARTM1_TX_BUFFER_SIZE ..... 128
126     // #define HTCFG_UARTM1_RX_BUFFER_SIZE ..... 128
127 #endif

```

圖 10. ht32_board_config.h 設定(以 HT32F52352 為例)

- uart_module.h 則為 Application Code 使用的 API 的標頭文件，用來設定相關的默認設定、以及函數定義等，默認設定這個機制寫法有個特性，會被外部配置覆蓋(外部配置例如：“ht32_board_config.h”文件中的設定)，如圖 11 所示。

```

41 #ifndef HTCFG_UARTM_CH0
42 # /* !!! NOTICE !!!
43  * Default Setting which will be override by configuration outside (such as "ht32_board_config.h"),
44  * shall be modified according to the device you are using.
45  */
46 #define HTCFG_UARTM_CH0 HTCFG_UARTM_CH0_USART0
47 #define HTCFG_UARTM0_TX_GPIO_PORT HTCFG_UARTM0_TX_GPIO_PORT_A
48 #define HTCFG_UARTM0_TX_GPIO_PIN HTCFG_UARTM0_TX_GPIO_PIN_4
49 #define HTCFG_UARTM0_RX_GPIO_PORT HTCFG_UARTM0_RX_GPIO_PORT_A
50 #define HTCFG_UARTM0_RX_GPIO_PIN HTCFG_UARTM0_RX_GPIO_PIN_5
51 #define HTCFG_UARTM0_TX_BUFFER_SIZE HTCFG_UARTM0_TX_BUFFER_SIZE_128
52 #define HTCFG_UARTM0_RX_BUFFER_SIZE HTCFG_UARTM0_RX_BUFFER_SIZE_128
53 #endif

```

圖 11. uart_module.h 默認設定

API 說明

- Application Code 範例程式的資料類型說明。

- USART_InitTypeDef

UART 基本配置結構體，包含 BaudRate、WordLength、StopBits、Parity 以及 Mode 相關配置，組成如下。

變量名稱	類型	說明
USART_BaudRate	u32	UART 通訊速率值
USART_WordLength	u16	UART 通訊數據字長，按照設定可以是 7, 8 or 9 位
USART_StopBits	u16	UART 通訊停止位元數，按照設定可以是 1 or 2 位
USART_Parity	u16	UART 通訊奇偶校驗，按照設定可以是 even、odd、mark、space 或是無奇偶校驗
USART_Mode	u16	UART 通訊模式，API 僅支援 normal mode

- 使用 API 函數之前，主程式中都需要先進行 UART 基本配置，本範例使用的 UART 基本配置如圖 12 所示，串列傳輸速率設為 115200bps，字長 8-bit，1 個停止位元，無奇偶校驗。

```

USART_InitTypeDef USART_InitStructure;
USART_InitStructure.USART_BaudRate = 115200;
USART_InitStructure.USART_WordLength = USART_WORDLENGTH_8B;
USART_InitStructure.USART_StopBits = USART_STOPBITS_1;
USART_InitStructure.USART_Parity = USART_PARITY_NO;
USART_InitStructure.USART_Mode = USART_MODE_NORMAL;

```

圖 12. UART 基本配置

- 圖 13 為 uart_module.h 文件中擷取之 API 函數宣告，接著底下則透過表格來依序說明各函數的功能、輸入參數與用法。

```

80 /* Exported functions -----
81 void UARTM_Init(u32 CH, USART_InitTypeDef *pUART_Init, u32 uRxTimeOutValue);
82
83 u32 UARTM_WriteByte(u32 CH, u8 uData);
84 u32 UARTM_Write(u32 CH, u8 *pBuffer, u32 uLength);
85
86 u32 UARTM_ReadByte(u32 CH, u8 *pData);
87 u32 UARTM_Read(u32 CH, u8 *pBuffer, u32 uLength);
88
89 u32 UARTM_GetReadBufferLength(u32 CH);
90 u32 UARTM_GetWriteBufferLength(u32 CH);
91
92 u8 UARTM_IsTxFinished(u32 CH);
93
94 void UARTM_DiscardReadBuffer(u32 CH);

```

圖 13. uart_module.h API 函數宣告

函數	void UARTM_Init(u32 CH, USART_InitTypeDef *pUART_Init, u32 uRxTimeOutValue)	
功能	UART Module 初始化函數	
輸入	CH	UART通道
	pUART_Init	UART基本配置結構指標
	uRxTimeOutValue	UART RX FIFO溢出值。當RX FIFO接收到新資料，計數器將被重置，然後開始計數。一旦計數到設定之溢出值，若溢出中斷有致能，則會產生逾時中斷
用法	UARTM_Init(UARTM_CH0, &USART_InitStructure, 40); //進行 UART 基本配置 //USART_InitStructure 配置可參考圖 12	

函數	u32 UARTM_WriteByte(u32 CH, u8 uData)	
功能	UART module 寫位元組操作(TX)	
輸入	CH	UART通道
	uData	寫操作資料
輸出	SUCCESS	成功
	ERROR	錯誤
用法	UARTM_WriteByte(UARTM_CH0, 'A'); //UART write byte 'A'	

函數	u32 UARTM_Write(u32 CH, u8 *pBuffer, u32 uLength)	
功能	UART module 寫操作(TX)	
輸入	CH	UART通道
	pBuffer	緩衝區指標
	uLength	寫操作長度
輸出	SUCCESS	成功
	ERROR	錯誤
用法	u8 Test[] = "This is test!\r\n"; UARTM_Write(UARTM_CH0, Test, sizeof(Test) - 1); // UART write pBuffer data	

函數	u32 UARTM_ReadByte(u32 CH, u8 *pData)	
功能	UART module 讀位元組操作(RX)	
輸入	CH	UART通道
	pData	讀操作資料位址
輸出	SUCCESS	成功
	ERROR	錯誤(無資料)
用法	u8 TempData; if (UARTM_ReadByte(UARTM_CH0, &TempData) == SUCCESS) { UARTM_WriteByte(UARTM_CH0, TempData); } //UARTM_ReadByte() success then UART write first 1 byte data	

函數	u32 UARTM_Read(u32 CH, u8 *pBuffer, u32 uLength)	
功能	UART module 讀操作(RX)	
輸入	CH	UART通道
	pBuffer	緩衝區指標
	uLength	讀操作長度
輸出	Read count	讀操作數據長度
用法	<pre> u8 Test2[10]; u32 Len; Len = UARTM_Read(UARTM_CH0, Test2, 5); if (Len > 0) { UARTM_Write(UARTM_CH0, Test2, Len); } //UARTM_Read() reads 5 bytes of data and stores data into Test2, and assigns the read byte count to Len //Write the data sourced from Test2 </pre>	

函數	u32 UARTM_GetReadBufferLength(u32 CH)	
功能	取得讀緩衝區長度(RX)	
輸入	CH	UART通道
輸出	uLength	讀緩衝區長度
用法	<pre> UARTM_Init(UARTM_CH0, &USART_InitStructure, 40); //UART module Init while (UARTM_GetReadBufferLength(UARTM_CH0) < 5); // Wait until UARTM_ReadBuffer receive 5 bytes Data </pre>	

函數	u32 UARTM_GetWriteBufferLength(u32 CH)	
功能	取得寫緩衝區長度(TX)	
輸入	CH	UART通道
輸出	uLength	寫緩衝區長度

函數	u8 UARTM_IsTxFinished (u32 CH)	
功能	取得 TX 狀態	
輸入	CH	UART通道
輸出	TRUE	TX狀態：已完成
	FALSE	TX 狀態：未完成
用法	<pre> UARTM_WriteByte(UARTM_CH0, 'O'); #if 1 // "uart_module.c" SVN >= 525 required while (UARTM_IsTxFinished(UARTM_CH0) == FALSE) #else while (1) #endif //透過此 API 能夠進行 TX 狀態判斷，如上面用法所示，可等到 UARTM_WriteByte() API 完成(TX 狀態會改為 TRUE)才接續進行後續動作。 //由於此功能在 uart_module.c 文件中 SVN 版本號為 525 時才加入，因此加入限制。 </pre>	

函數	void UARTM_DiscardReadBuffer(u32 CH)	
功能	丟棄讀取緩衝區資料	
輸入	CH	UART通道

API 使用實例

接著本章節透過流程圖方式，介紹“Module_UART” Application Code，其中流程圖分為初始化流程與“UART_Module_Example” Application Code 流程，來演示寫操作以及讀操作 API 使用實例。使用 API 前請注意需將 API 的標頭文件透過#include 方式包含進主程式原代碼文件的開始來進行使用(include "middleware/uart_module.h")。

初始化流程如圖 14 所示，進入初始化流程後，首先進行 UART 基本配置結構體的定義，接著對 UART 基本配置結構體內的成員進行設定，有 BaudRate、WordLength、StopBits、Parity 與 Mode，最後呼叫 API 初始化函數進行設定，至此初始化流程結束，接著就可以透過設定的 UART 之基本配置進行寫操作以及讀操作。

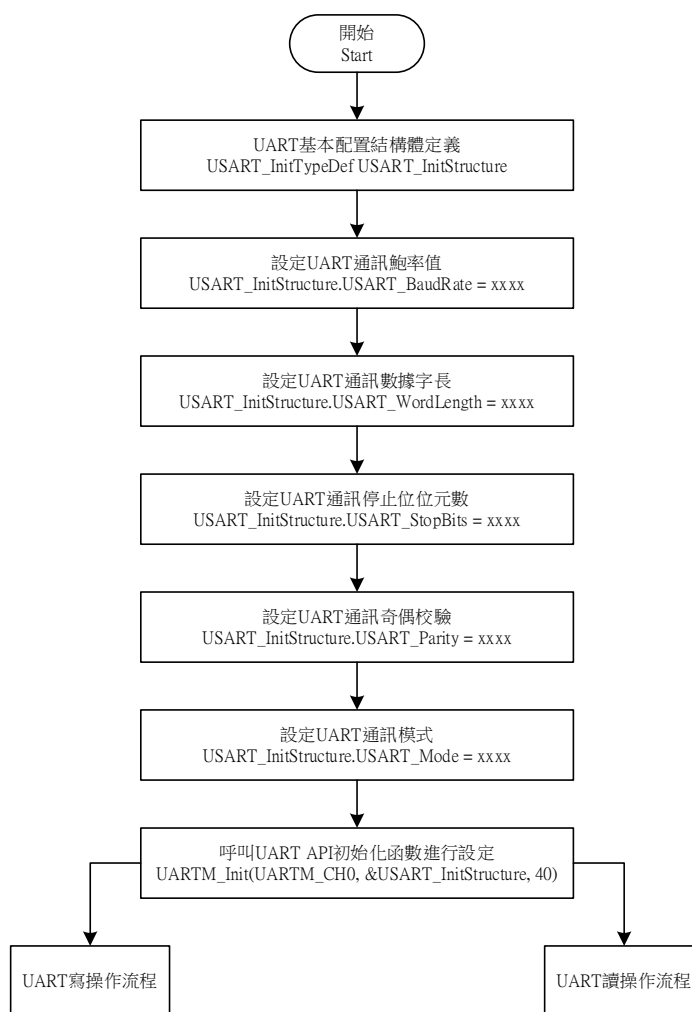


圖 14. 初始化流程

“UART_Module_Example” Application Code 是以 Loopback 的方式來進行 API 讀與寫操作範例，而在此程式中的寫操作與讀操作流程如圖 15 所示。其中使用的 API 函數有以下幾種，“UARTM_WriteByte() 、 UARTM_Write() 、 UARTM_ReadByte() 、 UARTM_Read() 與 UARTM_GetReadBufferLength()”，相關的 API 函數說明可從“API 說明”章節看到。

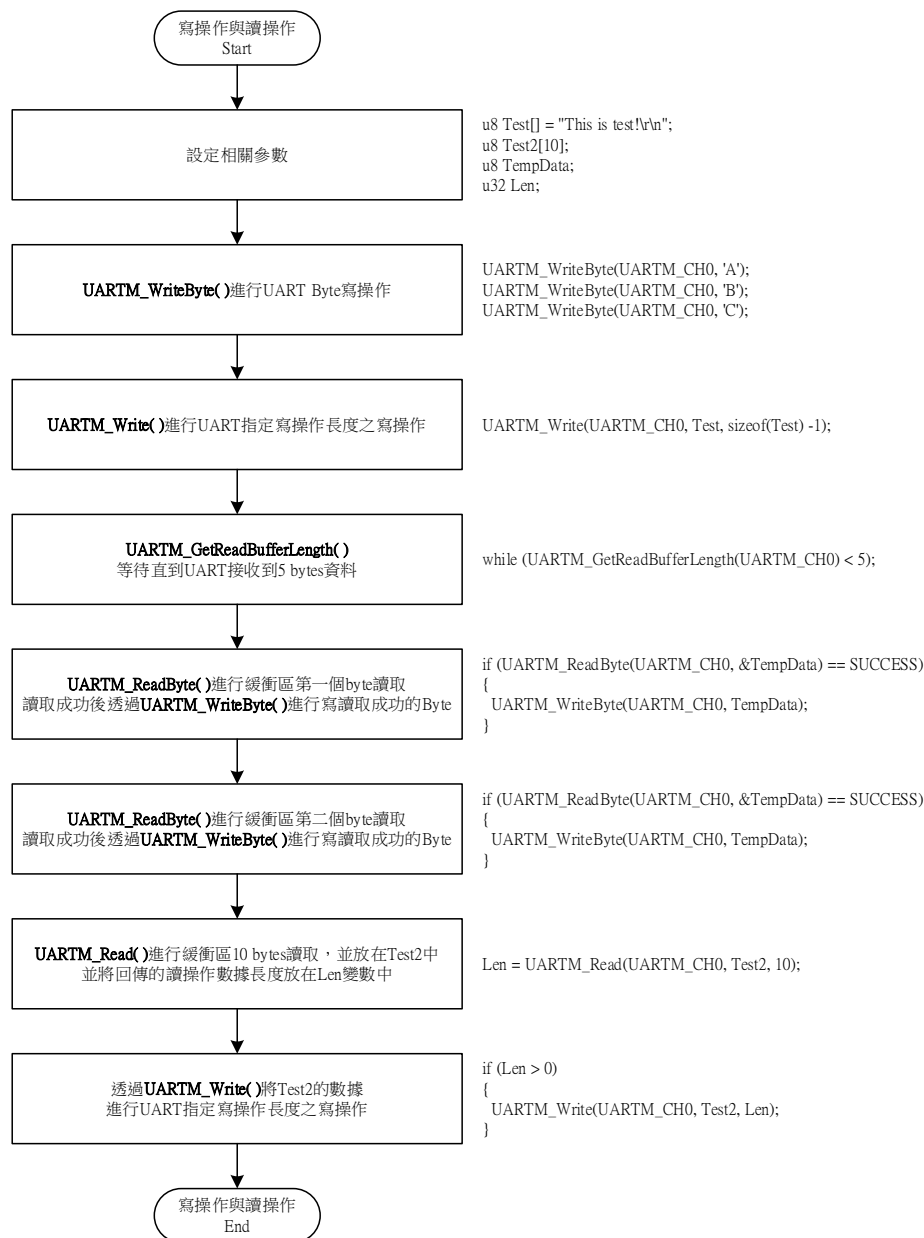


圖 15. 寫操作與讀操作範例流程

應用範例程式下載下來後，資料夾 Module_UART 下還有另一個“UART_Bridge” Application Code，在“目錄結構”章節有說明此 Application Code 的相關檔案及檔案說明。

“UART_Bridge” Application Code 展示透過設定開啟兩個 UART 通道(UART CH0~UART CH1)，然後將兩個 UART 通訊裝置透過 COMMAND 結構來進行自定義的通訊協定，其中 COMMAND 結構透過 uart_bridge.c 定義，如下所示(gCMD1 & gCMD2)。

UARTBridge_CMD1TypeDef gCMD1：

變量名稱	類型	說明
uHeader	u8	Header
uCmd	u8	Command
uData[3]	u8	Data

UARTBridge_CMD2TypeDef gCMD2 :

變量名稱	類型	說明
uHeader	u8	Header
uCmdA	u8	Command A
uCmdB	u8	Command B
uData[3]	u8	Data

“UART_Bridge” Application Code 主要為了實現透過 gCMD1 來進行接收資料作為 Command Packet，接收後會對資料進行解析，並透過自定義的通訊協定將 gCMD2 做設定，然後將 gCMD2 作為 Response Packet 傳送出去的行為。底下則是此 Application Code 中 gCMD1 與 gCMD2 之 Command Packet 與 Response Packet 舉例。

Command Packet (UARTBridge_CMD1TypeDef gCMD1) :

Byte 0	Byte 1	Byte 2 ~ Byte 4
uHeader	uCmd	uData [3]
“A”	“1”	“x, y, z”

Response Packet (UARTBridge_CMD2TypeDef gCMD2) :

Byte 0	Byte 1	Byte 2	Byte 3 ~ Byte 5
uHeader	uCmdA	uCmdB	uData [3]
“B”	“a”	“1”	“x, y, z”

資源佔用

Module UART 的資源佔用如表 4，以 HT32F52352 作為舉例。

	HT32F52352
ROM Size	946 Bytes
RAM Size	40 ¹ +256 ² Bytes

Note：1. 一個 CH 情況下所需要的全域變數為 40 Bytes，其中包含旗標與狀態。

2. 一個 CH 情況下，且設定的 TX/RX Buffer Size 為 128/128 Bytes (可依照需求設定)。

表 4. 範例程式資源佔用

- 編譯環境 MDK-Arm V5.36，ARMCC V5.06 update 7 (build 960)
- 優化選項 Level 2 (-O2)

使用指南

本章節將介紹“Module UART” Application Code 所需環境準備、編譯與測試步驟。

環境準備

“Module UART” Application Code 所需環境準備之硬體/軟體部分所需如下表。

準備硬體/軟體	數量/單位	備註
Starter Kit	1 個	本文以 HT32F52352 Starter Kit 為例
USB 數據線	1 條	Micro USB，連接 PC
Application Code	—	下載路徑及檔案目錄配置可參考“資源下載與準備”章節，準備好之路徑如下：“\application\Module_UARTUART_Module_Example”
Tera Term	—	終端機軟體可參考“終端機軟體”章節
Keil IDE	—	Keil uVision V5.xx

表 5. 環境準備硬體/軟體

首先以 HT32F52352 Starter Kit 與 e-Link32 Lite 的 Virtual COM Port (VCP)功能搭配進行 UART 應用範例說明，請完成以下硬體之環境準備操作：

- (1) 板上有兩個 USB 介面，使用 USB 數據線將 PC 與板上 e-Link32 Lite 介面連接，e-Link32 Lite 介面如圖 16 (a)所示。
- (2) Application Code 需用到 e-Link32 Lite 的 Virtual COM Port (VCP) 功能，因此確保 UART Jumper-J2^{*1}的 PAX^{*2}與 DAP Tx 透過短路帽短路，J2 位置如圖 16 (b)所示。

Note : 1.J2 位於 SK 上有兩個跳線設定，分別為 PAx 與 DAP_Tx 短路或 PAx 與 RS232_Tx 短路，其設定意義請參考 SK 的使用手冊。

2. SK 不同，則設定的 MCU UART RX 各有不同，這裡以 PA_x 來做統稱。

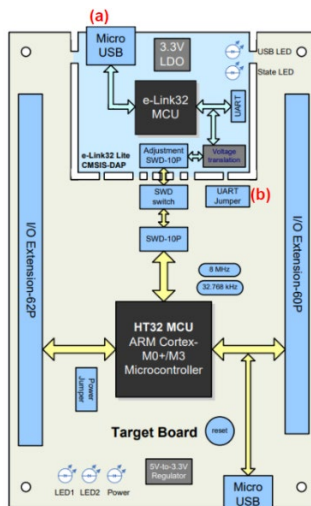


圖 16. HT32 Starter Kit 方塊圖

接著以 User Target Board 與 e-Link32 Pro 的 Virtual COM Port (VCP) 功能搭配進行 UART 應用範例說明，請完成以下硬體之環境準備操作。

- (1) 將 e-Link32 Pro 一邊透過 Mini USB 數據線與 PC 相連，另一邊則由於 e-Link32 Pro 支援 SWD 介面，因此使用 e-Link32 Pro 上的 10-pin 灰排線中 SWD 介面腳位透過杜邦線接至用戶設計之 Target Board 的 SWD 介面對應腳位，如圖 17 (a)所示。
- (2) 接著，e-Link32 Pro 串列通訊功能腳位為 Pin#7-VCOM_RXD 與 Pin#8-VCOM_TXD，因此需將 e-Link32 Pro 串列通訊功能腳位與 Target Board 做連接(TX 與 RX 請記得交叉連接)，如圖 17(b)所示。

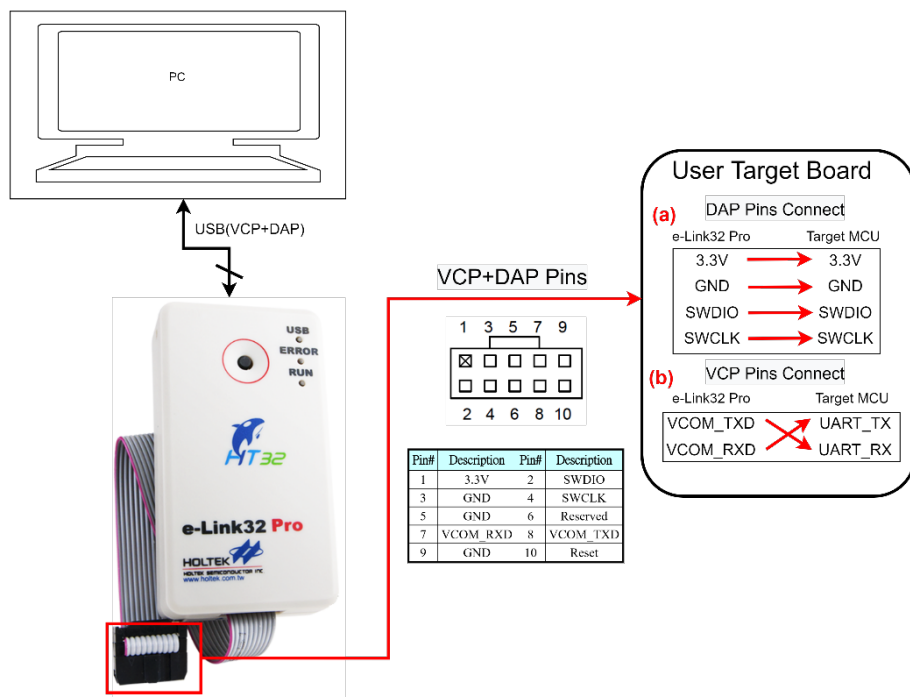


圖 17. e-Link32 Pro + User Target Board 方塊圖

編譯與測試

本文以“application\Module_UART\UART_Module_Example”，進行編譯與測試說明，首先請確認“環境準備”章節部分內容是否已架設完畢，並且終端機軟體 Tera Term 已下載完成。

接著說明上電測試、產生專案以及進行編譯與測試步驟流程，步驟如下。

Step 1. 上電測試

請根據“環境準備”章節將硬體環境設定完成。上電後 SK 的 D9 Power LED 會被點亮 (板子左下方)，等 USB 完成列舉後 e-Link32 Lite 的 D1 USB LED 被點亮 (板子右上)。若長時間 D1 未被點亮，請確認 USB 線是否可以通訊或重新插拔 USB。

Step 2. 產生專案

打開 application\Module_UART\UART_Module_Example，執行_CreateProject.bat 來產生專案，如圖 18 所示，執行結束後請打開 MDK_ARMv5 資料夾，由於本文使用的 SK 為 HT32F52352 Starter Kit，因此開啟 Keil IDE 專案“Project_52352.uvprojx”。

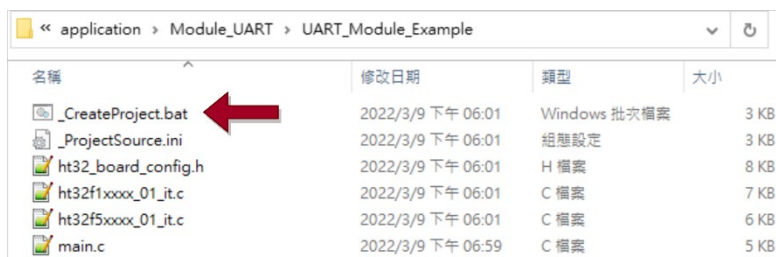


圖 18. 執行_CreateProject.bat 產生專案

Step 3. 編譯與燒錄

開啟專案後接著執行編譯與燒錄，首先“Build”(快捷鍵“F7”)，接著“Download”(快捷鍵“F8”)，完成後可以從 Build Output 看到“Build”與“Download”的結果，如圖 19 所示。

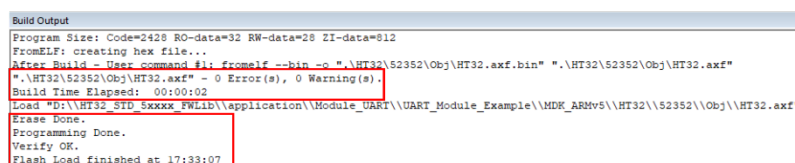


圖 19. Build 與 Download 結果圖

Step 4. 開啟終端機軟體與設定介面配置

打開終端機軟體 Tera Term 並開啟 COM Port，這邊需要特別留意 COM Port 編號是否正確，其中 COM Port 編號由“SK”產生。接著在 Setup >> Serial Port 部分設定介面配置，“Module_UART” Application Code 使用之介面配置於“終端機軟體”章節有說明，介面配置結果如圖 20 所示。

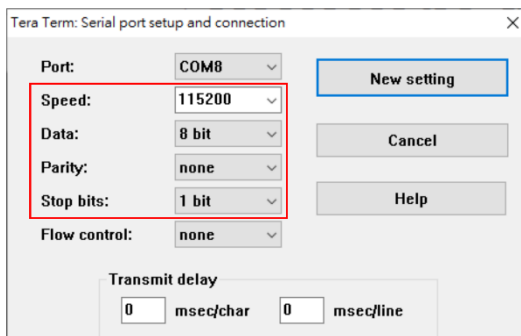


圖 20. Tera Term 介面配置結果

Step 5. 重置系統與測試

按壓 SK 重置按鈕(B1 Reset)，此時從 Tera Term 視窗可以看到透過 API 進行傳送的資料“ABCThis is Test!”，如圖 21 所示，透過 API 完成簡單的傳送功能測試。接著進行接收功能測試，當從 Tera Term 視窗輸入資料給 PC 接收，則會透過 API 進行接收緩衝區長度判斷，當 PC 接收到的資料達到 5 bytes，則會將接收到的 5 bytes 資料依序傳送出來，如圖 22 所示，依序輸入的值為“1,2,3,4,5”，經過 API 接收後判斷，在輸入之五筆資料後方印出“1,2,3,4,5”。

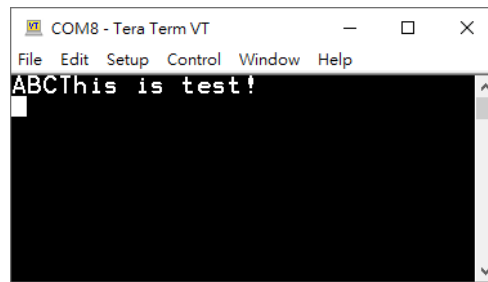


圖 21. “Module_UART” Application Code 功能測試_傳送

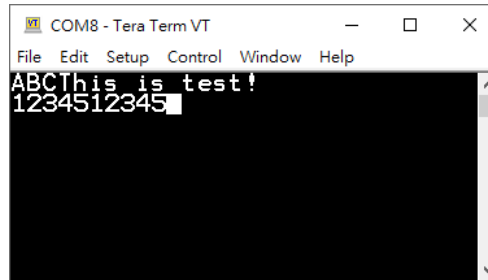


圖 22. “Module_UART” Application Code 功能測試_接收

移植說明

本章節則介紹如何整合 API 到使用者的專案。

- Step 1. 加入 uart_module.c file 到專案內，請對專案 User 資料夾按右鍵，選擇“Add Existing Files to Group ‘User’ ...”，選擇 uart_module.c file 後點擊 Add 加入，如圖 23 所示。
(檔案位置與說明請參考“目錄結構”章節)

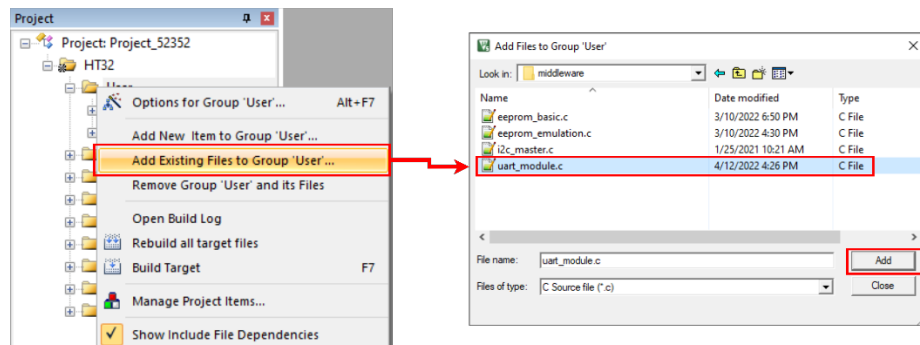


圖 23. 加入 uart_module.c file 到專案內

- Step 2. 加入 ring_buffer.c file 到專案內，請對專案 User 資料夾按右鍵，選擇“Add Existing Files to Group ‘User’ ...”，選擇 ring_buffer.c file 後點擊 Add 加入，如圖 24 所示。
(檔案位置與說明請參考“目錄結構”章節)

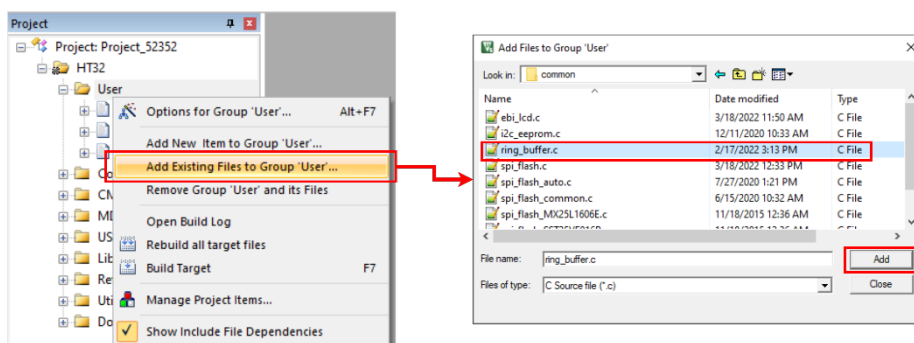


圖 24. 加入 ring_buffer.c file 到專案內

Step 3. 將 API 標頭文件以#include 方式包含進 main.c 文件的開始，如圖 25 所示。

(例如：#include "middleware/uart_module.h")

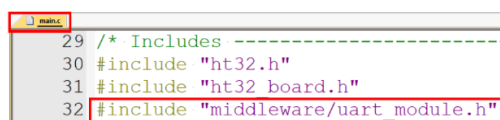


圖 25. Include API 標頭文件進 main.c

Step 4. 最後進行 UART 通訊所需的設定(ht32_board_config.h)，此部分可以參考“設定說明”與“設定變更及常見問題”章節，如此就完成整合 API 於專案中。

設定變更及常見問題

此小節包含說明如何進行 UART 設定變更以及使用上常見問題。

變更 UART 引腳分配

變更 UART 引腳，這裡以 HT32F52352 的 HTCFG_UARTM_CH0 當前設定作舉例說明：

1. 首先打開 HT32F52352 Datasheet 中的“Pin Assignment”章節，找到 Alternate Function Mapping 表格，這部分則可以對應我們使用的型號封裝之腳位對應的 AFIO 功能，由於要設定 UART，可看到表格中的“AF6 USART/UART”部分，如圖 26 紅框中所示。

Table 3. Pin Assignment for 33-pin QFN, 48/64-pin LQFP Packages

Package				Alternate Function Mapping									
64 LQFP	48 LQFP	33 QFN	System Default	AF0 GPIO	AF1 ADC	AF2 CMP	AF3 MCTM /GPTM	AF4 SPI	AF5 USART /UART	AF6 I ² C	AF7 SCI	AF8 EBI	AF9 I2S
1	1	1	PA0	ADC_IN0	GT1_CH0	SPI1_SCK	USR0_RTS	I2C1_SCL	SCI0_CLK	I2S_WS			
2	2	2	PA1	ADC_IN1	GT1_CH1	SPI1_MOSI	USR0_CTS	I2C1_SDA	SCI0_DIO	I2S_BCLK			
3	3	3	PA2	ADC_IN2	GT1_CH2	SPI1_MISO	USR0_TX			I2S_SDO			
4	4	4	PA3	ADC_IN3	GT1_CH3	SPI1_SEL	USR0_RX			I2S_SDI			
5	5	5	PA4	ADC_IN4	GT0_CH0	SPI0_SCK	USR1_TX	I2C0_SCL	SCI1_CLK				
6	6	6	PA5	ADC_IN5	GT0_CH1	SPI0_MOSI	USR1_RX	I2C0_SDA	SCI1_DIO				
7	7		PA6	ADC_IN6	GT0_CH2	SPI0_MISO	USR1_RTS		SCI1_DET				
8	8		PA7	ADC_IN7	GT0_CH3	SPI0_SEL	USR1_CTS			I2S_MCLK			

圖 26. HT32F52352 Alternate Function Mapping 表

2. 接著說明如何透過圖 26 找到對應的 UART 引腳，本範例 HT32F52352 預設使用的是 USART1。因此從 Alternate Function Mapping 表格中找到 USR1_TX 作為使用的 TX 腳，USR1_RX 作為使用的 RX 腳，本範例中使用的為 TX - PA4；RX - PA5。表格對應的方法與“ht32_board_config.h”變更設定如圖 27 所示，其中 Alternate Function Mapping 表格 Package

欄位空著的表示該封裝沒有此 GPIO。因此若需要變更 UART 使用的引腳，可以從表格中對應找出想要變更的腳位，並透過“ht32_board_config.h”進行設定。

Table 3. Pin Assignment for 33-pin QFN, 48/64-pin LQFP Packages

Package				Alternate Function Mapping														
64 LQFP	48 LQFP	33 QFN	System Default	GPIO	ADC	CMP	MCTM /GTM	AF5 SPI	AF6 USART /UART	AF7 PC	AF8 SCI	AF9 EBI	AF10 I2S	AF11 N/A	AF12 N/A	AF13 SCL	AF14 N/A	AF15 System Other
1	1	1	PA0		ADC_IN0		GT1_CH0	SPI1_SCK	USART0_RTS	I2C1_SCL	SCI0_CLK		I2S_WS					
2	2	2	PA1		ADC_IN1		GT1_CH1	SPI1_MOSI	USART0_CTS	I2C1_SDA	SCI0_DIO		I2S_BCLK					
3	3	3	PA2		ADC_IN2		GT1_CH2	SPI1_MISO	USART0_TX				I2S_SDO					
4	4	4	PA3		ADC_IN3		GT1_CH3	SPI1_SEL	USART0_RX				I2S_SDI					
5	5	5	PA4		ADC_IN4		GT0_CH0	SPI0_SCK	USART1_TX	I2C0_SCL	SCI1_CLK							
6	6	6	PA5		ADC_IN5		GT0_CH1	SPI0_MOSI	USART1_RX	I2C0_SDA	SCI1_DIO							
7	7		PA6		ADC_IN6		GT0_CH2	SPI0_MISO	USART1_RTS									
8	8		PA7		ADC_IN7		GT0_CH3	SPI0_SEL	USART1_CTS				I2S_MCLK					

```
111 #if defined(USE_HT32F52352_SK)
112 #define HTCFG_UARTM_CH0 ..... USART1
113 #define HTCFG_UARTM0_TX_GPIO_PORT ..... A
114 #define HTCFG_UARTM0_TX_GPIO_PIN ..... 4
115 #define HTCFG_UARTM0_RX_GPIO_PORT ..... A
116 #define HTCFG_UARTM0_RX_GPIO_PIN ..... 5
117 #define HTCFG_UARTM0_TX_BUFFER_SIZE ..... 128
118 #define HTCFG_UARTM0_RX_BUFFER_SIZE ..... 128
119
120 // #define HTCFG_UARTM_CH1 ..... USART0
121 // #define HTCFG_UARTM1_TX_GPIO_PORT ..... A
122 // #define HTCFG_UARTM1_TX_GPIO_PIN ..... 2
123 // #define HTCFG_UARTM1_RX_GPIO_PORT ..... A
124 // #define HTCFG_UARTM1_RX_GPIO_PIN ..... 3
125 // #define HTCFG_UARTM1_TX_BUFFER_SIZE ..... 128
126 // #define HTCFG_UARTM1_RX_BUFFER_SIZE ..... 128
127 #endif
```

圖 27. Alternate Function Mapping 表對應方法與變更設定

新增 UART 通道

新增 UART 通道，這裡以 HT32F52352 的 HTCFG_UARTM_CH1 當前設定作舉例說明。

- 修改 ht32_board_config.h

HT32 MCU 的 I/O 復用 UART 功能請參考各型號 Datasheet 中的“Pin Assignment”章節，這部分則可以對應使用的型號封裝之腳位對應的 AFIO 功能，其中由於 HTCFG_UARTM_CH0 使用的是 USART1，因此 HTCFG_UARTM_CH1 可設定新增如圖 28 上方紅框 USART0 的設定(TX - PA2；RX - PA3)，修改 ht32_board_config.h 內容，如圖 28 中底下紅色虛線程式碼第 120 行~126 行所示進行新增。

Table 3. Pin Assignment for 33-pin QFN, 48/64-pin LQFP Packages

Package				Alternate Function Mapping														
64 LQFP	48 LQFP	33 QFN	System Default	GPIO	ADC	CMP	MCTM /GTM	AF5 SPI	AF6 USART /UART	AF7 PC	AF8 SCI	AF9 EBI	AF10 I2S	AF11 N/A	AF12 N/A	AF13 SCL	AF14 N/A	AF15 System Other
1	1	1	PA0	ADC_IN0			GT1_CH0	SPI1_SCK	USART0_RTS	I2C1_SCL	SCI0_CLK		I2S_WS					
2	2	2	PA1	ADC_IN1			GT1_CH1	SPI1_MOSI	USART0_CTS	I2C1_SDA	SCI0_DIO		I2S_BCLK					
3	3	3	PA2	ADC_IN2			GT1_CH2	SPI1_MISO	USART0_TX				I2S_SDO					
4	4	4	PA3	ADC_IN3			GT1_CH3	SPI1_SEL	USART0_RX				I2S_SDI					
5	5	5	PA4	ADC_IN4			GT0_CH0	SPI0_SCK	USART1_TX	I2C0_SCL	SCI1_CLK							
6	6	6	PA5	ADC_IN5			GT0_CH1	SPI0_MOSI	USART1_RX	I2C0_SDA	SCI1_DIO							
7	7		PA6	ADC_IN6			GT0_CH2	SPI0_MISO	USART1_RTS									
8	8		PA7	ADC_IN7			GT0_CH3	SPI0_SEL	USART1_CTS				I2S_MCLK					

```

111 #if defined(USE_HT32F52352_SK)
112 #define HTCFG_UARTM_CH0 ..... USART1
113 #define HTCFG_UARTM0_TX_GPIO_PORT ..... A
114 #define HTCFG_UARTM0_TX_GPIO_PIN ..... 4
115 #define HTCFG_UARTM0_RX_GPIO_PORT ..... A
116 #define HTCFG_UARTM0_RX_GPIO_PIN ..... 5
117 #define HTCFG_UARTM0_TX_BUFFER_SIZE ..... 128
118 #define HTCFG_UARTM0_RX_BUFFER_SIZE ..... 128
119
120 // #define HTCFG_UARTM_CH1 ..... USART0
121 // #define HTCFG_UARTM1_TX_GPIO_PORT ..... A
122 // #define HTCFG_UARTM1_TX_GPIO_PIN ..... 2
123 // #define HTCFG_UARTM1_RX_GPIO_PORT ..... A
124 // #define HTCFG_UARTM1_RX_GPIO_PIN ..... 3
125 // #define HTCFG_UARTM1_TX_BUFFER_SIZE ..... 128
126 // #define HTCFG_UARTM1_RX_BUFFER_SIZE ..... 128
127 #endif

```

圖 28. 新增 UART 通道

常見問題

Q：為什麼在測試範例程式流程中，進行到 Step5 這個步驟，傳送功能測試正常，有出現“ABCThis is test!”，但接收功能部份，輸入五個值卻沒有返回值？

A：請檢查是否有將 UART Jumper-J2 的 MCU UART RX 與 DAP_Tx 透過短路帽進行短路。由於“Module_UART” Application Code 需用到 e-Link32 Lite 的 Virtual COM Port (VCP) 功能，因此 UART Jumper-J2 實際設定狀況如圖 29 所示，Jumper 短路應於左側。

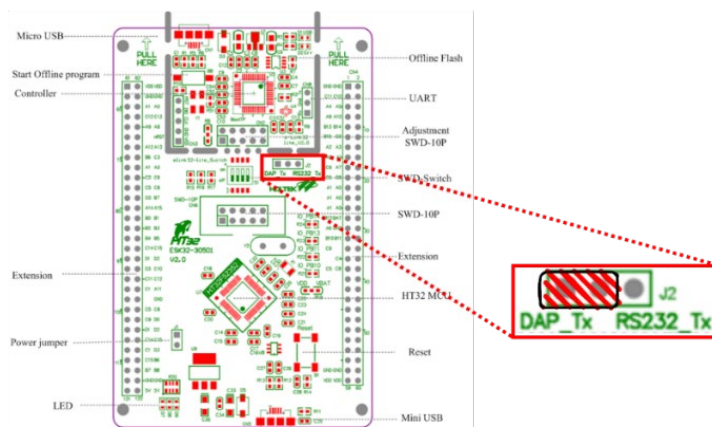


圖 29. UART Jumper-J2 設定

Q：為什麼進行“Build”(快捷鍵“F7”)指令後出現 Firmware Library 版本過舊錯誤訊息，如圖 30 所示？

A：由於“Module_UART” Application Code 需用到 uart_module.c/h 檔案，檔案有 Firmware Library 版本要求，因此遇到這問題意味當前使用的 Firmware Library 版本過舊，請從“Firmware Library”章節中下載最新版本的 Firmware Library。

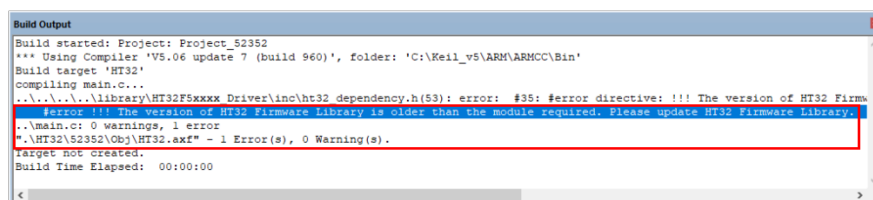


圖 30. Firmware Library 版本過舊錯誤訊息

結論

本文首先介紹了“Module_UART” Application Code 的簡介以及 UART 通訊協定，使得用戶能首先了解 Application Code 與 UART 的通訊協定相關基礎，接著提供資源下載與準備方法，功能說明部分則說明目錄結構、API 架構、說明與使用實例等資訊，並在使用指南章節按步驟演示了“Module_UART” Application Code 之環境準備、編譯與測試流程，最後介紹了移植說明與設定變更及常見問題，使用戶能快速了解 API 使用方法，縮短上手的時間。

參考資料

如需進一步瞭解，敬請瀏覽 Holtek 官方網站 www.holtek.com。

版本及修訂說明

日期	作者	發行	修訂說明
2022.04.30	蔡期育	V1.00	第一版

免責聲明

本網頁所載的所有資料、商標、圖片、連結及其他資料等（以下簡稱「資料」），只供參考之用，盛群半導體股份有限公司及其關聯企業（以下簡稱「本公司」）將會隨時更改資料，並由本公司決定而不作另行通知。雖然本公司已盡力確保本網頁的資料準確性，但本公司並不保證該等資料均為準確無誤。本公司不會對任何錯誤或遺漏承擔責任。

本公司不會對任何人士使用本網頁而引致任何損害（包括但不限於電腦病毒、系統故障、資料損失）承擔任何賠償。本網頁可能會連結至其他機構所提供的網頁，但這些網頁並不是由本公司所控制。本公司不對這些網頁所顯示的內容作出任何保證或承擔任何責任。

責任限制

在任何情況下，本公司並不須就任何人由於直接或間接進入或使用本網站，並就此內容上或任何產品、資訊或服務，而招致的任何損失或損害負任何責任。

管轄法律

本免責聲明受中華民國法律約束，並接受中華民國法院的管轄。

免責聲明更新

本公司保留隨時更新本免責聲明的權利，任何更改於本網站發佈時，立即生效。