

VATEK SDK 開發手冊

Release Date: March 2022

目錄

1	前言	1
2	晶片介紹	1
2.1	A Series	1
2.1.1	A Series 系統架構	1
2.1.2	系統功能操作	4
2.1.3	A Series 晶片系統流程圖	6
2.1.4	開機流程	6
2.2	B Series	8
2.2.1	B Series 系統架構	8
2.2.2	系統功能操作	11
2.2.3	B Series 晶片系統流程	12
2.2.4	開機流程	14
3	軟體開發包(SDK)	16
3.1	簡介	16
3.2	核心程式開發介面	17
3.2.1	Common 類別	18
3.2.2	Broadcast 類別	23
3.2.3	Transform 類別	32
4	SDK 編譯	39
4.1	編譯的需求 :	39
4.2	編譯選項	39
4.3	編譯 vatek_sdk_2 :	40
4.4	編譯 TSDuck_plugins (A3 裝置支援) :	45

4.4.1	Windows 作業系統.....	45
5	軟體開發包主應用	48
5.1	app_stream 範例程式 (A 系列使用)	48
5.2	app_broadcast 範例程式 (B 系列使用)	50
5.3	Vatek 裝置 Log 查看	51
6	系統除錯功能.....	54

Vision Advance Technology

1 前言

瞻誠科技全系列數位電視調變晶片，主要分別兩個產品線，B 系列涵蓋多媒體影音編碼與數位電視調變的單一晶片方案，A 系列為單純數位電視調變晶片。開發相關產品與應用時針對不同系列需要與不同的周邊進行整合，基於開發需求瞻誠科技提供軟體開發包 (SDK) 與嵌入式軟體開發包 (MDK) 協助開發者。

2 晶片介紹

VATek 開發的兩種系列晶片與數位電視相關，以下介紹 A 系列及 B 系列晶片：

2.1 A Series

2.1.1 A Series 系統架構

A 系列晶片為數位電視調變晶片，可藉由外部硬體介面輸入 TS 資料流，將接收到的訊號進行再製，提供加入 PSI table 及 PCR 校正的功能，並依照需求轉換成不同調變後輸出 TS 訊號進行廣播，系統架構主要分為三個部分。

➤ 系統 IO：

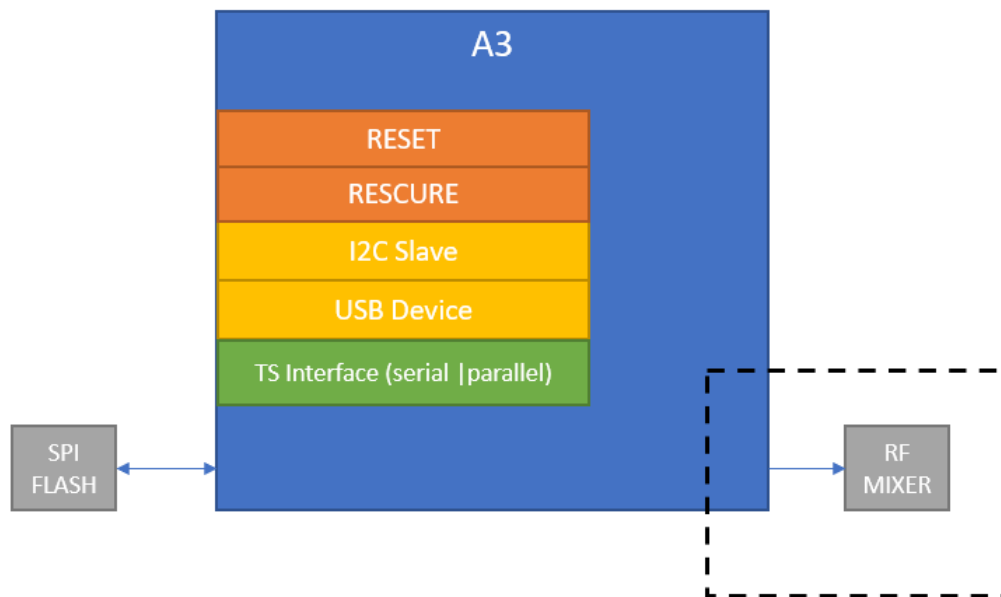
- RESET 訊號 - 系統 RESET 控制訊號，可由外部進行硬體復位 (參閱 2.1.4 開機流程)。
- RESCUE 訊號 - 救援功能，當韌體損壞時進入救援模式訊號 (參閱 2.1.4 開機流程)。

➤ 控制介面：

- 支援 MDK 結合外部 MCU 透過 I2C。
- 使用 SDK 提供的 USB 介面進行晶片功能設定與控制。

➤ 輸入介面：

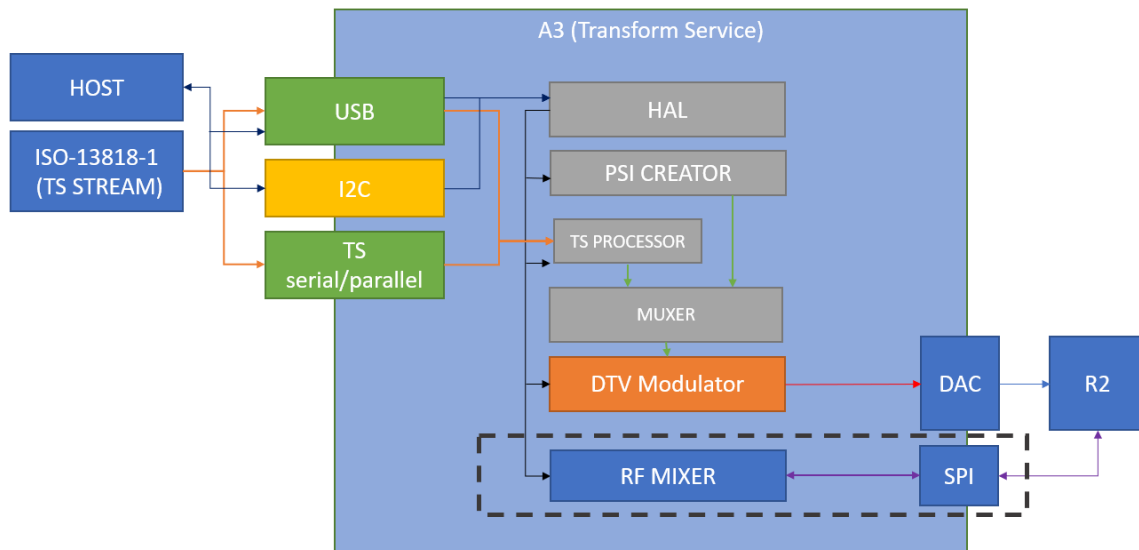
- TS 介面 - 支援 Serial 與 Parallel 模式
- USB BULK 介面 - USB 除了控制與設定外支援透過 Bulk Endpoint 輸入多媒體流



外部硬體另外包含兩個部分，第一為韌體載入功能，A 系列需依賴韌體(服務)，才可以完整提供相關功能，上電時需由 **SPI** 介面載入韌體(服務)，細節可參閱開機流程章節。第二部分為選用功能，如果搭配完整開發方案設計，使用搭配的 **RF MIXER** 則需透過同一介面連結對應 **RF MIXER** 以方便韌體(服務)提供 **RF** 相關能力。

除外部介面外，晶片內由許多功能單元結合而成，主要可以區分為：

- **HAL**：外部控制單元支援 I2C 與 USB 的外部協定實作，用來控制與協調內部功能。
- **PSI CREATOR**：數位電視需要之 PSI TABLES 產生單元
- **TS PROCESSOR**：具備解析與處理輸入多媒體流功能。
- **MUXER**：將 TS PROCESSOR 與 PSI CREATOR 結合 PCR 產生符合調製單元需要的 TS stream。
- **DTV Modulator**：數位電視調製器，多格式數位電視調製單元。
- **RF Mixer**：控制統一單元，控制預設支援的 RF MIXER (可選)。



A Series 系統架構圖

➤ I2C 協定

透過 CHIP 支援的 I2C Slave 介面，支援最高 400 Kb 傳輸速度，可以對 HAL REGISTER 進行寫入與讀取操作。

➤ USB 協定

透過 CHIP 支援的 USB Device 介面，透過標準的 EP0(end point) Setup Packet 對 HAL REGISTER 進行寫入與讀取操作。

➤ TS PROCESSOR 單元

由於不同數位電視調變規格與參數的設置會需要不同碼率與需求的 TS stream，此單元主要用來將輸入與 PSI CREATOR 結合，針對輸出需求從新產生新的 TS stream，核心具備下列功能。

- **TS FILTER**：過濾功能可以用來重新擷取 TS stream 的特別多媒體內容，最高可過濾 16 組 PID。
- **TS DEMUX**：解析功能，可以解析來源 TS stream 的內容架構與簡易訊息，便於產品設置時設置過濾或開發 PSI TABLE 使用。
- **REMUX**：復用功能，依據後端調製需求（碼率）將有效輸入結合 PSI TABLE

重新復用為實際輸出的 TS stream。

- **TS CAPTURE**：擷取功能，可用來解取小部分的流片段，已取得產生 PSI TABLE 所需要的內容詳細資訊。

➤ **PSI CREATOR 單元**

數位電視含聲音影像資料流外依據不同國家與標準，需加入 PSI TABLE，用來識別與定義頻道與多媒體內容，此單元提供兩種不同方式協助開發者完成所需要的 PSI TABLE。

- **PURE TABLE**：由開發者依據應用情境參考所需規格書，自行加入自訂義的 PSI TABLE。
- **DEFAULT**：針對不同國家與規格所定義的基礎 PSI TABLE，參數設置完畢後即可使用。

➤ **MUXER 單元**

基於不同調變模式與來源狀況，如碼率、多媒體流時間...都需經過 MUXER 加以重新排序，以符合數位電視廣播相關規範，MUXER 單元即是依據輸出需求與輸入條件進行 TS stream 的重新編排。

- **PADDING 功能**：於資料無法滿足時會使用 NULL PACKET 進行填充，MUXER 提供自定義與標準功能。
- **PCR INSERT**：PCR 的插入功能可加入獨立 PID 的 PCR，並且可控制 Interval 以符合應用需求，例如：在 remux 時插入 PSI table。
- **PCR REPLACE**：提供 PCR 複寫功能，於不同輸入源碼率轉換時可能使原始 PCR 精度變更，透過複寫功能可不同程度的修正此問題。

➤ **MODULATOR 單元**

廣泛支援全球數位電視調變規格，包括 DVB-T、DVB-C (J83a)、ATSC、j83b、DTMB、ISDB-T、J83c、DVB-T2。

2.1.2 系統功能操作

- Transport Stream (TS) 組成：

TS 為數位電視播放的來源格式，完整 TS 的組成主要包含下列三個元素：

1. **PES**：封裝影音訊號及其他如 CC 字幕等訊號的封包
2. **PCR**：可以參考 ISO-13818 規格，後端輸出至電視時 DEMOD 需要參考的時間，主要決定封包播放順序，其 PCR 精度好壞會影響播放品質
3. **PSI**：於各國電視播放時需遵從的規範，不同國家具備不同 PSI TABLE，如美國為 PSIP、日本為 ARIB...等

- 晶片依據前端輸入介面分為 USB 及 TS 傳輸介面來接收 TS 訊號

輸入介面	描述
USB interface	USB 介面可控制 TS 輸入速度，USB2.0 最高可達 480Mbps。
TS interface	須為連續資料，若中途 TS 中斷將造成晶片發生錯誤。

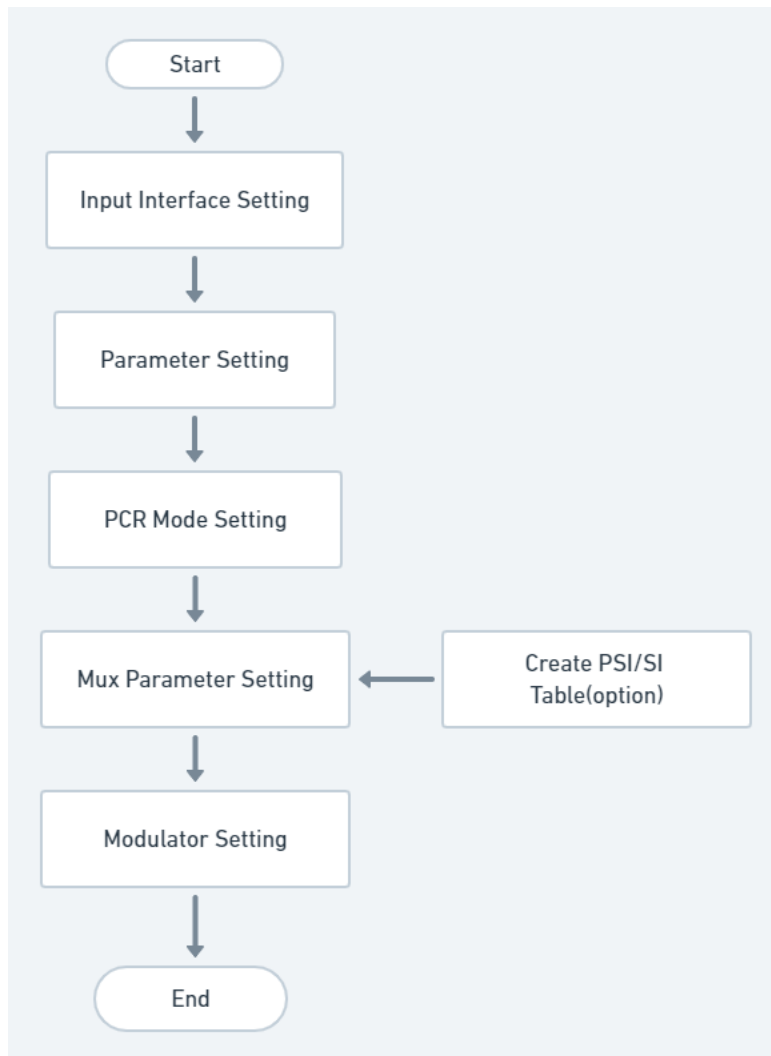
- 晶片依據前端輸入的 TS 完整度不同可以使用 PASSTHROUGH 及 REMUX 模式。

模式	適用情境
PASSTHROUGH	前端輸入 TS 已具備完整的內容，不希望更動到原本的 TS，僅需將訊號轉為數位電視訊號廣播。
REMUX	<ol style="list-style-type: none"> 1. 前端輸入 TS 不具備 PSI TABLE，需協助加入 PSI TABLE 2. 前端輸入 TS 的 PCR 需進行校正。

- A3 進行 REMUX 模式提供三種功能處理，可以針對 PCR 進行校正功能，如下表所敘

REMUX 功能	適用情境
RETAG	欲提高輸出的 PCR 精度，可使用該模式，其原理為依據前端 TS 輸入的 PCR 產生精度更高的 PCR 提供後端 TS 輸出的 PCR 參考
ADJUST	校正 DAC 的 PCR
DISABLE	不對 PCR 進行校正

2.1.3 A Series 晶片系統流程圖

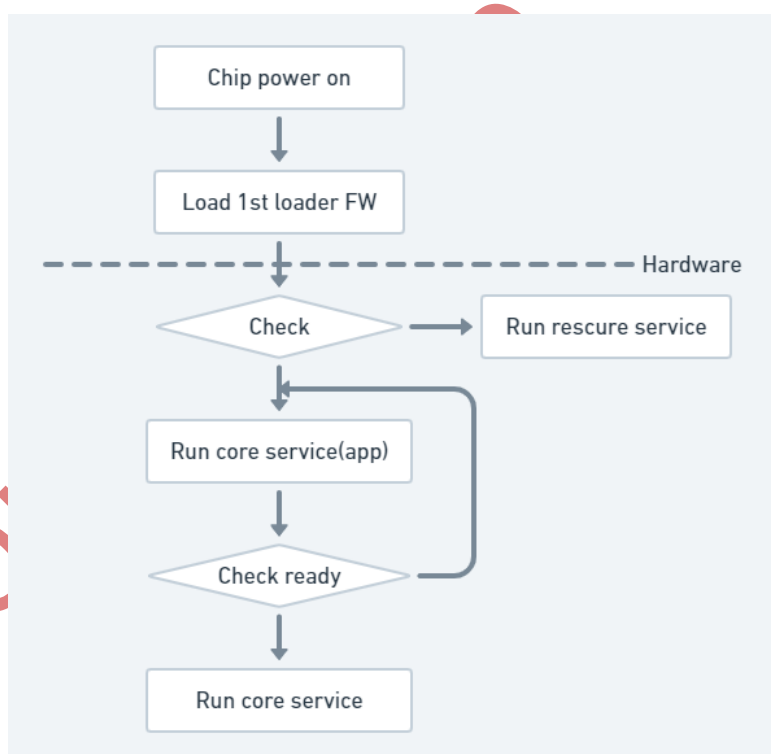
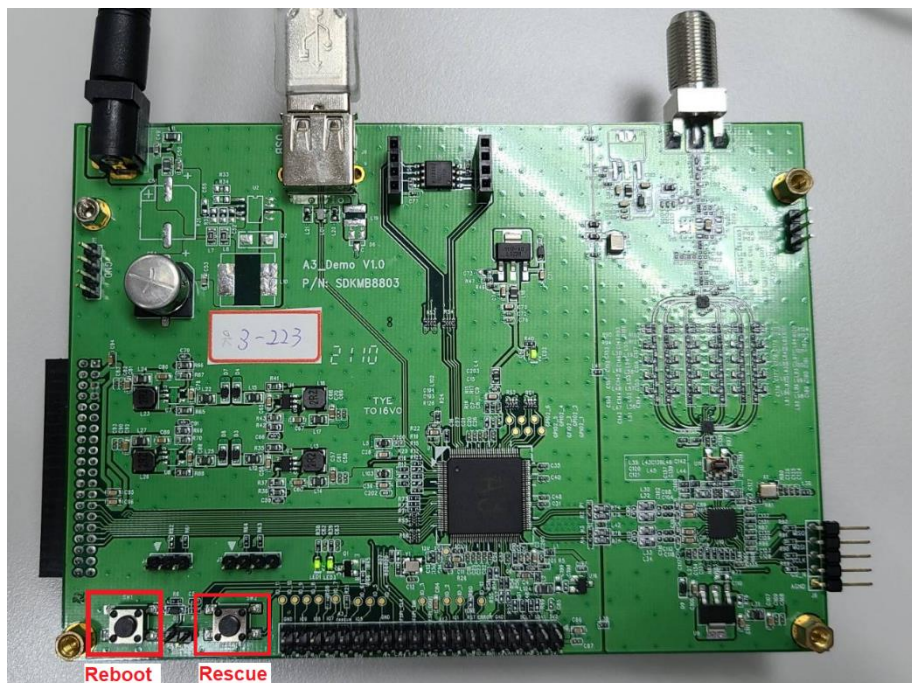


2.1.4 開機流程

系統執行主要由 Loader(開機程式) 與 Service(服務) 組成，當晶片完成上電程序，晶片會自動透過 SPI 介面載入並運行，運行後會檢測 Rescue 訊號狀態，決定是否強制執行 Rescue 服務，Rescue 主要提供更新韌體功能，通常使用在韌體無法正常讀取的救援模式。如果不須進入 Rescue 服務則會檢查韌體，如果為有效韌體則載入並運行 Service (服務)。

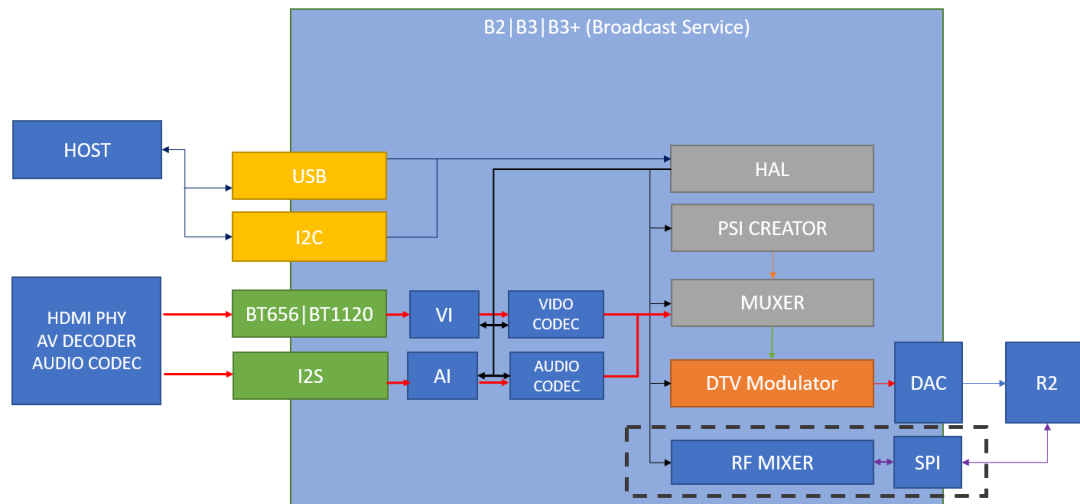
透過外部 I2C 與 USB 透過 HAL 控制單元控制晶片功能，兩個路徑可控制階段將不同，由於 USB 介面由服務提供底層 USB 服務，故進入 Rescue 與 Service (服務) 前無法與晶片溝通，而 I2C 則是在系統上電後即可進行讀取狀態與控制操作。

- 下圖為 A 系列裝置，若發生裝置無反應請重啟。



2.2 B Series

B 系列晶片為整合影音訊號編碼及數位電視調製的晶片，透過 PHY 接收影音訊號進行壓縮編碼處理，並加入 PSI table，最後經過調製單元轉換為各式調變制式電視訊號廣播。B 系列晶片支援 BT656、BT1120 的影像訊號和 I2S 聲音訊號格式，並使用 YUV420 色彩格式進行影像編碼壓縮，最高支援輸入 1080P60。



B Series 系統架構圖

2.2.1 B Series 系統架構

➤ VI/AI 輸入單元

B 系列 CHIP 支援的影像訊號輸入單元，提供最高 1080P 60 的影像來源。除實際影像輸入外，還可以由晶片內部來源作為輸入來源。輸入單元也提供輸入來源的設定：

- 內部來源

- 1 **COLORBAR**：可產生符合 SMPTE RP 219:2002(ARIB STD-B28) 測試影像。
- 2 **BOOTLOGO**：允許加入多個自定義圖形做為影像來源。

- **VI_FLAG (HALREG_VI_0_FLAGS)**：

- 1 **VI_BUSWIDTH_16**：若使用 HDMI 輸入需開啟以符合 HDMI 輸入規格。
- 2 **VI_SEPARATED_SYNC**：若使用 HDMI 輸入需開啟以符合 HDMI 輸入

規格。

3 **VI_EXT_HALF_FPS**：可將輸入訊號的 Frame rate 減半 (1080P60->1080P30, 720P60->720P30...)

- VI 單元可支援輸入的解析度格式遵循 CEA-861 規範，詳細可自行參閱，下表提供 B 系列晶片支援解析度之格式，B 系列 CHIP 支援的聲音訊號輸入單元，支援 32K、44.1K 與 48 K sample rate 最高可支援 2 channel 24 bits PCM。

Resolution \ FPS	FPS						
	60	59.94	50	30	29.97	25	23.97
1080P	V	V	V	V		V	V
1080I	V	V	V				
720P	V	V	V				
576P			V			V	
576I			V				
480P	V	V		V	V		
480I	V	V					

➤ 影音編碼格式

B 系列支援影像及聲音壓縮，依據不同晶片型號支援不同影像壓縮，支援格式如下表：

編碼格式		B2	B2+	B3	B3+
\CHIP 型號					
AUDIO ENCODER	MPEG1-L2	V	V	V	V
	AC-3	V	V	V	V
	AAC-ADTS	V	V	V	V

	AAC-LATM	V	V	V	V
VIDEO	H.264			V	V
ENCODER	MPEG2	V	V		V

- **MPEG 2 編碼器規格：**

1. 最高解析度 1080p 30 FPS
2. HIGH PROFILE (I、P Frame Only)
3. HIGH LEVEL 最高 Bitrate 30 Mbps (VBR)
4. YCrCb 4:2:0

- **H264 AVC 編碼器規格：**

1. 最高解析度 1080p 60 FPS
2. HIGH PROFILE(I、P Slice、No MBAFF)
3. Level 4.1 (30 Mbps)
4. YCrCb 4:2:0

- **Encoder Flag (HALREG_ENCODER_FLAGS)**

- 1 **ENC_EN_PROGRESSIVE_2_I**：輸入 Progressive 影像格式，啟用此功能輸出 1080i 60 (1080P60->1080I60...)。
- 2 **ENC_EN_DISABLE_DEINTERLACED**：在 B3+ 晶片中，若使用 Interlaced 影像格式輸入，啟用此功能將 interlaced 格式轉換為 progressive 格式 (1080I60->1080P60...)。
- 3 **ENC_EN_DISABLE_LATENCY_Q**：在調教畫質時 Latency 與 Q 值會相互影響，啟用此功能關閉兩者之間相互影響的狀況。

➤ **PSI CREATOR 單元**

數位電視含聲音影像資料流外，依據不同國家與標準，需加入 PSI TABLE，用來識別與定義頻道與多媒體內容，此單元提供兩種不同方式協助開發者完成所需要的 PSI TABLE。

- **PURE TABLE**：由開發者依據應用

- **PCR INSERT**：情境參考所需規格書，自行加入自訂義的 PSI TABLE。
- **DEFAULT**：針對不同國家與規格所定義的基礎 PSI TABLE，透過參數的設置即可使用。

➤ **MUXER 單元**

基於不同調變模式與來源狀況，如 **Bitrate**、多媒體流時間...需經過 MUXER 加以重新排序，以符合數位電視廣播相關規範，MUXER 單元即是依據輸出需求與輸入條件進行 TS stream 的重新編排。

- **PADDING 功能**：於資料無法滿足時會使用 NULL PACKET 進行填充，MUXER 提供自訂義與標準功能。
- **PCR INSERT**：PCR 的插入功能可加入獨立 PID 的 PCR，並且可控制 Interval 以符合應用需求。
- **PCR REPLACE**：提供 PCR 複寫功能，於不同輸入源 Bitrate 轉換時可能使原始 PCR 精度變更，透過複寫功能可不同程度的修正此問題。

➤ **MODULATOR 單元**

廣泛支援全球數位電視調變規格，包括 DVB-T、DVB-C (J83a)、ATSC、j83b、DTMB、ISDB-T、J83c、DVB-T2，不同晶片支援的調變類型如下表：

CHIP 型號	支援格式
B2	DVB-T、J83a、ATSC、J83b、DTMB、ISDB-T、J83c
B3	DVB-T、J83a、ATSC、J83b、DTMB、ISDB-T、J83c
B3+	DVB-T、J83a、ATSC、J83b、DTMB、ISDB-T、J83c、DVB-T2

2.2.2 系統功能操作

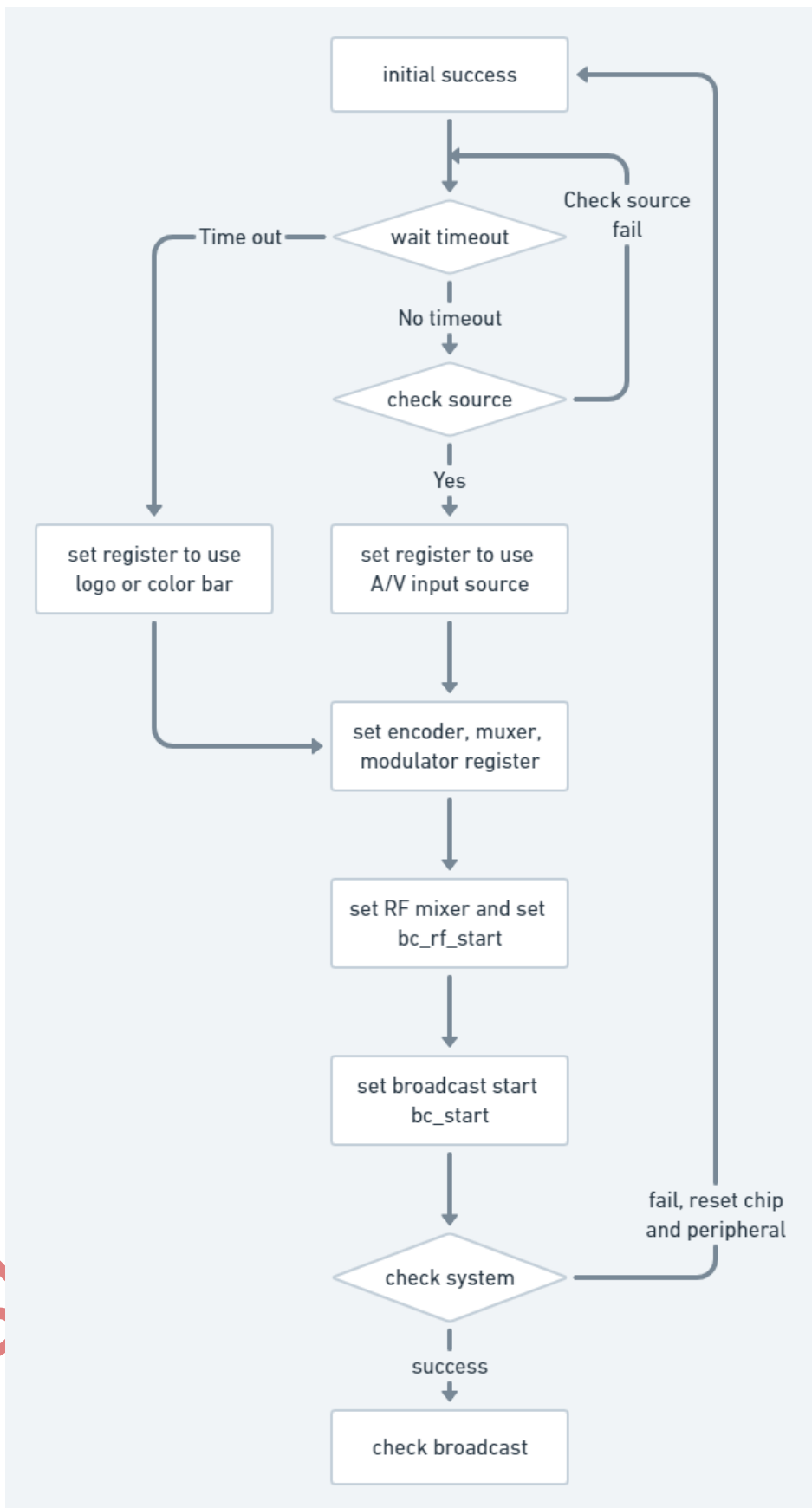
當晶片完成上電後，可透過 I2C 進行暫存器的讀取指令，需確認系統狀態是

否可正確讀取系統狀態暫存器 0x20，系統狀態大致分為以下三種：

1. 閒置狀態：當晶片完成上電且正確載入韌體時，將回傳 **idle** 狀態，表示晶片可以執行閒置階段的相關功能操作
2. 運行狀態：當晶片回傳 **running** 狀態，表示系統運作中，此時晶片只能執行運作階段的相關指令操作
3. 錯誤狀態：系統狀態暫存器必須以 **0xFF000000** 為基底，若沒有則表示沒有正確載入韌體，晶片上電後須確保韌體正確運行。

2.2.3 B Series 晶片系統流程

如要進行廣播功能，需先確認初始化是否完成，可以讀取晶片狀態暫存器 (0x20) 確認晶片是否處於 **IDLE** 狀態，確認晶片狀態正確後才能開始對晶片讀寫指令，使用者可自行決定使用實際輸入的來源 **VI** 或是 **CHIP** 內建的 **color bar** 進行數位電視廣播，其播放流程如下

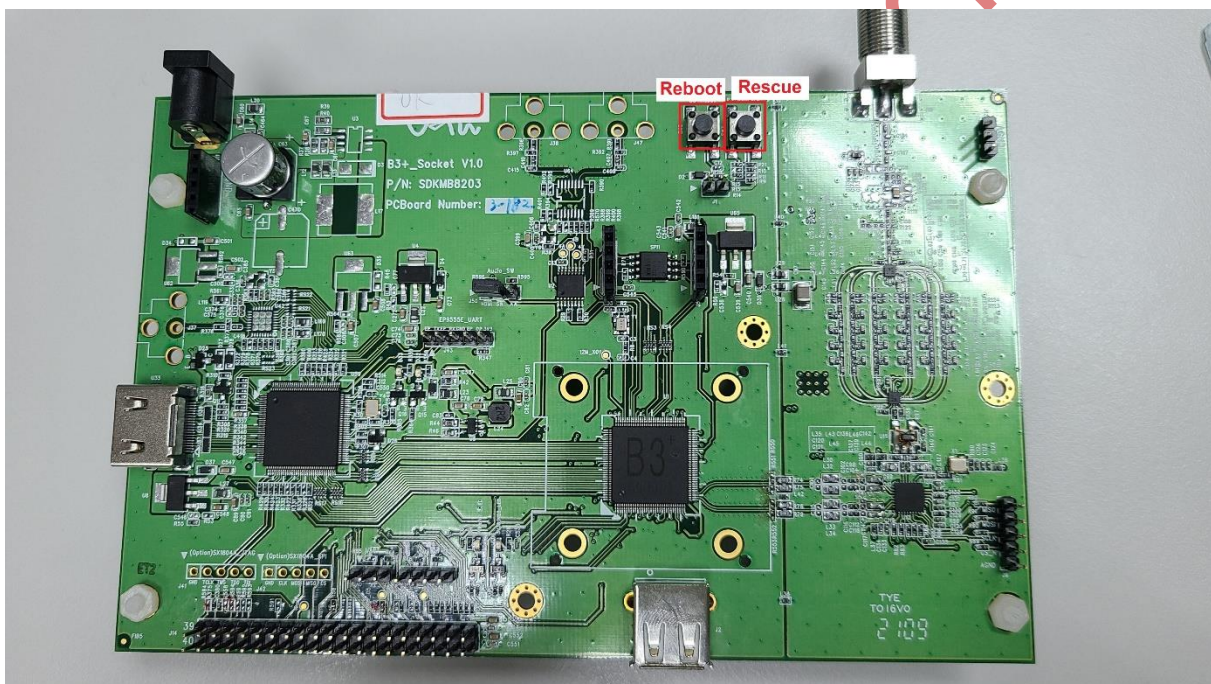


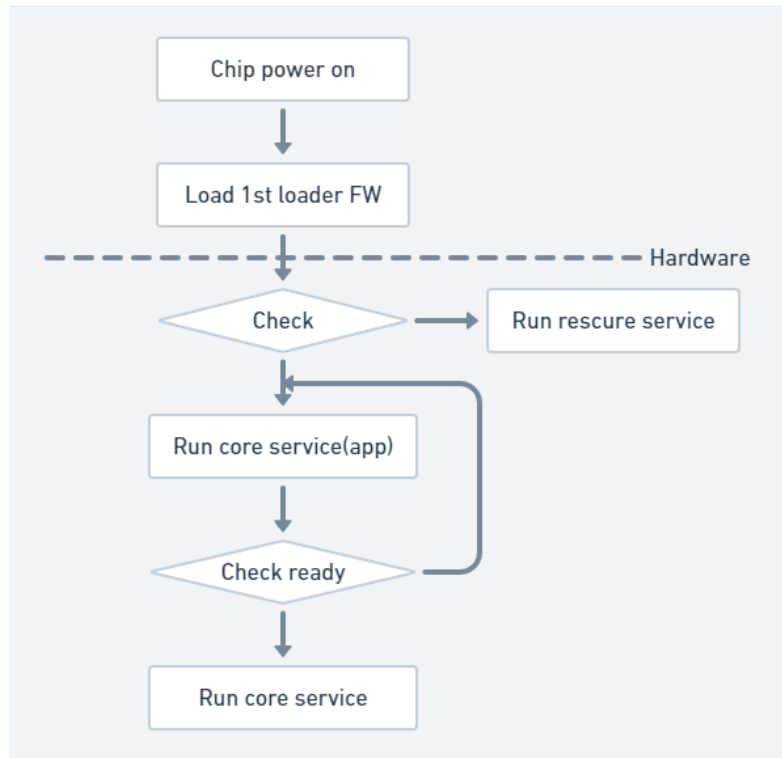
2.2.4 開機流程

系統執行主要由 Loader(開機程式) 與 Service(服務) 組成，當晶片完成上電程序，晶片會自動透過 SPI 介面載入並運行，運行後會檢測 Rescue 訊號狀態，決定是否強制執行 Rescue 服務，Rescue 主要提供更新韌體功能，通常使用在韌體無法正常讀取的救援模式。如果不須進入 Rescue 服務則會檢查韌體，如果為有效韌體則載入並運行 Service (服務)。

透過外部 I2C 與 USB 透過 HAL 控制單元控制晶片功能，兩個路徑可控制階段將不同，由於 USB 介面由服務提供底層 USB 服務，故進入 Rescue 與 Service (服務) 前無法與晶片溝通，而 I2C 則是在系統上電後即可進行讀取狀態與控制操作。

- 下圖為 B 系列裝置，若發生裝置無反應請重啟。





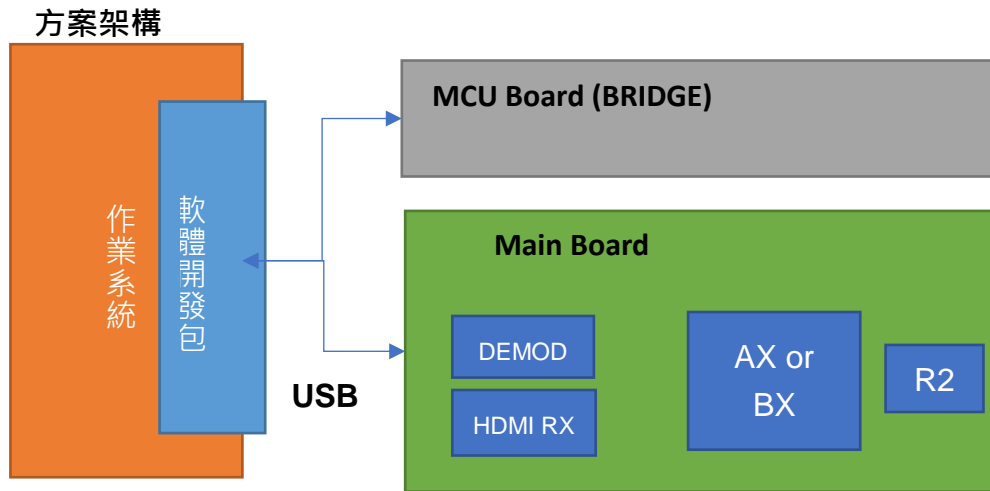
3 軟體開發包(SDK)

軟體開發包針對運行於高階作業系統(Windows、Linux 與 Android)的應用與產品發展而來，透過瞻誠調製晶片的 USB 接口 與高階作業系統連結，可以透過軟體開發包開發包含測試工具、更新工具....等。瞻誠提供的全系列應用工具就是基於此軟體開發包發展而成。

3.1 簡介

軟體開發包以原始碼的方式提供，使用 CMake 作為編譯工具，開發者可以在 Microsoft Windows 10 (以上)、Ubuntu 18.04 (或其他之支援相關編譯環境的系統中)編譯與開發相關應用程式。主要軟體包內容包含：

資料夾		描述
api	core	核心程式開發介面
	qt	基於 QT 的 GUI 元件介面
	calibration	系統調教程式介面
app	vatek_clibration	調教工具
	vatek_factory	韌體影像檔工具
	vatek_romtool	韌體更新工具
	vatek_toolkit	通用廣播工具
build		建置相關工具
extend	bridge_2	MCU Board 韌體原始碼
	obs-vatek	OBS 整合 A 系列範例
	tsduck-vatek	TS-Duck 整合 A 系列範例
sample	app_bridge	MCU Board 控制範例
	app_broadcast	B 系列控制範例
	app_romtool	韌體更新範例
	app_stream	A 系列控制範例
	common	通用原始碼



3.2 核心程式開發介面

軟體開發包主要透過核心程式開發介面提供相關功能，程式相關原始碼於軟體開發包中的 `api\core`，主要功能由根目錄下的程式介面提供，主要分為以下功能類別。

SDK > Release > vatek_sdk_2 > api > core > inc

名稱	修改日期
bridge	2022/1/5 上午 09:34
core	2022/1/6 下午 04:48
cross	2022/1/5 上午 09:34
mux	2022/1/5 上午 09:34
service	2022/1/5 上午 09:34
vatek_sdk_bridge	2021/8/24 下午 03:57
vatek_sdk_broadcast	2022/1/7 上午 10:48
vatek_sdk_device	2021/11/23 上午 11:54
vatek_sdk_storage	2021/8/17 下午 03:24
vatek_sdk_transform	2021/8/13 下午 02:19
vatek_sdk_usbmux	2021/6/22 上午 09:41
vatek_sdk_usbstream	2021/10/17 上午 02:48

程式介面定義	描述
vatek_sdk_bridge	Bridge (MCU Board)的相關控制與操作，可以控制設定開發版上的影音來源。
vatek_sdk_broadcast	B 系列晶片的主要控制程式介面
vatek_sdk_device	連結的調製裝置(A 與 B)的列舉、控制....
vatek_sdk_storgae	裝置資源與韌體相關程式介面
vatek_sdk_transform	A 系列晶片的主要控制程式介面
vatek_sdk_usbmux	A 整合作業系統影音編碼器的程式開發介面
vatek_sdk_usbstream	A 應用於 MPEG-TS 串流的程式介面

程式介面主要可以區分為三個主要類別分別為 Common、Broadcast 與 Transform 等。

3.2.1 Common 類別

- 裝置基本操作功能，例如：列舉系統內支援的相關裝置、裝置開啟及關閉及重啟、Calibration 數值調整，主要透過 **vatek_sdk_device** 來提供相關功能。

- ◆ 裝置列舉 (決定要由哪條 Bus 及晶片去列舉裝置)：

```
HAL_API vatek_result vatek_device_list_enum(uint32_t bus, hal_service_mode
service, hvatek_devices* hdevices);
```

- ◆ 取得列舉裝置服務 (藉由列舉裝置讀取該 Chip 的種類 [A or B])：

```
HAL_API hal_service_mode vatek_device_list_get_service(hvatek_devices
hdevices, int32_t idx);
```

- ◆ 裝置列舉清單清除 (清除裝置列舉所讀取的清單，以保證下次連接不會重複)：

```
HAL_API void vatek_device_list_free(hvatek_devices hdevices);
```

- ◆ 裝置連接 (列舉到的裝置開啟)：

```
HAL_API vatek_result vatek_device_open(hvatek_devices hdevices, int32_t idx,
hvatek_chip* hchip);
```

- ◆ 裝置停止 (將 RF 訊號及 Chip 運作都停止):

```
HAL_API void vatek_device_stop(hvatek_chip hchip);
```

- ◆ 裝置關閉 (將裝置的值清空，以保證下次連接不會重複):

```
HAL_API vatek_result vatek_device_close(hvatek_chip hchip);
```

- ◆ 裝置關閉並重啟 (將裝置的值清空，並且將裝置完全重新啟動):

```
HAL_API vatek_result vatek_device_close_reboot(hvatek_chip hchip);
```

- ◆ Calibration 值讀取 (讀取裝置裡儲存的 Calibration 值):

```
HAL_API vatek_result vatek_device_calibration_load(hvatek_chip hchip,
Pcalibration_param pcalibration);
```

- ◆ Calibration 值應用 (動態調整 Chip 的 Calibration 值):

```
HAL_API vatek_result vatek_device_calibration_apply(hvatek_chip hchip,
Pcalibration_param pcalibration);
```

- ◆ Calibration 值儲存 (將動態調整 Chip 的 Calibration 值儲存到裝置[Flash]):

```
HAL_API vatek_result vatek_device_calibration_save(hvatek_chip hchip,
Pcalibration_param pcalibration);
```

- 裝置儲存空間 Storage 基本組成是使用 64KB 的 loader 及 65KB 開始由 application 而組成，根據所需要的功能在 application 後面加上 header，每個資料都是利用 header 來尋找記憶體位址 (例如：Bootlogo、R2 table、Modulation config 等等設定)，主要透過 vatek_sdk_storage 來提供相關功能。

- ◆ 建立讀取裝置 Storage 的 handle (控制電腦讀取裝置的 Storage):


```
HAL_API vatek_result vatek_storage_create_chip_handle(hvatek_chip hchip,
Pstorage_handle* phandle, fprom_progress fpcb, void* cbparam);
```

- ◆ 建立讀取電腦裡 loader 和 application 的 handle (控制電腦讀取電腦的 loader 和 application) :

```
HAL_API vatek_result vatek_storage_create_file_handle(const char* fimage, const char*
floader, const char* fapp, Pstorage_handle* phandle, fprom_progress fpcb, void*
cbparam);
```

- ◆ 建立讀取電腦裡 v2image 的 handle (控制電腦讀取電腦的 v2image) :

```
HAL_API vatek_result vatek_storage_open_file_handle(const char* filename,
Pstorage_handle* phandle, fprom_progress fpcb, void* cbparam);
```

- ◆ 讀取 Storage 的 application (利用這個功能讀取 application 裡的資訊) :

```
HAL_API vatek_result vatek_storage_get_app(hvatek_storage hstorage,
Papp_header* papp);
```

- ◆ 讀取 Storage 的 loader (利用這個功能讀取 loader 裡的資訊) :

```
HAL_API vatek_result vatek_storage_get_loader(hvatek_storage hstorage,
Ploader_header* ploader);
```

- ◆ 讀取 Storage 的 Bootlogo :

```
HAL_API vatek_result vatek_storage_get_resource(hvatek_storage hstorage,
Pstorage_resource* pres);
```

- ◆ 建立 Bootlogo 所需要的空間及長寬 :

```
HAL_API vatek_result vatek_storage_resource_create(hvatek_storage hstorage,
uint32_t w, uint32_t h, Pstorage_resource* pres);
```

◆ 新增 Bootlogo 到 Storage 裡 :

```
HAL_API vatek_result vatek_storage_add_resource(hvatek_storage hstorage,
Pstorage_resource pres);
```

◆ 刪除 Storage 裡面的 Bootlogo :

```
HAL_API vatek_result vatek_storage_del_resource(hvatek_storage hstorage,
Pstorage_resource pres);
```

◆ 讀取 Storage 的 R2 table :

```
HAL_API Pr2_tune_handle vatek_storage_get_r2tune(hvatek_storage hstorage);
```

◆ 讀取 Storage 是否為 Broadcast 模式 (B 系列) :

```
HAL_API Pstorage_broadcast vatek_storage_get_broadcast(hvatek_storage
hstorage);
```

◆ 讀取 Storage 是否為 Transform 模式 (A 系列) :

```
HAL_API Pstorage_transform vatek_storage_get_transform(hvatek_storage
hstorage);
```

◆ 讀取 Storage 的 Config 設定 (此設定包含 Modulation、Chip 額外功能) :

```
HAL_API Pstorage_chip_config vatek_storage_get_config(hvatek_storage
hstorage);
```

◆ 將 Storage 儲存到 v2image (將 Storage 儲存在電腦檔案) :

```
HAL_API vatek_result vatek_storage_save(hvatek_storage hstorage, const char*
filename);
```

◆ 關閉 Storage :

```
HAL_API vatek_result vatek_storage_close(hvatek_storage hstorage);
```


◆ 建立 Storage 寫入裝置需要的檔案空間及長度：

```
HAL_API vatek_result vatek_storage_romfile_create(const char* romfle,
Promfile_handle* promfile);
```

◆ 將 Storage 寫入裝置：

```
HAL_API vatek_result vatek_storage_write_image(hvatek_storage hstorage,
Promfile_handle pimage);
```

◆ 清除及關閉寫入裝置的 Storage 檔案：

```
HAL_API vatek_result vatek_storage_romfile_free(Promfile_handle promfile);
```

- SDK 設計方案中所有影音來源的周邊控制需要透過 MCU BOARD(BRIDGE)控制，如 B 系列需要由 BT1120 與 I2S 介面由 HDMI RX 晶片提供原始影音資料、A 系列整合 DTV DEMOD 或其他 MPEG-TS 介面作為影音資料來源等，皆須透過 vatek_sdk_bridge 來控制與操作 MCU Board (BRIDGE)。

◆ 連接 Bridge (Bridge 是透過 I2C 連接)：

```
HAL_API vatek_result vatek_bridge_open(hvatek_chip hchip, hvatek_bridge*
hbridge);
```

◆ 讀取 Bridge 資訊：

```
HAL_API Pbdevice_info vatek_bridge_get_info(hvatek_bridge hbridge);
```

◆ 讀取 AV Source 的資訊：

```
HAL_API vatek_result vatek_bridge_get_av_source(hvatek_bridge hbridge,
int32_t idx,Pbridge_source psource);
```

◆ 讀取 AV Source 的名字：

```
HAL_API const char* vatek_bridge_get_av_source_name(hvatek_bridge
hbridge,Pbridge_source psource);
```

◆ 啟動外部來源 AV Source：

```
HAL_API vatek_result vatek_bridge_start_av_source(hvatek_bridge hbridge,
Pbridge_source psource);
```

◆ 取得 AV Source 狀態：

```
HAL_API vatek_result vatek_bridge_get_av_source_status(hvatek_bridge hbridge,
Pbridge_source psource);
```

◆ 停止外部來源 AV Source：

```
HAL_API vatek_result vatek_bridge_stop_av_source(hvatek_bridge hbridge);
```

◆ Bridge 連接中斷：

```
HAL_API void vatek_bridge_close(hvatek_bridge hbridge);
```

3.2.2 Broadcast 類別

- B 系列主要提供由外部影音晶片提供原始影音原始資料結合內部編碼器與調製功能直接輸出數位電視廣播訊號，此一系列完整的功能 Broadcast 服務來提供，高階軟體開發包提供 vatek_sdk_broadcast 程式界面讓使用這可以快速操作整個流程。

◆ B 系列服務開啟 (包含取得裝置資訊、判斷是否為 B 系列、檢查 RFmixer 是否支援、檢查 auxstream 是否支援)：

```
HAL_API vatek_result vatek_broadcast_open(hvatek_chip hchip,hvatek_broadcast*
hbc);
```

◆ 讀取 B 系列裝置資訊 :

```
HAL_API Pbroadcast_info vatek_broadcast_get_info(hvatek_broadcast hbc);
```

◆ B 系列裝置廣播啟動 (包含判斷是否使用 auxstream、RF 訊號啟動):

```
HAL_API vatek_result vatek_broadcast_start(hvatek_broadcast hbc,
Pbroadcast_param pbcparam, Pbroadcast_auxstream paux, uint32_t freqkhz);
```

◆ Auxstream 功能傳送 (auxstream 包含同步及非同步模式):

```
HAL_API vatek_result vatek_broadcast_polling(hvatek_broadcast hbc,
Pbroadcast_info* pinfo);
```

◆ B 系列裝置廣播停止 (包含 RF 訊號停止、裝置停止):

```
HAL_API vatek_result vatek_broadcast_stop(hvatek_broadcast hbc);
```

◆ 檢查是否支援 auxstream :

```
HAL_API vatek_result vatek_broadcast_check_auxstream(hvatek_chip hchip);
```

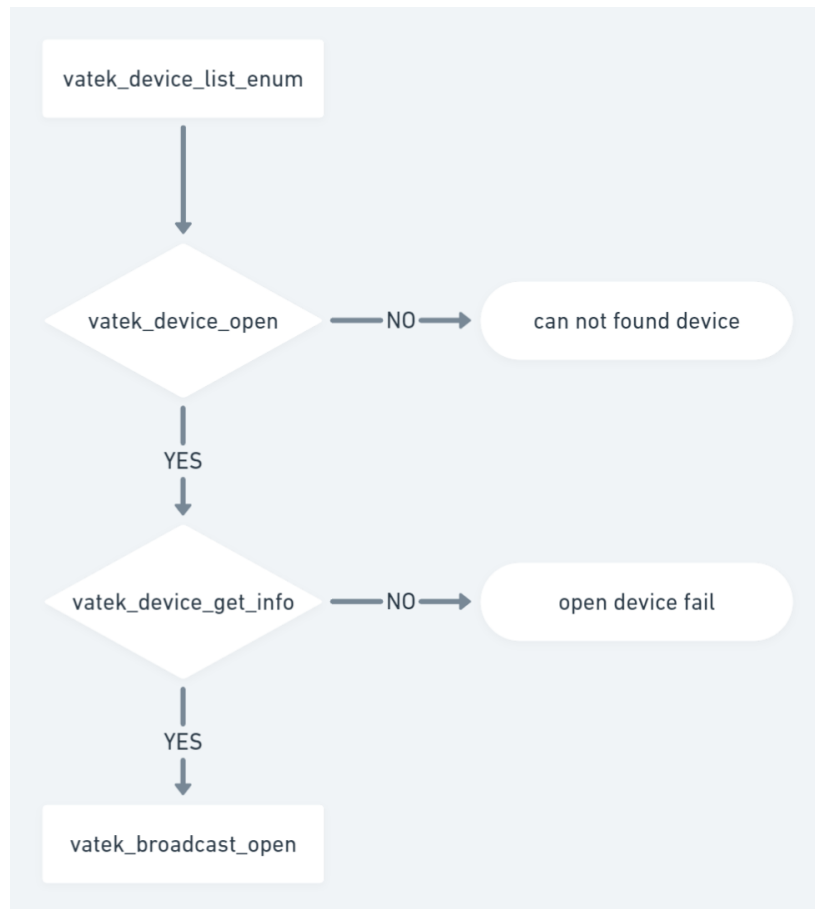
- B 系列主要使用方法以 app_broadcast.c (sample\app_broadcast) 當作範例說明，該程式將示範 B 系列運作的流程。

◆ 更改 B 系列編碼、調變相關參數：

```
/* broadcast parameters*/
static broadcast_param bc_param =
{
    .enc =
    {
#ifdef BROADCAST_EN_COLORBAR
        .mode = encoder_source_colorbar,
#else
        .mode = encoder_source_vi_0,
#endif
        .pmt_pid = 0x1000,
        .recv = 0,
        .encoder_flags = 0,
        .encoder_tag = 0,
        .video = { encvideo_mpeg2,resolution_1080p,framerate_59_94,aspectrate_16_9 },
        .quality = { rc_vbr,3,10,16,500,19000000, },
        .viparam = { EP9555E_VI_FLAG,148500,0,0, },
#ifdef BROADCAST_EN_COLORBAR
        .audio = { encaudio_aac_lc_adts,sample_rate_48,channel_mute, },
#else
        .audio = { encaudio_aac_lc_adts,sample_rate_48,channel_stereo, },
#endif
    }
};
```

```
        .video_pid = 0x1002,
        .audio_pid = 0x1003,
    },
    .mod =
    {
        .bandwidth_symbolrate = 6,
        .type = modulator_dvb_t,
        .ifmode = ifmode_disable,.iffreq_offset = 0,.dac_gain = 0,
        .mod = { .dvb_t = {dvb_t_qam64,fft_8k,guard_interval_1_16,coderate_5_6,},},
    },
    .mux =
    {
        .pcr_pid = 0x100,
        .padding_pid = 0x1FFF,
        .bitrate = 0,0,0,0,
    },
};
```

Step 1：初始化裝置與啟動。



- ◆ 裝置列舉 (此設定是開啟 USB 線路、並指定 service 為 broadcast):

```
nres = vatek_device_list_enum(DEVICE_BUS_USB, service_broadcast, &hdevlist);
```

- ◆ 裝置開啟:

```
nres = vatek_device_open(hdevlist, 0, &hchip);
```

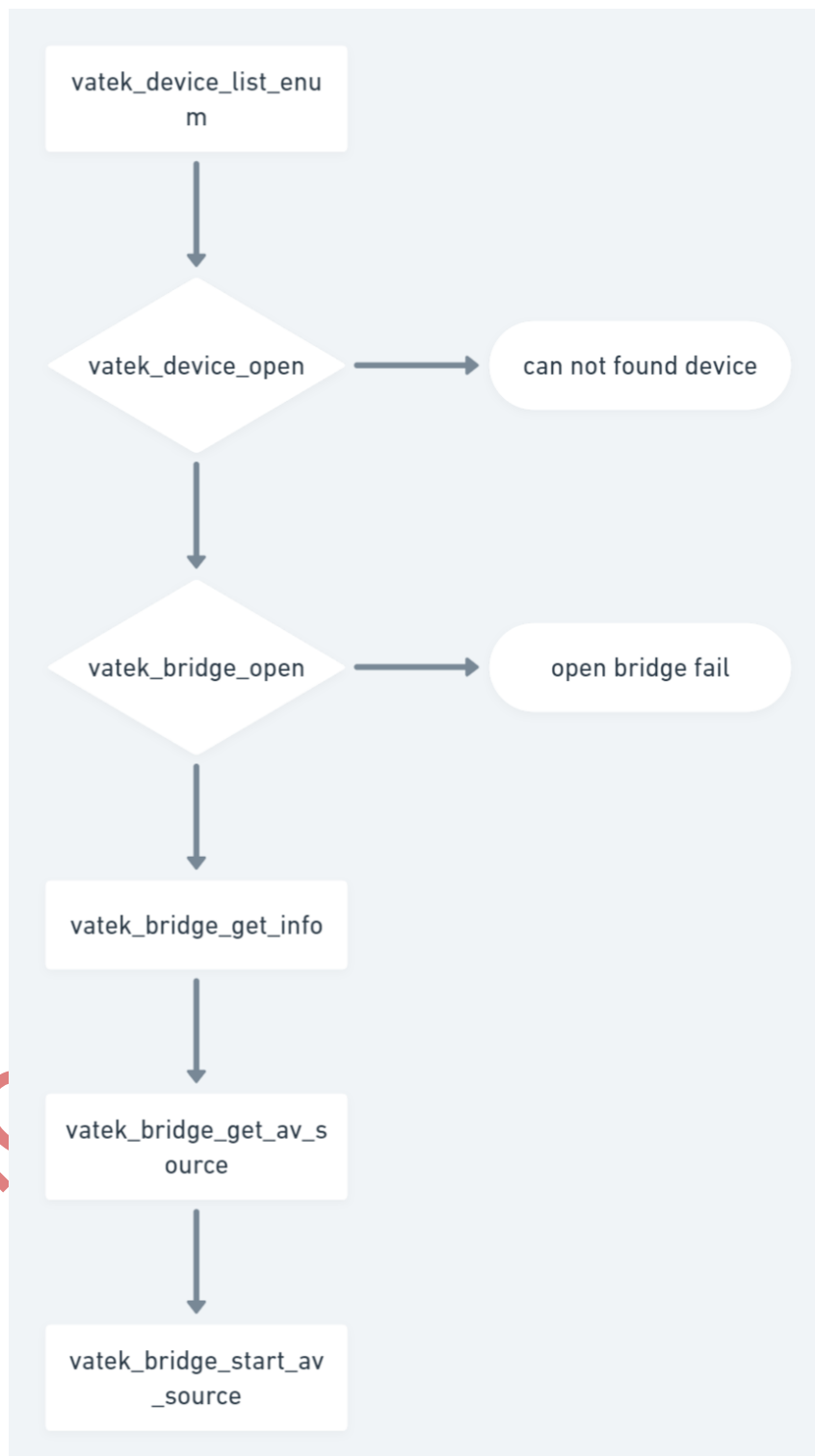
- ◆ 讀取裝置資訊:

```
pinfo = vatek_device_get_info(hchip);
printf_chip_info(pinfo);
```

- ◆ B 系列服務開啟:

```
nres = vatek_broadcast_open(hchip, &hbc);
```

Step 2：檢查 encoder 使用模式並確認影像及音源是否準備好（若 encoder 模式為 colorbar、bootlogo，則只決定晶片 encoder 設定及 modulation 設定、反之 encoder 模式為 AV source，則必須走 bridge 線路才能夠抓取外部訊號源。）



- ◆ 裝置列舉 (此設定是開啟 bridge 線路、並指定 service 為 broadcast):

```
nres = vatek_device_list_enum(DEVICE_BUS_BRIDGE, service_broadcast, &hblists);
```

- ◆ 連接裝置 :

```
nres = vatek_device_open(hblists, 0, &hbchip);
```

- ◆ 連接 bridge :

```
nres = vatek_bridge_open(hbchip, &hbridge);
```

- ◆ 取得 bridge 資訊 :

```
Pbdevice_info pbinfo = vatek_bridge_get_info(hbridge);  
printf_bridge_info(pbinfo);
```

- ◆ 讀取 av source 外部來源 :

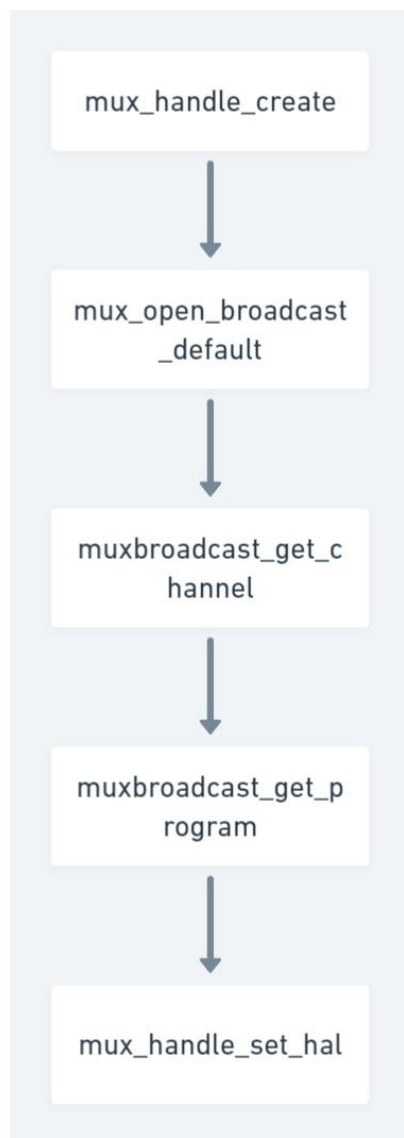
```
nres = vatek_bridge_get_av_source(hbridge, i, &bsource);
```

- ◆ 啟動 av source 外部來源到晶片 :

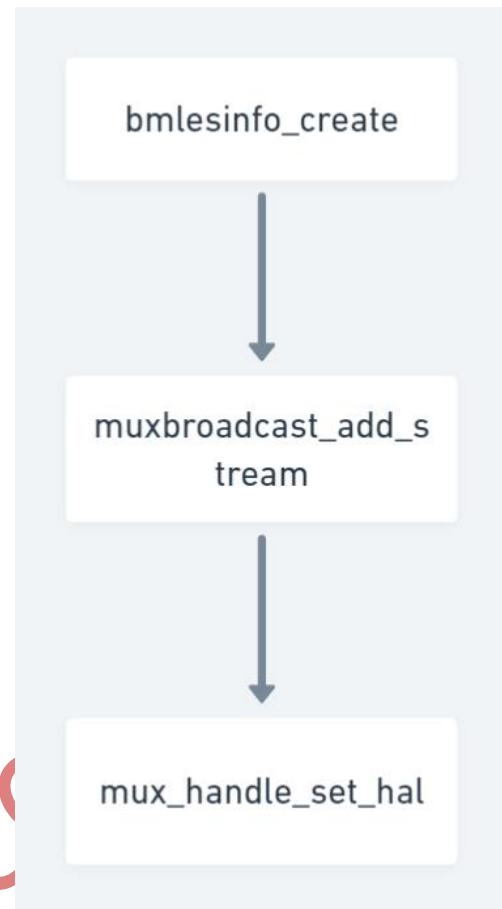
```
nres = vatek_bridge_start_av_source(hbridge, &bsource);
```

Step 3 : 啟動 PSI table 功能、BML 功能 :

PSI table



有開啟 auxstream 功能



◆ 建立 mux 所需空間、大小 :

```
nres = mux_handle_create(&hmux);
```

◆ 使用預設的 PSI table :

```
nres = mux_open_broadcast_default(hmux, pmux->pcr_pid, penc, mux_spec_arib,
arib_japan, &hmuxbc);
```


◆ 讀取 PSI table 頻道 :

```
nres = muxbroadcast_get_channel(hmuxbc, &pch);
```

◆ 讀取 PSI table 節目 :

```
nres = muxbroadcast_get_program(hmuxbc, &pprog);
```

◆ 建立 BML 資訊 :

```
bmlinfo_create(&bmltests[0]);
```

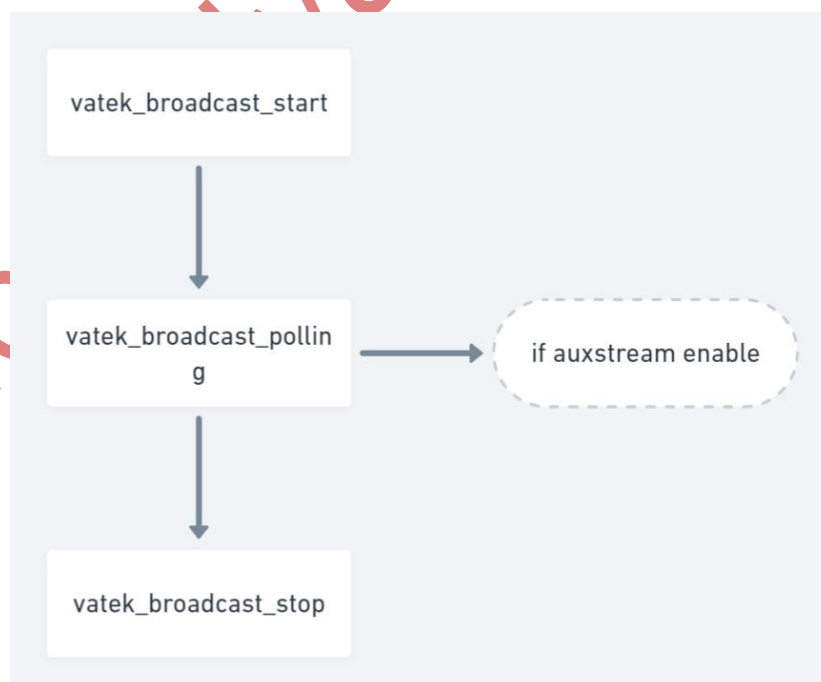
◆ BML 資訊加入 PSI table :

```
muxbroadcast_add_stream(hmuxbc, bmltests[0].pid, STREAMTYPE_ISO13818_6_TYPE_D, &bmltests[0].esbuf[0], bmltests[0].eslen);
```

◆ PSI table 寫入裝置 :

```
nres = mux_handle_set_hal(hmux, hchip);
```

Step 4 : B 系列裝置啟動廣播 :



◆ 裝置啟動廣播：

```
nres = vatek_broadcast_start(hbc, &bc_param, pauxstream, 473000);
```

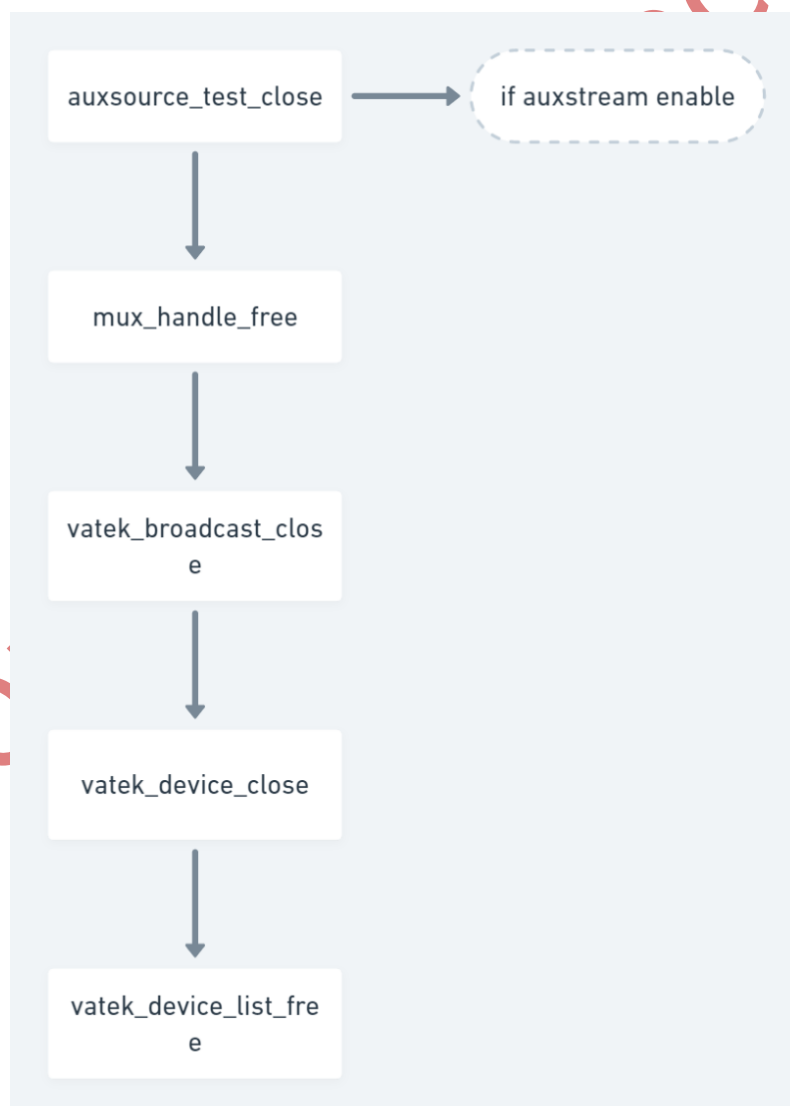
◆ 循環判斷裝置狀態及 Auxstream 功能傳送 (auxstream 包含同步及非同步模式)：

```
nres = vatek_broadcast_polling(hbc, &pbinfo);
```

◆ 裝置停止廣播：

```
nres = vatek_broadcast_stop(hbc);
```

Step 5： 停止程式前釋放及清除裝置及 source：



- ◆ auxsource 功能關閉 (如果有開啟的話):

```
if (pauxstream)auxsource_test_close(pauxstream);
```

- ◆ PSI table 資料釋放 :

```
if (hmux)mux_handle_free(hmux);
```

- ◆ 裝置關閉廣播 (停止廣播並釋放廣播資料):

```
if (hbc)vatek_broadcast_close(hbc);
```

- ◆ 裝置關閉 (釋放裝置):

```
if (hchip)vatek_device_close(hchip);
```

- ◆ 裝置列舉清單釋放 :

```
if (hdevlist)vatek_device_list_free(hdevlist);
```

3.2.3 Transform 類別

- A 系列主要提供由數位電視調變，將接收到的訊號進行再製，提供加入 PSI table 及 PCR 校正的功能，並依照需求轉換成不同調變後輸出 TS 訊號進行廣播，此一完整的功能 Transform 服務來提供，高階軟體開發包提供 vatek_sdk_transform 程式界面讓使用這可以快速操作整個流程。

- ◆ A 系列傳輸服務開啟(包含取得裝置資訊、判斷是否為 A 系列、檢查 RFmixer 是否支援、建立 PSI table 空間):

```
HAL_API vatek_result vatek_transform_open(hvatek_chip hchip,hvatek_transform* htr);
```

- ◆ 列舉 stream 內容 :

```
HAL_API vatek_result vatek_transform_start_enum(hvatek_transform htr, Ptransform_enum penum);
```

◆ A 系列裝置啟動廣播：

```
HAL_API vatek_result vatek_transform_start_broadcast(hvatek_transform htr,
Ptransform_broadcast pbc,uint32_t freqkhz);
```

◆ 循環判斷 A 系列裝置狀態：

```
HAL_API vatek_result vatek_transform_polling(hvatek_transform htr,
Ptransform_info* pinfo);
```

◆ 讀取 A 系列裝置資訊 (包含 current bitrate 、 data bitrate 、 transform 模式...
等等資訊)：

```
HAL_API Ptransform_info vatek_transform_get_info(hvatek_transform htr);
```

◆ 讀取 A 系列裝置封包：

```
HAL_API vatek_result vatek_transform_get_packets(hvatek_transform htr,
uint32_t* pktnums);
```

◆ 清除 A 系列裝置封包：

```
HAL_API vatek_result vatek_transform_commit_packets(hvatek_transform htr);
```

◆ 停止 A 系列裝置廣播 (包含 RF 訊號停止、裝置停止)：

```
HAL_API vatek_result vatek_transform_stop(hvatek_transform htr);
```

◆ A 系列裝置關閉廣播 (停止廣播並釋放廣播資料)：

```
HAL_API void vatek_transform_close(hvatek_transform htr);
```

- A 系列能使用 file 及 udp 及 rtp 輸入 TS 資料流，使 A 系列裝置能夠利用不同來源的 TS 資料流，高階軟體開發包提供 cross / cross_stream 程式界面讓使用這可以快速操作整個流程。

◆ 藉由 ts 檔案資料流傳輸：

```
HAL_API vatek_result cross_stream_open_file(const char* szfilename,
hcross_stream* hcstream);
```

◆ 藉由 udp 資料流傳輸：

```
HAL_API vatek_result cross_stream_open_udp(const char* szurl, hcross_stream*
hcstream);
```

◆ Null packet 測試模式：

```
HAL_API vatek_result cross_stream_open_mux(Pmodulator_param pmod,
hcross_stream* hcstream);
```

◆ 傳輸 stream：

```
HAL_API vatek_result cross_stream_start(hcross_stream hcstream);
```

◆ 取得 stream 區塊：

```
HAL_API vatek_result cross_stream_get_slice(hcross_stream hcstream,uint8_t**
pslice);
```

◆ stream 停止：

```
HAL_API void cross_stream_stop(hcross_stream hcstream);
```

◆ stream 關閉：

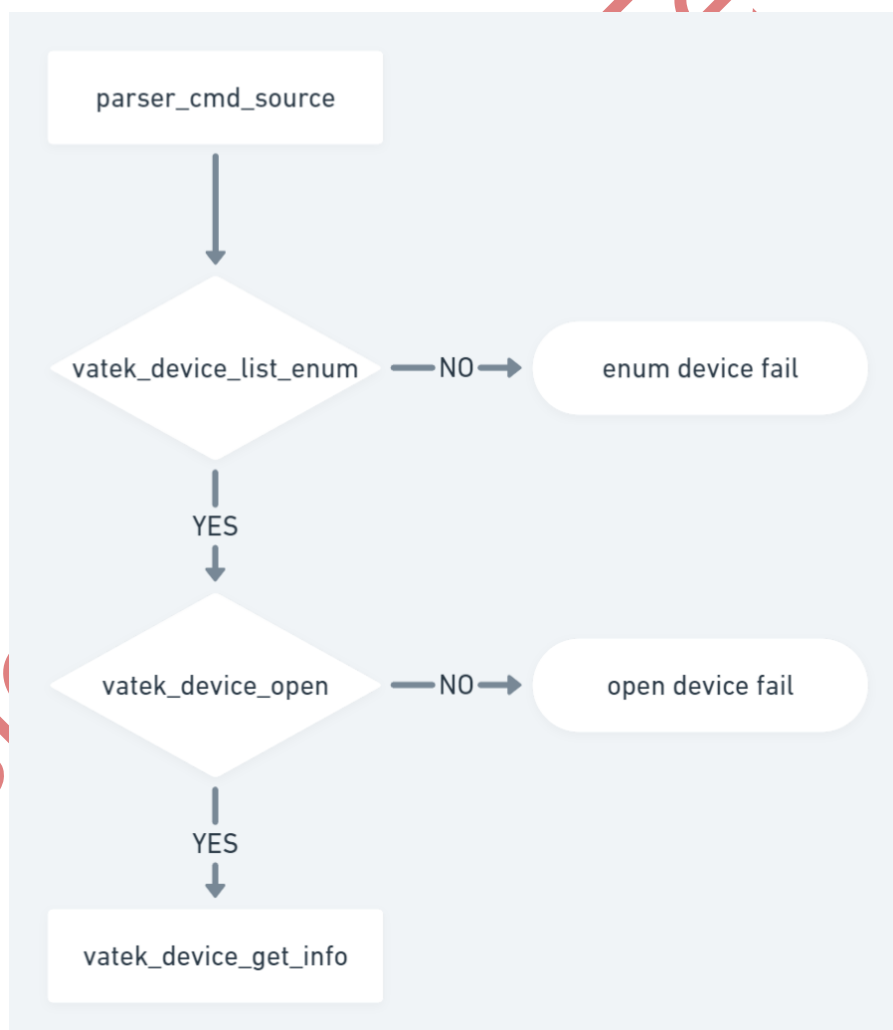
```
HAL_API void cross_stream_close(hcross_stream hcstream);
```

- A 系列主要使用方法以 app_stream.c(sample\app_stream\app_stream.c)當作範例說明，該程式將示範 A 系列運作的流程。

■ 更改調變參數

```
static usbstream_param usbcmd =
{
    .mode = ustream_mode_sync,
    .remux = ustream_remux_passthrough, /* remux */
    .pcradjust = pcr_disable, /* pcr adjustmode */
    .freq_khz = 473000, /* output_rf frequency */
    .modulator =
    {
        6, /* bandwidth or symbolrate */
        modulator_isdb_t, /* modulator type */
        ifmode_disable,0,0, /* dac ifmode */
        .mod = { .dvb_t = { dvb_t_qam64,fft_8k,guard_interval_1_16,code_rate_5_6,},}, /* modulator param */
    },
    .sync = {NULL,NULL},
};
```

■ Step 1：初始化裝置與啟動。



◆ 裝置列舉

```
nres = vatek_device_list_enum(DEVICE_BUS_USB,service_transform,&hdevlist);
```

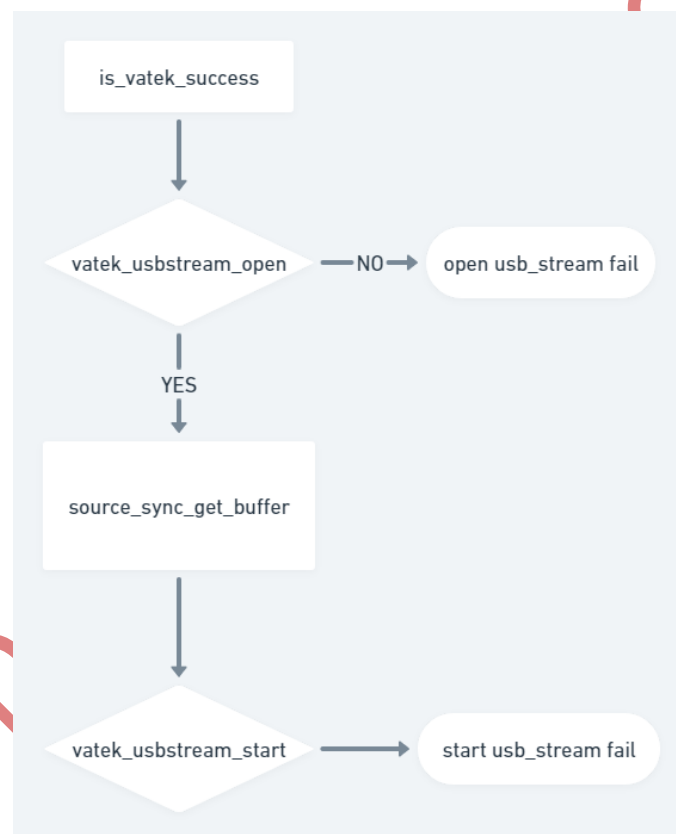
◆ 裝置連接

```
nres = vatek_device_open(hdevlist, 0, &hchip);
```

◆ 讀取裝置資訊

```
Pchip_info pinfo = vatek_device_get_info(hchip);
```

■ Step 2：連接 USB 串流、設定 USB 模式、裝置啟動。



◆ 連接 USB 串流

```
nres = vatek_usbstream_open(hchip, &hustream);
```

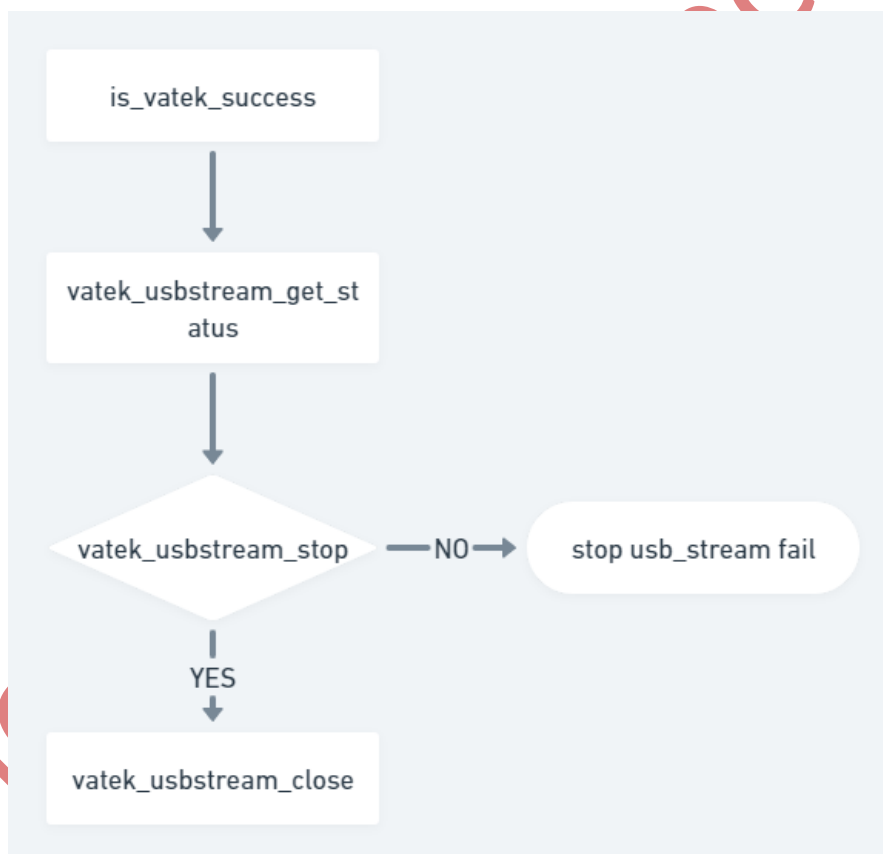
◆ 設定 USB 模式

```
usbcmd.mode = ustream_mode_sync;
usbcmd.sync.param = &streamsource;
usbcmd.sync.getbuffer = source_sync_get_buffer;
```

◆ 串流啟動

```
nres = vatek_usbstream_start(hustream,&usbcmd);
```

- Step 3: 當 `usb_stream` 具有有效緩衝區時，`source_sync_get_buffer` 將從內部調用，可以持續檢查 `usb_stream` 狀態和資訊，完成後關閉 `usb_stream`。



◆ 讀取串流狀態

```
usbstream_status status = vatek_usbstream_get_status(hustream,&pinfo);
```


◆ 串流狀態及資訊

```
pinfo->info.status,  
pinfo->info.data_bitrate,  
pinfo->info.cur_bitrate,
```

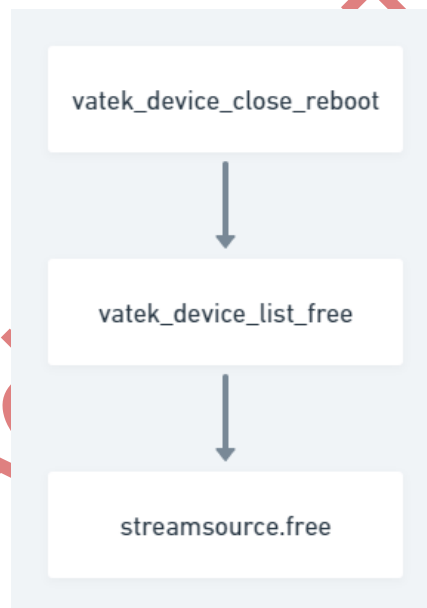
◆ 串流停止

```
nres = vatek_usbstream_stop(hustream);
```

◆ 串流關閉

```
vatek_usbstream_close(hustream);
```

■ Step 4 : 裝置重置及清除。



◆ 裝置重置

```
vatek_device_close_reboot(hchip);
```

◆ 裝置列表清除

```
vatek_device_list_free(hdevlist);
```

◆ 串流資源清除

```
streamsource.free(streamsource.hsource);
```

4 SDK 編譯

SDK 軟體開發包針對運行於高階作業系統(Windows、Linux 與 MacOS [開發中]) 的應用與產品發展，開發者可以在 Microsoft Windows 10、Ubuntu 18.04 (或其他之支援相關編譯環境的系統中) 編譯與開發相關應用程式，編譯工具目前以 CMake 為主。

4.1 編譯的需求：

- Microsoft Visual Studio 2019 (<https://visualstudio.microsoft.com/>) · 使用商業版本與社群免費版本皆可。
 - ◆ 安裝 Visual Studio 時請確保有選擇 C++ 的桌面開發
- CMake (<https://cmake.org/>) Windows 版本 (3.22 版本以上)
- Qt5 (適用於你的 Visual Studio 版本的 MSVC 包)
 - ◆ 目前版本為 Qt 5.14.2 開發 ([Qt5.14.2](#))

4.2 編譯選項

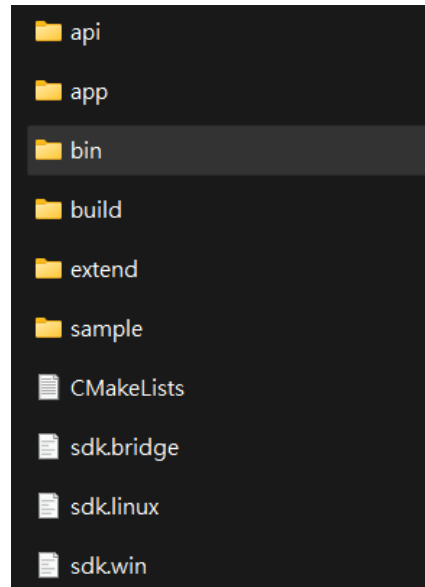
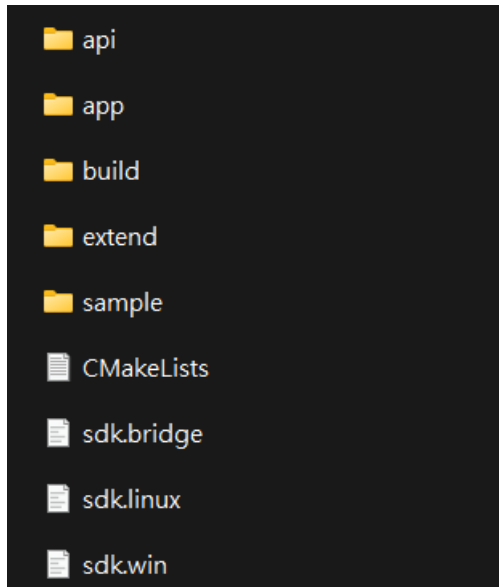
於 CMAKE 建置環境中提供下列參數可於編譯前針對需求設置。

參數	說明	數值
SDK2_EN_QT	是否編譯 QT 程式開發介面與應用程式	ON OFF
SDK2_EN_APP	是否編譯應用程式	ON OFF
SDK2_EN_SAMPLE	是否編譯範例程式	ON OFF
SDK2_QTDIR	設置使用的 QT 安裝資料夾	

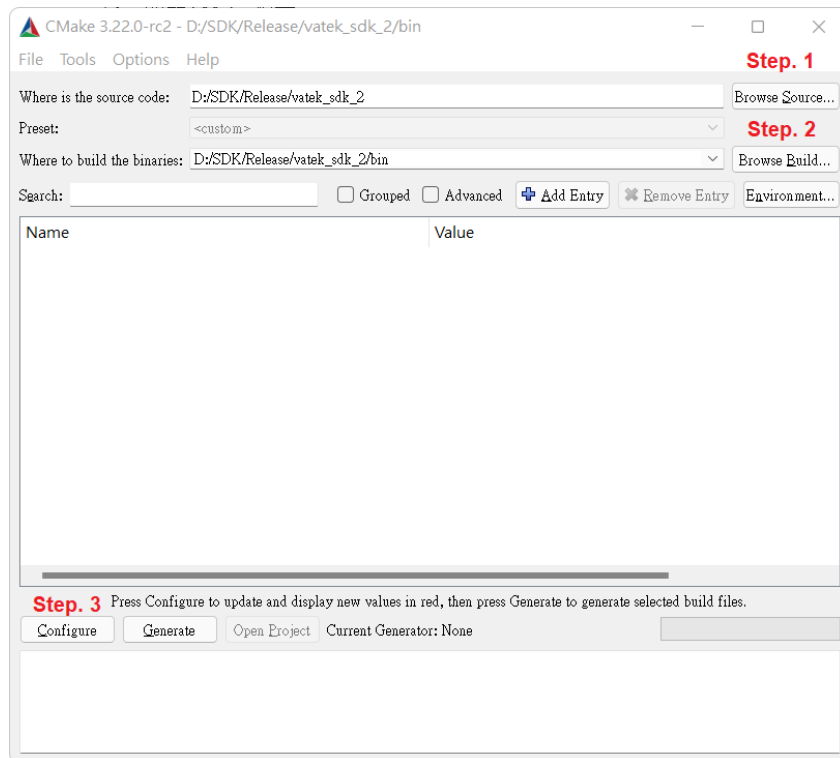
4.3 編譯 vatek_sdk_2 :

4.3.1 Windows 作業系統

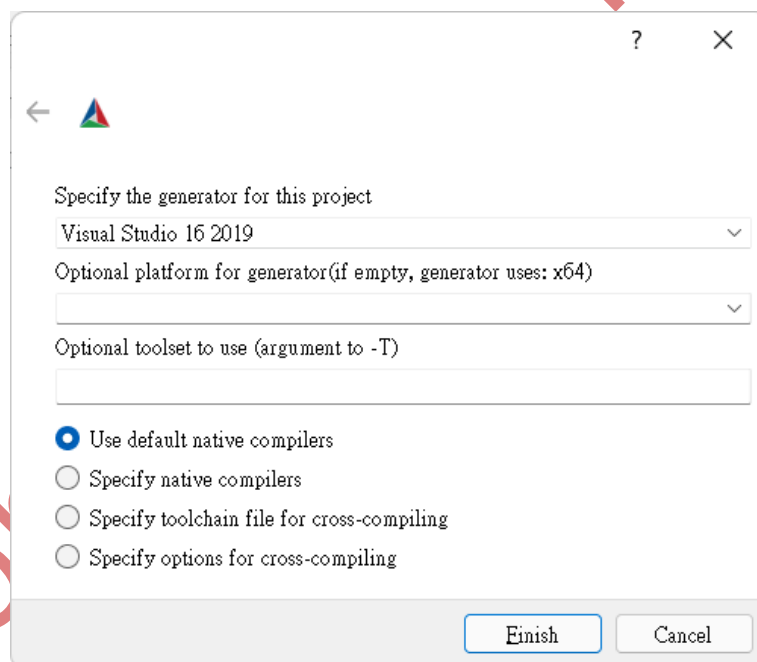
- 解壓縮 vatek_sdk_2.zip，可以看到下列的資料夾，可以選擇在 vatek_sdk_2 資料夾下新增一個資料夾，存放 CMake 編譯過後的檔案（以 bin 資料夾為例）



- 開啟 CMake 程式
 - ◆ **Step. 1** 在 source code 的部分選擇 vatek_sdk_2 資料夾
 - ◆ **Step. 2** 在 build the binaries 的部分選擇 vatek_sdk_2 資料夾裡的 bin 資料夾
 - ◆ **Step. 3** 點選 Configure 按鈕

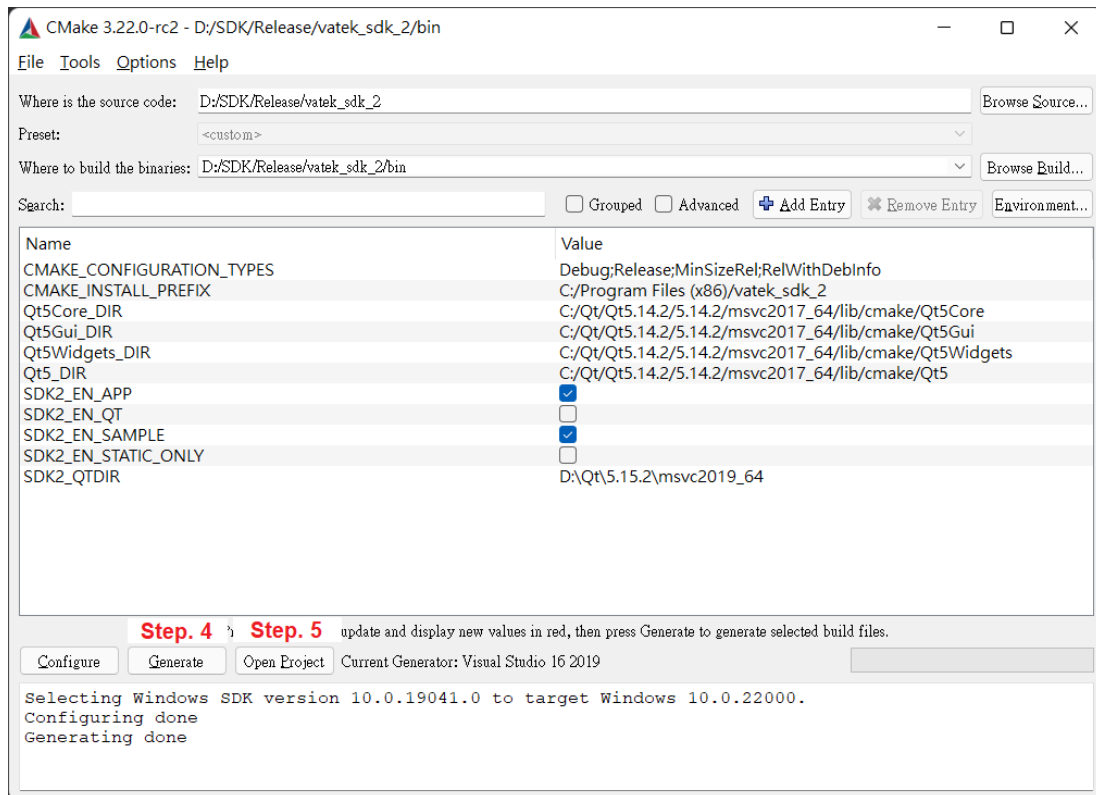


- 選擇 Visual Studio 2019 版本，預設使用 x64 編譯



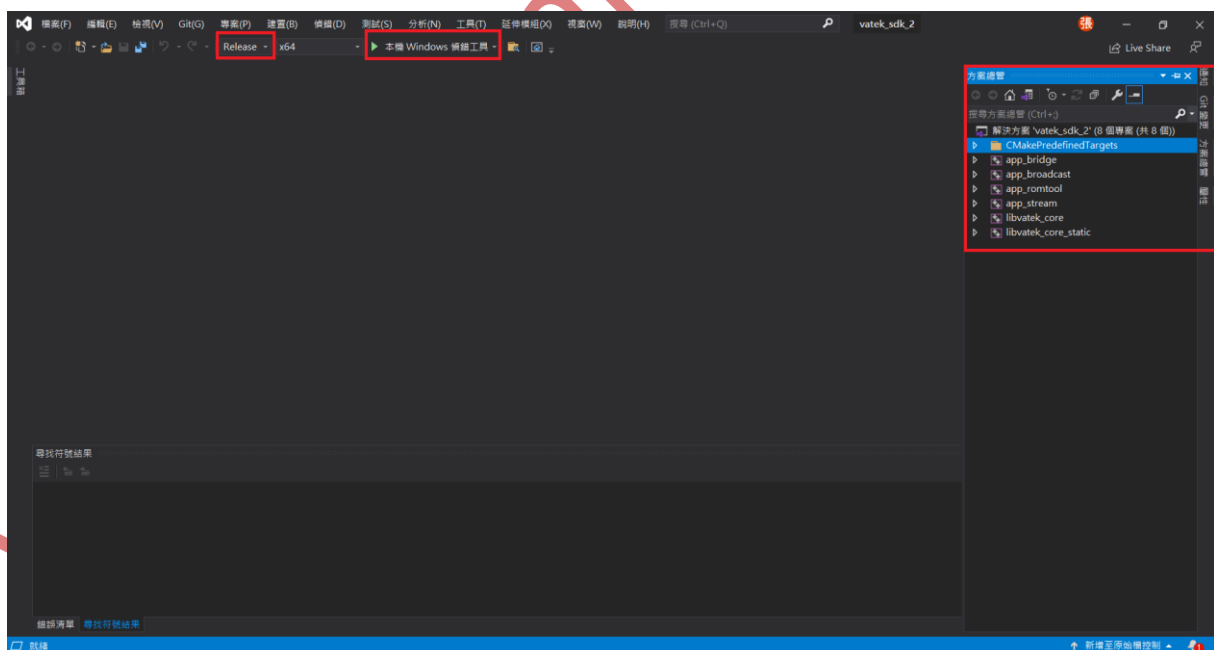
- ◆ **Step. 4** 點選 Generate 按鈕成功編譯會出現此視窗

- (可以選擇是否要使用 QT 介面，若不使用請將 SDK2_EN_QT 取消勾選)



◆ Step. 5 開啟專案檔

- 開啟 Visual Studio 後右邊方案總管可以看到 vatek_sdk_2 共有 6 個方案，偵錯工具旁邊可以選擇 Debug 模式或 Release 模式去執行編譯

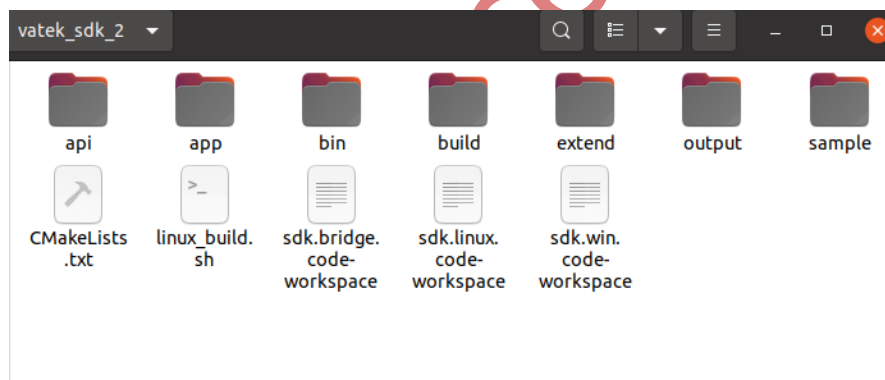


- 執行編譯成功之後 bin 資料夾裡面會有 Debug 資料夾或 Release 資料夾，裡面會包含能夠執行的 exe 檔及所有 api 的 dll (libvatek_core.dll)

名稱	修改日期	類型	大小
app_bridge.exe	2022/1/27 上午 09:29	應用程式	18 KB
app_broadcast.exe	2022/2/14 下午 04:12	應用程式	30 KB
app_romtool.exe	2022/1/27 上午 09:29	應用程式	17 KB
app_stream.exe	2022/2/18 上午 10:28	應用程式	29 KB
libvatek_core.dll	2022/2/22 下午 02:42	應用程式擴充	366 KB

4.3.2 Ubuntu 作業系統

- 解壓縮 vatek_sdk_2.zip，可以看到下列的資料夾，可以選擇在 vatek_sdk_2 資料夾下新增一個資料夾，存放 CMake 建置完成的檔案 (以 bin 資料夾為例)。

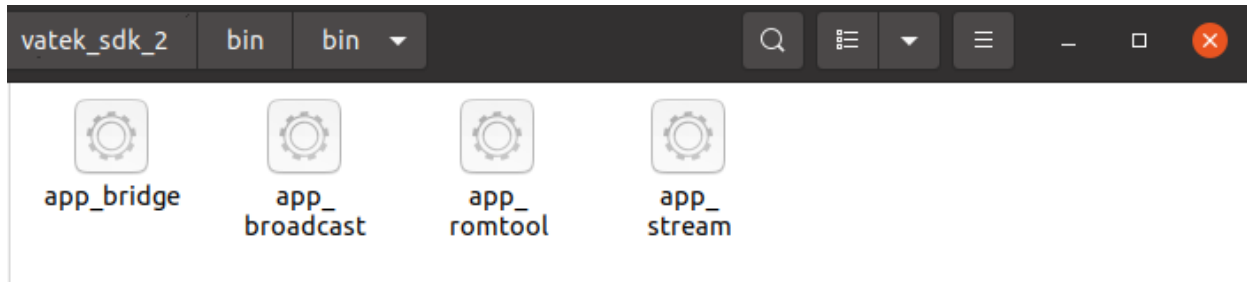


- 開啟 Terminal 將目前目錄移動到 `vatek_sdk_2/bin` 資料夾，執行 `cmake ..` 完成 CMake 軟體建置，建置完成的檔案都會放在 `vatek_sdk_2/bin` 資料夾裡，成功之後執行 `make` 開始編譯 SDK 的檔案。

```
richie@ubuntu:~$ ls
Desktop  Downloads  Music      Public     vatek_sdk_2  video
Documents fontconfig Pictures  Templates  vatek_sdk_2.zip Videos
richie@ubuntu:~$ cd vatek_sdk_2/bin
richie@ubuntu:~/vatek_sdk_2/bin$ cmake ..
-- The C compiler identification is GNU 9.3.0
-- The CXX compiler identification is GNU 9.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/richie/vatek_sdk_2/bin
richie@ubuntu:~/vatek_sdk_2/bin$ make -j8
```

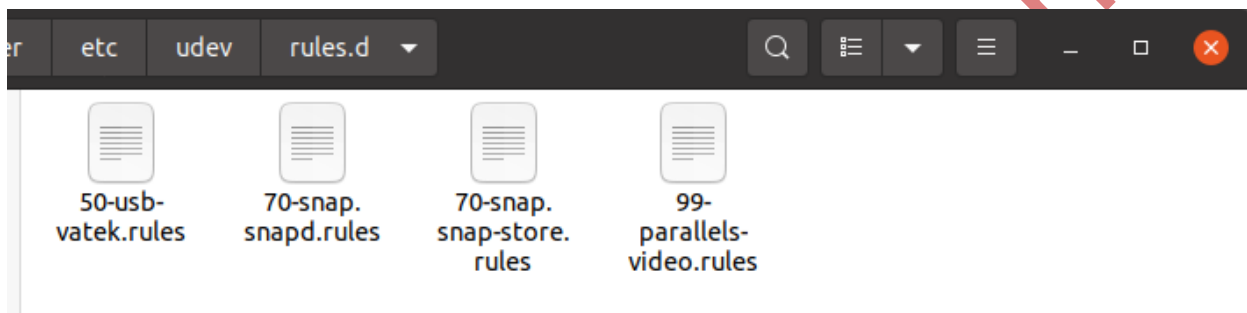
- 編譯成功後會顯示 100%成功，代表過程中都沒有出錯，可以在 bin 資料夾下看到成功編譯出來的 sample code。

```
richie@ubuntu: ~/vatek_sdk_2/build/resource/linux
[ 90%] Linking C executable ../../bin/app_romtool
[ 90%] Building C object sample/app_broadcast/CMakeFiles/app_broadcast.dir/__/common/src/tool_stream_file.c.o
[ 92%] Linking C executable ../../bin/app_bridge
[ 92%] Building C object sample/app_stream/CMakeFiles/app_stream.dir/__/common/src/tool_stream_test.c.o
[ 93%] Building C object sample/app_stream/CMakeFiles/app_stream.dir/__/common/src/tool_stream_udp.c.o
[ 94%] Building C object sample/app_stream/CMakeFiles/app_stream.dir/__/common/src/tool_tspacket.c.o
[ 96%] Building C object sample/app_broadcast/CMakeFiles/app_broadcast.dir/__/common/src/tool_stream_test.c.o
[ 96%] Built target app_romtool
[ 96%] Building C object sample/app_broadcast/CMakeFiles/app_broadcast.dir/__/common/src/tool_stream_udp.c.o
[ 97%] Building C object sample/app_broadcast/CMakeFiles/app_broadcast.dir/__/common/src/tool_tspacket.c.o
[ 97%] Built target app_bridge
[ 98%] Linking C executable ../../bin/app_stream
[100%] Linking C executable ../../bin/app_broadcast
[100%] Built target app_stream
[100%] Built target app_broadcast
```



- 在 Ubuntu 系統裡要能夠辨識我們的裝置必須將 USB 辨識檔案移動到系統資料夾裡，在 build\resource\linux 資料夾裡有檔名 linux_build.sh 的檔案，該檔案為 script 能夠自動將檔案安裝好，執行之後務必檢查是否有成功，否則在 Ubuntu 裡無法辨識裝置，安裝成功請重新開機。

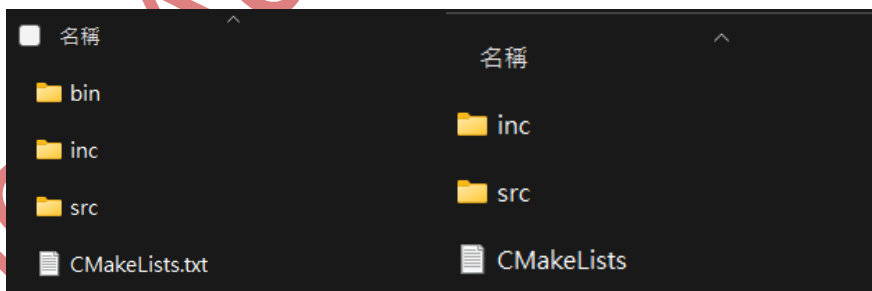
```
richie@ubuntu:~/vatek_sdk_2/build/resource/linux$ sudo ./linux_build.sh
```



4.4 編譯 TSDuck_plugins (A3 裝置支援):

4.4.1 Windows 作業系統

下圖資料夾是 TS Duck 整合 vatek_sdk_2 的開發包，創立 bin 資料夾存取 CMake 編譯後的檔案。



- 請依據電腦需求路徑去更改 CMakeLists 檔案
 - ◆ SDK2_DIR : vatek_sdk_2 資料夾位置
 - ◆ TSDUCK_BUILDCONFIG : 選擇 Debug 模式或 Release 模式
 - ◆ TSDUCK_DIR : TSDuck 資料夾位置 (請自行連接 TSDuck 資料夾位

置，每個人版本及位置不同，此 CMakeLists 只是範例)

```

CMakeLists.txt - 記事本
檔案 編輯 檢視

project(tsduck-vatek)
set_property(GLOBAL PROPERTY USE_FOLDERS ON)
set(TSDUCK_VATEK_PLUGIN tsplugin_vatek)
set(TSDUCK_VATEK_TOOL tsvatek)

set(CMAKE_CXX_STANDARD 14)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

set(SDK2_DIR "../../" CACHE STRING "default vatek_sdk_2 folder")
set(TSDUCK_BUILDCONFIG "Debug" CACHE STRING "build configure type")
set(TSDUCK_DIR "../../output/tsduck-3.29-2651" CACHE STRING "default tsduck source folder")
set(SDK2_EN_STATIC_ONLY ON)
set(SDK2_EN_QT OFF)
set(SDK2_EN_APP OFF)
set(SDK2_EN_SAMPLE OFF)

if(NOT EXISTS ${CMAKE_CURRENT_SOURCE_DIR}/${SDK2_DIR}/)
    message(FATAL_ERROR "vatek_sdk_2 not found please set SDK2_DIR before config")
endif()

if(NOT EXISTS ${CMAKE_CURRENT_SOURCE_DIR}/${TSDUCK_DIR}/)
    message(FATAL_ERROR "tsduck source not found please set TSDUCK_DIR before config")
endif()

if(MSVC)
    if(CMAKE_SIZEOF_VOID_P EQUAL 8)
        set(TSDUCK_LIB_DIR ${TSDUCK_DIR}/bin/${TSDUCK_BUILDCONFIG}-x64)
    else()
        set(TSDUCK_LIB_DIR ${TSDUCK_DIR}/bin/${TSDUCK_BUILDCONFIG}-Win32)
    endif()

    if(NOT EXISTS ${CMAKE_CURRENT_SOURCE_DIR}/${TSDUCK_LIB_DIR}/tsduck.lib)
        message(FATAL_ERROR "tsduck must build tsduck dll first ${TSDUCK_LIB_DIR}")
    endif()
endif()
    
```

- 執行 CMake 建置完成後會產生 Visual Studio 專案檔 (.sln)，開啟執行後進行編譯，編譯成功之後 bin 資料夾裡面會有 Debug 資料夾或 Release 資料夾，裡面會包含能夠執行的 exe 檔及所有 api 的 dll 檔，執行 tsvatek.exe – help 會出現 tsvatek options。

名稱	修改日期	類型	大小
tsplugin_vatek.dll	2022/2/18 下午 05:44	應用程式擴充	1,657 KB
tsplugin_vatek.pdb	2022/2/21 下午 01:31	Program Debug ...	8,348 KB
tsvatek.exe	2022/2/21 上午 09:13	應用程式	1,328 KB
tsvatek.pdb	2022/2/21 上午 09:36	Program Debug ...	7,284 KB

```
C:\Windows\System32\cmd.exe
D:\SDK\Release\vatek_sdk_2\extend\tsduck-vatek\bin\Debug>tsvatek.exe --help
VATEK modulation devices control
Usage: tsvatek
Options:
  --debug[=level]
    Produce debug traces. The default level is 1. Higher levels produce more
    messages.
  -d value
  --device value
    Specify the VATEK index number to list. Show detail device information
  --help
    Display this help text.
  -v
  --verbose
    Produce verbose output.
  --version
    Display the TSDuck version number.
D:\SDK\Release\vatek_sdk_2\extend\tsduck-vatek\bin\Debug>
```

- 產生的插件可以放到 TSDuck 裡的 bin 資料夾，與 TSDuck 套件做連結並使用，例如使用 tsp 程式輸入檔案再輸出到 vatek 裝置。

```
tsp -I file --infinite "D:\Video\atsc-cea-708-dtvcc-and-cea-608.ts" -O vatek
-m ISDB-T --freq 474000 --bandwidth 8 --cons 16-QAM -r 3/4 --guard 1/32 --
trans 8K
```

```
D:\SDK\Code\SDK_master_20220111\master\vatek_sdk_2\output\tsduck-3.29-2651\bin\Debug-x64>tsp -I file --infinite "D:\Video\atsc-cea-708-dtvcc-and-cea-608.ts" -O vatek -m ISDB-T --freq 474000 --bandwidth 8 --cons 16-QAM -r 3/4 --guard 1/32 --trans 8K
* vatek: modulation start - [./1-1//vatek-usb:isdb_t:]
```

5 軟體開發包主應用

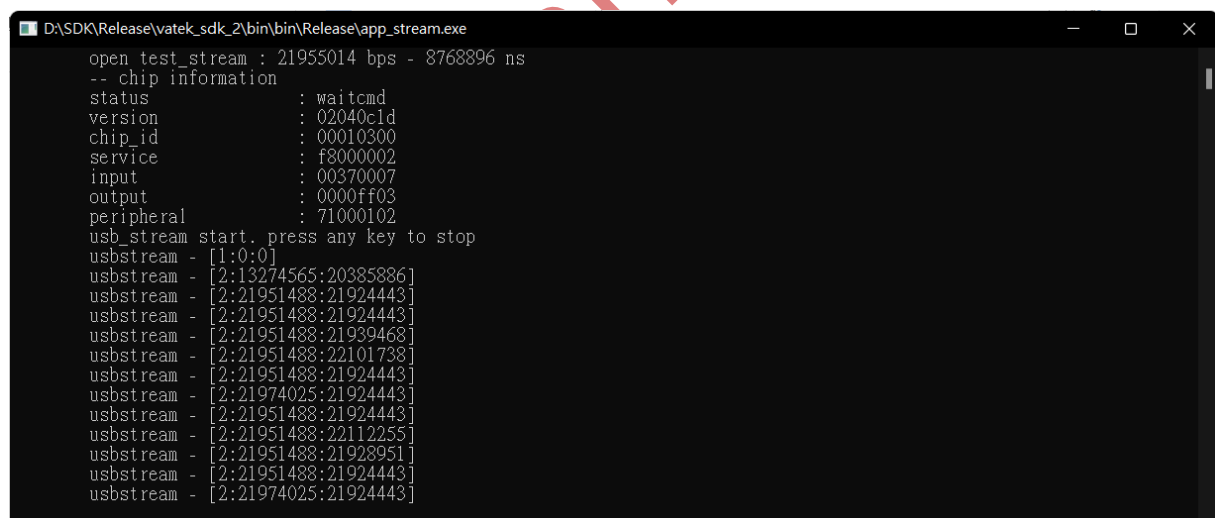
軟體開發包包含開發 USB 串流轉換數位電視所需要的程式開發介面與範例程式，主要的程式介面請參閱路徑 `vatek_sdk_2/api/core/inc` 內的相關程式介面定義，主要搭配下列範例程式去使用。

5.1 app_stream 範例程式 (A 系列使用)

`app_stream` 原始碼放置於 `vatek_sdk_2/sample/app_stream`，此範例完整示範透過 USB 進行數位電視轉換的相關操作流程及調變參數，請參閱原始檔 `app_stream.c` 的相關實作。編譯完成的 `app_stream` 具備下列功能。

```
D:\SDK\Code\SDK_master_20220111\master\vatek_sdk_2\bin\bin\Release>app_stream --help
support command below :
- app_stream [empty] : test stream mode
- app_stream file [*.ts|*.trp] [pcrlpassthrough]
- app_stream udp [ip address] [pcrlpassthrough]
- app_stream rtp [ip address] [pcrlpassthrough]
demo finished. press any key to quit
```

- 當裝置開始廣播時，會出現下列畫面
- `usbstream - [狀態：資料 Bitrate：目前 Bitrate]` → 有 Bitrate 代表正常運作



```
D:\SDK\Release\vatek_sdk_2\bin\bin\Release\app_stream.exe
open test_stream : 21955014 bps - 8768896 ns
-- chip information
status      : waitcmd
version     : 02040c1d
chip_id     : 00010300
service     : f8000002
input       : 00370007
output      : 0000ff03
peripheral  : 71000102
usb_stream start. press any key to stop
usbstream - [1:0:0]
usbstream - [2:13274565:20385886]
usbstream - [2:21951488:21924443]
usbstream - [2:21951488:21924443]
usbstream - [2:21951488:21939468]
usbstream - [2:21951488:22101738]
usbstream - [2:21951488:21924443]
usbstream - [2:21974025:21924443]
usbstream - [2:21951488:21924443]
usbstream - [2:21951488:22112255]
usbstream - [2:21951488:21928951]
usbstream - [2:21951488:21924443]
usbstream - [2:21974025:21924443]
```

1. `app_stream [empty]` 不使用額外的參數

透過 USB 裝置寫入一個由程式產生的 MPEG-TS 串流至調變裝置，轉換為定義數位電視調變。

2. `app_stream file [mpegts_filename]` 使用 `file` 與一個訂定的 mpeg-ts 串流檔案。將一個現成的 MPEG-TS 串流檔案轉換為數位電視調變，支援 188 或是 204 格式，必須包含至少一個 PCR。

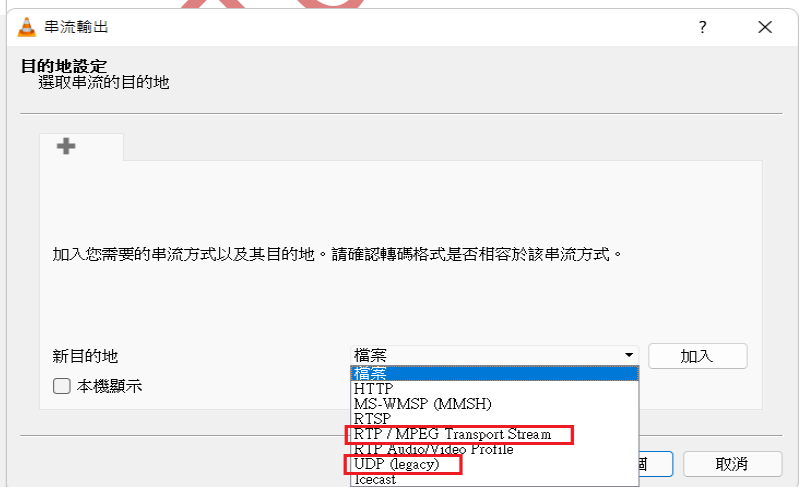
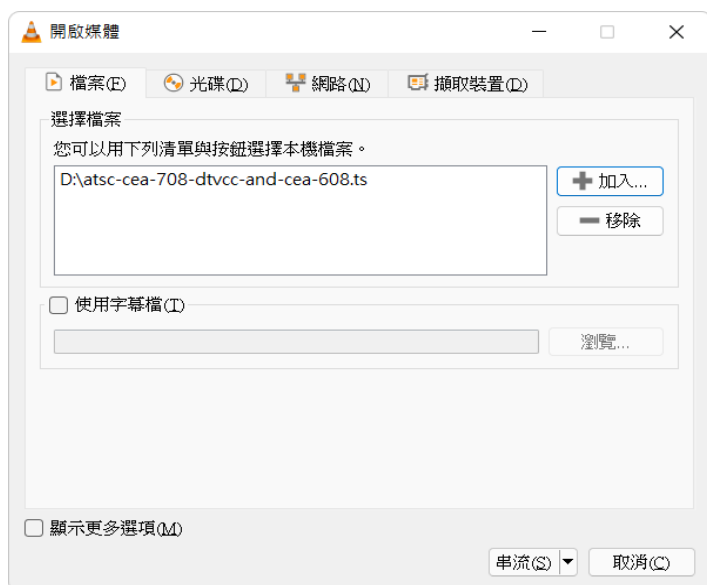
```
D:\SDK\Release\vatek_sdk_2\bin\bin\Release>app_stream file "D:\atasc-cea-708-dtvcc-and-cea-608.ts" pcr
```

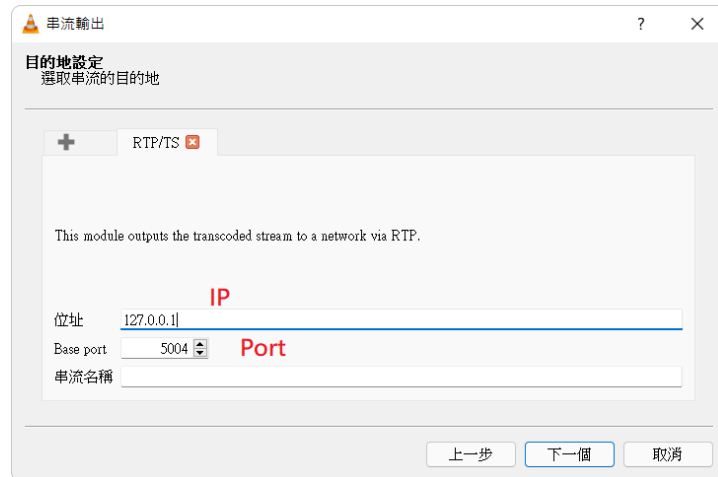
3. `app_stream udp udp://xxx.xxx.xxx.xxx:yyyy` 使用 UDP-MPEGTS 做為來源

```
D:\SDK\Release\vatek_sdk_2\bin\bin\Release>app_stream udp udp://127.0.0.1:5004 pcr
```

4. `app_stream rtp rtp://xxx.xxx.xxx.xxx:yyyy` 使用 RTP-MPEGTS 做為來源
網址格式範例為 `rtp://127.0.0.1:1234` 前面 xxx 為標準 IP 位置 yyyy 為 port，對應 VLC 的設置可以使用如下的選項做為測試來源。

```
D:\SDK\Release\vatek_sdk_2\bin\bin\Release>app_stream rtp rtp://127.0.0.1:5004 pcr
```





5.2 app_broadcast 範例程式 (B 系列使用)

app_broadcast 原始碼位置於 vatek_sdk_2/sample/app_broadcast，此範例完整示範控制 B 系列進行編碼與廣播的相關操作流程。運行後會依據設置調變參數，使 B 系列裝置輸出 COLORBAR，請參閱原始檔 app_broadcast.c 的相關實作。

- 當裝置開始正常運作時，會出現下列畫面
- usbstream – [狀態：資料 Bitrate：目前 Bitrate] → 有 Bitrate 代表正常運作

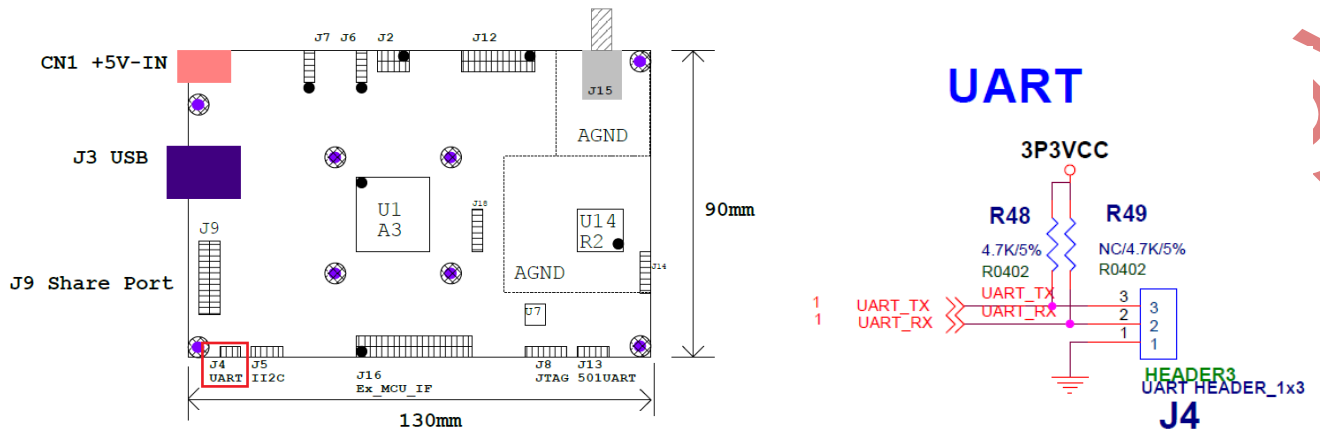
```

C:\Windows\System32\cmd.exe - app_broadcast
Microsoft Windows [版本 10.0.22000.376]
(c) Microsoft Corporation. 著作權所有，並保留一切權利。
D:\SDK\Release\vatek_sdk_2\bin\bin\Release>app_broadcast
-- chip information
status      : waitcmd
version     : 02050105
chip_id     : 00020301
service     : f8000001
input       : 0030f139
output      : 0000ff03
peripheral  : 01000302
connect to bridge device .... [-9]
broadcast start. press any key to stop
broadcast - [2:1137102:1223032]
broadcast - [2:1150593:1236555]
broadcast - [2:1150593:1236555]
broadcast - [2:1150593:1236555]
broadcast - [2:1150593:1236555]
broadcast - [2:1150593:1236555]
broadcast - [2:1150593:1238057]
broadcast - [2:1150593:1236555]
broadcast - [2:675516:749746]
broadcast - [2:1150593:1236555]
broadcast - [2:1150593:1236555]
broadcast - [2:1150593:1236555]

```

5.3 Vatek 裝置 Log 查看

開發裝置時可以連接裝置 Log 檢查是否正常，下圖是我們裝置的 mainboard 圖，我們提供 UART 介面可以連接裝置的 Log，只要連接 J4 (TX、RX、GND)，若是成功連接裝置，會顯示下列圖的 Log。



- 紅色方框的部分說明裝置的額外開能是否有開啟，例如：狀態燈、客製化 usb id。

```
vat-transform-2
- power by vatek technology inc.

[00000037:main ] - initializing units [transform service]...
[0000003c:main ] - start [chip] unit...
                  found chip_id : [a3]
                  pll      : 384 MHz
                  output   : 384 MHz
                  mod       : 96 MHz
[0000006f:main ] - start [memory] unit...
                  memory ip : [00000001] - [0]
                  - [system      :00100000:00000001] : [00000000:00:00]
                  - [highspeed   :00004000:00000002] : [08000000:00:00]
                  - [mempool     :00080000:00000003] : [00180000:00:00]
                  - [REMUX       :00400000:00011005] : [00200000:16:20]
                  - [MODULATOR  :00a00000:00011004] : [00600000:08:16]
                  storage initial : [0]
                  storage sections : 0
                  section[00085000] - [a7b60020:00001000]
                  section[00086000] - [a7b60004:00001000]
                  section[00087000] - [a7b60040:00001000]
                  chip config : [200507ff]
                  - disable r2 extend R
                  - enable dac extend R
                  - disable status led
                  - disable usb customized id
                  - disable usb customized string
[00000152:main ] - start [rfmixer_r2] unit...
                  check peripheral [rfmixer_r2] : [support:0]
[00000169:main ] - start [usb device] unit...
                  default usb - [2c42:1031:VAT-Device]
                  usb device ip : [00000002:2c42:1031] - [0]
                  units initialization [0]
```

- 紅色方框依序為 calibration 參數、R2 table、chip 型號、服務、Firmwave 版本。

```
[00000181:service ] - initializing core service [transform service]...
calibration reset default
calibration - [82080300:20210801] - [0] - [0:0:0:0]
              - 0 [04:83:00:04] - 1 [04:83:00:04]
r2 chip_id : [0101:1508]
r2 hw_rule : [I-04:Q-83:IMAGE-00:PHASE-04]
r2 tune table mode : [12]
- function flags : [0-00000001:1-00000000]
- [00: 79000] - [0-04:83:00:04:3033:21] - [1-04:83:00:04:3033:21]
- [01: 135000] - [0-04:83:00:04:3033:2a] - [1-04:83:00:04:3033:2a]
- [02: 255000] - [0-04:83:00:04:3033:33] - [1-04:83:00:04:3033:33]
- [03: 435000] - [0-04:83:00:04:3043:1e] - [1-04:83:00:04:3043:1e]
- [04: 495000] - [0-04:83:00:04:3043:15] - [1-04:83:00:04:3043:15]
- [05: 598000] - [0-04:83:00:04:3053:15] - [1-04:83:00:04:3053:15]
- [06: 646000] - [0-04:83:00:04:3063:15] - [1-04:83:00:04:3063:15]
- [07: 695000] - [0-04:83:00:04:3073:15] - [1-04:83:00:04:3073:15]
- [08: 750000] - [0-04:83:00:04:3083:15] - [1-04:83:00:04:3083:15]
- [09: 808000] - [0-04:83:00:04:3083:15] - [1-04:83:00:04:3083:15]
- [10: 900000] - [0-04:83:00:04:3093:0c] - [1-04:83:00:04:3093:0c]
- [11: 950000] - [0-04:83:00:04:30a3:0c] - [1-04:83:00:04:30a3:0c]
[0000020d:main ] - [transform service] ready - commands [00000600:00000023]
- [status :select] - waitcmd
- [errcode :uint32] - 0x00000000
- [chip_module :select] - a3
- [hal_service :select] - transform
- [version :uint32] - 0x02050211
- [peripheral_en :flags] - 0x71000102
- [input_support :flags] - 0x00370007
- [output_support :flags] - 0x0000ff03
category [system_cmds]
- [BASE_CMD_REBOOT :00000100] : [00000002:00022efc]
- [BASE_CMD_REBOOT_SECURE :00000200] : [00000002:00022f28]
- [BASE_CMD_CALIBRATION_SAVE :20000000] : [00000004:00022f54]
```

- 若是裝置沒有問題，則是顯示到最後。

```
[00000258:main ] - [transform service] running...
category [rfmixer_r2]
- [RFMIXER_CMD_START :00001000] : [00000006:00022cb8]
- [RFMIXER_CMD_STOP :00002000] : [00000006:00022e18]
[00000272:main ] - [transform service] running...
category [transform_cmds]
- [TR_START :00000001] : [00000002:00021288]
- [TR_START_SINE :00000004] : [00000002:00021658]
- [TR_START_TEST :00000008] : [00000002:00021790]
- [TR_STOP :00000002] : [00000004:00021930]
[0000029a:main ] - [transform service] running...
```


- 裝置啟動廣播後可以在紅框框看到 Modulation 的設定值是否有成功。

```
[0000029a:main ] - [transform service] running...
hal raise : [00000600:00001000:12:transform service]
current r2 rule item - [473000] [495000:0015:3043]
r2 calibration enable [calibration]
start r2 mixer : [473000:0:00000001] - [I-04:Q-83:IMG-00:PHASE-04:PA-3043:GPIO-15]
hal raise : [00000600:00000001:0:transform service]
[001de3af:service ] - transform start : [broadcast]
transform broadcast -
- broadcast source :
- [mode :select] - passthrough
- [usb_flags :flags ] - 0x00000000
- [pcrmode :select] - disable
- filter
ts stream take all
- broadcast modulator : isdb_t
- [type :select] - isdb_t
- [bandwidth_symbolrate:uint32] - 6
- [ifmode :select] - iq_offset
- [iffreq_offset :uint32] - 143
- [dac_gain :uint32] - 0
- [constellation :select] - qam_64
- [fft :select] - fft_8k
- [guardinterval :select] - gi_1_16
- [coderate :select] - cr_5_6
- [timeinterleaved :select] - mode_3
- [isdb_t flags :flags ] - 0x00000000
mux payload enable : ["rawtable"]
no register table
dac_gain - [0x02025555]
scl_0 - [0x003aa299]
scl_1 - [0x000cc0d2]
scl_2 - [0x00000377]
fft_8k - [0x00000630]
output config - [isdb_t:usb:0]
pid filter - disable
stream config - [usb:1048576:0]
muxer slice config - [15]
service start - [0]
```

- 裝置進行中會顯示此 Log (若發生非正常顯示的 Log，請與我們聯繫，協助找出原因)。

```
[001de4b8:service ] - transform start finish : [0]
transform status changed - [wait_source]
transform status changed - [broadcast]
source usb prepare - [passthrough:1394]
source usb start - [794]
```


6 系統除錯功能

使用裝置時若是跳出錯誤代碼，主要參考下列所示。

錯誤代碼	原因
-1	不知名錯誤 (unknown fail)
-2	功能不支援 (function not supported)
-3	參數設定錯誤 (parameter set not supported)
-4	緩衝區溢位 (buffer limited overflow)
-5	裝置狀態錯誤 (can not call at this device status)
-6	硬體溝通介面錯誤 (send command to device fail or call system api fail)
-7	等待操作超時 (wait operation timeout)
-8	系統忙碌 (system is busy)
-9	找不到裝置 (device not exists)
-10	格式錯誤 (format not current)
-11	記憶體位置配置錯誤 (memory alloc fail or overflow)