VATek
Software Development Guideline

Release Date: March 2022

Content

1 Description

Vision Advance Technology has a full range of digital TV modulation chips. There are mainly two product series, one is the single chip of multimedia video encoding and digital TV modulation, another is a simple digital TV modulation chip which is our A Series Chip. Based on different development needs of products and applications, VATek provides software development kit (SDK) and Microcontroller Development Kit (MDK) to assist developers.

2 Chip Introduction

Two Series developed by VATek are related to digital TV. The A Series (Modulator) and B Series (Enmoder) are described below:

2.1 A Series

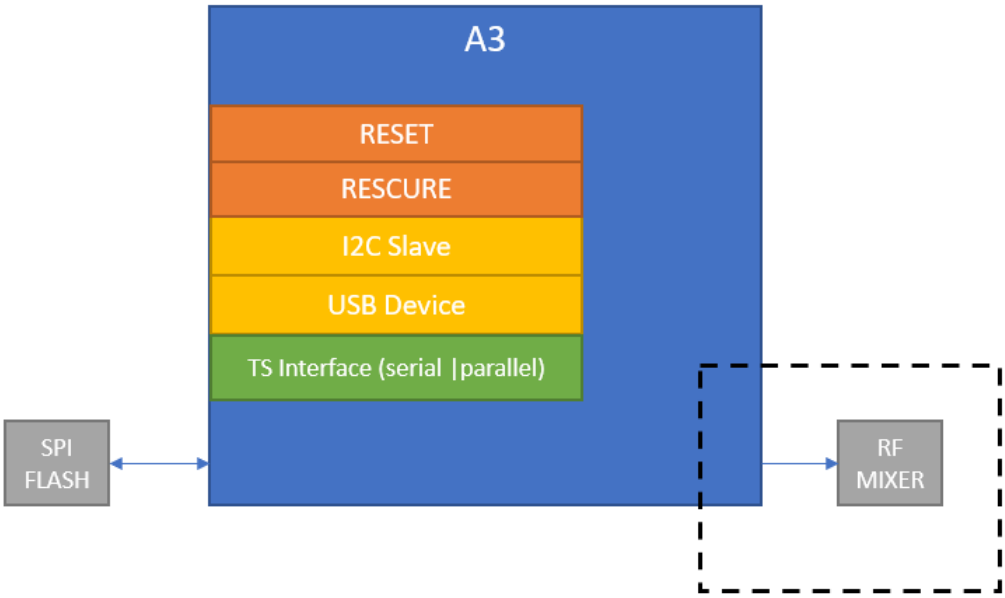
2.1.1 A Series System Structure

A Series Chip is a digital TV modulation chip, the MPEG2-TS data can be input through external hardware interface. It can receive MPEG2-TS data, reproduces the received signal then provides the function of adding PSI table and PCR correction. Moreover, it converts the stream into different modulation according to the demand, then outputs MPEG2-TS signal for broadcasting. The system structure is mainly divided into three parts:

- **System IO** :
  - RESET signal - The system reset signal can be reset by external hardware (refer to the boot process).
  - RESCUE signal - Rescue function, enter the rescue signal when the

firmware is damaged (see the boot process).

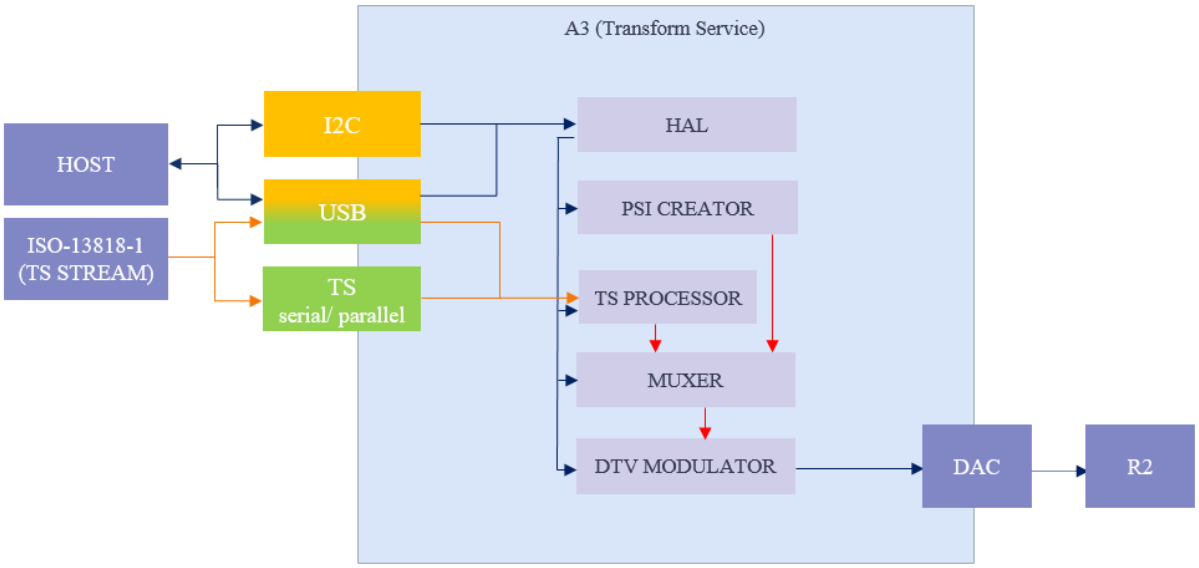
- **Control Interface** :
  - Support MDK combined with external MCU through I2C.
  - Use the USB interface provided by SDK for chip function setting and control.
- **Input Interface** :
  - TS Interface - Support Serial and Parallel mode.
  - USB Bulk Interface - In addition to control and setting, USB supports multimedia streaming through Bulk Endpoint, such as Aux Stream function.



The external hardware also includes A Series can fully provide relevant firmware (Service) load in functions only by relying on the firmware (Service) and be loaded by the SPI interface, please refer to “Boot Operation” section. The second part is the selection function. While using complete development schametic design and RF MIXER, the corresponding RF MIXER needs to be connected through the same interface to facilitate the firmware (Service) to provide RF related capabilities.

In addition to the external interface, the chip is composed of several functional units, which can be mainly divided into:

- **HAL**：The external control unit supports the implementation of the external protocol between I2C and USB to control and coordinate internal functions.
- **PSI CREATOR**：Generate PSI tables required by digital TV.
- **TS PROCESSOR**：The function of processing input multimedia streams.
- **MUXER**：Combine TS PROCESSOR and PSI CREATOR with PCR to generate MPEG2-TS meeting the needs of modulation unit.
- **DTV Modulator**：Multi format digital TV modulation unit.
- **RF Mixer**：Control unified unit and the default supported RF mixer (optional).



A Series System Structure Diagram

➢ I2C Protocol

Supports up to 400 Kb/s transmission speed, can write and read HAL register through the I2C Slave interface.

➢ USB Protocol

Write and read HAL register through the standard EP0(end point) Setup Packet through the USB device interface.

➢ HAL Control Unit

Support the operation of internal unit functions through USB and I2C interface. The internal unit is implemented in the form of HAL register according to the functional and requirements. HAL register takes 32 bits as the unit, the address represents a storage of 32 bits. The register can include the following four forms:

Register Catagories	Description
NORMAL	Provides the setting of internal unit properties,providing reading and writing.
COMMAND	Write properties or set and execute the specific functions.
STATUS	Provide information related to system status or execution results and provide read operations.
BUFFER	Fixed format and fixed length block characteristic, providing reading and writing.
PLAYLOAD	Block characteristic with variable format and length, providing reading and writing.

➢ TS PROCESSOR Unit

Since the setting of different DTV modulation specifications and parameters will require MPEG2-TS with different bitrates and requirements, a new MPEG2-TS will be generated according to the output requirements. The core has the following functions:

- **TS DEMUX**：Can analyze the content structure and simple messages of the MPEG2-TS, so as to set filtering or develop PSI table during product setting.
- **REMUX**：Re-multiplexes the effective input combined with PSI table into the actual output MPEG2-TS according to the back-end modulation requirements (bitrate).

➢ MUXER Unit

Based on different modulation modes and source conditions, such as bitrate, multimedia stream time, etc. Need to be reordered by MUXER to meet the relevant specifications of digital TV broadcasting. MUXER unit rearranges MPEG2-TS according to output requirements and input conditions.

- **PADDING**：When the data cannot be met, null packet will be used for filling. MUXER provides self-defined and standard functions.
- **PCR INSERT**：The insertion function of PCR can add independent PID PCR, and the Interval can be controlled to meet the application requirements.
- **PCR REPLACE**：It provides the function of PCR replication, which may change the original PCR accuracy during the conversion of different input source (bitrates). This problem can be corrected to varying degrees through the replication function.

➢ MODULATOR Unit

Supports global digital TV modulation standards, including DVB-T2, DVB-T, DVB-C (J83a), ATSC, J83b, DTMB, ISDB-T, J83c. The modulation types supported by different chips are as follows:

CHIP No.	Supported format
A3	DVB-T2、DVB-T、DVB-C (J83a)、ATSC、J83b、DTMB、ISDB-T、J83c

2.1.2 System Function Operation

➢ MPEG2-TS Composition：

MPEG2-TS is the format of digital TV broadcasting. The composition of the complete TS mainly includes the following three elements.

1. **PES**：Package of video, audio signals and other signals such as CC subtitles.

2. **PCR**: You can refer to ISO-13818 specification. When the back-end outputs to DTV, DEMOD needs to refer to the time, which mainly determines the packet broadcast order. The PCR accuracy will also affect broadcast quality.
3. **PSI** : The standard needs to be followed in different countries. Different countries have different PSI tables, such as PSIP in the United States, ARIB in Japan, etc.

- The chip receives MPEG2-TS signals according to the front-end input interface (USB or TS interface).

Input Interface	Description
USB interface	The USB interface can control MPEG2-TS input speed, the speed of USB 2.0 can be up to 480Mbps.
TS interface	It must be continuous data. The chip will show error if MPEG2-TS is interrupted in the middle.

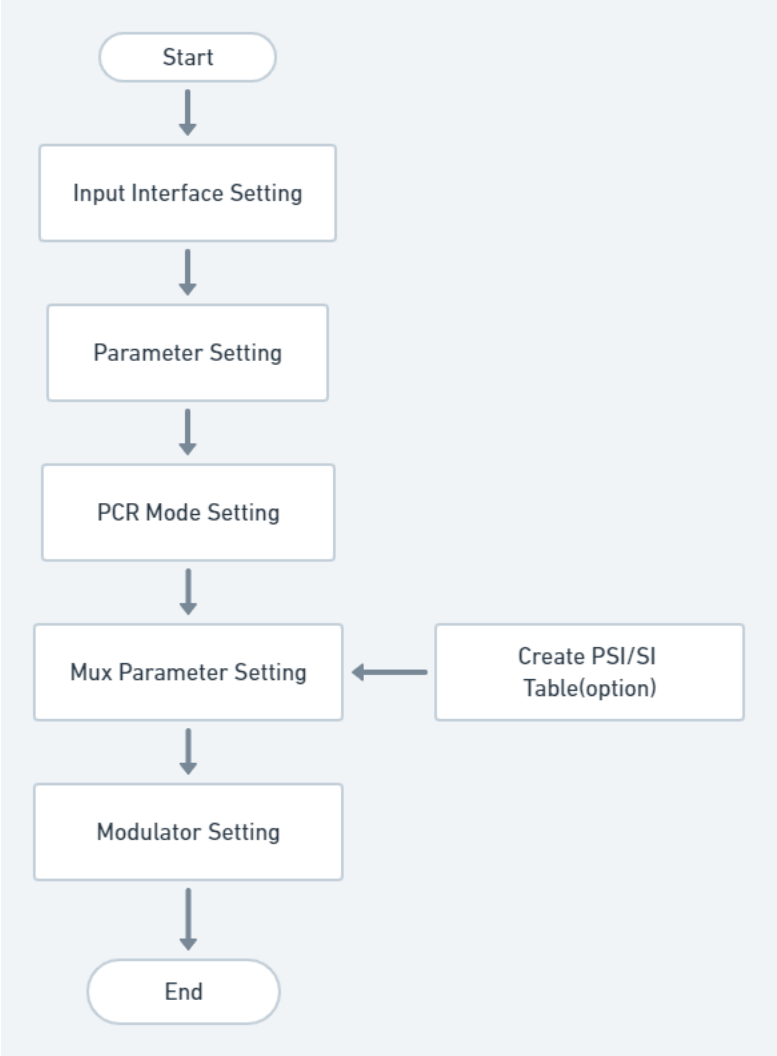
- The chip can use PASSTHROUGH and REMUX mode according to the TS integrity of the front-end input.

Mode	Applicable Situation
PASSTHROUGH	The front-end input MPEG2-TS has complete content, and only needs to convert the signal into digital TV signal broadcasting.
REMUX	1. The PCR of front-end input MPEG2-TS needs to be corrected. 2. When the data cannot meet the bitrate, NULL packet will be used for filling.

- A3 can perform correction function for PCR when Remux mode is processing, please refer to the following chart:

PCR mode Function	Applicable Situation
RETAG	To improve the accuracy of PCR output, this mode can be used. The principle is to generate PCR with higher accuracy based on the PCR input from front-end MPEG2-TS and provide PCR reference for back-end MPEG2-TS output. It is usually used in the more rigorous specification of ISDB-T. (Limitation: PCR PID and video PID cannot be the same.)
ADJUST	PCR will be corrected.
DISABLE	PCR will not be corrected.

2.1.3 A Series Chip System Flow



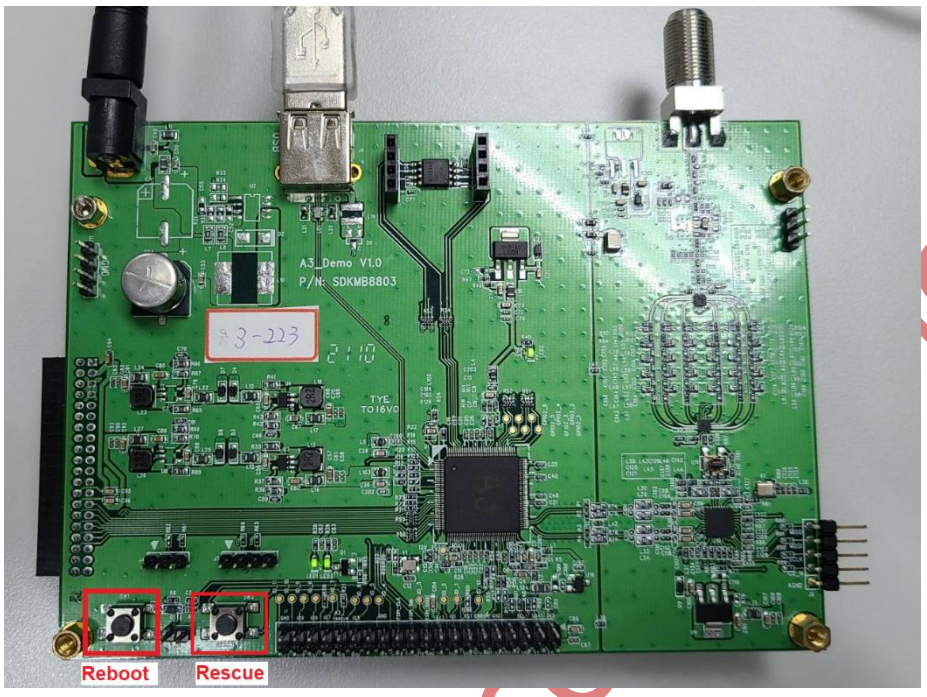
2.1.4 Boot Operation

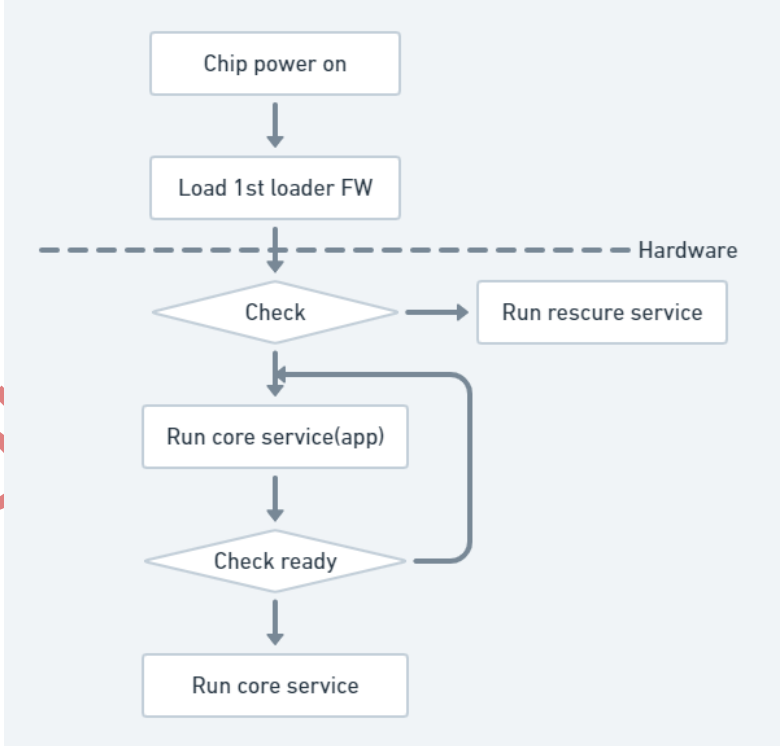
The system is mainly composed of Loader (boot application) and Service. When the chip completes the boot program, it will automatically load and run through the SPI interface. After running, it will detect the status of the RESCUE signal and decide whether to enforce the RESCUE service. RESCUE mainly provides the function of updating firmware. If it does not enter the RESCUE service, the firmware will be checked, and if it is valid, the Service will load and run.

By external I2C and USB, the chip functions are controlled through the HAL control unit, these two path controllable stages will be different. Since the USB interface is

provided with the basic USB service by Service, it is impossible to communicate with the chip before entering the RESCURE and Service, while I2C can read the status and control operation immediately after the system is booted.

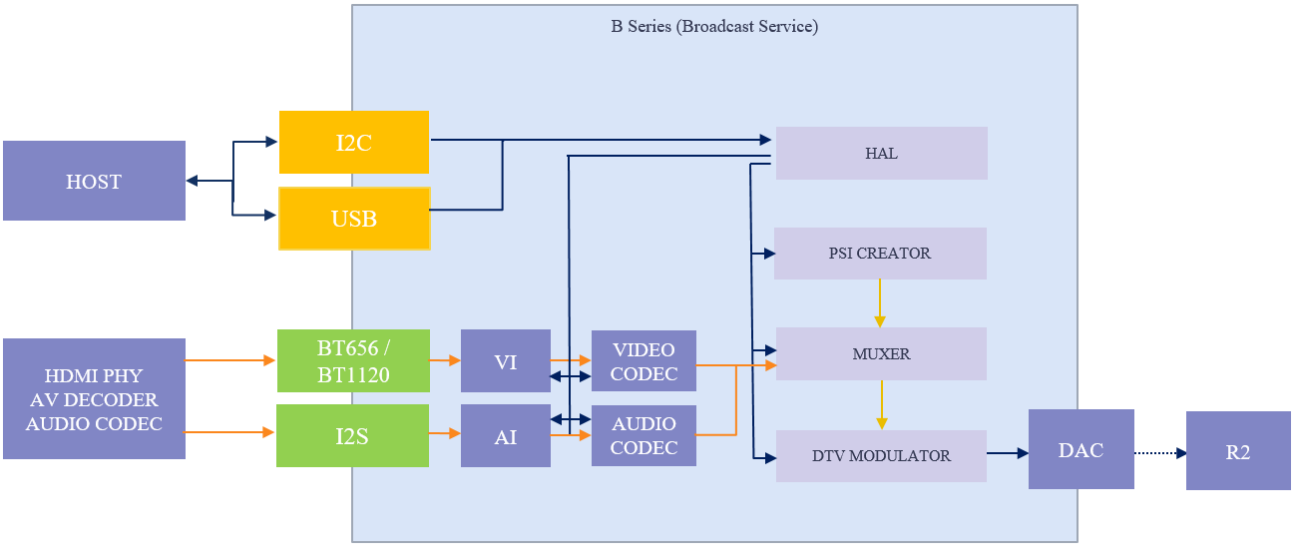
- If the device does not respond, please restart it.





2.2 B Series

B Series Chip are the chips that integrate video and audio signal encoding and DTV modulation. It receives video and audio signals through PHY for compression encoding processing, add PSI table then convert them into various modulation system TV signal broadcasting through modulation unit. B Series Chip supports video signal and I2S audio signal formats of BT656 and BT1120, it uses YUV420 color format for video coding and compression. The maximum input supports is 1080P60.



B Series System Structure Diagram

- **HAL :** The external control unit supports the implementation of the external protocol between I2C and USB to control and coordinate the internal function.
- **VI :** Video input unit.
- **AI :** Audio input unit.
- **AUDIO CODEC :** Audio encoding unit.
- **VIDEO CODEC :** Video encoding unit.
- **PSI CREATOR :** PSI table generation unit required by DTV.
- **MUXER :** Combine the information of encoder and PSI CREATOR unit to regenerate DTV stream.

- **MODULATOR :** DTV modulator, multi-format DTV modulation unit.
- **RF MIXER :** Controls the default supported RF MIXER (optional.)

2.2.1 B Series System Structure

➢ HAL Control Unit

Support the operation of internal unit functions through USB and I2C interface. The internal unit is implemented in the form of HAL register according to the functional and requirements. HAL register takes 32 bits as the unit, the address represents a storage of 32 bits. The register can include the following four forms:

Register Catagories	Description
NORMAL	Provides the setting of internal unit properties,providing reading and writing.
COMMAND	Write properties or set and execute the specific functions.
STATUS	Provide information related to system status or execution results and provide read operations.
BUFFER	Fixed format and fixed length block characteristic, providing reading and writing.
PLAYLOAD	Block characteristic with variable format and length, providing reading and writing.

➢ VI/AI Input Unit

The video signal input unit supported by B Series Chip provides an image source up to 1080P60. In addition to the actual image input, the internal source of the chip can also be used as the input source. The input unit also provides the setting of input source:

- **Internal Source**
  1 **COLORBAR** : Can generate test image conforming SMPTE RP 219:2002(ARIB STD-B28.)
  2 **BOOTLOGO** : Allows to add multiple custom graphics as image resources.
- **VI_FLAG** （**HALREG_VI_0_FLAGS**）:
  1 **VI_BUSWIDTH_16**：If HDMI input is used, it needs to be turned on the meet the HDMI input specification.
  2 **VI_SEPARATED_SYNC**：If HDMI input is used, it needs to be turned on the meet the HDMI input specification.
  3 **VI_EXT_HALF_FPS**：The frame rate of the input signal can be halved（1080P60->1080P30, 720P60->720P30, etc.）
- The audio signal input unit supported bt the VI unit supports 32K, 44.1K and 48K sample rate, and can support up to 2 channel 24 bits PCM.
- The resolution format of VI unit support input complies with CEA-861 specification. The following table provides the resolution format of B Series Chip.

Resolution \ 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					

➢ Video Audio Encoder Format

B Series supports video and audio compression. It supports different video compression according to different chip models. The support formats are as follows:

Encoder Format \CHIP Model		B2	B2+	B3	B3+
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 ENCODER	H.264			V	V
	MPEG2	V	V		V

- **MPEG 2 Encoder Specification**：
  1. Maximum resolution1080p 30 FPS
  2. HIGH PROFILE (I, P Frame Only)
  3. HIGH LEVEL maximum Bitrate 30 Mbps (VBR)
  4. YCrCb 4:2:0
- **H264 AVC Encoder Specification**：
  1. Maximum resolution 1080p 30 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** : Input Progressive video format and enable this function to output 1080i 60（1080P60->1080I60...）。

  2 **ENC_EN_DISABLE_DEINTERLACED**：In B3+ Chip, if the interlaced video format is used for input, enable this function to convert the interlaced format to progressive format（1080I60->1080P60, etc）.

  3 **ENC_EN_DISABLE_ADTS_CRC:** In ADTS audio encoder, using this function will skip the check and output audio information.

  4 **ENC_EN_DISABLE_LATENCY_Q** : When adjusting the video quality, latency and Q value will affect each other. Enable this function to turn off the interaction between them.

➢ PSI CREATOR Unit

DTV includes audio and video data streams. According to different countries and standards, PSI table needs to be added to identify channels and multimedia content. This unit provides two different ways to assist developers to complete the required PSI table:

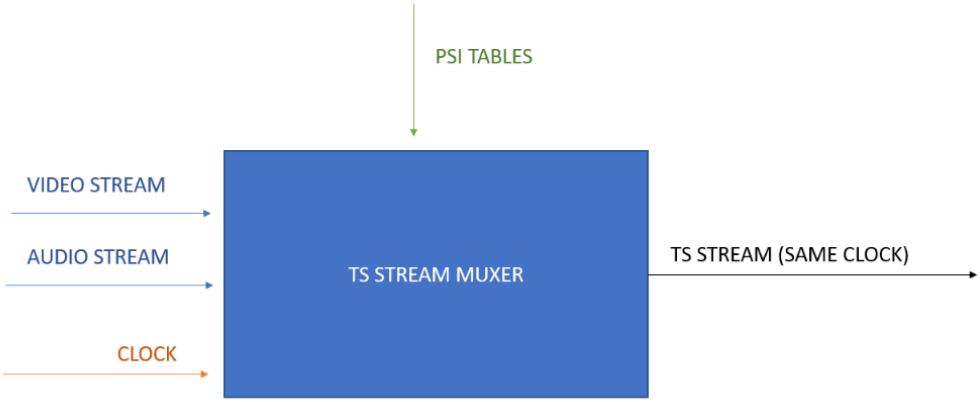
- **PURE TABLE** : Developer refers to the required specifications according to the application situation and adds the self-defined PSI table.
- **DEFAULT** : The basic PSI table defined for different countries and specifications can be used through parameter settings.

➢ MUXER Unit

Based on different modulation modes and source conditions, such as bitrate, multimedia stream time, etc. Need to be reordered by MUXER to meet the relevant specifications of DTV broadcasting. MUXER unit rearranges

MPEG2-TS according to output requirements and input conditions, mainly combining the encoded data of video encoder and PSI table generated by PSI CREATOR.

- **PADDING**：When the data cannot be met, null packet will be used for filling. MUXER provides self-defined and standard functions.
- **PCR INSERT**：The insertion function of PCR can add independent PID PCR, and the Interval can be controlled to meet the application requirements.
- **PCR REPLACE :** It provides the function of PCR replication, which may change the original PCR accuracy during the conversion of different input source (bitrates). This problem can be corrected to varying degrees through the replication function.



➢ MODULATOR Unit

Supports global digital TV modulation standards, including DVB-T2, DVB-T, DVB-C (J83a) , ATSC, j83b, DTMB, ISDB-T, J83c. The modulation types supported by different chips are as follows:

CHIP No.	Supported format
B2	DVB-T, DVB-C (J83a), ATSC, J83b, DTMB, ISDB-T, J83c

B3	DVB-T, DVB-C (J83a), ATSC, J83b, DTMB, ISDB-T, J83c
B3+	DVB-T2, DVB-T, DVB-C (J83a), ATSC, J83b, DTMB, ISDB-T, J83c

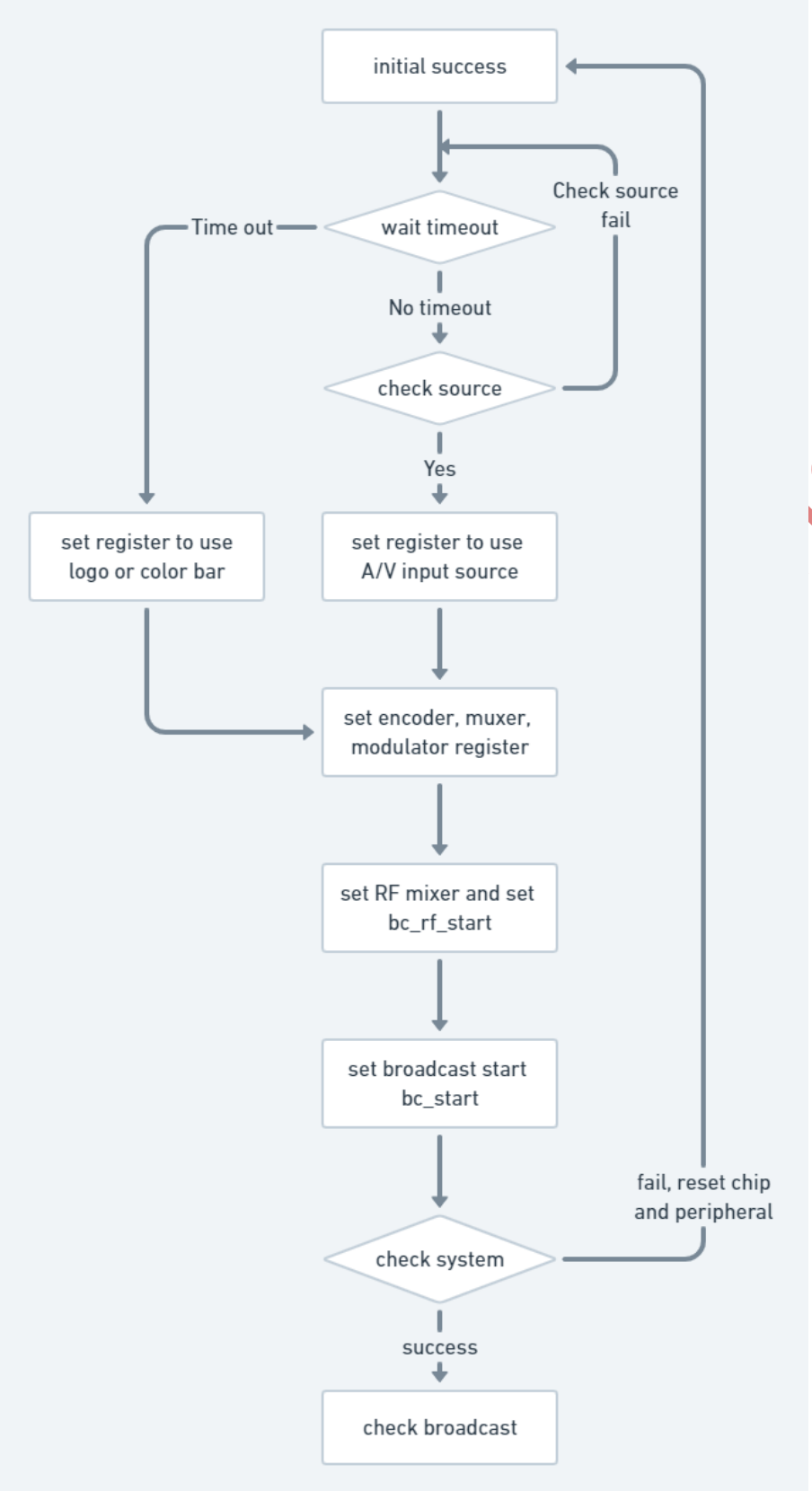
2.2.2 System Function Operation

After the Chip is boot, it can read command of the register through I2C. It is necessary to confirm if the system status can correctly read the system status register 0x20, the system status can be roughly divided into the following three types:

1. Idle Status: When the chip is boot and the firmware is loaded correctly, the chip will return idle status, indicating that the chip can perform relevant functions and operations in the idle phase.
2. Operation status: When the chip returns the running status, it indicates that the system is in operation. At this time, the chip can only execute the relevant instructions in the operation stage.
3. Error Status: The system status register must be based on 0xFF000000. If not, it means that the firmware is not loaded correctly. After the chip is boot, ensure that the firmware operate correctly.

2.2.3 B Series Chip System Operation

If needed to broadcast, you need to confirm if the initialization is completed.You can read register (0x20) to confirm the chip is in the idle status or not. After confirming that the chip status is correct, you can start reading and writing instructions to the chip. The user can decide to use the actually input source VI or built in colorbar for DTV broadcasting. The broadcasting process is as follows:

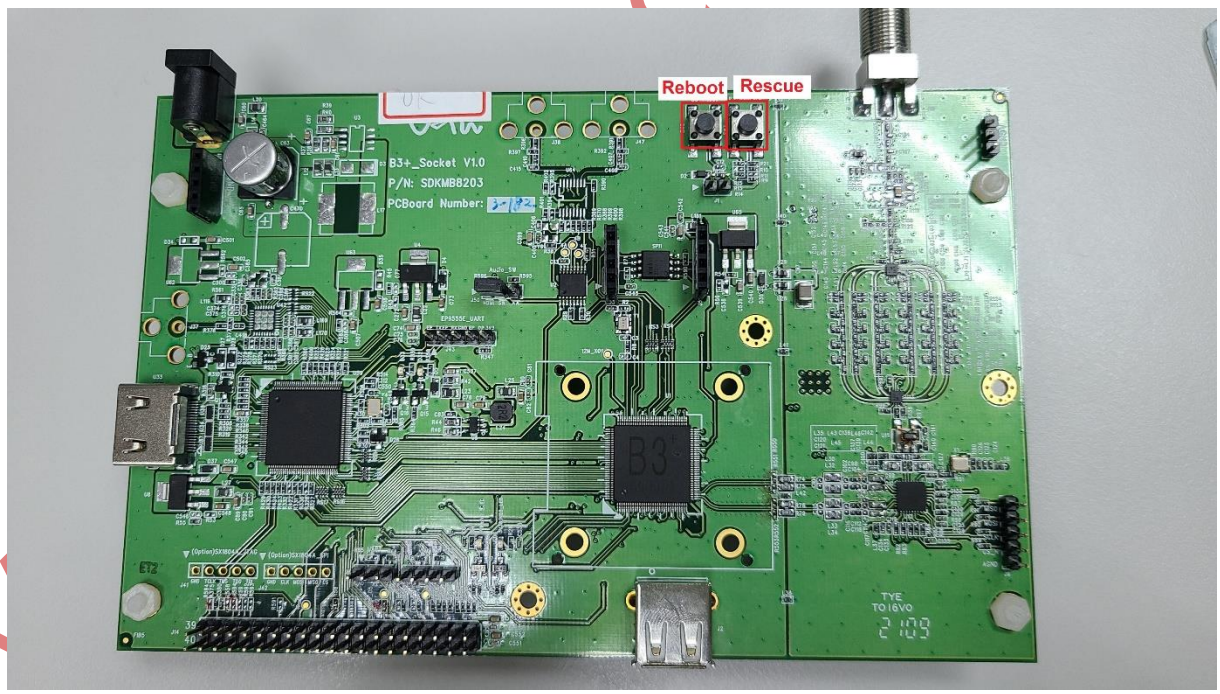


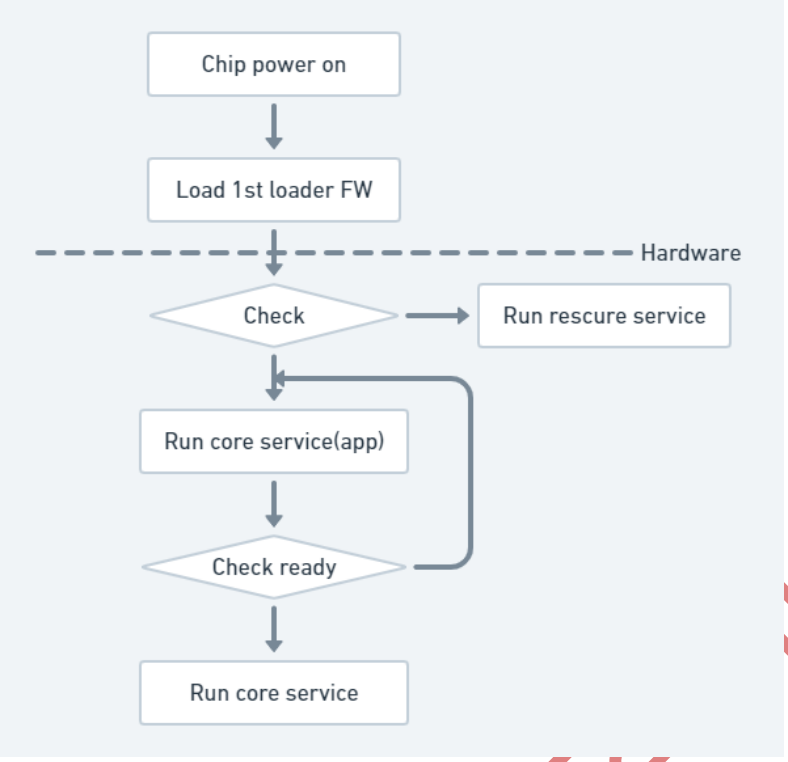
2.2.4 Boot Operation

The system is mainly composed of Loader (boot application) and Service. When the chip completes the boot program, it will automatically load and run through the SPI interface. After running, it will detect the status of the RESCUE signal and decide whether to enforce the RESCUE service. RESCUE mainly provides the function of updating firmware. If it does not enter the RESCUE service, the firmware will be checked, and if it is valid, the Service will load and run.

By external I2C and USB, the chip functions are controlled through the HAL control unit, these two path controllable stages will be different. Since the USB interface is provided with the basic USB service by Service, it is impossible to communicate with the chip before entering the RESCURE and Service, while I2C can read the status and control operation immediately after the system is boot.

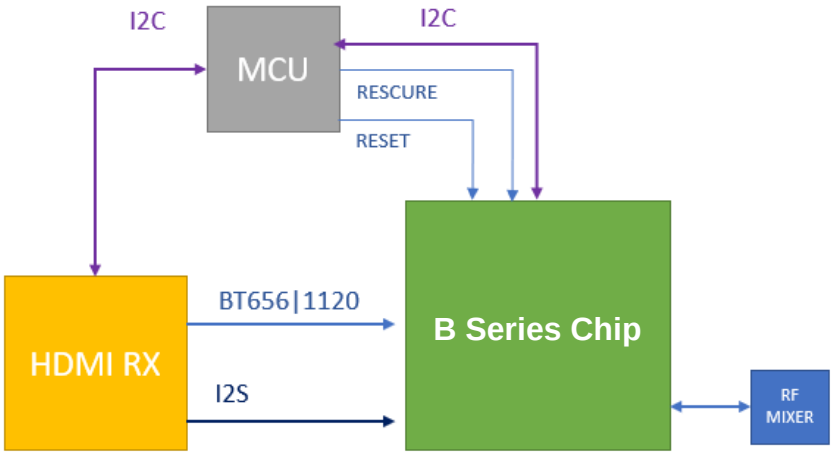
- The figure below shows the B Series device. If the device has no response, please restart it.





2.2.5 Product Design and Application

In terms of application, B Series is mainly used to convert external video into DTV signals through internal encoding unit and modulation unit, so it must be combined with surrounding video and audio sources, and can integrate HDMI RX, SDI RX, AV Decoder, etc. The product structure is basically shown in the figure:



At least one set of I2C is required for MCU selection to control the system, and required control interface is provided according to the selected video and audio source. The video and audio source needs to obtain the video source through the communication interface, including resolution, frame rate, aspect rate and output pixel clock,etc. The audio source needs to obtain sample rate, channel, etc. The system flow design takes MCU as the main subject to distinguish boot system initialization, video source detection and broadcast control.

- **Boot Initializaation**

  After the product is booted, the MCU starts to operate according to the initialization status of the layout design. In addition to the initialization of the peripheral functions of the MCU itself, it also initializes the periphery, such as hardware resetting the B Series chip, resetting the periphery of the video and audio source, and displaying the operation interface. Initialize the MCU control interface (I2C, SPI, etc), ensure that the video and audio source is in the correct status according to the selected periphery, and ensure that the system is in the idle status, so as to complete the initialization of the product.

3 Software Development Kit (SDK)

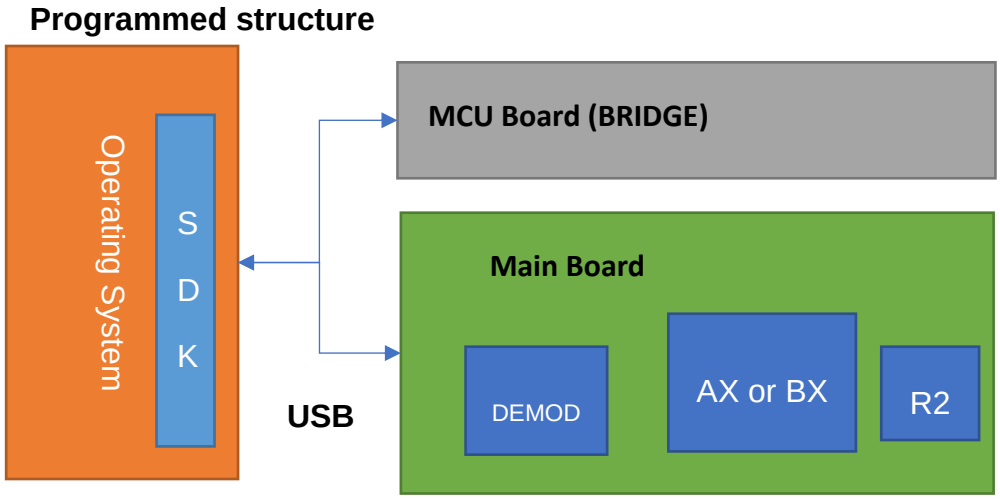
The SDK is developed for applications and products running on high-level operating systems (Windows, Linux, and Android.) It is connected with the high-level operating system through the USB receiver of VATek modulation chip. It can be developed through the SDK, including test tools and update tools, etc. The full range of application tools provided by VATek are developed based on this SDK.

3.1 Description

The SDK is provided in the form of source code. Using CMake as the build tool, developers can build and develop relevant applications in Microsoft Windows 10 and later, Ubuntu 18.04 (or other systems supporting relevant compiler environment). The main software package includes:

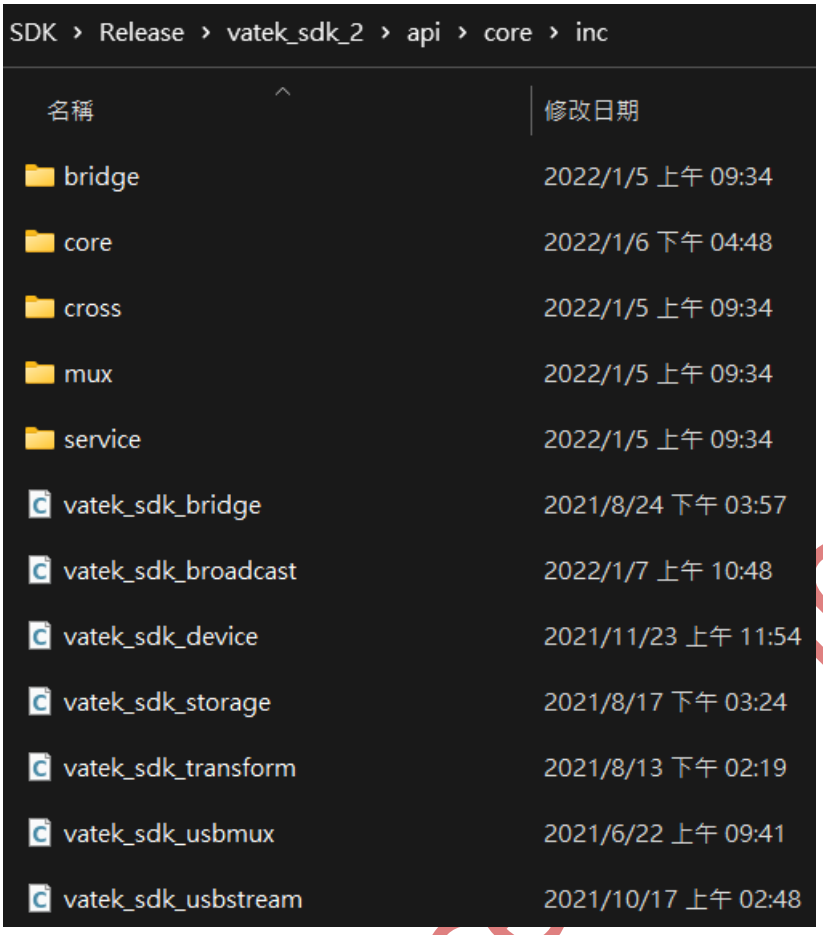
Folder		Description
api	core	Core program development interface
	qt	QT based GUI component interface
app	vatek_factory	Firmware image file tool
	vatek_romtool	Firmware update tool
	vatek_toolkit	Common broadcast tool
build		Build related tools
extend	bridge_2	MCU Board firmware source code
	obs-vatek	The example of OBS integrates A Series
	tsduck-vatek	The example of TSDUCK integrates A Series
sample	app_bridge	The example of programmed control

		board
	app_romtool	The example of firmware update
	app_stream	The example of control A Series
	common	Common source code



3.2 Core Program Development Interface

The SDK mainly provides related functions through the core program development interface. The program related source code is in api\core in the SDK. The main functions are provided by the program interface under the root directory, which is mainly divided into the following function categories.



Program Interface Definition	Description
vatek_sdk_bridge	The relevant control and operation of bridge (MCU board) can control and set the video and audio source on the demo board.
vatek_sdk_device	Enumeration, control of connected modulation devices.
vatek_sdk_storgae	Device resource and firmware related to program interface.
vatek_sdk_transform	Main control program interface of A series chip.
vatek_sdk_usbmux	A program development interface of integrated operating system video encoder.
vatek_sdk_usbstream	A program interface for MPEG-TS.

Program Interface can be mainly distinguished as three categories: Common, Broadcast, and Transform, etc.

3.2.1 Common API

- Basic operation functions of the e system, such as listing relevant devices supported in the system, device startup, shutdown and reboot, and Calibration value adjustment, mainly provide relevant function through vatek_sdk_device.

  - Device Enumeration (Decide which Bus and Chip to list the devices) :

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

  - Get enumerated device service (Read the type of the Chip by the enumeration device [A or B]) :

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

  - Clear the device enumeration list (Clear the list read by the device enumeration to ensure that the next connection will not be repeated) :

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

  - Device connection (The listed device is open) :

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

  - Device stop (Stop the RF and Chip operation) :

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

- Device close (Clear the value of the device to ensure that the next connection will not be repeated) :

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

- Close the device then reboot (Clear the value of the device and restart the device completely) :

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

- Read the Calibration value (Read the calibration value stored in the device):

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

- Apply the Calibration value (Dynamically adjust the calibration value of Chip) :

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

- Save the Calibration value (Store the calibration value of the dynamically adjusted Chip in the device [Flash]) :

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

➢ The basic composition of the device storage space is composed of 64KB loader and application. According to the required functions, add a header after the application. Each data uses the header to find the memory address (such as : Bootlogo, R2 table, Modulation config, etc.) , mainly provide relevant function through vatek_sdk_storage.

- Create the Storage handle of the reading device (Control of computer reading device Storage) :

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

- Create and read the handle of loader and application in the computer (Control the computer to read the loader and application of the computer):

```
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);
```

- Create and read the handle of v2image in the computer (Control the computer to read v2image) :

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

- Read the application of Storage (Use this function to read the information in the application) :

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

- Read the loader of Storage (Use this function to read the information in the loader) :

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

- Read the Bootlogo of Storage :

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

- The space, length, and width required to create Bootlogo :

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

- Add Bootlogo to Storage :

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

- Delete Bootlogo of Storage :

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

- Read R2 table of Storage :

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

- Read Storage, if it's Broadcast mode or not (B Series) :

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

- Read Storage, if it's Transform mode or not (A Series) :

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

- Read Config setting of Storage (This setting includes additional functions of Modulation and Chip) :

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

- Save v2image in Storage (Save Storage in PC file) :

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

- Close Storage :

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

- The file space and length required by writing device of create Storage :

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

- Write Storage into device :

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

- Clear and close the Storage file of writing into device :

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

➢ In the SDK design solution, the peripheral control of all video and audio sources needs to be controlled through MCU board (Bridge). For example, B Series needs to be provided with original video and audio data by BT1120 and I2S interface, HDMI RX chip, and A Series integrates DTV DEMOD or other MPEG-TS interfaces as video and audio data sources, which can control and operate MCU Board (Bridge)

through vatek_sdk_bridge.

- Connect Bridge (Bridge needs to be connect through I2C) :

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

- Read Bridge information :

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

- Read AV Source information :

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

- Read the name of AV Source :

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

- Start the external AV Source :

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

- Get AV Source status :

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

- Stop external AV Source :

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

◆ Bridge connect stop：

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

3.2.2 Broadcast API

➢ B Series mainly provides the original video and audio data provided by the external video and audio chip, and directly outputs the DTV broadcast signal in combination with the internal encoder and modulation function. B Series provide complete function by Broadcast Service, the high-level software development package provides vatek_sdk_broadcast software interface which allows users to efficiently operate the whole process.

◆ B Series service is enabled（including obtaining device information, judging if it is B Series, checking if RFmixer supports, and checking if auxstream supports or not）:

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

◆ Read B Series device infromation：

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

◆ B Series device broadcast start（including judging if using auxstrean or RF signal to start）:

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

- Auxstream function polling (auxstream including synchronous and asynchronous modes) :

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

- B Series device broadcast stop (including RF signal stop and device stop):

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

- Check if support auxstream or not :

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

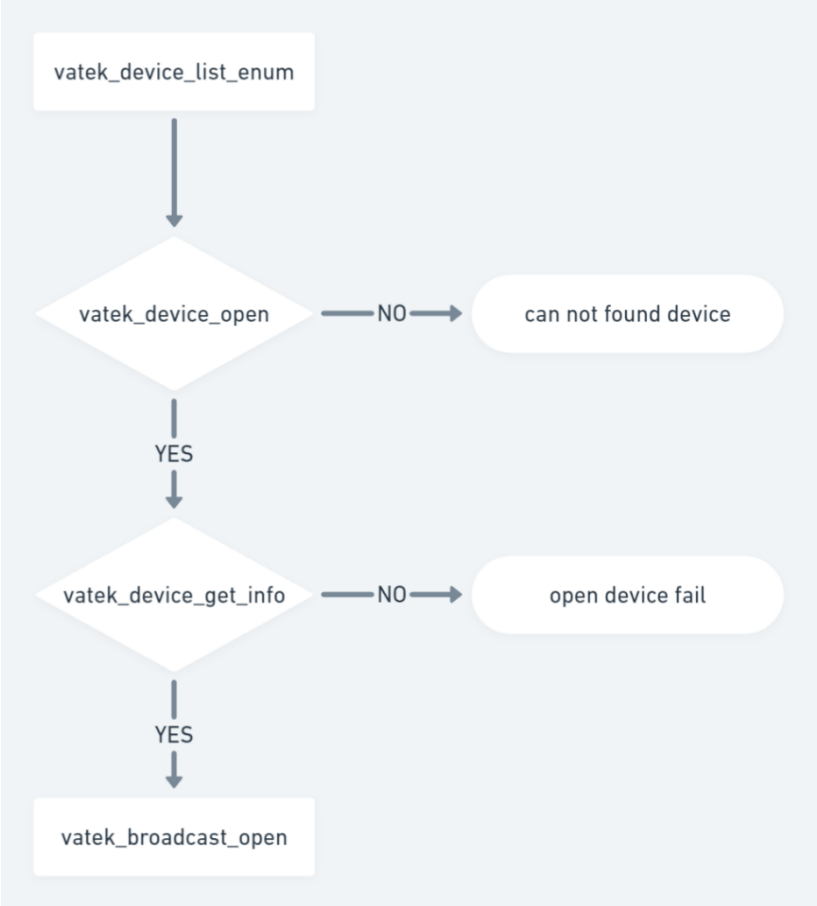
3.2.3 B Series examples：

The main usage method is app_ broadcast.c (sample \ app_broadcast) as an example, the program will demonstrate the process of B Series operation.

◆ Edit B Series encode, modulate related parameter：

```
/* broadcast parameters*/
static broadcast_param bc_param =
{
    .enc =
    {
#if 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, },
#if 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：Initialization device and boot.



- Device enumeration （This setting is to turn on the USB layout and specify the service as broadcast）:

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

- Open Device：

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

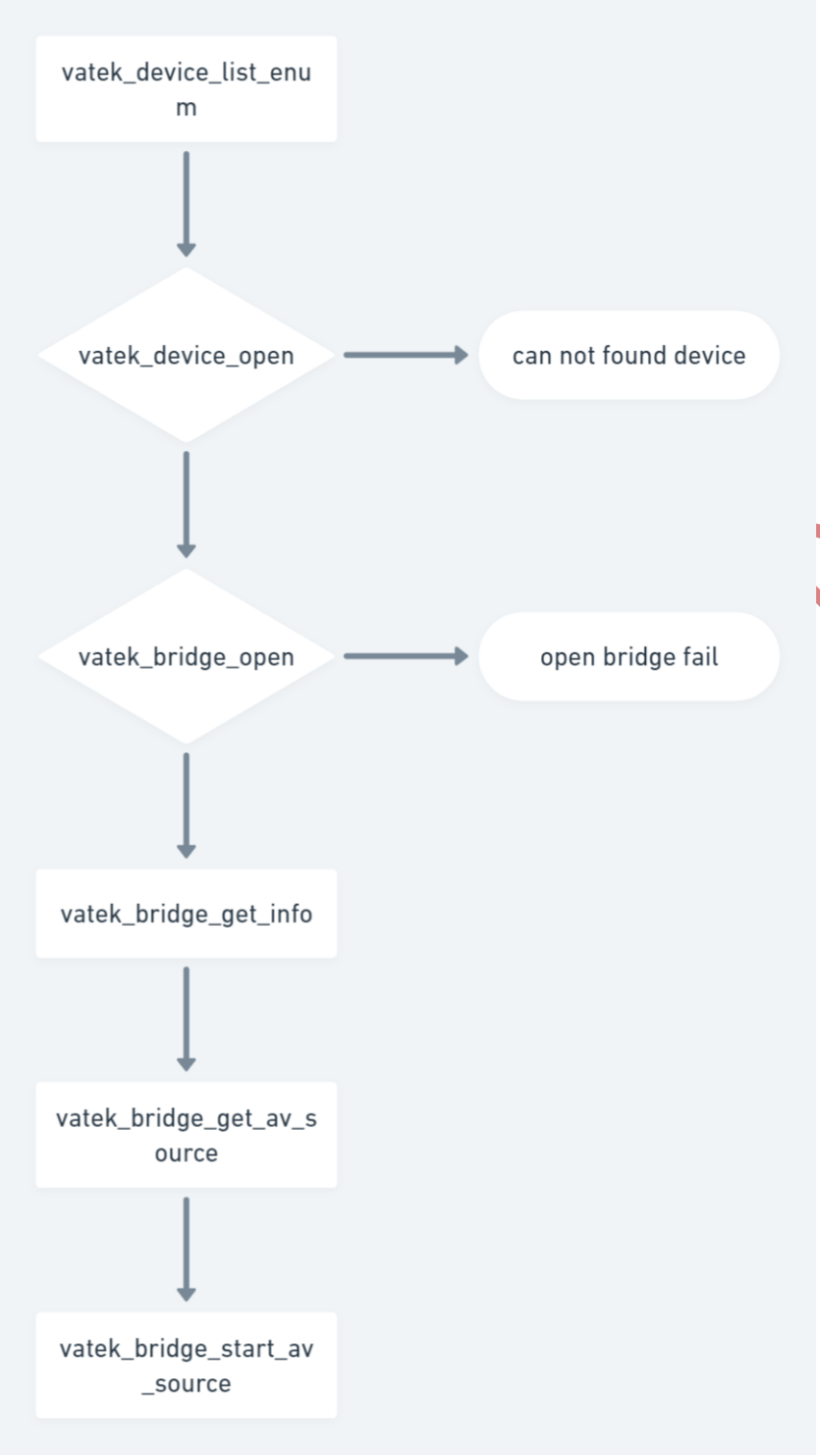
- Read device information：

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

◆ Open B Series service：

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

Step 2：Check the encoder operation mode and if the video and audio is ready or not. （If the encoder mode is Colorbar and Bootlogo, only the chip encoder setting and modulation setting are determined; otherwise, if the encoder mode is AV source, the bridge layout must be used to capture the external signal source.）



- Device enumaration (This setting is to open the bridge layout and specify the service as broadcast) :

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

- Connect device :

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

- Connect bridge :

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

- Get bridge infromation :

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

- Read external av source :

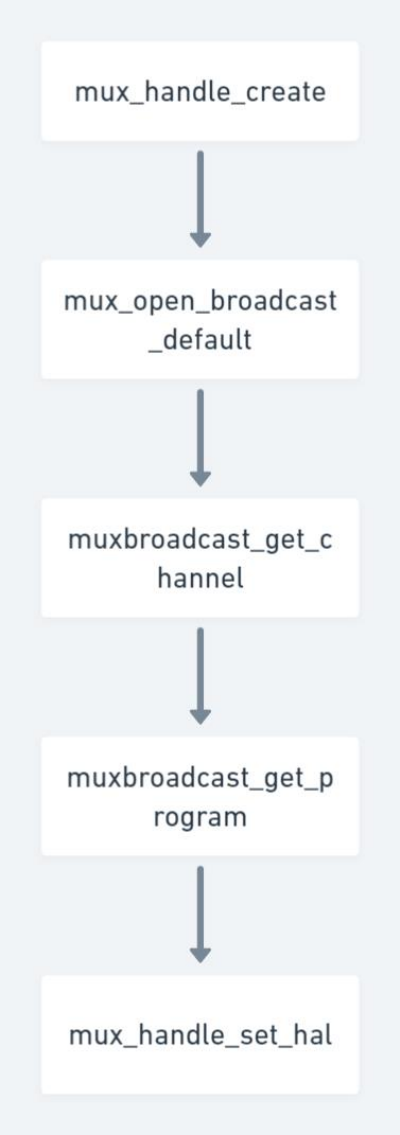
```
nres = vatek_bridge_get_av_source(hbridge, i, &bsource);
```

- Start external av source to chip :

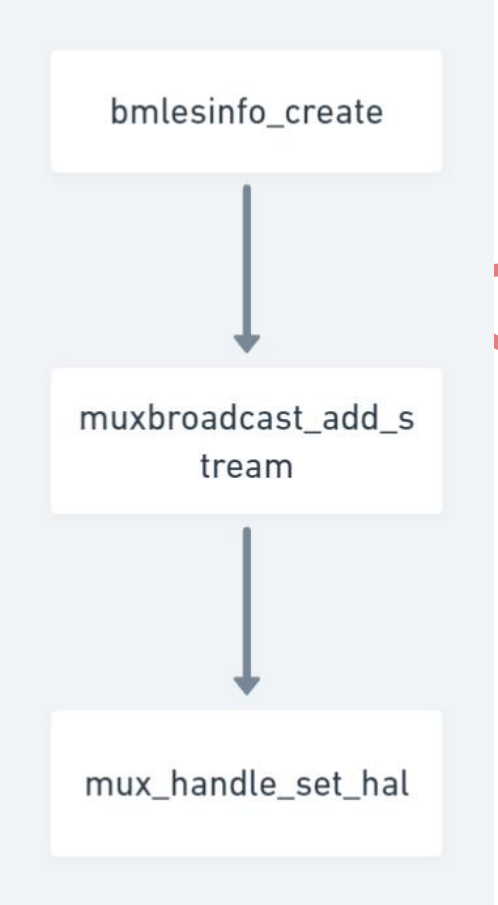
```
nres = vatek_bridge_start_av_source(hbridge, &bsource);
```

Step 3：Start PSI tablefunction and BMLfunction：

PSI table



Has open auxstream function



- Create the space and size of mux：

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

- Use the default PSI table：

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

◆ Read PSI table channel：

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

◆ Read PSI table program：

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

◆ Create BMLinformation：

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

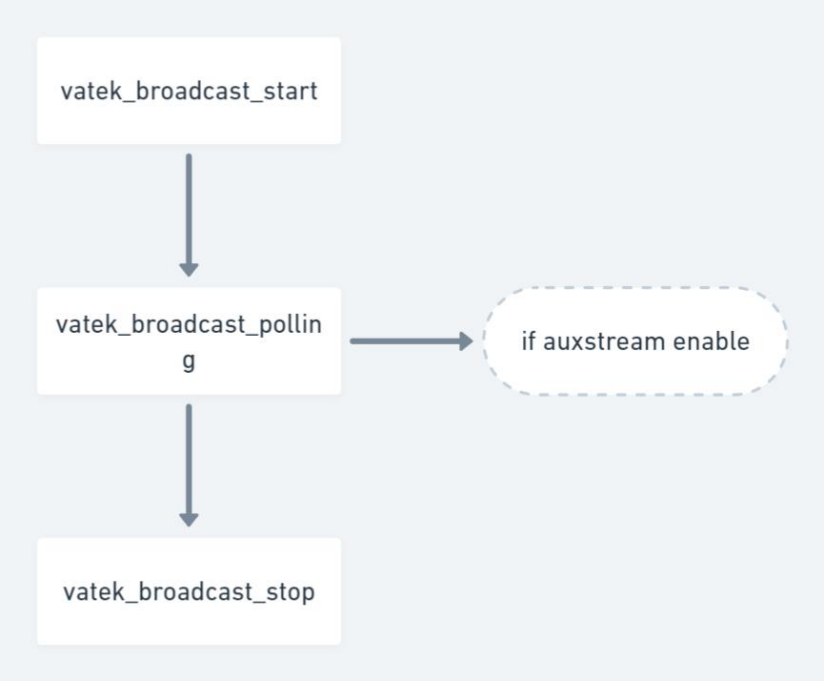
◆ BML information add to PSI table：

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

◆ PSI table write in HAL ：

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

Step 4： B Series device start the broadcast：



◆ Device start the broadcast：

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

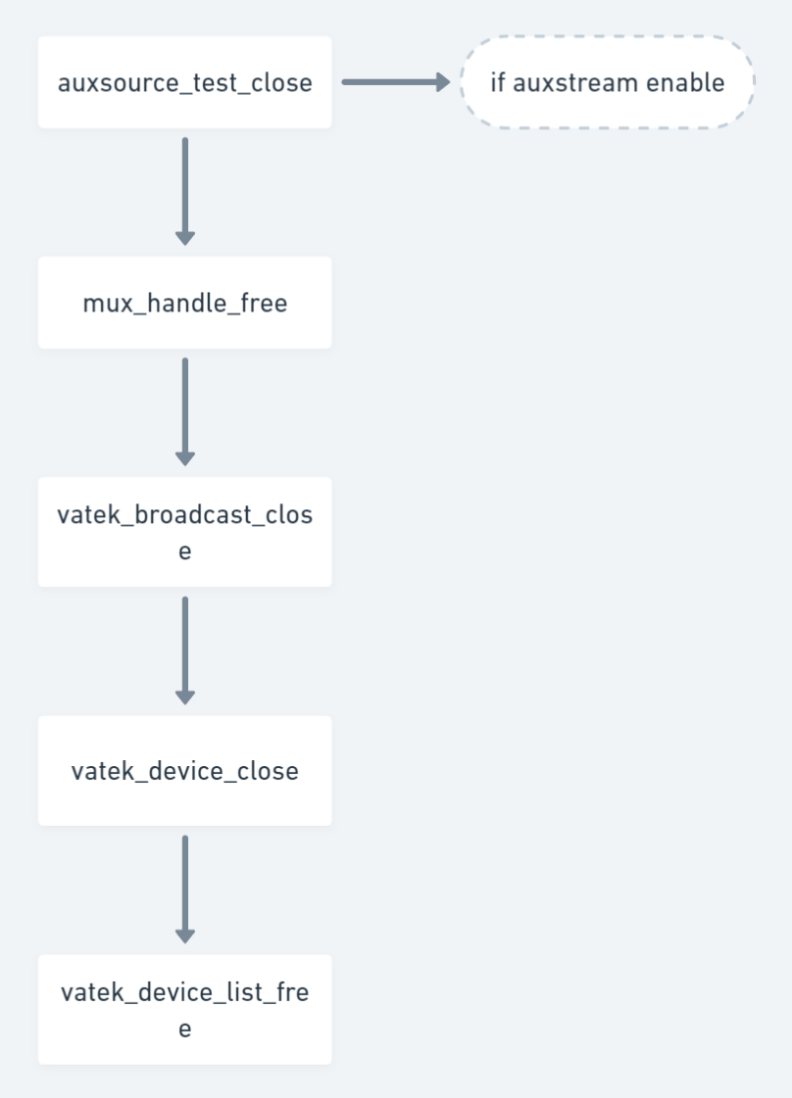
◆ Cycle judgment device status and Auxstream function transmission （auxstream includes synchronous and asynchronous modes）:

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

◆ Device stop broadcasting：

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

Step 5： Release and clear device before stopping the program source：



- Auxsource function close (if it is open) :

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

- PSI table information free :

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

- Device stop broadcast (Stop broadcast and free the broadcast data) :

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

- Device close (free the device) :

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

- Free the device enumaration :

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

3.2.4 Transform Category

➢ A Series mainly provides DTV modulation, reproduces the received signal, provides the function of adding PSI table and PCR correction, and converts it into different modulation and output MPEG2-TS signal for broadcast according to the demand. A Series of complete function is provided by Transform service, and the high-level software development package provides vatek_sdk_transform software interface which allows users to efficiently operate the whole process.

- A Series Service Transform open (It includes obtaining device information, checking if it is A Series, checking if rfmixer supports and establishing PSI table space) :

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

- Enumarate stream content :

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

- Start broadcast system of A Series :

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

- Check the status of A Series device polling :

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

- Read A Series device information (including current bitrate, data bitrate, and transform mode, etc.) :

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

- Read A Series packets :

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

- Clear A Series packets :

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

- Stop A Series device broadcast(including RF signal stop, and device stop.):

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

- A Series device stop broadcasting (stop broadcasting and release broadcast data) :

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

- A Series can use file, UDP and RTP to input TS data stream, so that A Series device can use MPEG2-TS from different sources. High-level software development kit provides cross / cross_stream program interface allows you to quickly operate the whole process.
  - Transmission through MPEG2-TS file :

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

  - Transmission through UDP file data stream :

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

  - Null packet testing mode :

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

  - Transfer stream :

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

  - Get stream slice :

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

- Stream stop :

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

- Stream close :

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

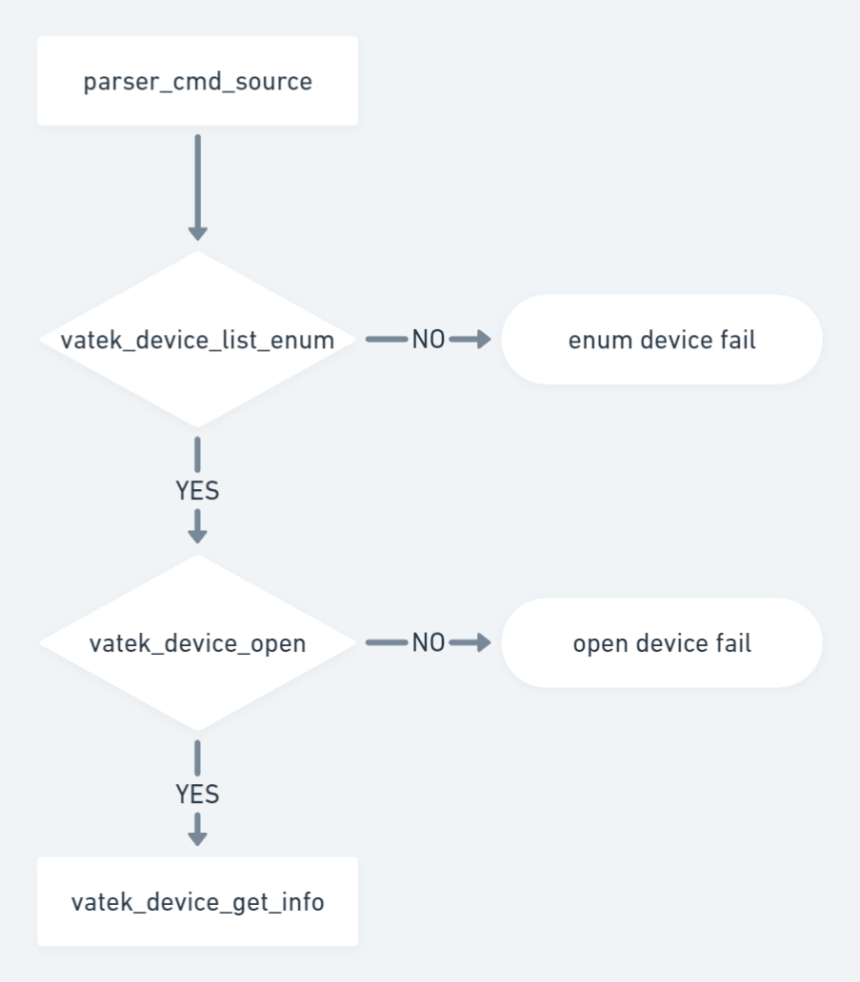
3.2.5 A Series Example：

The mainly useage of A Series is app_stream.c（sample\ app_stream\ app_stream.c） as an example, the program will demonstrate the process of A Series operation.

- **Adjust modulation parameter**

```
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,coderate_5_6,},},  /* modulator param */
},
    .sync = {NULL,NULL},
};
```

■ **Step 1：Initialize the device and start.**



◆ Device enumeration.

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

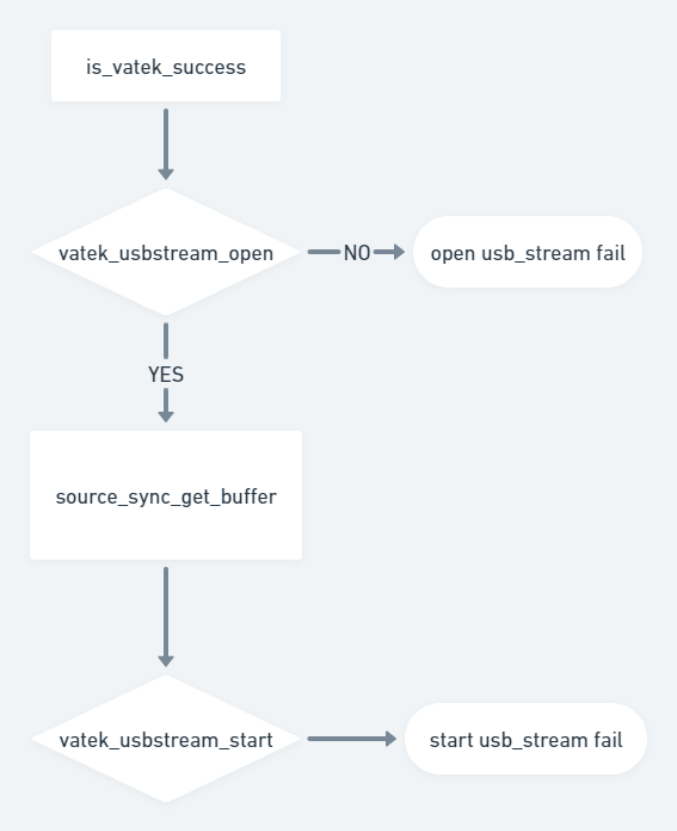
◆ Device connect.

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

◆ Read device information.

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

■ **Step 2：Connect USB stream, set USB mode and start the device.**



◆ Conncect USB stream.

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

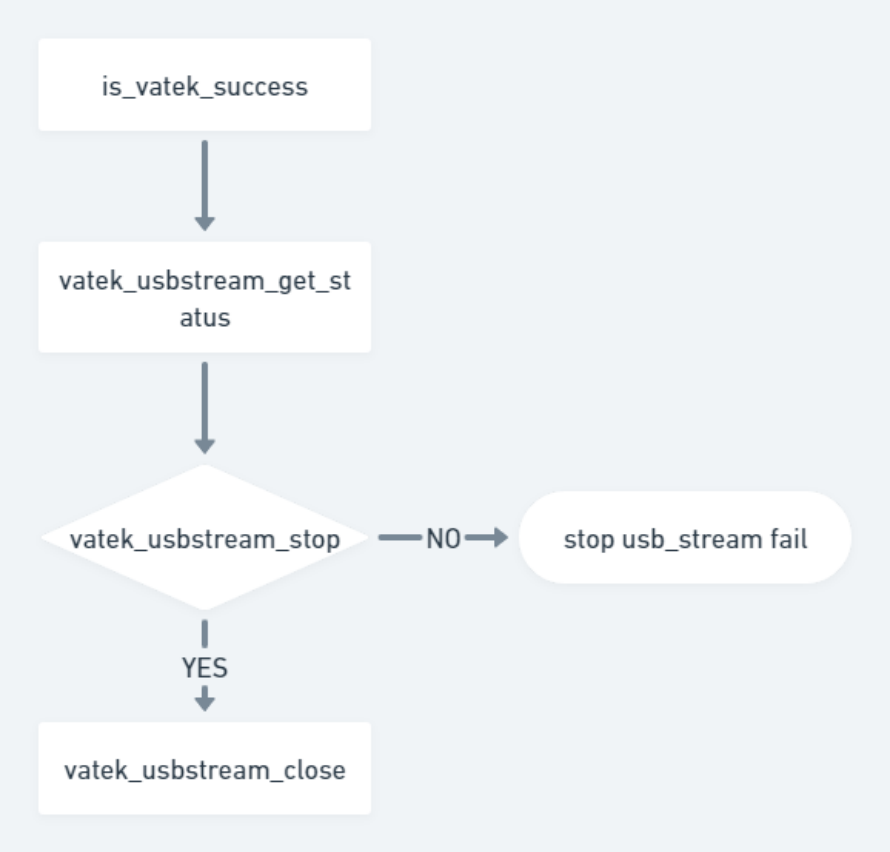
◆ Setting USB mode.

```
usbcmd.mode = ustream_mode_sync;

usbcmd.sync.param = &streamsource;

usbcmd.sync.getbuffer = source_sync_get_buffer;
```

◆ Stream start.

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

- **Step 3：When usb_stream have valid buffer, source_sync_get_buffer will be called internally, usb_stream can continuouslt check status and information, turn off usb_stream after completion.**



- Read stream status.

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

- Stream status and information.

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

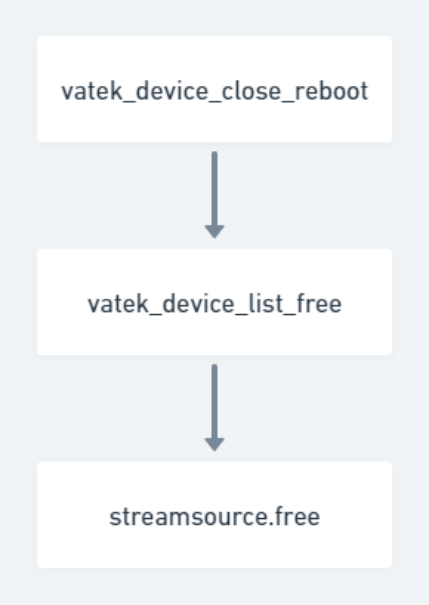
- Stop stream.

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

- Stop Stream.

```
vatek_usbstream_close(hustream);
```

■ **Step 4 : Device close and reboot.**



- Device reboot.

```
vatek_device_close_reboot(hchip);
```

- Clear device list.

```
vatek_device_list_free(hdevlist);
```

- Clear stream source.

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

4 SDK Building

The software development kit aims at the development of applications and products running on high-level operating systems （Windows, Linux, and macOS.） Developers can compile and develop relevant applications in Microsoft Windows 10 and later or Ubuntu 18.04 （or other systems supporting relevant compilation environment.）At present, the building tool is mainly CMake.

4.1 Requirements for Compling：

- Microsoft Visual Studio 2019 （https://visualstudio.microsoft.com/）, either commercial version or community free version is acceptable.
  - ◆ While setting up Visual Studio, please make sure to keep C++ desktop developing.
- CMake （https://cmake.org/）Windows version（3.22 version and later）
- Qt5 （MSVC package for your Visual Studio version）
  - ◆ The version of development is Qt 5.14.2（Qt5.14.2）

4.2 Build Options

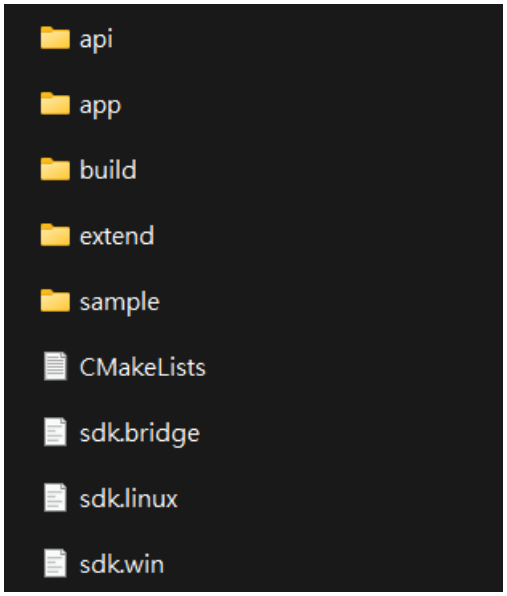
The following parameters are provided in CMAKE environment and can be set according to requirements before building.

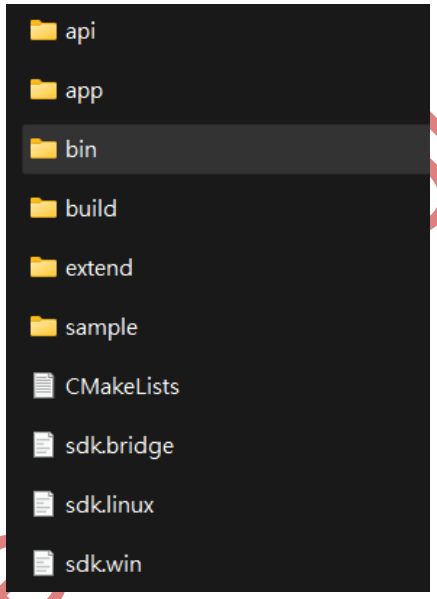
Parameter	Description	Value
SDK2_EN_QT	Whether to compile QT program development interface and application.	ON|OFF
SDK2_EN_APP	Whether compile application.	ON|OFF
SDK2_EN_SAMPLE	Whether compile sample program.	ON|OFF
SDK2_QTDIR	Set QT installation folder for used.	

4.3 Compiling vatek_sdk_2 :

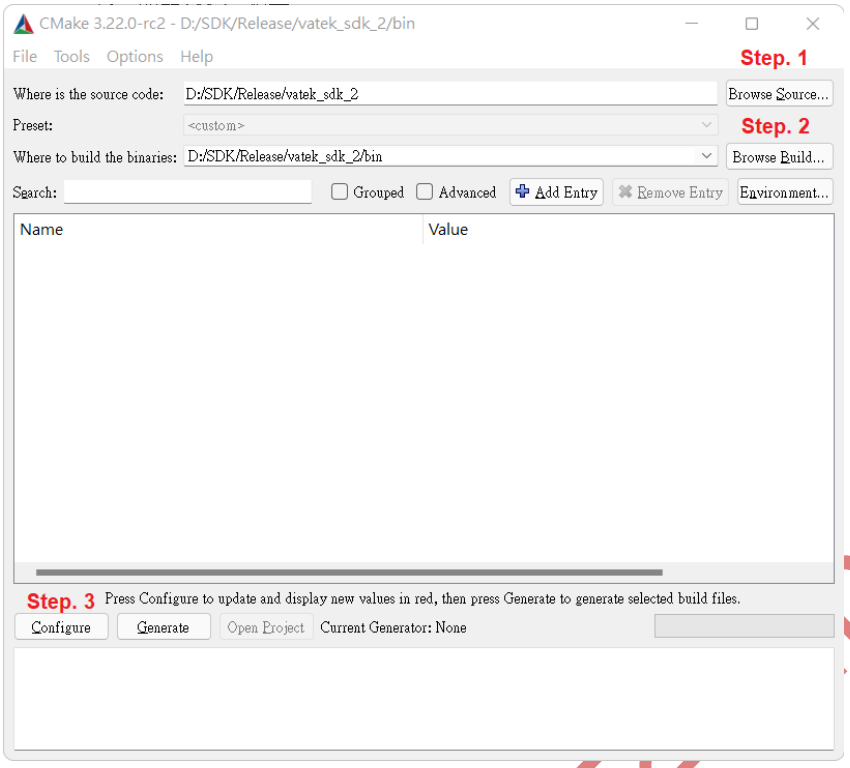
4.3.1 Windows operation system

- Decompress vatek_sdk_2.zip, you can see the following folder. You can choose to add a new folder under vatek_sdk_2 to save CMake building files (take “bin” folder as example).

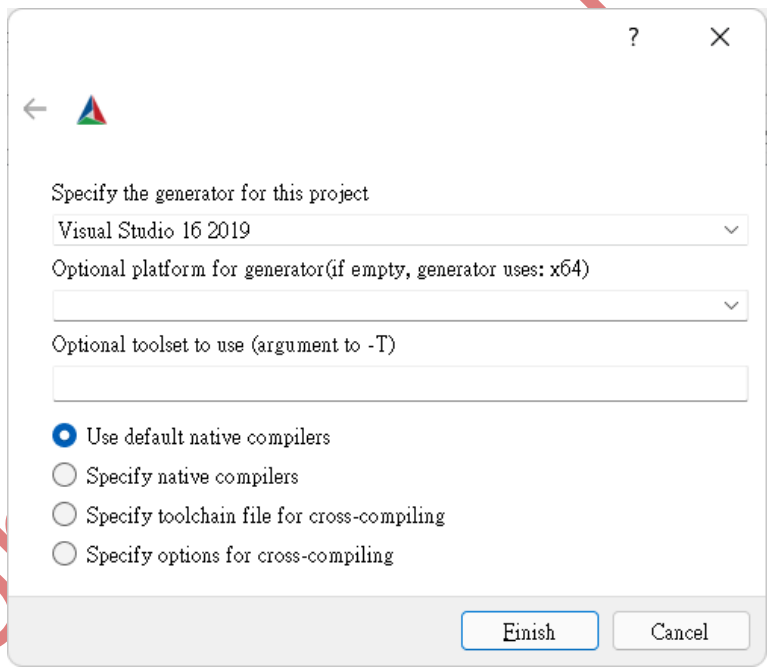




- Open CMake application
  1. **Step. 1** select vatek_sdk_2 folder in the “source code” section.
  2. **Step. 2** select bin folder under vatek_sdk_2 folder in the “build the binaries” section.
  3. **Step. 3** click Configure button.

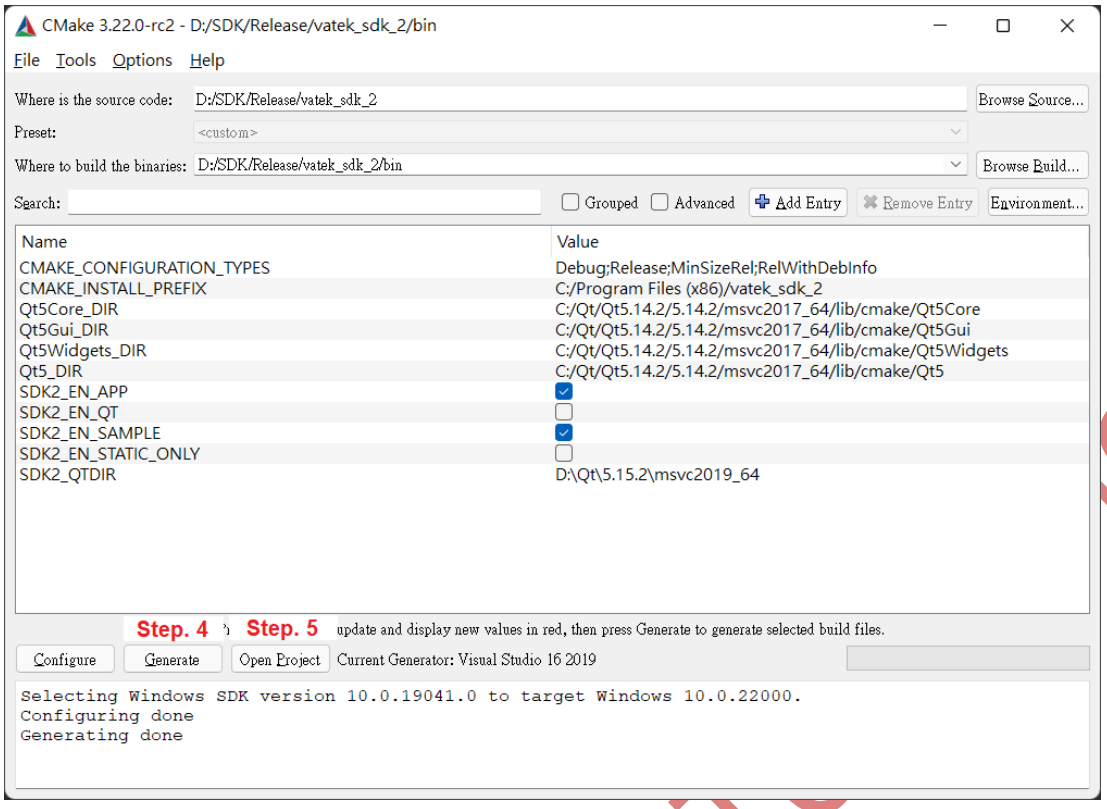


◆ Select Visual Studio 2019 version, the default to use x64 compiling.

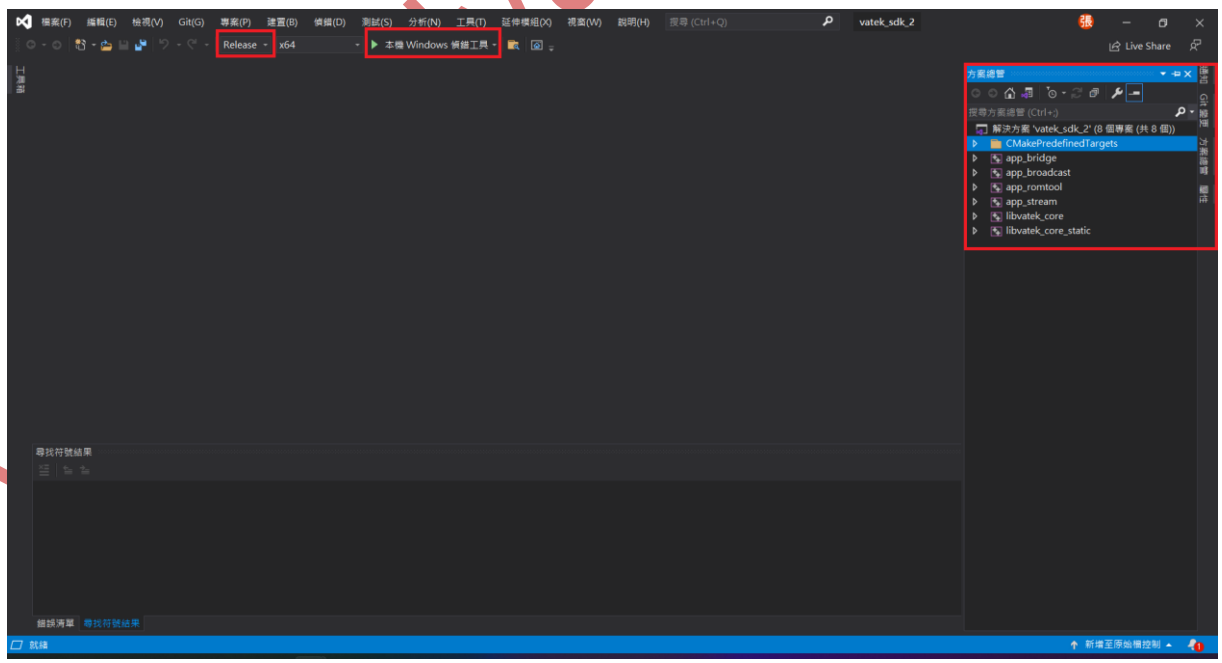


◆ **Step. 4** Select Generate button, the following window will appear when the compilation is successful.

- (You can choose whether to use QT interface. If not, please use sdk2_ EN_ QT uncheck.)



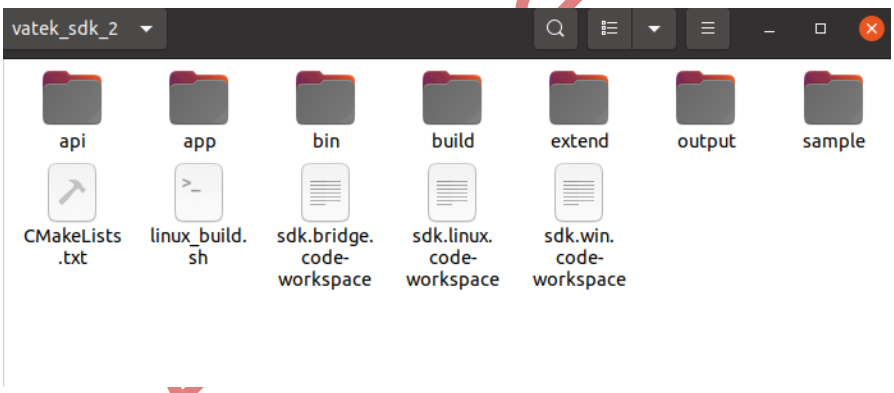
- **Step. 5** Open Project file.
  - After you open Visual Studio, you can see 6 projects under vatek_sdk_2 at the “Project Explorer”. Next to “debugging tool”, you can select Debug mode or Release mode to implement compilation.



❍ After the successful compiling, there will be a “Debug folder” or “Release folder” in the bin folder, which will contain executable exe files and DLL files of all APIs.

4.3.2 Ubuntu operation system

- Uncompress vatek_sdk_2.zip, you can see the following folders, which can be selected in vatek_sdk_2 folder then add a new folder to store the files completed by CMake (take bin folder as an example.)

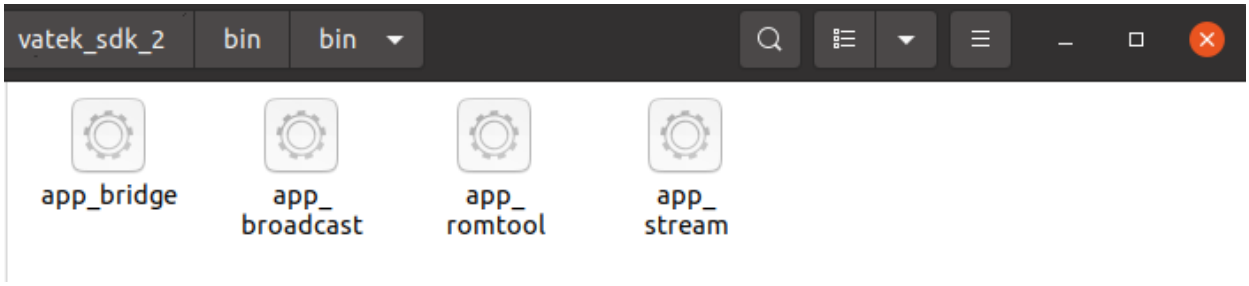


- Open Terminal to move the current content to *vatek_sdk_2/bin* folder, execute *cmake.* Complete CMake software development, the completed files will be placed in vatek_sd2_2/bin folder. After success execute *make*to start compiling SDK files.

```
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
```

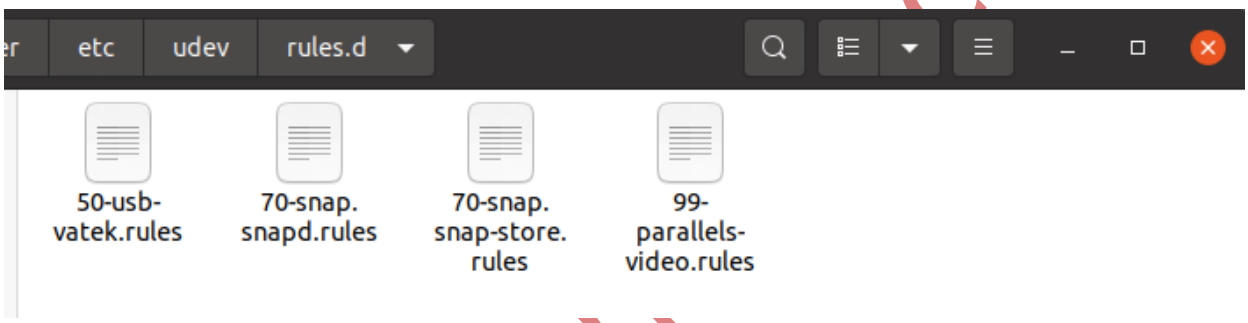
- After successful compilation, 100% success will be displayed, which means that there is no error in the process. You can see the successfully compiled sample code in the bin folder.

```
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/__/co
mmon/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/s
rc/tool_stream_test.c.o
[ 93%] Building C object sample/app_stream/CMakeFiles/app_stream.dir/__/common/s
rc/tool_stream_udp.c.o
[ 94%] Building C object sample/app_stream/CMakeFiles/app_stream.dir/__/common/s
rc/tool_tspacket.c.o
[ 96%] Building C object sample/app_broadcast/CMakeFiles/app_broadcast.dir/__/co
mmon/src/tool_stream_test.c.o
[ 96%] Built target app_romtool
[ 96%] Building C object sample/app_broadcast/CMakeFiles/app_broadcast.dir/__/co
mmon/src/tool_stream_udp.c.o
[ 97%] Building C object sample/app_broadcast/CMakeFiles/app_broadcast.dir/__/co
mmon/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
```



- In order to identify our device in the Ubuntu system, we must move the USB identification file to the system folder, and there is the file name linux_build.sh in build \linux folder. It can automatically install the file. After execution, be sure to check whether it is successful. Otherwise, the device cannot be recognized in Ubuntu. Please restart if the installation is successful.

```
user@user-VirtualBox:~/20220520/build/linux$ sudo bash ./linux_build.sh
```



5 Main Application of Software Development Package

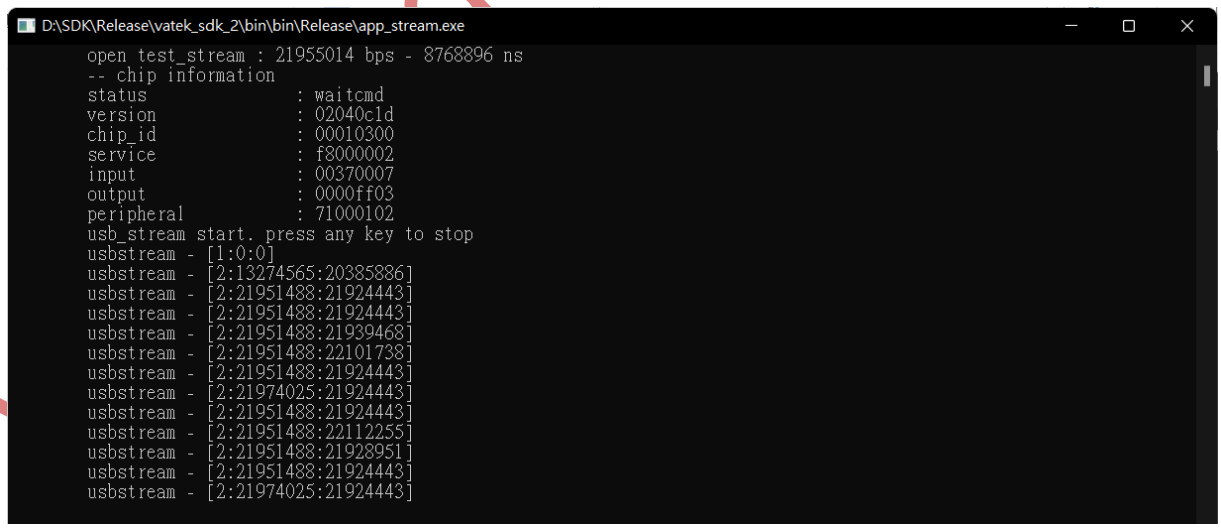
The software development includes the program development interface and sample program required for developing USB streaming DTV. For the main program interface, please refer to the relevant program interface definitions in the path vatek_sdk_2/api/core/inc, mainly used with the following example programs.

5.1 app_stream Example Program（Use for A Series）

app_stream source code is placed in vatek_sdk_2/sample/app_stream, this example fully demonstrates the relevant operation process and adjustment parameters of DTV conversion through USB. Please refer to the relevant implement original file app_stream.c . Compiled app_stream has the following functions.

```
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] [pcr|passthrough]
                - app_stream udp [ip address] [pcr|passthrough]
                - app_stream rtp [ip address] [pcr|passthrough]
        demo finished. press any key to quit
```

- When the device starts broadcasting, the following screen will appear.
- usbstream – [status : data Bitrate : current Bitrate] → Bitrate indicates normal operation.



1. app_stream [empty] no additional parameters are used.

   Through the USB device, write an MPEG-TS generated by the program to the modulation device and convert it to define DTV modulation.

2. app_stream file [mpegts_filename] use file and a specified MPEG-TS file.

   Convert a ready-made MEPG-TS file to DTV modulation, support 188 or 204 format, and must include at least one PCR.

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

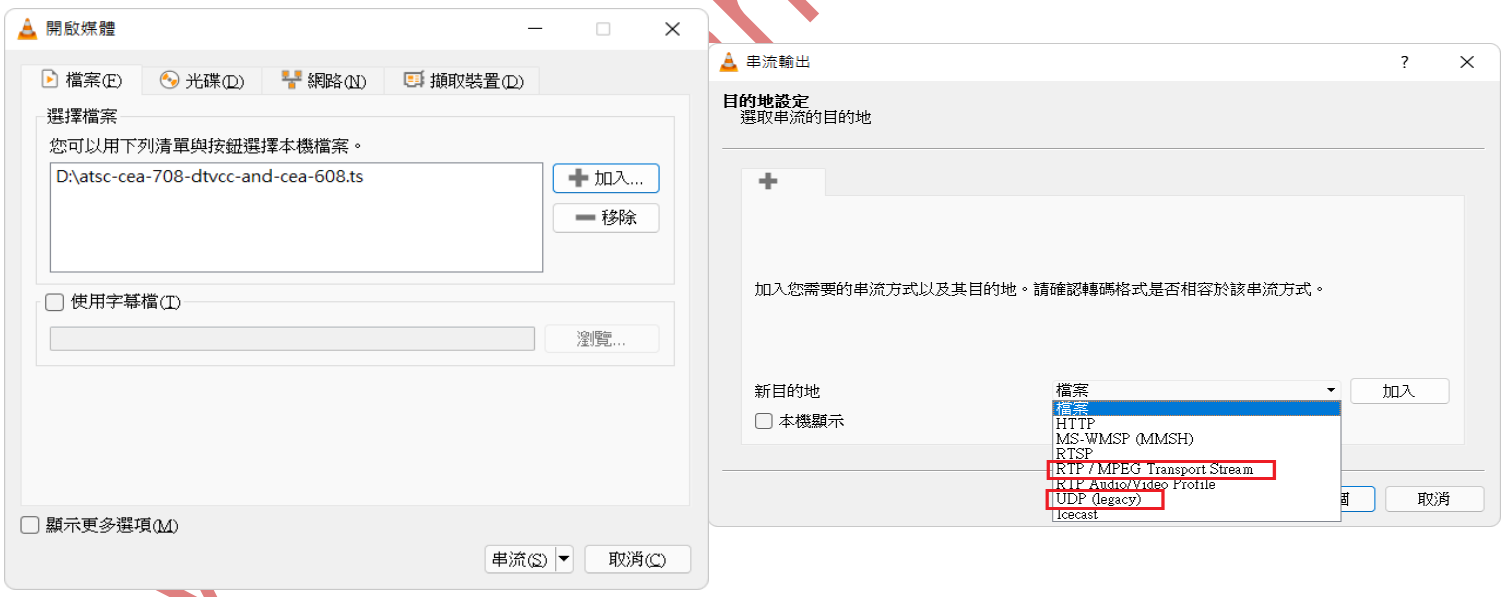
3. app_stream udp udp://xxx.xxx.xxx.xxx:yyyy use UDP-MPEGTS as the source.

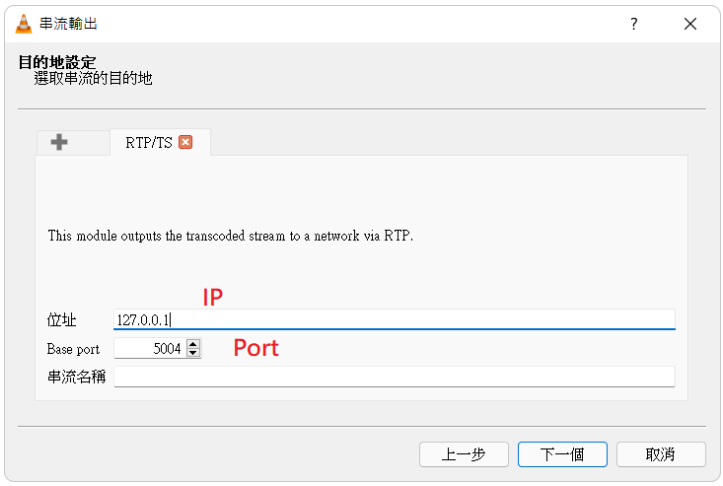
```
D:\SDK\Release\vatek_sdk_2\bin\bin\Release>app_stream udp udp://127.0.0.1:5004 pcr
```

4. app_stream rtp rtp://xxx.xxx.xxx.xxx:yyyy use RTP-MPEGTS as the source.

   An example of URL format is rtp://127.0.0.1:1234. In the front xxx is the standardIP location, yyyy is port, and the following options can be used as the test source for the corresponding VLC settings.

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

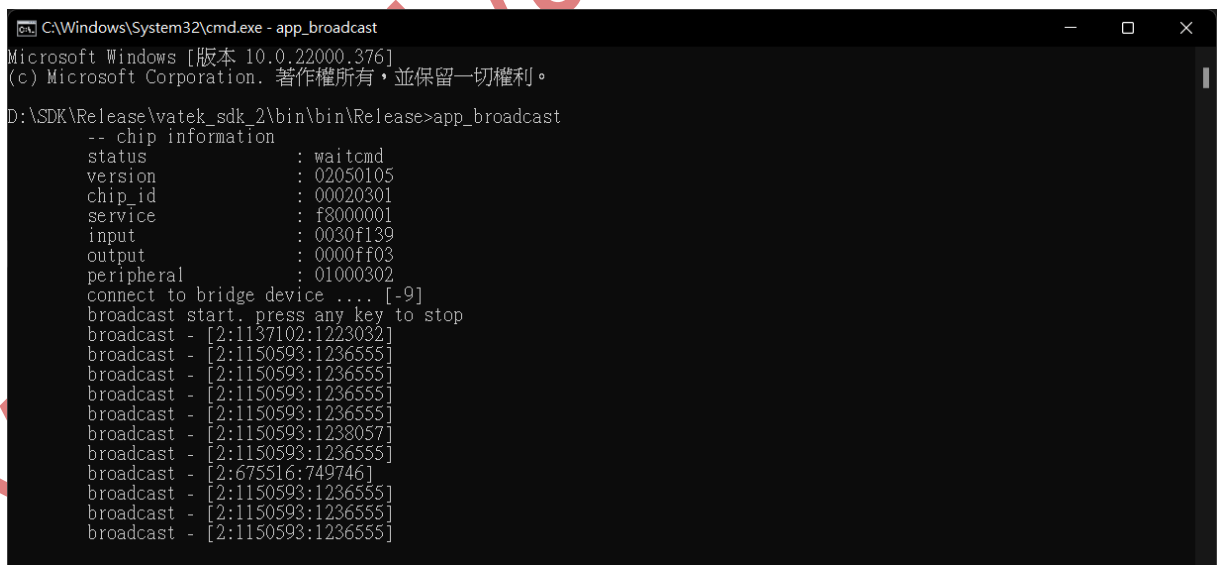




5.2 app_broadcast Example Program（Use for B Series）

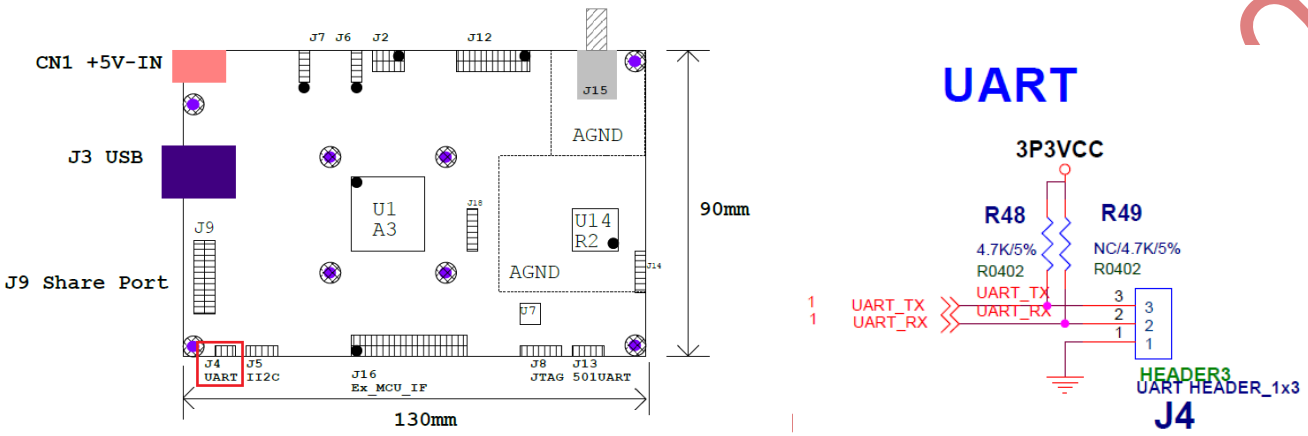
app_broadcast source code located in vatek_sdk_2/sample/app_broadcast, this example completely demonstrates the relevant operation flow of controlling the coding and broadcasting of B Series. After operation, the parameters will be adjusted according to the setting to make the B Series device output Colorbar. Please refer to the relevant implement original file app_broadcast.c.

- When the device starts broadcasting, the following screen will appear.
- usbstream – [status : data Bitrate : current Bitrate] → Bitrate indicates normal operation.



5.3 Check Log with VATek Device

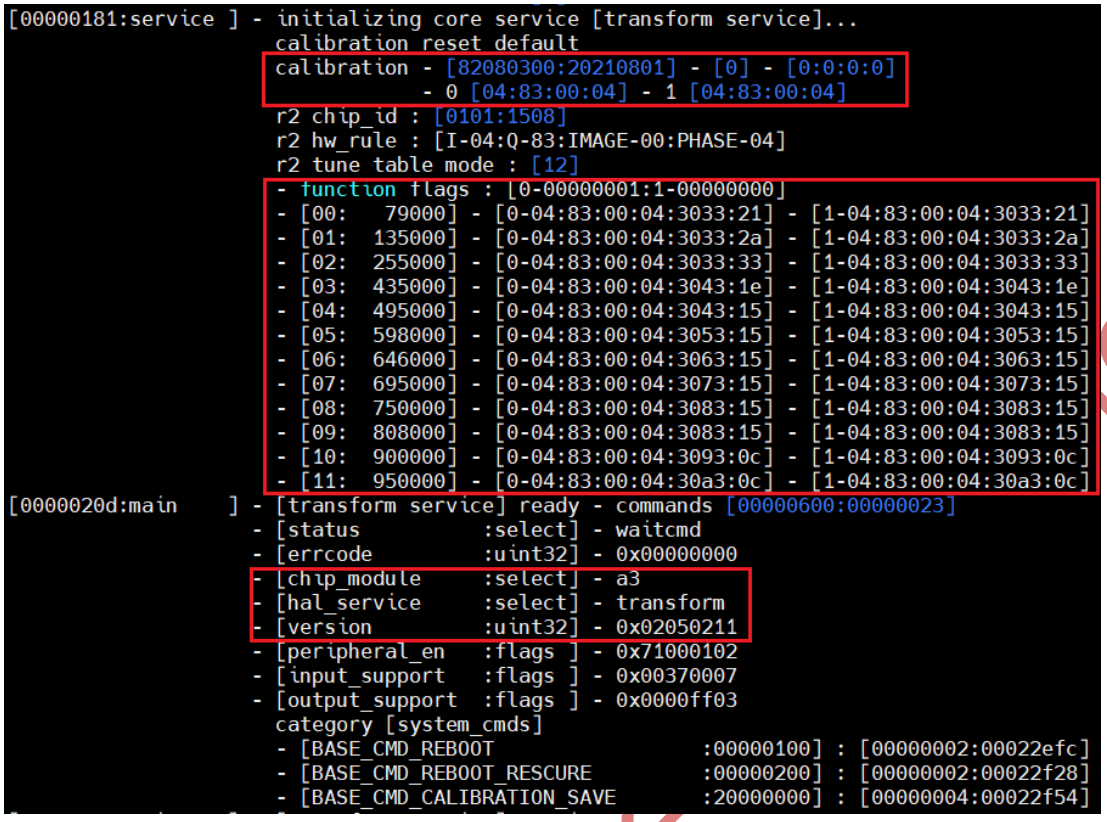
When developing the device, you can connect the device LOG to check if it is normal.The following figure is the mainboard diagram of our device. We provide UART interface to connect the device LOG. As long as you connect J4 (TX, RX, GND), if the device is successfully connected, the LOG of the following figure will be displayed.



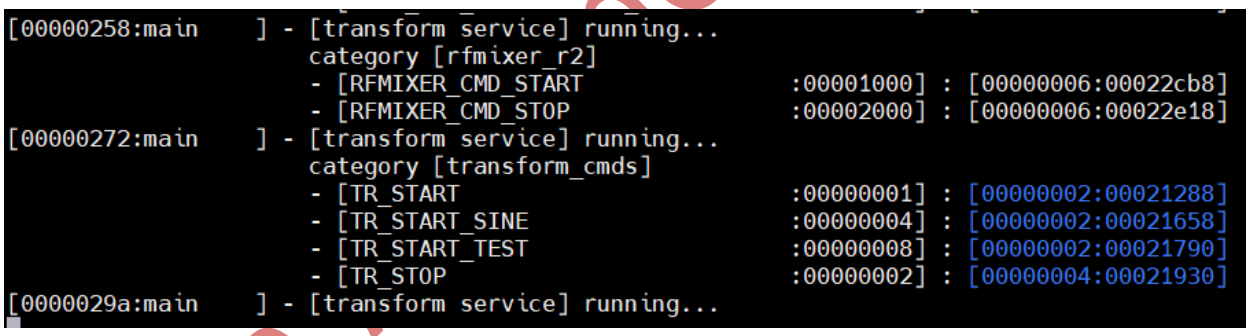
- The part of the red color box indivated if the additional function of the device is turned on, such as status light and customized USB ID.

```
 - 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]
```

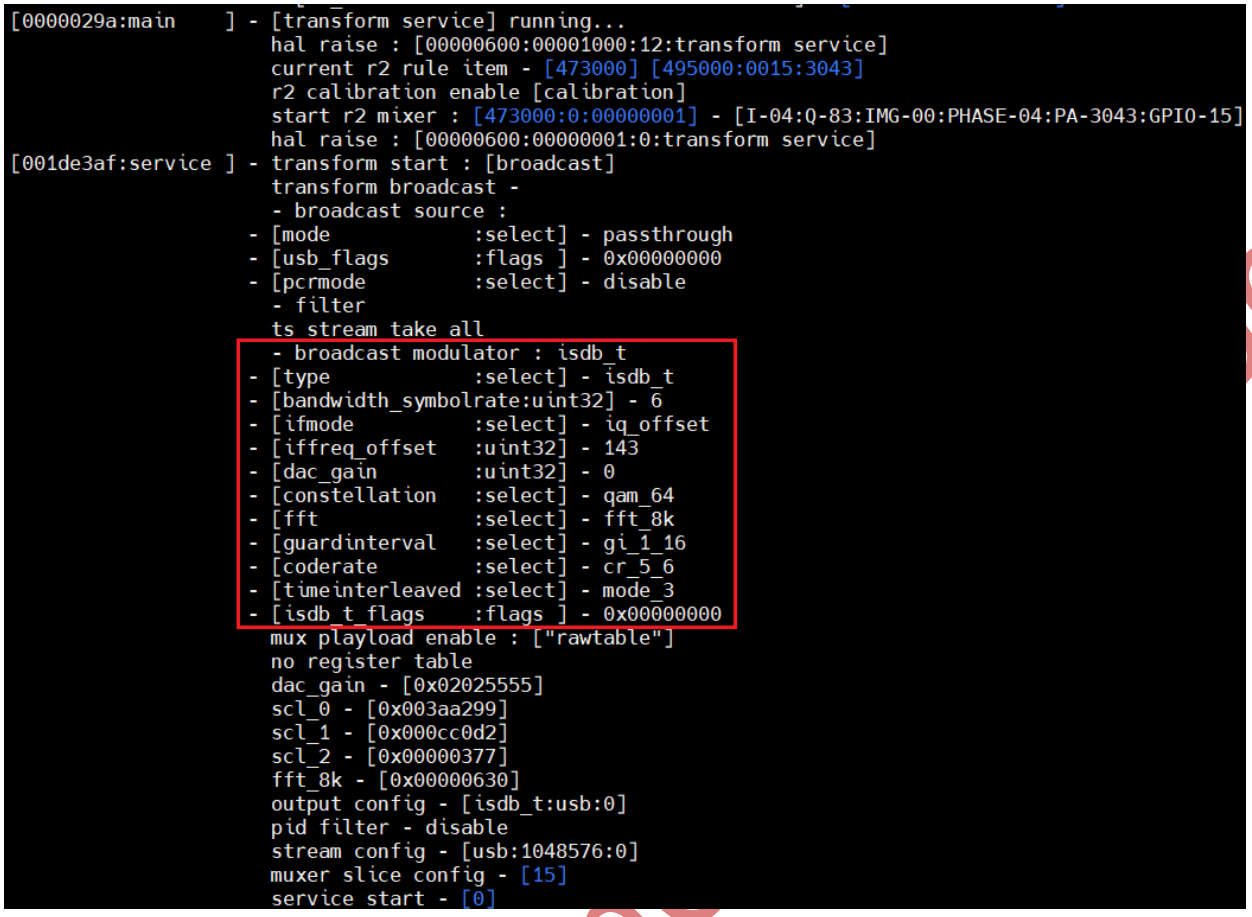
- The red boxes are the calibration parameter, R2 table, chip model, service and firmware version in order.



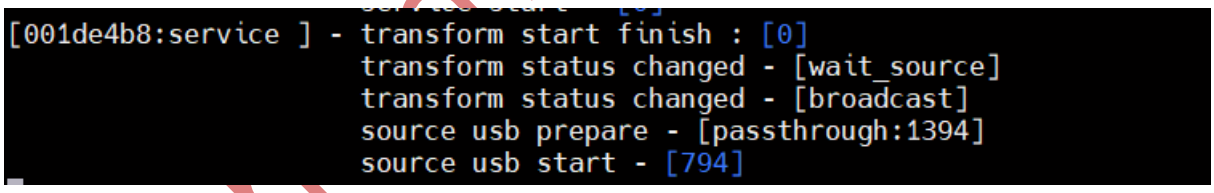
- If there is no problem with the device, it is displayed to the end.



- After the device starts broadcasting, you can see if the set value of modulation is successful in the red box.



- This LOG will be displayed when the device is in progress (in case of abnormal LOG, please contact us to help find out the issue).



6 System Debugging Function

If the error code pops out when using the device, please mainly refer to the following chart.

Error Code	Issue
-1	unknown fail
-2	function not supported
-3	parameter set not supported
-4	buffer limited overflow
-5	can not call at this device status
-6	send command to device fail or call system api fail
-7	wait operation timeout
-8	system is busy
-9	device not exists
-10	format not current
-11	memory alloc fail or overflow