Application Note

Telink RF4CE SDK Guide

AN-19111900-E1

Version 1.0.0

2019-11-25

Brief:

This document is the guide for Telink RF4CE SDK.

1.0.0 for 8258 and 826x family

TELINK SEMICONDUCTOR

Published by

Telink Semiconductor

Bldg 3, 1500 Zuchongzhi Rd,

Zhangjiang Hi-Tech Park, Shanghai, China

© Telink Semiconductor

All Right Reserved

Legal Disclaimer

This document is provided as-is. Telink Semiconductor reserves the right to make improvements without further notice to this document or any products herein. This document may contain technical inaccuracies or typographical errors. Telink Semiconductor disclaims any and all liability for any errors, inaccuracies or incompleteness contained herein.

Copyright © 2019 Telink Semiconductor (Shanghai) Ltd, Co.

Information:

For further information on the technology, product and business term, please contact Telink Semiconductor Company (www.telink-semi.com).

For sales or technical support, please send email to the address of:

telinkcnsales@telink-semi.com

telinkcnsupport@telink-semi.com

Revision History

Version 1.0.0 (2019-11-19)

This is the Initial release.

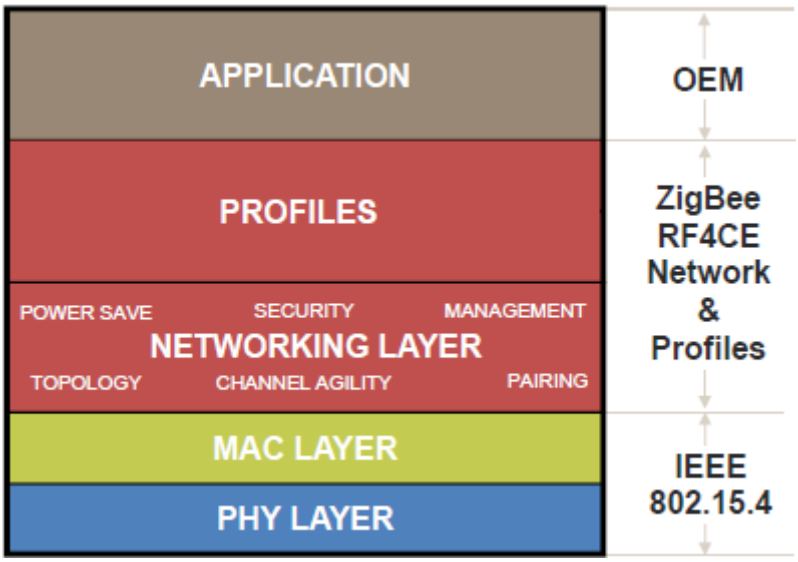
Contents

Contents of Figures

1. Introduction to RF4CE

1.1 Stack architecture

Figure 1-1 Outline of RF4CE Stack Architecture



- ✧ Physical (PHY) Layer
- ✧ IEEE 802.15.4 Medium Access Control (MAC) Layer
- ✧ RF4CE Network (NWK) layer
- ✧ Profiles layer
- ✧ Application layer

1.2 Device type

1. **RF4CE Target Node(Target)**: target maintains accepts or declines pairing requests and makes decision on operating channel (frequency agility),such as STB, TV, and so on, Telink will provide dongle as target.
2. RF4CE Controller Node (Remote Control Device):

- ✧ Initiates pairing and discovery process to Target Nodes
- ✧ Low power mode.

2. Functional Description

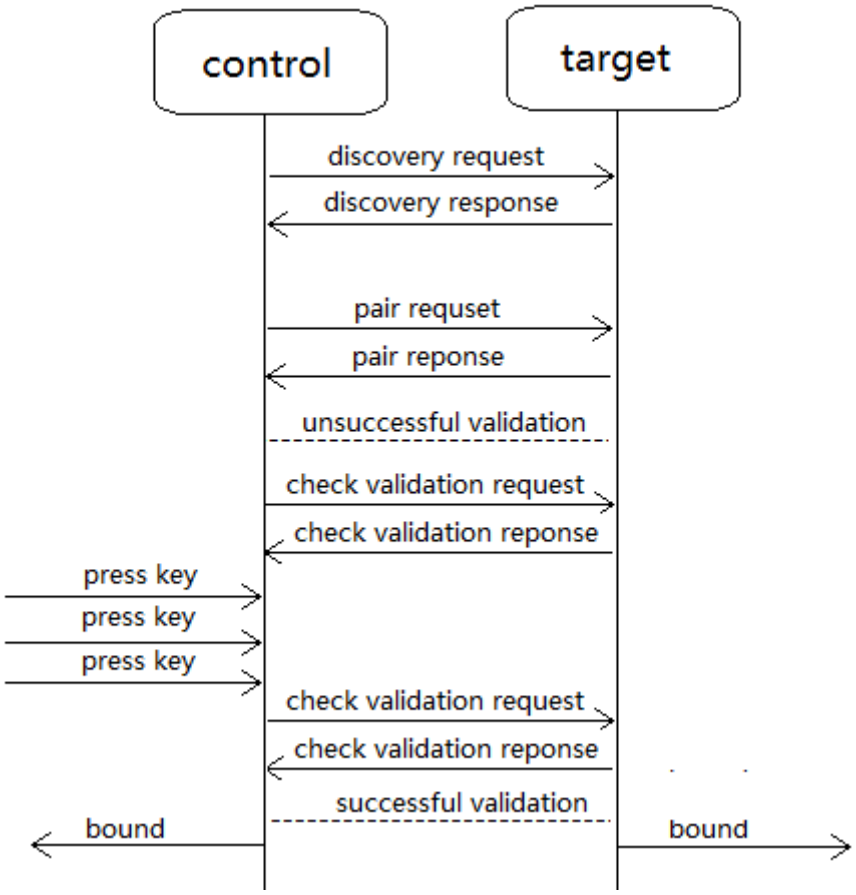
2.1 Not bound

The RC did not attempt to bind with a target yet, or, the RC did not succeed to bind with a target, or, the RC is unpaired with a target. All the above states are Not bound. When the RC is in this state, the NWK layer pairing table is empty. As such, no RF data frames can be sent to any target, but the IR data will be sent.

2.2 Binding

When user press the pairing key of RC, the RC will enters the binding state. The detailed pairing process is as follows:

Figure 2-1 Binding Procedure



Once the target is in the NWK layer pairing table, the controller can send data frames to the target that will only be used for validation purposes.

2.3 Bound

The RC successfully bound with the target: the RC has temporarily paired with the target, and has been informed that the validation procedure completed successfully, or, the RC did not succeed to bind with the target, but it was bound to the same or other target before, and this state was restored.

The target is in the NWK layer pairing table and the RC can send data frames to the target that can be used for all purposes.

3. Telink RF4CE SDK

3.1 Task Management

3.1.1 Timer task management

3.1.1.1 Timer task application

```
ev_on_timer (ev_timer_callback_t func, void *arg, u32 t_us)
```

“func”: task callback function.

```
typedef int (*ev_timer_callback_t)(void *data);
```

- ✧ Return 0: It indicates this time event always exists, and it retains the pre-configured time interval.
- ✧ Return -1: It indicates this time event will be automatically released after it’s executed.
- ✧ Others: It indicates this time event always exists, and the time interval change is the return value.

“t_us”: Interval for the time event. Unit: us.

“arg”: Incoming parameter of the time event.

```
ev_unon_timer (ev_time_event_t **te)
```

“te”: Pointer of the pointer of time event

3.2 Memory Management

Telink Zigbee SDK allocates three sizes of memory: 24B*8, 60B*8, 150B*5.

Before memory allocation, user must be able to invoke the function “**ev_buf_init()**”.

3.2.1 Memory Application

```
u8   *ev_buf_allocate(u16 size)
```

size: 24 or 48 or 162

3.2.2 Memory Release

```
buf_sts_t   ev_buf_free(u8 *pBuf)
```

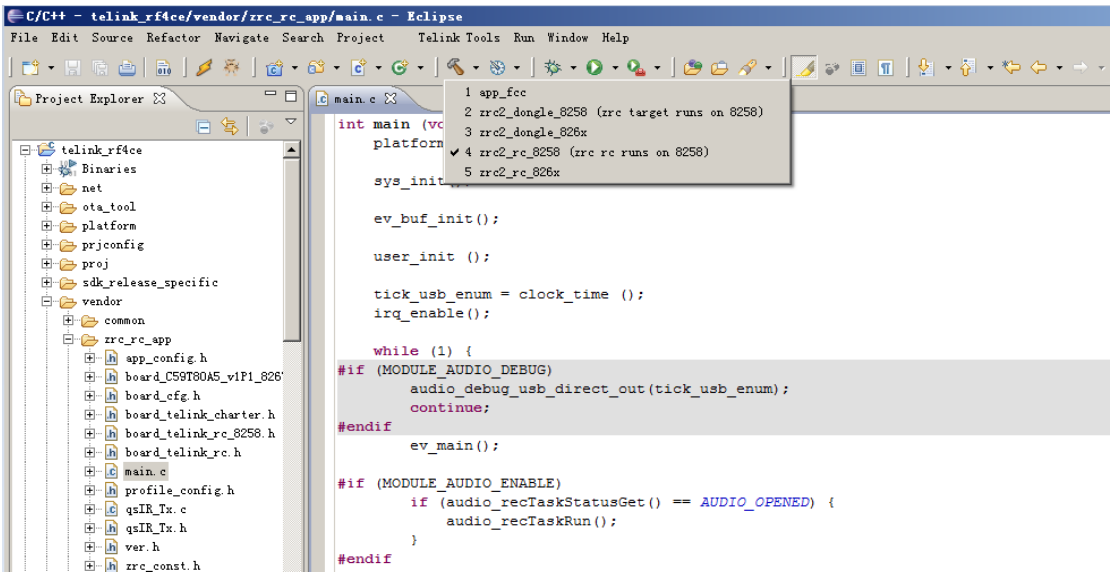
3.3 Supported platform

3.3.1 Project

Telink RF4CE SDK Supports platform 8269/8258

The project ending in 8258 is the project of 8258, and the project ending in 826x is the project of 8269.

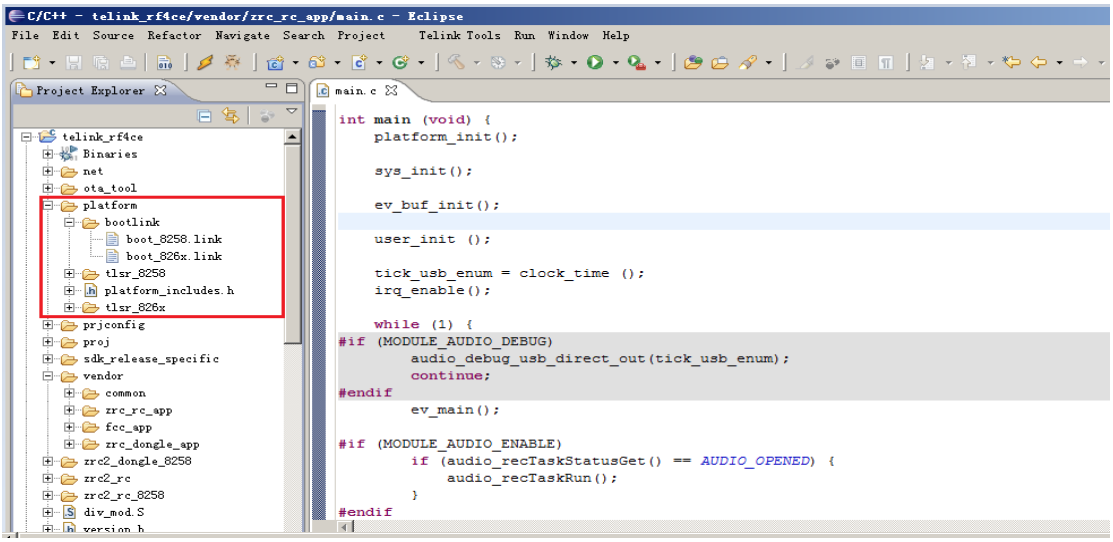
Figure 3-1 Compile Project on 8269/8258



3.3.2 Driver and bootlink

The underlying driver and bootlink files related to the chip are all in the platform folder:

Figure 3-2 Driver and Bootlink Paths on 8269/8258



3.4 Hardware Abstraction Layer

3.4.1 Initialization

```
cpu_wakeup_init()
clock_init()
gpio_init();
ZB_RADIO_INIT();//set reg about RF mode
```

3.4.2 Timer

```
TIMER_STATE_CLEAR(idx)
TIMER_STOP(idx)
TIMER_START(idx)
TIMER_TICK_CLEAR(idx)
TIMER_INTERVAL_SET(idx, cyc)
TIMER_INIT(tmrIdx, mode)
```

3.4.3 Interrupt

```
void irq_init(u32 irq_mask)
```

"irq_mask": Interrupt request (IRQ) bit to be enabled.

```
u8 irq_disable(void)
```

return: current irq mask

```
u8 irq_enable(void)
```

enable interrupt

```
u8 irq_restore(u8 r)
```

It's used in combination with "irq_disable（）", and it serves to restore the status before "irq_disable（）".

3.4.4 Flash

```
void flash_write(u32 addr, u32 len, u8 *buf);
```

addr: the address of the flash

len: the length will be written

buf: the address of the data will be written

```
void flash_read(u32 addr, u32 len, u8 *buf);
```

addr: the address of the flash

len: the length will be read

buf: the address of the data will be read

```
void flash_erase(u32 addr);
```

addr: the address of the flash

3.4.5 PWM

```
void drv_pwm_init(void);
```

pwm module initialization

```
void drv_pwm_cfg(u32 pwmId, unsigned short cmp_tick, unsigned short cycle_tick)
```

Configures the PWM carrier frequency and duty cycle

pwmId: channel ID

cmp_tick: system clock frequency/carrier frequency

cycle_tick: duty cycle, duty ration * cmp_tick

3.4.6 drv_uart

void drv_uart_init(uart_baud_rate_e baudrate, u8 *rxBuf, u8 rxBufLen, uart_irq_callback uart_recvCb)

UART module initialization

baudrate: UART_BAUDRATE_115200 and UART_BAUDRATE_9600

rxBuf: buffer for receiving data

rxBufLen: rxBuf length(maximum packet length)

uart_recvCb: callback for receiving a packet

void uart_rx_irq_handler(void)

UART Rx ISR

void uart_tx_irq_handler(void)

UART Tx ISR

u8 uart_tx_done(void)

Check the status of the transmitting.

1: Done, 0: Busy

u8 uart_tx_start(u8 *data, u32 len)

Start UART transmitting

data: the pointer to the data will be sent

len: the length of the data will be sent

return: 0: Failure, other: success

3.4.7 drv_pm

system enter low power mode

void platform_lowpower_enter(platform_mode_e mode, platform_wakeup_e src, u32 cycle_ms)

mode: low power mode, including

PLATFORM_MODE_SUSPEND,

PLATFORM_MODE_DEEPSLEEP,

and PLATFORM_MODE_MCU_STALL

src: wakeup source, including

PLATFORM_WAKEUP_PAD and

PLATFORM_WAKEUP_TIMER

cycle_ms: the interval of the low power if the wakeup source of PLATFORM_WAKEUP_TIMER is enable, unit: ms

3.4.8 IR

```
ir_init(IR_PWM_ID);
```

Initialize PWM channel IR_PWM_ID

```
IR_PWM_PIN_CFG;
```

Call void gpio_set_func(GPIO_PinTypeDef pin, GPIO_FuncTypeDef func)

uses IO port as PWM function

```
hwTmr_init(TIMER_IDX_1, TIMER_MODE_SCLK);
```

Init the timer for IR timer

3.4.9 ADC

```
unsigned char  drv_adc_init();
```

Initialize the ADC module, fixed the clock as 4MHz. For the 8258 chip, default use the misc channel, and the pre_scaler is 1/8.

return: 0: Failure, 1: success

```
void drv_adc_mode_pin_set(Drv_ADC_Mode mode, GPIO_PinTypeDef pin);
```

This function sets the ADC mode and the sampling pin

input parameter:

Mode:

0:Drv_ADC_BASE_MODE, the mode is used for ADC configuration of ADC IO voltage sampling

1: Drv_ADC_VBAT_MODE, the mode is used for ADC configuration of ADC supply voltage sampling

You can use the enum Drv_ADC_Mode in proj/drivers/drv_adc.h

Pin: Must be a pin that can be reused as an ADC function，for specific pins, please refer to the datasheet of the relevant chip.

It should be noted that when the mode is set to 1, that is, Drv_ADC_VBAT_MODE, it also needs to occupy a pin, but the physical connection is completed inside the chip and the user does not need to deal with it.

Return :none

```
void drv_adc_enable(bool enable);
```

This function is used to enable or disable the ADC

input parameter:

enable:

0: enable the ADC

1: disable the ADC

When initialization and mode Settings are complete, set 0 to to enable ADC. Or ADC operation completed, set 0 to disable ADC

Return:none

unsigned short drv_get_adc_data(void);

This function serves to set adc sampling and get results

input parameter: none

Return: the result of sampling, the data type is unsigned short

In the sdk, the two functions required to use ADC are as follows: Battery Dection and Audio, but the chip has only one ADC channel, so Battery detection and Audio can not run at the same time, enable Battery detection by default after power on, If Audio is needed, Battery detection will be turned off.

3.4.9.1 Battery Detection

```
void drv_adc_battery_detect_init(void)
```

This function is Used to set ADC to Drv_ADC_VBAT_MODE and initialize ADC pin, and it is called when power on or Audio is off

return: none

```
app_batteryDetect(void);
```

Call this function to get the current battery charge, the level of battery is stored in zrc_appVars.battSta, the position of this function is vendor/zrc_rc_app/zrcApp.c.

3.4.9.2 Audio

In this SDK, PCM data will be placed in a pre-set buffer, then the PCM data will be compressed into ADPCM, and then the ADPCM data will be sent out.

```
AMIC_PIN_CFG(inPin0, inPin1, biasPin)
```

In this SDK, the default is to use AMIC, this macro is used to set the pins: AmicSP, AmicSN, AmicBias.

i.e 8258 RC: AMIC_PIN_CFG(GPIO_PC0,GPIO_PC1,GPIO_PC4)

void audio_recInit(u32 sampleRate, audio_rec_ntf audioUserhandler)

This function sets Audio's sampling rate and the callback function pointer for state switching

input parameter:

sampleRate: the default value is 16000

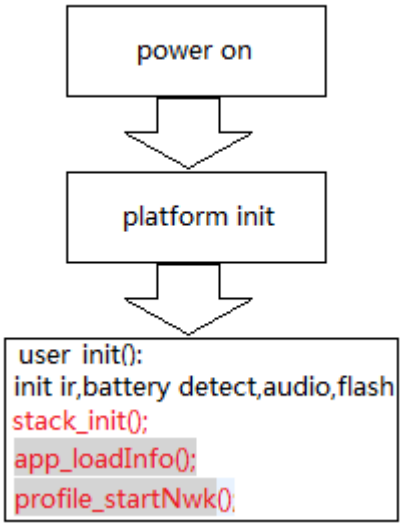
audioUserhandler: This callback function is called when audio is on or off, and allows the user to set state or handle UI functions

return: none

3.5 Application Development

3.5.1 Stack Initialization

Figure 3-3 Power on Initialization Block Diagram



(The red words are stack initialization codes.)

void profile_init(void);

Init RF4CE stack and start configured Profiles

void zrcApp_initPib(void);

Initialize the necessary Network PIB

void profile_registerUserCb(profile_cbFunc_t* pUserCb)

pUserCb-> pStartCnfFn: start stack confirm command

pUserCb-> pPairCnfFn: pairing confirm command

pUserCb-> pUnpairCnfFn: unpair confirm command

pUserCb-> pUnpairIndFn: unpair indication command

void zrc_registerUserCb(zrc_userCbFunc_t* pZrcCb)

Register the application call back of the ZRC event

void app_loadInfo(void)

load application info info from analog register/NV when RCU power on or wakeup from deep sleep.

void profile_startNwk(void)

Start RF4CE network

The above initialization is complete, and the interaction between the layers is completed in ev_main ()

3.5.2 APIs

3.5.2.1 Paring

```
profile_sts_t profile_startPair(u8 userSepcific,
u8 searchDevType,
u8 keyCnt,
u8 pressKey)
```

UserSpecific: 1 indicating if user string will be included,

0 indicating no user string

searchDevType: The type of device the application is attempting to discovery,

ref RF4CE_DEVICE_TYPE

keyCnt: The key exchange transfer count.

pressKey: The key exchange transfer count.

Return: ERR_NONE- Success

Other error code defined in

3.5.2.2 un-Paring

```
profile_unpairReq(zrc_appInfo.pairingRef)
```

pairingRef: the index od the paring table

return: NONE

3.5.2.3 Key Code Transmission

RF Transmission

```
u8 zrc_sendRFData(u8 profileId, u8 keyCode)
```

profileId: profile identifier, default is *PROFILE_ZRC2*

keyCode: key code

IR transmission

```
u8 zrc_sendIRData(u8 profileId, u8 keyCode, u8 single_key)
```

profileId: different IR code type

keyCode: key code

single_key: single key or repeat key

3.5.2.4 Audio

Initilization

```
void tl_audioRecInit(u8 profileId,
tl_audioRecInfo_t *info,
tl_audioUserCb_t userCb)
```

profileId: profile identifier, *PROFILE_ZRC2* is as default

info: audio information

userCb: the callback for application layer is to process the audio data,

for a example of sending out the audio data.

Start

```
void tl_audio_start(u8 profile, u8 pairingRef)
```

profile: profile identifier, *PROFILE_ZRC2* is as default

pairingRef: the index of the pairing, *0* is as default

Stop

```
void tl_audio_stop(u8 profile, u8 pairingRef)
```

profile: profile identifier, *PROFILE_ZRC2* is as default

pairingRef: the index of the pairing, *0* is as default

3.5.2.5 OTA

Initilization

```
void ota_initInfo(void)
```

Start

```
void ota_startReq(u8 pairingRef, ota_UserCb Cb)
```

pairingRef: the index of the pairing, *0* is as default

Cb: the callback for application layer is to notify the OTA is done

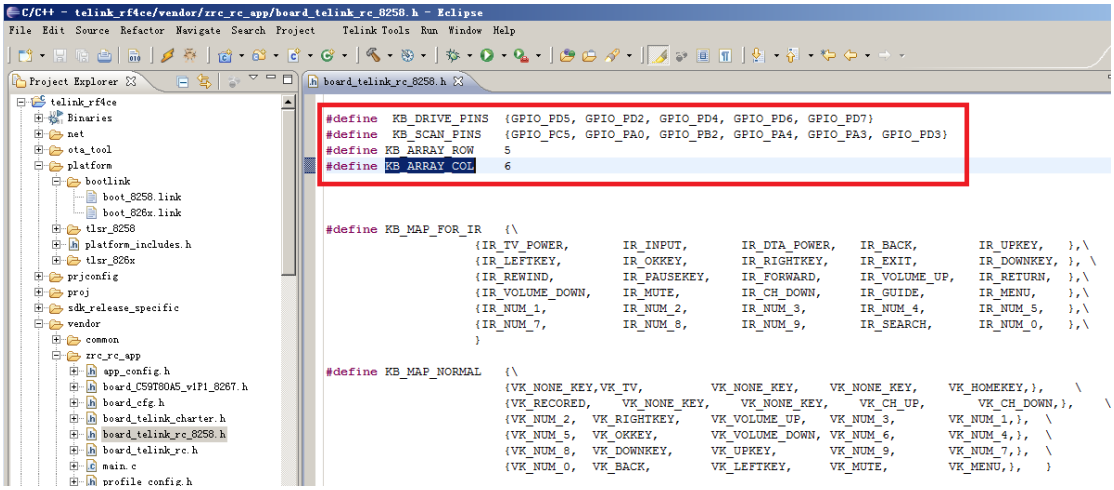
3.5.3 UI

Normally the user pressed the button to trigger operation such as paring, un-pairing, sending key code, starting OTA, and so on.

3.5.3.1 Key Config

Define the BOARD macro in board_cfg.h to define the open PCB, then configure KB_DRIVE_PINS and KB_SCAN_PINS, and modify KB_ARRAY_ROW and KB_ARRAY_COL.

Figure 3-4 Configure Key Pin and Key Mapping



3.5.3.2 Key Scan

```
u32 kb_scan_key (int numlock_status, int read_key)
```

numlock_status: number lock is supported, 0 as default

read_key: 0 as default

return: 0: no key is scanned, others: some key is scanned

4. Appendix 1 RF4CE OTA

OTA device includes RC and Target. In this section, RC and Target are taken as an example to introduce the OTA method of updating new image to End Device.

1. OTA parameter

Define current firmware version number. Please refer to the “zrcApp.c”.

```
.fileVer: Firmware. Should bigger than the current file version.
. imageType, manufaurerCode: Should be the same as the current image.
```

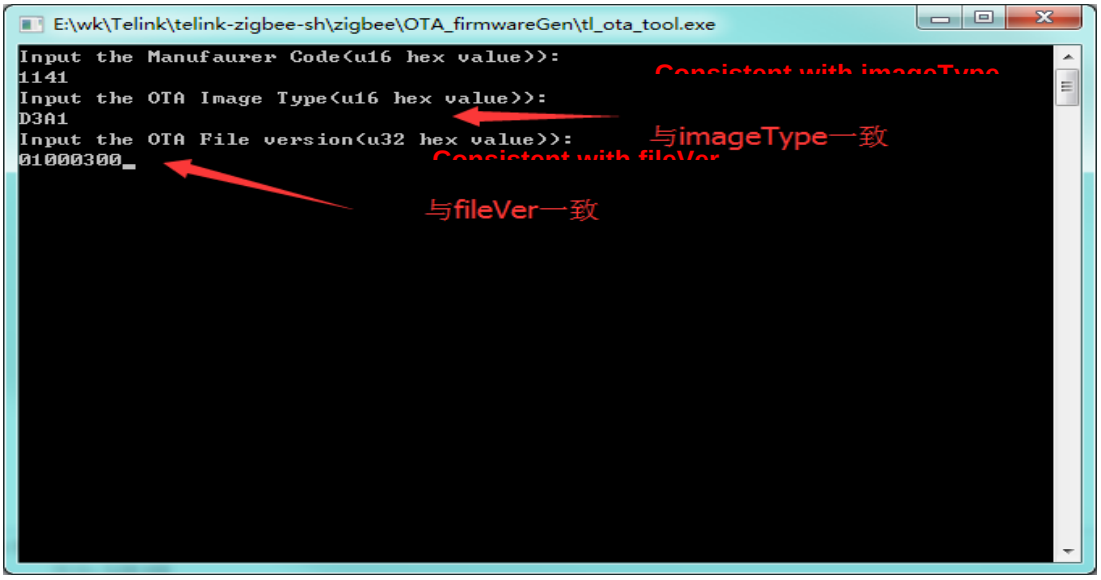
For example, suppose the “CURRENT_FILE_VERSION” is set as “0x01000104” and “0x01000105”, two firmware files “zrc2_rc_82xx_0x01000104.bin” and “zrc2_rc_82xx_0x01000105.bin” will be generated respectively.

First download the “zrc2_rc_82xx_0x01000104.bin” into the target End Device via Telink Wtcdb tool. Then use OTA method to update the “sampleSwitch_0x01000300.bin” into the End Device.

2. “Tl_ota_tool.exe” tool

Rename the “zrc2_rc_82xx_0x01000105.bin” as “zbGitVersionCode.bin”, and place it under the directory which contains the “tl_ota_tool.exe”. Run the “tl_ota_tool.exe”, input contents according to prompt information, and press the “Enter” key to confirm the input. Finally a file “ota.ota” will be generated.

Figure 4-1 “tl_ota_tool.exe” Tool Interface



3. Use the Wtcdb tool to write the “ota.ota” file into Coordinator flash address starting from 0x40000, so that it can be used for OTA.

   The command to write the “ota.ota” is shown as below:

```
.\tcdb.exe wf 40000 -b -i e:\ota.ota
```

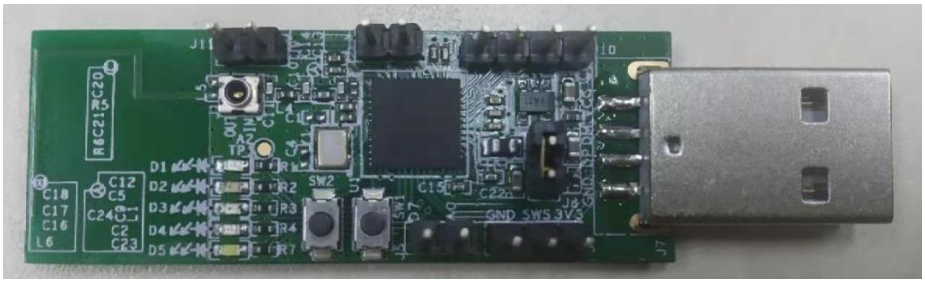
5. Appendix 2 RF4CE PAIRING

Take 8258 as an example to demonstrate the pairing process between RC and target.

Hardware:

Figure 5-1 8258 RC:C1T139A5_V1.4

Figure 5-2 8258 DONGLE:C1T139A3_V2.0A



Software

RC:zrc2_rc_8258.bin

DONGLE: zrc_dongle_8258.bin

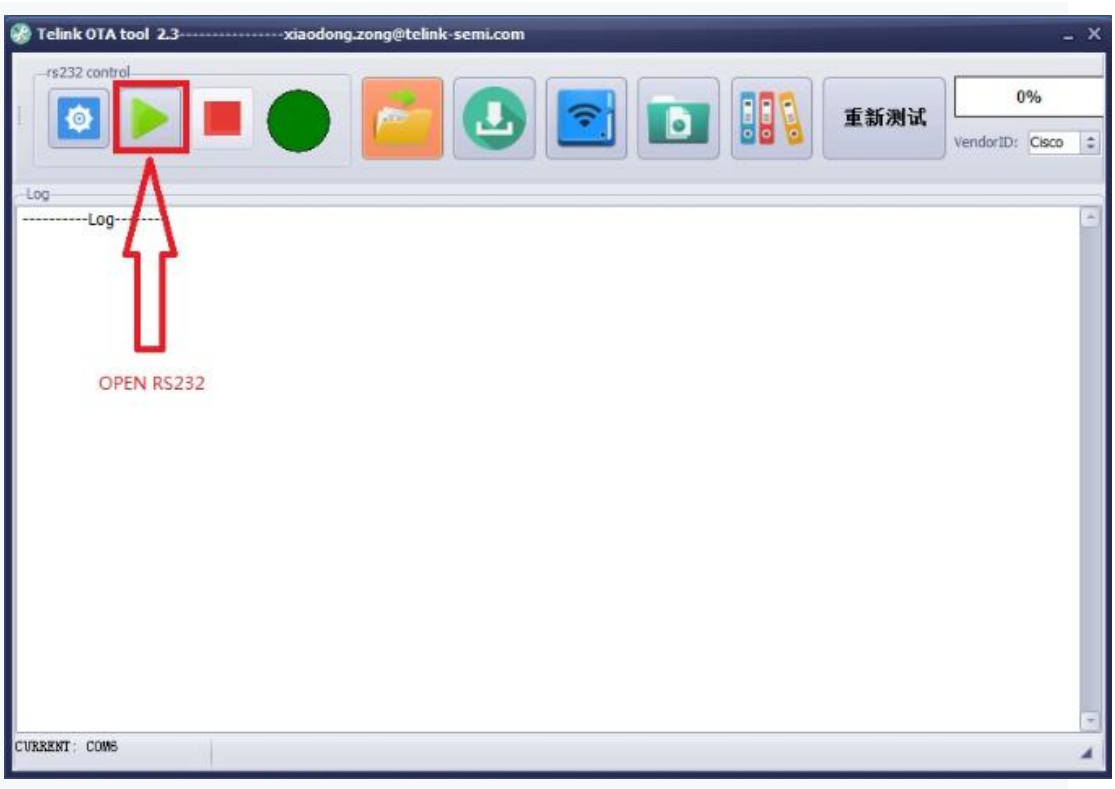
PC Tool

Telink OTA tool 2.3

Operation is as follows:

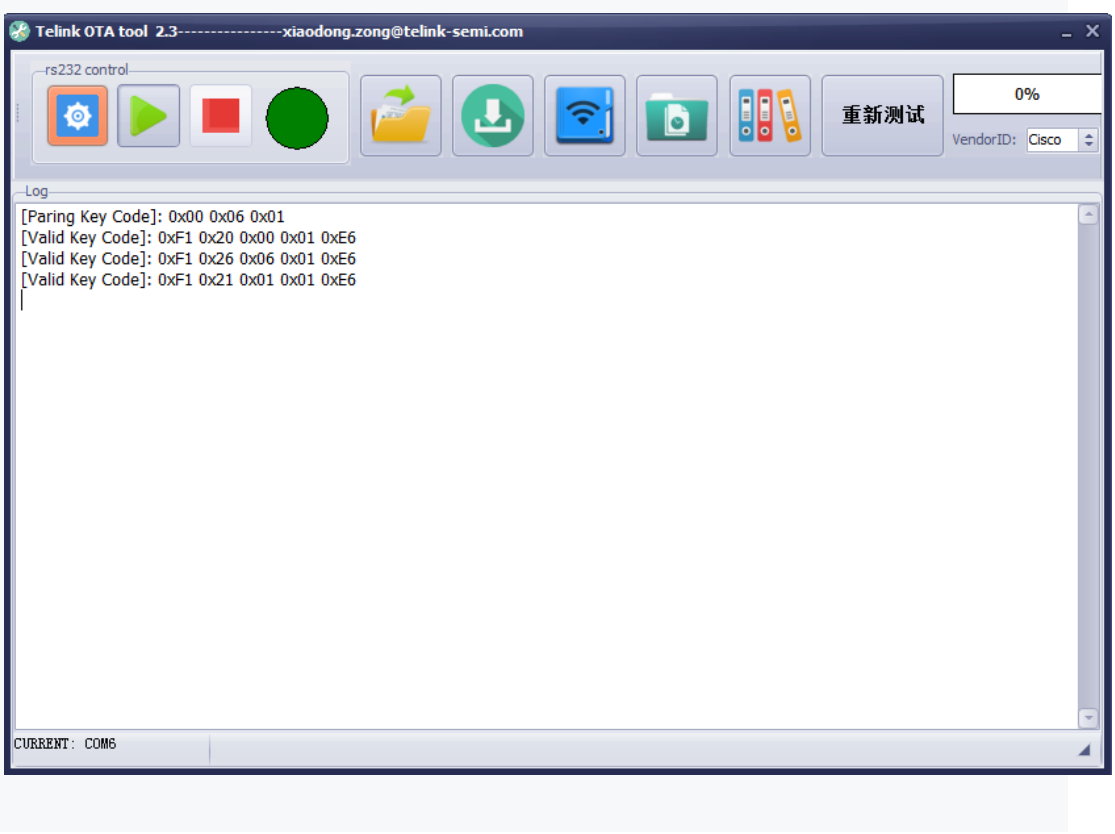
Step 1. Plug the 8258 dongle into your computer and open pc tool, then click the open rs232 button, as following picture shows:

Figure 5-3 Click the OPEN RS232 Button



Step 2. Long press the pair key(SDK default SW5) on RC, PC tool will then display three random Numbers ranging from 0 to 9, then press the corresponding number key on RC, and the key values will be displayed on PC tool.

Figure 5-4 Display and Enter Pairing Numbers



Step 3. After completing the above steps, the led on the RC will flash several times rapidly to indicate the completion of pairing

6. Appendix 3 RF4CE AUDIO TEST

After pairing, select listen for the Telink RF4CE microphone device, press the audio button to turn on the audio function, and you can hear the voice from the remote control in the speakers on the computer side.

Figure 6-1 Enable RF4CE Microphone Equipment