MT793X IOT SDK - HAL DRIVER EXAMPLES

Version: 1.0

Release date: 2022-10-27

Version History

Version	Date	Description
1.0	2022-10-27	Initial version

Table of Contents

Version History **2**
Table of Contents **3**
List of Figures **4**
1 Hal Example Project **6**
1.1 Source Code Architecture 8
2 How to Use **10**
2.1 Query Available Commands 10
2.2 Portnum Parameter 11
3 Sample Test **12**
3.1 GCPU (General Copy Protection Unit) 12
3.2 ECC (Elliptic Curve Crypto) 13
3.3 GDMA 13
3.4 ADC 14
3.5 GPT 15
3.6 WDT 16
3.7 SleepManager 16
3.8 NVDM 17
3.9 CACHE 17
3.10 MPU 18
3.11 FLASH 18
3.12 USB HOST 19
3.13 USB GADGET 20
3.14 I2C (Inter-Integrated Circuit) 21
3.15 SD 23
3.16 EINT 24
3.17 GPIO 26
3.18 NVIC 27
3.19 RTC 27
3.20 PWM 28
3.21 SDIO Master & SDIO Slave (Secure Digital Input and Output Master & Slave) 30
3.22 SPI Master & SPI Slave (Serial Peripheral Interface Master & Slave) 32
3.23 TRNG (True Random Number Generator) 34
3.24 I2S (Inter-IC Sound) 34
Exhibit 1 Terms and Conditions **36**

List of Figures

Figure 1 MT7933CT hal examples ... 7
Figure 2 MT7931AN hal examples ... 8
Figure 3 dispatch command ... 9
Figure 4 help messages ... 10
Figure 5 available examples (MT7933CT) ... 11
Figure 6 available examples (MT7931AN) ... 11
Figure 7 example of GCPU ... 12
Figure 8 example of ECC ... 13
Figure 9 example of GDMA ... 13
Figure 10 example of ADC ... 14
Figure 11 input voltage to GPIO22 ... 15
Figure 12 example of GPT ... 16
Figure 13 example of WDT ... 16
Figure 14 example of SleepManager ... 17
Figure 15 example of NVDM ... 17
Figure 16 example of CACHE ... 18
Figure 17 example of MPU ... 18
Figure 18 flash layout ... 19
Figure 19 Flash default testing range ... 19
Figure 20 example of Flash ... 19
Figure 21 insert USB disk ... 20
Figure 22 example of USB host ... 20
Figure 23 connect micro-USB cable ... 21
Figure 24 example of USB gadget ... 21
Figure 25 EEPROM location ... 22
Figure 26 short JP9 1 and 2 ... 22
Figure 27 example of I2C ... 23
Figure 28 SD Co-pad ... 24
Figure 29 short GPIO 13&14(MT7933CT) ... 25
Figure 30 GPIO 13&14 location, G means Gnd (MT7933CT) ... 25
Figure 31 short GPIO 19&20(MT7931AN) ... 26
Figure 32 example of EINT ... 26
Figure 33 example of GPIO ... 27
Figure 34 example of NVIC ... 27
Figure 35 example of RTC ... 28
Figure 36 example of PWM ... 28
Figure 37 Location for LA to measure (MT7933CT) ... 29
Figure 38 LA for checking (MT7933CT) ... 29
Figure 39 LA's result ... 30
Figure 40 SDIO testing flow ... 31
Figure 41 SDIO's limitation ... 32
Figure 42 short SPI Master & SPI Slave for testing ... 33

Figure 43 example of SPI 34
Figure 44 example of TRNG 34
Figure 45 example of I2S 35

1 Hal Example Project

This document introduces MT7931/33 IOT SDK hal driver usage examples. The example projects are under SDK_root/project/mt7933_hdk/apps/xxx. All the example projects can be executed with MTK RFB (Reference Board).

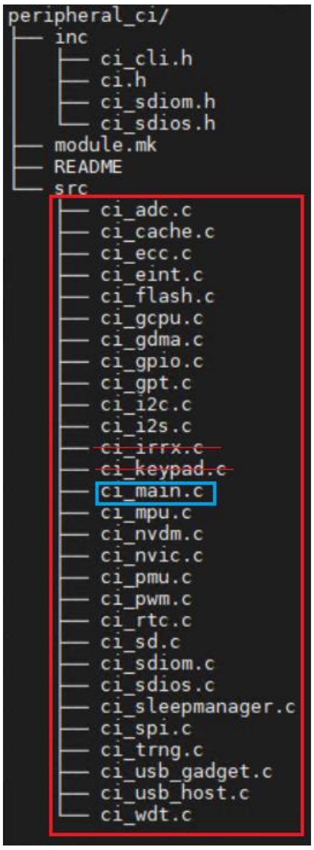
For MT7933CT RFB (Board number: MTK3294)
Project folder: SDK_root/project/mt7933_hdk/apps/hal_examples
Example source code: hal_examples/peripheral_ci

For MT7931AN RFB (Board number: MTK2997)
Project folder: SDK_root/project/mt7933_hdk/apps/qfn_hal_examples
Example source code: qfn_hal_examples/peripheral_ci

MT7931/33 IOT SDK provides example code for developer reference. There is total 26 examples code for MT7933CT and 21 for MT7931AN, the developer could refer to Figure 1 for MT7933CT RFB, Figure 2 for MT7931AN RFB.

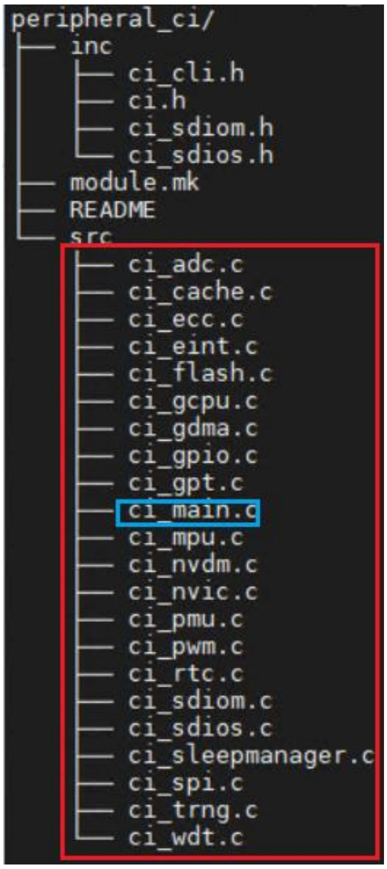
Theoretically, all examples can be executed in a signal project. However, due to the PinMux limitation, not all examples can be executed at the same time. for instance, SDIO master and SDIO slave share the same PinMux that SDIO master and slave can’t be performed at the same time.

Note: IOT SDK3.0 doesn’t support ci_irrx.c and ci_keypad.c



```
peripheral_ci/
├── inc
│   ├── ci_cli.h
│   ├── ci.h
│   ├── ci_sdiom.h
│   └── ci_sdios.h
├── module.mk
├── README
└── src
    ├── ci_adc.c
    ├── ci_cache.c
    ├── ci_ecc.c
    ├── ci_eint.c
    ├── ci_flash.c
    ├── ci_gcpu.c
    ├── ci_gdma.c
    ├── ci_gpio.c
    ├── ci_gpt.c
    ├── ci_i2c.c
    ├── ci_i2s.c
    ├── ~~ci_irrx.c~~
    ├── ~~ci_keypad.c~~
    ├── ci_main.c
    ├── ci_mpu.c
    ├── ci_nvdm.c
    ├── ci_nvic.c
    ├── ci_pmu.c
    ├── ci_pwm.c
    ├── ci_rtc.c
    ├── ci_sd.c
    ├── ci_sdiom.c
    ├── ci_sdios.c
    ├── ci_sleepmanager.c
    ├── ci_spi.c
    ├── ci_trng.c
    ├── ci_usb_gadget.c
    ├── ci_usb_host.c
    └── ci_wdt.c
```

Figure 1 MT7933CT hal examples



```
peripheral_ci/
├── inc
│   ├── ci_cli.h
│   ├── ci.h
│   ├── ci_sdiom.h
│   └── ci_sdios.h
├── module.mk
├── README
└── src
    ├── ci_adc.c
    ├── ci_cache.c
    ├── ci_ecc.c
    ├── ci_eint.c
    ├── ci_flash.c
    ├── ci_gcpu.c
    ├── ci_gdma.c
    ├── ci_gpio.c
    ├── ci_gpt.c
    ├── ci_main.c
    ├── ci_mpu.c
    ├── ci_nvdm.c
    ├── ci_nvic.c
    ├── ci_pmu.c
    ├── ci_pwm.c
    ├── ci_rtc.c
    ├── ci_sdiom.c
    ├── ci_sdios.c
    ├── ci_sleepmanager.c
    ├── ci_spi.c
    ├── ci_trng.c
    └── ci_wdt.c
```

Figure 2 MT7931AN hal examples

1.1 Source Code Architecture

The developers could refer the source code in the following path, such as

peripheral_ci: hal examples folder

peripheral_ci/inc: includes common macro and some SDIO macro

README: some information about example code

peripheral_ci/module.mk: makefile for peripheral_ci

peripheral_ci/src: hal examples source code

peripheral_ci/src/ci_main.c: all hal examples entry which is used to dispatch command to related c file. (Please refer to Figure 3)

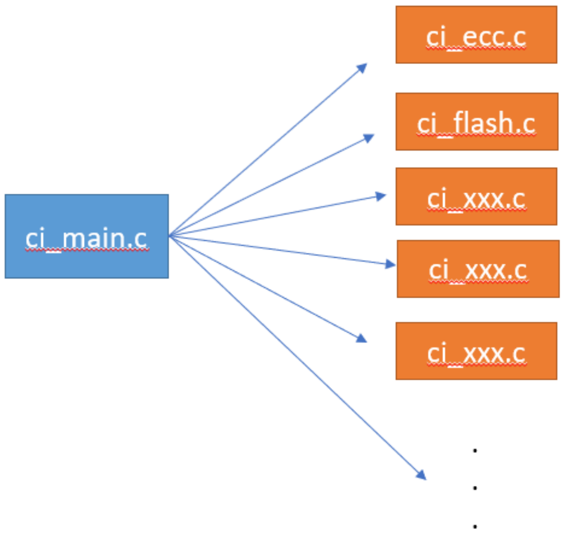
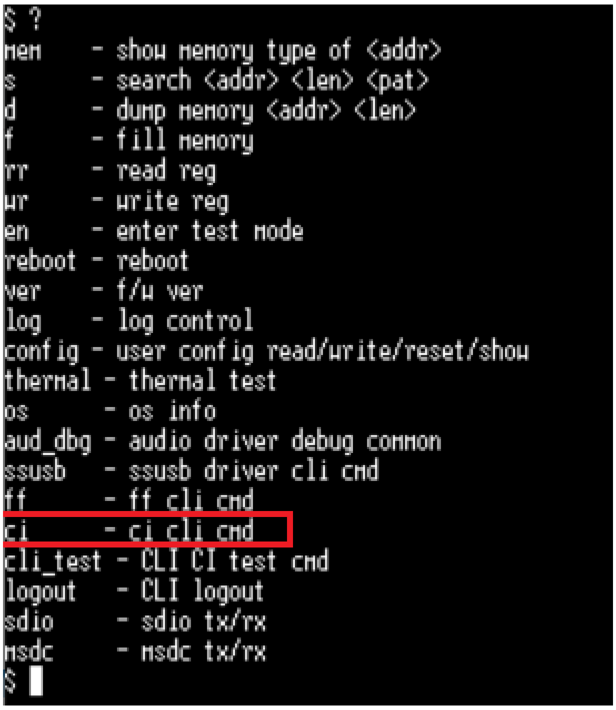


Figure 3 dispatch command

2 How to Use

2.1 Query Available Commands

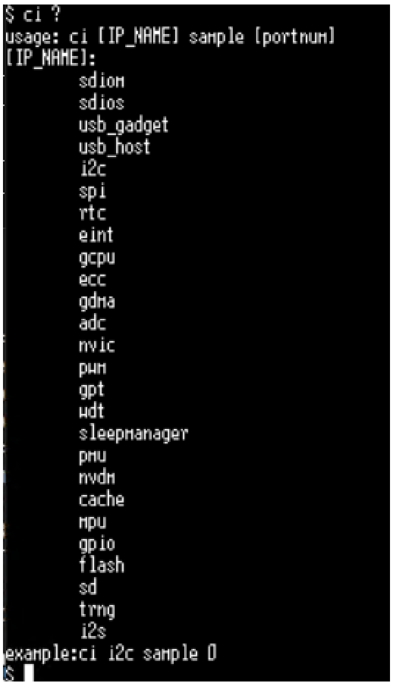
The developers can use the question mark to list the supported commands at current command line level. The hal examples are located in “ci” level, please refer to Figure 4 as below, marked with red frame



```
$ ?
mem    - show memory type of <addr>
s      - search <addr> <len> <pat>
d      - dump memory <addr> <len>
f      - fill memory
rr     - read reg
wr     - write reg
en     - enter test mode
reboot - reboot
ver    - f/w ver
log    - log control
config - user config read/write/reset/show
thermal - thermal test
os       - os info
aud_dbg - audio driver debug common
ssusb   - ssusb driver cli cmd
ff       - ff cli cmd
ci       - ci cli cmd
cli_test - CLI CI test cmd
logout   - CLI logout
sdio     - sdio tx/rx
msdc     - msdc tx/rx
$ 
```

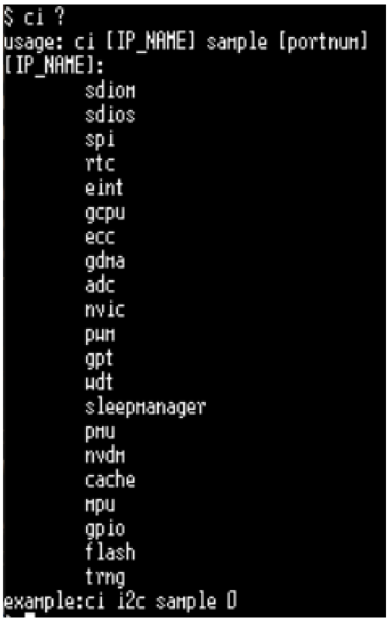
Figure 4 help messages

The developers can use the command “ci ?” to list the message of supported hal examples. The hal examples are located in “ci” level, please refer to Figure 5 and Figure 6



```
$ ci ?
usage: ci [IP_NAME] sample [portnum]
[IP_NAME]:
        sdiom
        sdios
        usb_gadget
        usb_host
        i2c
        spi
        rtc
        eint
        gcpu
        ecc
        gdma
        adc
        nvic
        pwm
        gpt
        wdt
        sleepmanager
        pmu
        nvdm
        cache
        mpu
        gpio
        flash
        sd
        trng
        i2s
example:ci i2c sample 0
$
```

Figure 5 **available examples (MT7933CT)**



```
$ ci ?
usage: ci [IP_NAME] sample [portnum]
[IP_NAME]:
        sdiom
        sdios
        spi
        rtc
        eint
        gcpu
        ecc
        gdma
        adc
        nvic
        pwm
        gpt
        wdt
        sleepmanager
        pmu
        nvdm
        cache
        mpu
        gpio
        flash
        trng
example:ci i2c sample 0
```

Figure 6 **available examples (MT7931AN)**

2.2 Portnum Parameter

The argument of “portnum” isn’t used in these examples, please fill in with zero to avoid unexpected errors.

3 Sample Test

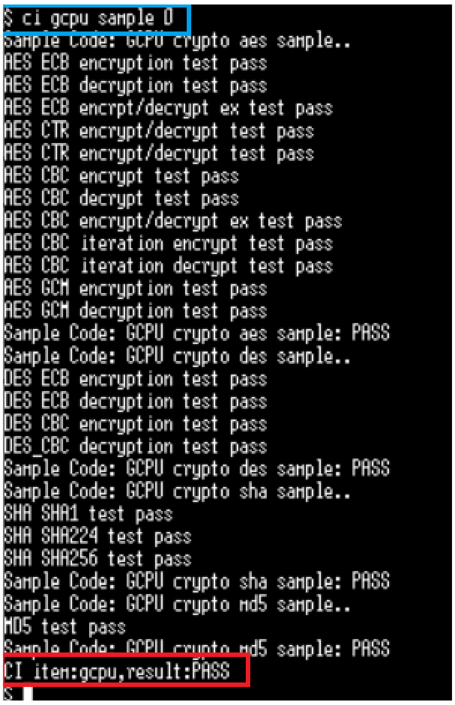
The developers could refer to “MT7933_RFB_Users_Guide 4.6 Extension connectors” and “MT7931_RFB_Users_Guide 4.7 Extension connectors” in MTK DCC system for peripherals which require PinMux settings. This chapter describes the detail about how to configurate PinMux (multi-function) for peripherals.

Source code location:
Project/mt7933_hdk/apps/hal_examples/peripheral_ci/src
Project/mt7933_hdk/apps/qfn_hal_examples/peripheral_ci/src

3.1 GCPU (General Copy Protection Unit)

The main function of this module is to provide various copy protection algorithms, such as CPPM, CPRM, AES/AES-CMAC/AES-XCBC-MAC, 3DES/DES, SHA-1/SHA-224/SHA-256, MD5, RSA, TRNG and etc.

Need PinMux configuration	No
Need rework board	No
Source code	ci_gcpu.c
Support RFB type	MT7933CT & MT7931AN



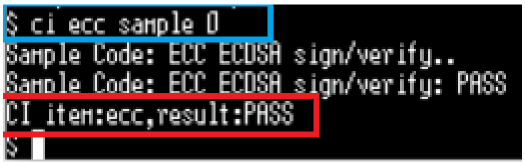
```
$ ci gcpu sample 0
Sample Code: GCPU crypto aes sample..
AES ECB encryption test pass
AES ECB decryption test pass
AES ECB encrpt/decrypt ex test pass
AES CTR encrypt/decrypt test pass
AES CTR encrypt/decrypt test pass
AES CBC encrypt test pass
AES CBC decrypt test pass
AES CBC encrypt/decrypt ex test pass
AES CBC iteration encrypt test pass
AES CBC iteration decrypt test pass
AES GCM encryption test pass
AES GCM decryption test pass
Sample Code: GCPU crypto aes sample: PASS
Sample Code: GCPU crypto des sample..
DES ECB encryption test pass
DES ECB decryption test pass
DES CBC encryption test pass
DES_CBC decryption test pass
Sample Code: GCPU crypto des sample: PASS
Sample Code: GCPU crypto sha sample..
SHA SHA1 test pass
SHA SHA224 test pass
SHA SHA256 test pass
Sample Code: GCPU crypto sha sample: PASS
Sample Code: GCPU crypto md5 sample..
MD5 test pass
Sample Code: GCPU crypto md5 sample: PASS
CI item:gcpu,result:PASS
$
```

Figure 7 example of GCPU

3.2 ECC (Elliptic Curve Crypto)

The main function of this module is to provide ECDSA hardware crypto.

Need PinMux configuration	No
Need rework board	No
Source code	ci_ecc.c
Support RFB type	MT7933CT & MT7931AN



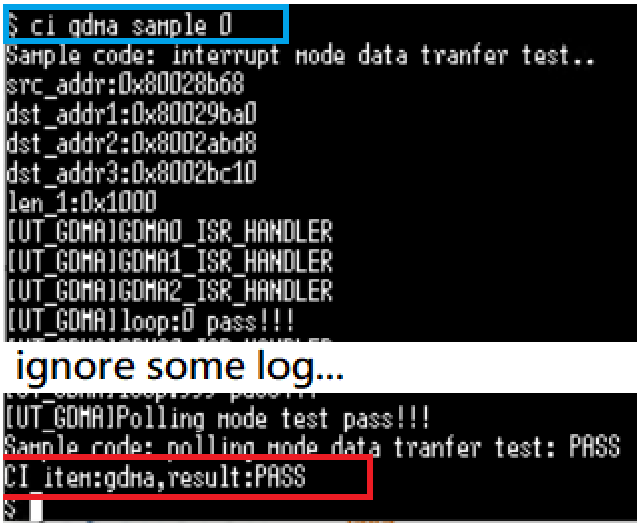
```
$ ci ecc sample 0
Sample Code: ECC ECDSA sign/verify..
Sample Code: ECC ECDSA sign/verify: PASS
CI_item:ecc,result:PASS
$
```

Figure 8 **example of ECC**

3.3 GDMA

The main function of this module is to provide hardware direct memory to memory access.

Need PinMux configuration	No
Need rework board	No
Source code	ci_gdma.c
Support RFB type	MT7933CT & MT7931AN



```
$ ci gdma sample 0
Sample code: interrupt mode data tranfer test..
src_addr:0x80028b68
dst_addr1:0x80029ba0
dst_addr2:0x8002abd8
dst_addr3:0x8002bc10
len_1:0x1000
[UT_GDMA]GDMA0_ISR_HANDLER
[UT_GDMA]GDMA1_ISR_HANDLER
[UT_GDMA]GDMA2_ISR_HANDLER
[UT_GDMA]loop:0 pass!!!
```

ignore some log...

```
[UT_GDMA]Polling mode test pass!!!
Sample code: polling mode data tranfer test: PASS
CI_item:gdma,result:PASS
$
```

Figure 9 **example of GDMA**

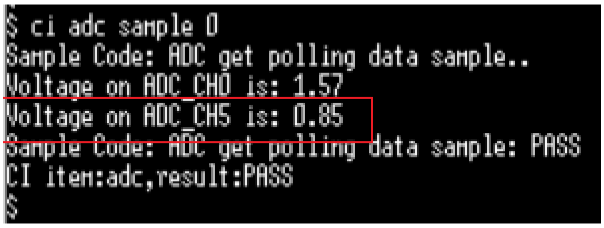
3.4 ADC

The main function of this module is to provide analog to digital converter.

Need PinMux configuration	Yes (GPIO17 or GPIO22)
Need rework board	No
Source code	ci_adc.c
Support RFB type	MT7933CT & MT7931AN

Note:

- Input voltage up to 1.8V to GPIO17 or GPIO 22 for testing
- The following example (Figure 10) is that input 0.85V to GPIO22(Figure 11), but ADC_CH0(GPIO17) doesn't input any voltage, please ignore the meaningless number (1.57)



```
$ ci adc sample 0
Sample Code: ADC get polling data sample..
Voltage on ADC_CH0 is: 1.57
Voltage on ADC_CH5 is: 0.85
Sample Code: ADC get polling data sample: PASS
CI item:adc,result:PASS
$
```

Figure 10 example of ADC

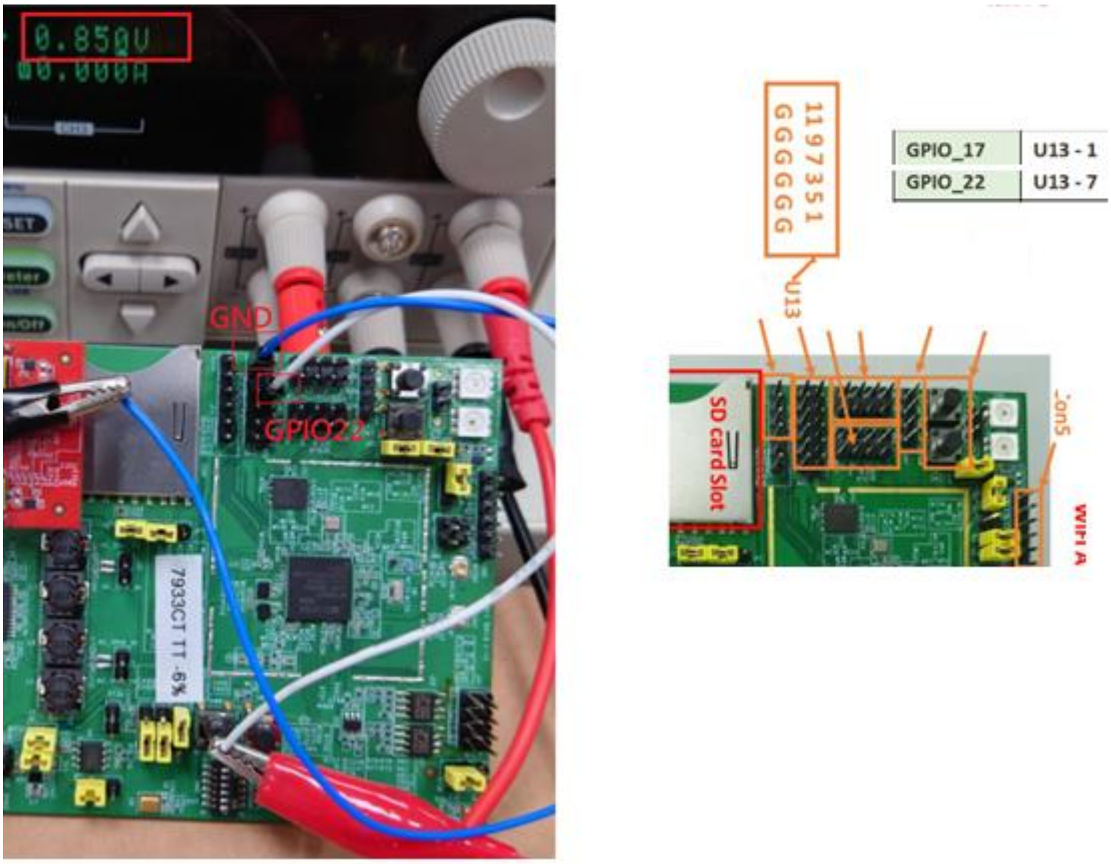
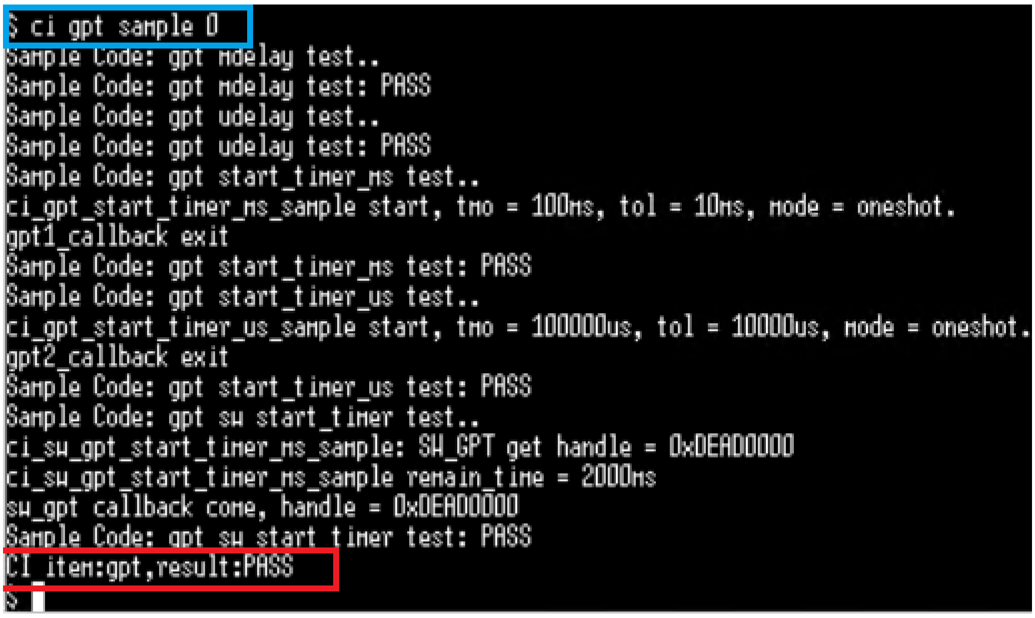


Figure 11 **input voltage to GPIO22**

3.5 GPT

The main function of this module is to provide a general-purpose timer.

Need PinMux configuration	No
Need rework board	No
Source code	ci_gpt.c
Support RFB type	MT7933CT & MT7931AN



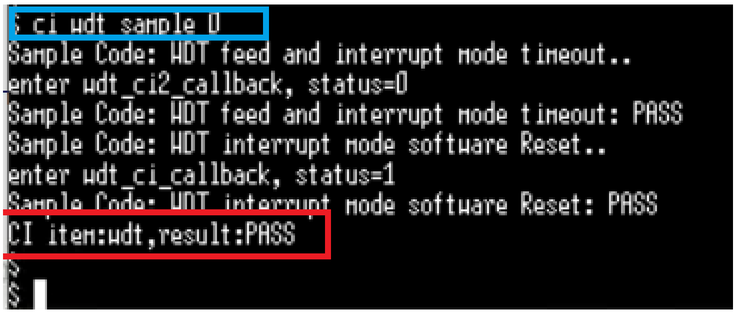
```
$ ci gpt sample 0
Sample Code: gpt mdelay test..
Sample Code: gpt mdelay test: PASS
Sample Code: gpt udelay test..
Sample Code: gpt udelay test: PASS
Sample Code: gpt start_timer_ms test..
ci_gpt_start_timer_ms_sample start, tmo = 100ms, tol = 10ms, mode = oneshot.
gpt1_callback exit
Sample Code: gpt start_timer_ms test: PASS
Sample Code: gpt start_timer_us test..
ci_gpt_start_timer_us_sample start, tmo = 100000us, tol = 10000us, mode = oneshot.
gpt2_callback exit
Sample Code: gpt start_timer_us test: PASS
Sample Code: gpt sw start_timer test..
ci_sw_gpt_start_timer_ms_sample: SW_GPT get handle = 0xDEAD0000
ci_sw_gpt_start_timer_ms_sample remain_time = 2000ms
sw_gpt callback come, handle = 0xDEAD0000
Sample Code: gpt sw start_timer test: PASS
CI_item:gpt,result:PASS
$
```

Figure 12 example of GPT

3.6 WDT

The main function of this module is to provide a watch dog reset function.

Need PinMux configuration	No
Need rework board	No
Source code	ci_wdt.c
Support RFB type	MT7933CT & MT7931AN



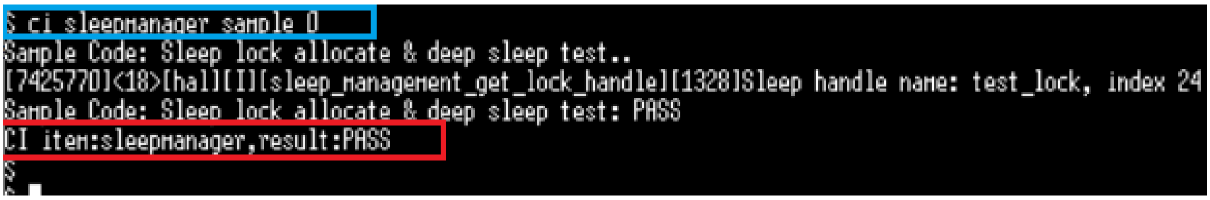
```
$ ci wdt sample 0
Sample Code: WDT feed and interrupt mode timeout..
enter wdt_ci2_callback, status=0
Sample Code: WDT feed and interrupt mode timeout: PASS
Sample Code: WDT interrupt mode software Reset..
enter wdt_ci_callback, status=1
Sample Code: WDT interrupt mode software Reset: PASS
CI item:wdt,result:PASS
$
$
```

Figure 13 example of WDT

3.7 SleepManager

The main function of this module is to provide a test of deep sleep mode.

Need PinMux configuration	No
Need rework board	No
Source code	ci_sleepmanager.c
Support RFB type	MT7933CT & MT7931AN



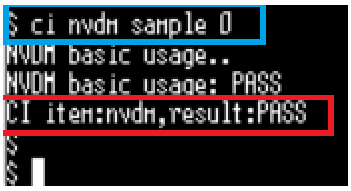
```
$ ci sleepmanager sample 0
Sample Code: Sleep lock allocate & deep sleep test..
[7425770]<18>[hal][I][sleep_management_get_lock_handle][1328]Sleep handle name: test_lock, index 24
Sample Code: Sleep lock allocate & deep sleep test: PASS
CI item:sleepmanager,result:PASS
$
$
```

Figure 14 example of SleepManager

3.8 NVDM

The main function of this module is to provide a test of non-volatile dynamic memory.

Need PinMux configuration	No
Need rework board	No
Source code	ci_nvdm.c
Support RFB type	MT7933CT & MT7931AN



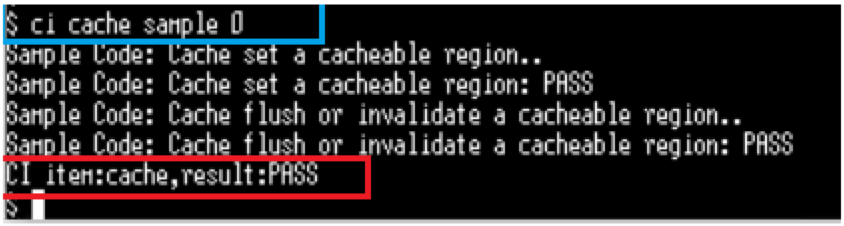
```
$ ci nvdm sample 0
NVDM basic usage..
NVDM basic usage: PASS
CI item:nvdm,result:PASS
$
$
```

Figure 15 example of NVDM

3.9 CACHE

The main function of this module is to provide a test of CM33's cache.

Need PinMux configuration	No
Need rework board	No
Source code	ci_cache.c
Support RFB type	MT7933CT & MT7931AN



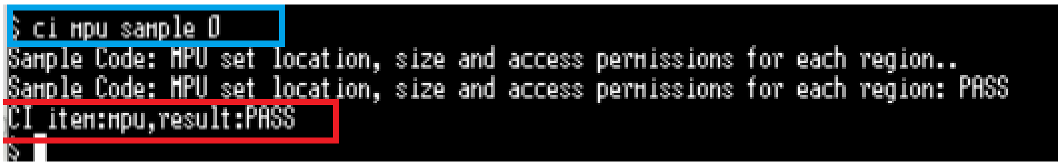
```
$ ci cache sample 0
Sample Code: Cache set a cacheable region..
Sample Code: Cache set a cacheable region: PASS
Sample Code: Cache flush or invalidate a cacheable region..
Sample Code: Cache flush or invalidate a cacheable region: PASS
CI item:cache,result:PASS
$
```

Figure 16 example of CACHE

3.10 MPU

The main function of this module is to provide a test of ARM CM33 memory protection unit.

Need PinMux configuration	No
Need rework board	No
Source code	ci_mpu.c
Support RFB type	MT7933CT & MT7931AN



```
$ ci mpu sample 0
Sample Code: MPU set location, size and access permissions for each region..
Sample Code: MPU set location, size and access permissions for each region: PASS
CI item:mpu,result:PASS
$
```

Figure 17 example of MPU

3.11 FLASH

The main function of this module is to provide a test of flash operation, including read, write and erase.

Need PinMux configuration	No
Need rework board	No
Source code	ci_flash.c
Support RFB type	MT7933CT & MT7931AN

Note:

- 0xC00000 ~ 0xC01000 is used for erase, write and read test.
- If this tested flash range is designed for the system, such as XiP section, FW image section. The developers have to revise an unused address for the testing or the data in this area will be modified. (refer to Figure 19)

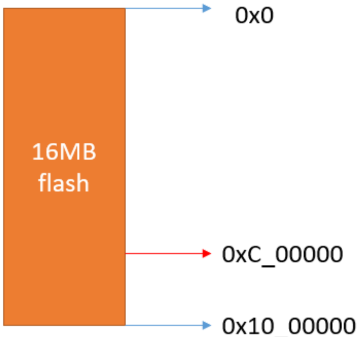
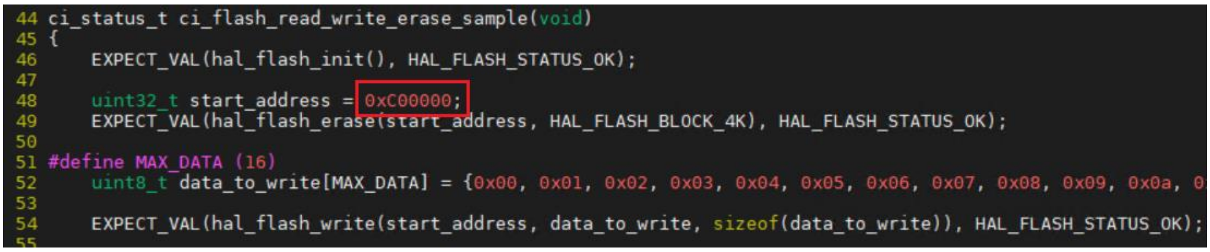
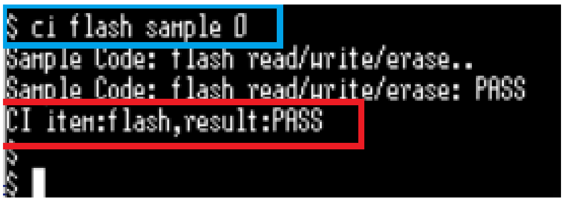


Figure 18 flash layout



```
ci_status_t ci_flash_read_write_erase_sample(void)
{
    EXPECT_VAL(hal_flash_init(), HAL_FLASH_STATUS_OK);

    uint32_t start_address = 0xC00000;
    EXPECT_VAL(hal_flash_erase(start_address, HAL_FLASH_BLOCK_4K), HAL_FLASH_STATUS_OK);

#define MAX_DATA (16)
    uint8_t data_to_write[MAX_DATA] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0

    EXPECT_VAL(hal_flash_write(start_address, data_to_write, sizeof(data_to_write)), HAL_FLASH_STATUS_OK);
```

Figure 19 Flash default testing range



```
$ ci flash sample 0
Sample Code: flash read/write/erase..
Sample Code: flash read/write/erase: PASS
CI item:flash,result:PASS
$
$ 
```

Figure 20 example of Flash

3.12 USB HOST

The main function of this module is to provide a test of usb host.

Need PinMux configuration	Yes (GPIO31 to GPIO34)
Need rework board	No
Source code	ci_usb_host.c
Support RFB type	MT7933CT

Note:

- First of all, please insert a USB disk to RFB board (refer to Figure 21)

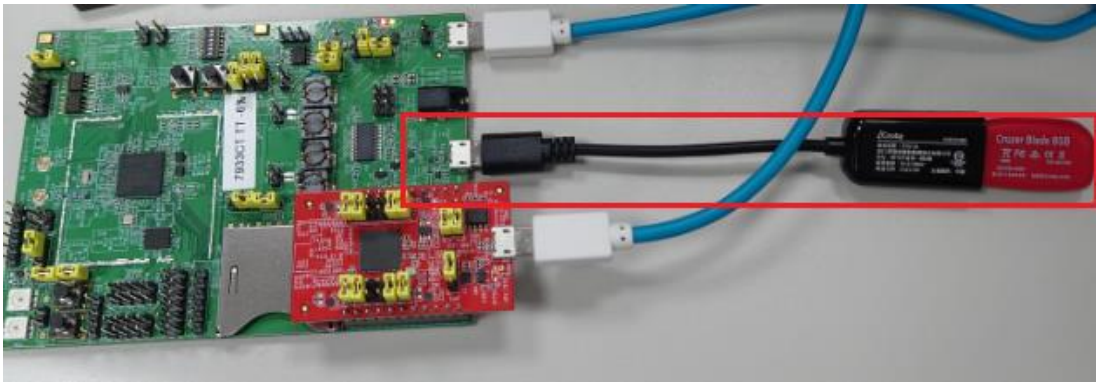
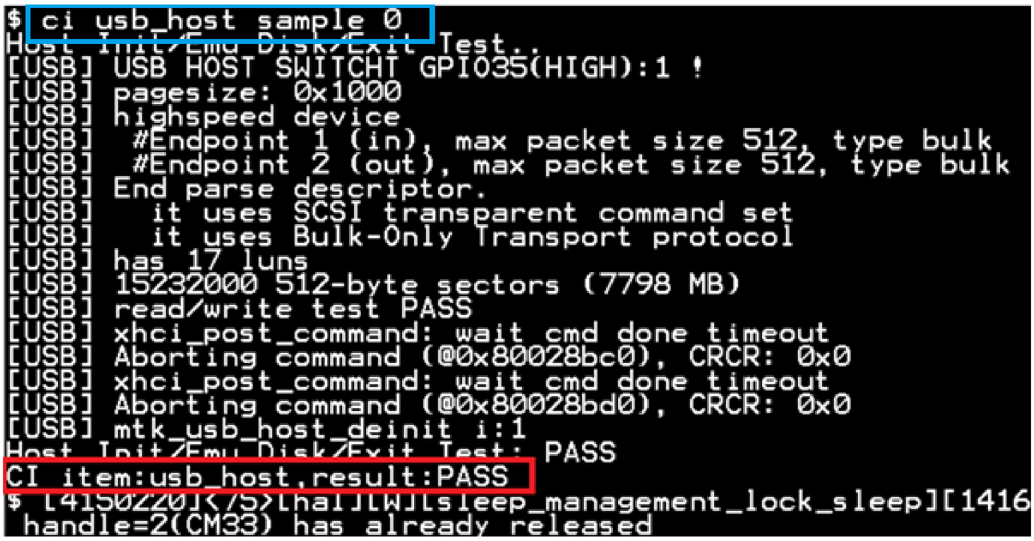


Figure 21 insert USB disk



```
$ ci usb_host sample 0
Host Init/Emu Disk/Exit Test..
[USB] USB HOST SWITCHT GPIO35(HIGH):1 !
[USB] pagesize: 0x1000
[USB] highspeed device
[USB]  #Endpoint 1 (in), max packet size 512, type bulk
[USB]  #Endpoint 2 (out), max packet size 512, type bulk
[USB] End parse descriptor.
[USB]   it uses SCSI transparent command set
[USB]   it uses Bulk-Only Transport protocol
[USB] has 17 luns
[USB] 15232000 512-byte sectors (7798 MB)
[USB] read/write test PASS
[USB] xhci_post_command: wait cmd done timeout
[USB] Aborting command (@0x80028bc0), CRCR: 0x0
[USB] xhci_post_command: wait cmd done timeout
[USB] Aborting command (@0x80028bd0), CRCR: 0x0
[USB] mtk_usb_host_deinit i:1
Host Init/Emu Disk/Exit Test: PASS
CI item:usb_host,result:PASS
$ [4150220]<75>[hal][W][sleep_management_lock_sleep][1416
 handle=2(CM33) has already released
```

Figure 22 example of USB host

3.13 USB GADGET

The main function of this module is to provide a test of usb gadget.

Need PinMux configuration	Yes (GPIO31 to GPIO34)
Need rework board	No
Source code	ci_usb_gadget.c
Support RFB type	MT7933CT

Note:

- The windows PC needs to install MTK USB driver (The developers can get the driver in MTK DCC system, it's included in FlashTool package).
- Connect the RFB board to the windows PC by using a micro-USB cable. (Refer to Figure 23)

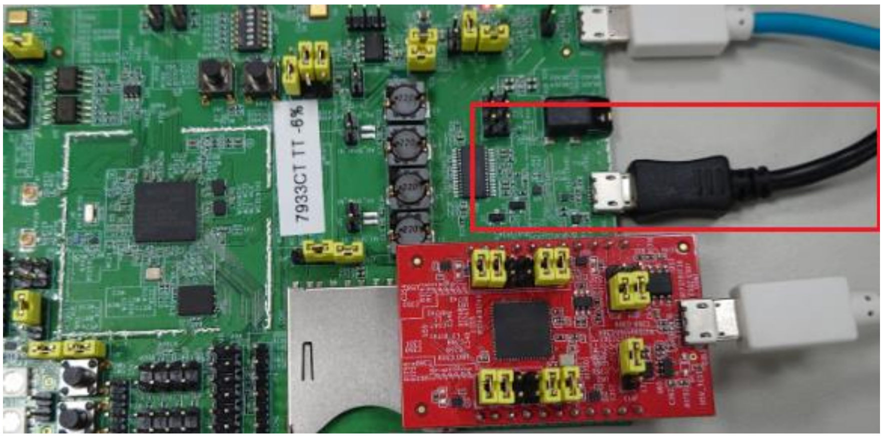
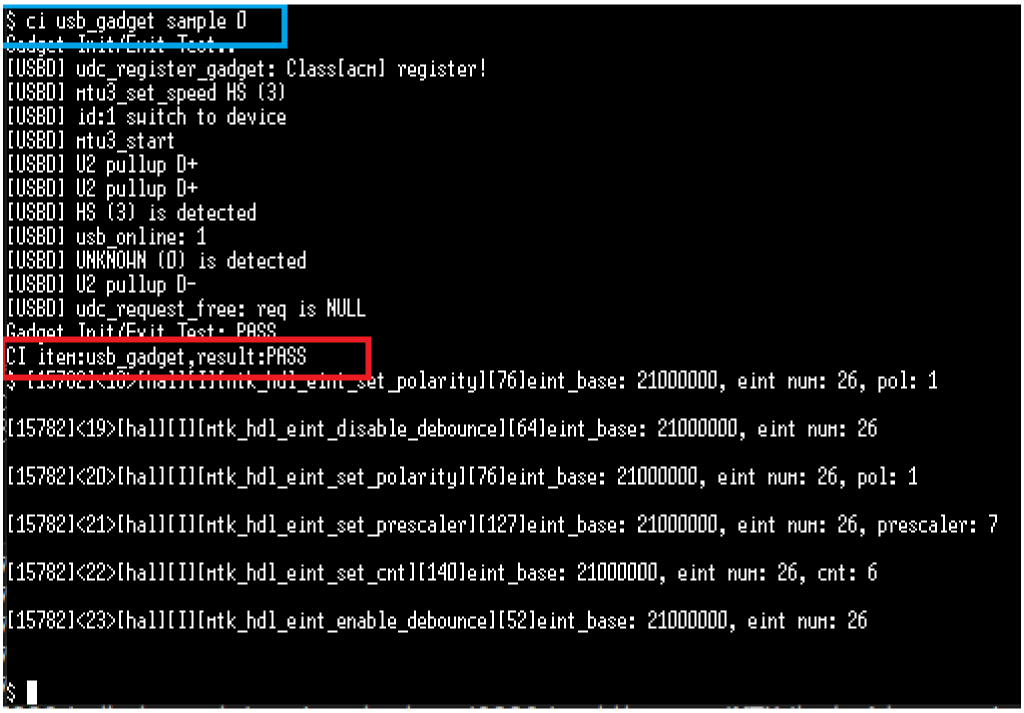


Figure 23 connect micro-USB cable



```
$ ci usb_gadget sample 0
Gadget Init/Exit Test...
[USBD] udc_register_gadget: Class[acm] register!
[USBD] mtu3_set_speed HS (3)
[USBD] id:1 switch to device
[USBD] mtu3_start
[USBD] U2 pullup D+
[USBD] U2 pullup D+
[USBD] HS (3) is detected
[USBD] usb_online: 1
[USBD] UNKNOWN (0) is detected
[USBD] U2 pullup D-
[USBD] udc_request_free: req is NULL
Gadget Init/Exit Test: PASS
CI item:usb_gadget,result:PASS
$ [15782]<18>[hal][I][mtk_hdl_eint_set_polarity][76]eint_base: 21000000, eint num: 26, pol: 1

[15782]<19>[hal][I][mtk_hdl_eint_disable_debounce][64]eint_base: 21000000, eint num: 26

[15782]<20>[hal][I][mtk_hdl_eint_set_polarity][76]eint_base: 21000000, eint num: 26, pol: 1

[15782]<21>[hal][I][mtk_hdl_eint_set_prescaler][127]eint_base: 21000000, eint num: 26, prescaler: 7

[15782]<22>[hal][I][mtk_hdl_eint_set_cnt][140]eint_base: 21000000, eint num: 26, cnt: 6

[15782]<23>[hal][I][mtk_hdl_eint_enable_debounce][52]eint_base: 21000000, eint num: 26


$
```

Figure 24 example of USB gadget

3.14 I2C (Inter-Integrated Circuit)

The main function of this module is to provide a test of I2C.

Need PinMux configuration	Yes (GPIO41&43 forI2C0, GPIO45&46 for I2C1)
Need rework board	No
Source code	ci_i2c.c

Support RFB type	MT7933CT

Note:

- Two EEPROMs (AT24C128C-SSHM) are required for I2C testing. One is for I2C0, the other is for I2C1. (Refer to Figure 25)
- Please make sure JP9 connect 1 with 2 (refer to Figure 26)

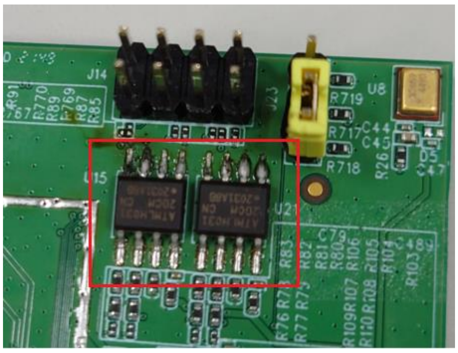


Figure 25 EEPROM location

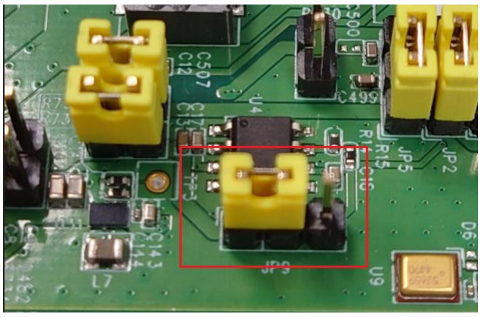
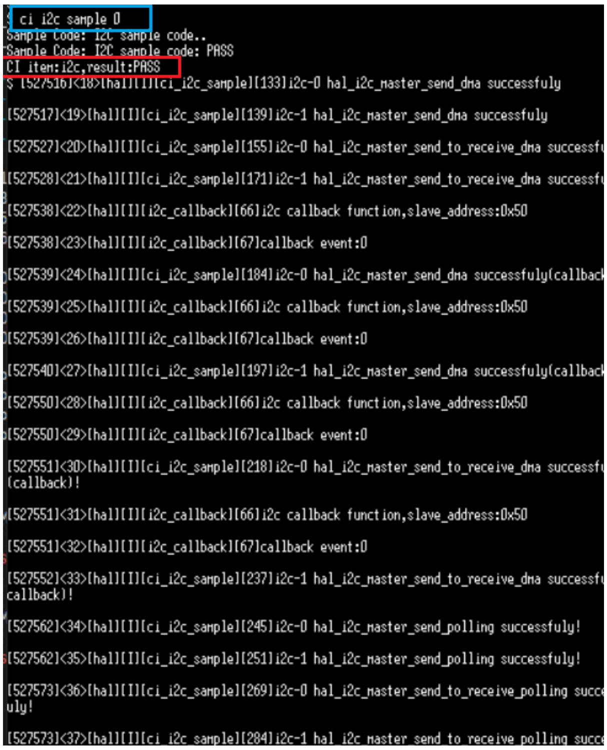


Figure 26 short JP9 1 and 2



```
$ ci i2c sample 0
Sample Code: I2C sample code..
Sample Code: I2C sample code: PASS
CI item:i2c,result:PASS
$ [527516]<18>[hal][I][ci_i2c_sample][133]i2c-0 hal_i2c_master_send_dma successfuly

[527517]<19>[hal][I][ci_i2c_sample][139]i2c-1 hal_i2c_master_send_dma successfuly

[527527]<20>[hal][I][ci_i2c_sample][155]i2c-0 hal_i2c_master_send_to_receive_dma successfu

[527528]<21>[hal][I][ci_i2c_sample][171]i2c-1 hal_i2c_master_send_to_receive_dma successfu

[527538]<22>[hal][I][i2c_callback][66]i2c callback function,slave_address:0x50

[527538]<23>[hal][I][i2c_callback][67]callback event:0

[527539]<24>[hal][I][ci_i2c_sample][184]i2c-0 hal_i2c_master_send_dma successfuly(callback

[527539]<25>[hal][I][i2c_callback][66]i2c callback function,slave_address:0x50

[527539]<26>[hal][I][i2c_callback][67]callback event:0

[527540]<27>[hal][I][ci_i2c_sample][197]i2c-1 hal_i2c_master_send_dma successfuly(callback

[527550]<28>[hal][I][i2c_callback][66]i2c callback function,slave_address:0x50

[527550]<29>[hal][I][i2c_callback][67]callback event:0

[527551]<30>[hal][I][ci_i2c_sample][218]i2c-0 hal_i2c_master_send_to_receive_dma successfu
(callback)!

[527551]<31>[hal][I][i2c_callback][66]i2c callback function,slave_address:0x50

[527551]<32>[hal][I][i2c_callback][67]callback event:0

[527552]<33>[hal][I][ci_i2c_sample][237]i2c-1 hal_i2c_master_send_to_receive_dma successfu
callback)!

[527562]<34>[hal][I][ci_i2c_sample][245]i2c-0 hal_i2c_master_send_polling successfuly!

[527562]<35>[hal][I][ci_i2c_sample][251]i2c-1 hal_i2c_master_send_polling successfuly!

[527573]<36>[hal][I][ci_i2c_sample][269]i2c-0 hal_i2c_master_send_to_receive_polling succe
uly!

[527573]<37>[hal][I][ci_i2c_sample][284]i2c-1 hal_i2c_master_send_to_receive_polling succe
```

Figure 27 example of I2C

3.15 SD

The main function of this module is to provide a test of secure digital memory card.

Need PinMux configuration	Yes (GPIO6~12)
Need rework board	Yes (refer to Note below)
Source code	ci_sd.c
Support RFB type	MT7933CT

Note:

- All six R/0/ohm/0402 need to be resoldered from SDIO side to MSDC side (Figure 28), except C_GPIO_B_0/MSDC0_RST this Co-Pad need keep at R756 side

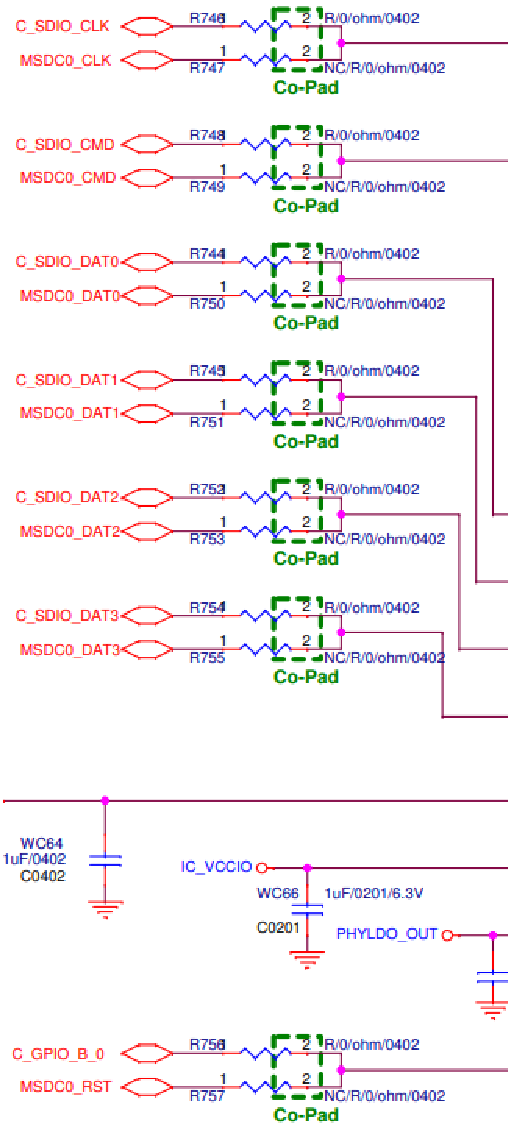


Figure 28 SD Co-pad

3.16 EINT

The main function of this module is to provide a test of external interrupt controller.

Need PinMux configuration	Yes (GPIO13&14 for MT7933CT) (GPIO19&20 for MT7931AN)
Need rework board	No
Source code	ci_eint.c
Support RFB type	MT7933CT & MT7931AN

Note:

- Need to connect GPIO13&14 for testing for MT7933CT board (please refer to Figure 29 and Figure 30)
- Need to connect GPIO19&20 for testing for MT7931AN board (please refer to Figure 31)

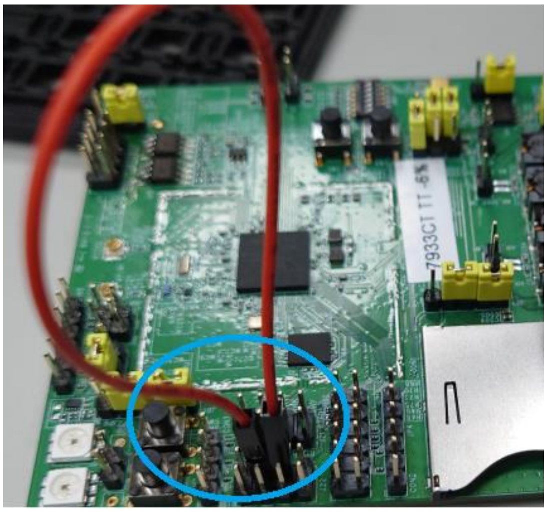
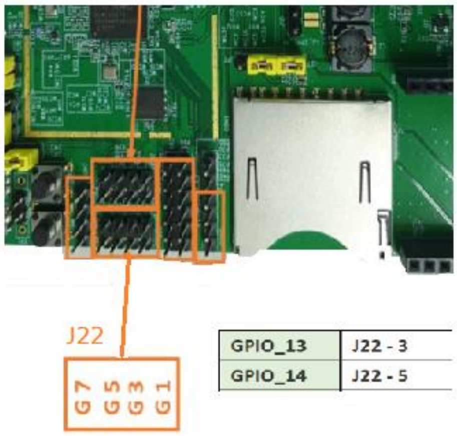


Figure 29 short GPIO 13&14(MT7933CT)



GPIO_13	J22 - 3
GPIO_14	J22 - 5

Figure 30 GPIO 13&14 location, **G means Gnd (MT7933CT)**

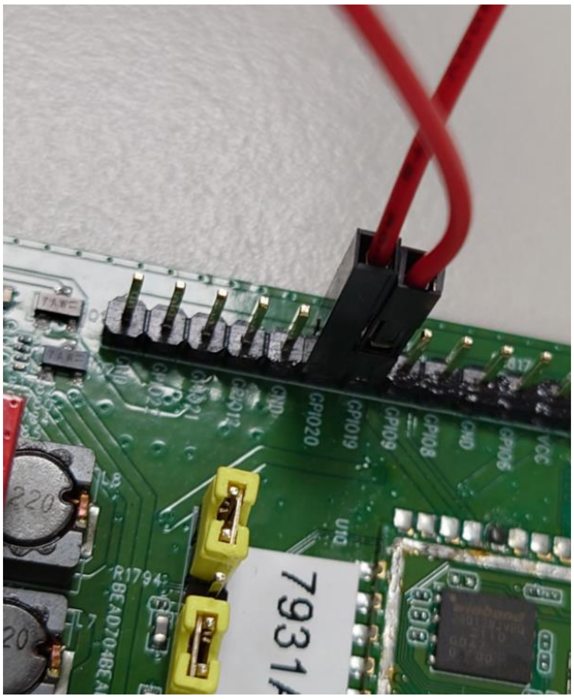
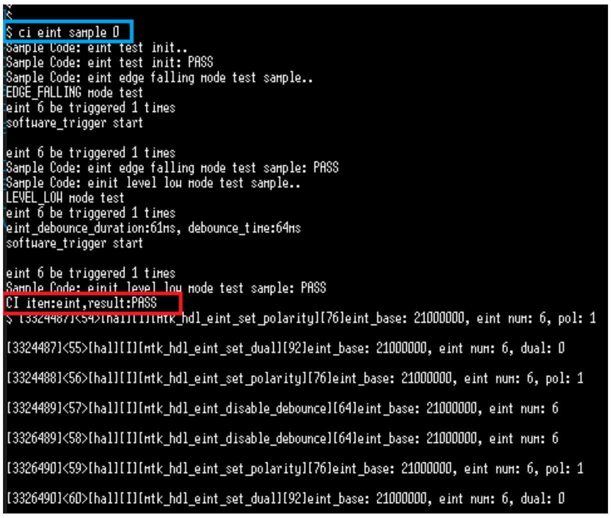


Figure 31 short GPIO 19&20(MT7931AN)



```
$
$ ci eint sample 0
Sample Code: eint test init..
Sample Code: eint test init: PASS
Sample Code: eint edge falling mode test sample..
EDGE_FALLING mode test
eint 6 be triggered 1 times
software_trigger start

eint 6 be triggered 1 times
Sample Code: eint edge falling mode test sample: PASS
Sample Code: einit level low mode test sample..
LEVEL_LOW mode test
eint 6 be triggered 1 times
eint_debounce_duration:61ms, debounce_time:64ms
software_trigger start

eint 6 be triggered 1 times
Sample Code: einit level low mode test sample: PASS
CI item:eint,result:PASS
$ [3324487]<54>[hal][I][mtk_hdl_eint_set_polarity][76]eint_base: 21000000, eint num: 6, pol: 1

[3324487]<55>[hal][I][mtk_hdl_eint_set_dual][92]eint_base: 21000000, eint num: 6, dual: 0

[3324488]<56>[hal][I][mtk_hdl_eint_set_polarity][76]eint_base: 21000000, eint num: 6, pol: 1

[3324489]<57>[hal][I][mtk_hdl_eint_disable_debounce][64]eint_base: 21000000, eint num: 6

[3326489]<58>[hal][I][mtk_hdl_eint_disable_debounce][64]eint_base: 21000000, eint num: 6

[3326490]<59>[hal][I][mtk_hdl_eint_set_polarity][76]eint_base: 21000000, eint num: 6, pol: 1

[3326490]<60>[hal][I][mtk_hdl_eint_set_dual][92]eint_base: 21000000, eint num: 6, dual: 0
```

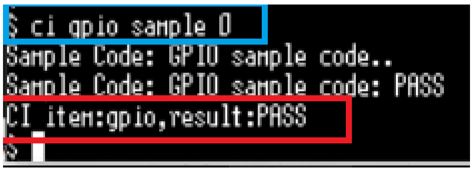
Figure 32 example of EINT

3.17 GPIO

The main function of this module is to provide a test of general-purpose input output.

Need PinMux configuration	Yes (GPIO36 for MT7933CT) (GPIO22 for MT7931AN)
Need rework board	No

Source code	ci_gpio.c
Support RFB type	MT7933CT & MT7931AN



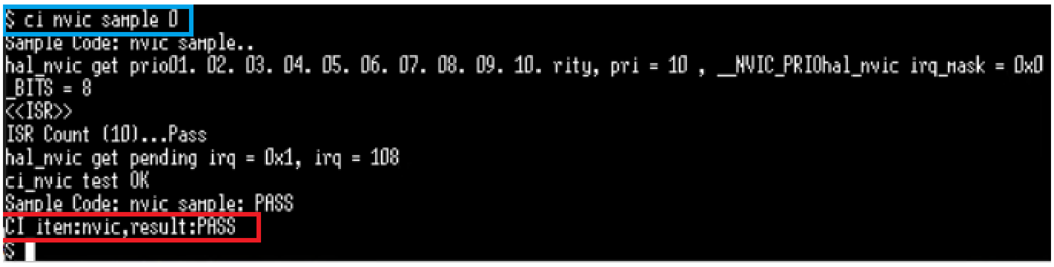
```
$ ci gpio sample 0
Sample Code: GPIO sample code..
Sample Code: GPIO sample code: PASS
CI item:gpio,result:PASS
$ 
```

Figure 33 example of GPIO

3.18 NVIC

The main function of this module is to provide a test of nested vectored Interrupt controller.

Need PinMux configuration	No
Need rework board	No
Source code	ci_nvic.c
Support RFB type	MT7933CT & MT7931AN



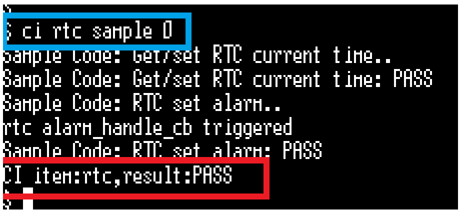
```
$ ci nvic sample 0
Sample Code: nvic sample..
hal_nvic get prio01. 02. 03. 04. 05. 06. 07. 08. 09. 10. rity, pri = 10 , __NVIC_PRIOhal_nvic irq_mask = 0x0
_BITS = 8
<<ISR>>
ISR Count (10)...Pass
hal_nvic get pending irq = 0x1, irq = 108
ci_nvic test OK
Sample Code: nvic sample: PASS
CI item:nvic,result:PASS
$ 
```

Figure 34 example of NVIC

3.19 RTC

The main function of this module is to provide a test of real-time clock.

Need PinMux configuration	No
Need rework board	No
Source code	ci_rtc.c
Support RFB type	MT7933CT & MT7931AN



```
$
$ ci rtc sample 0
Sample Code: Get/set RTC current time..
Sample Code: Get/set RTC current time: PASS
Sample Code: RTC set alarm..
rtc alarm_handle_cb triggered
Sample Code: RTC set alarm: PASS
CI item:rtc,result:PASS
$
```

Figure 35 example of RTC

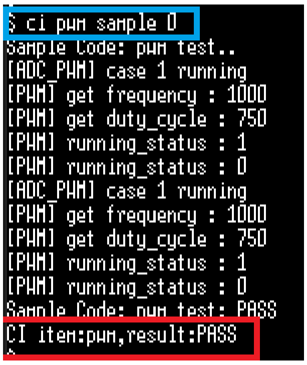
3.20 PWM

The main function of this module is to provide a test of pulse-width modulation.

Need PinMux configuration	Yes (GPIO38)
Need rework board	No
Source code	ci_pwm.c
Support RFB type	MT7933CT

Note:

- Use the LA to check the duty cycle and frequency (Refer to Figure 37,Figure 38 and Figure 39)



```
$ ci pwm sample 0
Sample Code: pwm test..
[ADC_PWM] case 1 running
[PWM] get frequency : 1000
[PWM] get duty_cycle : 750
[PWM] running_status : 1
[PWM] running_status : 0
[ADC_PWM] case 1 running
[PWM] get frequency : 1000
[PWM] get duty_cycle : 750
[PWM] running_status : 1
[PWM] running_status : 0
Sample Code: pwm test: PASS
CI item:pwm,result:PASS
$
```

Figure 36 example of PWM

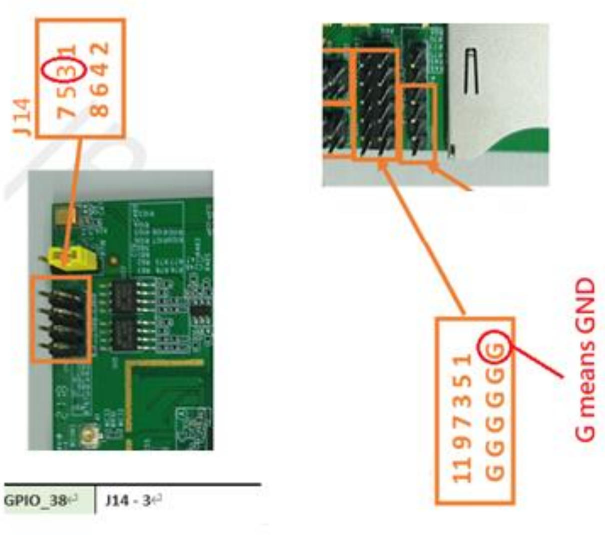


Figure 37 Location for LA to measure (MT7933CT)

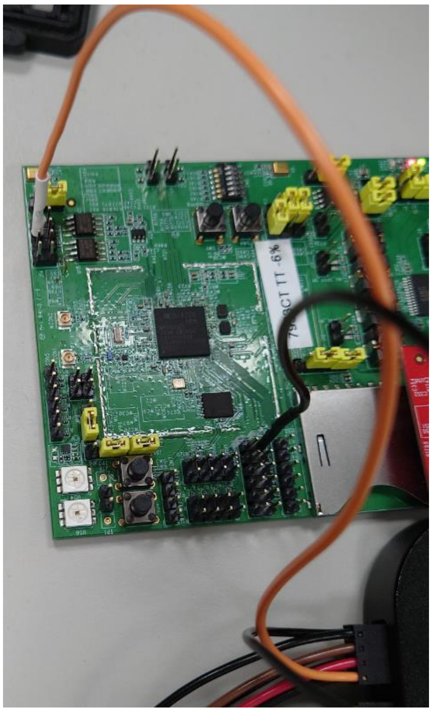
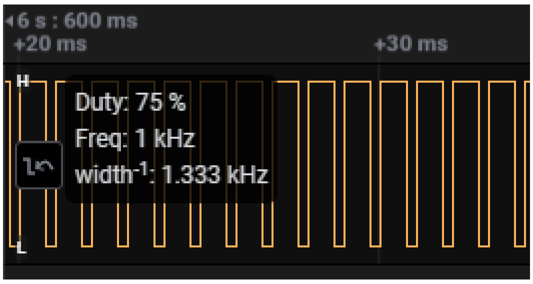


Figure 38 LA for checking (MT7933CT)



6 s : 600 ms
+20 ms
+30 ms

Duty: 75 %
Freq: 1 kHz
width^{-1}: 1.333 kHz

Figure 39 LA's result

3.21 SDIO Master & SDIO Slave (Secure Digital Input and Output Master & Slave)

The main function of this module is to provide a test between SDIO master and SDIO slave.

Need PinMux configuration	Yes (GPIO6 to GPIO11 & GPIO16)
Need rework board	Yes
Source code	ci_sdiom.c and ci_sdios.c
Support RFB type	MT7933CT

Note:

- Please refer to the ci_sdiom.c and ci_sdios.c for code level flow (Please also refer to Figure 40)
- For SDIO testing, the hardware rework is needed and it's very complex (Please contact with MTK hardware FAE for the detail)
- SDIO has some limitations as below (please refer to Figure 41)

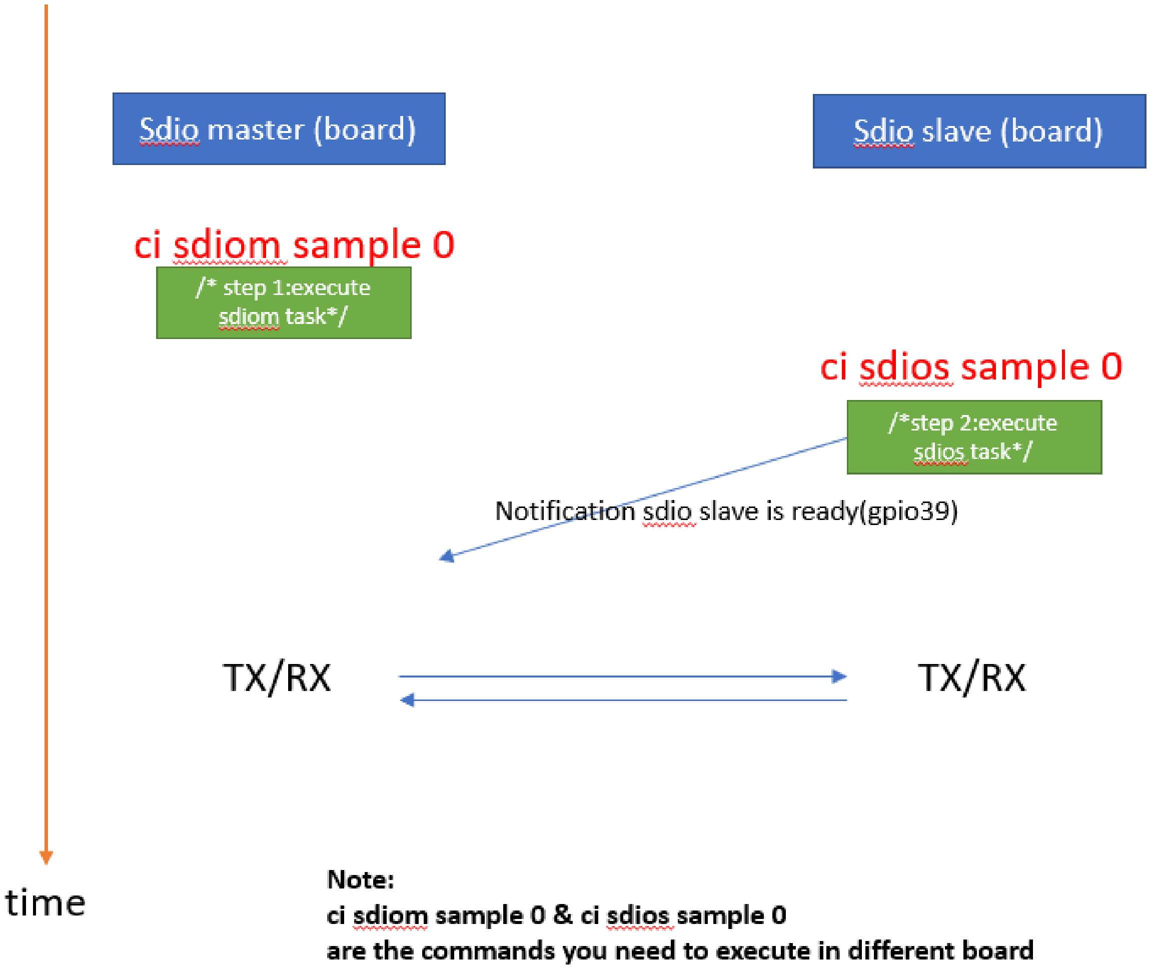
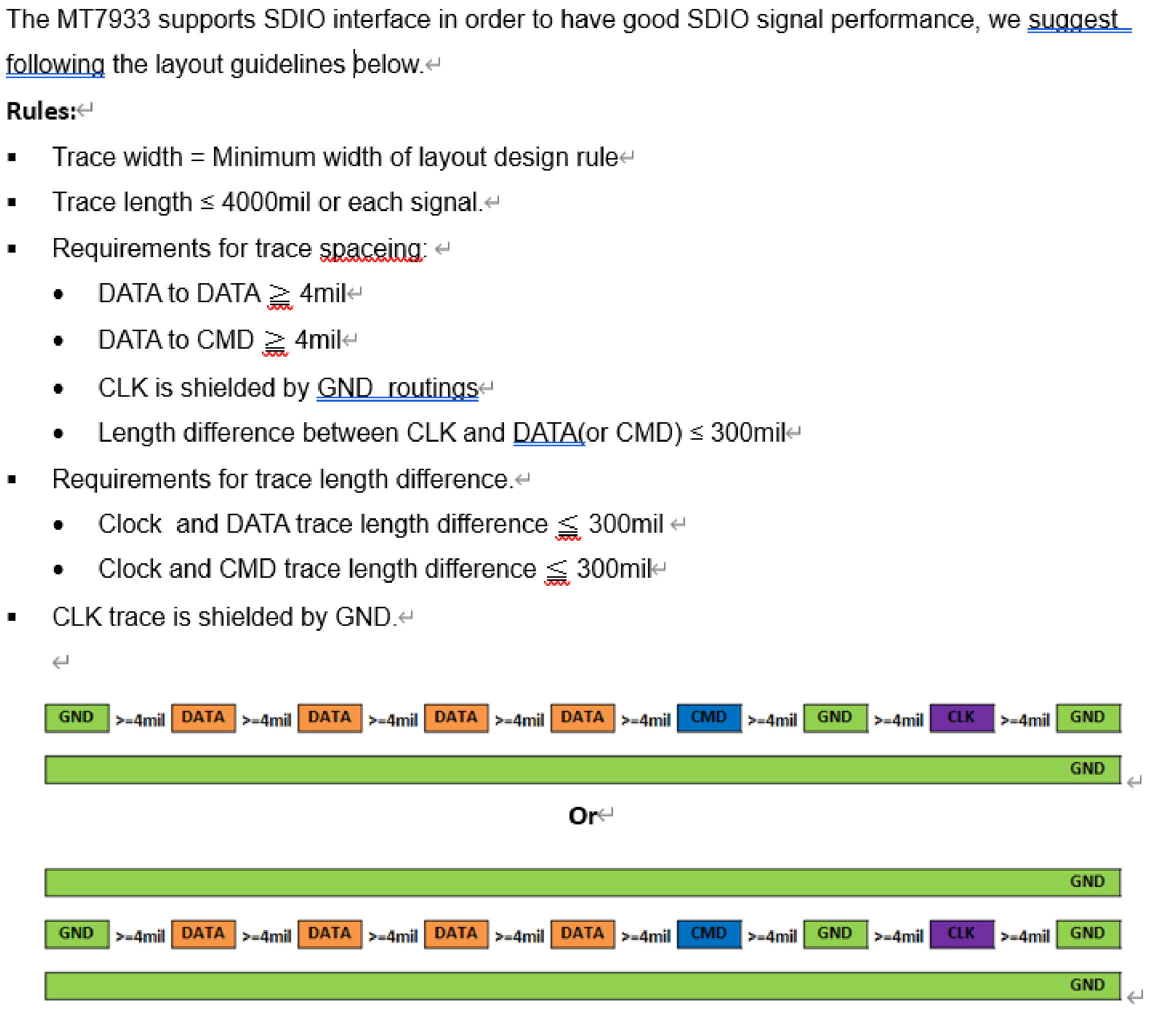

Sdio master (board)
Sdio slave (board)
ci sdiom sample 0
/* step 1:execute sdiom task*/
ci sdios sample 0
/*step 2:execute sdios task*/
Notification sdio slave is ready(gpio39)
TX/RX
TX/RX
time
Note:
ci sdiom sample 0 & ci sdios sample 0
are the commands you need to execute in different board

Figure 40 SDIO testing flow



The MT7933 supports SDIO interface in order to have good SDIO signal performance, we suggest following the layout guidelines below.

Rules:

- Trace width = Minimum width of layout design rule
- Trace length ≤ 4000mil or each signal.
- Requirements for trace spaceing:
  - DATA to DATA ≧ 4mil
  - DATA to CMD ≧ 4mil
  - CLK is shielded by GND routings
  - Length difference between CLK and DATA(or CMD) ≤ 300mil
- Requirements for trace length difference.
  - Clock and DATA trace length difference ≦ 300mil
  - Clock and CMD trace length difference ≦ 300mil
- CLK trace is shielded by GND.

Figure 41 SDIO's limitation

3.22 SPI Master & SPI Slave (Serial Peripheral Interface Master & Slave)

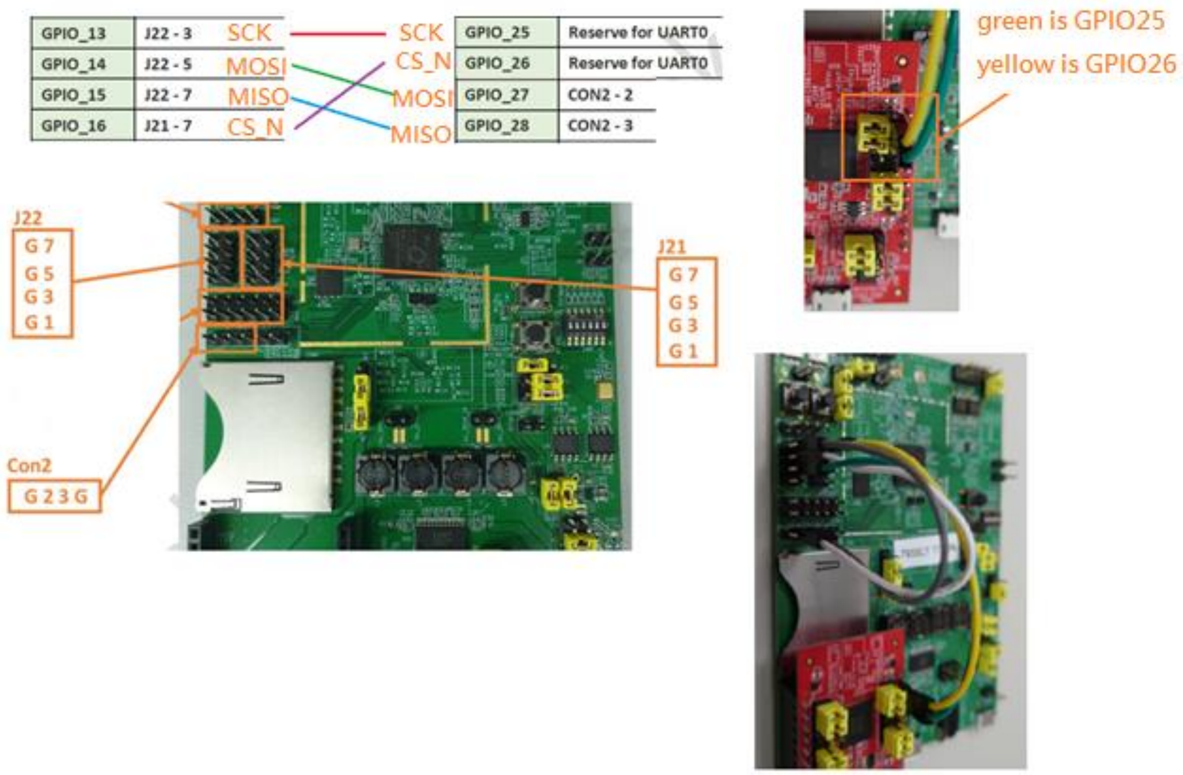
The main function of this module is to provide a test between spi master and spi slave.

Need PinMux configuration	Yes (GPIO13~16 for SPI Master. GPIO25~28 for SPI Slave)
Need rework board	No
Source code	ci_spi.c
Support RFB type	MT7933CT

Note:

- MT7931AN hasn't SPI Slave function, so it can't execute this testing
- Need to short SPI Master and SPI Slave for testing (please refer to Figure 42)

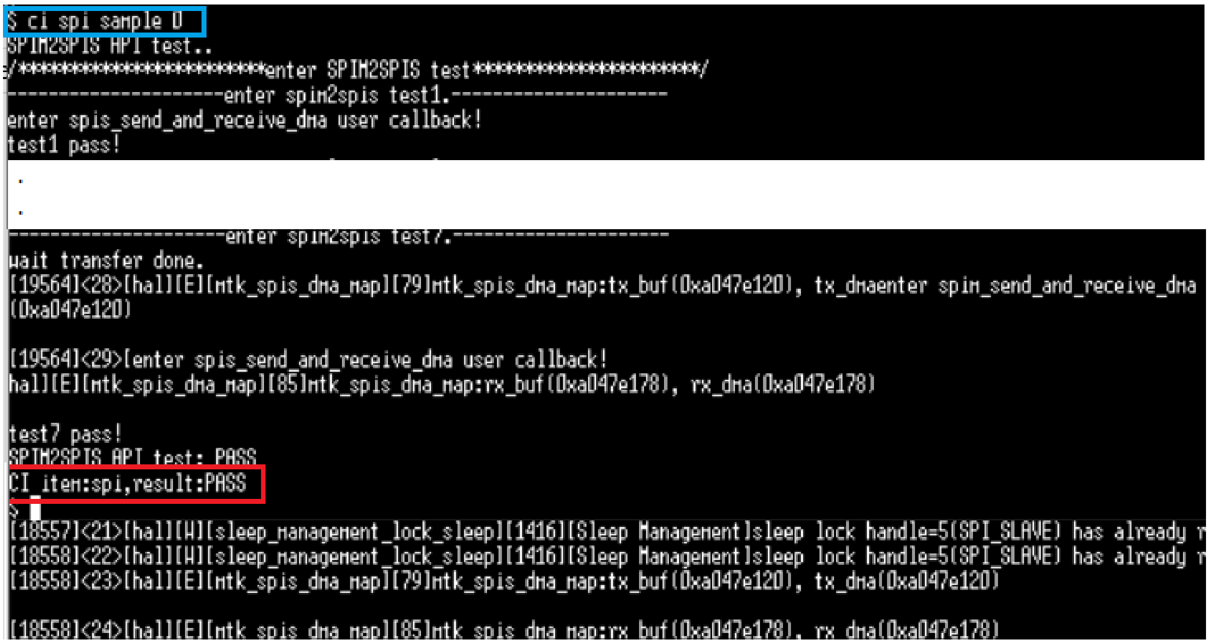
- GPIO 27(CON2-2) is strap pin, so the DuPont wire (CON2-2) need to be removed during boot up time or the system can’t be power on (That means DuPont wire need to be insert to CON2-2 after booting up)



GPIO_13	J22 - 3	SCK
GPIO_14	J22 - 5	MOSI
GPIO_15	J22 - 7	MISO
GPIO_16	J21 - 7	CS_N

SCK	GPIO_25	Reserve for UART0
CS_N	GPIO_26	Reserve for UART0
MOSI	GPIO_27	CON2 - 2
MISO	GPIO_28	CON2 - 3

Figure 42 short SPI Master & SPI Slave for testing



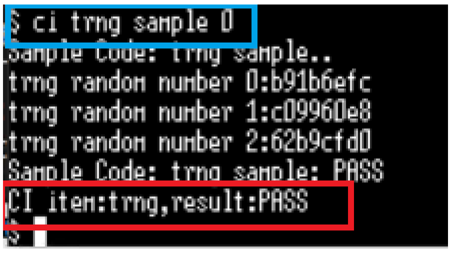
```
$ ci spi sample 0
SPIM2SPIS API test..
/***********************enter SPIM2SPIS test**********************/
---------------------enter spim2spis test1.---------------------
enter spis_send_and_receive_dma user callback!
test1 pass!
 .
 .
---------------------enter spim2spis test7.---------------------
wait transfer done.
[19564]<28>[hal][E][mtk_spis_dma_map][79]mtk_spis_dma_map:tx_buf(0xa047e120), tx_dmaenter spim_send_and_receive_dma
(0xa047e120)

[19564]<29>[enter spis_send_and_receive_dma user callback!
hal][E][mtk_spis_dma_map][85]mtk_spis_dma_map:rx_buf(0xa047e178), rx_dma(0xa047e178)

test7 pass!
SPIM2SPIS API test: PASS
CI_item:spi,result:PASS
$ █
[18557]<21>[hal][W][sleep_management_lock_sleep][1416][Sleep Management]sleep lock handle=5(SPI_SLAVE) has already r
[18558]<22>[hal][W][sleep_management_lock_sleep][1416][Sleep Management]sleep lock handle=5(SPI_SLAVE) has already r
[18558]<23>[hal][E][mtk_spis_dma_map][79]mtk_spis_dma_map:tx_buf(0xa047e120), tx_dma(0xa047e120)

[18558]<24>[hal][E][mtk_spis_dma_map][85]mtk_spis_dma_map:rx_buf(0xa047e178), rx_dma(0xa047e178)
```

Figure 43 example of SPI

3.23 TRNG (True Random Number Generator)

The main function of this module is to provide a test of TRNG.

Need PinMux configuration	No
Need rework board	No
Source code	ci_trng.c
Support RFB type	MT7933CT & MT7931AN



```
$ ci trng sample 0
Sample Code: trng sample..
trng random number 0:b91b6efc
trng random number 1:c09960e8
trng random number 2:62b9cfd0
Sample Code: trng sample: PASS
CI item:trng,result:PASS
$
```

Figure 44 example of TRNG

3.24 I2S (Inter-IC Sound)

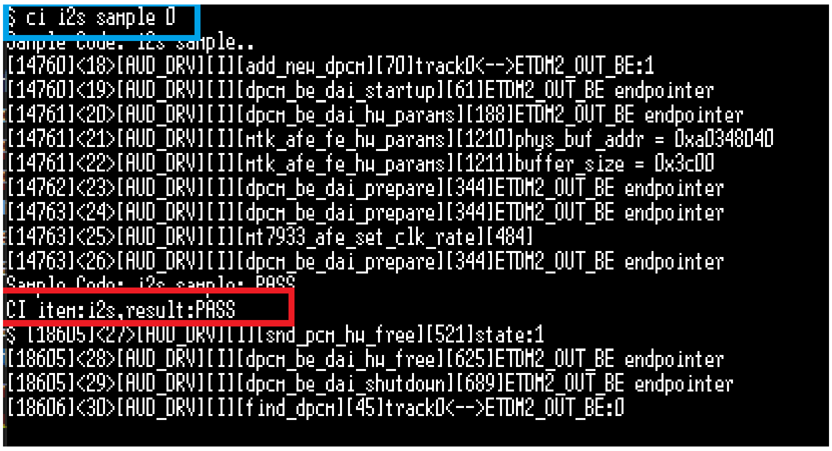
The main function of this module is to provide a test of I2S.

Need PinMux configuration	Yes (GPIO11 & GPIO13~GPIO15)
Need rework board	No
Source code	ci_i2s.c
Support RFB type	MT7933CT

Note:

Please contact with MTK hardware/software FAE for the detail, if you don’t know how to test it

- Connect to external DAC device, check if the voice is output
- Connect to oscilloscope, check if the output signal of GPIO pin is expected



```
$ ci i2s sample 0
Sample Code: i2s sample..
[14760]<18>[AUD_DRV][I][add_new_dpcm][70]track0<-->ETDM2_OUT_BE:1
[14760]<19>[AUD_DRV][I][dpcm_be_dai_startup][61]ETDM2_OUT_BE endpointer
[14761]<20>[AUD_DRV][I][dpcm_be_dai_hw_params][188]ETDM2_OUT_BE endpointer
[14761]<21>[AUD_DRV][I][mtk_afe_fe_hw_params][1210]phys_buf_addr = 0xa0348040
[14761]<22>[AUD_DRV][I][mtk_afe_fe_hw_params][1211]buffer_size = 0x3c00
[14762]<23>[AUD_DRV][I][dpcm_be_dai_prepare][344]ETDM2_OUT_BE endpointer
[14763]<24>[AUD_DRV][I][dpcm_be_dai_prepare][344]ETDM2_OUT_BE endpointer
[14763]<25>[AUD_DRV][I][mt7933_afe_set_clk_rate][484]
[14763]<26>[AUD_DRV][I][dpcm_be_dai_prepare][344]ETDM2_OUT_BE endpointer
Sample Code: i2s sample: PASS
CI item:i2s,result:PASS
$ [18605]<27>[AUD_DRV][I][snd_pcm_hw_free][521]state:1
[18605]<28>[AUD_DRV][I][dpcm_be_dai_hw_free][625]ETDM2_OUT_BE endpointer
[18605]<29>[AUD_DRV][I][dpcm_be_dai_shutdown][689]ETDM2_OUT_BE endpointer
[18606]<30>[AUD_DRV][I][find_dpcm][45]track0<-->ETDM2_OUT_BE:0
```

Figure 45 example of I2S

Exhibit 1 Terms and Conditions

Your access to and use of this document and the information contained herein (collectively this “Document”) is subject to your (including the corporation or other legal entity you represent, collectively “You”) acceptance of the terms and conditions set forth below (“T&C”). By using, accessing or downloading this Document, You are accepting the T&C and agree to be bound by the T&C. If You don’t agree to the T&C, You may not use this Document and shall immediately destroy any copy thereof.

This Document contains information that is confidential and proprietary to MediaTek Inc. and/or its affiliates (collectively “MediaTek”) or its licensors and is provided solely for Your internal use with MediaTek’s chipset(s) described in this Document and shall not be used for any other purposes (including but not limited to identifying or providing evidence to support any potential patent infringement claim against MediaTek or any of MediaTek’s suppliers and/or direct or indirect customers). Unauthorized use or disclosure of the information contained herein is prohibited. You agree to indemnify MediaTek for any loss or damages suffered by MediaTek for Your unauthorized use or disclosure of this Document, in whole or in part.

MediaTek and its licensors retain titles and all ownership rights in and to this Document and no license (express or implied, by estoppels or otherwise) to any intellectual propriety rights is granted hereunder. This Document is subject to change without further notification. MediaTek does not assume any responsibility arising out of or in connection with any use of, or reliance on, this Document, and specifically disclaims any and all liability, including, without limitation, consequential or incidental damages.

THIS DOCUMENT AND ANY OTHER MATERIALS OR TECHNICAL SUPPORT PROVIDED BY MEDIATEK IN CONNECTION WITH THIS DOCUMENT, IF ANY, ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. MEDIATEK SPECIFICALLY DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, COMPLETENESS OR ACCURACY AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MEDIATEK SHALL NOT BE RESPONSIBLE FOR ANY MEDIATEK DELIVERABLES MADE TO MEET YOUR SPECIFICATIONS OR TO CONFORM TO A PARTICULAR STANDARD OR OPEN FORUM.

Without limiting the generality of the foregoing, MediaTek makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does MediaTek assume any liability arising out of the application or use of any product, circuit or software. You agree that You are solely responsible for the designing, validating and testing Your product incorporating MediaTek’s product and ensure such product meets applicable standards and any safety, security or other requirements.

The above T&C and all acts in connection with the T&C or this Document shall be governed, construed and interpreted in accordance with the laws of Taiwan, without giving effect to the principles of conflicts of law.